

HOPEX Data Governance

HOPEX V5



Information in this document is subject to change and does not represent a commitment on the part of MEGA International.

No part of this document is to be reproduced, transmitted, stored in a retrieval system, or translated into any language in any form by any means, without the prior written permission of MEGA International.

© MEGA International, Paris, 1996 - 2021

All rights reserved.

HOPEX IT Business Management , HOPEX IT Portfolio Management,, HOPEX Business Architecture & Strategic Planning and HOPEX are registered trademarks of MEGA International.

Windows is a registered trademark of Microsoft Corporation.

The other trademarks mentioned in this document belong to their respective owners.

CONTENTS



Contents	1

Introduction to HOPEX Data Governance	21

The Scope Covered by the HOPEX Data Governance Solution	22

Data Discovery	22

Business Glossary	22

Data Architecture	22

<i>Three modeling levels</i>	22

<i>Data category</i>	23

<i>Conception workflow</i>	23

Use of Data	23

Data Quality and Compliance	24

<i>Definition of responsibilities</i>	24

<i>Definition of rules and standards</i>	24

<i>Traceability of Data (Data Lineage)</i>	24

<i>Evaluation of data</i>	24

<i>Analysis reports</i>	25

The HOPEX Data Governance desktop	26

Connecting to HOPEX Data Governance	26

Displaying the Working Environment of an Enterprise	26

<i>Creating an enterprise and its working environment</i>	26

Profiles of HOPEX Data Governance	28
Business Roles of HOPEX Data Governance	29

DATA CATALOGS

Description of a Data Catalog	1
Data Catalog Content	2
Deployed Data Store	2
<i>Meta dataset</i>	2
<i>Meta field</i>	2
Regulations Associated with the Catalog	3
Data Lineage of a Data Catalog	4
Concept Domain Map of a Data Catalog	4
Creating a Data Catalog	5
Data Catalog Workflow	5
Associating Keywords with Metadata	6
Defining Metadata Categories	7
Report and Dashboard of a Data Catalog	8
Dashboard of the Data Catalog	8
Data Catalog Report	8
Connecting Metadata to Business Information	9
Collecting Business Data	10
Monitoring of Data Call Forms	11
Catalog Data Quality	12
Connecting a Data Quality Policy to the Data Catalog	12
Connecting a Data Quality Policy to Physical Metadata	12
Snapshot of a Data Catalog	13

Data Discovery	15
The HOPEX Data Discovery Module	16
Deployment	16
Connection Options to Data Sources	16
Running HOPEX Data Discovery	17
Access conditions	17
Online mode	17
Stand-alone mode	21
Incremental import	21
Options de connexion aux sources de données	22

Data Lineages	23
Functional Data Lineage	24
Creating a Functional Data Lineage and its Diagram	24
<i>Initialisation of the diagram</i>	24
<i>Data lineage workflow</i>	25
Objects of the Data Lineage Diagram	27
<i>Participant</i>	27
<i>Steps</i>	28
<i>Processing software</i>	28
<i>Technology</i>	29
<i>Règle de calcul</i>	29
<i>Transitions</i>	29
Example of Lineage of a Business Data	29
Technical Data Lineage	30
Overview of the MANTA Import	30
Intégration de HOPEX Data Governance avec MANTA	31
Importing a Technical Data Lineage	31
Viewing the Data Technical Lineage	31
<i>Prerequisites</i>	31
<i>Accessing Technical Data Lineage</i>	32
Mapping Matrices	32

BUSINESS GLOSSARY

Introduction to the Creation of a Business Ontology	35
Vocabulary Management Process	35
<i>Analysis and organization of business concepts</i>	36
<i>Concept realization</i>	36
Consulting the Business Glossary	37
Searching for Terms in the Business Glossary	38
Scope of the Search	38
<i>Prerequisite</i>	38
Starting the Search	38
<i>Search filters</i>	39
Displaying the Details of a Term	40
<i>Standard characteristics</i>	41
<i>Advanced characteristics</i>	41

Generating a Glossary	42
Launching a Glossary Report	42
Using the Glossary in a Multilingual Context	42
Using Data ID Card	43
Defining Business Information	45
Objects Used	46
Concept and Term	46
Links Between Concepts	46
<i>Definition links</i>	47
<i>Dependency links</i>	47
Concept Properties	48
Concept Instances: Individuals	49
The Life Cycle of a Concept or Individual	50
<i>Concept life cycle</i>	50
<i>Individual life cycle</i>	51
Periods	52
Classifying Concepts and the Concept Type Notion	53
The Concept View	54
Dictionary Element Realization	54
Presentation of Concept Modeling Diagrams	55
Concept Diagram	55
Concept structure diagram	56
Concept type structure diagram	56
State concept state structure diagram	56
Individual structure diagram	56
The concept life cycle structure diagram	57
Business Dictionary	58
The Elements of a Business Dictionary	58
<i>Accessing the elements of a business dictionary</i>	59
<i>Importing business information</i>	59
Work Business Dictionary	59
Creating a Business Dictionary	60
Initializing a Business Dictionary Using Logical or Physical Data	60
<i>Initializing a Business Dictionary Using Logical Data</i>	60
<i>Initializing a Business Dictionary Using Physical Data</i>	61
<i>Initializing a Business Dictionary from Meta Datasets</i>	61
<i>Initializing a Business Dictionary when Creating Logical or Physical Data</i>	62
<i>Displaying the Realization Chart</i>	62
Concept Domain Map	65
Creating a Concept Domain Map	65
The Components of a Concept Domain Map	65
Example of a Concept Domain Map	66
Reports Available on a Concept Domain Map	67
Concept Domain	68
Creating a Concept Domain	68
Creating the Structure Diagram for a Concept Domain	68
Building a Concept Diagram	68

<i>Creating a concept diagram of a concept domain</i>	69
<i>The components of a concept diagram</i>	69
<i>Activating the views window</i>	69
<i>Adding a concept diagram element</i>	70
<i>Using the object insert toolbar</i>	70
<i>Overview of links between objects</i>	71
<i>Accessing link properties in a concept diagram</i>	72
Defining the Components of a Concept Domain	74
<i>The components of a concept domain</i>	74
<i>Defining the CRUD for the components of a concept domain</i>	75
Concept	77
Accessing the List of Concepts	77
Creating Concepts	77
Concepts and Terms	78
<i>Connecting an existing concept to a term</i>	78
<i>Creating terms in multiple languages from a concept</i>	78
<i>Creating synonyms in multiple languages</i>	78
Concept Properties	79
<i>Characteristics</i>	79
<i>Components</i>	79
<i>Super types</i>	79
<i>Realizations</i>	80
<i>Regulations</i>	80
<i>Data Quality</i>	80
<i>Reporting</i>	80
<i>Workflows</i>	81
Concept Components	82
Accessing Concept Components	82
Creating a Concept Component from a Diagram	82
Describing Concept Power Components	83
Describing a Computed Concept Component	84
Concept Properties	85
Creating a Concept Property	85
<i>Creating a Concept Property</i>	85
<i>Connecting a Concept Property to a concept</i>	86
<i>Connecting two Concept Properties</i>	86
Creating a Computed Concept Property	86
Concept Inheritances	87
Accessing Concept Inheritances	87
Creating a Concept Inheritance from a Concept Diagram	87
Defining Inheritance of a Concept Component	88
Creating a Concept Component Substitution	88
Concept structure diagram	89
Individuals	91
Accessing the List of Individuals	91
Creating an Individual from a Business Dictionary	91
Individual Properties	92
Creating an Individual Classification	92
Creating a Dictionary Entity Component	93
Individual Structure Diagram	93

Concept or Individual States	94
Describing State Concepts	94
<i>Accessing the state concepts list</i>	95
<i>Creating a state concept from a business dictionary</i>	95
<i>State concept properties</i>	96
Describing Event Concepts	97
<i>Accessing the event concept list</i>	97
<i>Creating an event concept from a business dictionary</i>	97
<i>Event concept properties</i>	98
<i>Connecting an event concept to its concept</i>	98
State Concept Structure Diagram	99
Describing Individual States and Events	100
<i>Accessing the individual state and event list</i>	101
<i>Creating an Individual state from a concept domain</i>	101
<i>Individual state properties</i>	102
<i>Creating an Individual event from a concept domain</i>	102
<i>Connecting an individual event to an individual</i>	102
Concept life cycle structure diagram	102
<i>Creating a concept life cycle</i>	103
<i>Creating a concept life cycle structure diagram</i>	104
<i>Adding a concept life cycle event</i>	104
<i>Creating a concept life cycle transition</i>	104
Using periods	105
Concept Type	106
Accessing the Concept Types List	106
Creating a New Concept Type	106
Concept Type Properties	106
Describing Concept Type Components	107
<i>Accessing concept type components</i>	108
<i>Creating a concept type component from a concept domain</i>	108
Describing Concept Type Variations	109
<i>Accessing concept type variations</i>	109
<i>Creating a concept type variation from a concept domain</i>	109
The Concept Type Structure Diagram	109
Concept View	111
Creating a Concept View	111
Defining the Concept View Content	112
<i>Displaying objects in the view</i>	112
<i>Adding a source object to the concept view</i>	112
<i>Adding a component to the concept view</i>	113
The View Report	113
Calculation Rule on Concepts	115
Creating a Calculation Rule on a Business Object	116
Calculation Rule on a Concept Component	116
Calculation Rule on a Concept Property	116
<i>Example</i>	117

Connecting the Business Concepts to the Logical and Physical architecture. 119

Realization of Concept	120
Defining the Object that Realizes a Concept	120
Defining the Concept Realized by a Class	121
Using Realization Matrices	122
Realization Levels	122
Creating a Realization Matrix	122

DATA ARCHITECTURE AND TOOLS

Modeling Data dictionaries. 127

Logical Data Modeling Options	128
Formalisms	128
Notations	128
Overview of Logical Data	129
Data Dictionary	129
Data Domain Map	129
Logical Data Domain	129
Logical Data View	129
Data Model	130
<i>Example</i>	130
Data Dictionary	131
Elements of a Data Dictionary	131
<i>Accessing the elements of a data dictionary</i>	132
<i>Importing logical data</i>	132
Data Domain Map	133
Creating a Data Domain Map	133
Components of a Data Domain Map	133
Logical and Application Data Domains	134
Creating a Data Domain	134
The Data Domain Diagram	134
<i>Example of diagram</i>	135
<i>Creating a Logical Data Domain Diagram</i>	135
<i>Adding an object to the diagram</i>	135
Adding a Component to a Data Domain	136
<i>Defining the access mode to the component (CRUD)</i>	136
Logical Data View	137
Creating a logical data view	137
<i>Creating a data view (from a list of views)</i>	138
<i>Creating a data view directly from an object</i>	138
Displaying source objects in the data view	139
Defining the Data View Components	140
<i>Embedded component</i>	140

<i>Referenced component</i>	140
<i>Using a view in another view</i>	141
Class Diagram	142
Creating a Package	142
Creating a Class Diagram	143
Datatypes	144
Data type packages	144
Creating a New Datatype Package	145
<i>Creating a datatype</i>	145
Referencing Datatype Packages	146
Assigning Types to Attributes	146
Data Model	147
Summary of Concepts	147
<i>Data model</i>	147
<i>Data diagram</i>	147
Building a data model	148
<i>Creating a Data Model</i>	148
<i>Creating a Data Diagram</i>	148
Entities	149
<i>Creating an entity</i>	149
Attributes	150
<i>Creating attributes</i>	151
<i>Inherited attributes</i>	151
Associations	151
<i>Creating an Association</i>	153
<i>Defining association roles (ends)</i>	154
<i>Multiplicities</i>	155
<i>Aggregation</i>	157
<i>Composition</i>	157
Reflexive Associations	158
"N-ary" Association	159
Constraints	160
Normalization Rules	161
<i>First Normal Form</i>	161
<i>Second Normal Form</i>	161
<i>Third Normal Form</i>	162
Generalizations	163
<i>What is a generalization?</i>	163
<i>Multiple sub-entities</i>	165
<i>Multiple inheritance</i>	166
<i>Creating a generalization</i>	167
<i>Discriminator</i>	167
Entity Identifier	168
<i>Identification by an attribute</i>	168
Data Model Mapping	169
<i>Functional Objectives</i>	169
<i>Running the mapping editor</i>	170
<i>Creating a mapping</i>	170
<i>Deleting a mapping</i>	170
<i>Mapping details</i>	170
<i>Example of mapping between data models</i>	172

IDEF1X Notation	173
Prerequisite	173
About Data Modeling with IDEF1X	173
Concept Synthesis for Data Modeling (IDEF1X)	174
Creating a Data Model (IDEF1X)	174
<i>Data Diagram (IDEF1X)</i>	174
Entities (IDEF1X)	174
<i>Creating an entity</i>	175
<i>Attributes</i>	175
Associations (IDEF1X)	176
<i>Mandatory identifying relationship</i>	178
<i>Mandatory non-identifying relationship</i>	179
<i>Mandatory Non-Identifying Relationship</i>	179
<i>non-specific relationship</i>	180
<i>Associative entity</i>	181
<i>Defining Association Roles</i>	182
<i>Multiplicities</i>	183
Categorization Relationships (Generalizations) - (IDEF1X)	184
<i>What is a Categorization (Generalization)?</i>	184
<i>Creating a Categorization</i>	184
<i>Multiple Categories</i>	185
<i>Multiple Category Clusters</i>	186
<i>Complete Categorization</i>	186
<i>Discriminator</i>	187
I.E. Notation	188
Prerequisite	188
About Data Modeling with I.E.	188
Concept Synthesis for Data Modeling (I.E)	189
Creating a Data Model (I.E)	189
<i>Data Diagram (I.E.)</i>	190
Entities (I.E.)	190
<i>Creating an entity</i>	190
<i>Attributes</i>	191
Associations (I.E)	191
<i>Overview</i>	192
<i>Associations and their Multiplicities</i>	192
Sub-types (I.E)	194
<i>What is sub-type?</i>	194
<i>Multiple Subtypes</i>	196
<i>Advantages of sub-types</i>	196
<i>Multiple inheritance</i>	197
<i>Creating a sub-type</i>	197
The Merise Notation	199
Prerequisite	199
About Data Modeling (Merise)	199
Concept Synthesis for Data Modeling (Merise)	199
Creating a Data Model (Merise)	200
<i>Data Diagram (Merise)</i>	200
The entities (Merise)	200
<i>Creating an entity</i>	201
The associations (Merise)	201
<i>Examples of associations</i>	202

<i>Reflexive relationships</i>	203
<i>"n-ary" relationships</i>	203
<i>Participations or cardinalities</i>	204
<i>Creating an Association (Relationship)</i>	205
Attributes (Information) - Merise	206
<i>Properties</i>	206
<i>Identifier</i>	207
<i>Creating Attributes</i>	207
Normalization Rules (Merise)	208
<i>First Normal Form</i>	208
<i>Second Normal Form</i>	209
<i>Third Normal Form</i>	209
Refining Data Model Specification (Merise)	210
<i>Ordering Attributes</i>	210
<i>Attribute Description</i>	210
<i>Participations or cardinalities</i>	211
Sub-typing (Merise)	213
<i>What is sub-type?</i>	213

Modeling Databases 217

Logical Formalism and Synchronization	218
Database 219	
Creating Databases	219
Database Properties	219
Associating a Package with a Database	220
Importing a DBMS Version	220
Physical Data Maps and Relational Schemas 221	
Physical Data Maps	221
<i>Creating a physical data map</i>	221
<i>Components of a physical data map</i>	221
Relational Schema	222
<i>Creating a Relational Schema</i>	222
<i>Relational Schema Diagram</i>	222
Relational Diagram 224	
Creating the Relational Diagram	224
<i>Creating objects in the diagram</i>	225
<i>Configuring display of relational diagrams</i>	225
Database Components 227	
Database Tables	227
<i>Creating a table</i>	227
<i>Deleting a table</i>	227
Table Columns	228
<i>Viewing columns</i>	228
<i>Creating a column</i>	228
<i>Deleting a column</i>	229
Modifying Keys and Indexes	229
Creating a Key	230
<i>Primary key</i>	230

Foreign key	230
Creating an Index	231
Adding a Column to a Key or Index	232
Primary and foreign keys	233
Specifying Primary Keys	233
Specifying Foreign Keys	234
Column Primary Key of Two Tables	235
Column Primary Key of Three Tables	235
Data Types and Column Datatypes	236
Attribute Datatypes	236
Determining Column Datatypes from Attribute Types	236
Pivot Types	236
Connecting a Datatype to a Pivot Type	238
Connecting a Datatype to a Pivot Type in UML Notation	238
Mappings Between Pivot Types and Datatypes	241
Example of correspondence between pivot types and Oracle 8 datatypes	242
Creating New Datatypes	244
Creating Datatypes - Example for Oracle 10	244
Creating Datatypes - Example for SQL Server 7	247
Database Modeling Rules	249
Data Categorization	251
Defining Data Categories	252
Importing the Solution Pack of Categories	252
Accessing Data Categories	252
Creating a Data Category	252
Indicating the Category of a Data Item	253
Defining a Category on a Business Data	253
Associating a Data Category with a Regulation	253
Data Responsibility.	255
Use of Data Responsibilities	256
Roles Applicable to Data	256
Accessing Notifications	257
Defining Responsibilities for Data	258
Viewing the Persons in Charge of an Object	258
Automatic Assignment of an Object	258
Explicit Assignment of an Object	258
Consulting Data Governance reports	259

Use of Data by the Information System.	261
Defining the Data Used by Applications	262
Creating the Inventory of Application Assets	262
Connecting Data to an Application	262
<i>Creating a data store on an application</i>	<i>262</i>
<i>Defining data access mode</i>	<i>263</i>
Connecting Data to an Application Flow Scenario	263
<i>Creating a data store on an application flow scenario</i>	<i>264</i>
Defining the Data Used by Processes	265
Creating the Inventory of Processes	265
<i>Process Types</i>	<i>265</i>
<i>Creating a Process</i>	<i>266</i>
Connecting Data to a Process	266
<i>Creating a data store</i>	<i>266</i>
<i>Defining data access mode</i>	<i>266</i>
Connecting Data to a Content	266
Synchronizing logical and physical models	269
Synchronization Display Options	270
"Logical to Physical" Synchronization Rules.	271
Logical > physical synchronization: the application and logical data areas	271
Logical to Physical Synchronization: the Entities (or Classes)	271
<i>General rule.</i>	<i>271</i>
<i>Sub-entity</i>	<i>272</i>
<i>Abstract entity</i>	<i>272</i>
<i>Realized entity</i>	<i>272</i>
Logical to Physical Synchronization: the Associations	272
<i>Constraint associations (multiplicities: 0,1 or 1,1)</i>	<i>272</i>
<i>Constraint associations (multiplicities: 0,1 and 0,1)</i>	<i>274</i>
<i>Deadlocks</i>	<i>274</i>
<i>Non-constraint association.</i>	<i>276</i>
<i>Association class</i>	<i>277</i>
Logical to Physical Synchronization: the Parts (UML)	277
<i>Example 1: None / *.</i>	<i>278</i>
<i>Example 2: Aggregation / *.</i>	<i>278</i>
<i>Example 3: Composition / 0..1.</i>	<i>278</i>
From the Logical Model to the Physical Model	279
Running Synchronization (Logical > Physical)	279
<i>Step 1: Selecting the source objects to be synchronized</i>	<i>279</i>
<i>Step 2: Synchronization options.</i>	<i>281</i>
<i>Step 3: Protecting objects</i>	<i>281</i>
<i>Step 4: Validating results</i>	<i>281</i>
Using Options (Logical > Physical)	282
<i>Take account of optimizations</i>	<i>282</i>
<i>Take account of deletions</i>	<i>282</i>
<i>Possible option combinations</i>	<i>283</i>

Protecting Objects	284
<i>Frozen mode</i>	284
<i>Realized mode</i>	284
Synchronization Results: Correspondences	285
<i>Mapping characteristics</i>	286
Reduced Synchronization (Logical to physical mode)	288
Reduced Synchronization Source Objects (Logical > Physical)	288
<i>Running from a data model</i>	288
<i>Running from a data model entity</i>	288
<i>Running on an entity outside context</i>	289
Reduced Synchronization Strategies (Logical > Physical)	289
<i>Impact of synchronized object on other objects</i>	289
<i>Impact of other objects on synchronized object</i>	290
<i>All impacts</i>	292
Running Reduced Synchronization (Logical > Physical)	292
<i>Reduced synchronization options</i>	293
Running Synchronization After Modifications	294
Synchronization after Modification of the Data Diagram	294
<i>Newly created entities, associations, and attributes in the data diagram</i>	294
<i>Entities, associations, or attributes deleted from the data diagram</i>	294
<i>Modified attribute characteristics</i>	294
<i>Modified name of an attribute, entity, or association</i>	295
<i>Modified maximum multiplicity of an association</i>	295
<i>Modified association links</i>	295
Synchronization after Modifications to the Physical Diagram	295
<i>Deleted table or column</i>	295
<i>Created objects</i>	296
<i>Modified characteristics of objects created by synchronization</i>	296
<i>Modified order</i>	296
From the Physical Model to the Logical Model	297
"Physical to Logical" Synchronization Rules	297
Running Synchronization (Physical > Logical)	300
<i>Step 1: Selecting objects to be synchronized</i>	300
<i>Step 2: Synchronization options</i>	300
<i>Step 3: Protecting objects</i>	301
<i>Step 4: Validating results</i>	301
<i>Reduced synchronization</i>	301
"Physical to Logical" Synchronization Results	301
<i>Owner data model</i>	301
<i>Data diagrams</i>	301
<i>Mappings</i>	302
Configuring Synchronization	303
Preparing Synchronization	303
Creation Options	303
<i>On a database</i>	303
<i>On the DBMS</i>	305
Configuring Name Generation	305
<i>Naming rules</i>	305
<i>Modifying a naming rule</i>	307
<i>Entering the SQL mask</i>	307
<i>Configuring PK column names (implicit identifier)</i>	309

Diagram Synchronization	312
Case of Diagram Update at Synchronization	312
<i>After source diagram modification</i>	312
<i>After target diagram modification</i>	312
<i>After modification of both diagrams</i>	313
<i>No modification detected</i>	313
<i>Particular case: an entity mapping with two tables</i>	313
Model Mapping	313
The Database Editor	314
Run the editor on a database	314
<i>Creating a Logical/Physical Mapping Tree</i>	314
Creating a Mapping	314
Deleting a mapping	316
Mapping Details	317
Mapping Properties	317
Mapping Report	318
Object status	319
<i>Saving display of editor indicators</i>	319
Mapping Source	319
Mapping Drawing	321
Denormalizing logical and physical models	323
Denormalization Principles	324
Denormalization: consistency of models	324
<i>Transferring mappings</i>	324
<i>Deleting source objects</i>	324
Synchronization and Denormalization	324
<i>Combining denormalization and synchronization options</i>	325
Denormalization: Use Case	325
Logical Denormalization	328
Running Logical Denormalization	328
<i>Logical denormalization example</i>	328
Logical Denormalization Wizards	330
<i>Transform association to entity</i>	330
<i>Transform entity to association</i>	330
<i>Transform generalization to association</i>	331
<i>Transform association to generalization</i>	332
<i>Vertical partition of an entity</i>	332
<i>Horizontal partition of an entity</i>	333
<i>Merging of entities</i>	334
<i>Merging of ascending entities</i>	334
<i>Merging of descending entities</i>	335
<i>Copy/paste of attributes</i>	335

Physical Denormalization	336
Running Physical Denormalization	336
<i>Physical denormalization example</i>	336
List of Physical Denormalization Wizards	337
<i>Vertical partition of a table</i>	337
<i>Horizontal partition of a table</i>	338
<i>Merging of tables</i>	339
<i>Transform foreign key to table</i>	340
<i>Transform table to foreign key</i>	340
<i>Copy/paste of columns</i>	341
 Generating SQL scripts	 343
 Running SQL Generation	 344
SQL Generation Objects	344
Start the generation wizard	344
Incremental Generation	346
Incremental Generation Objects	346
Running Incremental Generation	346
<i>Generation options</i>	346
<i>Start the generation wizard</i>	346
Configuring SQL generation	349
Configuring the DBMS Version	349
<i>Supported DBMS versions</i>	349
<i>Modifying DBMS version properties</i>	349
Configuring Database Generation	350
Prefixing Object Names	352
<i>Inheritance</i>	352
<i>DBMSs concerned</i>	352
Supported Syntax	353
CREATE TABLE Instruction	353
<i>Managing NOT NULL</i>	353
<i>PRIMARY KEY clause</i>	354
<i>FOREIGN KEY clause</i>	354
<i>UNIQUE clause</i>	356
CREATE INDEX Instruction (Oracle, Sybase, SQL Server)	356
<i>Definition of an index</i>	356
<i>Processing and generating SQL commands</i>	357
<i>CREATE VIEW Clause</i>	357
Defining Database Views	358
Creating Database Views	358
<i>Add a table or a column to a view</i>	359
SQL Definition	359
<i>View joints</i>	359
<i>User mode</i>	359
<i>Fields</i>	360
Defining a Data Group	360
Defining Triggers for a Database	361
Creating Triggers	361

Trigger triggering	361
References	361
SQL Definition	362
Repository Integrity	362
Using Stored Procedures	363
Adding Physical Properties to Database Objects	365
Target DBMSs	365
Creating Physical Properties	365
Objects containing physical parameters.	366
Creating a new clause.	366
Connecting a clause	367
Naming clauses	367
Physical Model Customization Example	368
Generating the SQL File	371
Reverse engineer tables	373
Running Reverse Engineering	374
Physical Properties Reverse Engineering	376
Default Values	376
Eliminating Redundant and Transverse Values	376
Specific Cases	377
Physical properties of tablespaces	377
Clusters Reverse Engineering.	377
Extracting Database Schema Description from Data Sources.	378
Required Data Source Configuration	378
Downloading HOPEX Data Source Extractor	378
Starting Data Extraction	378
Extraction Report File	384
Extraction Results File.	384
Customizing ODBC Extraction	385
Using the Odwdbex.ini file and customized queries.	385
Using ODBC standard APIs	386
Select Clause Formats	386
Primary Keys	387
Foreign Keys	387
Indexes.	388
Columns	389

Pivot Types and Datatypes Correspondence Tables	391

COMPLIANCE

Rules and Regulations	483

Creating an Inventory of Regulatory Frameworks (External Regulations) 484

Importing the Module of Regulatory Frameworks	484
Structure of a Regulatory Framework	484
Creating a Regulatory Framework	485
<i>Defining the objects in your organization constrained by a regulatory framework</i>	<i>485</i>
<i>Viewing the objects constrained by a regulatory framework</i>	<i>486</i>
Adding a Section to the Regulatory Framework	486
<i>Defining the objects constrained by a section of a regulatory framework</i>	<i>486</i>
Adding an Article to a Section	487
Adding a Definition to an Article	487
Creating the Inventory of Control Directives	488
<i>Accessing the list of control directives</i>	<i>488</i>
<i>Supported and supporting directives</i>	<i>488</i>
<i>Enforcement level of control directives</i>	<i>489</i>
<i>Associating a control directive with an article of regulatory framework</i>	<i>489</i>
<i>Viewing articles associated with a control directive</i>	<i>490</i>

Creating an Inventory of the Policy Frameworks (Internal Regulations) 491

Creating a Policy Framework	491
Adding a Section to the Policy Framework	491
Adding a Business Policy	492
Analyzing Information Constrained by a Regulation	492

Importing UCF Documents 493

Using UCF Import	493
<i>UCF Import Prerequisites</i>	<i>493</i>
<i>Parameterizing UCF Import</i>	<i>494</i>
<i>Importing Data from the Common Controls Hub</i>	<i>494</i>
Defining the Applicable Regulatory Content	494
<i>Regulatory content relevance</i>	<i>494</i>
<i>Reviewing regulatory frameworks after UCF import</i>	<i>495</i>
<i>Selecting relevant content for your organization</i>	<i>495</i>

Ensure Compliance with Data: Create Business Rules 496

Defining a Business Rule	496
Rules Report	497

Data Quality	499
Defining a Data Quality Policy	500
Creating a Data Quality Policy	500
Defining Data Quality Controls	501
Identifying Data Quality Issues	501
Addressing the Data Quality Issues Encountered: Action Plans	502
<i>Properties of an action plan in HOPEX Data Governance</i>	503
Assess Data Quality	505
Quality Dimensions	505
Evaluation Objects	507
Direct Assessment	507
Assessment By Campaign	508
Data Quality Report	509

Data Reports and Analysis Tools	511
Analysis Report	512
Accessing HOPEX Data Governance Reports	512
Description Reports	512
<i>The View Report</i>	512
<i>Glossary Report</i>	512
<i>Data Domain Map</i>	513
<i>Relization Chart</i>	514
<i>Data Domain Dependencies</i>	514
<i>Data Categories Dendrogram</i>	516
Word Cloud Reports	516
<i>Amount of Information in Information Areas</i>	516
<i>Extent of the Description of the Information</i>	516
<i>Use of Information in Data Area</i>	516
Data Usage Reports	516
<i>Use of information held by a container</i>	516
<i>Use of information in an domain</i>	517
<i>Use of information of an information map</i>	517
<i>Use of information</i>	518
<i>Use of information of the domains of a container</i>	518
Regulatory Reports	519
<i>Regulatory Framework Report</i>	519
<i>Rules Report</i>	519
Data Catalog Reports	519
<i>Data Catalog Report</i>	519
<i>Data Quality Report</i>	519
Report DataSets	520
Creating a Report DataSets	520
Example of a Report Dataset	520
<i>Definition of terms used</i>	520
<i>Business dictionary x Concept Matrix</i>	521

Data Validation Workflow. 523

Validation workflow steps 523
Generating a workflow report 524

Data Import and Export 525

Importing Business Data from an Excel File 526

Downloading the Excel File Template 526
 Content of the Excel Template 526
 Term Sheet. 526
 Concept Sheet. 527
 Information Item Sheet (Concept Properties) 528
 Synonym Sheet. 529
 Component sheet 529
 State Concept sheet 530

Importing Logical Data from an Excel File 531

Downloading the Excel File Template 531
 Content of the Excel Template 531
 Data Dictionary sheet. 531
 Data Type sheet 532
 Data Type Component sheet 532
 Class sheet 533
 Attribute sheet 534
 Relationship sheet 534
 Generalization sheet 535

Importing Data Assessments 536

Import Example 536
 Content of the Excel Template 537
 Downloading the Excel Template 537
 Importing an Excel File of Data Assessments 537

INTRODUCTION TO HOPEX DATA GOVERNANCE



HOPEX Data Governance allows you to improve the quality of the data that circulates within your enterprise. It is used to construct the global architecture of data and trace the use of information through all the functions of your organization.

- 3 [The Scope Covered by the HOPEX Data Governance Solution](#)
- 3 [The HOPEX Data Governance desktop](#)
- 3 [Profiles of HOPEX Data Governance](#)
- 3 [Business Roles of HOPEX Data Governance](#)

THE SCOPE COVERED BY THE HOPEX DATA GOVERNANCE SOLUTION

HOPEX Data Governance has adopted an approach in accordance with data governance. As such, it offers a set of features that cover the following dimensions:

- data discovery, through ready-to-use connectors
- the definition of a business glossary based on the terms and their definitions
- data architecture
- the specification of business rules and the management of compliance with the regulations that apply to the enterprise
- the implementation of data lineage to monitor the path of information
- data quality evaluation
- data analysis, via standard reports supplied

Data Discovery

Through the **HOPEX Data Discovery** tool, you browse different data sources and define the metadata to import into your data catalog.

See [Data Discovery](#).

Business Glossary

HOPEX Data Governance allows you to draw up an inventory of enterprise terms and generate a business glossary that you can use to consult their definition, as well as their synonyms and components.

See [Consulting the Business Glossary](#).

Data Architecture

Three modeling levels

The **HOPEX Data Governance** solution covers the three levels of data modeling for an organization:

- Business (conceptual) level: used to define the business architecture concepts and generate glossaries. These concepts can be implemented

by objects at the logical level and be described by data models.

See [Introduction to the Creation of a Business Ontology](#).

- Logical level: intended for clients seeking to develop general business-oriented models. Here it consists of modeling the data of an area, application or business process. It represents what we wish to do and where we want to go, irrespective of technical questions related to implementation. Data is represented in a data model or a class diagram. See: [Modeling Data dictionaries](#).
- Physical level: consists of defining models intended to persist in a DBMS. It comprises detailed specifications for production of the physical diagram of the repository. It is represented by the relational diagram. The physical level also defines the way in which data is stored and how it can be accessed. It enables use of data by DBMSs. See [Modeling Databases](#).

Data category

You can classify repository data by category. A dedicated tree lists the various categories and associated data. Data thus classified can be used in the **HOPEX Privacy Management** solution specific to sensitive data and compliance with the GDPR.

See [Data Categorization](#).

Conception workflow

As a data designer (information asset manager) or creator (which concerns all IA profiles), you can launch a workflow on certain objects of the data architecture (such as a data lineage or a data domain) to track their design, their update and their validation.

Workflow reports allow you to view the number of objects that are found at each workflow step (number of objects undergoing design, analysis, etc).

Use of Data

To specify which business information is handled at the architecture level of the IS, **HOPEX Data Governance** offers a "Realization" function that connects the business information to the IS objects.

See [Connecting the Business Concepts to the Logical and Physical architecture](#).

You can also specify which processes and applications use which data, whether this concerns business, logical or physical data.

See [Use of Data by the Information System](#).

Data Quality and Compliance

To guarantee reliable and complete data, **HOPEX Data Governance** provides tools to define responsibilities for data, the rules and standards with which the enterprise must comply, and assess to what extent the data meets the requirements of the organization and its stakeholders.

Definition of responsibilities

When data is designed, managers are defined. They are notified of update or validation requests in the workflow framework of design or evaluation concerning the data in question.

See [Data Responsibility](#).

Definition of rules and standards

HOPEX Data Governance allows you to create an inventory of regulations and enterprise rules and to precisely define which articles or sections with which the various information and entities of an organization must comply. The solution supplies in particular content from the BCBS 239 banking regulations and the Solvability (Solvability) II regulation.

See [Regulations and Business Rules](#).

Traceability of Data (Data Lineage)

Through data lineage, you can represent the various processing procedures for the data to facilitate the identification of errors and reduce the risk of non-compliance. This allows the Data Steward and Data Owner to ensure the quality of the data used.

See [Describe a Data Lineage](#).

Evaluation of data

HOPEX Data Governance is used to ensure the quality of data using evaluations. For this, the solution provides an evaluation model that assesses the data in six areas: completeness, uniqueness, timeless, validity, accuracy and consistency.

Evaluations deal with metadata. When information exists in third-party tools on data instances, experts such as the data scientist or the data quality manager can relay this metadata information in **HOPEX Data Governance**.

For example, the "Account Manager" information is used in different data processing tools. The expert who observes missing data in some records of this information, for example the details of a person, can note this lack of completeness in HOPEX at the metadata level (the "Account Manager" class for example) so that an action plan can be created to correct the situation (via mandatory fields for example).

The evaluation can take place directly in the data properties dialog box or via an evaluation questionnaire sent to the managers of the data in question.

See [Data Quality](#).

Analysis reports

Analysis reports are dynamic reports that are used to analyze repository data: data completeness, data use, responsibilities, etc. **HOPEX Data Governance** supplies standard reports by default that allow you to check the quality, the use and the compliance of your data.

See [Data Analysis Reports](#).

THE HOPEX DATA GOVERNANCE DESKTOP

Connecting to HOPEX Data Governance

To connect to **HOPEX Data Governance**, see HOPEX Common Features, "HOPEX Web Front-End Desktop".

* *For more details on using the Web platform for HOPEX solutions, see the **HOPEX Common Features** guide.*

The menus and commands available in **HOPEX Data Governance** depend on the profile with which you are connected.

See [Profiles of HOPEX Data Governance](#).

Displaying the Working Environment of an Enterprise

A repository can be partitioned into *Enterprises*.

An enterprise is a purposeful undertaking, an effort conducted by one or more organizations, aiming at delivering goods and services, in accordance with the enterprise mission in its changing environment. The enterprise establishes the enterprise goals to be achieved as well as the strategic action plans used to achieve these goals. It comprises transformation stages in which the capacities or deliverables to be reached are defined.

When associated with a working environment, enterprises are entry points into IA; the environment provides privileged access to the objects held and used by the enterprise in question.

Creating an enterprise and its working environment

The creation of an enterprise and its working environment is performed by the IA functional administrator.

To create an enterprise in **HOPEX Data Governance**:

1. Click the navigation menu, then **Environment**.
2. In the navigation pane click **Standard Navigation**.
3. In the edit area click the **Enterprises** tile.
4. Click **New**.
5. In the creation wizard that appears, enter the enterprise name.
6. To create the enterprise environment at the same time, select the "Information Architecture" environment.
7. Click **OK**.

If no environment was created at the same time as the enterprise, you can create it later.

To assign a working environment to an existing enterprise in **HOPEX Data Governance**:

1. Select the project or the enterprise concerned to display its properties.
 - * *Click the Properties button of the edit area if properties are not displayed.*
2. Select the **Working Environment Assignment** page.
3. Click **New**.
4. Rename if needed the new environment and select the "Information Architecture" type.
5. Click **OK**.
 - * *You can also create the working environment if an enterprise during the enterprise creation.*

See also [Using Enterprises](#).

PROFILES OF HOPEX DATA GOVERNANCE

In **HOPEX Data Governance**, there are default user profiles with which specific rights and accesses are associated.

Profil	Description
Data Architect	He/She is responsible for modeling the data (business, logical and physical) as well as the data areas used to exploit this information in process or application mapping.
Data Asset Manager	Responsible for data governance
Data Functional Administrator	He/She is responsible for managing all the product's administrative tasks. The/She has rights to all objects. - He/She manages the users and their profile assignments. - He/She prepares the work environment and creates elements required for information management. - He/She can intervene in: • Dictionaries • Concept Domains • Concepts, concept views • etc.
Data Contributor	The data contributor is involved in the data workflow and participates in the definition and description of the data.

BUSINESS ROLES OF HOPEX DATA GOVERNANCE

In **HOPEX Data Governance**, objects can be assigned to persons with the following roles:

- **Chief Data Officer (DCO)**: responsible for data and its governance.
- **Data Owner**: this is the authority who decides on data access and use. The data owner can be the data designer, one of its users or a third party. Data stewards can ask data owners to check or complete the value of a field, to correct for example a default in the quality of data.
- **Data Designer**: the person responsible for the design of an object (such as a package, a data area, a database, etc.).
- **Data Engineer**: builds and maintains the tools and the infrastructures necessary for the analysis of data by data scientists. He/she creates solutions able to process large volumes of data while guaranteeing its security. He/she is the first link in the IT chain.
- **Data Scientist**: is responsible for bringing together the data designer (business and logical data) and the managers of the processes who use this data.
- **Data Quality Manager**: must ensure the relevant and usefulness of the enterprise data. To do so, he/she must implement data control procedures.
- **Data Steward** : guarantees the quality of data; this is the person who possesses the knowledge of the data and its metadata.

See also [Data Responsibility](#).



Data Catalogs



DESCRIPTION OF A DATA CATALOG



A data catalog is a metadata management tool that centralizes physical metadata in one place. It facilitates access to a perimeter of metadata and allows the organization to manage its definition, use and quality.

You can connect the metadata in the catalog to the organization's business objects, reference the regulations that apply to the metadata in the catalog and to which the organizations at stake are subject. You can also create a data lineage from the catalog to better track metadata usage, quality, and compliance with rules.

To define the scope of the metadata, **HOPEX Data Governance** offers a data discovery tool. It allows you to connect to multiple data sources (SQL, No SQL, files, storage, etc.) and extract a scope of metadata. See [Data Discovery](#).

- ✓ [Data Catalog Content](#)
- ✓ [Creating a Data Catalog](#)
- ✓ [Associating Keywords with Metadata](#)
- ✓ [Defining Metadata Categories](#)
- ✓ [Report and Dashboard of a Data Catalog](#)
- ✓ [Connecting Metadata to Business Information](#)
- ✓ [Collecting Business Data](#)
- ✓ [Catalog Data Quality](#)
- ✓ [Snapshot of a Data Catalog](#)

DATA CATALOG CONTENT

A data catalog is a metadata management tool in HOPEX.

To access the data catalogs:

- Click the navigation menu then **Catalogs > Inventory**.

The data catalog references data sources (deployed data stores) that contain physical metadata.

To the data catalog you can associate:

- the managed data sources (deployed data stores)
- the organizations at stake, which use or consume the data in the catalog.
- the regulatory frameworks and business policies that apply to the metadata in the catalog, and to which the organizations involved are subject. See [Regulations Associated with the Catalog](#).
- a concept domain map
- data lineages

Deployed Data Store

A Deployed Data Store is a collection of metadata. It is held by a Data Center.



A data center is a physical site that groups together IT facilities responsible for storing and distributing data through an internal network or via Internet access.

*The list of data centers is accessible under the **Environment > Infrastructure Landscape**.*

The data store can contain:

- Meta dataset schemas, which group meta datasets
- Meta datasets (e.g. tables).

Meta dataset

A Meta Dataset represents the structure of a collection of related data held in a data source. The represented data structure can be relational (Table) or NoSQL (triple store, document store, flat files, etc.).

Meta field

A Meta Field is a Physical Data Property that is the specification of the content of a Meta Dataset.

A Meta Field corresponds to a column in a table or an Excel sheet column.

It can be of the following types:

- simple meta field
- structured meta field: it is a data field composed of other data fields (example : postal address)
- primitive type of a field: type that identifies the field (example: "Address", "Code", "Date")

To connect a meta field (simple or structured) to a meta dataset, you will first indicate whether it is a field that identifies the metadata, a referenced field or a field that is neither identifying nor referenced.

To connect a meta field to a meta dataset:

1. Open the properties of the meta dataset.
2. In the **Meta Fields** page, click **New**.
3. In the window that appears, select the desired field type, which indicates whether the field to be connected is identifying or referenced:
 - Identifier: meta field that allows to distinguish the object. It can be a simple or structured meta field.
 - Reference: meta field that is not held by the meta dataset. It can be a simple or structured meta field.
 - Standard: meta field that is neither identified nor referenced. It can be a simple or structured meta field.

Regulations Associated with the Catalog

You can connect regulatory frameworks or business policy frameworks to the catalog.

To connect a regulation to the catalog:

1. Under the catalog name, hover over the **Regulatory Frameworks and Business Policies** folder.
2. Click the link that appears to the right of the folder.
The search window appears.
3. Select the type of regulation and click the **Search** button.
A list of matching regulations appears.
4. Select the desired regulation and click **Connect**.

You can further define which sections of a regulation and/or business rule an entity must comply with.

Example

An organization is constrained by the regulation concerning building accessibility and the adaptation of working conditions for handicapped persons.

To define the elements of a data catalog regulation:

1. Click the catalog icon and select **Edit the data catalog**.

The data catalog editing window appears.

On the left side of the editor are the regulatory frameworks or business policy frameworks attached to the data catalog. On the right you can see the data catalog.

To define the data catalog items, drag and drop the regulatory sections from the left tree into the right tree

Data Lineage of a Data Catalog

You can associate the data catalog with a data lineage to visualize the implementation and track the regulations that constrain the data catalog within the organization.

To connect a data lineage to the data catalog:

1. Open the properties of the data catalog.
2. Click the **Activities** page.
3. Under the **Data Lineage** section, click **New** or **Connect** depending on whether you want to create a data lineage or connect an existing data lineage.

See [Data Lineages](#).

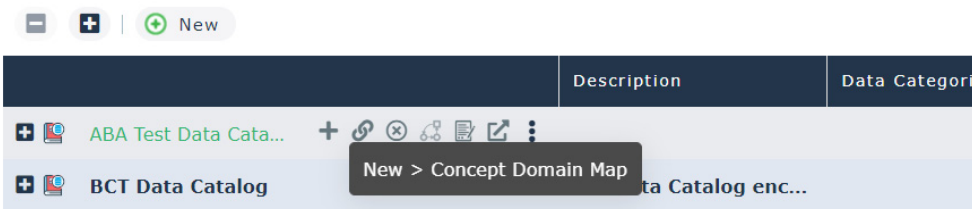
Concept Domain Map of a Data Catalog

A Concept Domain Map organizes into domains the use of business data that is implemented by the meta dataset and metadata fields. Thus, the data catalog not only references physical and technical data sources but also provides a business-oriented entry point via this map.

📖 For more information on Concept Domain Maps, see [Concept Domain Map](#) in the "Defining Business Information" chapter.

To connect a concept comain map to a data catalog:

1. Move the mouse over the data catalog.
2. Click **+ New > Concept Domain Map**.



Business data connected to the data catalog through the map is displayed in the Business Data page of the catalog properties.

See also: [Collecting Business Data](#).

SEARCHING THE DATA CATALOG

You can search for discovered and imported metadata in the data catalog. The search tool allows you to:

- identify the origin and the people responsible for the metadata
- see the use of metadata in processes and applications
- access technical data lineage, to visualize data processing and data flows
- see associated quality policies and regulations

The screenshot displays the 'Data Catalog' search interface. At the top, a search bar contains the term 'Order'. Below the search bar, there are filters for 'Data Source', 'Schema', 'Data Owner', 'Business Term', 'Used in applications', 'Used in processes', and 'Data Processing'. The search results are listed on the left, showing four items: 'ReferenceOrderLineID' (Simple Meta Field), 'ReferenceOrderID' (Simple Meta Field), 'TransactionHistory' (Meta Dataset), and 'ReorderPoint' (Simple Meta Field). The 'TransactionHistory' item is highlighted with a blue border. On the right, the 'TransactionHistory' section shows a 'Meta Dataset structure' diagram titled 'Dendrogramme des traitements de métadonnée'. This diagram illustrates the data lineage, showing 'TransactionHistory' branching into 'Meta Fields' and 'AdventureWorks2017'. 'Meta Fields' further branches into 'ActualCost', 'ModifiedDate', 'ProductID', 'Quantity', 'ReferenceOrderID', 'ReferenceOrderLineID', 'TransactionDate', 'TransactionID', and 'TransactionType'. 'AdventureWorks2017' branches into 'Data Catalog-1 (EN)', 'Data Center-1 (EN)', and 'Microsoft SQL Server (EN)'. 'Production' also branches into 'AdventureWorks2017'.

Performing a Search

The list of metadata concerned are meta datasets, simple and structured meta fields, schemas, and deployed data stores.

This tool searches for character strings in the Name and Description of metadata.

To search for metadata in the data catalog:

1. In the navigation menu click **Catalogs** > **Search**.
2. In the edit area, in the **Search** field, enter the name or the first letters of the name of the searched metadata.

➡ Used at the end of a word as in "Liberal*", the asterisk allows you to find the different suffixes like "Liberalism" or "Liberalization".

The list of matching objects appears.

Search filters

You can refine the results using the filters in the left panel:

- Data source
- Responsible party
- Data category
- Link to a business term
- Compliance with a quality policy or regulation
- Use of the metadata
- Presence of a data lineage

The values proposed in the filters depend on the results obtained: for example, when the searched metadata appears in several sources, the filter allows you to restrict the search to one of the sources found.

Metadata characteristics

When you select a metadata in the result list, further information appears on the right side of the edit area.

The **Overview** tab shows:

- Responsibilities on the metadata
- Indicators about the objects associated with the metadata
- A description

The **Advanced** tab shows:

- Data lineages
- Applications and processes that use the metadata
- Quality policies that address the metadata
- Regulations

The **Data Processing** tab displays in a dendrogram report details of physical data processing scanned and imported from Manta in **HOPEX Data Governance**.

CREATING A DATA CATALOG

To create a data catalog:

1. Go to **Catalogs > Inventory**.
2. In the edit area, click **New**.
3. In the catalog creation window you can define:
 - the catalog name
 - the owner
 - the organizations at stake
 - the regulatory frameworks and business policies that apply to the metadata in the catalog

 For more details on regulations, see [Rules and Regulations](#).
4. Click **OK**.

Once the catalog is created, you can define the Deployed Data Stores.

Data Catalog Workflow

A workflow is available to follow the different states of a data catalog:

- Preparation
- Validation
- Update
- Production

It can be instantiated by the owner of the data catalog and any manager assigned to it can participate in its progress.

Notifications are sent to the different managers whenever the status of the data catalog changes.

You can define the managers of a data catalog in the catalog properties, under the **Characteristics > Responsibilities** page.

ASSOCIATING KEYWORDS WITH METADATA

You can add tags to meta dataset, and more precisely to their fields, and thus link them to the business glossary to share the definition.



A tag is a classifying description used to characterize objects. Tags are used as #tags (hashtags) in the quick search tool.

To add a tag to a meta dataset:

1. Under **Catalogs** > **Inventory**, select the catalog that holds the relevant meta dataset.
2. Unfold the catalog tree and select the meta dataset.
3. Open its property window.
4. In the **Characteristics** page, under the **Identification** section, select under the **#Tags** field the appropriate tag.

DEFINING METADATA CATEGORIES

To define the category of a metadata:

1. Under **Catalogs** > **Inventory**, select the catalog that holds the relevant meta dataset.
2. Unfold the catalog tree and select the meta dataset.
3. Open its property window.
4. In the **Characteristics** page, under the **Identification** section, select under the **Data Category** field the appropriate category.

See also [Data Categorization](#).

REPORT AND DASHBOARD OF A DATA CATALOG

Two tools are available to analyze the content of a data catalog:

- The dashboard integrated into the catalog
- The Data Catalog Report

Dashboard of the Data Catalog

The data catalog dashboard is available in the data catalog properties **Dashboard** page. It provides an overview of the catalog properties.

Data Catalog Report

The data catalog report is used to analyze the content of one or more data catalog(s). This report is available for all **HOPEX Data Governance** profiles.

To generate a data catalog report:

1. Click the navigation menu then click **Reports > Data Catalog Reports**.
2. In the edit area, click the **Data Catalog Report** tile.
3. Select the relevant data catalog from the drop-down list.
4. Refresh the report.

It includes the following chapters:

- Organizations at stake: entities that use or consume the data in the catalog.
- Referenced Regulatory and Business Policy Frameworks
- Categories/Sections
- Data lineage See also [Data Lineages](#).

CONNECTING METADATA TO BUSINESS INFORMATION

You can initialize a business dictionary from metadata. This consists of creating in a business dictionary the concepts and information items that correspond to the physical metadata and the metadata fields, and thus establishing a realization link between these objects.

For more details, see [Initializing a Business Dictionary from Meta Datasets](#).

COLLECTING BUSINESS DATA

Under the **Business Data** page of a data catalog's properties appear the business data implemented by the metadata and/or defined in the concept domains of the data catalog.

➤ For the implementation of business data through metadata, see [Connecting Metadata to Business Information](#).

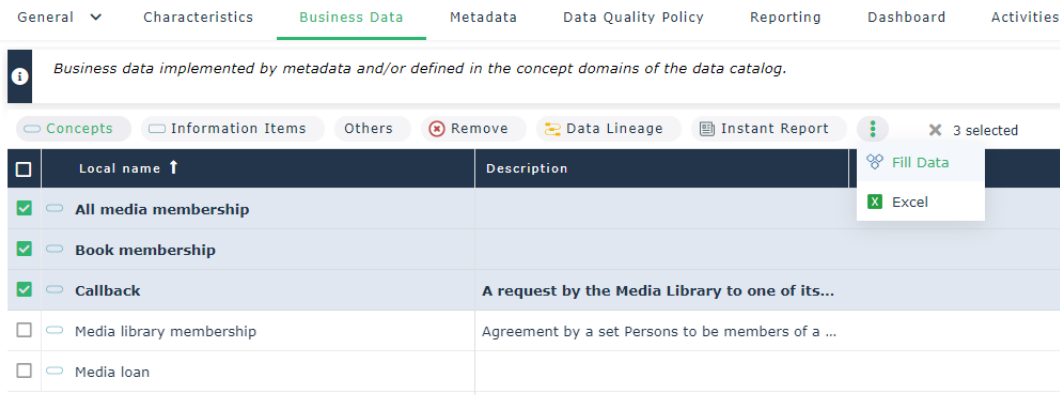
➤ For the definition of concept domains in a catalog, see [Concept Domain Map of a Data Catalog](#).

The data catalog administrator can ask business owners to complete the business data properties associated with the catalog. This request is made through an evaluation session.

The managers contacted are the Data Owner, the Data Designer and the Data Steward defined on the data in question, or, failing that, those defined on the business dictionary or the data domain.

To collect business data:

1. Open the properties of the data catalog concerned.
2. Click the **Business Data** page.
3. From the list displayed, select the data to be completed.
4. Click the button **More...**  Then **Fill Data**.



General ▾ Characteristics **Business Data** Metadata Data Quality Policy Reporting Dashboard Activities

i Business data implemented by metadata and/or defined in the concept domains of the data catalog.

Concepts Information Items Others Remove Data Lineage Instant Report 3 selected

☐	Local name ↑	Description	 Fill Data
☒	All media membership		 Excel
☒	Book membership		
☒	Callback	A request by the Media Library to one of its...	
☐	Media library membership	Agreement by a set Persons to be members of a ...	
☐	Media loan		

The assessment wizard appears.

5. Plan the assessment session if necessary.
6. Click **Next**.
7. Select the items you want to send to the data managers:
 - one or more property pages (e.g. the Data Quality page if you want managers to evaluate data quality)
 - advanced characteristics (special attributes, e.g. modification date, or technical properties)
8. Click **Next**.

9. Start the session immediately or at a specific date and time.
A notification is sent to the data owners, containing direct access to the form.

Monitoring of Data Call Forms

To access the data call forms received or sent:

1. Click the navigation menu then **Assessment**.
2. In the edit area, click the **Data Call Forms Follow-up** tile.
3. Indicate the type of form to be display:
 - Sent forms
 - Answered forms
 - Unanswered forms
 - Late forms

CATALOG DATA QUALITY

To ensure the quality of data in a catalog, you can:

- connect a data quality policy to the data catalog
- more precisely connect a data quality policy to physical metadata in the data catalog.

Connecting a Data Quality Policy to the Data Catalog

To connect a data quality policy to a data catalog:

1. Click the navigation menu then **Catalogs > Inventory**.
2. In the edit area, select the desired catalog and display its properties window.
3. In the properties window, click the **Data Quality Policy** page.
4. Click **New** or **Connect** depending on whether you want to create a data quality policy or connect an existing data quality policy.

Connecting a Data Quality Policy to Physical Metadata

To connect a data quality policy to a metadata:

1. Click the navigation menu then **Compliance > Data Quality Policies**.
2. Open the properties of the data quality policy.
3. Click the **Characteristics** page.
4. In the **Quality Policy Subjects** field, select the object that is constrained by the quality policy.

See also: [Defining a Data Quality Policy](#).

SNAPSHOT OF A DATA CATALOG

The purpose of snapshots is to collect performance indicator values at regular intervals that allow for data quality control.

You can create a snapshot of a set of metadata when importing the metadata into **HOPEX Data Governance**.

To view a snapshot of a physical data set:

1. Click the navigation menu then **Catalogs > Snapshots**.
2. Select:
 - the data catalog
 - the deployed data store
 - the snapshot created during the import of the metadata. See [Running HOPEX Data Discovery](#).

For more details on the data catalog contents, see [Data Catalog Content](#).



DATA DISCOVERY



The HOPEX Data Discovery tool allows you to extract data from various sources and import them into the data catalog of your repository, where they can be analyzed via graphical representations that promote the visualization of useful data and possible anomalies.

- ✓ [The HOPEX Data Discovery Module](#)
- ✓ [Running HOPEX Data Discovery](#)

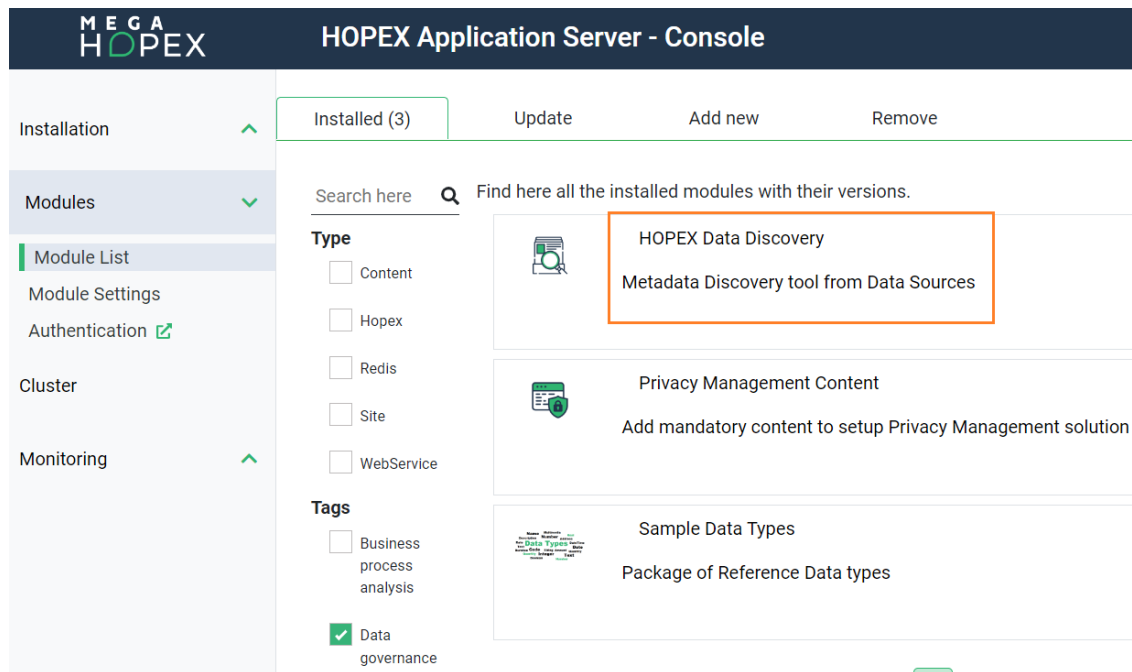
THE HOPEX DATA DISCOVERY MODULE

Deployment

The HOPEX Data Discovery tool is available as a module, downloadable from the HOPEX HAS console.

To implement the HOPEX Data Discovery module under HAS:

- 1 From your HOPEX version, open the HAS Administration Console and download the module. Once the module is installed, you must update the environment.
for the complete procedure, see [Importing a Module into HOPEX](#).



Connection Options to Data Sources

The JDBC driver connection options for each data source are detailed in the Appendix of the HOPEX Data Governance Guide. See "Appendix - Data Discovery Connection Options".

RUNNING HOPEX DATA DISCOVERY

The **HOPEX Data Discovery** tool allows you to browse different data sources and define the metadata to import into your data catalogs.

Data Discovery is available in Online mode in the HOPEX desktop. It allows you to connect to data sources and import metadata directly into HOPEX.

If you cannot connect to the data source from the **HOPEX Data Governance** desktop, for example for security reasons, you can download the standalone version of **Data Discovery** to connect to the data sources. In this mode, the metadata description is exported to a Json file that you can then import into HOPEX. In this mode, the metadata description is exported in a Json file. that you can then import into HOPEX.

Access conditions

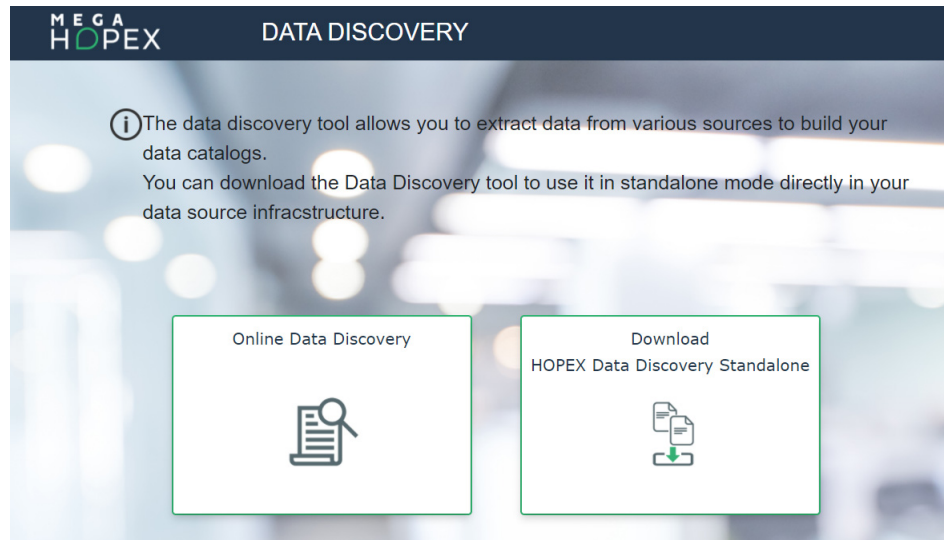
The Metadata Discovery Tool is available to users with the profile "IA Functional Administrator" and "Data Architect."

Online mode

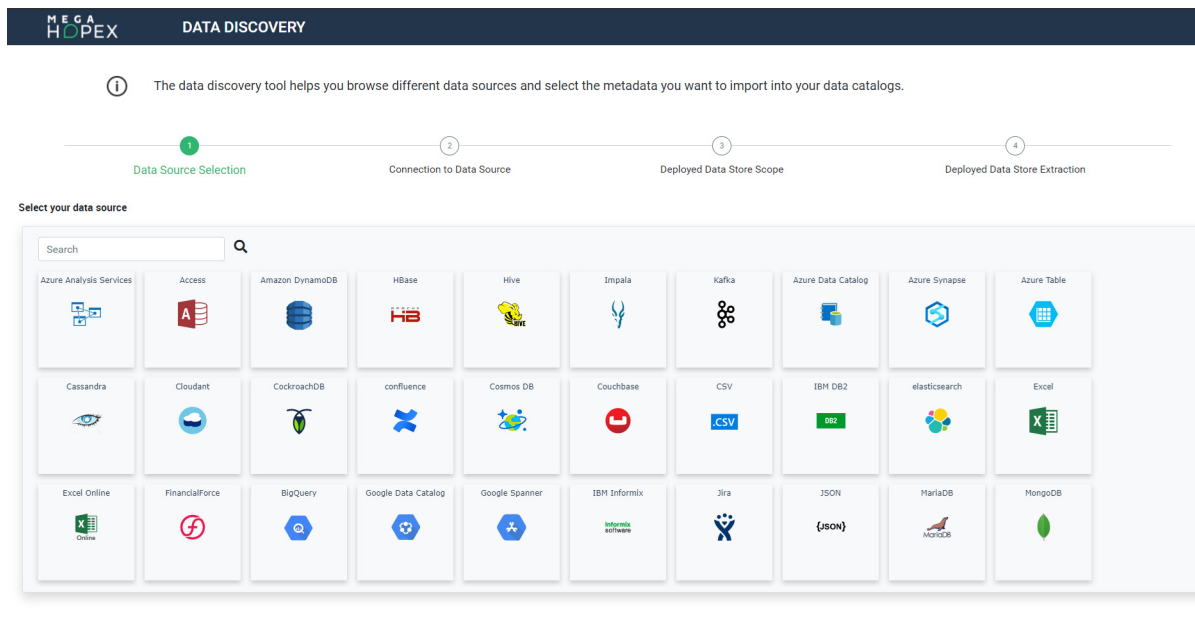
To run the Online Data Discovery:

1. In the navigation menu, click **Catalogs > Inventory**.
2. In the edit area, click the **Data Discovery** button.
The HOPEX Data Discovery tool opens in a new browser tab.

3. Click **Online Data Discovery**.



The list of data sources appears.



4. Select the data source. Click the **Connect** button on the source. The connection settings appear. They depend on the chosen data source. You can view the associated information.

*In the appendix of the **HOPEX Data Governance** guide are detailed the connection options of the JDBC driver for each data source.*

MEGA
HOPEX
DATA DISCOVERY
?

1
2
3
4

Data Source Selection
Connection to Data Source
Deployed Data Store Scope
Deployed Data Store Extraction

Set your credential to connect to your Data Source.

Microsoft SQL Server

Server		?	
User	vpl	?	
Password		?	
Port	1433	?	
Azure Tenant		?	
Options		?	

Advanced

« Previous

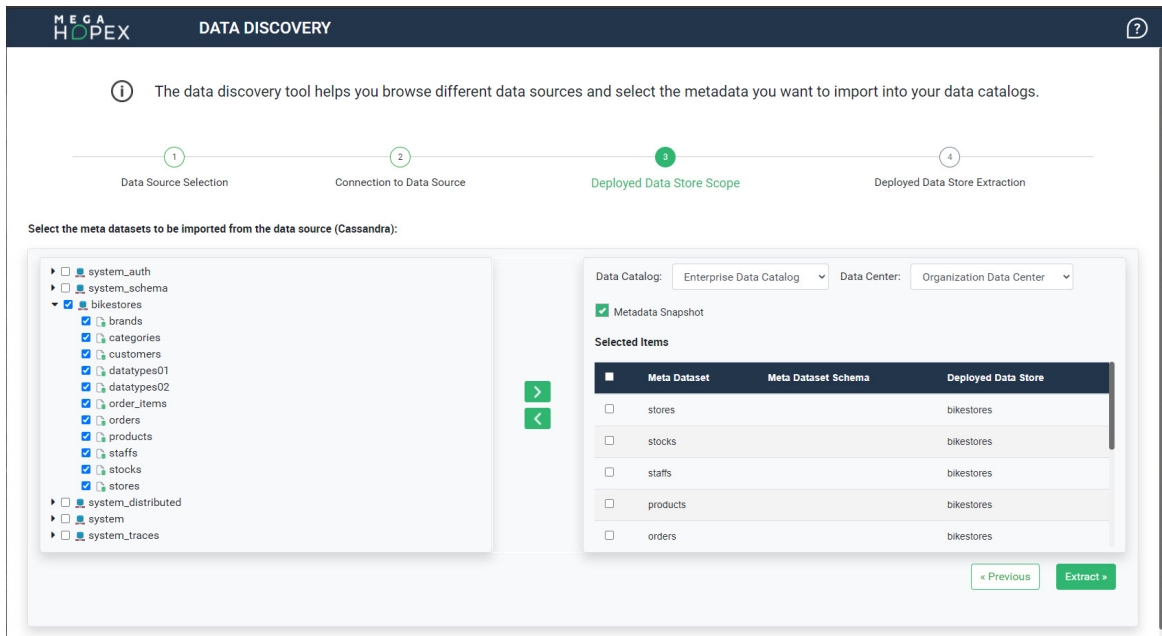
Connect »

The port of the MS SQL Server.

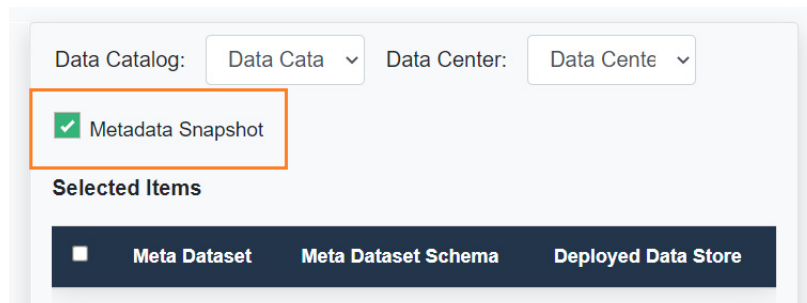
Some parameters are optional, you can display them with the **Advanced** button.

5. Once connected, in the left tree, select the objects to import into HOPEX.
6. On the right, select the repository data center that will hold the metadata and the catalog that references it.

7. Using the first arrow, add the desired items to the catalog tree.



You can create a snapshot of the selected metadata set by checking the **MetaData Snapshot** box. See [Snapshot of a Data Catalog](#).



8. Click **Extract**.

To view the objects imported in HOPEX:

1. In the navigation menu, click **Catalogs > Inventory**.
2. In the edit area, select the catalog.
3. Expand the **Deployed Data Stores** folder.

Stand-alone mode

Prerequisite

The workstation on which the module is installed in standalone mode must have .Net Core 6 and Microsoft Edge WebView2 installed.

Running the import in standalone mode

To run the import in standalone mode:

1. In the navigation menu, click **Catalogs > Inventory**.
2. In the edit area, click the **Data Discovery** button.
The HOPEX Data Discovery tool opens in a new browser tab.
3. Click **Download HOPEX Data Discovery Standalone** to download the module.
A zip file with the extension *.haspkg is downloaded.
4. Unzip the file.
A folder with the same name is created.
5. Click on has.desktop to deploy and launch the module.
6. Once the module is launched, select the data source.
The list of parameters depends on the chosen data source. You can view the associated information.
7. Once connected, in the left tree, select the objects to import into HOPEX.
8. Using the first arrow, add the desired items to the catalog tree.
9. Click **Extract**.
10. When the extraction is complete, import the Json file.

Importing the JSON file

To import metadata using the .json file:

1. In the navigation menu, click **Catalogs > Inventory**.
2. In the edit area, click the **Metadata Import** button.
3. In the dialog box that appears, select:
 - the data catalog to which to import data
 - the holding data center
 - the .json file
4. Click **Next**.
The wizard presents the result of the import.
5. Click **OK**.

See also: [Connection Options to Data Sources](#).

Incremental import

After a first initialization of metadata in HOPEX, it is possible to make other imports from the same source. In this case, only new objects and property updates are taken into account (example: creation of a new meta dataset, new meta field).

Metadata in HOPEX is identified by its name + the name of the deployed datastore (from the source database) + the name of the datacenter. As soon as a component of the name is modified, the correspondence is broken and new objects are created. So, for example, if you change the deployed data store of a table in HOPEX, when you import it again from the same source, this table is recreated in the original data store.

DATA LINEAGES



Financial regulations and new frameworks such as the General Data Protection Regulation (GDPR) require the availability of a mapping and controls for data processing. **HOPEX Data Governance** provides a data lineage tool to ensure the traceability of data flows.

HOPEX Data Governance provides a data lineage tool to ensure the traceability of data flows. A data lineage tool describes the information processing and transformation rules, from source to restitution. It thus facilitates the identification of errors in processing and reduces the risk of data non-compliance.

- ✓ [Functional Data Lineage](#)
- ✓ [Technical Data Lineage](#)

FUNCTIONAL DATA LINEAGE

Functional data lineage is a way of specifying, defining or viewing data lineage without the technical information that the business user does not need.

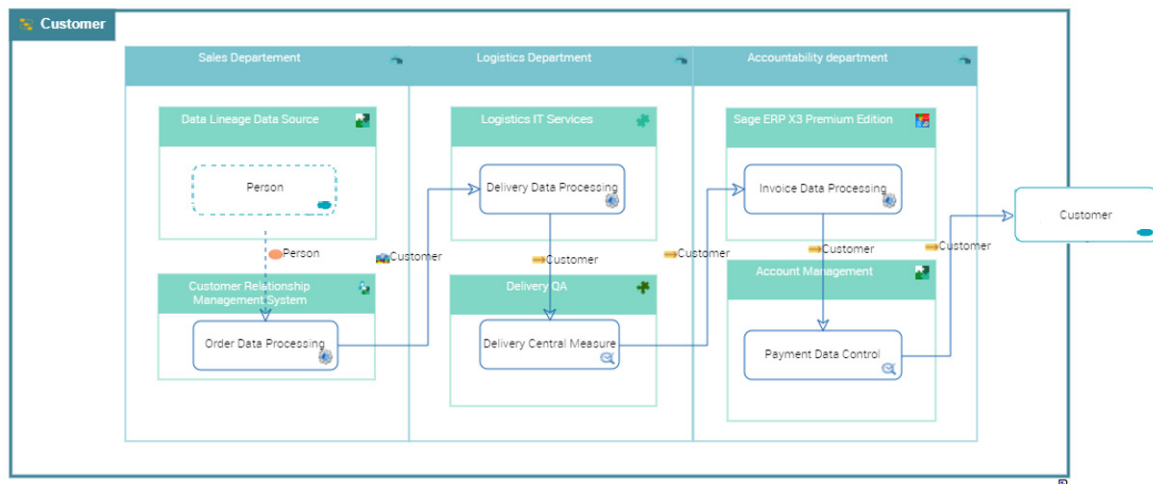
To specify or consult a functional data lineage, data managers have a diagram in which they can draw the various processes and transformations that apply to the data independently of their implementation. They can thus specify:

- Where the data is located and how it is stored in an environment, for example on a site or in a data store.
- By which processes and systems the data is used, and with which access rights.
- Who is responsible for updating the data.
- Etc.

An analysis report shows the dependencies that exist between enterprise data.

Example of Lineage of a Business Data

Below is an example of the lineage of the "Customer" concept: the lineage presents the path of the data, with the participating departments (Sales, Logistics, Accountability), the treatments performed and by which applications or technologies.



Creating a Functional Data Lineage and its Diagram

You can create a data lineage on a concept, a class or a table.

To create a data lineage in **HOPEX Data Governance**:

1. Open the properties of the object (concept, class or table).
2. Click the **Characteristics** page.
3. On the right of the page, under the **Data lineage** field, click the **New** button.
4. Select the type of diagram.

Example: a physical data lineage diagram for a table.

The **Initialize diagram** option is checked by default. See [Initialisation of the diagram](#) below.

5. Click **OK**.
6. Select the relevant object and click **Next**.
The data lineage diagram opens.

Initialisation of the diagram

When the data described by the lineage is used in content that exists or is associated with calculation rules, the **Initialize the Diagram** option appears; it is used to initialize the diagram with the content and rules in question.

More precisely, initialization is based on:

- the content that carries the data item in the exchange flows between the software (applications, Application Systems, etc.), described in the flow scenarios.
- Scenario software is used to initialize the data sources in the lineage. Flow transition, its reception and its direction in the scenario enable initialization of data processing, its transitions and directions.
- When the tasks performed by the software of a scenario are detailed in an application process for example, they also appear in the lineage processing nodes.
- the calculation rules available for this data item. The information defined as input parameters of these rules initialize the original nodes of the lineage.

Data lineage workflow

When creating a data lineage (business, logical or physical), the first transition of the workflow -"In design" - is automatically created.

For more details on the workflow, see [Data Validation Workflow](#).

Objects of the Data Lineage Diagram

The toolbar on the left of the diagram shows the objects you can insert into the diagram. Some, such as Org-Units, can be connected. Other objects, such as Applications or Application services, can also be created.



Participant

A participant is used to define where a processing takes place. This can be an org-unit or a position.

Adding a participant

You can add an existing org-unit or position to the diagram

To add an existing org-unit:

1. In the object insertion bar, click the **Org-Unit** icon and then in the diagram.
2. In the window that appears, enter the name of the org-unit.
3. Select the org-unit that appears in the list.
4. Click **OK**.

Origin business data

This is the input data of the data lineage.

Data flow

The data flows represent the transitions between the data lineage steps, through which the input and output data are provided.

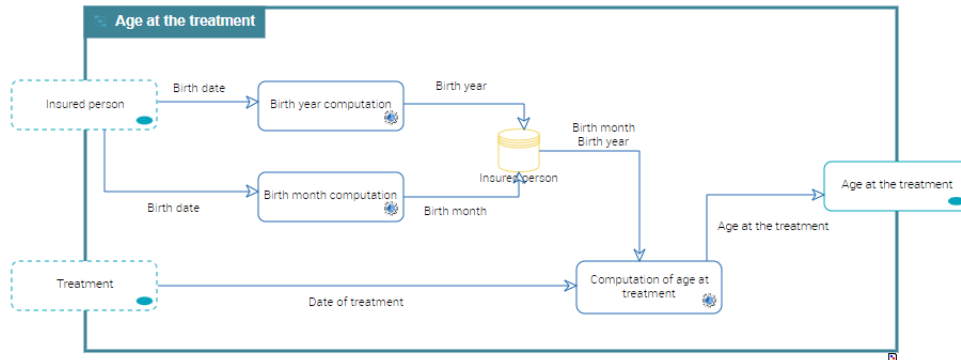
Concept property rule

A concept property rule is the collection and handling of data elements to produce significant information.

In the example below, the concept property rule
"Computation of age at treatment" is used to calculate the

age of the patient at the time of treatment. The rule takes as parameters:

- the month and year of birth available in the "Insured Person" concept store and deduced from the properties of the "Insured Person" concept.
- the date of the treatment, taken from the "Treatment" concept.



Creating a concept property rule

In the example above, to define the rule that calculates the age of the insured person at the time of treatment, you must:

- Create a concept property rule
- Create the flows between the information sources and the concept property rule
- Specify on the flows the information transmitted
- Specify on the rule the input and output information

To do this:

1. In the object insertion bar, click **Concept Property Rule**.
2. Click in the diagram.
3. Specify the name of the rule ("Computation of age at treatment") and click **Next**.
The rule appears in the diagram.
4. Click the **Data Flow** icon and create a data flow between the "Insured Person" lineage concept store and the concept property rule.
5. Select the data flow and click the **Properties** button.
6. Click the **Characteristics** page.
7. Under the **Owned Data Property Lineage** section, add the properties "Birth month" and "Birth year".
8. Create a second flow between the "Treatment" Concept and the concept property rule.
9. On this flow add the property "Date of treatment".
10. Select the concept property rule to display its Properties.
11. Click the **Characteristics** page.

12. Under the **Parameters** section, select :
 - the rule's input parameters: "Birth month", "Birth year" and "Date of Treatment".
 - the output parameter of the rule: "Age at the treatment".

Data quality measure

A data quality measurement checks that the data issued by a process is compliant with the quality requirements. It is therefore used to prevent risks (erroneous or missing data for example) linked to data calculation or processing.

Adding a data quality measure

In the lineage diagram you can connect existing measures. To do this:

1. In the object insertion bar, click the **Data Quality Measure** icon.
2. Click the diagram.
3. In the window that appears, enter the name or the first letters of the name of the data quality measure.
The corresponding quality measure appears in the drop-down list.
4. Select it.
5. Click **OK**.

To access all data quality measurements, click the navigation menu then **Compliance > Operational Assurance > Data Lineages > Quality Measurement Inventory**.

Processing software

These are software systems that perform data processing. You can select one of the following elements as a software source:

- Application System
- Application
- Micro-service
- IT dept

You also display the traceability of data use at the software level. The software system can be under the responsibility of an org-unit or a type of position.

You can define the following in the software system:

- data stores (data areas) that contain the saved information (original data, data flows, etc.), the CRUD defined.
- software components that access the data store
- application processes where the data is processed

Adding a processing software

You can add an existing software to the diagram or create a new one.

To create an application in the diagram for example:

1. In the object insertion bar, click the **Application** icon and then in the diagram.
2. In the window that appears, enter the name of the application.

3. Click **Next**.
4. Indicate its owner.
5. Click **OK**.
The application appears in the diagram.

Technology

The technologies represent the components required for the operation of software (servers, operating systems, etc.) that are part of the data lifecycle.

TECHNICAL DATA LINEAGE

A technical data lineage is the technical and detailed representation of the transformations that apply to the data. This representation is intended for technical data managers and data engineers.

From HOPEX you can access the technical data lineages available in Manta. Manta is an automated data lineage platform designed to extract, from a selection of source systems (data sources, file systems), the technical information of data flows, transformations and dependencies.

Manta Import Overview

Manta and HOPEX integration involves importing into HOPEX the dictionary of metadata scanned by Manta, together with their descriptions and references (URLs), so that their technical lineage can be consulted in Manta from HOPEX.

Imported objects are metadata from data sources such as deployed data stores, schemas, meta dataset and meta fields.

Object mapping		
Manta	HOPEX	Properties
Server	Deployed server	Manta identifier, name
Database	Deployed data store	Manta identifier, name
Schema	Meta dataset schema	Manta identifier, name, Manta link
Table	Meta dataset	Manta identifier, name, description, Manta link
Column	Simple meta field	Manta identifier, description, data type, length, decimal, Manta link, identifier field, referencing field
View	Metadata view	Manta identifier, name, Manta link
DBMS	Software technology	Name

Processes (stored procedures, functions, triggers, programs, scripts, reports, technical lineage) are not imported into HOPEX. This information is stored in Manta and referenced by HOPEX.

Prerequisite Conditions

Technical data lineage functionalities are available with the **HTDL** product option, which is protected by a license.

To view Manta technical data lineages from HOPEX, access must be authorized between the Manta server and the HOPEX server, which can be deployed on site or in the Cloud.

Also, make sure that the URLs of the objects extracted from Manta and imported into HOPEX are public and accessible to the **HOPEX Data Governance** solution. These URLs are generated in the "Dictionary.csv" file, in the "manta_hyperlink" column.

Contact your Manta administrator to specify a public URL for access to Manta from HOPEX.

D		E	F
name	object_name	object_description	manta_hyperlink
	VP-SNY-MANTA\SQLSERVER2019		<a >visualization="" a="" data="" href="http://vp-sny-manta:8080/manta-dataflow-server/viewer/dataflow/link?depth=3&direction=BOTH&filter=1&filterEdges=false&level=TOP&startNode=(MSSQL,MSSQL),(%22VP-SNY-MANTA%5C%5CSQLSERVER2019%22,Server)" in="" lineage="" manta<="" of=""> <a >visualization="" a="" data="" href="http://vp-sny-manta:8080/manta-dataflow-server/viewer/dataflow/link?depth=3&direction=BOTH&filter=1&filterEdges=true&level=TOP&startNode=(MSSQL,MSSQL),(%22VP-SNY-MANTA%5C%5CSQLSERVER2019%22,Server)" in="" indirect="" lineage="" manta<="" of="">
	Hotel_Booking		<a >visualization="" a="" data="" href="http://vp-sny-manta:8080/manta-dataflow-server/viewer/dataflow/link?depth=3&direction=BOTH&filter=1&filterEdges=false&level=TOP&startNode=(MSSQL,MSSQL),(%22VP-SNY-MANTA%5C%5CSQLSERVER2019%22,Server),(Hotel_Booking,Database)" in="" lineage="" manta<="" of=""> <a >visualization="" a="" data="" href="http://vp-sny-manta:8080/manta-dataflow-server/viewer/dataflow/link?depth=3&direction=BOTH&filter=1&filterEdges=true&level=TOP&startNode=(MSSQL,MSSQL),(%22VP-SNY-MANTA%5C%5CSQLSERVER2019%22,Server),(Hotel_Booking,Database)" in="" indirect="" lineage="" manta<="" of="">
	tel_Book dbo		<a >visualization="" a="" data="" href="http://vp-sny-manta:8080/manta-dataflow-server/viewer/dataflow/link?depth=3&direction=BOTH&filter=1&filterEdges=false&level=TOP&startNode=(MSSQL,MSSQL),(%22VP-SNY-MANTA%5C%5CSQLSERVER2019%22,Server),(Hotel_Booking,Database),(dbo,Schema)" in="" lineage="" manta<="" of=""> <a >visualization="" a="" data="" href="http://vp-sny-manta:8080/manta-dataflow-server/viewer/dataflow/link?depth=3&direction=BOTH&filter=1&filterEdges=true&level=TOP&startNode=(MSSQL,MSSQL),(%22VP-SNY-MANTA%5C%5CSQLSERVER2019%22,Server),(Hotel_Booking,Database),(dbo,Schema)" in="" indirect="" lineage="" manta<="" of="">

Importing Manta Data

Data from Manta is exported in a .csv file, named Dictionary.csv. This file can then be imported into HOPEX as a .zip file, for use in a data catalog.

To import the Dictionary.csv file in **HOPEX Data Governance**:

1. In the navigation menu, click **Tools > Manta Import**.
2. The data lineage import wizard opens.
3. Specify:
 - the data catalog, which will reference the imported metadata
 - the data center that will hold the metadata
 - the .zip file containing the data lineage from Manta.

☛ For import into HOPEX to work, make sure that the .csv files are at the root of the .zip file and not in folders.

4. Click **OK**.
The scheduler window appears. It indicates that the file will be imported in batch mode.
5. Click **OK**.

At the end of the import, a notification and an e-mail are sent to the user who launched the import and to the owner of the data catalog.

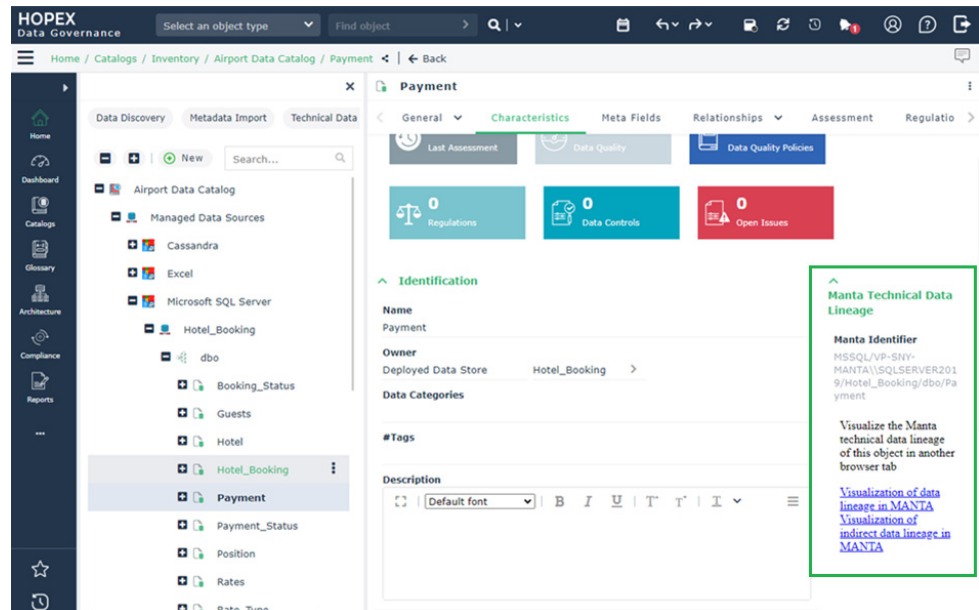
Once the metadata has been imported into HOPEX, the technical data lineage can be viewed by calling up the Manta Viewer tool.

Viewing the Technical Data Lineage

Accessing the technical data lineage

You can access from **HOPEX Data Governance** the Manta data lineage of a meta dataset. To do this:

1. Open the properties of the meta dataset.
It must be a metadata that has been imported from Manta.
2. Click the **Characteristics** page.
3. On the right side of the page, click the Visualization of data lineage in Manta.



4. Log in to your Manta account.



INTRODUCTION TO THE CREATION OF A BUSINESS ONTOLOGY

HOPEX Data Governance and **HOPEX Information Architecture** offer a solution for managing and sharing the vocabulary specific to your enterprise. They enable inventory, definition, classification and organization of business concepts to establish a pertinent link with technical objects implemented at the information system level.

At the business level, they offer business users a tool to describe the concepts they handle and the links that manage their organization. To do this, **MEGA** is based on widely used semantic Web principles, as well as ontological frameworks such as IDEAS or standard ISO 15926 (high level, life cycle and event type).

At the IS architecture level, **HOPEX Data Governance** and **HOPEX Information Architecture** offer features to establish correspondence between application data, based on UML formalism, and information described at the business level.

☛ For more details on the interface and functions of **HOPEX** in general, see [HOPEX Desktop](#)

Vocabulary Management Process

The definition of business information requires the creation of a business dictionary and the terms in this dictionary.

A term is the designation of a concept in a given language.

Example: the concept "Country" has the terms "Pays" in French and "Country" in English.

The concept carries the definition of the term. A term can be associated with several concepts, thus several definitions.

Example: the term "Order" can mean "Arrangement of elements in a set" or "Authoritative indication to be obeyed".

So that business users and IS users share a common vocabulary, **HOPEX Data Governance** and **HOPEX Information Architecture** are based on two major functions:

- The analysis and organization of business concepts,
- The relationship setting of business concepts with information system architecture elements.

Analysis and organization of business concepts

This is carried out by a business user. It consists of describing all business concepts, using a simple semantic model based on notions of concept, event and state.

- A concept, representing a business object, is characterized by:
 - its scope, ie. its relationships with other concepts

For example, a work is characterized by its author, its title, its publication date, etc.

- its inheritance links with other concepts

For example, a subscription is a book or media subscription.

- its occurrences

For example, Alexandre Dumas is an occurrence of Author.

- A State Concept enables identification of an evolution in time of a concept,

For example, a work is available or on loan.

- An Event represents a significant fact modifying the state of one or several concepts.

For example, publication of a work.

The "Data Architect" standard profile is provided to ensure business concept analysis and organization work.

Concept realization

Business concepts are generally implemented in the IS using the UML method and formalism.

The "Concept realization" work consists of connecting the data model elements with business concepts to:

- define more precisely objects handled at IS architecture level,
- assure improved vocabulary sharing and improved global communication between business users and IS users.

The Data Architect ensures the "concept realization" work.

See [Connecting the Business Concepts to the Logical and Physical architecture](#).

CONSULTING THE BUSINESS GLOSSARY



HOPEX Data Governance and **HOPEX Information Architecture** offer a tool for easy consultation of terms, from which you can generate a business glossary.

See:

- ✓ [Searching for Terms in the Business Glossary](#)
- ✓ [Generating a Glossary](#)
- ✓ [Using Data ID Card](#)

You can also initialize a business glossary from existing data. See [Initializing a Business Dictionary Using Logical or Physical Data](#).

The terms created can be classified in business dictionaries. The description of business dictionaries and all the construction elements of the business ontology enriches the glossaries. For more detailed information, see [The Elements of a Business Dictionary](#).

SEARCHING FOR TERMS IN THE BUSINESS GLOSSARY

You can search for terms in the business glossary of your repository.

A term is the name of a concept, concept property or other business information in a given language. These concepts or concept properties carry the definition of the term.

Example: the "Country" concept has the "Pays" in French and "Country" in English.

See also: [Concept and Term](#).

Prerequisite

The use of the glossary requires that the repository be indexed.

To index a repository, see "Enabling and Customizing Repository Indexing" in the Administrator guide.

Scope of the Search

The search for a term in the glossary looks at all the definitions in the glossary that contain the term in question, as well as the glossary items that are associated with the term.

For example, if "Booking" is associated with "Customer", when searching on "Customer", "Booking" will also appear in the results.

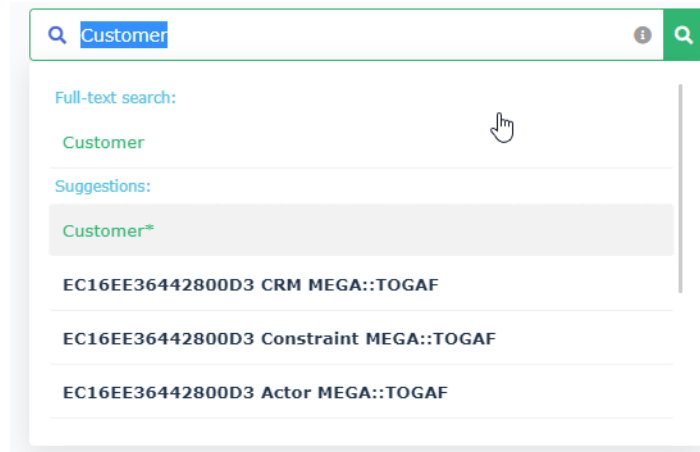
Starting the Search

To display a term definition:

1. Click the navigation menu then **Glossary > Search**.
2. In the edit area, in the **Search** field, enter the term or the first few letters of the search term to display a list of matching terms.

☛ *Used at the end of a word as in "Liberal*", the asterisk allows you to find the different suffixes like "Liberalism" or "Liberalization".*

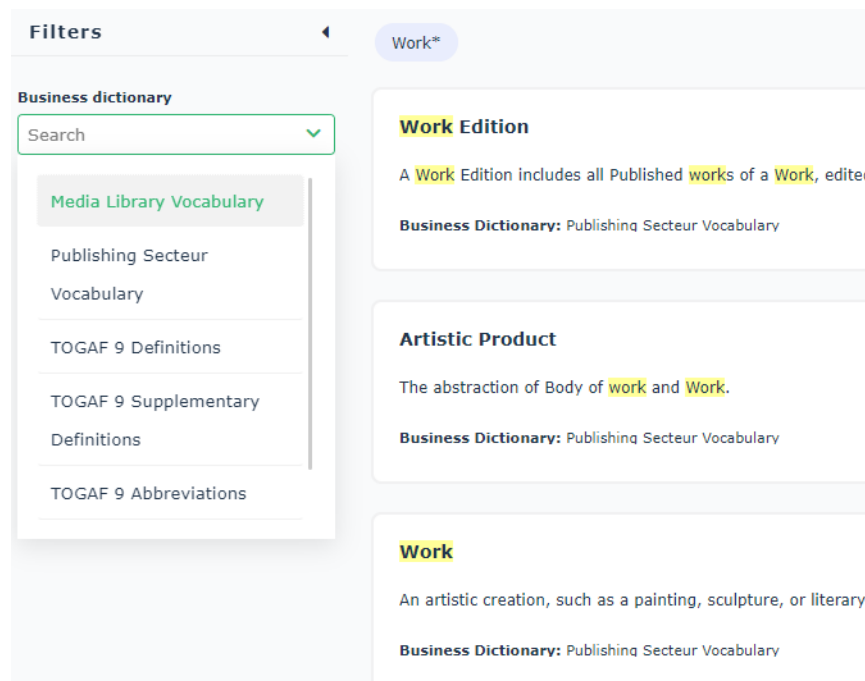
3. Select the term in question from the list and click the magnifying glass.



The term, if it exists, appears in the search results.

Search filters

You can refine the results using the filters in the left panel. The values proposed in the filters depend on the results obtained: for example, when the searched term appears in several business dictionaries, the filter allows you to restrict the search to one of the business dictionaries found.



When the filters are hidden, you can display them by clicking on the **Filters** button located on the right of the search bar.

Displaying the Details of a Term

To display the details of a term:

- 1 In the search results, click the term in question.
A window appears to the right of the search, with two tabs: **Standard** and **Advanced**.

The screenshot displays the 'Business Glossary' interface. At the top, there is a search bar with the text 'Customer' and a 'Filters' button. Below the search bar, a list of search results is shown. The first result is 'Boundaryless Information Flow' with a 'Update: Yesterday' tag and the business dictionary 'TOGAF 9 Definitions'. The second result is 'Insurance Customer' with a 'Update: Yesterday' tag and the business dictionary 'Insurance'. The third result is 'Customer' with a 'Update: Yesterday' tag and the business dictionary 'Vacation Booking'. To the right of the search results, a detailed view of the 'Customer' term is displayed. This view has two tabs: 'Overview' (selected) and 'Advanced'. The 'Overview' tab shows the following information:

- Responsibilities:** A table with two columns. The first column is 'Data Owner' with the value 'None'. The second column is 'Data Steward' with the value 'None'.
- Definition:** A text box containing the definition: 'a person or organization that buys goods or services from a store or business.'
- Type:** A table with two columns. The first column is 'Concept' with the value 'Concept'. The second column is 'Status' with the value 'None'.

Standard characteristics

The **Standard** tab displays:

- the Data Owner and the Data Steward
- the type of object, for example, Concept, Concept type, etc.
- the status in the data approval workflow. See [Data Validation Workflow](#).
- the definition
- synonyms, data category, business domain, etc.
- the related properties: for example, the concept "Customer" is associated with the concept "Booking". This is why the concept "Booking" appears in the result of the search for the term "Customer".

Related Properties

Name	Definition	Type	Presence	Cardinality
Booking		Concept	Always	*

- (in **HOPEX Data Governance**) the related metadata, which realizes the sought business data.
For each metadata are displayed:
 - the data catalog
 - the data source
 - the object type: Meta Dataset or Meta Field
 - the quality: the displayed value is the average of the different values of the last assessment performed on the metadata.
 - the related regulations

➡ For more information on realizations, see [Connecting the Business Concepts to the Logical and Physical architecture](#).

Advanced characteristics

The **Advanced** tab displays:

- Data lineages that contain the data. See [Describe a Data Lineage](#).
- Applications and processes that use the data. See [Use of Data by the Information System](#).
- Quality policies associated with the data See [Defining a Data Quality Policy](#).
- Regulations that deal with the data. See [Regulations and Business Rules](#).

GENERATING A GLOSSARY

HOPEX Data Governance and **HOPEX Information Architecture** provide a ready-to-use glossary report to automatically build the business glossary with terms derived from a set of Business dictionaries or, where appropriate, libraries. For each term, the glossary displays a list of associated definitions with their text, synonyms and components list.

Launching a Glossary Report

To launch a glossary report:

1. Click the navigation menu then **Reports > Description Reports**.
2. In the edit area, click the **Business Glossary Report** tile.
3. Select the source library or business dictionary.
 *You can select more than one.*
4. Refresh the report to display its content.

For more details, see [Glossary Report](#).

Using the Glossary in a Multilingual Context

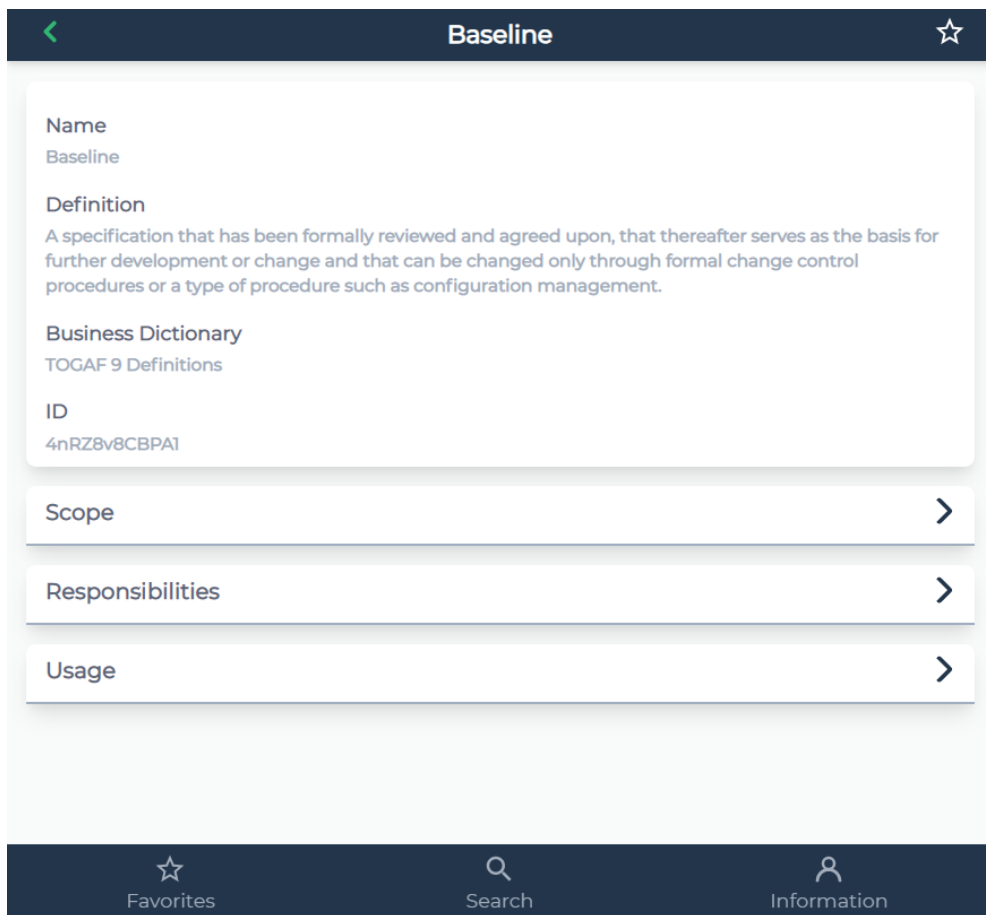
For more details, see "Using HOPEX in a Multilanguage Context" in the HOPEX Common Features guide.

USING DATA ID CARD

Data ID Card is an application that allows you to consult the contents of the HOPEX repository.

In Data ID Card you can:

- Search for a business data by entering a character string in the search tool. The application displays the name of the associated terms, with their main characteristics.
- Declare business data as favorites to facilitate their consultation.



The Data ID Card is available in the HAS Console. See [Importing a Module into HOPEX](#).



DEFINING BUSINESS INFORMATION



HOPEX Data Governance and **HOPEX Information Architecture** allow you to describe the architecture of your company's business information according to an approach whose various stages are described in this chapter.

- ✓ [Objects Used](#)
- ✓ [Presentation of Concept Modeling Diagrams](#)
- ✓ [Business Dictionary](#)
- ✓ [Concept Domain Map](#)
- ✓ [Concept Domain](#)
- ✓ [Concept](#)
- ✓ [Concept Components](#)
- ✓ [Concept Properties](#)
- ✓ [Concept Inheritances](#)
- ✓ [Concept structure diagram](#)
- ✓ [Individuals](#)
- ✓ [Concept or Individual States](#)
- ✓ [Concept Type](#)
- ✓ [Concept View](#)

OBJECTS USED

With **HOPEX Data Governance** and **HOPEX Information Architecture** you can create a business dictionary that describes and defines elements of your business vocabulary.

The basic component of a business dictionary is the **Concept**.



A concept expresses the essential nature of a being, an object, or a word through its properties and characteristics or its specific qualities.

The word that is associated with a **Concept** and which depends on language is a **Term**.



A term is a word or word group, that is used for a specific meaning in a specific context.

Concept and Term

A term is specific to a language and cannot be translated.

The same term in different languages can represent different concepts.

Example: the term "car" in English refers to a private car, while the same term in French represents a collective transport vehicle.

In the same language, the same term can represent several concepts and the meaning that is given to this term depends on its context of use.

For example, the word "ring" in English refers to a bell as well as a ring.

As a consequence, for the same language, the same **Term** can be connected to several concepts. Each concept gives a specific definition of this term in its Business dictionary.

As a consequence, with **HOPEX**, a concept carries the name of its associated term in the language chosen by the user. To modify the name of a concept in a given language, you must change the name of the associated term.

See also:

[Concept](#)

[Searching for Terms in the Business Glossary](#)

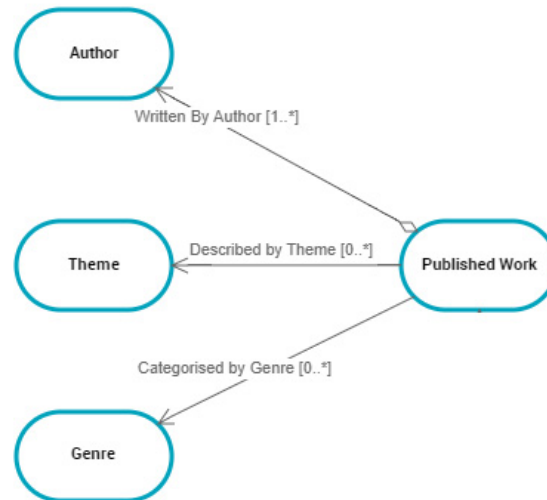
Links Between Concepts

To define semantics of a concept, you can draw several types of link between concepts: definition links or dependency links.

Definition links

Definition links enable characterization of a concept.

For example, a work is defined by its genre, its author, its theme.



A definition link is described by a **Concept Component**, which can, if appropriate, be associated with a term.

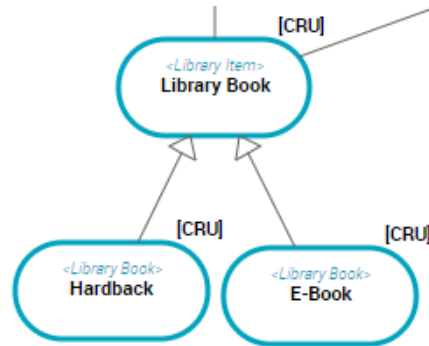
 A concept component enables representation of a dependency relationship between two concepts. This relationship is directional.

 For more details, see [Concept Components](#).

Dependency links

Certain business concepts are versions of other concepts; they inherit the same concept components.

For example, the "Library Book" concept is broken down into "Hardback" and "E-Book". These two book types inherit the links at the level of the "Library Book" concept.



This relationship is described by a **Variation**.

 A variation describes how a concept can be varied under another form. The variant is an object similar to the varied object, but with properties or relationships that may differ.

➡ For more details on variations, see the **HOPEX Common Features** guide, "Handling Repository Objects", "Object Variations".

A **Variation** can also be created between two **Concept Components**.

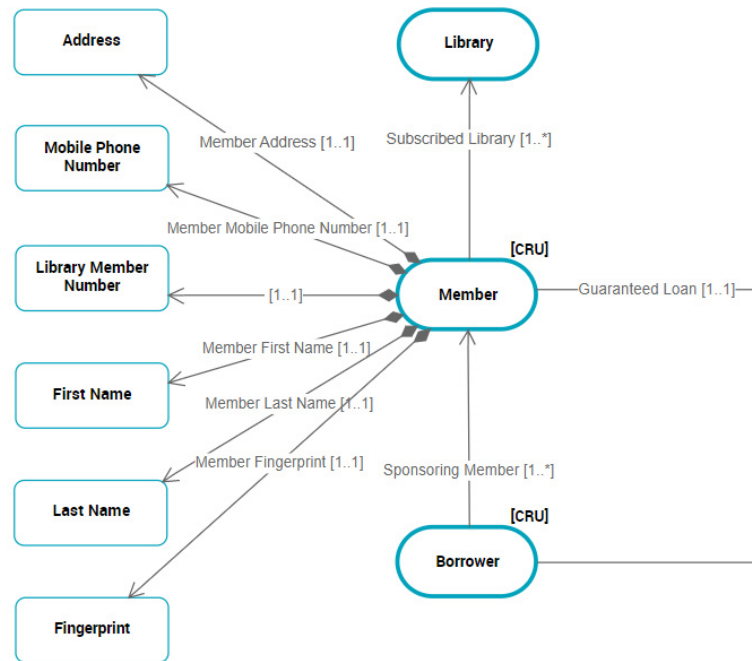
For example, the "Subscriber" is also a "Member".

➡ For more details, see [Concept Inheritances](#).

Concept Properties

To describe the characteristics associated with a concept, you can link a concept to concept Properties.

For example, a person ("Member") is associated with a mandatory and unique postal address, a first name, last name, telephone number, etc.



The link between a concept and a concept property is described by a **Sub-property** that can, if necessary, be associated with a term.

A concept property component enables specification of the relationship between two concept properties.

A sub-property is used to specify the relationship between a concept and a concept property.

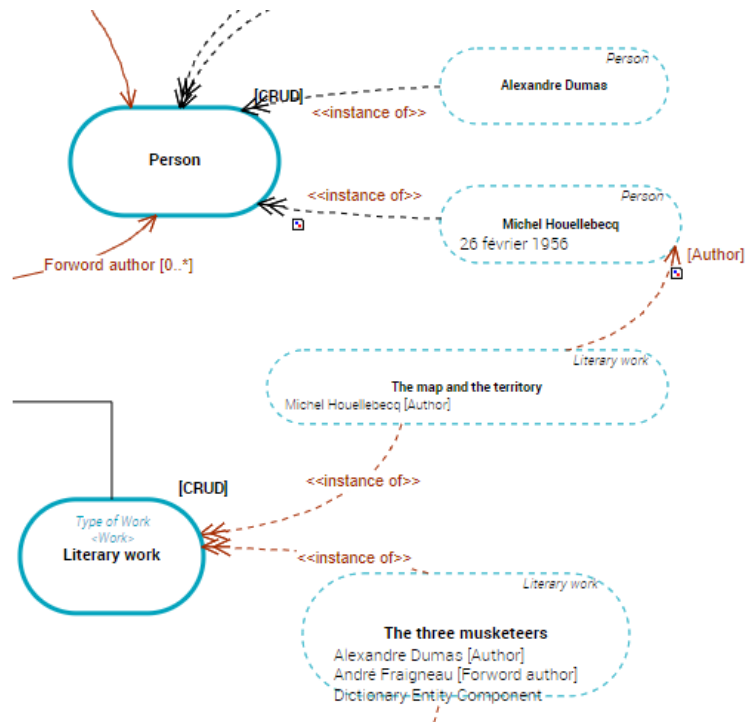
For more details, see [Concept Properties](#).

Concept Instances: Individuals

To validate the semantic model created from concepts, you can define concept instances, ie. real objects.

In this way you can create your semantic model using two approaches: either from real objects to deduce concepts, or from concepts to subsequently introduce real objects.

For example, "Michel Houellebecq" is an instance of "Person" and "The map and the territory" is an instance of "Literary Work".



A concept occurrence is an **Individual**.

An individual represents the occurrence of a concept.

The relationship between a concept and its occurrences is described by an **Individual Classification**.

An individual classification is used to connect an individual to the concept that characterizes it.

You can also connect two individuals with a **Dictionary Entity Component** relationship type.

An entity component is used to connect an individual to a dictionary element.

It is then possible to specify that "Asimov" is the author of the work "The Robots".

It is not possible to describe variations between individuals or between individuals' classifications.

For more details, see [Individuals](#).

The Life Cycle of a Concept or Individual

to take into account the evolution over time of business concepts, you have two particular concepts:

- The **State Concept**, which enables identification of an evolution in time of a concept,

 A state concept is a situation in a concept life cycle during which it satisfies certain conditions, executes a certain activity or waits for a concept event. A state concept represents a time interval of which limits are two concept events. A state concept is a phase through which the concept passes during its life cycle.

- The **Event Concept**, which represents a significant fact modifying the state of one or of several concepts.

 An event concept represents an event occurring during concept life, for example a change of season. An event concept marks the impact on a concept of a phenomenon internal or external to the concept. Concept events can be distinguished as concept start events, end events and intermediate events.

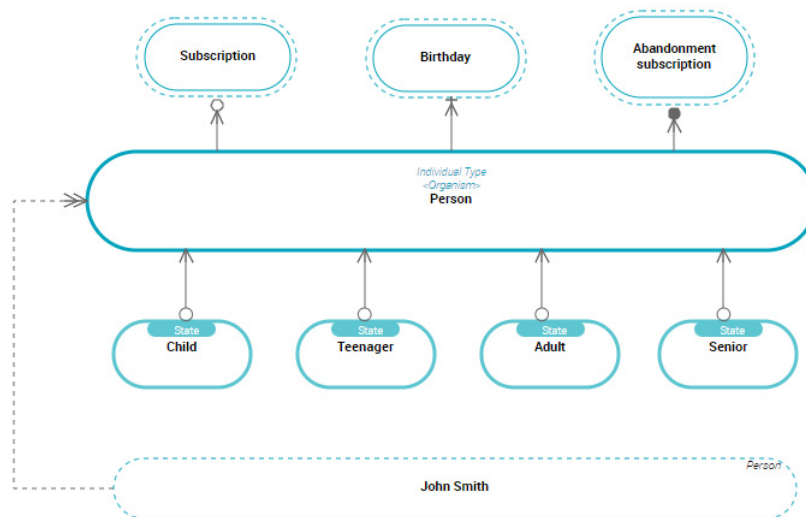
State Concepts and **Event Concepts** can be described in the same way as any other concept.

Concept life cycle

The same business concept can take several states.

For example, the same subscription holder can pass from "Child" state to "Adolescent" state, then to "Adult" state and finally "Senior".

Passage from one state to another can be connected to a event, a "Birthday" for example.



The relationship between a concept and its **State Concept** is described by a **Dictionary State Of**.

 A dictionary state enables connection of a concept to a concept state, and specification of the state nature.

The relationship between a concept and its **Event Concept** is described by:

- a **Start Event**,
- an **End Event**,
- or an **Intermediate Event**.

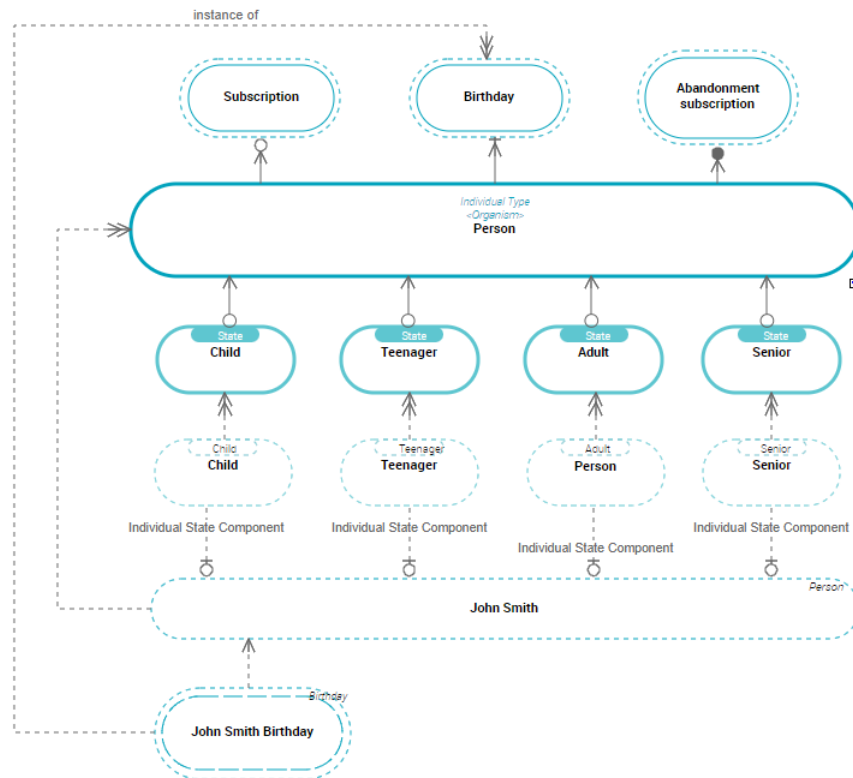
➡ For more details, see [Concept or Individual States](#).

Individual life cycle

➡ For more details, see [Describing Individual States and Events](#).

If a concept is associated with states and events, occurrences of this concept can also be associated with events and states.

For example, "John Smith" is a "Person" who can pass from one state to another on his birthday.



To represent the individual state notion, **HOPEX Information Architecture** proposes the **Individual State**.

 An individual state is an instance of a concept state to which the dictionary state is connected. It represents an individual state during its life cycle.

The relationship between an individual and its **Individual State** is described by an **Individual State Component**.

 An individual state component is used to connect an individual to an individual state.

In addition, the switch from one individual state to another can be conditioned by an **Individual Event**.

 An individual event represents an event occurring during the life of the individual. It is an instance of an event concept of the concept to which the individual is connected.

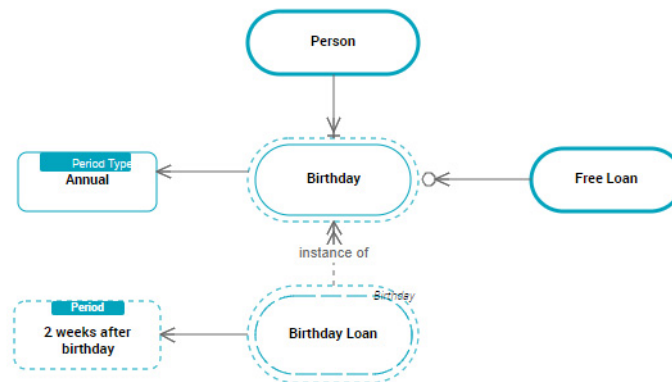
The relationship between an individual and its **Individual Event** is described by a **Entity Component**.

 An entity component is used to connect an individual to a dictionary element.

Periods

Periods are used to add time-related information to events.

For example, a free loan may be offered to subscribers on each anniversary. This annual loan is valid for a period of two weeks.



A **Period type** is connected to an **Event concept**.

 An event concept represents an event occurring during concept life, for example a change of season. An event concept marks the impact on a concept of a phenomenon internal or external to the concept. Concept events can be distinguished as concept start events, end events and intermediate events.

The **Period** is connected to an **Individual event**.



An individual event represents an event occurring during the life of the individual. It is an instance of an event concept of the concept to which the individual is connected.

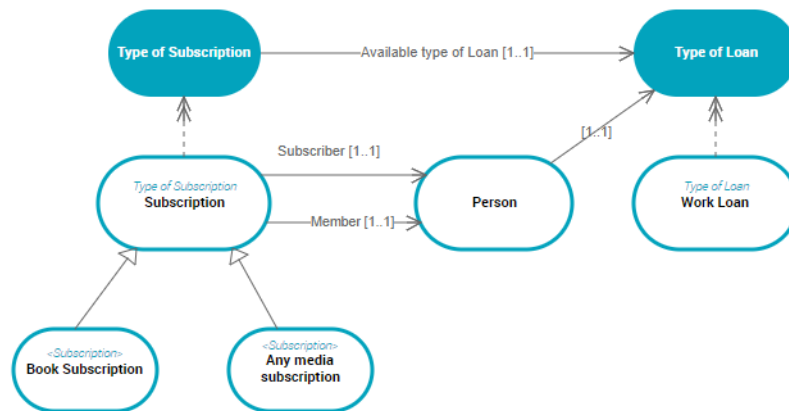


For more details, see [Using periods](#).

Classifying Concepts and the Concept Type Notion

A concept type enables classification of concepts. Relationships between concept types are represented by concept type components.

For example, "Subscriptions" can be classified by "Subscription Type". A "Subscription Type" being characterized by a "Loan Type".



HOPEX Data Governance And **HOPEX Information Architecture** offer features to create the following relationships:

- the relationship between two **Concept Types** is described by a **Concept Type Component**.

For example, a "Subscription Type" is characterized by an "Available Loan Type".



A concept type component enables specification of the relationship between two concept types.

- The relationship between a **Concept Type** and a Concept Type is described by a **Concept Classification**.

For example, all "Subscriptions" must correspond to a "Subscription Type".



A concept classification enables connection of a concept to the concept that characterizes it.

- The relationship between a concept and a **Concept Type** is described by a **Concept Power Component**.

For example, each member "Person" could be characterized by a "Loan Type".



A concept power component enables connection of a concept to concept type to characterize a property of the concept.

The Concept View

A concept view enables representation of the semantic scope covered by a business object. A concept view is based on the selection of several concepts specific to the view.

From a start concept linked to the business object you wish to describe, you browse the semantic links that define it. In this way you identify several concepts that define the described object in a particular context.

☛ You can create different views for the same business object.

☛ For more details, see [Concept View](#).

Dictionary Element Realization

Through **Realizations**, you can ensure consistency between the objects that make up your organizational and technical repository, on the one hand, and the business concepts that make up your dictionary, on the other.



A realization of concept connects a technical or organizational object of the repository to a dictionary element.

For more details on realizations, see the HOPEX IA - Logical Layer documentation.

For more details on generating the dictionary, see [Glossary Report](#).

PRESENTATION OF CONCEPT MODELING DIAGRAMS

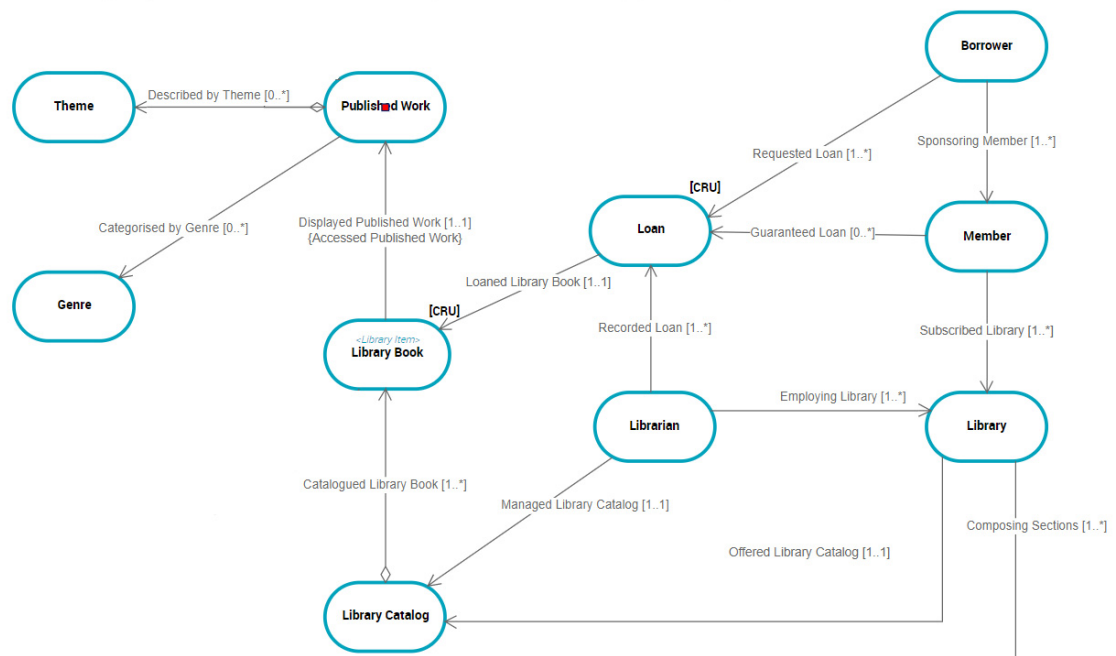
For business data definition, **HOPEX Data Governance** and **HOPEX Information Architecture** provide different types of diagram.

Concept Diagram

A concept domain provides a partial view of ontological models for the business information. It is described by a concept diagram presenting concepts, their components, super-types and links.

Link direction provides a natural mechanism of reading and deducing the scope defining the "business object".

The following concept domain diagram shows a partial view of the "Library" Business dictionary.



➡ For more details, see [Building a Concept Diagram](#).

Concept structure diagram

The content of business objects can be represented in a "Concept Structure Diagram", which can be initialized from concept diagram elements.

➤ For more details, see [Concept structure diagram](#).

Concept type structure diagram

Concept types can be represented in a "Concept Type Structure Diagram", which can be initialized from concept graph elements.

➤ For more details, see [The Concept Type Structure Diagram](#).

State concept state structure diagram

State concept states can be represented in a "State Concept Structure Diagram", which can be initialized from concept graph elements.

➤ For more details, see [State Concept Structure Diagram](#).

Individual structure diagram

The individual structure diagram describes the internal structure of the concept instance and the links between all components. This diagram can be initialized from concept graph elements.

➤ For more details, see [Individual Structure Diagram](#).

The concept life cycle structure diagram

The concept life cycle structure diagram is used to describe the sequence of state concepts operating during the concept life cycle. Each state concept, which can be considered as point in time, is followed by other state concepts.

Passage from one state to another is modeled by a transition.

➤ For more details, see [Concept life cycle structure diagram](#).

BUSINESS DICTIONARY

A business dictionary collects and structures a set of concepts that expresses the knowledge of a particular area.

Example of dictionary: medical ontology

You can break down a business dictionary into concept domains.

Examples of concept domain: psychology, pediatrics.

See [Concept Domain](#).

Business dictionaries can be created with the **Data Architect** profile.

The Elements of a Business Dictionary

A business dictionary is used to describe all the elements defining your information architecture:

- Concepts



A concept expresses the essential nature of a being, an object, or a word through its properties and characteristics or its specific qualities.



For more details, see [Concept](#).

- Terms



A term is a word or word group, that is used for a specific meaning in a specific context.



For more details, see [Searching for Terms in the Business Glossary](#).

- Concept variations



A variation describes how a concept can be varied under another form. The variant is an object similar to the varied object, but with properties or relationships that may differ.



For more details, see [Concept Components](#).

- Concept types



A concept type enables classification of concepts. Relationships between concept types are represented by concept type components.



For more details, see [Concept Type](#).

- State concepts



A state concept is a situation in a concept life cycle during which it satisfies certain conditions, executes a certain activity or waits for a concept event. A state concept represents a time interval of which limits are two concept events. A state concept is a phase through which the concept passes during its life cycle.



For more details, see [Concept or Individual States](#).

- Event concepts



An event concept represents an event occurring during concept life, for example a change of season. An event concept marks the impact on a concept of a phenomenon internal or external to the concept. Concept

events can be distinguished as concept start events, end events and intermediate events.

☛ For more details, see [Describing Event Concepts](#).

- Individuals



An individual represents the occurrence of a concept.

☛ For more details, see [Individuals](#).

- Individual states



An individual state is an instance of a concept state to which the dictionary state is connected. It represents an individual state during its life cycle.

☛ For more details, see [Concept or Individual States](#).

A business dictionary can be completely or partially described by a concept diagram.

☛ For more details on environment components, see [Presentation of Concept Modeling Diagrams](#).

Accessing the elements of a business dictionary

To access the elements of a business dictionary in **HOPEX Data Governance** :

1. Click the navigation menu then **Architecture > Hierarchy View**.
2. In the edit area, expand the folder of the business dictionary that interests you.

The list of concepts and terms appears in the dedicated folders.

☛ Concepts carry the name of the term associated with the concept in the data language. For more details, see [Using the Glossary in a Multilingual Context](#).

If you expand the folder associated with a concept, the terms and synonyms are accessible in all languages available in your environment **HOPEX**.

☛ The number of languages proposed from folders depends on your **HOPEX** environment. To configure the list of languages, see the **HOPEX Power Supervisor** guide, chapter "Managing Options", "Managing Languages", "Installing Additional Languages".

☛ The list of elements of a business dictionary is also accessible in the dictionary properties window, in the **Characteristics** page, **Business information** section.

Importing business information

You can import existing business information into your repository using an Excel file. See [Importing Business Data from an Excel File](#).

Work Business Dictionary

When initializing business data, a business dictionary is created by default to store the created data and define its owner.

See [Initializing a Business Dictionary Using Logical or Physical Data](#).

Creating a Business Dictionary

To create a business dictionary:

1. Click the navigation menu then **Glossary > Dictionaries**.
2. In the edit area, right-click the **Business Dictionaries** folder and click **New > Business Dictionary**.
3. Enter the name of the dictionary and click **OK**.

Initializing a Business Dictionary Using Logical or Physical Data

Initialize a business dictionary consists of creating, in a business dictionary, the concepts that correspond to logical or physical data, and thus creating a realization link between the business concepts and the logical or physical data in question.

A realization report allows you to view these realization links.

There are three ways to initialize a business dictionary:

- from a logical data dictionary (a data package)
- from a physical data dictionary (database)
- from a metadata inventory (data catalog)
- when creating logical or physical data: an automatic initialization option allows you, when you create logical or physical data, to automatically create the corresponding concepts in the working dictionary.

Initializing a Business Dictionary Using Logical Data

To initialize a dictionary using logical data:

1. Click the navigation menu then **Tools > Initialize Business Dictionary**.
2. Click the **Logical Data** tile.
A wizard opens.
3. Under **Options**, specify whether the attributes become Concept Properties or Concept Property Components.
4. Under **Source selection**, click the **Add** button and select the source package.
5. Once the package is selected, click **Next**.
The wizard displays the name of the corresponding business dictionary.
6. Click **OK**.
The business dictionary is created. It appears in the list of working business dictionaries in the repository.

Mappings between logical data and business objects

Logical data	Business Objects
Class	Concept
Part	Concept component
Attribute	Concept Property Concept Property component

Initializing a Business Dictionary Using Physical Data

To initialize a dictionary using physical data:

1. Click the navigation menu then **Tools > Initialize Business Dictionary**.
2. Click the **Physical Data** tile.
3. In the wizard that appears, under **Source selection**, select the source database.
4. Once the database is selected, click **Next**.
The wizard displays the name of the corresponding business dictionary.
5. Click **OK**.
The business dictionary is created. It appears in the list of working business dictionaries in the repository.

Mappings between physical data and business objects

Physical data	Business Objects
Table	Concept
Column	Concept Property

Initializing a Business Dictionary from Meta Datasets

To initialize a dictionary from meta datasets:

1. Click the navigation menu then **Tools > Initialize Business Dictionary**.
2. Click the **Meta Data** tile.
3. In the wizard that appears, under **Source selection**, select the source deployed data store.
4. Click **Next**.
The wizard displays the name of the corresponding business dictionary.
5. Click **OK**.
The business dictionary is created. It appears in the list of working business dictionaries in the repository.

Mappings between meta datasets and business objects

Meta datasets	Business Objects
Meta dataset	Concept
Meta data field	Concept property

Initializing a Business Dictionary when Creating Logical or Physical Data

By default, when you create logical or physical data, the corresponding business concepts are automatically created and associated with that data.

These concepts appear in a working business dictionary that has the same name as the data dictionary that holds the logical or physical data from which the concepts are derived.

This automatic initialization is defined by an option. You can modify the option in order to disable automatic creation of concepts or to create/reuse concepts in other dictionaries.

Modifying the Initialization Option

To modify the initialization option:

1. Click the icon of the user profile and select **Settings > Options**. The options window appears.
2. In the left pane of the options window, select **HOPEX solutions > Information Architecture > Data Governance**.
3. In the right pane, under the **Business dictionary initialization** field, select the required automatic initialization value:
 - **Never**: disables automatic initialization of the business glossary.
 - **Work business dictionary**: enables automatic initialization of the business dictionary. The concepts are created automatically in the work business dictionary.
 - **All business dictionaries**: enables automatic initialization of the business dictionary. The concepts can be created or reused on all the business dictionaries.

Displaying the Realization Chart

You can use the realization graph to visualize by which architecture elements dictionary elements are implemented.

To access the report:

1. Click the navigation menu then **Reports > Description Reports**.
2. In the edit area, click **Realization Graph**.
3. Select the object on which the realization graph is based (see below "Report parameters") and refresh the report.

 The realization graph is also accessible in the **Reports** page of the concerned object.

Report parameters

This consists of defining report input data.

Parameter	Parameter type	Constraint
Object list	Org-unit Application Library Capability Class Concept State concept Event concept Concept type Content Exchange contract Concept life cycle Exchange Entity (DM) functionality Business function System process Functional process Business process Organizational process IT service Data view Concept view	One object is mandatory

Report example

The example below shows the objects that implement the "Purchase" concept.



CONCEPT DOMAIN MAP

A Concept Domain Map is a business information urbanization tool. It represents the concept domains of a business dictionary and their dependency links.



A concept domain is a sub-set of elements of a business dictionary that reduces the scope of a field.

Dependency links between business domains are automatically deduced from the objects used in each of the domains of the map, you don't have to create them.

For more details on links between business information, see [Overview of links between objects](#).

Concept Domain Maps can be created with the **Data Architect** profile.

Creating a Concept Domain Map

To create the concept domain map for a business dictionary:

1. Click the navigation menu then **Glossary > Business Data**.
2. In the edit area, click **Concept Domain Maps**.
3. Click **New**.

To create the diagram of the concept domain map:

1. Click the icon of the concept domain map and select **New > Diagram**.
2. Select **Concept Domain Map** and click **OK**.
The diagram appears in the edit area.

The Components of a Concept Domain Map

You can add internal and external components to a concept domain map.

Internal components are concept domains that are part of the scope of the map (whether or not they belong to the owner business dictionary).

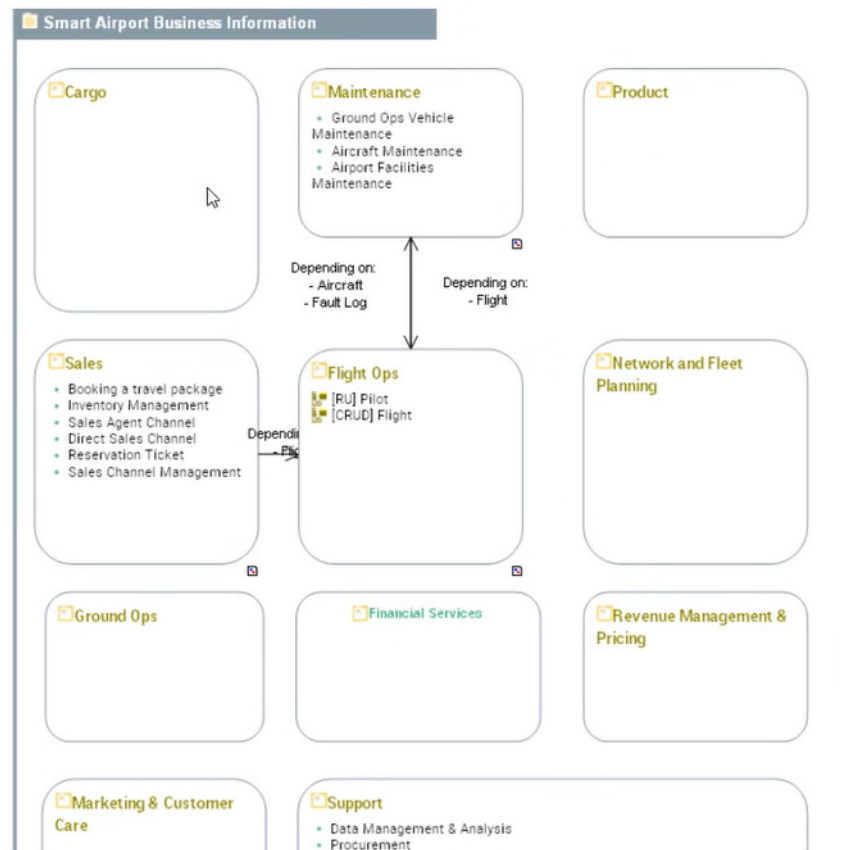
The external components are those used in the map but that are not part of the scope analyzed.

To add a component to a concept domain map:

1. Open the properties of the concept domain map in question.
2. Click the **Components** page.
3. Select the **Internal Components** or **External Components** tab depending on the type of component to be added, and click **New**.
A wizard opens. You can create a concept domain or link an existing concept domain.

Example of a Concept Domain Map

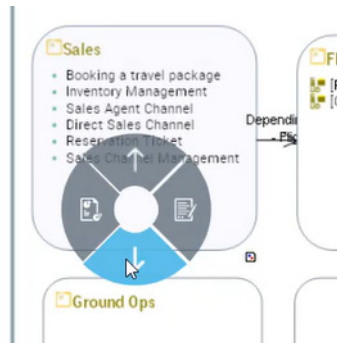
Below is the concept domain map of an airport. Links between the concept domains indicate the dependencies that exist between concepts in these domains.



From this map you can access the details of a concept domain and see the diagram that describes it, if applicable.

You can access the details of a concept domain:

- using the pie menu in a diagram preview.



- or the **pop-up menu** of the domain when you are in the edit mode.

Reports Available on a Concept Domain Map

In the properties of a concept domain map, reports allow you to visualize:

- The hierarchy of concept domains in a map, and whether these domains use sensitive or reference data. For more details, see [Data Domain Map](#).
- dependencies between concept domains of the map. See [Data Domain Dependencies](#).
- the architecture elements that implement the information of the map. See [Displaying the Realization Chart](#).
- Use of information of a concept domain map. See [Use of information of an information map](#).

CONCEPT DOMAIN

A concept domain is a sub-set of elements of a business dictionary that reduces the scope of a field. A concept domain is described in a concept diagram.

Concept Domains can be created with the **Data Architect** profile.

Creating a Concept Domain

To create a concept domain:

1. Click the navigation menu then **Glossary > Business Data**.
2. In the edit area, click **Concept Domain**.
3. Click **New**.

Creating the Structure Diagram for a Concept Domain

A structure diagram defines the sub-domains of the concept domains and their relationships.

To create the structure diagram of a concept domain:

1. Right-click the concept domain and select **New > Diagram**.
2. Select **Concept Domain Structure Diagram**.
3. Click **OK**.

The structure diagram associated with the concept domain opens in the edit window.

Building a Concept Diagram

A concept diagram is a graphical representation of the concepts used in the context of a concept domain, as well as the links that exist between these concepts.

A concept domain can be described by a number of concept diagrams.

A conceptual object belongs to a business dictionary from which it was created but can be used/referenced by a concept domain of a different business dictionary.

See also [Concept Domain](#).

Creating a concept diagram of a concept domain

To create the concept diagram of a concept domain:

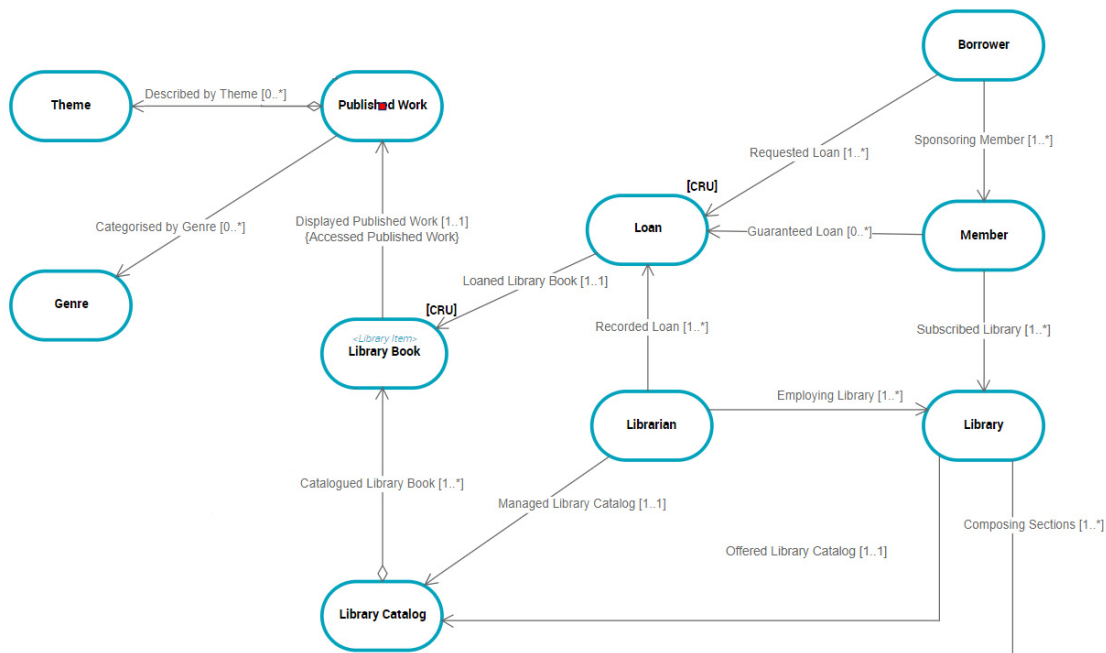
1. Right-click the concept domain and select **New > Diagram**.
2. Select **Concept Domain Diagram**.
3. Click **OK**.

The concept diagram opens in the Edit window.

The components of a concept diagram

A concept diagram describes the information architecture. By default, you see in the concept diagram concepts, variations and individuals only.

The following concept diagram partially describes the "Media Library" business dictionary.



Concept graph diagram with standard views

Activating the views window

The **Views and Details** window presents an extended list of views (object types to be displayed).

To activate the **Views and Details** window:

1. In a diagram, click  **Views and Details**.
The list of views (object types to be displayed) appears.
2. Select or clear the views you want to display or not.

The views available for a concept domain are:

- Concepts,
- Concept types,
- State concepts,
- Event concepts,
- Individuals,
- Individual states,
- Individual events,
- Concept Views



A concept view enables representation of the semantic scope covered by a business object. A concept view is based on the selection of several concepts specific to the view.

☞ For more details, see [Concept View](#).

Adding a concept diagram element

For example, to add an existing concept to a concept domain:

1. In the concept diagram object toolbar, click  **Concept**.
2. Click in the diagram.
The add concept dialog box opens and asks you to select a concept.
3. Select the concept that interests you.
4. Click **Add**.
The concept appears in the diagram.

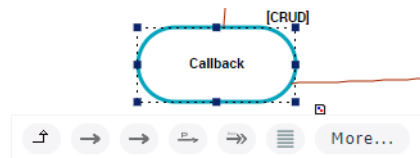
☞ For more details on concept creation, see [Concept](#).

Using the object insert toolbar

An insert toolbar available on each object simplifies object creation by proposing object selection help. It proposes only the objects that can be connected to the current object.

To create, for example, a concept from a diagram concept:

1. Click on the concept of the diagram that interests you.
A bar containing the objects you can insert appears.



2. Select the desired object type.

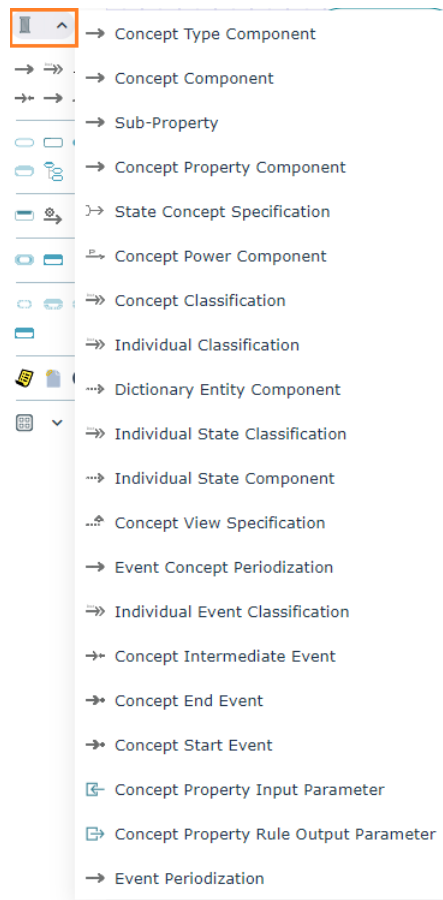
For example: **Concept Component**

3. Click in the graph at the point where you wish to place the object.
The object is created, with the link to the previous object.

Overview of links between objects

In each concept graph, relationships between concepts, concept types and concept individuals are represented by links.

The link direction provides a natural mechanism for reading and deducing the scope defining "the business object".



☛ For more details on accessing the properties of concept graph links, see [Accessing link properties in a concept diagram](#).

Link type	Definition and Comment
Concept Type Component	A concept type component enables specification of the relationship between two concept types.
Concept Component	A concept component enables representation of a dependency relationship between two concepts. This relationship is directional.
Sub-Property	

Link type	Definition and Comment
Concept Property Component	Component of a representation type such as lastname and firstname of a name.
State Concept Specification	Directed Relationship which states that a State Concept is a state of a Business Information Concept.
Concept Power Component	A concept power component enables connection of a concept to concept type to characterize a property of the concept.
Concept Classification	A concept classification enables connection of a concept to the concept that characterizes it.
Individual Classification	An individual classification is used to connect an individual to the concept that characterizes it.
Dictionary Entity Component	An entity component is used to connect an individual to a dictionary element.
Individual State Classification	An individual state classification enables connection of an individual state to the state concept that characterizes it. This link is available with "Individual State" view.
Individual State Component	An individual state component is used to connect an individual to an individual state. This link is available with "Individual State" view.
Concept View Specification	
Event Concept Periodization	
Individual Event Classification	An individual event classification is used to connect an individual to the event concept that characterizes it. This link is available with "Individual State" view.
Concept Intermediate Event	An event concept represents an event occurring during concept life, for example a change of season. An event concept marks the impact on a concept of a phenomenon internal or external to the concept. Concept events can be distinguished as concept start events, end events and intermediate events. These links are available with "Event Concept" view.
Concept End Event	
Concept Start Event	
Concept Property Input Parameter	
Concept Property Rule Output Parameter	
Event Periodization	

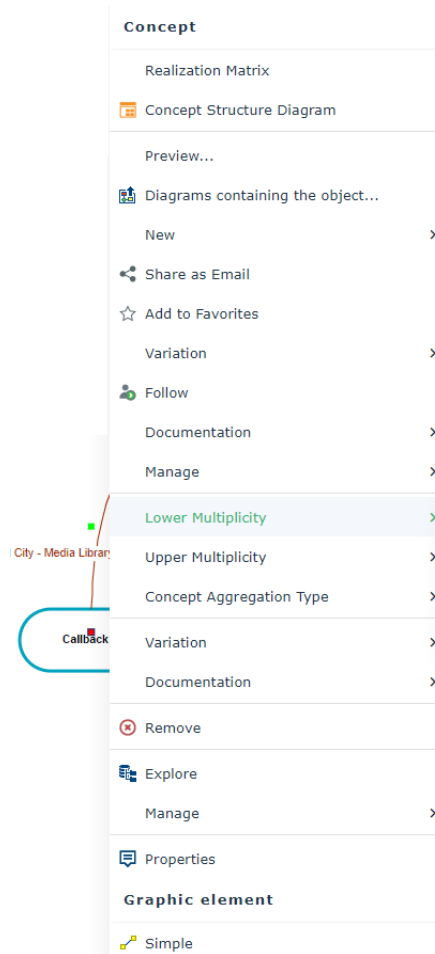
Accessing link properties in a concept diagram

In a concept diagram, links are directional and access the properties of both the link and the link target object.

➤ For more details on the list of links available in a concept domain, see [Overview of links between objects](#).

The pop-up menu of a **Concept Component** link type for example presents:

- commands specific to the object type used by the component (the concept)
- commands relating to the component itself
for example **Multiplicity**
- commands relating to the graphics.



To access properties of a link of "component" type:

for example **Concept Component**

1. Right-click the link to open its pop-up menu.
2. Select the link and click **Properties**.
The link properties dialog box opens.

In the **Characteristics** page of the link property window, the last **Component** section indicates:

- The **Name** of the link, which corresponds by default to the target dictionary element or term associated with the link.

 For more details on association of a term with a link, see [Concept Components](#).

- The **Composed Concept** targeted by the link.
- The **Owner** who is the dictionary element at the origin of the link.
- The **Minimum Multiplicity** is the number of origin elements that can access the same target elements.

For example, how many "Works" can belong to the same "Work Category".

- The **Maximum Multiplicity** is the number of target elements that can be connected to the same origin elements.

For example, a "Work" can only belong to only one "Work Category".

- The **Abstract Concept** check box, which enables specification of the concrete or abstract character of a concept,
- The **concept aggregation Type** which can be one of the following:
 - "Referencing": to indicate that the target concept is referenced by a link,
 - "Embedded": to indicate that the target concept exists in its own right, but is included in the concept that is the source of the link,
 - "Composite": to indicate that the target concept is a component of the concept that is the source of the link; if the target concept is destroyed, the composite is also destroyed.
- The **Designation** of the link and the **Definition Text** field enable association of a term and a definition to the link.

 For more details on association of a term with a link, see [Concept Components](#).

For more details on defining concepts, see [Concept](#).

Defining the Components of a Concept Domain

You can update your business dictionaries using [Concept Domains](#) and dictionary elements that already exist: [Term](#), [Concept](#), [State Concept](#), [Event Concept](#) or [Concept View](#).

The components of a concept domain

A concept domain includes or references a set of concepts or sub-domains.

You can display the elements that belong to the concept domain in the properties window of the domain in question, in the **Components** page.

Adding a concept to a concept domain

To add a concept to a concept domain:

1. Open the properties window of the concept domain.
2. Select the **Components** page.
3. Click **New**.
The business information creation wizard appears.
4. Select the object type "Concept" and enter its name.
5. Click **Next**.
The concept is added to the list of concept domain components.

☺ You can also drag the concepts in question from the hierarchical view of the business dictionary into the section.

Connecting or deleting a component from a concept diagram

To connect a dictionary element to a list of components for a concept domain:

1. Open the concept diagram associated with the concept domain.
2. Add the dictionary element that interests you in the diagram.
3. Right-click on this element to open its pop-up menu.
4. Select **Add to "Current concept domain name"**.
The element is added to the list of concept domain elements, in the **Component** page of the domain properties window. The shape changes in the diagram.

To delete a dictionary element from a concept domain:

1. Right-click the dictionary element concerned to open its pop-up menu.
2. Select **Delete from "Current concept domain name"**.
The element is deleted from the list of concept domain elements.

Defining the CRUD for the components of a concept domain

You can specify the access rights to each of the components of a concept domain through the CRUD of the component in question (Create, Read, Update Delete).

Artistic Work		
General ▼ Characteristics Domain Entities Relationships ▼		
New Reorganize Inherited Excluded		
☐	Local name ↑	Data Access
☐	Body of work	CRUD
☐	Cinematographic work	CRUD
☐	Literary work	CRUD
☐	Musical work	CRUD
☐	Person	CRUD
☐	Work	CRUD

To define the CRUD on a component of the concept domain:

1. Open the properties window of the concept domain.
2. Select the **Components** page.
3. Select the line of the component in question.
Commands are added, including the **CRUD** button.
4. Click this button.
5. In the window that appears, select or deselect the check boxes associated with the actions: Create, Read, Update, Delete.

The content of the **Data access** column is calculated automatically according to the selected actions. This result appears in object form in the concept diagram associated with the concept domain.

CONCEPT

The concept is the basic element of a business dictionary.

A concept expresses the essential nature of a being, an object, or a word through its properties and characteristics or its specific qualities.

A concept is associated with one or more terms that designate the concept in a given language. See [Searching for Terms in the Business Glossary](#).

Accessing the List of Concepts

To access concepts in **HOPEX Data Governance**:

1. Click the navigation menu then **Glossary > Business Data**.
2. In the edit area, click **Concepts**.
 Under **Architecture > Hierarchy View**, you can also display concepts from the relevant business dictionary.

To access concepts in **HOPEX Information Architecture**:

1. Click the navigation menu then **Glossary > Business Data**.
2. Unfold the relevant business dictionary folder to view its concepts.

Creating Concepts

To create a *concept*:

1. Click the **New** button associated with the concept list or the concept folder.
2. Enter the **Name** of the concept and the **Owner Business Dictionary**.
3. The **Existing Terms** section lists terms with the same name as the new concept. You can choose to use an already existing term, or create a new term.

 A term is a word or word group, that is used for a specific meaning in a specific context.

 If a term has already been created with the same name as the new concept, this term is automatically connected and appears automatically in the **Term** section.

4. In the **Description** field, enter the text of the concept definition.
5. Click **Next** to associate an image with the concept or **OK** to finish.
The name of the new concept appears in the tree. It also appears in the tree structure of the holding business dictionary. A new term with the same name as the concept is also created.

Concepts and Terms

Connecting an existing concept to a term

The same term can be connected to several objects. You can connect a term when creating a concept or at a later date.

To connect a term to a concept:

1. Open the properties of the concept.
2. In the properties, select the **Characteristics** page.
3. Expand the **Identification** section.
4. Click **Designated Term** then **Connect**.
5. Select the desired term.

You can also connect a concept to a term in the term properties:

1. Open the properties of the term.
2. Click the **Characteristics** page.
The list of objects connected to the term appears in the **Identified Dictionary Type** field.

 *Objects for which the term is declared as a synonym do not appear in the properties dialog box.*

3. Click the **Connect** button.
The query dialog box appears.
4. Select the "Concept" object type and click the **Find** button.
The list of the existing concepts appears.
5. Select the desired concept and click **Connect**.

Creating terms in multiple languages from a concept

You can associate terms with a concept for each of the data languages in your environment.

To create a term from a concept:

1. Open the properties of the concept that interests you and select the **Characteristics > Characteristics** page.
2. Expand the **Identification** section and click **New**.
A term creation dialog box opens.
3. Specify the **Local Name** of the term.
4. Select the **Language** and click **OK**.
The new term appears in the concept properties.

Creating synonyms in multiple languages



A synonym is a term interchangeable with another term in the context of a concept of this term that has the same or almost the same meaning.

It is possible to add synonyms to a concept in several languages. This function serves to indicate to the user that a concept defined and used in a certain context corresponds to other synonyms in another language.

Concept Properties

To access concept properties:

- 1 Select the concept concerned and click the **Properties**  button in the edit area.

Characteristics

The **Characteristics** page of the concept properties window provides access to the main characteristics of the concept.

A concept is described by:

- the **Abstract Concept** check box, which enables specification of the concrete or abstract character of a concept
- its **Description**, i.e. the text of its definition
- its designation which is represented by one or more **Terms**

 To modify the name of a concept in the corresponding language, you must access concept properties and modify the name of the term in the specific language. For more details See also [Searching for Terms in the Business Glossary](#).

- Its **synonyms**, **Hypernyms** and **Hyponyms**

For example, in the Financial domain, the term "Advance" is recognized as a synonym of "Down payment".

 A synonym is a term interchangeable with another term in the context of a concept of this term that has the same or almost the same meaning.

Components

The **Components** page presents:

- the list of concept components owned, for more details see [Concept Components](#).
- the list of concept power components, for more details see [Describing Concept Power Components](#).

 State concepts connected to a concept are not present in the properties dialog box, for more details see [Concept or Individual States](#).

Super types

The **Relationships > Super-Type** page presents concepts whose properties are inherited by the described concept, for more details see [Concept Inheritances](#) t

 For more details, see [Concept Type](#).

Realizations

The **Relationships > Realization** page allows to define the objects that the concept implements. It can be one of two types of implementation:

- A regulatory requirement: a policy framework, a business policy, a quality dimension, etc.
- A business information realization.

➡ For more details, see [Connecting the Business Concepts to the Logical and Physical architecture](#).

Regulations

The **Regulations** page allows you to define which internal or external regulations apply to the object in question.

When a concept belongs to a data category covered by a regulation, the name of the regulation appears on this page.

To connect a regulation to the concept:

1. Click on the **Regulations** page.
2. Click on the **Connect** button.
A window appears.
3. Indicate the type of regulation you want to link (Regulatory Framework, Business Policy, etc.).
4. From the list that appears, select the regulation in question.

➡ For more details on regulations, see [Rules and Regulations](#).

Data Quality

In the **Data Quality** page, you can indicate whether a quality policy applies to the object in question.

➡ For more details, see [Data Quality](#).

Reporting

Different types of reports are available on the concept:

- **Data Dendrogram**: illustrates the environment of the concept.
- **Information usage and processing graph report**
- **Realization graph report**: shows by which architecture elements the concept is implemented or the information elements the concept realizes.
- **Use of information**. The report displays:
 - the domains that use the concept, with which access rights
 - in which systems (applications, application systems, application service or microservice) these domains are used and by which components of these systems, specifying the access mode (read or read/write).

For more details on data reports, see [Data Reports and Analysis Tools](#).

Workflows

The workflow page allows you to launch the data-specific design workflow on the concept.

See also [Data Validation Workflow](#).

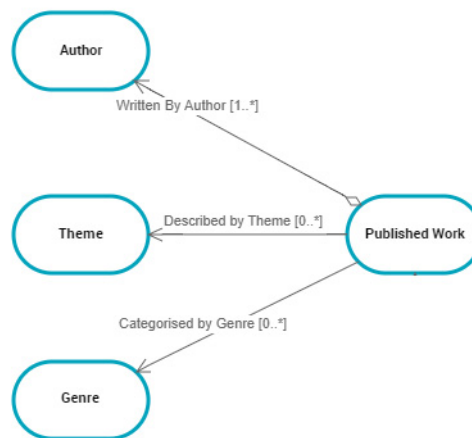
CONCEPT COMPONENTS

A concept can be connected to another concept to characterize it.

For example, the "Work" concept is connected to the "Person" concept to characterize the "Author" of a work.

This relationship is described by a **Concept Component**, which can be associated with a term.

 A concept component enables representation of a dependency relationship between two concepts. This relationship is directional.



Accessing Concept Components

To access concept components

1. Open the properties of a concept.
2. Click the **Characteristics** page.
3. Expand the **Components** section.
The list of components owned appears.

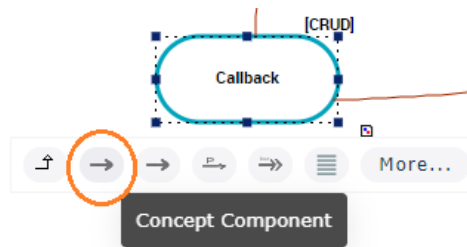
 You can also view the list of components of a concept in its concept structure diagram. For more details, see [Concept structure diagram](#).

Creating a Concept Component from a Diagram

The procedure for creating the "Author" concept component between "Work" and "Person" concepts is described as an example.

To create a concept component between two concepts of a concept domain:

1. In the concept diagram associated with the concept domain, click on the concept that holds the link, for example "Work".
A bar containing the objects you can insert appears.
2. Click on the icon that represents the **Concept component**.



3. Slide the cursor to the target concept, for example "Person".
4. When the cursor becomes a double chain link, release the mouse button.
The Concept Component creation wizard opens.
5. Enter a **Name**, for example "Author".
6. Given that the term "Author" must be created, select the "Creation with term" check box.

In the section **Term** appears in the creation creation dialog box.

A term is a word or word group, that is used for a specific meaning in a specific context.

7. In the **Definition Text** field, enter the text of the Concept Component definition and click **OK**.

The concept component appears in the graph.

A new term with the same name as the concept component is also created.

You can also create a concept component in a concept structure diagram. In this case, you must specify the target concept in the concept component creation wizard.

For more details, see [Concept structure diagram](#).

Describing Concept Power Components

Just as a **Concept** can be characterized by a link to another concept, a concept can also be characterized by a link to a **Concept Type**.



A concept type enables classification of concepts. Relationships between concept types are represented by concept type components.

For example, each member "Person" could be characterized by a "Loan Type".



For more details, see [Concept Type](#).

The relationship between a **Concept** and a **Concept Type** is described by a **Concept Power Component**.



A concept power component enables connection of a concept to concept type to characterize a property of the concept.

To create a **Concept Power Component** between a concept and a concept type in a concept domain diagram:

1. In the insert toolbar, click the **Link** button.
2. Click the concept that owns the link.

For example, "Person"

3. Click the target concept type.

For example, "Loan Type".

The Concept Power Component creation wizard opens.

4. Specify the **Local Name**.
5. If no term is to be created, select the "Creation without term" check box.
6. Click **OK**.

The Concept Power Component appears in the diagram.

Describing a Computed Concept Component

See [Calculation Rule on a Concept Component](#).

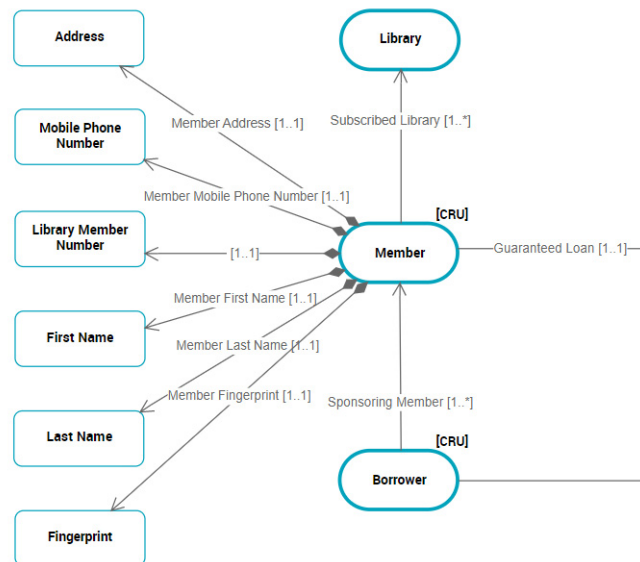
CONCEPT PROPERTIES

Concept Properties are used to define the characteristics associated with a concept.

For example, a person is associated with a mandatory and unique postal address, possibly an email address and one or more telephone numbers.

A **Concept Property** may itself be connected to other Concept Properties.

For example, the postal address is defined using the name of the street and the town.



Creating a Concept Property

Creating a Concept Property

To create a Concept Property:

1. In the concept diagram associated with the concept domain click the **Concept Property** icon of the insert toolbar.
2. Click in the diagram.
3. Indicate the name of the Concept Property and click **Add**.

Connecting a Concept Property to a concept

The link between a Concept and a Concept Property is described by a **Sub-Property** that can, if necessary, be associated with a term.



A sub-property is used to specify the relationship between a concept and a concept property.

Connecting two Concept Properties

The link between Concept Properties is described by a **Concept Property Component** that can, if necessary, be associated with a term.



A concept property component enables specification of the relationship between two concept properties.

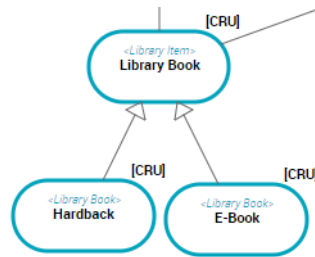
Creating a Computed Concept Property

See [Calculation Rule on Concepts](#).

CONCEPT INHERITANCES

Certain business concepts are versions of other concepts; they are characterized by the same concepts.

For example, the "Library Book" concept is broken down into "Hardback" and "E-Book". These two book types inherit the links at the level of the "Library Book" concept.



Accessing Concept Inheritances

Inheritances are represented by variations.

To access concept variations:

1. Open the properties of a concept.
2. Select the **Relationships > Super Types** page.
The list of variations associated with the concept appears.

Creating a Concept Inheritance from a Concept Diagram

You can specify that a concept inherits characteristics defined for another concept.

For example, the "E-Book" concept inherits from the "Library Book" concept.

To specify in a concept diagram that a concept inherits another concept:

1. In the insert toolbar, click the **Link** button.
2. Click the inheriting concept and drag the pointer to the root concept before releasing the mouse button.
The variation is represented by a link, but it is in fact a **HOPEX** object.
3. Specify the **Name** of the variation and click **Add**.
A directional link from the inheriting concept to the root concept appears.

Defining Inheritance of a Concept Component

An inheritance can also be created between two **Concept Components**.

For example, the "Subscriber" is also a "Member".

To define an inheritance between two concept components, they should be connected to the same concepts, either directly or via variations.

To create a variation between two concept components:

1. Open the properties of the concept component to be varied.
2. Select the **Variation** tab.
3. Click the **New** button.
The variation creation wizard opens.
4. Select the options:
 - "Initialization of attributes"
 - "Initialization of diagrams" so that the variation appears in diagrams.
5. Click **OK**.
The variation is created.

 A variation between **Concept Components** is represented graphically in a concept structure diagram. For more details, see [Concept structure diagram](#).

Creating a Concept Component Substitution

If, unlike a variation, a link is another definition of another link, you must create a **substitution**.

 A substitution determines which element can be used to replace another, or is effectively replaced by an element existing in a given context (for example in the context of a variation). Unlike a variation, a substitution does not involve inheritance but a functional equivalence.

 For more details on variations and substitutions, see the **HOPEX Common Features** guide, "Handling Repository Objects", "Object Variations".

To define a substitution between two concept components, they should be connected to the same concepts, either directly or via variations.

To create a substitution between two concept components from a concept structure diagram:

1. In the insert toolbar, click the **Substitution** button.
2. Click the component to be substituted and drag the pointer to the substituting component before releasing the mouse button.
3. Specify the **Name** and click **Add**.
A dotted line directional link from the component to be substituted to the substituting component appears.

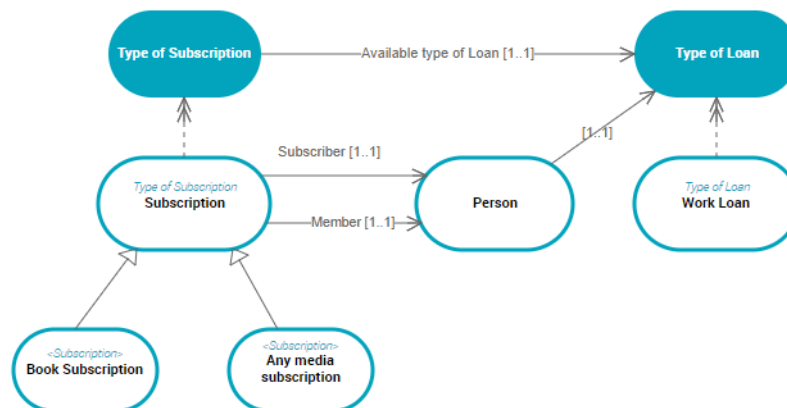
 The substitution is represented by a link, but it is in fact a **HOPEX** object.

CONCEPT STRUCTURE DIAGRAM

In **HOPEX Data Governance** and **HOPEX Information Architecture**, a concept structure diagram assembles all information relating to the concept. This diagram is initialized from concept diagram elements.

For example, "Subscriptions" can be classified by "Type of Subscription".

A "Type of Subscription" is characterized by a "Type of Loan".



The diagram includes:

- **variations** between components

For example, "Subscriptions" can be classified by "Subscription Type". A "Subscription Type" being characterized by a "Loan Type".

 A variation describes how a concept can be varied under another form. The variant is an object similar to the varied object, but with properties or relationships that may differ.

 For more details, see [Defining Inheritance of a Concept Component](#).

- **Substitutions** between components

 A substitution determines which element can be used to replace another, or is effectively replaced by an element existing in a given

context (for example in the context of a variation). Unlike a variation, a substitution does not involve inheritance but a functional equivalence.

☛ For more details, see [Creating a Concept Component Substitution](#).

- **Concept components** describing the relationship between two **Concepts**

For example, a "Subscription Type" is characterized by an "Available Loan Type".



A concept component enables representation of a dependency relationship between two concepts. This relationship is directional.

☛ For more details, see [Concept Components](#).

- **Concept power components** enabling concept characterization from **Concept Types**

For example, each member "Person" could be characterized by a "Loan Type".



A concept power component enables connection of a concept to concept type to characterize a property of the concept.

☛ For more details, see [Describing Concept Type Variations](#).

- **start events, intermediate events** and **end events** enabling definition of events contributing to change of state of a concept,

For example, the change of state of a member can be caused by a birthday.



An event concept represents an event occurring during concept life, for example a change of season. An event concept marks the impact on a concept of a phenomenon internal or external to the concept. Concept events can be distinguished as concept start events, end events and intermediate events.

☛ For more details, see [Describing State Concepts](#).

INDIVIDUALS

An individual represents the occurrence of a concept.

Accessing the List of Individuals

To access Individuals of your repository in **HOPEX Data Governance**:

1. Click the navigation menu then **Glossary > Dictionaries**.
 In **HOPEX Information Architecture**, click the navigation menu then **Business Glossary > Business Information Assets**.
2. In the edit area, expand the folder of the business dictionary that interests you.
3. Expand the **Individuals** folder.
 The list of individuals of the business dictionary appears.

Creating an Individual from a Business Dictionary

To create an individual from a business dictionary:

1. Right-click the business dictionary that interests you and click **New > Business Information Building Block**.
 The creation wizard opens.
2. Select the **Individual** object type.
3. Specify the **Local Name** and click **OK**.
4. In the **Individual Classification** section, you can click **New** to specify the concept to which the dictionary individual is connected.
 For more details, see [Creating an Individual Classification](#).
5. Click **OK**.
 The name of the new individual appears in the tree under the business dictionary.

Individual Properties

The individual properties window presents the following elements:

- Its main **Characteristics** with its **Local Name** and owner dictionary
- The **Classifications**



An individual classification is used to connect an individual to the concept that characterizes it.



For more details, see [Creating an Individual Classification](#).

- The **Individual States**, which present the different states of an individual



For more details, see [Describing Individual States and Events](#).

- The **Components**

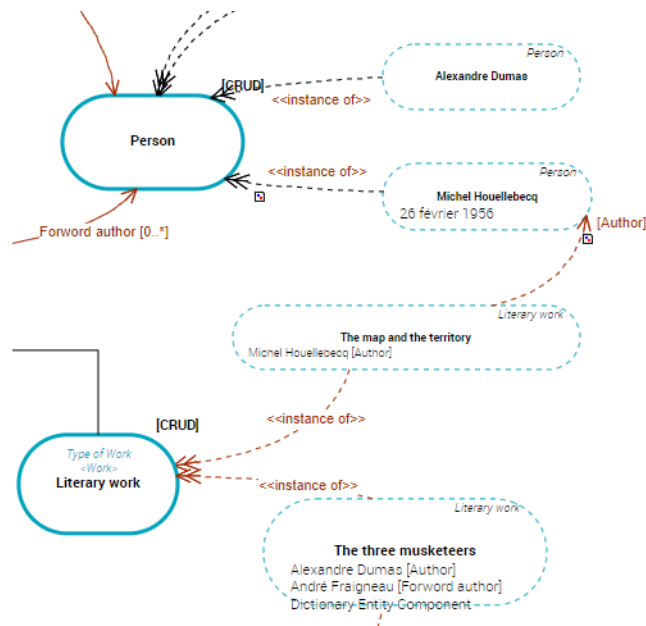


For more details, see [Creating a Dictionary Entity Component](#).

Creating an Individual Classification

An individual classification is used to connect an individual to the concept that characterizes it.

For example, the individual "Michel Houellebecq" is an instance of "Person" and "The Map and the Territory" is an instance of "Library Work".



To create an individual classification:

1. Open the properties window of the individual carrying the relationship.
For example, the "Michel Houellebeck" individual.
2. Select the **Characteristics** tab.
3. In the **Classification** section, click the **New** button.
The individual classification creation wizard opens.
4. At the left of the **Characterizing Element** field, click the **Connect** button.
The query wizard opens.
5. Select the concept you want to connect.
For example, the "Person" concept.
6. Click **OK**.
The individual classification into individuals is created.

Creating a Dictionary Entity Component

An entity component is used to connect an individual to a dictionary element.

You can also connect two Individuals with a **Dictionary Entity Component** relationship type.

For example, you can specify that "Michel Houellebeck" is the author of the work "The Map and the Territory".

To create a dictionary entity component between two individuals:

1. Open the properties window of the individual carrying the relationship.
For example, the "Michel Houellebeck" individual.
2. Select the **Components** tab.
3. Click the **New** button.
The dictionary entity component creation wizard opens.
4. At the left of the **Characterizing Element** field, click the **Connect** button.
The query wizard opens.
5. Select the individual you want to connect.
For example, the "The Map and the Territory" individual.
6. Click **OK**.
The entity component is created. It appears in the individual structure diagram of the described object.

➡ For more details, see [Individual Structure Diagram](#).

Individual Structure Diagram

The individual structure diagram describes the internal structure of the concept instance and the links between its components. This diagram is initialized from concept graph elements.

This diagram is composed of *dictionary entity components* used to connect two individuals.

It is then possible to specify that "Michel Houellebeck" is the author of the work "The Map and the Territory".



An entity component is used to connect an individual to a dictionary element.



For more details, see [Creating a Dictionary Entity Component](#).

CONCEPT OR INDIVIDUAL STATES

A business object can have a life cycle during which it takes different states according to events. If a concept is connected to a business object, other concepts can be connected to different states of the business object and to events at the causing changes of state. With **HOPEX Data Governance** and **HOPEX Information Architecture**, it is possible to associate a life cycle with a concept, as well as state concepts and event concepts.

Individuals can also be connected to individual states and individual events that are instances of state concepts and event concepts.

Describing State Concepts

The notion of state of a concept is represented by the **State Concept**.

 *A state concept is a situation in a concept life cycle during which it satisfies certain conditions, executes a certain activity or waits for a concept event. A state concept represents a time interval of which limits are two concept events. A state concept is a phase through which the concept passes during its life cycle.*

For example, the same subscription holder can pass from "Child" state to "Adolescent" state, then to "Adult" state and finally "Senior".

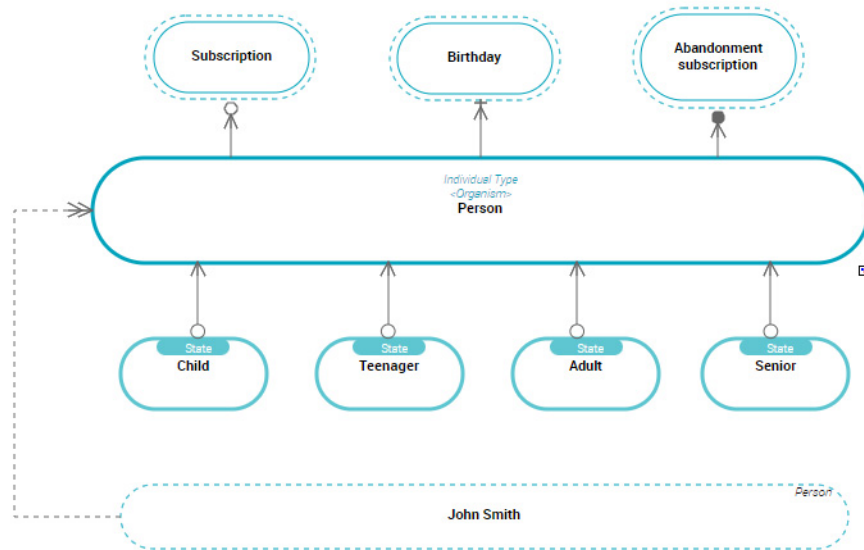
Passage from one state concept to another can be conditioned by an **Event Concept**.

 *An event concept represents an event occurring during concept life, for example a change of season. An event concept marks the impact on a concept of a phenomenon internal or external to the concept. Concept*

events can be distinguished as concept start events, end events and intermediate events.

For example, passage from one state to another can be connected to a event, a "Birthday" for example.

➡ For more details, see [Describing Event Concepts](#).



Accessing the state concepts list

To access business dictionary state concepts in **HOPEX Data Governance**:

1. Click the **Glossary > Dictionaries** navigation pane.

➡ In **HOPEX Information Architecture**, click the navigation menu then **Business Glossary > Business Information Assets**.

2. In the edit area, expand the folder of the business dictionary that interests you.
3. Expand the **State Concepts** folder.

The list of state concepts of the business dictionary appears.

Creating a state concept from a business dictionary

At creation of a state concept, **HOPEX** also creates a **Dictionary State of**, which represents the relationship between a state concept and its concept.

📖 A dictionary state enables connection of a concept to a concept state, and specification of the state nature.

To create a state concept from a business dictionary:

1. Right-click the business dictionary that interests you and click **New > Business Information Building Block**.
2. In the wizard select the **State Concept** object type.
3. Specify the **Local Name** and click **OK**.

4. In the **State Individual Type** field, specify to which concept the state concept you are creating is connected.
 - ☞ A **Dictionary State Of** is automatically created between the concept and the state concept.
5. In the **Term** section, the **Existing Terms** section lists terms with the same name as the new state concept.
 - 📖 A term is a word or word group, that is used for a specific meaning in a specific context.
 - ☞ If a term has already been created with the same name as the new state concept, this term is automatically connected and appears automatically in the **Term** section.
6. In the **Definition Text** field, enter the text of the state concept definition and click **OK**.
The name of the state concept appears in the tree under the business dictionary.
 - ☞ You can also create a state concept in a concept domain.

State concept properties

State concept characteristics

The **Characteristics** tab of state concept properties enables access to its main characteristics.

The state concept is described by:

- its **Designation**, which is represented by one or several terms,
 - ☞ To modify the name of a concept in the corresponding language, you must access concept properties and modify the name of the term in the specific language. For more details, see [Concept and Term](#).
- the **Definition Text**,
- The **Synonyms** section enables specification of a list of synonym concepts,
 - 📖 A synonym is a term interchangeable with another term in the context of a concept of this term that has the same or almost the same meaning.
 - ☞ For more details, see [Concept and Term](#).
- The **Realization** section enables association of an application architecture element to the concept.
 - ☞ For more information, see **HOPEX Logical Data**.

Links between a state concept and other dictionary elements

In addition to terminology characteristics, a state concept is characterized by its relationships with other dictionary elements.

- The **Super-Type** tab presents concepts whose properties are inherited by the described concept, for more details see [Concept Inheritances](#)
- The **Components** tab presents:
 - the list of concept components owned, for more details see [Concept Components](#).
 - the list of concept power components, for more details see [Describing Concept Power Components](#).

☛ *Concepts connected to a state concept are not present in the properties dialog box.*

Describing Event Concepts

An event concept represents an event occurring during concept life, for example a change of season. An event concept marks the impact on a concept of a phenomenon internal or external to the concept. Concept events can be distinguished as concept start events, end events and intermediate events.

Accessing the event concept list

To access business dictionary event concepts in **HOPEX Data Governance**:

1. Click the **Glossary > Dictionaries** navigation pane.

☛ *In **HOPEX Information Architecture**, click the navigation menu then **Business Glossary > Business Information Assets**.*

2. In the edit area, expand the folder of the business dictionary that interests you.
3. Expand the **Event Concepts** folder.

The list of event concepts of the business dictionary appears.

☛ *By expanding the folder of a concept, you can access event concepts attached to it.*

Creating an event concept from a business dictionary

To create an event concept from a business dictionary:

1. Right-click the business dictionary that interests you and click **New > Business Information Building Block**.
In the wizard select the **Event Concept** object type.
2. Specify the **Local Name** and click **OK**.
3. The **Existing Terms** section lists terms with the same name as the new event concept.

📖 *A term is a word or word group, that is used for a specific meaning in a specific context.*

☛ *If a term has already been created with the same name as the new event concept, this term is automatically connected and appears in the **Term** section.*

4. In the **Definition Text** field, enter the text of the event concept definition and click **OK**.
The name of the event concept appears in the tree under the business dictionary.

 You can also create an event concept in a concept domain.

Event concept properties

The **Characteristics** tab of event concept properties enables access to its main characteristics.

With **HOPEX Information Architecture**, the event concept is described by:

- its **Designation**, which is represented by one or several terms,
 To modify the name of a concept in the corresponding language, you must access concept properties and modify the name of the term in the specific language. For more details, see [Concept and Term](#).
- the **Definition Text**,
- The **Synonyms** section enables specification of a list of synonym concepts,
 A synonym is a term interchangeable with another term in the context of a concept of this term that has the same or almost the same meaning.
 For more details, see [Concept and Term](#).
- The **Realization** section enables association of an application architecture element to the concept.
 For more information, see **HOPEX Logical Data**.

Connecting an event concept to its concept

The relationship between a concept and its event concept is described by:

- a **Start Event**,
- an **End Event**,
- or an **Intermediate Event**.

To connect an event concept to its concept in a diagram associated with a concept domain:

1. In the insert toolbar, click the **Link** button.
2. Click the concept to which the event concept is attached.

For example, "Person"

3. Click the event concept to be connected.

For example, "Birthday".

A wizard proposes selection of an event type:

- **Concept Start Event**
 - **Concept End Event**
 - **Concept Intermediate Event**
4. Select the event type and click **OK**.
The creation wizard of the selected concept event type opens.
 5. Specify the **Local Name**.
 6. If no term is to be created, select the "Creation without term" check box.

7. Click **OK**.

The link between the concept and the event concept appears in the diagram with an icon representing its type.

State Concept Structure Diagram

A state concept structure diagram assembles all information relating to the state concept diagram described. This diagram is initialized from concept diagram elements.

For example,

The diagram includes:

- **variations** between components

For example, "Subscriptions" can be classified by "Subscription Type". A "Subscription Type" being characterized by a "Loan Type".



A variation describes how a concept can be varied under another form. The variant is an object similar to the varied object, but with properties or relationships that may differ.

For more details, see [Defining Inheritance of a Concept Component](#).

- **Substitutions** between components



A substitution determines which element can be used to replace another, or is effectively replaced by an element existing in a given context (for example in the context of a variation). Unlike a variation, a substitution does not involve inheritance but a functional equivalence.

For more details, see [Creating a Concept Component Substitution](#).

- **Sub-properties**



A sub-property is used to specify the relationship between a concept and a concept property.

For more details, see [Concept Properties](#).

- **Concept components** describing the relationship between two **Concepts**

For example, a "Subscription Type" is characterized by an "Available Loan Type".



A concept component enables representation of a dependency relationship between two concepts. This relationship is directional.

For more details, see [Concept Components](#).

- **start events**, **intermediate events** and **end events** enabling definition of events contributing to change of state of a concept,

For example, the change of state of a member can be caused by a birthday.



An event concept represents an event occurring during concept life, for example a change of season. An event concept marks the impact on a concept of a phenomenon internal or external to the concept. Concept events can be distinguished as concept start events, end events and intermediate events.

For more details, see [Describing State Concepts](#).

Describing Individual States and Events

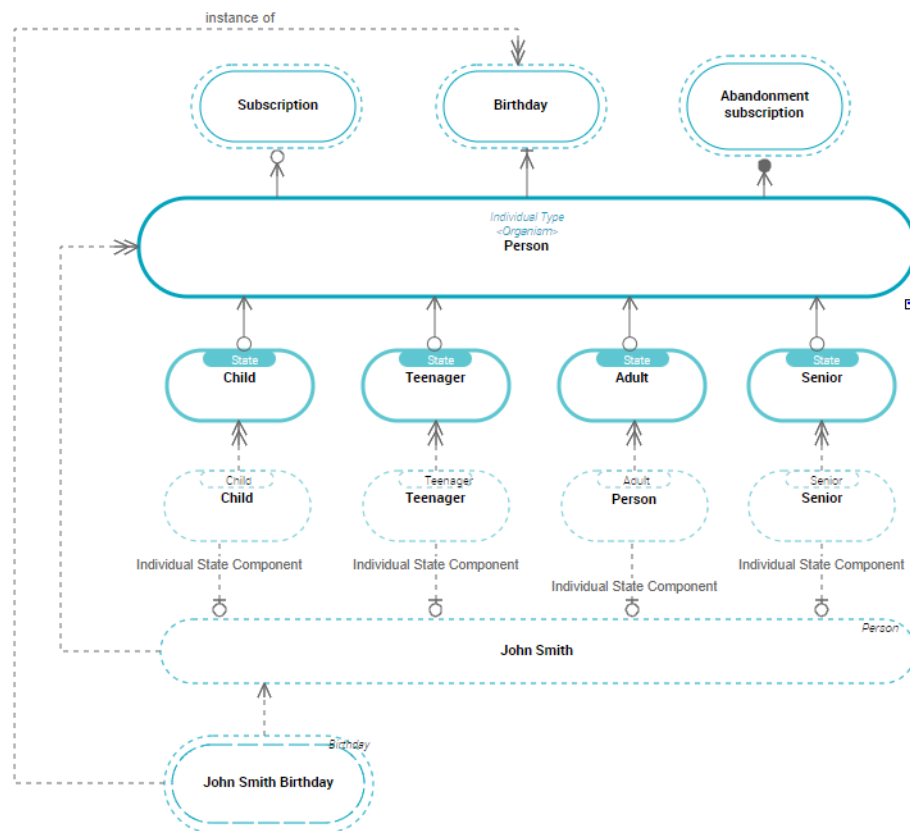
If a concept is associated with states, occurrences of this concept can also be associated with states. **HOPEX Data Governance** and **HOPEX Information Architecture** therefore proposes the **Individual State**.

 An individual state is an instance of a concept state to which the dictionary state is connected. It represents an individual state during its life cycle.

In addition, the switch from one individual state to another can be conditioned by an **Individual Event**.

 An individual event represents an event occurring during the life of the individual. It is an instance of an event concept of the concept to which the individual is connected.

For example, "John Smith" is a "Person" who can pass from one state to another on his birthday.



The relationship between an individual and its **Individual State** is described by an **Individual State Component**.

 An individual state component is used to connect an individual to an individual state.

The relationship between an individual and its **Individual Event** is described by a **Dictionary Entity Component**.



An entity component is used to connect an individual to a dictionary element.

With **HOPEX Information Architecture**:

- an individual state is an instance of a state concept



A state concept is a situation in a concept life cycle during which it satisfies certain conditions, executes a certain activity or waits for a concept event. A state concept represents a time interval of which limits are two concept events. A state concept is a phase through which the concept passes during its life cycle.

- an individual event is an instance of an event concept



An event concept represents an event occurring during concept life, for example a change of season. An event concept marks the impact on a concept of a phenomenon internal or external to the concept. Concept events can be distinguished as concept start events, end events and intermediate events.

Accessing the individual state and event list

To access the individual states of a business dictionary in **HOPEX Data Governance**:

1. Click the **Glossary > Dictionaries** navigation pane.

In **HOPEX Information Architecture**, click the navigation menu then **Business Glossary > Business Information Assets**.

2. In the edit area, from the business dictionary that interests you, expand the **Individual States** folder.
The list of individual states of the business dictionary appears.
3. Expand the **Individual Events** folder.
The list of individual events appears.

Creating an Individual state from a concept domain

The relationship between an individual and its **Individual State** is described by an **Individual State Component**.



An individual state component is used to connect an individual to an individual state.

If you create an individual state in a diagram, you can automatically create the individual state component of the associated individual.

To create an individual state from a concept diagram:

1. In the diagram, roll the mouse over the individual who owns the individual state.
2. Select **Individual state**.
3. Click in the diagram.
The individual state creation wizard opens.
4. Specify the **Local Name** and click **Add**.
The new individual state appears in the diagram.

You can also create an individual state from its business dictionary.

Individual state properties

The individual state properties dialog box presents the following elements in the **Characteristics** tab:

- Its **Local Name**
- The individual classifications, which appear in the **Classification** section.



An individual state component is used to connect an individual to an individual state.



For more details, see [Creating an Individual Classification](#).

- The **Component** tab, presenting the individuals who define the described individual.



For more details, see [Creating a Dictionary Entity Component](#).

Creating an Individual event from a concept domain

To create an individual event from a concept domain:

1. In the insert toolbar, click **Individual Event** and click in the diagram.
The individual event creation wizard opens.
2. Specify the **Name** and click **Add**.
The individual event appears in the diagram.

Connecting an individual event to an individual

The relationship between an individual and its **Individual Event** is described by a **Dictionary Entity Component**.



An entity component is used to connect an individual to a dictionary element.

To connect an event concept to its concept in the diagram:

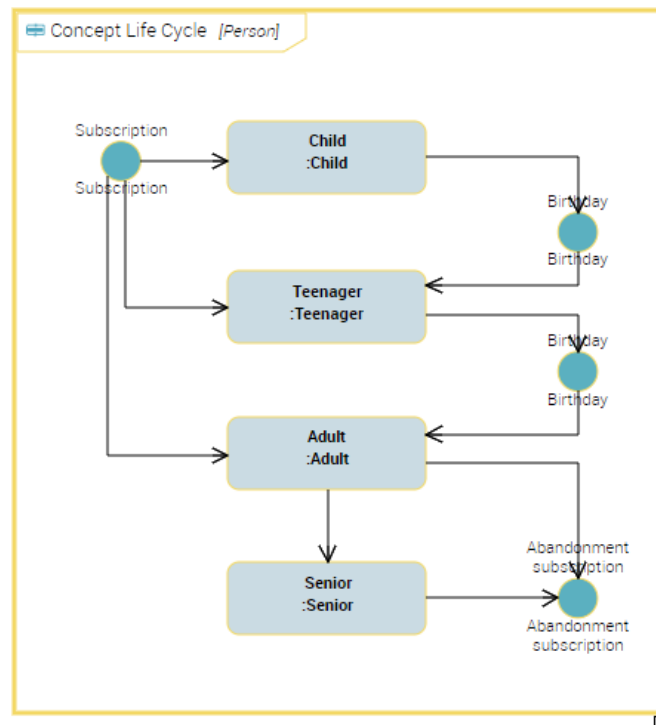
1. In the insert toolbar, click the **Link** button.
2. Click the individual event.
3. Click the event.
The link appears in the diagram.

Concept life cycle structure diagram

The concept life cycle structure diagram is used to describe a concept life cycle.

For example, a "Person" becomes visible in a media library after "Registration". It can be registered with state "Child", "Adolescent", "Adult" or "Senior". Passage from

one state to another can be connected to a event, a "Birthday" for example.



A concept life cycle structure diagram includes the following elements:

- **Concept Life Cycle Phases**, which are connected to state concepts of the "Person" concept
 - A state concept is a situation in a concept life cycle during which it satisfies certain conditions, executes a certain activity or waits for a concept event. A state concept represents a time interval of which limits are two concept events. A state concept is a phase through which the concept passes during its life cycle.
 - For more details on state concepts, see [Describing State Concepts](#)
- **Concept Life Cycle Events**, which are connected to event concepts of the "Person" concept
 - An event concept represents an event occurring during concept life, for example a change of season. An event concept marks the impact on a concept of a phenomenon internal or external to the concept. Concept events can be distinguished as concept start events, end events and intermediate events.
 - For more details on event concepts, see [Describing Event Concepts](#)
- **Concept Life Cycle Transitions**, which represent sequence flows between concept states and events.

Creating a concept life cycle

To create a concept life cycle structure diagram and to describe sequence flows of states defining the concept life cycle, you must first create the **Concept Life Cycle**.

To create a concept life cycle from a business dictionary:

1. Right-click the business dictionary that interests you and click **New > Business Information Building Block**.
2. In the wizard select the **Concept Life Cycle** object type.
3. Specify the **Local Name** and click **OK**.
4. In the **Life Cycle of**, specify the concept to which the life cycle relates.

For example, "Person"

5. The **Existing Terms** section lists terms with the same name as the created object.

 If a term has already been created with the same name as the now concept, this term is automatically connected to the concept and appears automatically in the **Term** section.

6. In the **Definition Text** field, enter the text of the state concept definition and click **OK**.
The name of the new concept life cycle appears in the tree under the business dictionary.

Creating a concept life cycle structure diagram

To create a concept life cycle structure diagram from a concept life cycle:

1. Right-click the concept life cycle that interests you and select **New > Concept Life Cycle Structure Diagram**.
The diagram opens in the edit area. State concepts associated with the described concept are positioned in the diagram via objects of **Concept Life Cycle Phases** type.

Adding a concept life cycle event

To add a concept life cycle event in the concept life cycle structure diagram:

For example, the concept life cycle event representing "Registration".

1. In the diagram insert toolbar, click the **Concept Life Cycle Event** button.
2. Click in the frame of the concept life cycle frame.
A concept life cycle event creation dialog box opens
3. In the **Composite Type** field, specify the name of the event concept to which the new object relates.

For example, "Registration".

 If a selection dialog box opens, select the object that interests you.

4. Specify the **Local Name**.
5. If no term is to be created, select the "Creation without term" check box.
6. Click **OK**.
The concept life cycle event appears in the diagram.

Creating a concept life cycle transition

To represent sequence flow from a phase to a concept life cycle event, you must create a concept life cycle transition.

To create a concept life cycle transition:

1. In the diagram insert toolbar, click the **Concept Life Cycle Transition** button.
2. Click the triggering concept life cycle phase (or event), and, holding the mouse button down, drag the cursor to the triggered phase (or event).
3. Release the mouse button.
The link appears in the diagram.

Using periods

A **period** adds time-related information to an **individual event**.



An individual event represents an event occurring during the life of the individual. It is an instance of an event concept of the concept to which the individual is connected.

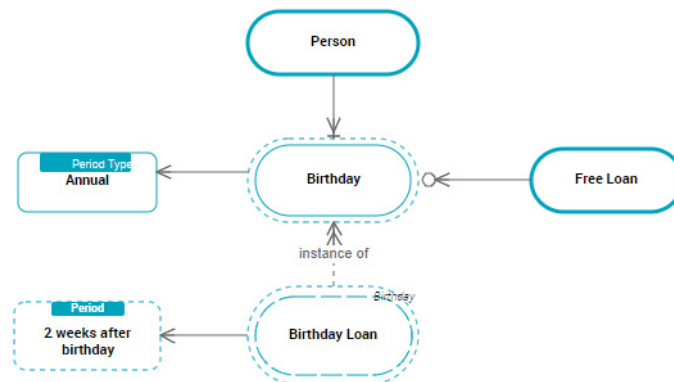
For example, a free loan may be offered to subscribers on each anniversary. This loan is valid for a period of two weeks after the anniversary date.

A **period type** is used to specify an **event concept**.



An event concept represents an event occurring during concept life, for example a change of season. An event concept marks the impact on a concept of a phenomenon internal or external to the concept. Concept events can be distinguished as concept start events, end events and intermediate events.

For example, a free anniversary loan is offered every year.



The relationship between a **Period type** and an **Individual event** is described by an **Event type periodization**.

The relationship between a **Period** and an **Event concept** is described by an **Event periodization**.

CONCEPT TYPE

A concept type enables classification of concepts. Relationships between concept types are represented by concept type components.

Accessing the Concept Types List

To access business dictionary type concepts in **HOPEX Data Governance**:

1. Click the **Glossary > Dictionaries** navigation pane.
In HOPEX Information Architecture, click the navigation menu then Business Glossary > Business Information Assets.
2. In the edit area, from the business dictionary that interests you, expand the **Concept Types** folder.

Creating a New Concept Type

To create a concept type from a business dictionary:

1. Right-click the business dictionary that interests you and click **New > Business Information Building Block**.
2. In the wizard select the **Concept Type** object type.
3. Click **Next**.
4. Specify the **Local Name** and click **OK**.
5. The **Existing Terms** section lists terms with the same name as the new concept type .

 A term is a word or word group, that is used for a specific meaning in a specific context.

 If a term has already been created with the same name as the new concept type, this term is automatically connected and appears in the **Term** section.

6. In the **Definition Text** field, enter the text of the concept type definition and click **Finish**.
The name of the new concept type appears in the tree under the business dictionary.

 A new term with the same name as the concept type is also created.

Concept Type Properties

Concept type characteristics

The **Characteristics** tab of concept type properties enables access to its main characteristics.

With **HOPEX Information Architecture**, the concept type is described by:

- its **Designation**, which is represented by one or several terms,
 - ✎ *To modify the name of a concept in the corresponding language, you must access concept properties and modify the name of the term in the specific language. For more details, see [Concept and Term](#).*
- the **Definition Text**,
- The **Synonyms** section enables specification of a list of synonym concepts,
 - 📖 *A synonym is a term interchangeable with another term in the context of a concept of this term that has the same or almost the same meaning.*
 - ✎ *For more details, see [Concept and Term](#).*
- The **Realization** section enables association of an application architecture element to the concept.
 - ✎ *For more information, see **HOPEX Logical Data**.*

Links between a concept and other dictionary elements

In addition to terminology characteristics, a concept is characterized by its relationships with other dictionary elements.

- The **Component** tab presents the list of owned concept type components , for more details see [Concept Components](#).
- The **Super-Type** tab presents concept types whose properties are inherited by the described concept type, for more details see [Describing Concept Type Variations](#)

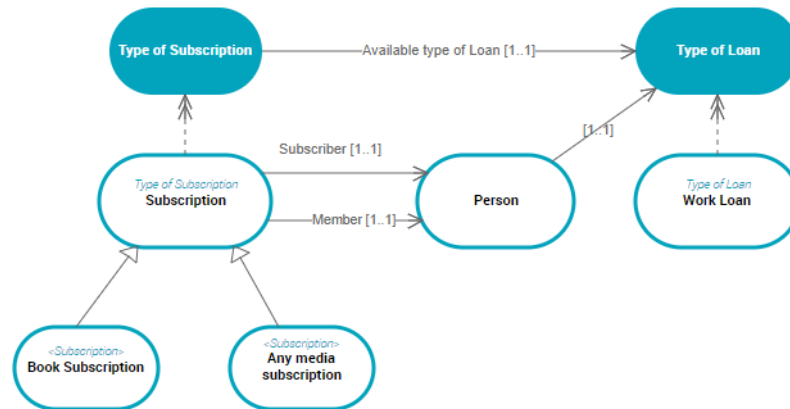
Describing Concept Type Components

A concept type can be connected to another concept type to characterize it.

For example, a "Subscription Type" is characterized by a "Loan Type".

This relationship is described by a **Concept Type Component**, which can be associated with a term.

 A concept type component enables specification of the relationship between two concept types.



Accessing concept type components

To access concept type components of a concept type:

1. Open the concept type properties dialog box.
2. Select the **Components** tab.

The list of concept type components associated with the concept appears.

 You can also consult the list of components of a concept type from its concept life cycle diagram. For more details, see [The Concept Type Structure Diagram](#).

Creating a concept type component from a concept domain

To create a concept type component between two concept types in a concept domain diagram:

1. In the insert toolbar, click the **Link** button.
2. Click the concept type that owns the link.

For example, "Subscription Type".

3. Click the target concept type.

For example, "Loan Type".

The concept type component creation wizard appears.

4. Specify the **Local Name**.
5. If no term is to be created, select the "Creation without term" check box.
6. Click **OK**.

The Concept Type component appears in the diagram.

You can also create a concept type component in a concept type structure diagram. In this case, you must specify the target concept type in the concept type component creation wizard.

☛ For more details, see [The Concept Type Structure Diagram](#).

Describing Concept Type Variations

Certain concept types are versions of other concept types; they are characterized by the same concept type components.

This relationship is described by a **Variation**.



A variation describes how a concept can be varied under another form. The variant is an object similar to the varied object, but with properties or relationships that may differ.



*For more details on variations and substitutions, see the **HOPEX Common Features** guide, "Handling Repository Objects", "Object Variations".*

Accessing concept type variations

To access concept type variations

1. Open the concept type properties dialog box.
2. Select the **Super-Type** tab.

The list of variations associated with the concept appears.

Creating a concept type variation from a concept domain

To specify, from a concept domain diagram, that a concept type inherits characteristics defined for another concept type:

1. In the insert toolbar, click the **Link** button.
2. Click the concept type to be varied and drag the cursor to the new concept before releasing the mouse button.
3. Specify the **Name** and click **Add**.

A directional link from the concept type to be varied to the root concept type appears.



*The variation is represented by a link, but it is in fact a **HOPEX** object.*

The Concept Type Structure Diagram

A concept type structure diagram describes the internal structure of the concept type instance using relationships defined for other concept types it characterizes.

This diagram includes *concept type components* enabling characterization of the concept type by connecting it to other concept types.

For example, a "Subscription Type" is characterized by a "Loan Type".



A concept type component enables specification of the relationship between two concept types.

➡ For more details, see [Describing Concept Type Components](#).

CONCEPT VIEW

A concept view enables representation of the semantic scope covered by a business object. A concept view is based on the selection of several concepts specific to the view.

An editor allows you to create and visualize business views and their components.

➡ On the same principle, the *Data View* can be used to navigate from *Classes* or *Entities*. For more details, see [Logical Data View](#).

Creating a Concept View

To create a concept view:

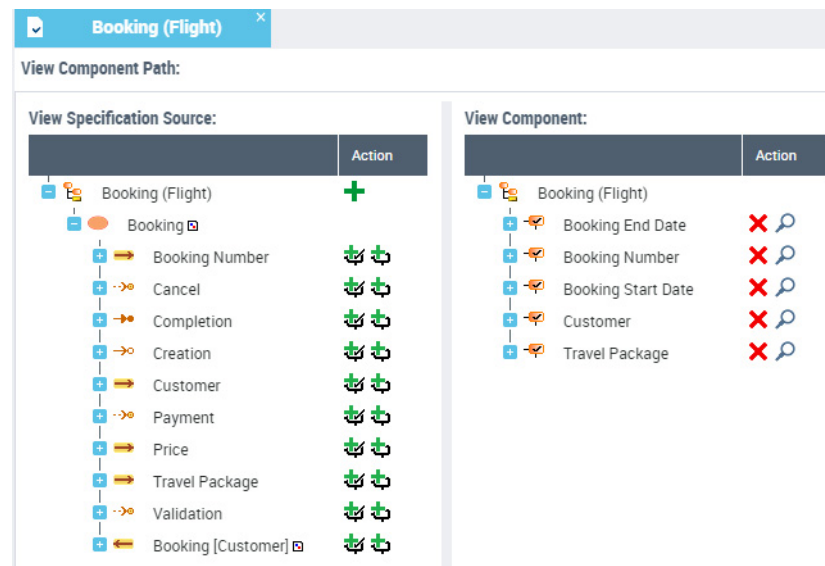
1. Right-click the name of the concept and select **New > Concept view**. The concept view creation wizard appears.
2. Enter the **Local Name**.
3. The **Existing Terms** field lists terms with the same name as the view.
 *A term is a word or word group, that is used for a specific meaning in a specific context.*
4. In the **Definition Text** space, enter the text of the definition of the view and click **Next**.
5. To specify the source concept of the concept view, click **New**.
6. In the dialog box that appears, enter:
 - the **MetaClass** concerned by the view (concept, state concept or event concept)
 - The reference concept of the data view
7. Click **Add**.
8. Click **OK** to close the concept view creation wizard. The new concept view appears in the list.

Defining the Concept View Content

Displaying objects in the view

The view editor is made up of a number of parts:

- the left part presents all the source concept components held by the view, as defined in the data dictionary
- the right part presents the concept components that will be kept for the concept view created
- the buttons in the **Action** column are used to add the components to the concept view.



Adding a source object to the concept view

To add a source object to a concept view:

1. Open the concept view.
2. On the source object side, under the **Action** column, click the .
3. In the dialog box that appears, indicate:
 - the **MetaClass** concerned by the view (concept, state concept or event concept)
 - The reference concept of the data view
4. Click **Add**.

Once the source concept is defined, you can select the components of this concept - or the concept itself - to be added to the concept view.

Adding a component to the concept view

Using the source objects in the view, you can define embedded components and referenced components.

An embedded component is used to bring all the information making up the object into the view. A reference component references only the object in the view.

To add a component to the concept view:

1. In the tree on the left, select the component that you want to add to the view.
2. Click **Add a View Inclusion Component**.
The component added appears in the tree to the right.

 You can **Add a referenced component** in the same way.

A check mark appears in front of the objects embedded in the view, as opposed to referenced objects.

The views are then accessible in a report. For more details, see [Report DataSets](#).

The View Report

The view report provides a report on a concept view and its components.

To generate a view report:

1. Click the navigation menu, then **Reports**.
2. In the navigation pane click **Description Reports**.
3. In the edit area, click the **View Report** tile.
4. In the **View** field, select the view in question and refresh the report.

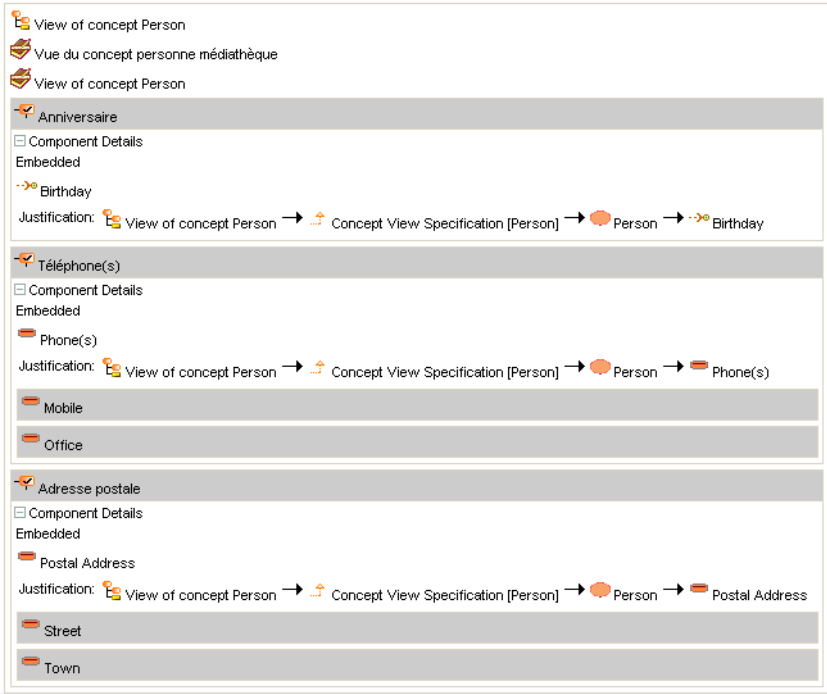
Report parameters

This consists of defining report input data.

Parameters	Parameter type	Constraints
View	View	Mandatory.
Sub-view	yes or no	Yes by default
Justification	yes or no	Yes by default
Depth level	Short	

Report example

The following example show the elements in the view based on the "Person" concept.





CALCULATION RULE ON CONCEPTS



HOPEX Data Governance and **HOPEX Information Architecture** allow you to define calculated data elements, the value of which is calculated from the values of one or more other data elements.

CREATING A CALCULATION RULE ON A BUSINESS OBJECT

You can create calculation rules on concept components and concept properties.

Calculation Rule on a Concept Component

To indicate that a concept component is calculated, you must link the concept and the component with a link of type **Computed Concept Component** and define the calculation rule for the component.

The calculation rule defines the input and output objects as well as the expression of the rule. The input and output objects can consist of different types of business information. The output objects are concepts only.

To create a computed concept component between two concepts of a concept domain:

1. In the concept graph associated with the concept domain, click on the concept that holds the other one.
A bar containing the objects you can insert appears.
2. Click on the icon that represents the **Computed concept component**.
3. Slide the cursor to the target concept.
4. When the cursor becomes a double chain link, release the mouse button.
The computed concept component creation wizard appears.
5. Enter the **Name** of the data area.
6. Specify if a term must be created or not.
7. Click **Next**.
8. Associate a concept rule and click **OK**.
The concept component appears in the graph.

 A new term with the same name as the concept component is also created if you have selected creation of a term.

To access the definition of the rule:

1. Open the properties of the concept component.
2. In the **Characteristics > Calculation rule** page, enter the input and output parameters of the rule, as well as the description of the rule.

Calculation Rule on a Concept Property

To indicate that a concept property is calculated, you must link the concept and the concept property with a link of type **Computed Sub-Property** and define the calculation rule for the concept property.

The calculation rule defines the input and output objects as well as the expression of the rule. The input and output objects of the rule are concept properties.

To create a computed sub-property:

1. In the concept graph create the concept property.

2. Click the concept that owns the concept property.
A bar containing the objects you can insert appears.
3. Click on the icon that represents the **Computed Sub-Property**.
4. Slide the cursor to the target concept property.
5. When the cursor becomes a double chain link, release the mouse button.
The creation wizard of a computed sub-property appears.
6. Specify if a term must be created or not.
7. Click **Next**.
8. Associate a rule and click **OK**.
The computed sub-property appears in the graph.

☛ A new term with the same name as the concept property is also created if you have selected creation of a term.

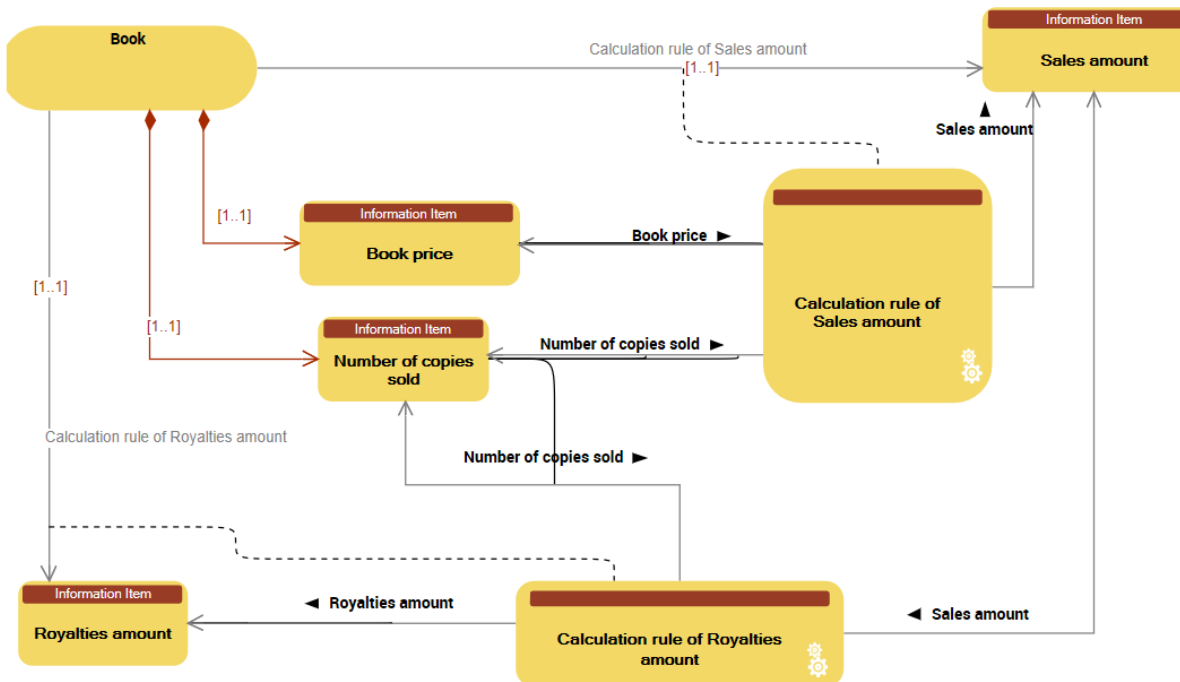
To access the definition of the calculation rule:

1. Open the properties of the computed sub-property.
2. In the **Characteristics > Calculation rule** page, enter the input and output parameters of the rule, as well as the description of the rule.

Example

The "Book" concept is described by the following concept properties:

- "Book price"
- "Number of copies sold"
- "Sales amount"
- "Royalties amount"



The "Sales amount" is calculated according to:

- the "Book price"
- the "Number of copies sold"

The definition of the sales amount calculation rule is as follows:

Sales amount = Number of copies sold x Book price

The royalties amount is calculated according to the number of copies sold. The calculation rule is as follows:

- 8% of the sales amount if the number of copies sold is less than 10,000.
- 10% of the sales amount if the number of copies sold is less than 20,000 and greater than 10,000.
- 12% of the sales amount if the number of copies sold is less than 20,000.

CONNECTING THE BUSINESS CONCEPTS TO THE LOGICAL AND PHYSICAL ARCHITECTURE



You can indicate how the business concepts are implemented in the IS by connecting them to the objects in the logical or physical layer.

The "Concept realization" work consists of connecting the data model or database elements with business concepts to:

- precisely define objects handled at IS architecture level,
- assure improved vocabulary sharing and improved global communication between business users and IS users.

The following points are covered here:

- ✓ ["Realization of Concept", page 152](#)
- ✓ ["Using Realization Matrices", page 154](#)

REALIZATION OF CONCEPT

Using the "Realization" concept, you can connect logical or physical view elements with dictionary elements.

The realization can be defined on the realized object (concept) or the realizing object (logical or physical data), and extended to the components of the realized and realizing objects.

Note that it is also possible to connect business information with other business information.

Defining the Object that Realizes a Concept

To define the object that realizes a concept in **HOPEX Data Governance** :

1. Click the navigation menu, then **Glossary > Business Data**.
2. In the edit area, select the concept concerned and display its properties.
3. In the property page, from the drop-down list, click the **Characteristics > Realizations** page.
4. In the **Realizer Objects** tab, click **New**.
The business realization creation wizard appears.
5. Specify:
 - the object type that realizes (logical or physical data)
 - the name of the object concerned
6. Click **Add**.
The realizer object appears in the concept properties. When you select this object, a matrix appears in the next section. It displays the components of the object that realizes (the class) in rows and the components of the realized object (the concept) in columns. From this matrix you can define which components of the class (eg. Attributes) realize which concept components.

(Class of Block Component...)	Catalog_Endin... 📄	Catalog_Id 📄	Catalog_Startin... 📄
📄 Catalog_Ending_date	✓		
📄 Catalog_Id		✓	
📄 Catalog_Starting_date			✓

Defining the Concept Realized by a Class

To specify the concept realized by a class:

1. Click the navigation menu then **Architecture > Classes**.
2. In the edit area, select the class that you want to connect to a concept and open its properties.
3. Select the **Characteristics > Realization** page.
4. In the **Realized Objects** tab, click **New**.
The **Add an Owned Realization** dialog box appears.
5. In the **Object type** field, select "Business information realization" and click **Next**.
6. In the **MetaClass** field, select "Concept".
7. In the **Business information realized** field, select the concept you are interested in.
8. Click **Add**.
The concept appears in the properties of the class. When you select this concept, a matrix appears in the next section. It displays the components of the object that realizes (the class) in rows and the components of the realized object (the concept) in columns. From this matrix you can define which components of the class (eg. Attributes) realize which concept components.

USING REALIZATION MATRICES

Realization matrices allow you to define and view the realization links between objects in the repository.

Example

Realization of business data by logical data

This matrix is used to specify that logical data (classes, data views, etc.) realize business information (concepts, concept types, etc.).

Realization Levels

Business Information X Business Information

This matrix is used to specify that business data (concepts, concept types, etc.) realize other business information.

Business Information X Logical Data

Realization of business data by logical data

Business Information X Metadata

Realization of business data by metadata

Logical Data X Logical Data

Realization of logical data by other logical data

Concept Domain Maps by Data Domain Maps

Realization of concept domain maps by data domain maps

Concept Domains X Logical Data Domains

Realization of concept domains by logical data domains

Logical Data Domains X Data Domains

Realization of logical data domains by data domains

☛ The realization of logical data (classes, data views, etc.) by physical data (tables, table views, etc.) takes place via the synchronization tool. See "[Synchronizing logical and physical models](#)".

Creating a Realization Matrix

To create a realization matrix in **HOPEX Data Governance**:

1. Click the navigation menu then **Tools > Data realization**.
2. In the edit area, select the type of matrix to be created.

3. Click **New**.
The matrix appears in the edit area.
4. Add the data that realizes it in a row and the data realized in a column.
5. To specify that one object realizes another object, click on the cell of the matrix that connects the two objects in question.

Business Information R... ✕							
 Add a row		 Add column		 Excel			
(ER Data Entity / Class of Business...)	Actor	Bank Account	Booking	Customer	Deliverable	Enterprise	
Call center agent							
Client							
Delivery							
Employee							
Invoice							

See also: ["Initializing a Business Dictionary Using Logical or Physical Data"](#)



Data Architecture and Tools



MODELING DATA DICTIONARIES



Company organizers and architects can describe operations using **HOPEX Data Governance** and **HOPEX Information Architecture** by modeling data used when implementing business processes and applications. To this end, **HOPEX Data Governance** et **HOPEX Information Architecture** make available a number of tools and notations.

Using logical data models, you can build corresponding physical models, that is, create database tables, with its columns, indexes and keys as well as the relational diagram drawings. See [Synchronizing logical and physical models](#).

You can also inventory applications that use the modeled logical data. See [Use of Data by the Information System](#).

- ✓ [Logical Data Modeling Options](#)
- ✓ [Overview of Logical Data](#)
- ✓ [Data Dictionary](#)
- ✓ [Data Domain Map](#)
- ✓ [Logical and Application Data Domains](#)
- ✓ [Logical Data View](#)
- ✓ [Datatypes](#)
- ✓ [Class Diagram](#)
- ✓ [Data Model](#)
- ✓ [IDEF1X Notation](#)
- ✓ [I.E. Notation](#)
- ✓ [The Merise Notation](#)

LOGICAL DATA MODELING OPTIONS

Formalisms

You can model logical data using two formalisms:

- the data package, to build class diagrams (UML notation)
- the data model, for data diagrams (standard notations, IDEF1X, I.E, Merise)

To display one of the formalisms:

1. On the desktop, click **Main Menu > Settings > Options.**
2. In the options navigation tree, expand the **HOPEX Solutions > Information Architecture** folder.
3. Click **Data Formalism.**
4. In the right part of the window pane select the formalism(s) that you want to display.
5. Click **OK.**
The folders corresponding to the packages and data models appear in the **Logical data** navigation pane.

See also: [Logical Formalism and Synchronization.](#)

Notations

You have access to a standard data model notation, selected by default. To display another notation (DEF1X, I.E or Merise):

1. On the desktop, click **Main Menu > Settings > Options.**
2. In the options navigation tree, expand the **HOPEX Solutions > Information Architecture** folder.
3. Click **Data Notation.**
4. In the right part of the window, select the notations that you want to display.
5. Click **OK.**

OVERVIEW OF LOGICAL DATA

In **HOPEX Data Governance** and **HOPEX Information Architecture**, access to logical data in the repository is reserved for the **Data architect**.

Data Dictionary

A data dictionary collects and holds a set of logical data, and provides them with a namespace.

The data dictionary therefore participates in the organization of data in the HOPEX repository.

See [Data Dictionary](#).

Data Domain Map

A data domain map represents the data domains of a data dictionary and their dependency links.

See [Data Domain Map](#).

Logical Data Domain

A logical data domain represents a restricted data structure dedicated to the description of a software Data Store. It is made of classes and/or data views and can be described in a Data Domain Diagram.

For more details, see [Logical and Application Data Domains](#).

To address these specific use cases, you can create Data Views in which you can see and modify the scope covered by the classes.

Logical Data View

From the perimeter of an object in a data dictionary or a data domain, a logical data view allows you to define a set of information for a specific use. See [Logical Data View](#).

Data Model

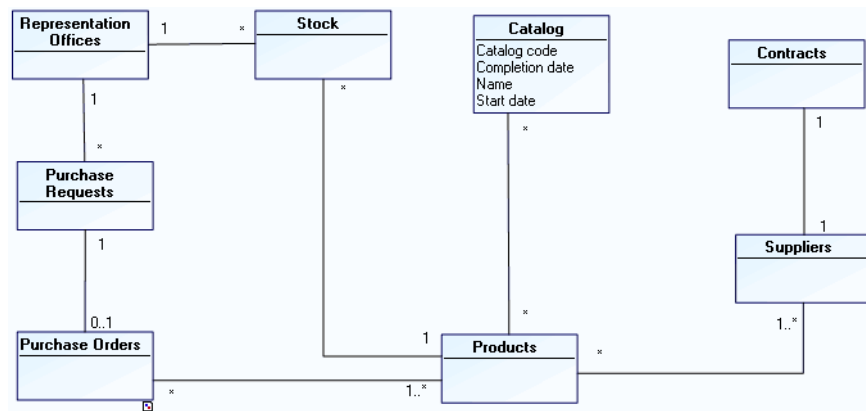
When you choose to work with the "Data Model" formalism, business dictionaries are represented by data models (not data dictionaries).

See [Logical Data Modeling Options](#).

For more details on creating and updating a data model, see [Data Model](#).

Example

The data model of the "Purchase Request Automation" project is presented below.



The application manages purchase requests, orders and product stock levels in each of the representation offices.

A centralized catalog of products and suppliers is installed.

Contracts with referenced suppliers are also accessible from the application.

DATA DICTIONARY

A data dictionary collects and holds a set of logical data.

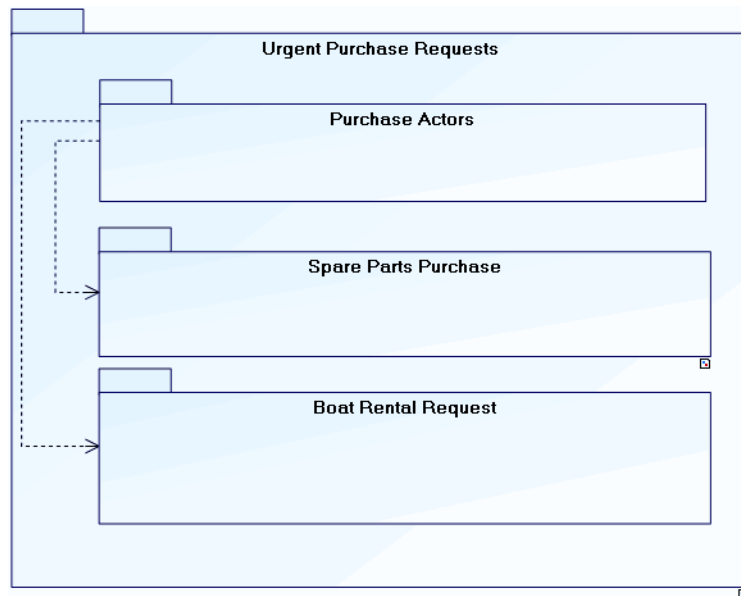
It can be broken down into logical data domains. See [Logical and Application Data Domains](#).

Elements of a Data Dictionary

A data dictionary is used to describe all the elements defining your logical data architecture:

- Data Domains
- Classes
- Attributes
- Parts
- etc.

A data dictionary is implemented by a **package** that collects data. You can create sub-packages.



Urgent purchase requests are provided to process purchase of spare parts and boat rental requests. In both of these cases, users are actors of the purchasing domain.

➡ For more details on the use of packages, see the **HOPEX IT Architecture** guide.

Accessing the elements of a data dictionary

To access the elements of a data dictionary in **HOPEX Data Governance** :

1. Click the navigation menu then **Architecture > Hierarchy View**.

 In **HOPEX Information Architecture**, click the navigation menu then **Data Architecture > Logical Data Assets**.

2. In the edit area, expand the folder of the data dictionary that interests you.
Elements that make up the dictionary appear.

Importing logical data

You can import existing logical data into your repository using an Excel file. See [Importing Logical Data from an Excel File](#).

DATA DOMAIN MAP

A data dictionary can be split into a set of logical data domains. A data domain map is used to visualize the dependencies between logical data domains.

Dependency links between domains are automatically deduced from the objects previously described in each domain of the map.

➡ For more information on data dictionaries, see [Data Dictionary](#).

Creating a Data Domain Map

You can create a data domain map from the data dictionary it describes.

To create a (logical) data domain map in **HOPEX Data Governance**:

1. Click the navigation menu then **Architecture > Hierarchy View**.
➡ In **HOPEX Information Architecture**, click the navigation menu then **Data Architecture > Logical Data Assets**.
2. In the edit area, expand the **Data Dictionaries** folder.
3. Right-click on the data domain and select **New > Data Domain Map**.
The map appears in the tree.

To create the diagram of the data domain map:

1. Right-click the map and select **New Diagram**.
2. Select the diagram type **Data Domain Map**.
3. Click **OK**.
The diagram appears in the edit area.

Components of a Data Domain Map

You can add internal and external components to a data domain map.

The internal components are data domains that are part of the map scope (whether they belong to the owner package or not).

The external components are those used in the map but that are not part of the scope analyzed.

LOGICAL AND APPLICATION DATA DOMAINS

Logical and application data domains are used to define a logical data structure made up of classes and class views.

- The logical data domain is used to describe data stores (internal or external) of logical application systems.
- The application data domain is used to describe the data stores of software (Application system, Application, Application service or Micro Service).

☛ For more details on how to use data domains in an application architecture, see the documentation of **HOPEX IT Architecture** > "Modeling technical and functional architectures".

Both are owned by a package and can reference objects held in other packages.

You can define the access mode (CRUD) to the objects referenced by a data domain by integrating them as components of the data domain.

☛ A corresponding physical structure can be defined via a physical data domain. It is made up of tables and table views. See [Modeling Databases](#).

Creating a Data Domain

You can create a data domain from the data dictionary it describes.

To create a (logical) data domain in **HOPEX Data Governance**:

1. Click the navigation menu then **Architecture** > **Hierarchy View**.

☛ In **HOPEX Information Architecture**, click the navigation menu then **Data Architecture** > **Logical Data Assets**.

2. In the edit area, expand the **Data Dictionaries** folder.
3. Right-click the data dictionary and click **New** > **Data Domain**.
The data domain appears in the tree.

The Data Domain Diagram

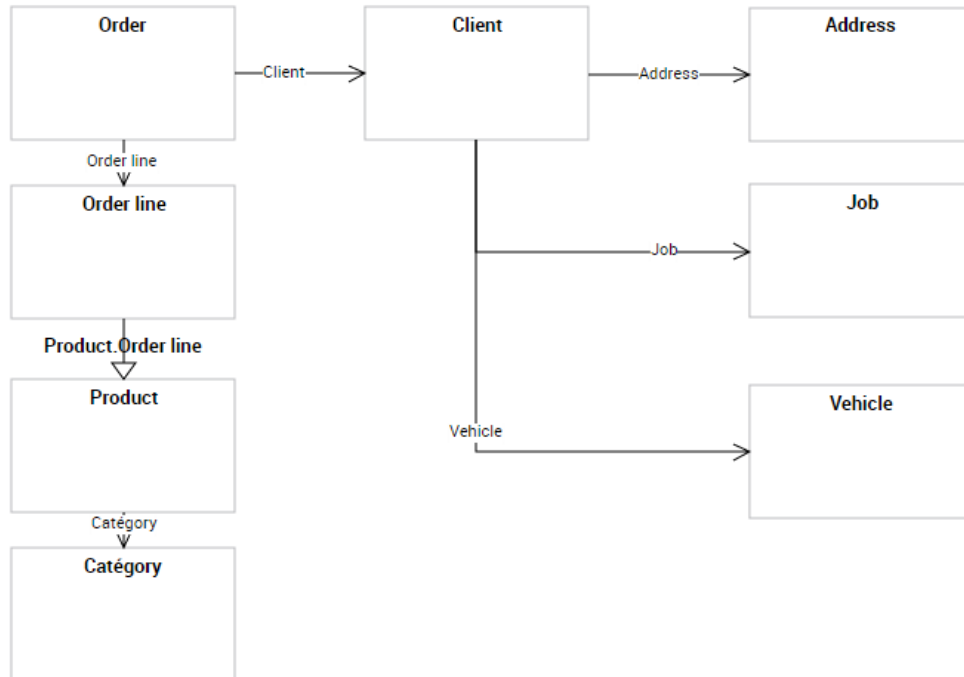
Logical and application data domains can be described by a diagram.

A data domain diagram is a structure diagram which defines classes and their relationships in a Whole/Part formalism in connection with the subject of the data domain described.

You can connect one or more data domain diagrams to a data domain, according to what you want to describe.

Example of diagram

The following data domain diagram represents a data structure relating to Orders; it describes classes and their relationships in a Whole/Part formalism.



Creating a Logical Data Domain Diagram

To create a data domain diagram from a logical data domain:

1. Right-click the data domain and select **New > Diagram**.
2. Select the diagram type **Data Domain Entity Diagram**.
3. Click **OK**.

Adding an object to the diagram

In the data domain diagram, you can add a new object or connect an existing object.

The objects visible in a data diagram are not automatically linked to the data domain. A command allows you to define the objects as components of the domain. See [Adding a Component to a Data Domain](#).

Adding a class

To add a new class to a diagram:

1. In the diagram insert toolbar, click **Class**, then click in the diagram.
The **Add A Class** dialog box appears.

2. Enter the name of the class and click **Add**.

Add a data view

To add a new data view to a diagram:

1. In the diagram insert toolbar, click **Data View**, then click in the diagram. The **Add Data View** dialog box appears.
2. Enter the name of the data view and click **Add**.
3. The editor view appears. It is used to define the components of the view. See [Logical data views](#).

Adding a Component to a Data Domain

You can connect objects to a data domain through components. A component references an object (class or class view) and defines the type of access to the object in question (read-only, modification, deletion, etc.).

The data domain is attached to the package; objects directly created from components are automatically connected to the package of the data domain.

You can create a component from an object in the diagram or using the properties of the data domain.

To create a component from an object of the data domain diagram:

1. In the diagram, right-click the object in question and select **Add to (name of the data domain)**.

The name of the component created appears in the properties of the data domain. By default it has the name of the object that it references.

Defining the access mode to the component (CRUD)

You can specify the access rights to each component of a data domain by defining the CRUD of the component in question (Create, Read, Update Delete).

To define the access mode to the object in the data domain:

1. Open the properties window of the data domain.
2. Select the **Components** page.
3. Select the line of the component in question. Commands are added, including the **CRUD** button.
4. Click this button.
5. In the window that appears, select or deselect the check boxes associated with the actions: Create, Read, Update, Delete.

The content of the **Data access** column is calculated automatically according to the selected actions. This result appears in object form in the diagram associated with the data domain.

LOGICAL DATA VIEW

A data view enables representation of the scope covered by a data model element. A data view is based on a selection of classes connected to the specific context of the view.

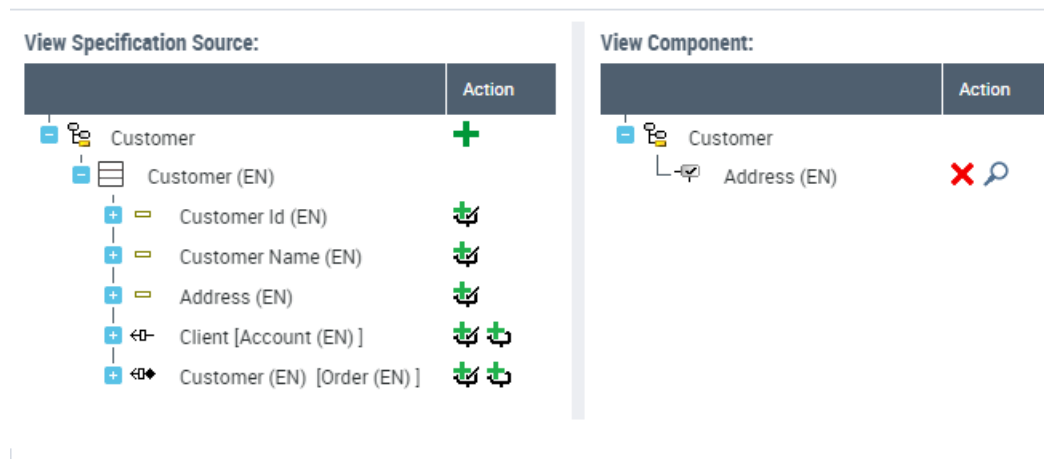
➤ According to the same principle, the design view is used to view the semantic scope of a business object. For more details, see [Concept View](#).

Creating a logical data view

Creating a logical data view consists of:

- defining source objects concerning the view (a class or a data view)
- defining more precisely the properties of source objects to be taken into account in the view (attributes, parts)

For example, for order management, you must retrieve the delivery address available for each client. To take into account this information only, you will create a view on the Client class that takes the "Address" attribute only, without taking into account other attributes that can contain the Client class.



Using the source objects (left tree), you can define embedded components and referenced components in the view.

An embedded component specifies that all the information that comprises the source object is to be taken into account when using the view (for example, the parts and the attributes associated with a class). A referenced component references only the object in the view.

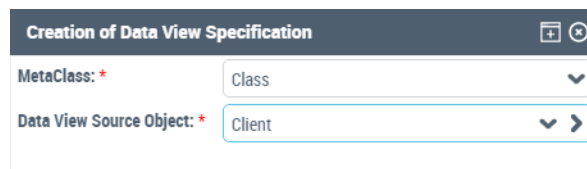
Creating a data view (from a list of views)

To create a data view in **HOPEX Data Governance**:

1. Click the navigation menu then **Architecture > Hierarchy View**.

*☛ In **HOPEX Information Architecture**, click the navigation menu then **Data Architecture > Logical Data Assets**.*

2. In the edit area, expand the **Data Dictionaries** folder.
3. Right-click the data dictionary and click **New > Data view**.
The data view creation dialog box opens.
4. To specify the source object in the data view, click **New**.
5. In the dialog box that appears, enter:
 - the **Type of object** concerned by the view.
 - The **Source object for the data view**.



6. Click **Add**.
7. Repeat the procedure to add other source object if necessary.
8. Click **OK**.
The new view appears in the list of data views.

Creating a data view directly from an object

You can define the source object of a view by creating the view directly on the object in question.

☛ You can subsequently add another object to the view.

To create a data view on an object:

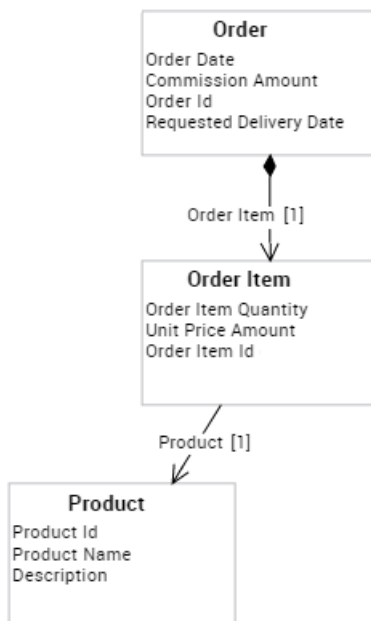
1. Right-click the object concerned and select **New > Data view**.
The data view creation wizard opens.
2. Enter the name of the view.
3. If appropriate, enter the name of the owner.
4. Click **OK**.
The editor view appears.

Displaying source objects in the data view

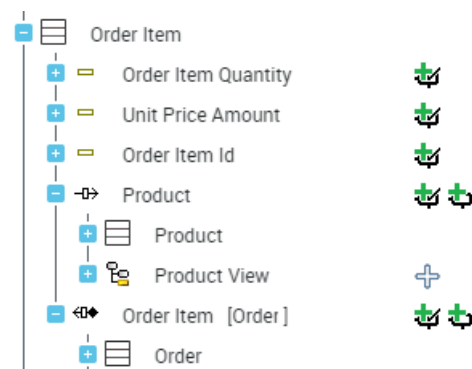
	Class
	Attribute
	Part
	Part (composing class)
	Part (composed class)

Example

Logical model



Logical data view



Defining the Data View Components.

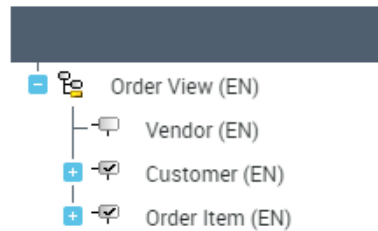
Embedded component

An embedded component brings all the information that makes up the object into the view (for example, the parts and the attributes associated with a class).

To add an embedded component to the view:

1. Open the data view.
2. On the source object side , select the element to add to the data view.
3. Under the **Action** column, click  **Add a View Inclusion Component.**
The object appears to the right of the view editor.

View Component:



Referenced component

By referencing a component in the view, you can display the object in the view, without embedding all its properties.

You can reference the objects that contain a certain amount of information, such as classes, in the view. For attributes, only the inclusion button is available.

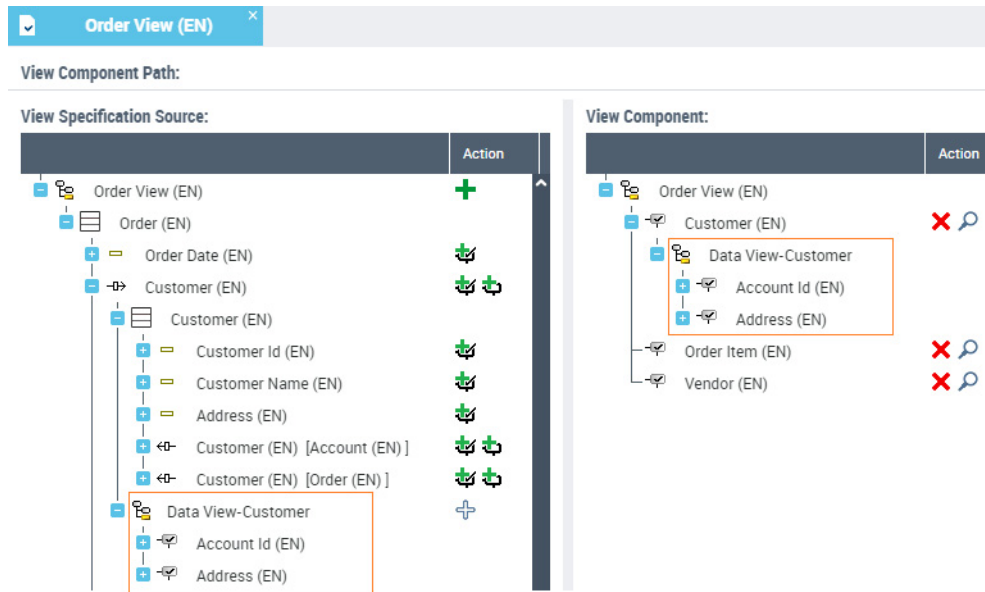
To reference an object in the view:

1. Open the data view.
2. On the source object side , select the element to add to the data view.
3. Under the **Action** column, click  **Add a View Referencing Component.**
The object appears to the right of the view editor.

Using a view in another view

When you embed a class in a data view, all the attributes of the class are added by default to the view. You can limit the list of attributes to those already defined in a view.

Below, only the attributes defined in the "Customer" view (Account Id and Address) are added to the "Order" view.



To add a data view (source) to a data view (target) :

1. Open the target data.
2. On the left part, expand the class concerned by the source view to be added.

The source view has been previously embedded in the target view.

3. Select the source view and under the **Action** column, click **Add a View**.

The view associated with the class appears in the right part of the view editor, under the name of the class in question.

CLASS DIAGRAM

HOPEX Data Governance and **HOPEX Information Architecture** provides two formalisms to describe logical data:

- the data package (represented by a data dictionary), to build class diagrams (UML notation)
- the data model, for data diagrams (standard notations, IDEF1X, I.E, Merise) See [Data Model](#).

Data description in UML notation is carried out in a class diagram.

Creating a Package

A package (or data dictionary) partitions the domain and the associated work. It is designed to contain the modeled elements. Graphical representation of all or of certain of these elements is in a class diagram.

A database can be connected to a data model from the time of its creation. It is on this database that the different data processing tools can then be run (generation, synchronization etc.). The database package is the default owner of the classes and associations represented in the class diagram.

See also: [Data Dictionary](#).

Creating a package (data dictionary)

To create a package with **HOPEX Data Governance** :

1. Click the navigation menu then **Architecture > Hierarchy View**.

 In **HOPEX Information Architecture**, click the navigation menu then **Data Architecture > Logical Data Assets**.

2. In the edit area, right-click the **Data Dictionaries** folder and click **New > Package**.
3. Enter the name of the package.
4. Click **OK**.

Connecting a Package to a Database

To create a package from a database:

1. Click the navigation menu then **Architecture > Hierarchy View**.

 In **HOPEX Information Architecture**, click the navigation menu then **Data Architecture > Physical Data Assets**.

2. In the edit area, under the **Databases** folder, select the desired database.
3. Click the associated **Properties** button.
4. In the database properties, click the **Characteristics** page.
5. Expand the **Data Package** section.
6. Click **New**.
7. Enter the name of the package and click **OK**.

To connect an existing package to a database:

1. In the database properties, click the **Characteristics** page.
2. Expand the **Data Package** section.
3. Click **Connect**.
The query dialog box appears.
4. Click the **Search** button.
5. Select the package and click **OK**.

Creating a Class Diagram

A class diagram is used to represent the static structure of a system, in particular the types of objects manipulated in the system, their internal structure, and the relationships between them.

A class diagram includes:

- Classes, which represent the basic concepts (client, account, product, etc.).
- Parts, which define the relationships between the different classes.
- Attributes which define the characteristics of classes.
- Operations, which can be executed on objects of the class.

☛ *Operations are not taken into account by **HOPEX Information Architecture** tools (synchronization, generation etc.).*

To create the class diagram of a package in **HOPEX Data Governance**:

1. Click the navigation menu then **Architecture > Hierarchy View**.
☛ *In **HOPEX Information Architecture**, click the navigation menu then **Data Architecture > Logical Data Assets**.*
2. In the edit area, expand the **Data Dictionaries** folder.
3. Right-click the package and select **New > Diagram**.
4. Select the **Class Diagram** diagram type.
5. Click **OK**.
The new class diagram opens.

Note that when you create a package from a database, a class diagram is automatically created at the same time.

For more details on building a class diagram, see [The Class Diagram](#).

DATATYPES

A datatype is used to group characteristics shared by several attributes. Data types are implemented as classes.

A datatype package is a reference package owning all or part of the datatypes used in the enterprise. All the other packages are declared as clients of the reference package of datatypes.

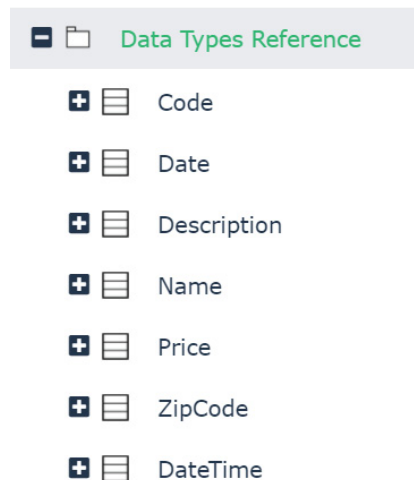
Data type packages

A datatype defines the type of values that a data can have. This can be simple (whole, character, text, Boolean, date, for example) or more elaborate and composite.

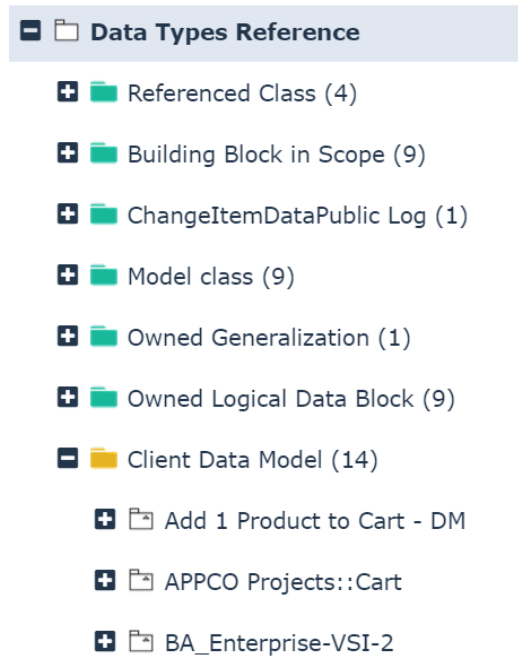
To type attributes of an entity, only datatypes defined for the data model that contains this entity are proposed.

When you create a data model, the "Datatype Reference" datatype package is automatically associated with it by default.

This "Datatype Reference" package owns datatypes "Address", "Code", "Date", etc.



Opening the explorer on this datatype package, you can see that it is referenced by several data models.



The attributes of entities of these models can therefore be typed using the datatypes "Address", "Code", "Date", etc.

Creating a New Datatype Package

You can define a new reference datatype package owning the datatypes used by the enterprise.

To create your own datatype package in **HOPEX Data Governance**:

1. Click the navigation menu then **Architecture > Hierarchy View**.
*In HOPEX Information Architecture, click the navigation menu then **Data Architecture > Logical Data Assets**.*
2. In the edit area, right-click the **Data Dictionaries** folder and click **New**.
3. Enter the package name and click **OK**.
4. Open the properties of the package.
5. Click the **Characteristics** page.
6. Select the "Datatype Package" stereotype.

You can then add types to this package.

Creating a datatype

To create a datatype:

1. Open the **Properties** of the package.

2. Click the **Data Types** page.
3. Click **New**.
The datatype creation dialog box opens.
4. Enter the name of the datatype and click **OK**.

Compound datatype

You can create compound datatypes by adding to them a list of attributes, for example an "Address" type comprising number, street postal code, city and country.

Literal value

You can allocate to a datatype literal values that define the values it can take. Attributes based on such a datatype can take only those values defined by the datatype.

When the new datatype package has been created, it should be referenced on the client data model.

Referencing Datatype Packages

To connect a datatype package to a data dictionary or data model:

1. Open the properties of the data dictionary or data model.
2. Click the **Characteristics** page.
3. Under the **Reference** section, use the first arrow to select **Datatype Package**.
4. With the second arrow, click **Connect Datatype Package**.
The query dialog box appears.
5. Click the **Find** button.
6. Select the desired package and click **OK**.

Assigning Types to Attributes

When the datatype package has been referenced for a data dictionary or a data model, the list of types it contains is available on each attribute of the dictionary classes or model entities. All that is required is to select the one that is suitable.

To define the type of an attribute:

1. Open the **properties** of the class or the entity concerned.
2. Click the **Attributes** page.
3. In the **Datatype (DM)** column corresponding to the attribute, select the desired type in the list.

See also [Data Types and Column Datatypes](#).

DATA MODEL

Data modeling consists of identifying the entities representing the activity of the company, and defining the associations existing between them. The entities and associations in the data diagram associated with a sector of the company must be sufficient to provide a complete semantic description. In other words, one should be able to describe the activity of a company by using only these entities and associations.

This does not mean that each word or verb used in this explanation corresponds directly to an object in the data diagram. It means that one must be able to state what is to be expressed using these entities and associations.

Data model specification is often considered the most important element in modeling of an information system.

To help you describe logical data, **HOPEX Information Architecture** offers a simple notation, based on the data model.

Summary of Concepts

Data model

A data model is used to represent the static structure of a system, particularly the types of objects manipulated in the system, their internal structure, and the relationships between them.

A data model is a set of entities with their attributes, the associations existing between these entities, the constraints bearing on these entities and associations, etc.

Data diagram

A data diagram is a graphical representation of a model or of part of a model.

A data diagram is represented by:

- **Entities**, which represent the basic concepts (customer, account, product, etc.).
- **Associations**, which define the relationships between the different entities.
- **Attributes**, which describe the characteristics of entities and, in certain cases, of associations.

The attribute or set of attributes that enables unique identification of an entity is called an identifier.

The data diagram also contains multiplicity definitions.

Building a data model

Creating a Data Model

Use of data models requires selection of an option. See [Formalisms](#).

To create a data model in **HOPEX Data Governance**:

1. Click the navigation menu then **Architecture > Hierarchy View**.

☛ *In **HOPEX Information Architecture**, click the navigation menu then **Data Architecture > Logical Data Assets**.*

2. In the edit area, display the list of **Data Models**.
3. Click the **New** button.
4. In the dialog box that appears, enter the name of the data model, and an owner if necessary.
5. Click **OK**.

The data model created appears in the list of data models.

Creating a Data Diagram

A data diagram is a graphical representation of a model or of part of a model.

A data diagram is represented by:

- **Entities**, which represent the basic concepts (customer, account, product, etc.).
- **Associations**, which define the relationships between the different entities.
- **Attributes**, which describe the characteristics of entities and, in certain cases, of associations.

The attribute or set of attributes that enables unique identification of an entity is called an identifier.

The data diagram also contains multiplicity definitions.

To create a data diagram:

1. Right-click the data model and select **New > Data Diagram**.
The data diagram opens.

Datatypes

A type is used to group characteristics shared by several attributes.

When you create a data model, the "Datatype Reference" datatype package is automatically connected with it by default. The list of **datatypes** it contains is available on each attribute of entities of the model. You can however assign to it another **datatype package**.

The reference datatype package of a data model is displayed in the properties dialog box of the model, in the **Characteristics** tab.

For more detailed information, see [Data type packages](#).

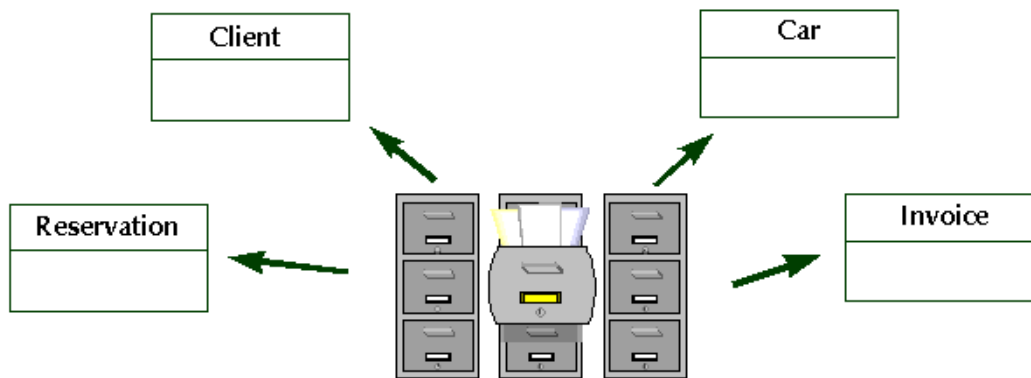
Entities

 An entity groups objects that share the same characteristics and have similar behavior. Entities are management elements considered useful for representing enterprise activity, and are therefore reserved for this purpose. They may, for example, have corresponding tables in a database.

An **entity** is described by a list of attributes.

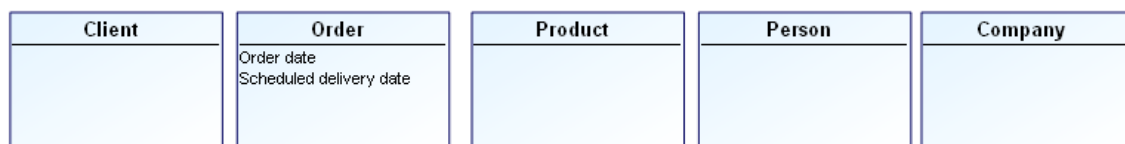
An entity is linked to other entities via associations. The set of entities and associations forms the core of a data model.

We can illustrate the entity concept by comparing entities to index cards filed in drawers.



Entities can represent management objects.

Examples: Customer, Order, Product, Person, Company, etc.



Entities can represent technical objects used in industry.

Examples: Alarm, Sensor, Zone

Creating an entity

To create an entity:

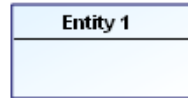
1. In the data diagram insert toolbar, click the **Entity** button 
2. Click in the diagram.
The **Add Entity (DM)** dialog box opens.

3. Enter the entity **Name**.

☛ When the **OK** or **Create** buttons are grayed, this is because the requirements for the dialog box in which they appear have not been completed.

4. Click **Add**.

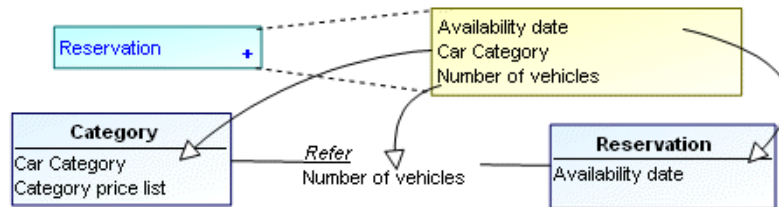
The entity appears in the diagram.



☺ You can create several entities successively without having to click the toolbar each time. To do this, double-click the **Entity** button. To return to normal mode, press <Esc>, or click on another button in the toolbar such as the arrow.

Attributes

Entities and associations can be characterized by *attributes*.



These attributes can be found by studying the content of messages circulating within the enterprise.

📖 An attribute is the most basic data saved in the enterprise information system. An attribute is a property when it describes an entity or association, and an identifier when selected as a means of identification of each instance of an entity.

Examples:

- "Client Name" (property of the client entity).
- "Client No." (identifier of the client entity).
- "Account Balance" (property of the account entity).

An attribute characterizes an association when the attribute depends on all the entities participating in the association.

In the diagram below, the role that a “Consultant” plays in a “Contract” depends on the consultant and on the contract, and therefore on the “Intervene” association.



Creating attributes

To create an attribute on an entity:

1. Right-click the entity and select **Properties**.
The entity properties dialog box opens.
2. Click the drop-down list then **Attributes**.
The Attributes page appears.
3. Click the **New** button.
A default name is automatically proposed for the new attribute. You can modify this name.
4. Click **OK**.

You can specify its **Data type**.

Example: Numeric value.

☛ See [Data Types and Column Datatypes](#) for more details on *data types* that can be assigned to an attribute.

Inherited attributes

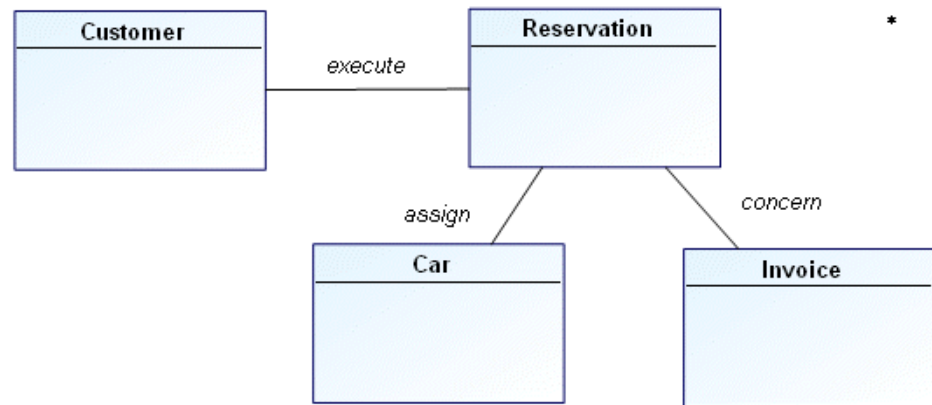
When a generalization exists between a general entity and a more specialized entity, the specialized entity inherits the attributes of the general entity.

See [Generalizations](#).

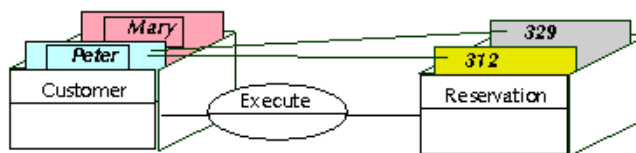
Associations

📖 An association is a relationship that exists between two or more entities. It can carry attributes that characterize the association between these entities

Associations can be compared to links between index cards.



The following drawing provides a three-dimensional view of the situations a data diagram can store.



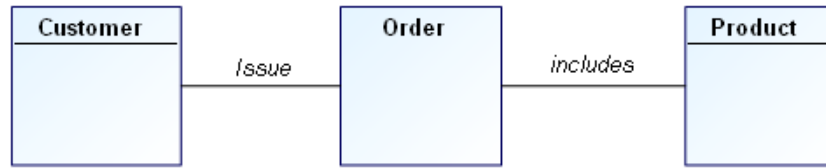
Peter and Mary are clients. Peter has made reservations numbers 312 and 329.

A data diagram should be able to store all situations in the context of the company, but these situations only.

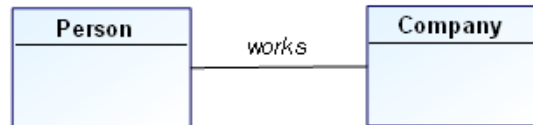
☛ The diagram should not allow representing unrealistic or aberrant situations.

Examples of associations:

- A client issues an order.
- An order includes several products.



- A person works for a company.



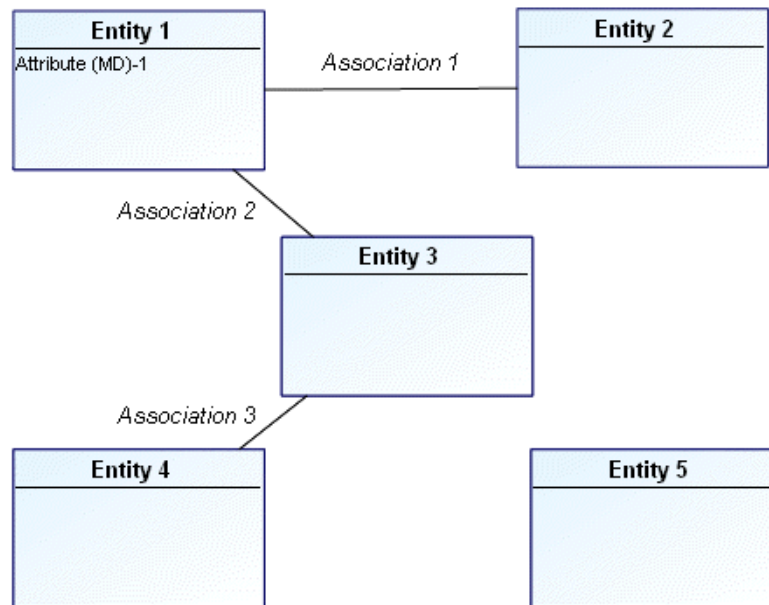
- An alarm is triggered by a sensor.
- A sensor covers a zone.
- A window displays a string of characters.

Creating an Association

To create an association:

1. In the data diagram objects toolbar, click the **Association** button. 
2. Click one of the entities concerned, and holding the mouse button down, drag the mouse to the other entity, before releasing the button.
A line appears in the diagram to indicate the association.
3. To specify the association name, right-click the association and select **Properties**.
Make sure you click on the line indicating the association and not one of the roles located at the ends of the association.
4. In the **Characteristics** page, in the **Local Name** field, enter the association name.
5. Click **OK**.

Example



You can also delete an element or link you created in error by right-clicking it and selecting **Delete**.

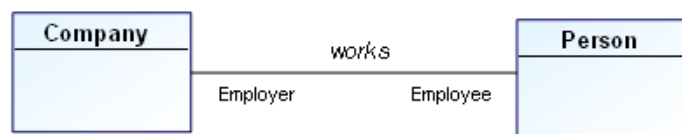
Defining association roles (ends)



A role enables indication of one of the entities concerned by the association. Indication of roles is particularly important in the case of an association between an entity and itself.

Each end of an association specifies the role played by the entity in the association.

The role name is distinguished from the association name in the drawing by its position at the link end. In addition, the role name appears in a normal font, while the association name is italicized.



The status bar (located at the bottom of the window) also allows identification of the different zones: when you move your mouse along the association, it indicates if you are on an association or on a role.

When two entities are linked by only one association, the names of the entities are often sufficient to describe the role. Role names are useful when several associations link the same two entities.

Multiplicities

Each role in an association has an indicated multiplicity to specify how many objects in the entity can be linked to an object in the other entity. Multiplicity is information related to the role and is specified as a completely bounded expression. This is indicated in particular for each role that entities play in an association.

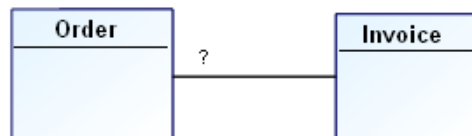
Multiplicity specifies the minimum and maximum number of instances of an entity that can be linked by the association to each instance of the other entity.

The usual multiplicities are "1", "0..1", "*" or "0..*", "1..*", and "M..N" where "M" and "N" are integers:

- The "1" multiplicity indicates that each object of the entity is linked by this association once and once only.
- The "0..1" multiplicity indicates that at most one instance of the entity can be linked by this association.
- The "*" or "0..*" multiplicity indicates that any number of instances of the entity can be linked by the association.
- The "1..*" multiplicity indicates that at least one instance of the entity is linked by the association.
- The "M..N" multiplicity indicates that at least M instances and at most N instances of the entity are linked by the association.

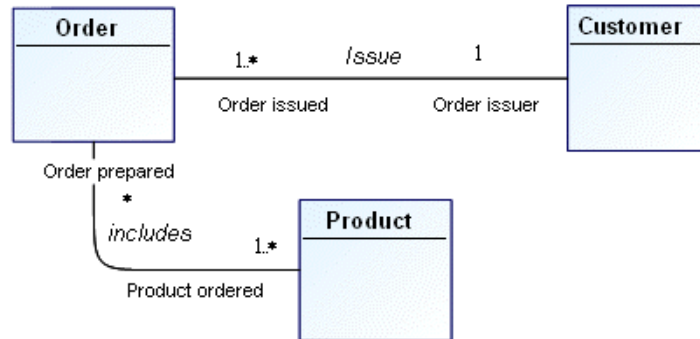
1	One and one only
0 / 1	Zero or one
M..N	From M to N (natural integer)
*	From zero to several
0..*	From zero to several
1..*	From one to several

Example:



0 / 1	An order corresponds to zero or at most one invoice.
*	No restriction is placed on the number of invoices corresponding to an order.
1	Each order has one and only one corresponding invoice.
1..*	Each order has one or more corresponding invoices.

Other examples of multiplicity:

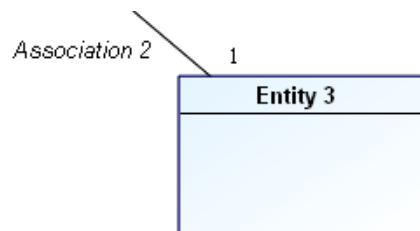


- 1..* A client can issue one or more orders.
- 1 An order is issued by one and only one client.
- 1..* An order contains one or more products.
- *
- 0 / 1 A person works for a company.
- 1..* An alarm is triggered by one or more sensors.
- 1 A sensor covers one and only one zone.
- 1..* A window displays one or more strings.

To specify role multiplicity:

1. In the data diagram, right-click the line between the association and the entity, to open the pop-up menu for the role.
2. Click **Properties**.
The Properties dialog box of the role opens.
3. Click the drop-down list then **Characteristics**.
4. In the **Multiplicity** field, select the required multiplicity.

The representation of the association changes according to its new multiplicities.

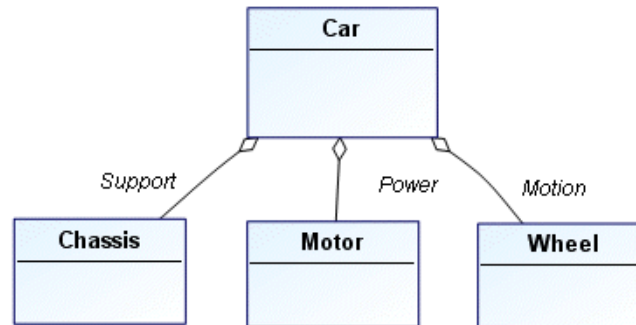


Aggregation

Aggregation is a special form of association, indicating that one of the entities contains the other.

Example of *aggregation*:

A car includes a chassis, an engine, and wheels.



To define the aggregation between the "Car" and "Motor" entities:

1. Right-click the role played by the "Car" entity in its association with the "Motor" entity and select **Properties**.
Role properties appear.
2. Click **Characteristics**.
3. In the **Whole/Part** field, select "Aggregate".
A diamond now appears on the role, representing the aggregation.

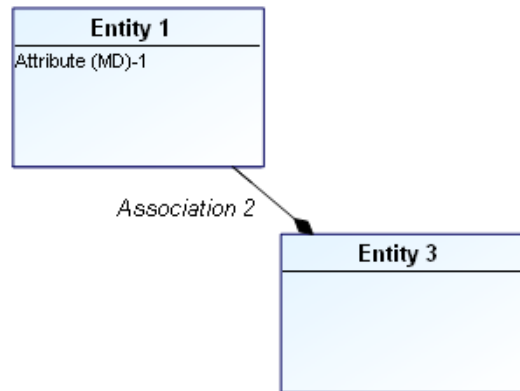
Composition

A composition is a strong aggregation where the lifetime of the components coincides with that of the composite. A composition is a fixed aggregation with a multiplicity of 1.

Example of *composition*:

An order consists of several order lines that will no longer exist if the order is deleted.

Composition is indicated by a black diamond.

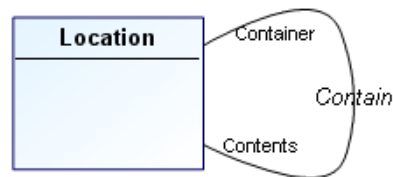


To specify composition of a role:

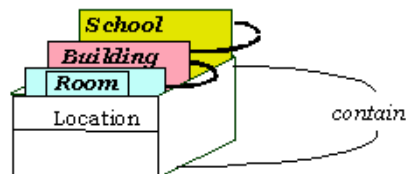
1. Right-click the role and select **Properties**.
Role properties appear.
2. Click the drop-down list then **Characteristics**.
3. In the **Whole/Part** field, select "Composite".

Reflexive Associations

Certain associations use the same entity several times.



A classroom, a building, and a school are all locations.



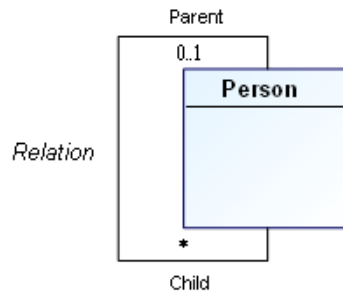
A classroom is contained in a building, which is contained in a school.

A reflexive association concerns the same entity at each end.

To create a reflexive association:

1. In the data diagram objects toolbar, click the **Association** button. 
2. Select the entity concerned and drag the mouse outside the entity, then return inside it and release the mouse button.
The reflexive association appears in the form of a half-circle in a broken line.

☛ If there is association of an entity with itself, the roles need to be named in order to distinguish between the corresponding links in the drawing.



Below, “Parent” and “Child” are the two *roles* played by the “Person” entity in the association.

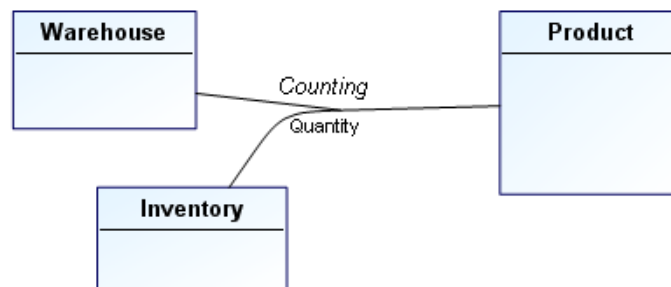
📖 A role enables indication of one of the entities concerned by the association. Indication of roles is particularly important in the case of an association between an entity and itself.

You can segment a line by adding joints to modify its path. You can in particular segment a role to avoid an obstacle for example. You can also change the line to a curve.

“N-ary” Association

Certain associations associate more than two entities. These associations are generally rare.

Example: When taking inventory, a certain quantity of product was counted in each warehouse.



To create a ternary association:

1. In the data diagram, create the association between two entities.
2. Click the **Association Role**  button and connect the third entity to the association.

Constraints



A constraint is a declaration that establishes a restriction or business rule that must be applied on execution of processing.

Most *constraints* involve associations between entities.

Examples of constraints:

The person in charge of a department must belong to the department.

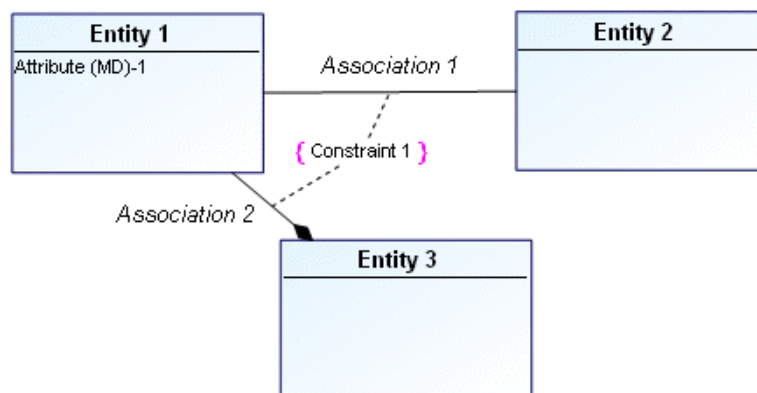
Any invoiced order must already have been delivered.

The delivery date must be later than the order date.

A sensor covering a zone can trigger an alarm for that zone only.

To create a constraint:

1. In the diagram insert toolbar, click the **Constraint**  button.
2. Then click one of the associations concerned by the constraint, and drag the mouse to the second association before releasing the mouse button. The **Add constraint** dialog box opens.
3. Enter the name of the constraint, then click **Add**. The constraint then appears in the drawing.



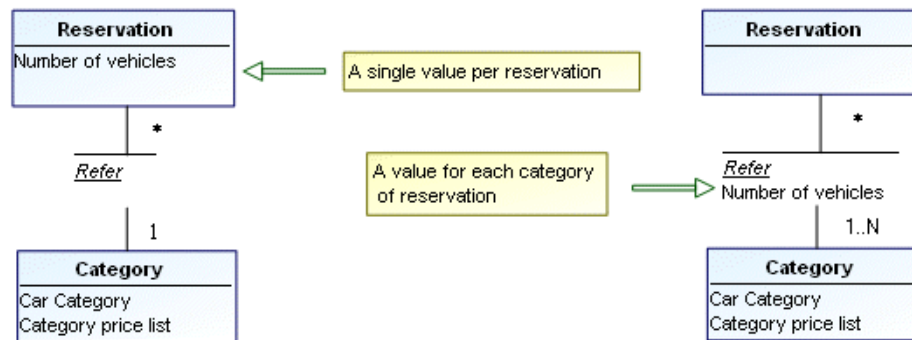
*Save your work regularly using the **Save** button* 

Normalization Rules

Normal forms are rules that are designed to avoid modeling errors. Currently, there are six or seven normal forms. We will discuss the first three.

First Normal Form

Rule: The value of an attribute is uniquely set when the object(s) concerned are known.

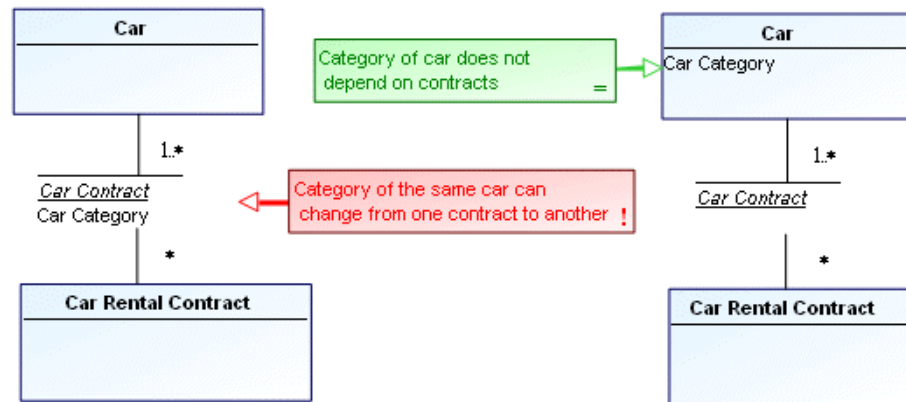


If the number of vehicles is an attribute of the "Reservation" entity, you can only indicate the total number of vehicles for a reservation. You must therefore make one reservation per category of rental vehicle (multiplicity of 1).

If the number of vehicles is an attribute of the association, you can specify the number of vehicles reserved for each category in the association. You can therefore make a single reservation for several categories of vehicles (multiplicity of 1..N).

Second Normal Form

Rule: The value of an association attribute is set only when all the entities concerned are known.

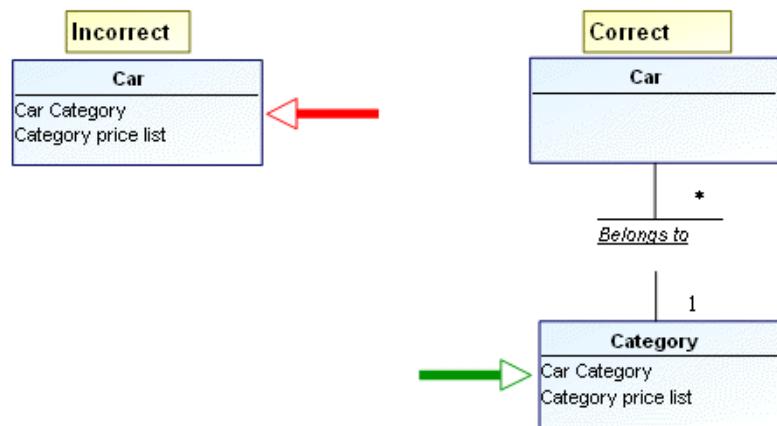


If the car category is an attribute of the "Car Contract" association, this assumes that the car category may change from one contract to the next, which would not be very honest.

If the car category is to be independent of the contract, it must be an attribute of the "Car" entity.

Third Normal Form

Rule: An attribute depends directly and uniquely on the entity it describes.



If the "Category Price List" is an attribute of the "Car" entity, this indicates that two cars in the same category can have a different "Category Price List".

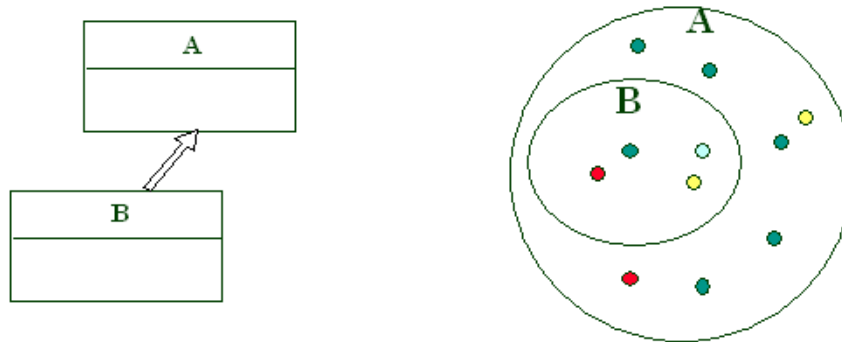
To avoid this, we need to create a "Category" entity that contains the price list.

☺ This rule is used to reveal concepts that were not found during the first draft of the data diagram.

Generalizations

What is a generalization?

 A generalization represents an inheritance relationship between a general entity and a more specific entity. The specific entity is fully consistent with the general entity and inherits its characteristics and behavior. It can however include additional attributes or associations. Any object of the specific entity is also a component of the general entity.



Entity A is a **generalization** of entity B. This implies that all objects in entity B are also objects in entity A. In other words, B is a subset of A. B is then the sub-entity, and A the super-entity.

Example:

A: Person, B: Bostonian.

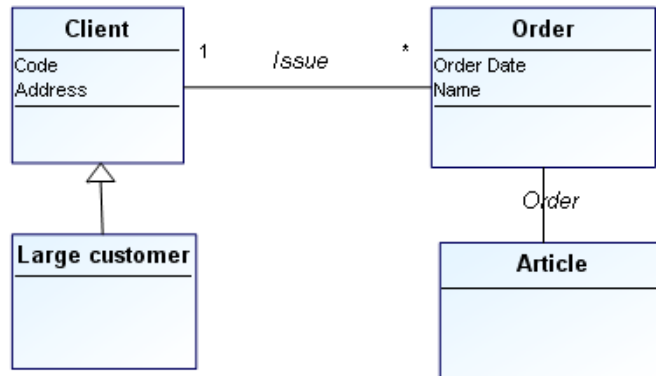
B being a subset of A, the instances of entity B "inherit" the characteristics of those in entity A.

It is therefore unnecessary to redescribe for entity B:

- Its attributes
- Its associations

Example:

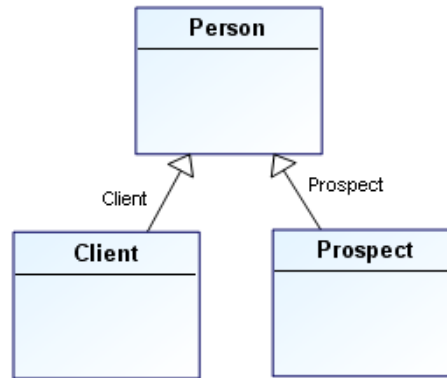
The "Large Client" entity, representing clients with a 12-month revenue exceeding \$1 million, can be a specialization of the Client entity (origin).



In the above example, the associations and attributes specified for "Client" are also valid for "Large client".

Other examples of generalizations:

"prospect" and "client" are two sub-entities of "person".



"export order" is a sub-entity of the "order" entity.

"Individual person" and "corporate person" are two sub-entities of the "person" entity.

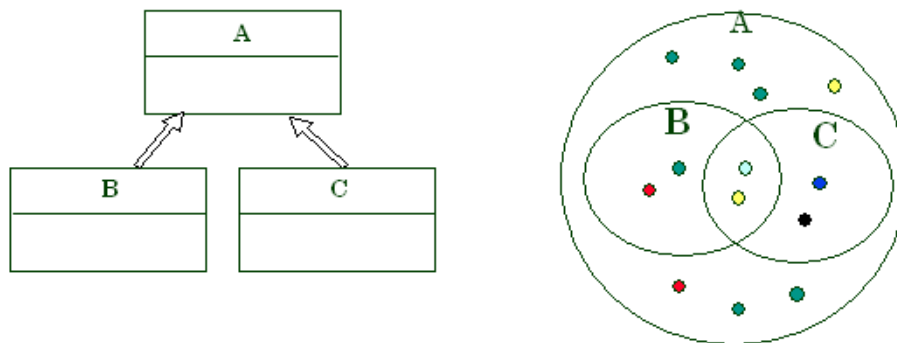
"polygon", "ellipse" and "circle" are sub-entities of the "shape" entity.

"oak", "elm" and "birch" are sub-entities of the "tree" entity.

"motor vehicle", "off-road vehicle" and "amphibious vehicle" are sub-entities of the "vehicle" entity.

"truck" is a sub-entity of the "motor vehicle" entity.

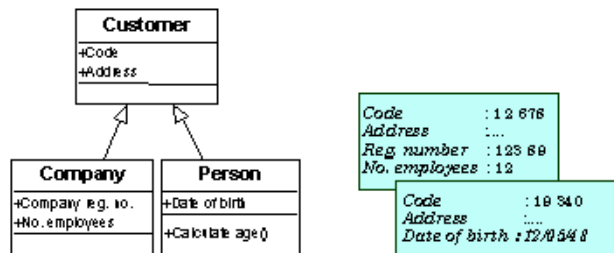
Multiple sub-entities



Several sub-entities of the same entity:

- are not necessarily exclusive.
- do not necessarily partition the set.

Advantages of sub-entities

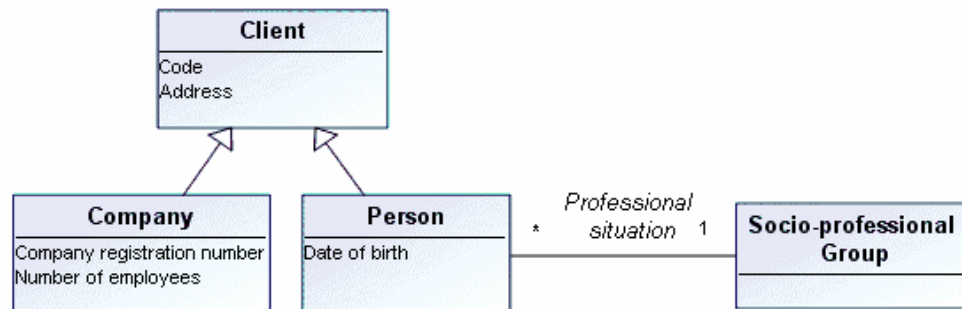


A sub-entity inherits all the attributes and associations of its super-entity, but can have attributes or associations that the super-entity does not have.

A sub-entity can also have specific attributes. These only have meaning for that particular sub-entity. In the above example:

- "Registry number" and "number of employees" only have meaning for a "company".
- "Date of birth" is a characteristic of a "person", not a "company".

A sub-entity can also have specific associations.



- A "person" falls into a "socio-professional group": "manager", "employee", "shopkeeper", "grower", etc. This classification makes no sense for a "company". There is also a classification for companies, but this differs from the one for persons.

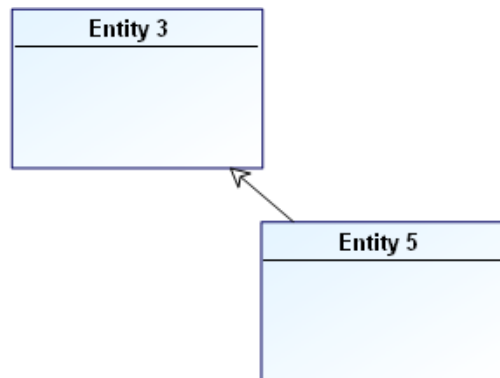
Multiple inheritance

It is sometimes useful to specify that an entity has several super-entities. The sub-entity inherits all the characteristics of both super-entities. This possibility should be used carefully.

Creating a generalization

To create a *generalization*:

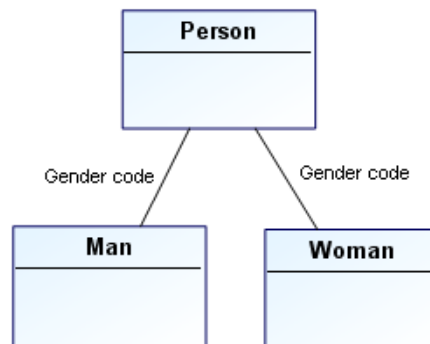
1. In the data diagram insert toolbar, click the **Generalization** button. 
2. Click the sub-entity, in this example "Entity 5", and drag the mouse to the general entity, in this example "Entity 3", then release the button. The generalization is now indicated in the diagram by an arrow.



Discriminator

The discriminator is the general entity attribute whose value partitions the objects into the sub-entities associated with the generalization.

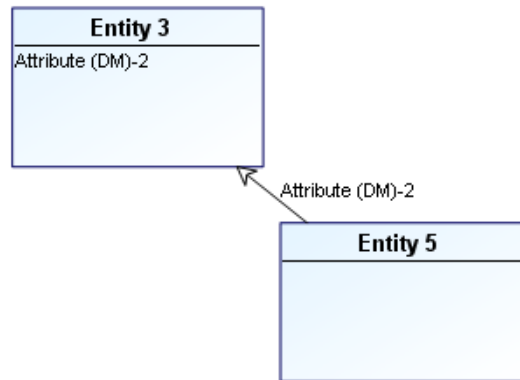
For example, the gender code attribute divides the objects in the person entity into the man and woman sub-entities.



To create a discriminator on a generalization:

1. Open properties of the generalization.
2. Click the drop-down list then **Characteristics**.
3. In the **Discriminator** field, click the arrow and select **Connect Attribute (DM)**.

- Find and select the discriminator among the super-entity attributes. Once selected, the discriminator is displayed on the generalization.

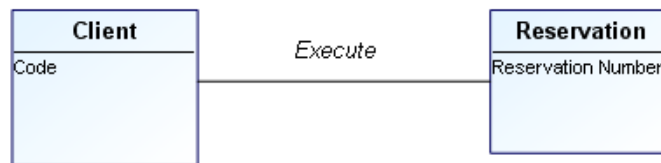


☞ You can also indicate if the generalization is **Complete**: in this case all instances of the generic entity belong to at least one of the category entities of the generalization.

Entity Identifier

Each object has an identity that characterizes its existence. The *identifier* provides an unambiguous way to distinguish any object in an entity. It is one way to distinguish between two objects with identical attribute values.

📖 An identifier consists of one or several mandatory attributes or roles that enable unique identification of an entity.



Customer number 2718 executes Reservation number 314159.

Each entity has a unique identifier whose value can be used to find each of its instances.

By default, the identifier is implicit. In this case a primary key will be automatically generated from the entity name.

Identification by an attribute

It is also possible to select one of the attributes of the entity as its identifier. To do this:

- Open properties of the entity.

2. Click the drop-down list then **Attributes**.
The list of attributes appears.
3. For the chosen attribute, select "Yes" in the **Identifier** column.

Data Model Mapping

Data modeling reflects the activity of an enterprise and is based on the business function history. Differences observed between models are generally cultural or linked to conventions that vary from one person to another and over time. In addition, in expressing a business function requirement, the modeler must take account of what already exists and reconcile different views of the same reality.

Mapping of data models simplifies alignment of this heterogeneous inheritance on a common semantic base.

Functional Objectives

Distinguishing enterprise definitions and business function data

To ensure consistency of business function data, modelers can refer to enterprise definitions serving as the reference framework.

Data model mapping establishes a distinction between enterprise level definitions and business data, while assuring traceability. The Dictionary tool supplements this approach, enabling compilation of business function vocabulary structured as a dictionary.

Integrating existing models

Existing models describing applications assets must be taken into account when creating new models or at the time of a revision project. Requirements vary according to use cases:

- "As-is to-be" type approach: development of a data model is progressive and is based on a stable reference state, which generally corresponds to data of the system in production.
- Software package installation: each software package (PGI, CRM, etc.) imposes its data model, encouraging a trend towards fragmentation and compartmentalization of the IS. Hence the need to have an independent model, linked to the different imposed models.

Mapping of data models is a means of bringing together the data models from different sources.

Use case

A typical case of data mapping occurs in the context of exchanges between applications, each with their own data models. When the number of applications becomes too high, you can install a reference pivot model that will serve as intermediary between the applications and thus avoid multiplication of mappings.

Running the mapping editor

The mapping editor tool is used to align two data models or to map the logical and physical view of a database. It comprises a mapping tree that juxtaposes the views of two models.

You can run the mapping editor from:

- The HOPEX navigation menu
- A data model
- A data package
- A database

To run the mapping editor from the navigation menu:

1. Click the navigation menu then **Tools > Mapping Editor**.
A dialog box appears:
2. Leave the **Create Mapping Tree** default option selected and click **Next**.
3. Indicate the name of the new mapping tree.
4. In the **Nature** list box, select the nature of the tree.
5. In the **Left Object** and **Right Object** frames, from the object types concerned, select the models you wish to align.
6. Click **OK**.
The editor displays the mapping tree juxtaposing the two models.

When the mapping tree has been created, you can subsequently find it in the mapping editor.

Creating a mapping

To create a mapping between two objects:

1. In the mapping editor, successively select the two objects concerned.
2. Click the **Create mapping item** button.

The mapping is created from the last object selected.

Deleting a mapping

To delete a mapping on an object:

1. Select the object in question and click the **Delete mapping item** button.

Mapping details

Objects with mappings are ticked green. When you select one of these objects in the mapping tree, its mapping appears in the details window, which by default is at

the bottom of the mapping editor. It groups the names of connected objects, the object types and comments where applicable.

Catalog DB (Logical/Physical)

Reporting...

	Validity	Frozen
Catalog DB		
Catalog		
Data Diagram	✓	
Catalog	✓	
Contact	✓	
Insurance Product	✓	

MappingItem List

Validity	Logical object	Physical object	Type
Valid	Catalog	Catalog	Entity - Table

Mapping properties

To view mapping properties:

- 1 In the editor details window, select the mapping item and click the **Properties** button.

Object status

Indicators enable indication of status of synchronized objects.

Object status can be characterized as:



Valid



Invalid (when an object has kept a mapping to an object that no longer exists)



No mapping

Mapping source

When you select an object in the tree of one of the models in the editor, you can find its mapping in the other model .

To display an object mapping:

1. Select the object in question.
If there is a mapping item for the object, it is displayed at the bottom of the mapping editor.
2. Select the mapping item and click the **Locate** button 
The mapped objects appear in bold in the editor.

Example of mapping between data models

Different modeling levels can cover distinct requirements. Take the example of two data models. A business function data model "Order Management (DM)" is at conceptual level. It describes at business function level how orders should be managed.

At logical level, the "Order Management (Agency)" data model presents an operational view of IS system data specific to each agency.

We find identical concepts in each of the models. These are however distinct objects.

You can map the two data models to favor cohesion between the business function requirements and the systems that support them.

To do this:

1. Open the **Mapping Editor**.
2. Create a mapping tree.
3. Select the two models to be aligned.
4. Click **OK**.
The editor displays the mapping tree juxtaposing the two models.
5. Create mappings between similar objects and then save.

When models have been mapped, you will know which logical objects is attached to business function objects. You can also analyze the impact of changes carried out at business function level on operational level and vice versa.

IDEF1X NOTATION

Prerequisite

To use the IDEF1X notation, you must select the corresponding option:

1. On the desktop, click **Main Menu > Settings > Options.**
2. In the options navigation tree, expand the **HOPEX Solutions > Information Architecture** folder.
3. Click **Data Notation.**
4. In the right-hand side of the window, select the IDEF1X notation:
5. Click **OK.**

See also [Logical Data Modeling Options](#).

About Data Modeling with IDEF1X

Modeling data consists of identifying management objects (entities) and the associations or relationships between these objects, considered significant for representation of company activity.

IDEF1X is used to produce a graphical information model which represents the structure and semantics of information within an environment or system or an enterprise. Use of this standard permits the construction of semantic data models which may serve to support the management of data as a resource, the integration of information systems, and the building of computer databases.

A principal objective of IDEF1X is to support integration. The IDEF1X approach to integration focuses on the capture, management, and use of a single semantic definition of the data resource referred to as a "Conceptual Schema." The "conceptual schema" provides a single integrated definition of the data within an enterprise which is unbiased toward any single application of data and is independent of how the data is physically stored or accessed. The primary objective of this conceptual schema is to provide a consistent definition of the meanings and interrelationship of data which can be used to integrate, share, and manage the integrity of data. A conceptual schema must have three important characteristics:

- It must be consistent with the infrastructure of the business and be true across all application areas.
- It must be extendable, such that, new data can be defined without altering previously defined data.
- It must be transformable to both the required user views and to a variety of data storage and access structures.

The basic constructs of an IDEF1X model are:

- Things about which data is kept, eg., people, places, ideas, events, etc., represented by a box;
- Relationships between those things, represented by lines connecting the boxes; and
- Characteristics of those things represented by attribute names within the box.

Concept Synthesis

In **HOPEX Information Architecture**, a data model (IDEF1X) is represented by:

- Entities, which represent the basic concepts (client, account, product, etc.).
- Associations, which define relationships between the different entities.
- Attributes which define the characteristics of entities.

The attribute that enables unique identification of an entity is called an identifier.

The data model is completed by definition of multiplicities (or cardinalities).

Creating a Data Model (IDEF1X)

To create a data model:

1. In **HOPEX**, click the **Logical data** navigation pane.
2. In the navigation pane, click **All data models**.
3. In the edit area, click **New**.
The data model mapping creation dialog box opens.
4. Enter the name of the model.
5. Click **OK**.
The data model appears in the list of data models.

Data Diagram (IDEF1X)

A data diagram is a graphical representation of a model or of part of a model.

To create a data diagram:

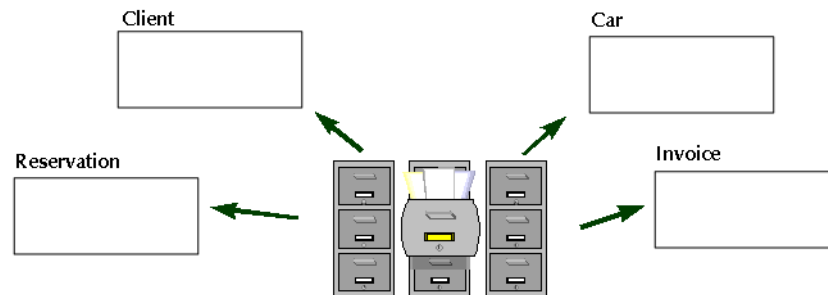
1. Right-click the data model and select **New > Data Diagram (IDEF1X)**.
The data diagram opens.

Entities (IDEF1X)



An entity represents a set of real or abstract things (people, objects, places, events, ideas, combinations of things, etc.) which have common attributes or characteristics. An individual member of the set is referred to as an "entity instance."

You can compare the *entity* concept to sheets in files for example.



An entity represents a particular object class, of which all instances can be described in the same way.

An entity is “independent” if each instance of the entity can be uniquely identified without determining its relationship to another entity. An entity is “dependent” if the unique identification of an instance of the entity depends upon its relationship to another entity.

An entity is represented as a box. If the entity is identifier-dependent, then the corners of the box are rounded.

Creating an entity

To create an entity:

1. Click the **Entity** button  in the diagram objects toolbar.
2. Click in the diagram.
The **Add Entity (DM)** dialog box opens.
3. Enter the entity name.
4. Click **Create** (Windows Front-End) or **Add** (Web Front-End).
The entity appears in the diagram.

Attributes

 *An attribute represents a type of characteristic or property associated with a set of real or abstract things. An instance of an entity will usually have a single specific value for each associated attribute. An attribute or a combination of attributes can be an identifier when selected as a means of identification of each instance of an entity.*

Examples of *attributes*:

- “Client Name” (property of the client entity).
- “Client No.” (identifier of the client entity).
- “Account Balance” (property of the account entity).

Defining attributes

To create an attribute:

1. Right-click the entity and select **Properties**.
The entity properties dialog box opens.
2. Select the **Attributes** tab.
3. To add a new attribute to the entity, click button  .
A default name is automatically proposed for the new attribute. You can modify this name.

You can specify its **Data type**.

Example: Numeric value.



A datatype is used to group characteristics shared by several attributes. Data types are implemented as classes.



*See [Data Types and Column Datatypes](#) for more details on **data types** that can be assigned to an attribute.*

Inherited attributes

When a categorization relationship (generalization) exists between a general entity and a more specialized entity, the specialized entity inherits the attributes of the general entity.

See [Generalizations](#).

Specifying the entity identifier

To specify the entity identifier:

1. Open the properties dialog box of the entity.
2. Select the **Attributes** tab.
3. For the chosen attribute, select "Yes" in the **Identifier** column.



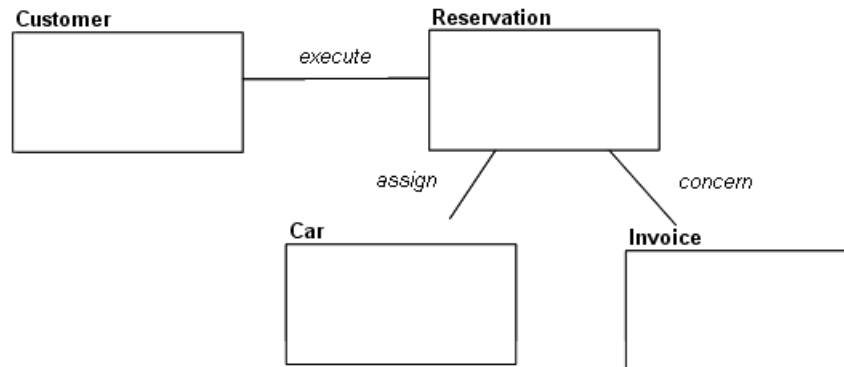
For more details, see [Entity Identifier](#).

Associations (IDEF1X)

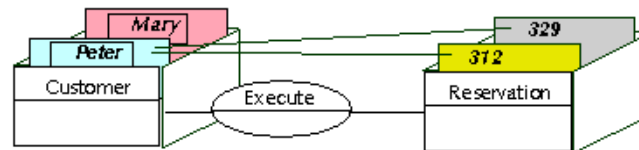


An association is a relationship that exists between two or more entities. It can carry attributes that characterize the association between these entities

Associations can be compared to links between index cards.



The following drawing provides a three-dimensional view of the situations a data diagram can store.



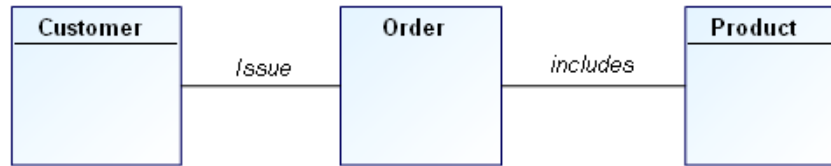
Peter and Mary are clients. Peter has made reservations numbers 312 and 329.

A data diagram should be able to store all situations in the context of the company, but these situations only.

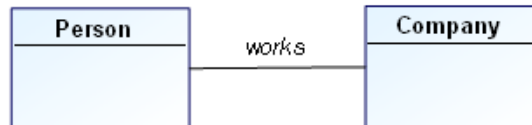
☛ The diagram should not allow representing unrealistic or aberrant situations.

Examples of associations:

- A client issues an order.
- An order includes several products.



- A person works for a company.



- An alarm is triggered by a sensor.
- A sensor covers a zone.
- A window displays a string of characters.

Mandatory identifying relationship



A mandatory identifying relationship is an association between entities in which each instance of one entity is associated with zero, one or more instances of the second entity and each instance of the second entity is associated with one instance of the first entity and identified by this association. The second entity is always an identifier-dependant entity represented by a rounded corner box. The identifying relationship is represented by a solid line with a dot at the dependant entity end of the line.

If an instance of the entity is identified by its association with another entity, then the relationship is referred to as an "identifying relationship", and each instance of this entity must be associated with exactly one instance of the other entity. For example, if one or more tasks are associated with each project and tasks are only uniquely identified within a project, then an identifying relationship would exist between the entities "Project" and "Task". That is, the associated project must be known in order to uniquely identify one task from all other tasks. The child in an identifying relationship is always existence-dependent on the parent, i.e., an instance of the child entity can exist only if it is related to an instance of the parent entity.

To create an *identifying relationship*:

1. In the diagram objects toolbar, click the **Mandatory identifying relationship** button 
2. Click the parent entity, and holding the mouse button down, drag the mouse to the child entity before releasing the button.

The association appears in the diagram. It is represented by a solid line with a dot at the dependent entity end of the line. The shape of the dependent entity is automatically changed to a rounded corner box.



Mandatory Identifying Relationship

In the above example, an order is composed of order lines, and each order line is identified through its association with the order. The order line is a dependent entity represented by a rounded corner box.

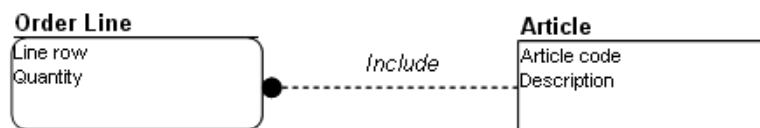
Mandatory non-identifying relationship

A mandatory non-identifying relationship is an association between entities in which each instance of one entity is associated with zero, one or more instances of the second entity and each instance of the second entity is associated with one instance of the first entity but not identified by this association. It is represented by a dashed line with a dot at the dependant entity end of the line.

If every instance of an entity can be uniquely identified without knowing the associated instance of the other entity, then the relationship is referred to as a “non-identifying relationship.” For example, although an existence-dependency relationship may exist between the entities “Buyer” and “Purchase Order”, purchase orders may be uniquely identified by a purchase order number without identifying the associated buyer.

To create a *non-identifying relationship*:

1. In the diagram objects toolbar, click the **Mandatory non-identifying relationship** button
2. Click the parent entity, and holding the mouse button down, drag the mouse to the child entity before releasing the button.
The association appears in the diagram.



Mandatory Non-Identifying Relationship

In the above example, an order include one article, but is not identified through its association with the article.

Mandatory Non-Identifying Relationship

A mandatory non-identifying relationship is an association between entities in which each instance of one entity is associated with zero, one

or more instances of the second entity and each instance of the second entity is associated with one instance of the first entity but not identified by this association. It is represented by a dashed line with a dot at the dependant entity end of the line.

In an **optional non-identifying relationship**, each instance of the child entity is related to zero or one instances of the parent entity.

An optional non-identifying relationship represents a conditional existence dependency. A dashed line with a small diamond at the parent end depicts an optional non-identifying relationship between the parent and child entities.

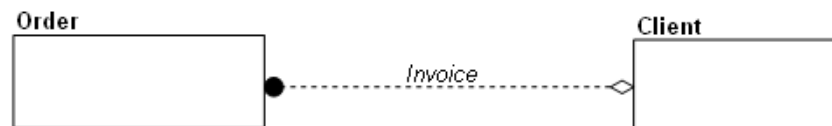
An instance of the child in which each foreign key attribute for the relationship has a value must have an associated parent instance in which the primary key attributes of the parent are equal in value to the foreign key attributes of the child.

To create an optional non-identifying relationship:

1. In the diagram insert toolbar, click the **Optional relationship** button



2. Click the parent entity, and holding the mouse button down, drag the mouse to the child entity before releasing the button.
The association appears in the diagram.



Optional relationship

In the above example, an order should be invoiced to a client, but it is not mandatory (delivery problems, etc.).

non-specific relationship



A non-specific relationship is an association between entities in which each instance of the first entity is associated with zero, one or many instances of the second entity and each instance of the second entity is associated with zero, one or many instance of the first entity. It is depicted as a line drawn between the two associated entities with a dot at each end of the line.

Non-specific relationships are used in high-level Entity-Relationship views to represent many-to-many associations between entities.

In the initial development of a model, it is often helpful to identify “non-specific relationships” between entities. These non-specific relationships are refined in later development phases of the model.

A non-specific relationship, also referred to as a “many-to-many relationship,” is an association between two entities in which each instance of the first entity is associated with zero, one, or many instances of the second entity and each instance of the second entity is associated with zero, one, or many instances of the first entity. For example, if an employee can be assigned to many projects and a project can have many employees assigned, then the connection between the entities

"Employee" and "Project" can be expressed as a non-specific relationship. This non-specific relationship can be replaced with specific relationships later in the model development by introducing a third entity, such as "Project Assignment", which is a common child entity in specific connection relationships with the "Employee" and "Project" entities. The new relationships would specify that an employee has zero, one, or more project assignments. Each project assignment is for exactly one employee and exactly one project. Entities introduced to resolve non-specific relationships are sometimes called "intersection" or "associative" entities.

A non-specific relationship may be further defined by specifying the cardinality from both directions of the relationship.

To create a non-specific relationship:

1. In the diagram insert toolbar, click the **non-specific relationship** button .
2. Click the first entity, and holding the mouse button down, drag the mouse to the second entity before releasing the button. The association appears in the diagram.



In the above example, an article can appear in zero, one or several catalogs and a catalog can contain zero, one or several articles.

Associative entity

 An associative entity is an entity that is introduced to resolve a non-specific relationship or to display attributes as properties of an association.

Non-specific relationships are used in high-level Entity-Relationship views to represent many-to-many associations between entities. In a keybased or fully-attributed view, all associations between entities must be expressed as specific relationships. However, in the initial development of a model, it is often helpful to identify "non-specific relationships" between entities. These non-specific relationships are refined in later development phases of the model.

Entities introduced to resolve non-specific relationships are sometimes called "intersection" or "associative" entities.

To create an *associative entity*:

1. In the diagram objects toolbar, click the **Entity** button .
2. Click in the diagram.
The **Add Entity (DM)** dialog box opens.
3. Enter the associative entity name.

4. Click **Create** (Windows Front-End) or **Add**(Web Front-End). The entity appears in the diagram.
5. Click the **Mandatory identifying relationship**  button.
6. Click the first entity, and holding the mouse button down, drag the mouse to the associative entity before releasing the button. The association appears in the diagram. The shape of the associative entity changes for the a rounded corner box indicating that it is a dependent entity.
7. Create in the same way the second association by clicking the second entity, and holding the mouse button down, dragging the mouse to the associative entity before releasing the button.

 You can add attributes to the associative entity.



Associative entity

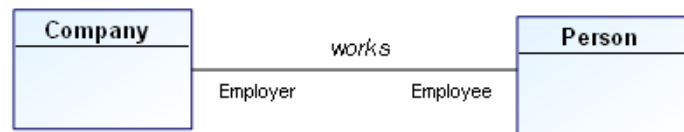
In the above example, an article can be discounted for zero, one or several clients and a client can have discounts for zero, one or several articles. In each case, the discount rate is indicated on the associative class.

Defining Association Roles

 A role enables indication of one of the entities concerned by the association. Indication of roles is particularly important in the case of an association between an entity and itself.

Each end of an association specifies the role played by the entity in the association.

The role name is distinguished from the association name in the drawing by its position at the link end. In addition, the role name appears in a normal font, while the association name is italicized.



 The status bar (located at the bottom of the window) also allows identification of the different zones: when you move your mouse along the association, it indicates if you are on an association or on a role.

When two entities are linked by only one association, the names of the entities are often sufficient to describe the role. Role names are useful when several associations link the same two entities.

Certain associations may associate more than two entities. These associations are generally rare.

To add a role to an association:

1. Click on the **Association Role** button  and connect the association to the entity.

Multiplicities

Each role in an association has an indicated multiplicity to specify how many objects in the entity can be linked to an object in the other entity. Multiplicity is information related to the role and is specified as a completely bounded expression. This is indicated in particular for each role that entities play in an association.

Multiplicity specifies the minimum and maximum number of instances of an entity that can be linked by the association to each instance of the other entity.

The usual multiplicities are "1", "0..1", "*" or "0..*", "1..*", and "M..N" where "M" and "N" are integers:

- The "1" multiplicity indicates that each object of the entity is linked by this association once and once only.
It is represented as a mandatory relationship with a dot on the role and no dot on the opposite role.
- The "0..1" multiplicity indicates that at most one instance of the entity can be linked by this association.
It is pictured by a "Z" (for zero) on the role.
- The "*" or "0..*" multiplicity indicates that any number of instances of the entity can be linked by the association.
This is the default visibility.
- The "1..*" multiplicity indicates that at least one instance of the entity is linked by the association.
It is pictured by a "P" (for positive) on the role.
- The "M..N" multiplicity indicates that at least M instances and at most N instances of the entity are linked by the association.

1	One and one only
0 / 1	Zero or one (Z)
M..N	From M to N (natural integer)
*	From zero to several
0..*	From zero to several
1..*	From one to several (P)

To specify role multiplicity:

1. Right-click the line between the association and the entity, to open the pop-up menu for the role.
2. Click **Properties**.
The properties page of the role opens.
3. Click the **Characteristics** tab.
4. In the **Multiplicity** field, select the required multiplicity.

The representation of the association changes according to its new multiplicities.

☛ In HOPEX Windows Front-End, multiplicity is also displayed in the role's pop-up menu. If the menu you see does not propose multiplicity, check that you clicked on that part of the line indicating the role and not the association.

Categorization Relationships (Generalizations) - (IDEF1X)

📖 A generalization represents an inheritance relationship between a general entity and a more specific entity. The specific entity is fully consistent with the general entity and inherits its characteristics and behavior. It can however include additional attributes or associations. Any object of the specific entity is also a component of the general entity.

What is a Categorization (Generalization)?

Categorization relationships are used to represent structures in which an entity is a "type" (category) of another entity.

Entities are used to represent the notion of "things about which we need information." Since some real world things are categories of other real world things, some entities must, in some sense, be categories of other entities. For example, suppose employees are something about which information is needed.

Although there is some information needed about all employees, additional information may be needed about salaried employees which differs, from the additional information needed about hourly employees. Therefore, the entities "Salaried employee" and "Hourly employee" are categories of the entity "Employee". In the IDEF1X notation, they are related to one another through categorization relationships (*generalization*).

In another case, a category entity may be needed to express a relationship which is valid for only a specific category, or to document the relationship differences among the various categories of the entity. For example, a "Full-time employee" may qualify for a "Benefit", while a "Part-time employee" may not.

A "categorization relationship" or "generalization" is a relationship between one entity, referred to as the "generic entity", and another entity, referred to as a "category entity" or "specialized entity". Cardinality is not specified for the category entity since it is always zero or one.

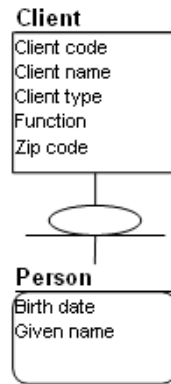
Category entities are also always identifier-dependent.

Creating a Categorization

To create a categorization relationship:

1. Click the **Generalization**  button in the objects toolbar.

2. Click the category entity, drag the mouse to the generic entity, then release the button.
The generalization is pictured in the diagram by an underlined circle, connected by a line to the generic entity and by another line to the category entity.



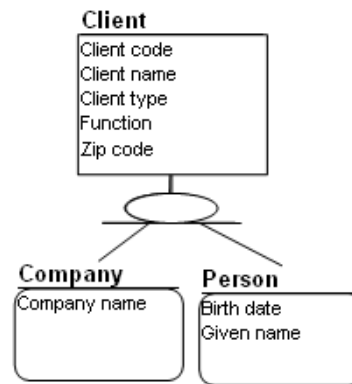
Categorization relationship

In the above example, attributes are interesting on persons that are of no avail for other categories of clients. Person is a dependent entity represented by a rounded corner box.

Multiple Categories

A "category cluster" is a set of one or more categorization relationships. An instance of the generic entity can be associated with an instance of only one of the category entities in the cluster, and each instance of a category entity is associated with exactly one instance of the generic entity. Each instance of the category entity represents the same real-world thing as its associated instance in the generic entity. From the example above, EMPLOYEE is the generic entity and SALARIED-EMPLOYEE and HOURLY-EMPLOYEE are the category entities. There are two categorization

relationships in this cluster, one between "Employee" and "Salaried employee" and one between "Employee" and "Hourly employee".



Multiple Categories

In the above example, companies and persons are two categories of clients.

Multiple Category Clusters

Since an instance of the generic entity cannot be associated with an instance of more than one of the category entities in the cluster, the category entities are mutually exclusive. In the example, this implies that an employee cannot be both salaried and hourly. However, an entity can be the generic entity in more than one category cluster, and the category entities in one cluster are not mutually exclusive with those in others. For example, "Employee" could be the generic entity in a second category cluster with "Female employee" and "Male employee" as the category entities. An instance of "Employee" could be associated with an instance of either "Salaried employee" or "Hourly employee" and with an instance of either "Female employee" or "Male employee".

Complete Categorization

In a "complete category cluster", every instance of the generic entity is associated with an instance of a category entity, ie., all the possible categories are present. For example, each employee is either male or female, so the second cluster is complete. In an "incomplete category cluster", an instance of the generic entity can exist without being associated with an instance of any of the category entities, ie., some categories are omitted. For example, if some employees are paid commissions rather than an hourly wage or salary, the first category cluster would be incomplete.

It is possible to specify whether a categorization relationship is complete or not in the **Characteristics** tab of the generalization properties dialog box. If the value of the characteristic **Complete** is set to "Yes", then all instances of the generic entity belong to at least one of the category entities of the generalization.

Discriminator

An attribute in the generic entity, or in one of its ancestors, may be designated as the discriminator for a specific category cluster of that entity. The value of the discriminator determines the category of an instance of the generic. In the previous example, the discriminator for the cluster including the salaried and hourly categories might be named "Employee type". If a cluster has a discriminator, it must be distinct from all other discriminators.

To create a discriminator on a generalization:

1. Open properties of the generalization.
2. Click **Characteristics**.
3. In the **Discriminator** field, choose the discriminator among the super-entity attributes.
Once selected, the discriminator is displayed on the generalization.

I.E. NOTATION

Prerequisite

To use the I.E. notation, you must select the corresponding option:

1. On the desktop, click **Main Menu > Settings > Options.**
2. In the options navigation tree, expand the **HOPEX Solutions > Information Architecture** folder.
3. Click **Data Notation.**
4. In the right-hand side of the window, select the I.E. notation:
5. Click **OK.**

See also [Logical Data Modeling Options.](#)

About Data Modeling with I.E.

"Information Engineering" was originally developed by Clive Finkelstein in Australia the late 1970's. He collaborated with James Martin to publicize it in the United States and Europe.

Information Engineering is an integrated and evolving set of tasks and techniques for business planning, data modeling, process modeling, systems design, and systems implementation. It enables an enterprise to maximize its resources - including capital, people and information systems - to support the achievement of its business vision.

Business-driven Information Engineering is one of the dominant systems development methodologies used world-wide, as organizations position themselves to compete in the turbulent 1990s and beyond.

Its focus is on data before process, which ensures that organizations identify "what" is required by the business before analysis of "how" it will be provided. IE provides a rich set of techniques for strategic business analysis not reflected in "process first" methodologies.

Information Engineering guides the organization through a series of defined steps that allow it to identify all information important to the enterprise and establish the relationships between those pieces of information. As a result, information needs are defined clearly based on management input, and can be translated directly into systems that support strategic plans.

Most information systems development during the past 25 years has been done from a "stovepipe" or application-specific perspective. The result is that many organizations have separate systems that are incapable of sharing data. In this situation, systems cannot begin to meet their potential and can actually become a burden on the business. IE clearly identifies data sharing requirements throughout

the organization so that systems can be integrated accordingly.

Using IE, organizations have a stable yet flexible framework on which subsequent development activities can be based. This eliminates redundancy and leads to the reuse of program modules and the sharing of data required throughout the business, which helps alleviate the maintenance burden.

Modeling data consists of identifying management objects (entities) and the associations or relationships between these objects, considered significant for representation of company activity.

I.E. is used to produce a graphical information model which represents the structure and semantics of information within an environment or system or a company. Use of this standard permits the construction of semantic data models which may serve to support the management of data as a resource, the integration of information systems, and the building of computer databases.

The basic constructs of an Information Engineering data model are:

- Things about which data is kept, eg., people, places, ideas, events, etc., represented by a box;
- Relationships between those things, represented by lines connecting the boxes; and
- Characteristics of those things represented by attribute names within the box.

Concept Synthesis

In **HOPEX Logical Data** a data model (I.E) is represented by:

- Entities, which represent the basic concepts (client, account, product, etc.).
- Associations, which define relationships between the different entities.
- Attributes which define the characteristics of entities.

The attribute that enables unique identification of an entity is called an identifier.

The data model is completed by definition of multiplicities (or cardinalities).

Creating a Data Model (I.E)

An I.E data model shows entity-types as square cornered boxes (an entity is any person or thing about which data is stored.) The entity types are associated with one another; for example, a "Product" entity is purchased by a "Customer" entity. Lines linking the boxes show these associations. The lines have cardinality (multiplicity) indicators.

To create a data model:

1. In **HOPEX**, click the **Logical data** navigation pane.
2. In the navigation pane, click **All data models**.
3. In the edit area, click **New**.
The data model mapping creation dialog box opens.
4. Enter the name of the model.
5. Click **OK**.
The data model appears in the list of data models.

Data Diagram (I.E.)

A data diagram is a graphical representation of a model or of part of a model. The creation of a diagram varies slightly depending on whether you are in Windows Front-End or Web Front-End.

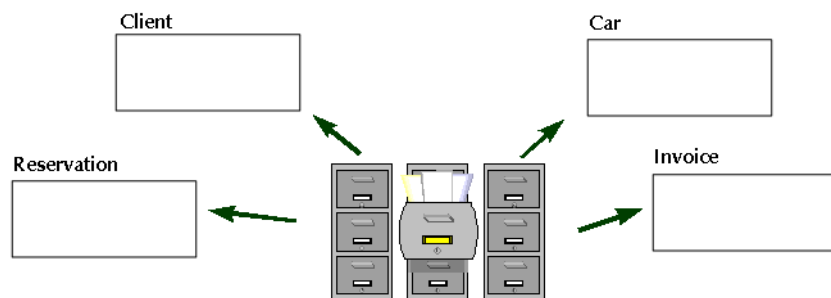
To create a data diagram:

1. Right-click the data model and select **New > Data Diagram (I.E)**.
The data diagram opens.

Entities (I.E.)

 An entity represents a person, place, thing or concept that has characteristics of interest to the enterprise. An entity has various attributes that can be stored in the system. Example: Customer, Employee, Order, Invoice, etc.

You can compare the *entity* concept to sheets in files for example.



An entity represents a particular object class, of which all instances can be described in the same way. An entity is represented as a square cornered box.

Creating an entity

To create an entity:

1. Select the **Entity** button  in the objects toolbar by clicking it with the left mouse button.

2. Click in the diagram.
The **Add Entity (DM)** dialog box opens.
3. Enter the entity name.
4. Click **Create** (Windows Front-End) or **Add**(Web Front-End).
The entity appears in the diagram.

Attributes

Examples of *attributes*:

- "Client Name" (property of the client entity).
- "Client No." (identifier of the client entity).
- "Account Balance" (property of the account entity).

 *An attribute represents a type of characteristic or property associated with a set of real or abstract things. An instance of an entity will usually have a single specific value for each associated attribute. An attribute or a combination of attributes can be an identifier when selected as a means of identification of each instance of an entity.*

Defining attributes

To create an attribute:

1. Right-click the entity and select **Properties**.
The entity properties dialog box opens.
2. Select the **Attributes** tab.
3. To add a new attribute to the entity, click button  .
A default name is automatically proposed for the new attribute. You can modify this name.

You can specify its **Data type**.

Example: Numeric value.

 *A datatype is used to group characteristics shared by several attributes. Data types are implemented as classes.*

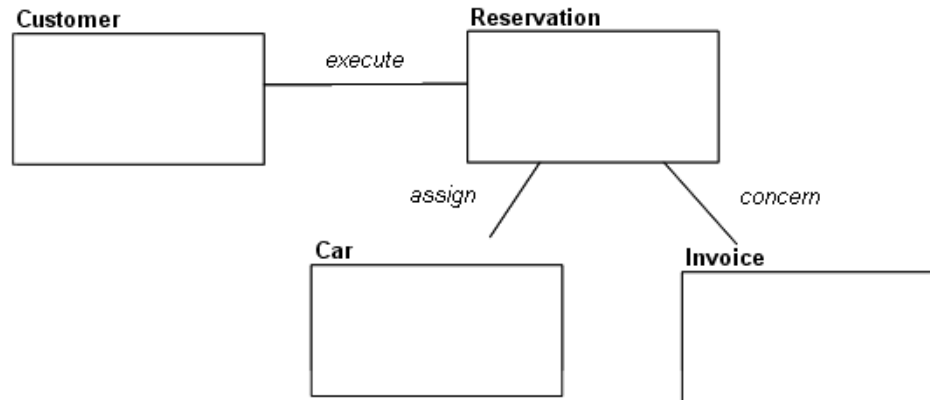
 See [Data Types and Column Datatypes](#) for more details on *data types* that can be assigned to an attribute.

Associations (I.E)

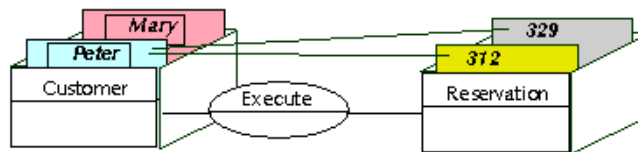
 *An association is a meaningful link between two objects. Associations are used to capture data about the relationship between two objects.*

Overview

Associations can be compared to links between index cards.



The following drawing provides a three-dimensional view of the situations a data diagram can store.



Peter and Mary are clients. Peter has made reservations numbers 312 and 329.

Associations and their Multiplicities

Each role in an association has an indicated multiplicity to specify how many objects in the entity can be linked to an object in the other entity. Multiplicity is information related to the role and is specified as a completely bounded expression. This is indicated in particular for each role that entities play in an association.

To indicate that a role is optional, a circle "O" is placed at the other end of the line, signifying a minimum multiplicity of 0.

To indicate that a role is mandatory, a stroke "|" is placed at the other end of the line, signifying a minimum multiplicity of 1.

A crow's-foot is used for a multiplicity of "many".

In conjunction with a multiplicity of 0 or 1, a stroke "|" is often used to indicate a maximum multiplicity of 1.

With this arrangement, the combination "O|" indicates "at most one" and the combination "| |" or just a single "|" indicates "exactly one".

Mandatory relationship

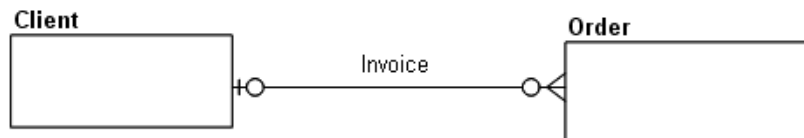
 A mandatory relationship means that each instance of the first entity is associated with exactly one instance of the second entity and that the second entity can be associated with zero, one or many instances of the first entity.



In the above example, a client can issue zero, one or many orders, but an order is always issued by one and only one client.

Optional relationship

 An optional relationship means that each instance of the first entity is associated with zero or one instance of the second entity and that the second entity can be associated with zero, one or many instances of the first entity.



In the above example, a client can be invoiced for zero, one or many orders, and an order should be invoiced to a client, but it is not mandatory (delivery problems, etc.).

non-specific relationship

 A non-specific relationship means that each instance of the first entity is associated with zero, one or many instances of the second entity and that the second entity can be associated with zero, one or many instances of the first entity.



In the above example, an article can appear in zero, one or several catalogs and a catalog can contain zero, one or several articles.

Creating an Association

To create an association:

1. Select the type of association by clicking the corresponding button ,  or  in the objects toolbar.
2. Click one of the entities concerned, and holding the mouse button down, drag the mouse to the other entity, before releasing the button. The **Add Association** dialog box opens.
3. Enter the name of the association, then click **Create**.

The association appears in the diagram.

To modify role multiplicity:

1. Right-click the line between the association and the entity, to open the pop-up menu for the role.
2. Click **Properties**.
The properties page of the role opens.
3. Click the **Characteristics** tab.
4. In the **Multiplicity** field, select the required multiplicity.

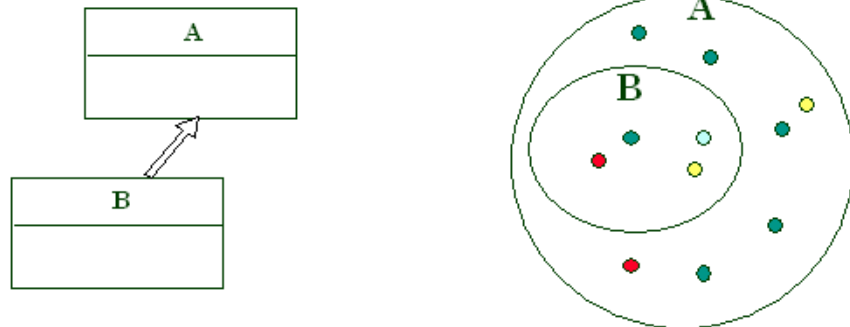
The representation of the association changes according to its new multiplicities.

 In HOPEX Windows Front-End, multiplicity is also displayed in the role's pop-up menu. If the menu you see does not propose multiplicity, check that you clicked on that part of the line indicating the role and not the association.

Sub-types (I.E)

 A generalization represents an inheritance relationship between a general entity and a more specific entity. The specific entity is fully consistent with the general entity and inherits its characteristics and behavior. It can however include additional attributes or associations. Any object of the specific entity is also a component of the general entity.

What is sub-type?



An entity B is a *subtype* of entity A. This assumes that all instances of entity B are also instances of entity A. In other words, B is a subset of A. B is then the subtype, and A the supertype.

Example:

A: Person, B: Bostonian.

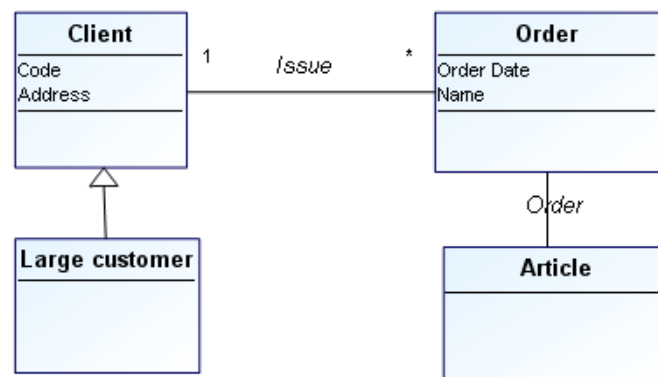
B being a subset of A, the instances of entity B "inherit" the characteristics of those in entity A.

It is therefore unnecessary to redescribe for entity B:

- Its attributes
- Its associations

Example:

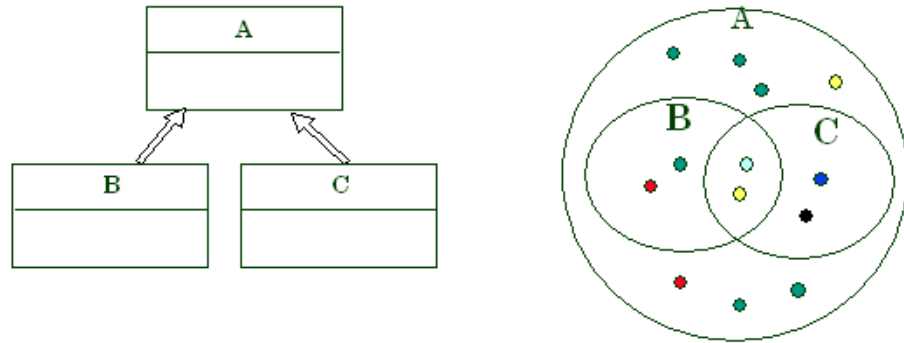
The "Large Client" entity, representing clients with a 12-month revenue exceeding \$1 million, can be a subtype of the Client entity.



A subtype inherits all attributes, associations, roles and constraints of its supertype, but it can also have attributes, associations, roles or constraints that its supertype does not have.

In the above example, the attributes, associations, roles and constraints specified for "Client" are also valid for "Large Client".

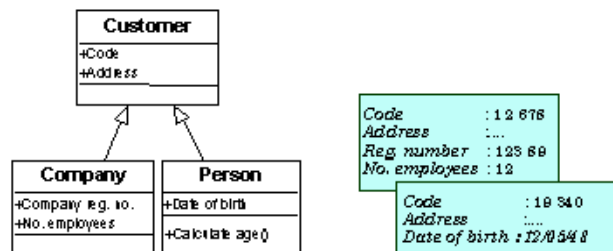
Multiple Subtypes



Several subtypes of the same entity:

- are not necessarily exclusive.
- do not necessarily partition the type.

Advantages of sub-types

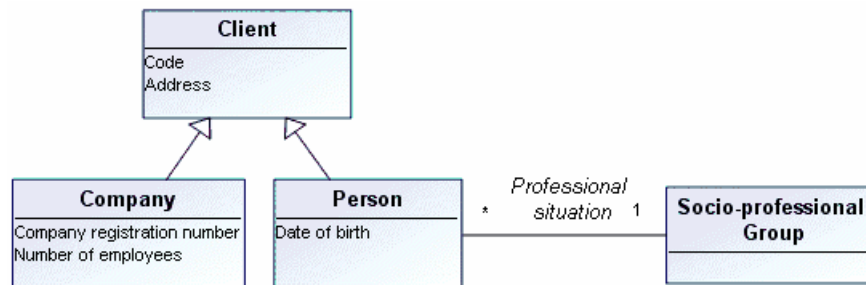


A subtype entity inherits all the attributes and associations of its supertype entity, but can have attributes or associations that the supertype entity does not have.

A subtype entity can also have specific attributes. These only have meaning for that particular sub-entity. In the above example:

- "Registry number" and "number of employees" only have meaning for a "company".
- "Date of birth" is a characteristic of a "person", not a "company".

A subtype entity can also have specific associations.



- A "person" falls into a "socio-professional group": "manager", "employee", "shopkeeper", "grower", etc. This classification makes no sense for a "company". There is also a classification for companies, but it differs from that for persons.

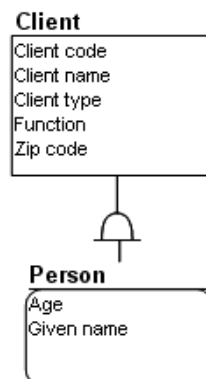
Multiple inheritance

It is sometimes useful to specify that an entity has several supertypes. The subtype inherits all the characteristics of both supertypes. This possibility should be used carefully.

Creating a sub-type

To create a subtype:

1. Click the **Generalization**  button in the objects toolbar.
2. Click the subtype entity, drag the mouse to the supertype entity, then release the button.
The generalization is now pictured in the diagram by an underlined semicircle connected by a line to the supertype entity and by another line to the subtype entity.



In the above example, attributes are interesting on persons that are of no avail for other categories of clients. The subtype entity is represented by a rounded corner box.

THE MERISE NOTATION

Prerequisite

To use the Merise notation, you must select the corresponding option:

1. On the desktop, click **Main Menu > Settings > Options.**
2. In the options navigation tree, expand the **HOPEX Solutions > Information Architecture** folder.
3. Click **Data Notation.**
4. In the right-hand side of the window, select the Merise notation:
5. Click **OK.**

See also [Logical Data Modeling Options.](#)

About Data Modeling

Modeling data consists of identifying management objects (entities) and the associations or relationships between these objects, considered significant for representation of company activity.

The entities, associations and properties that constitute the data model associated with a sector of the company must be sufficient to provide a complete semantic description.

In other words, one should be able to describe the activity of a company by using only the entities, associations and properties that have been selected.

This does not mean that there will be a direct equivalent in the data model for each word or verb in the explanation. It means one must be able to state what is to be expressed, using these entities, associations and properties.

Concept Synthesis

In **HOPEX Information Architecture**, a data model (Merise) is represented by:

- Entities, which represent the basic concepts (client, account, product, etc.).
- Associations, which define relationships between the different entities.
- Attributes (information or properties), which define the characteristics of entities and in certain cases, associations.

The attribute that enables unique identification of an entity is called an identifier.

The data model is completed by definition of cardinalities.

Creating a Data Model (Merise)

To create a data model:

1. In **HOPEX Information Architecture**, click **Logical Data** in the navigation pane.
2. In the navigation pane, click **All data models**.
3. In the edit area, click **New**.
The data model mapping creation dialog box opens.
4. Enter the name of the model.
5. Click **OK**.
The data model appears in the list of data models.

Data Diagram (Merise)

A data diagram is a graphical representation of a model or of part of a model. The creation of a diagram varies slightly depending on whether you are in Windows Front-End or Web Front-End.

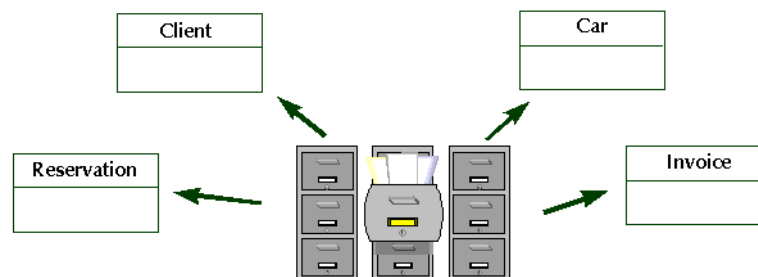
To create a data diagram:

1. Right-click the data model and select **New > Data Diagram (Merise)**.
The data diagram opens.

The entities (Merise)

 An entity is a management object considered of interest in representing enterprise activity. An entity is described by a list of informations (properties) linked to the entity. An entity is linked to other entities via associations. The set of entities and associations forms the core of a data model.

You can compare the *entity* concept to sheets in files for example.

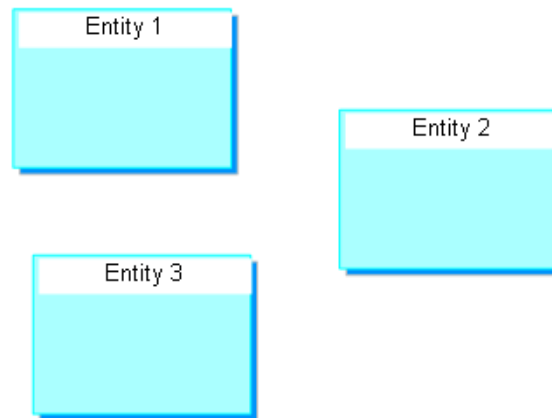


An entity represents a particular object class, of which all instances can be described in the same way.

Creating an entity

To create an entity:

1. Click the **Entity**  button in the diagram objects toolbar.
2. Click in the diagram.
The **Add Entity (DM)** dialog box opens.
3. Enter the entity name.
4. Click **Create** (Windows Front-End) or **Add** (Web Front-End).
The entity appears in the diagram.



☛ To continue creating org-units without having to keep clicking on the toolbar, double-click button . To return to normal mode, press the Esc key or click on a different button in the toolbar, such as the arrow .

☛ The objects you have created, and their characteristics and links, are saved automatically each time the pointer changes to the shape . The diagram drawing is not saved until you explicitly request this by clicking the **Save** button .

Specifying the entity identifier

To specify the entity identifier:

1. Open the properties dialog box of the entity.
2. Select the **Attributes** tab.
3. For the chosen attribute, select "Yes" in the **Identifier** column.

☛ For more details, see [Entity Identifier](#).

The associations (Merise)

An association is a relationship that exists between two or more entities. An association is said to be binary when it connects two entities, ternary when it

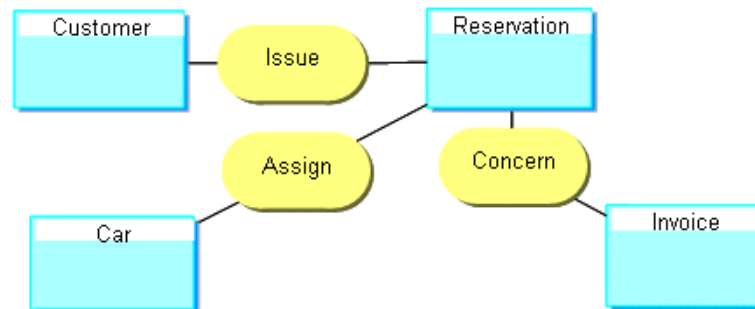
connects three, etc. It can carry properties, ie. attributes that characterize association of the entities.

Examples of associations

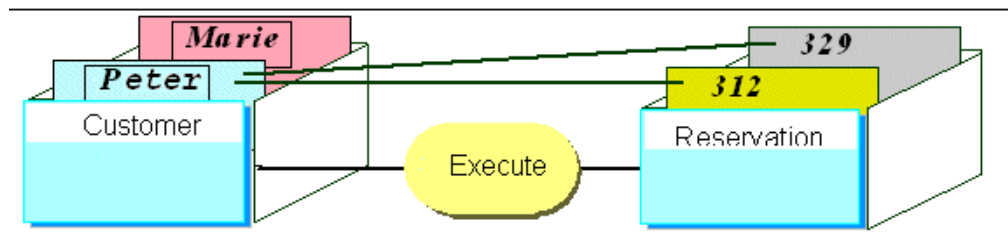
To model that an "employee" is responsible for a "service" and to specify the "start date" of his or her functions, the following data model is created, where start date is a property of the association.



Other comparison: links between sheets.



The following drawing provides a three-dimensional view of the situations a data model can store.



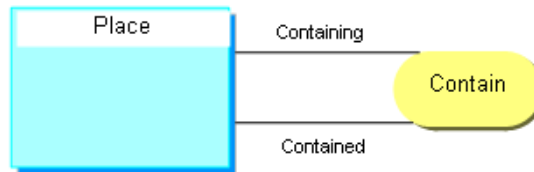
Peter and Mary are clients. Peter has made reservations numbers 312 and 329.

A data model should be able to store all situations in the context of the company, but only these situations.

☛ *The model should not allow representation of unrealistic or aberrant situations.*

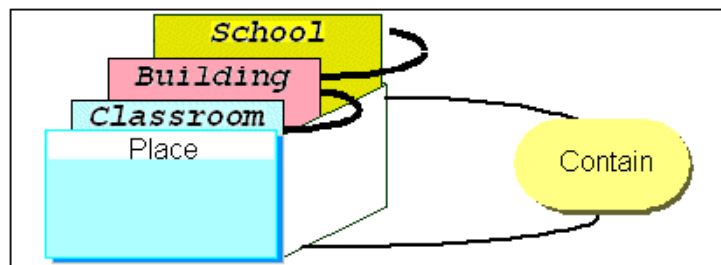
Reflexive relationships

Certain *associations* use the same entity.



Example

A classroom, a building, and a school are all locations.



A classroom is contained in a building, which is contained in a school.

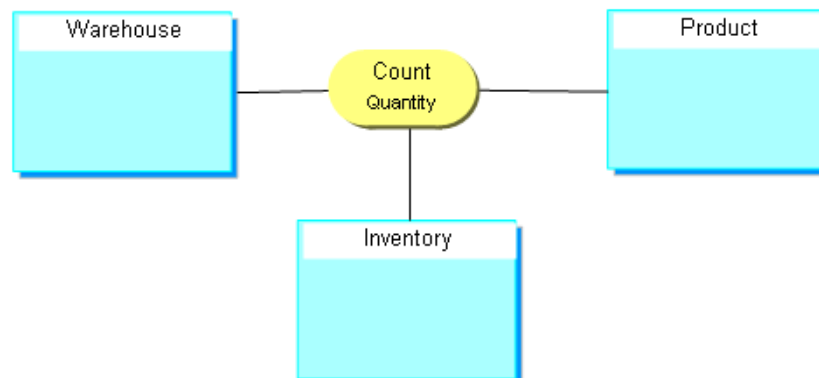
"n-ary" relationships

Certain associations associate more than two entities.

These associations are generally rare.

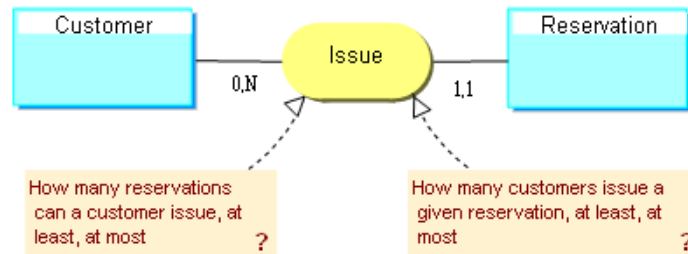
Example

When taking inventory, a certain quantity of product was counted in each warehouse.



Participations or cardinalities

Minimum and maximum cardinalities express the minimum and maximum number of participations of an instance of the entity in an association.



The most common participations or cardinalities are 0,1 1,1 0,N 1,N.

- Optional cardinality: minimum cardinality 0 indicates that the association is not necessarily specified.
- Mandatory participation Minimum cardinality 1 indicates that the association is necessarily specified.
- Unique participation : Maximum cardinality 1 indicates that the entity can be linked by the association once only at most.
- Not unique participation : Maximum cardinality N indicates that the entity can be linked by the association several times.

Example

The following example illustrates the significance of the different cardinalities or participations:

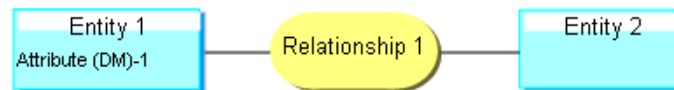


- 0,1** An order corresponds to zero or at most one invoice.
- 0,N** No restriction is placed on the number of invoices corresponding to an order. This is the default visibility.
- 1,1** Each order has one and only one corresponding invoice.
- 1,N** Each order has one or more corresponding invoices.

Creating an Association (Relationship)

To create an association:

1. Click the **Association** button  in the objects toolbar.
2. Click one of the entities concerned and drag the mouse to the other entity before releasing the button.
The **Add Association** dialog box appears.
The arrow at the right of the **Name** box opens a menu that allows you to:
 - **Query** of existing associations, via the **Query** dialog box.
 - **List** associations in the repository.
 - **Create** an association.
3. Enter the name of the association, then click **Create** (Windows Front-End) or **Add** (Web Front-End)
The association appears in the diagram.



☛ In case of error, you can delete an object by right-clicking it and selecting the **Delete** command in its pop-up menu.

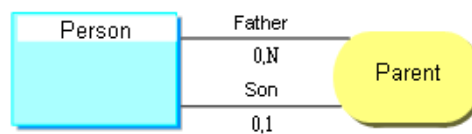
Reflexive relationships

If the creation request is made on an entity without moving the cursor, a reflexive association (also called "reflexive link") is automatically created on the entity.

If there is association of an entity with itself, the roles need to be named in order to distinguish between the corresponding links in the drawing.

Example:

"Father" and "Son" are the two roles played by the "Person" entity in the "Parent" association.



Specifying participations

In the **Characteristics** tab of the property window of roles, you can indicate the minimum and maximum number of participations of each entity to the relationship (cardinalities).

The screenshot shows a dialog box titled "Properties of Invoiced order" with several tabs: General, Characteristics (selected), SQL, Objectives and Requirements, Complements, and Texts. The "Local name" field contains "Invoiced order". The "Owner" field is set to "Association (DM)" with a dropdown arrow, and the "Invoice" field has a right-pointing arrow. Below these are several fields with dropdown arrows: "Identifying Role", "Whole/Part", "Multiplicity", "Min Participation" (set to "Mandatory"), and "Max Participation" (set to "Unique").

Attributes (Information) - Merise

Properties

Entities and associations can be characterized by attributes:

These attributes can be found by studying the content of messages circulating within the company.

☛ *An attribute is the most basic data saved in the enterprise information system. An attribute is a property when it describes an entity or association, and an identifier when selected as a means of identification of each instance of an entity.*

A property characterizes an association when the property depends on all the classes participating in the association.

In the diagram below, the "Role" that a "Consultant" plays in a "Contract" depends on the consultant and on the contract, and therefore on the "Intervene" association.

Examples of attributes:

"Client Name" (property of the client entity).

"Client No." (identifier of the client entity).

"Account Balance" (property of the account entity).

Identifier



Customer number 2718 executes Reservation number 314159.

Each entity has a unique *identifier* whose value can be used to find each of its instances.

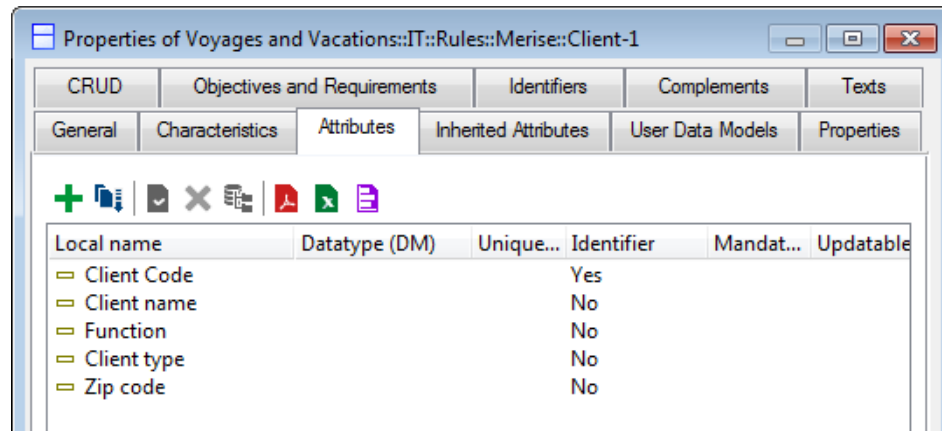
☛ *An identifier consists of one or several mandatory attributes or roles that enable unique identification of an entity.*

By default, associations do not have their own identifiers: an association is identified by the identifiers of the linked entities.

Creating Attributes

Attributes are created in the properties dialog boxes of associations and entities.

The **Attributes** tab of this dialog box shows attributes already linked to the entity or association.



To create an attribute:

- 1 Click the New button and enter the name of the attribute.

You can specify its characteristics (see [Attribute Description](#) for further details).

☛ *You can specify its **Length**, if necessary complemented by the number of **Decimals**; it should be noted that the number of decimals is not added to the length; an information of length 5 with two decimals being presented in the form " 999.99 ".*

When you have completed this, close the properties dialog box.

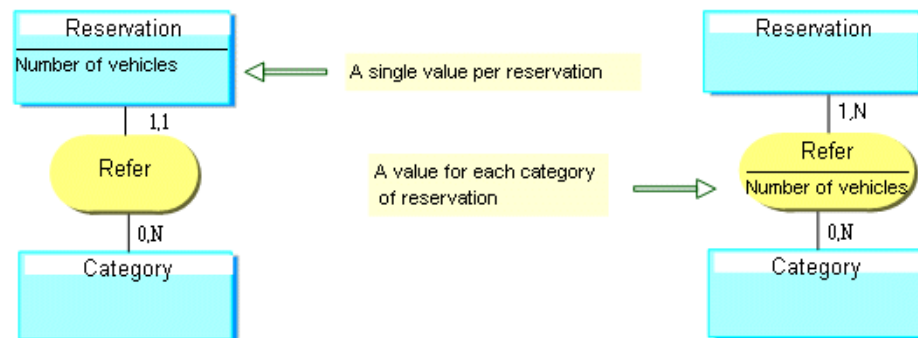
Normalization Rules (Merise)

Normal forms are rules that are designed to avoid modeling errors.

Currently, there are six or seven normal forms. We will discuss the first three.

First Normal Form

The value of an entity (or association) Property is fixed uniquely as soon as the entity concerned is known (concerned entities).

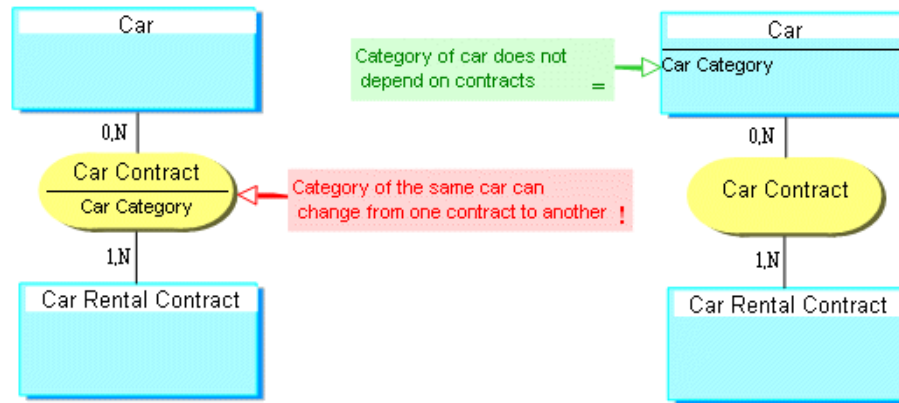


If the number of vehicles is an attribute of the "Reservation" entity, you can only indicate the total number of vehicles for a reservation. You must therefore make one reservation per category of rental vehicle (cardinalities 1,1).

If the number of vehicles is an attribute of the association, you can specify the number of vehicles reserved for each category in the association. You can therefore make a single reservation for several categories of vehicle (cardinalities 1,N).

Second Normal Form

The value of an association Property is set only when all the entities concerned are known.

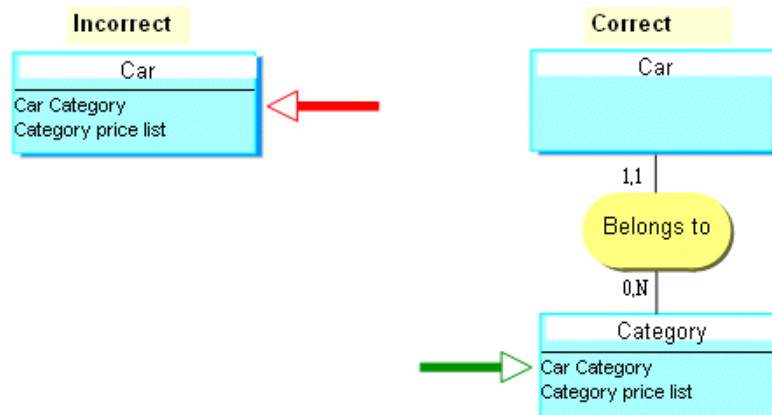


The fact that the car category is an attribute of the "Car Contract" association assumes that the car category may change from one contract to the next, which would not be very honest.

If the car category is to be independent of the contract, it must be an attribute of the "Car" entity.

Third Normal Form

A Property must directly and uniquely depend on the entity it describes.



If the "Category Price List" is an attribute of the "Car" entity, this indicates that two cars in the same category can have a different "Category Price List". To avoid this, we need to create a "Category" entity that contains the price list.

☛ This rule is used to reveal concepts that were not found during the first draft of the data model.

Refining Data Model Specification (Merise)

During specification, it is often necessary to complement the data model.

Complements to the specification consist of:

- Specifying Length and Decimal characteristics and documenting attributes.

In the data model, it is also possible to specify:

- Sub-type entities.
- Constraints that must be respected by data in documentary terms. These constraints are imposed by checks carried out during data update processing.

Ordering Attributes

The initial order of attributes is their order of creation (or of creation of the link with the entity or association).

To modify this order:

1. In the **Attributes** of the properties dialog box of the object, click the  **Reorder**.

The **Order Modification** dialog box appears.

To reorder attributes:

1. Select the attribute to be moved by clicking its name with the left mouse button.
2. Move the cursor to the desired position; it takes the following shape: 
The attribute is placed in the desired position, and the order of the link with the entity is modified.

This order will be used to generate the order of columns and tables. It will also be used in the document associated with the data model.

Attribute Description

Attributes can be described in two ways:

- By entering this description in the various fields of the list presented in the **Attributes** tab.
- In the properties dialog box of each attribute. This dialog box is opened by selecting **Properties** in the attribute pop-up menu.

You can enter the attribute characteristics values in the corresponding fields.

- The **Data Type** which is the class used to specify the attribute type.
- The **Identifier** field indicates if the attribute forms part of the entity identifier.
- The **Mandatory** field enables indication of whether or not entry of a value for this attribute is mandatory.
- The **Uniqueness** field enables indication of whether or not two instances of this entity can have the same value for this attribute.
- The **Updatable** field enables indication of whether or not the value of this attribute can be modified after it has been entered.

Participations or cardinalities

To modify the participations or *cardinalities* of an association:

1. Open the properties dialog box of the association.
2. Click the **Characteristics** tab.
3. Enter participation (cardinality) values.

Local name	Entity (DM)	Whole/...	Multiplicity	Min Participation	Max Participation	Identifying Role
Article	Article			Optional	Not Unique	
Client	Client-1			Optional	Not Unique	

☛ A cardinality is the minimum (or maximum) number of times an entity "participates" in an association (see also multiplicity).

Cardinalities or participations most commonly used are:

- 0 or 1 for minimum cardinality (optional or mandatory minimum participation).
- 1 or N for maximum cardinality (unique or not unique maximum participation).

Different values are permitted.

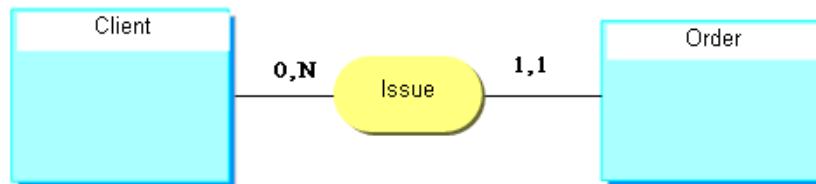
When several roles, ie. several links, exist between an entity and an association, the cardinalities are defined for each role.

Cardinality of an entity in an association can also be defined as follows:

- For a binary association, it is the minimum (or maximum) number of instances of the other entity in the association that can be linked to the initial entity.
- For a ternary association, it is the number of pairs of other entities in the association that can be linked to the initial entity.
- For a quaternary association, it is the number of triplets, etc.

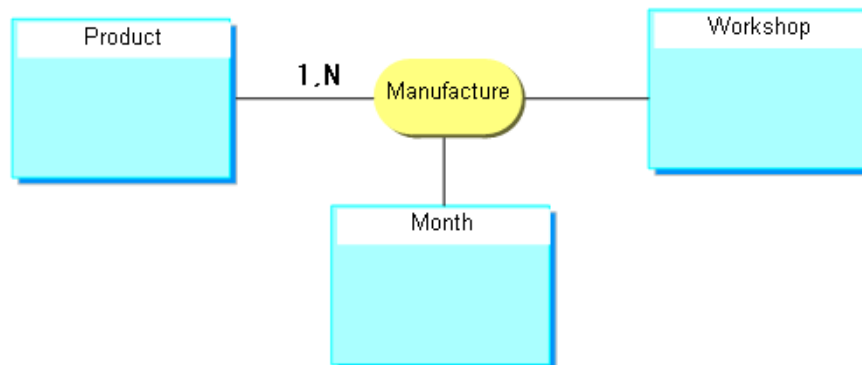
☛ If expression of cardinalities is not sufficient to describe the link that exists between an entity and an association, for example when a cardinality depends on an organizational context, it is possible to use cardinality constraints, which enable more precise description.

Examples



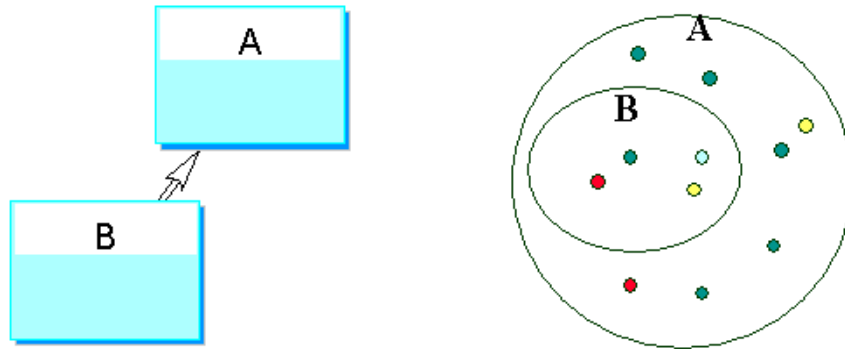
0,N : The client can issue no order, can issue a maximum of N orders (N indeterminate).

1.1: The order must be issued by one and only one client.



1,N : A product must be manufactured at minimum in 1 workshop over a period of 1 month. It can be manufactured in several workshops and/or over a period of several months (several workshop-month pairs).

Sub-typing (Merise)



What is sub-type?

An entity B is a sub-type of entity A. This assumes that all instances of entity B are also instances of entity A. In other words, B is a subset of A.

Example A: Person, B: Bostonian.

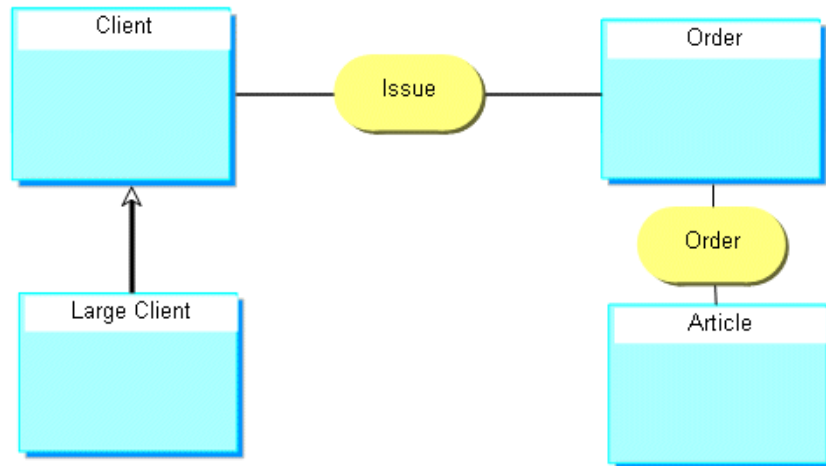
B being a subset of A, the instances of entity B "inherit" the characteristics of those in entity A.

It is therefore unnecessary to redescribe for entity B:

- its properties
- Its associations

Example

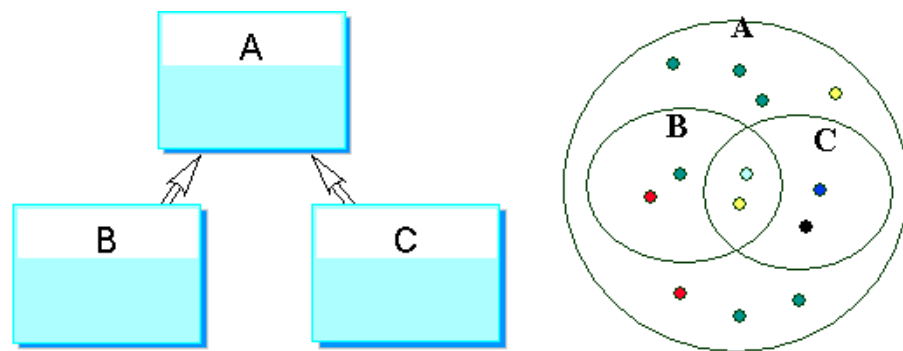
The "Large Client" entity, representing clients with a 12-month revenue exceeding \$1 million, can be a subtype of the Client entity (origin).



A sub-type inherits all properties, associations, roles and constraints of its super-type, but it can also have properties, associations, roles or constraints that its super-type does not have.

In the above example, the properties, associations, roles and constraints specified for "Client" are also valid for "Large Client".

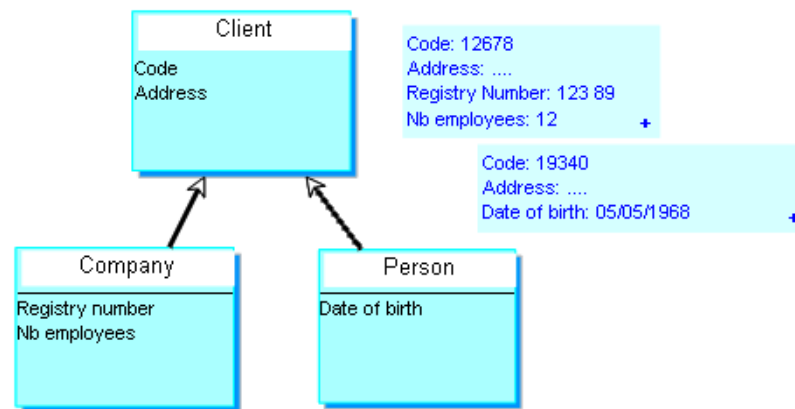
Multiple Subtypes



Several sub-types of the same entity

- are not necessarily exclusive.
- do not necessarily partition the type.

Advantages of sub-types



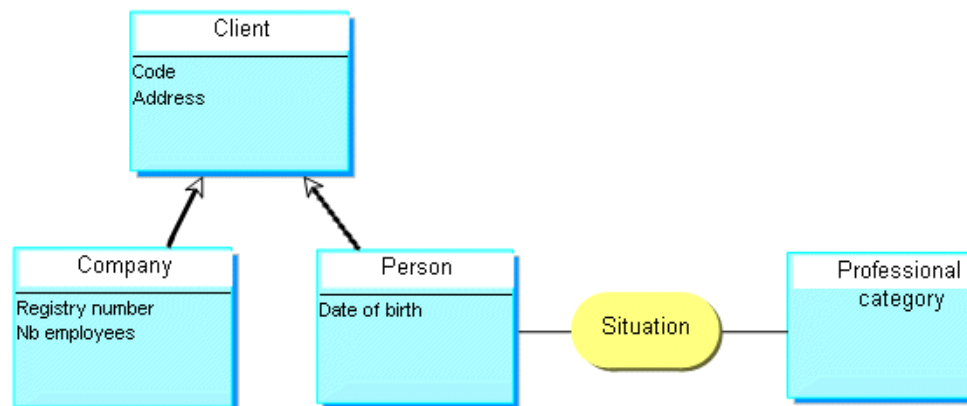
A sub-type entity can have specific properties. These only have meaning for that particular sub-type. In the above example:

- "Registry number" and "number of employees" only have meaning for a "company".
- "Date of birth" is a characteristic of a "person", not a "company".

 An entity *B* is a sub-type of an entity *A*, if *B* represents a subset of *A* and the instances of entity *B* inherit the descriptions of those of entity *A* and if they have specific descriptive elements.

The Sub-Type link is represented graphically by a double arrow.

A subtype entity can also have specific associations.



A "person" falls into a "socio-professional group": "manager", "employee", "shopkeeper", "grower", etc. This classification makes no sense for a "company". There is also a classification for companies, but this differs from the one for persons.



MODELING DATABASES

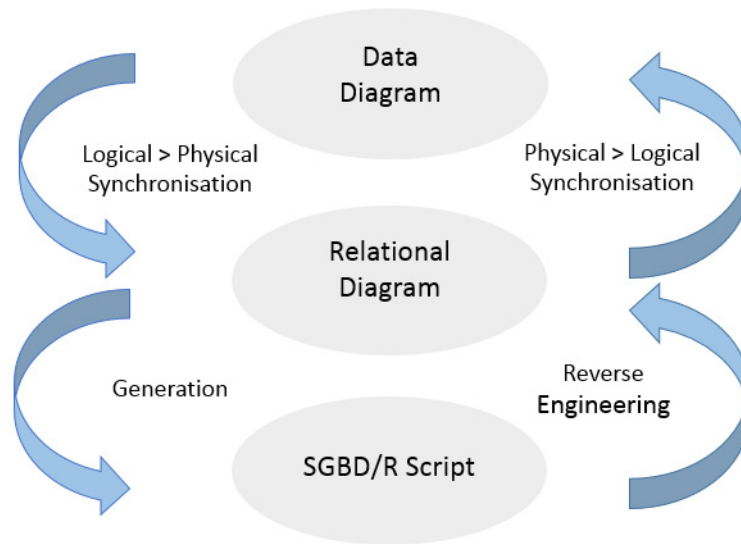


A database is the physical object that enables storage and organization of logical data for use by programs corresponding to distinct applications, to facilitate the independent evolution of the data and the application programs.

HOPEX Data Governance and **HOPEX Information Architecture** integrate the logical and physical modeling levels and allow to switch from one model to another. You can therefore:

- Build a data diagram or a class diagram, See [Defining Logical Data](#).
- From this diagram, create database tables and their columns, indexes, and keys, as well as the drawings for the corresponding relational diagrams. See [Synchronizing logical and physical models](#).
- Optimize the resulting relational model and generate SQL commands to define the tables. **HOPEX Data Governance** and **HOPEX Information Architecture** in particular take account of changes in the conceptual model without losing optimizations made to the relational model. See [Denormalizing logical and physical models](#).
- Reverse generate a database definition using the ODBC protocol to create the corresponding tables and columns in **HOPEX Information**

Architecture, and obtain the corresponding data diagram or class diagram. See [Reverse engineer tables](#).



Logical Formalism and Synchronization

The logical formalism applied by default in the synchronization is the UML notation. It integrates handling of parts, not associations. When you synchronize a data model into a physical model, associations of the model are not taken into account in the synchronization.

It is possible to take into account the UML notation and the data models with the treatment of associations and not of parts. You can change the formalism in the **HOPEX Administration** application.

The option set in **HOPEX Administration** applies by default to all databases in the repository, but you can change the formalism only on one database in its synchronization options.

To access the option in **HOPEX Administration** :

1. Open the Administration tool.
2. Open the options of the environment concerned.
3. In the Options window, in the tree on the left, unfold the **Data Modeling** folder.
4. Click **Database synchronization**.
5. In the right pane of the window, in **Default correspondence type**, select the desired value.
 - UML - Physical: default option (with the processing of parts)
 - Datamodel - Physical: old option (with the processing of associations)

See also: [Logical Data Modeling Options](#).

DATABASE

On a database, and depending on the target DBMS, control parameters of the various data processing tools (synchronization, generation, reverse generation etc.) will be defined.

Creating Databases

A database enables specification of data physical storage structure.

To create a *database* in **HOPEX Data Governance**:

1. Click the navigation menu then **Architecture > Hierarchy View**.

*☛ In HOPEX Information Architecture, click the navigation menu then **Data Architecture > Physical Data Assets**.*

2. Right-click the databases folder and click **New > Database**.
3. Enter the name of the database.
4. Click **OK**.

The database created appears in the list of databases.

Database Properties

To access the properties of a database:

1. Select the database and click the **Properties**  button.
The properties window of the database appears.
2. Click on the drop-down list to access the different properties pages.

Properties pages are used to:

- Access the **Components** of the database (tables, physical views, data groups, etc.).
- Modify the **Characteristics** of the database (name, target DBMS, etc.).
- Define **Responsibilities**.
- Define the **Risks** associated, the **Standards** used, the **Objectives and Requirements**.
- To define the **Options** linked to:
 - the generation of tables. See [Configuring Database Generation](#).
 - synchronization. See [Configuring Synchronization](#).

Associating a Package with a Database

You can create a data package from the database or connect an existing package to it. The package enables representation of the structure of the database, the classes it contains and their parts.

The database package is the default owner of the objects represented in the class diagram. However it is possible to use objects held in other packages.

 *In the same way, you can connect a data model to a database, if the corresponding formalism has been selected. See [Formalisms](#).*

To create a package from a database:

- 1. Click the database icon and select **New > Package**. This command creates a new package as well as its associated class diagram.

To connect an package to a database:

1. Click the database icon and select **Connect > Data Package**. The query dialog box appears.
2. Click **Find**.
3. Select the desired package and click **Connect**.

You can see the name of the packages associated with a database in the database properties, in the **Characteristics** page.

Importing a DBMS Version

At creation of a new repository, only the **SQL ANSI/ISO 9075:1992** DBMS version is installed. If you choose another target DBMS for a database, you must import the module for SQL datatypes that is compressed in the **DBMS SQL Type.exe** file.

The **DBMS SQL Type.exe** file is available in the Utilities of your HOPEX installation directory.

Once the module is imported in **HOPEX Information Architecture**, you can select the appropriate DBMS in the database properties windows.

See also [Configuring Synchronization](#).

PHYSICAL DATA MAPS AND RELATIONAL SCHEMAS

A database can be split into a set of relational schemas.

A relational schema is used to define a physical data structure.

A physical data map is used to visualize the dependencies between relational schemas.

Physical Data Maps

A physical data map is an urbanization tool for physical information. It represents a set of relational schemas in a particular context.

Creating a physical data map

To create a physical data map in **HOPEX Data Governance** :

1. Click the navigation menu then **Architecture > Hierarchy View**.

*In HOPEX Information Architecture, click the navigation menu then **Data Architecture > Physical Data Assets**.*

2. In the edit area, right-click the database and click **New > Physical Data Map**.

The map created appears.

To create the diagram of the physical data map:

1. Right-click the map and select **New > Diagram**.
2. Select **Physical Data Map Diagram**.
3. Click **OK**.

The diagram appears in the edit area.

Components of a physical data map

You can add internal and external components to a physical data map.

The internal components are relational schemas that are part of the map scope (whether they belong to the same owner element or not).

The external components are those used in the map but that are not part of the scope analyzed.

To add a relational schema to the map:

1. In the diagram insert toolbar, click the **Relational Schema** button then click in the map.
2. Indicate the name of the relational schema and click **OK**.

Relational Schema

A relational schema represents a restricted relational data structure.

Creating a Relational Schema

To create a relational schema:

1. Click the navigation menu then **Architecture** > **Hierarchy View**.
2. In the edit area, unfold the **Database** folder.
3. Right-click the name of the database concerned and click **New** > **Relational Schema**.

The new relational schema is created. You can open its properties to modify or complete its characteristics.

Relational Schema Diagram

A relational schema is made up of tables and/or physical views and can be described in two types of diagram:

- the table diagram which is used to display a set of tables and their relationships (FK).
- the structure diagram that is used to break down a relation a schema into sub-domains.

You can connect one or more diagrams to a relational schema according to what you want to describe.

To create a diagram from a relational schema:

1. Right-click on the relational schema and select **New** followed by the type of diagram (tables or structure).

Adding a component to a relational schema

You can connect objects to a relational schema through components.

A relational schema can include:

- Sub-domains, visible in the structure diagram
- Tables or physical views, to which the type of access (read only, modification, deletion, etc.) is defined and which are visible in the relational schema's table diagram.

To add a component to the relational schema:

1. Open the properties of the relational schema in question.
2. Click the **Components** page.
The first section allows you to add domains.
The second section allows you to add objects of type table or physical view.
3. Click **New** to add a component.

Defining the access mode to the referenced object

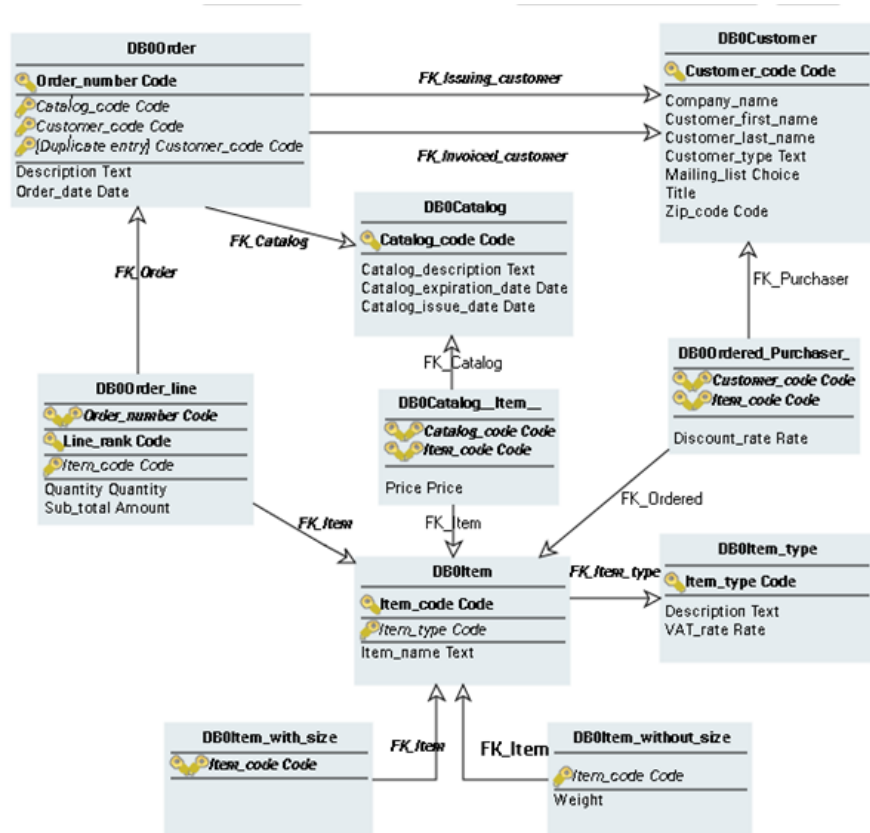
On components of type table or physical view you can define the access mode to the referenced object (creation, read-only, deletion, etc.).

To define the access mode to the object in the relational schema:

1. Open the properties window of the relational schema.
2. Select the **Components** page.
3. Select the line of the component in question.
Commands are added, including the **CRUD** button.
4. Click this button.
5. In the window that appears, select or deselect the check boxes associated with the actions: Create, Read, Update, Delete.

RELATIONAL DIAGRAM

The Relational Diagram (RD) describes a database: it represents the physical data structures used by application programs.



Description in **HOPEX Information Architecture** of relational diagrams makes it possible to interface with the selected DBMS, guaranteeing semantic consistency between design data and production data.

Creating the Relational Diagram

The relational diagram is generally built in two phases:

1. Automated synchronization of the data diagram or diagrams produces the "raw" diagram.
See [Synchronizing logical and physical models](#).

2. Optimizing the diagram, or denormalization, to take into account the data access requirements of the application and to fine-tune the database performance.
See [Denormalizing logical and physical models](#).

The key concept in a relational diagram is the table, which is derived from an entity or association.

 *A table is a logical structure of data, used as the reference for the switch to production, the table is the central element of the database. A table is accessible by means of a primary key, and if necessary foreign keys; it is described by an ordered sequence of columns. A table is generally derived from an entity or association.*

A **table** is accessible by one or several keys, whose type indicates whether they are primary or foreign keys. It is possible to define indexes for a table, specifying their sort order (ascending or descending) and whether they are unique. Keys and indexes are connected to the columns that they contain.

Creating objects in the diagram

To create a key or an index in the relational diagram:

1. Click the table concerned.
The list of commands associated with the table appears.
2. Click **Key**  or **Index** 

 *Check that the columns used by the key or index already exist in the table.*

You can also use the **Components** page in the properties dialog box of the database to create these objects. See [Creating a Key](#) and [Creating an Index](#).

To create a foreign key:

1. Select the **Key**  button, click the first table, and then hold the button down while dragging the mouse to the second table.
The creation dialog box opens.
2. Specify the name of the key and click **Add**.
A second window asks you if you want to automatically create the columns of the foreign key from those of the primary key.
3. Click **Yes** to validate or **No** to create.

Configuring display of relational diagrams

As for data diagrams, you can specify which elements are to appear in the diagram:

- Either by using the **Views and Details** button to indicate globally the types of objects to be displayed in the diagram.
- Or by using display options that enable definition of which object characteristics should be presented.

To configure the display for a selected object:

- 1 Right-click the object and select **Shapes and Details**.

 When configuring the display of an object, the **Display** dialog box first shows the shapes that can be used to represent the object. Selecting an element in the tree causes its content to appear.

DATABASE COMPONENTS

A database is a set of data organized for use by distinct applications, to facilitate the independent evolution of the data and the application programs.

A database consists of tables, columns, keys and indexes:

- A **table** is the logical entity where columns are stored.
- A **column** is contained in a table.
- Just as an identifier uniquely identifies a class, the **primary key** for the table uniquely identifies a row in the table.
- A **foreign key** accesses another table, and imposes consistency between the corresponding columns in the tables concerned.
- An **index** accelerates access to data. It can be unique or not, and may be ascending or descending.

Database Tables

Database tables can be viewed or updated in two ways:

- In its relational diagram, that is the diagram of the tables in the database.
- In its properties, in the **Components** page.
The components page displays the database tables.
The name of the associated **Data Group** is indicated if applicable.

Creating a table

See previously: [Database Properties](#).

To create a table from the properties of the database:

1. Open the database properties dialog box.
2. Click the **Characteristics** page.
3. In the **Physical Data** section, click **Tables**.
4. Click **New**.
5. Enter the table name and click **OK**.

Deleting a table

To delete a table and its columns, keys and indexes:

1. Right-click the table and select **Remove** .
A message asks if you want to remove the table from the database or delete it from the repository.
2. Select **Delete** and click **Remove**.

 The deleted table will not be automatically recreated at a new synchronization. To recreate a table, at the synchronization results validation step validate the creation action proposed for this table (select the corresponding check box). See [Step 4: Validating results](#).

Table Columns

Viewing columns

See previously: [Database Properties](#).

To view the columns of a table:

1. Open the database properties dialog box.
2. Click the **Columns** page.

Presented for each column are:

- Its **Local Name**
- Its **Datatype**

 The administrator can add to the list of datatypes (see [Creating New Datatypes](#)).

- Its length **Ln** and its number of decimals **Dcml** where appropriate.
- The value of its **NotNull** attribute.
- Its **default value**: on generation of the table, the default value taken is that of the attribute from which it originated. If no initial value is specified for the attribute, or if you want to modify the value of a column, enter a value in this field.
- The fact that the column is connected or not connected to a primary key (PK) or foreign key (FK). This is indicated by **Y** ("Yes") or **N** ("No").

You can modify the **Local Name** for a column by clicking its name and then entering the new name. This local name will be used in the script generated for the table.

An **SQL Name** can be specified directly in the **SQL** page of the properties of an attribute in the data diagram. Then all columns created from this attribute will have the same local name. In addition, the name will be reused during successive synchronizations, including total or partial reinitializations.

It is also possible to modify the value of other column characteristics.

 These modifications will be retained in subsequent synchronizations.

It is possible to create columns not derived from attributes in the data diagram whether in a table generated or created by the user.

Creating a column

See previously: [Database Properties](#).

To create a column:

1. Open the properties of the table concerned.
2. In the **Columns** section, click **New**.

 When creation of a column is not carried out from the **Properties** of a table, but for example from the explorer, it is necessary to previously select the table that will contain it, otherwise a message will indicate that creation is impossible.

Deleting a column

To delete a column:

1. Right-click the column and select **Remove**.
A message requests confirmation of the final deletion.
2. Click **Delete**.

 The deleted column will not be automatically recreated at a new synchronization. To recreate a column, at the synchronization results validation step validate the creation action proposed for this column (select the corresponding check box). See [Step 4: Validating results](#).

The **Reorder** button  accesses the Modify Order dialog box.

Modifying Keys and Indexes

The automatic creation of primary and foreign keys, and of indexes on these keys, is indicated in the synchronization configuration.

When these creations are requested:

- The primary keys use the columns corresponding to the identifiers.
- The foreign keys use the columns that are included in the tables because of a constraint association.

An index is created for each key.

It is possible to add to, modify, or delete the keys and indexes proposed during generation. To do this:

1. Right-click the table concerned and select **Properties**.
2. Click the **Characteristics** page.
The page presents the **Keys** and **Index** of the table.

The following is specified in the **Keys** section:

- The type of key (**Key-Type**): Foreign or Primary.
- In the case of a foreign key:
 - The table referenced.
 - Management of repository integrity on update (**On Update**) and on deletion (**On Delete**); consult the target DBMS documentation for the order types managed.

 When a "migratory" column is created in a table to reflect a constraint association, you can instruct the DBMS to verify the updated value in this column. The DBMS then verifies that this value still exists in the original table (database integrity).

When an update (**On Update**) or delete (**On Delete**) command is applied to the original table, the DBMS may:

- Update the values in the tables concerned, with the **Cascade** option.
- Do nothing, with the **NoAction** option.
- Prohibit updates or deletes, with the **Restrict** option.
- Reset to the default value in the tables concerned, with the **Set Default** option.
- Set the value to **Null** in the tables concerned, with the **Set Null** option.

The following is specified in the **Index** section:

- Its **Type**: Bitmap, Standard, Unique, Unique where not null.
- Its **Sort Order** (**Ascending** or **Descending**).
- If a grouped index **Clustered**.

✎ Creation of a column from a key or index is not possible. A column must first be created in the table, then connected to the key or index.

Creating a Key

See previously: [Database Tables](#).

To create a key:

1. Right-click the table concerned and select **Properties**.
2. Click the **Characteristics** page.
3. In the **Keys** section, click **New**.
The key creation dialog box appears.
4. Select the type of key to be created: "foreign" or "primary". Creation of the key varies according to the type selected.

Primary key

When you select "Primary" type, the key appears in the properties of the table.

To define the properties of the key:

1. Right-click the key and select **Properties**.
In the **Columns** page, you can specify the columns concerned by the key.

It is also possible to specify the primary key of a table in the **Identifiers** page of the properties dialog box of the entity to which the table belongs. See [Entity Identifier](#).

It is also possible to manually specify the primary key by relating it to elements that can be attributes of the entity or the primary key of another table connected by a constraint association (multiplicity 1).

In all cases, the key specified will be created in the table on synchronization.

Foreign key

When the key created is a foreign key, a list of database tables is presented.

1. Select the reference table to which the foreign key relates.
If the table you select includes a primary key, a dialog box opens.



2. Select **Yes**.
the key appears in the properties of the table.

You can modify the **Local Name** of the key (the full name of the key is composed of the name of the database to which it belongs, followed by the name of the table then the local name: in the above example, "Exchange DB::Concern::Key1").

For a foreign key, as when editing a key, it is possible to specify repository integrity management on update (**On Update**) and on deletion (**On Delete**).

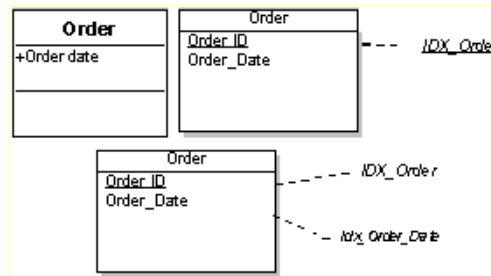
Creating an Index

See previously: [Database Tables](#).

Indexes are created automatically for primary and foreign keys. It is possible to add to the generated indexes columns used frequently in search criteria.

☺ *It is also possible to specify an index in the **Identifiers** page of the properties dialog box of the entity to which the table belongs. An index specified in this way will be created in the table during synchronization.*

Examples of indexes:



To create an index:

1. Right-click the table concerned and select **Properties**.
The properties window appears.
2. Click the drop-down list then **Components**.
3. In the **Index** section, click **New**.
The **Create Index** page appears.
4. Specify the **Local Name** and click **OK**.

Depending on the possibilities offered by the DBMS, you can specify the index **Type**, index **Sort direction** ("Ascending" or "Descending"), and if it is a grouped index (**Clustered**).

It is then possible to select the columns of the key (or index) in the **Columns** page of the properties dialog box of the index.

Adding a Column to a Key or Index

See previously: [Database Tables](#).

To add a column to a key (or to an index):

1. Right-click the table concerned and select **Properties**.
The table properties dialog box opens.
2. Click the drop-down list then **Components**.
3. In the **Index** section, right-click on the index and select **Properties**.
The properties dialog box of the index appears.
4. Click the drop-down list then **Columns**.
5. Click the **Connect** button.
The query dialog box appears.
6. Find and select the column to be added to the key (index).

It is possible to indicate the sort order of the key or index, which can be "Ascending" or "Descending".

PRIMARY AND FOREIGN KEYS

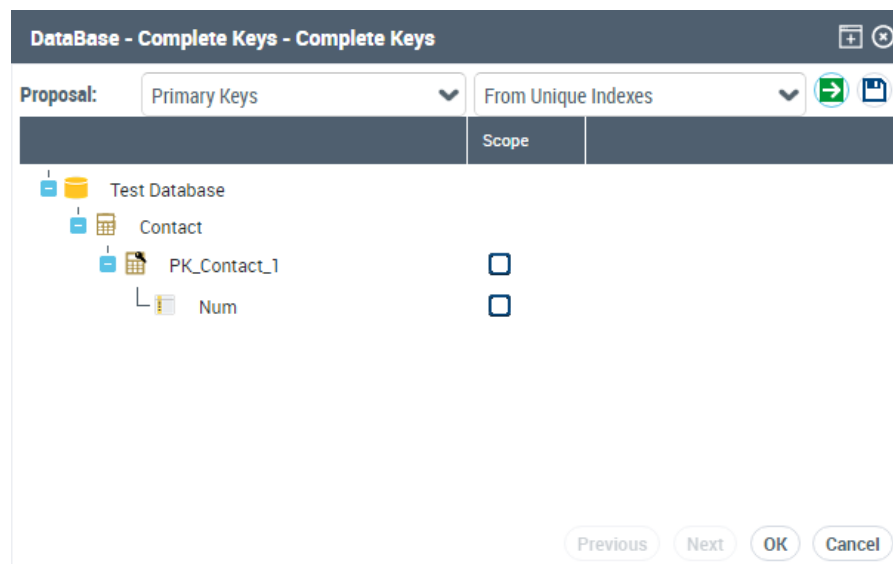
When the keys of a database are not completely specified, you must **Complete** them.

Specifying Primary Keys

To specify the primary keys of the database:

- 1 Right-click the database and select **Complete keys**.
The **Complete keys** window appears.

When database specification is completed, the dialog box presents an empty list: no additional specification is required.



The **Proposition** list is used to complete the keys:

- **From unique indexes**: the columns that belong to a unique index are proposed as components of the primary key.
- **From mandatory columns**: these columns are proposed as components of a key.
- **Through name comparison**: if the same column name is found in several tables, the column is proposed as primary key.

Each key is proposed under the table to which it belongs.

To validate a primary key:

1. Select the check box of the **Scope** column corresponding to the key.
The associated columns are automatically selected by default. You can eliminate those that correspond to search criteria but are not components of the key.
2. Click the **Apply** button.
The **Apply** button removes from the list the propositions of keys explicitly accepted or rejected.

For foreign keys, two keys including the same column on the same table are incompatible: acceptance of one automatically results in rejection of the other.

It is not possible to select several primary keys on the same table: acceptance of one results in rejection of the others.

☺ *You can complete the specification of keys in several stages. This allows you to consult the contents of the database while making your selections. To do this:*

- Click the **Apply** button to save your modifications.
- Click the **Cancel** button to exit this dialog box without starting processing.

Specifying Foreign Keys

To specify the foreign keys of the database:

1. Right-click the database and select **Complete keys**.
The **Complete keys** window appears.

☞ *When database specification is completed, the dialog box presents an empty list: no additional specification is required.*

The **Proposition** list is used to complete the foreign keys:

- **From indexes**
- **From name comparison**

If the proposition is made from indexes, it is based on non-unique indexes of the table. The reference table is indicated after the name of the key.

To validate a foreign key:

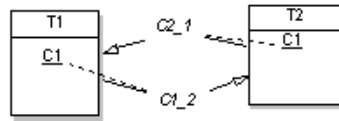
1. Select the check box of the **Scope** column corresponding to the key.
2. Click the **Apply** button.
The **Apply** button removes from the list the propositions of keys explicitly accepted or rejected.

If no reference table is specified, the wizard automatically proposes the selection of possible tables. Keys that do not have a reference table cannot be accepted.

On proposition of keys, several tables may be found with an identical primary key. This could for example be the case for tables corresponding to different sub-types of the same entity.

Column Primary Key of Two Tables

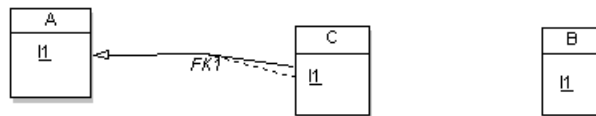
When the same column is the primary key of two tables, the foreign key proposition permits creation of each of these two keys.



A choice is then made of which of the two keys should effectively be taken into account.

Column Primary Key of Three Tables

When the same column is the primary key of three tables, the foreign key proposition permits creation of one foreign key from a column of a table.



The key proposition permits creation of only one foreign key from table C. The other should be added in data entry of tables in the database.

DATA TYPES AND COLUMN DATATYPES

Not all data has the same value type. Datatype determination enables indication of format and therefore facilitates handling by the different data processing tools.

HOPEX manages datatypes at different modeling levels, assuring correspondence of datatypes at the logical level with datatypes handled by the different supported DBMSs.

Attribute Datatypes

A type is used to group characteristics shared by several attributes.

To type attributes of an entity, only those datatypes defined for the *data model* that contains this entity are proposed.



A data model is used to represent the static structure of a system, particularly the types of objects manipulated in the system, their internal structure, and the relationships between them. A data model is a set of entities with their attributes, the associations existing between these entities, the constraints bearing on these entities and associations, etc.

For more details on data types and reference data type packages, see [Datatypes](#).

Determining Column Datatypes from Attribute Types

Datatypes defined at the logical level level are not always comprehensible for the target DBMS. In this case, they need to be converted to datatypes corresponding to the target DBMS.

This conversion intervenes notably at synchronization. Datatypes of attributes defined in the logical model are translated to datatypes for the generated columns.

Conversion is assured by an equivalence link with pivot types. The pivot types are an intermediary between logical datatypes and generated datatypes.

Pivot Types

Pivot types are datatypes defined independently of the target DBMS, which you can use when you do not yet know the system in which the database will be hosted, or when several systems may be used.

Pivot types have an equivalent datatype in each supported DBMS. They therefore enable you to define the attribute types just once, then to reinterpret them later as a function of the target DBMS.

To use the datatypes of a DBMS, you must import the corresponding module. See [Importing a DBMS Version](#).

List of pivot types

Once imported, the pivot types are available in the **Logical data** navigation pane, in the "Pivot" datatype package.

Alphanumeric types

P-String	Alphanumeric character chain
P-Text	Alphanumeric character chain
P-Character	Alphanumeric string of fixed length
P-Varchar	Alphanumeric string of variable length

Other Information

Length

Numeric types

P-Decimal	Decimal
P-Double	
P-Float	
P-Integer	Short
P-Long Integer	
P-Long Real	
P-Real	
P-Smallint	
P-Tinyint	

P-Numeric	Number	Length, decimal places
P-Currency	Amount expressed as currency	Length, decimal places

Date types

P-Date	Date
P-Time	Time
P-DateTime	Date and time

Binary types

P-Binary	Binary string
P-Byte	Binary string
P-Timestamp	Identification automatically generated from the date and time, expressed in thousandths of seconds since 01 January 1970

P-Boolean	Boolean, equals 0 or 1
P-Multimedia	Binary string
P-Varbinary	Binary string

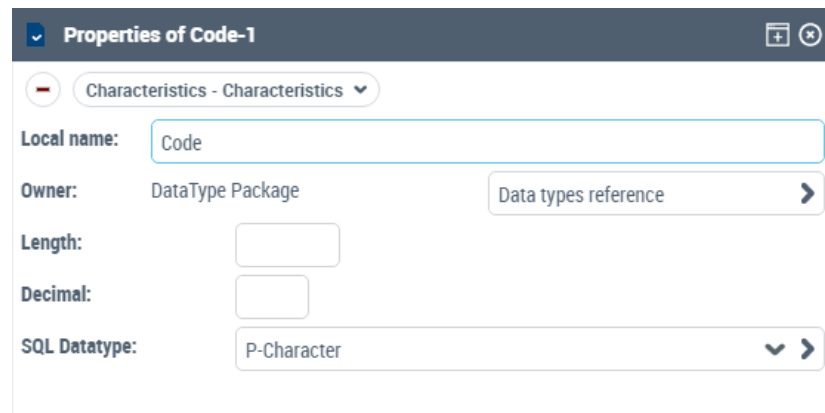
Connecting a Datatype to a Pivot Type

Datatypes contained in the "Datatypes Reference" package and associated by default with all new data models are connected to these pivot types. Therefore when you create new datatypes, these must be connected to the corresponding pivot types so that they can subsequently be used at physical level.

To connect a datatype to a pivot type:

1. Right-click the datatype and select **Properties**.
The properties dialog box of the datatype properties dialog box appears.
2. Click the drop-down list then **Characteristics**.
3. In the **SQL Datatype** field, select the pivot type.

Take the "Code" datatype. Open its properties dialog box and click the **Characteristics** page. In the **SQL Datatype** field, you can see that it is connected to the "P-Character" pivot type .



At synchronization of a logical model to a physical model, this pivot type "P-Character" will give a datatype CHAR, VARCHAR, LONG or TEXT depending on the DBMS concerned by synchronization. You can modify the target DBMS without having to modify the datatype, **HOPEX** assuring automatic conversion. See [Mappings Between Pivot Types and Datatypes](#).

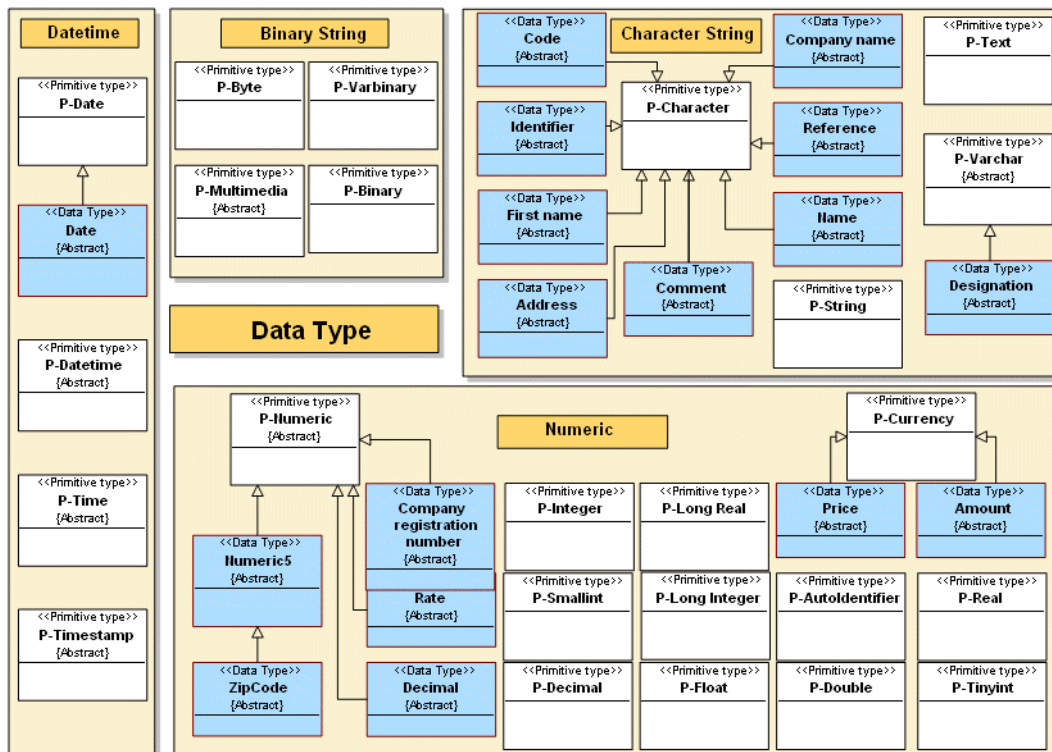
Connecting a Datatype to a Pivot Type in UML Notation

If you use UML notation and class diagrams to modify your data - and for reasons of compatibility with earlier versions of **HOPEX Database Builder** - other methods of referencing pivot types are possible.

- By inheritance
- By a correspondence link
- By an equivalence link
- By creating a compound datatype

By inheritance

You can define your own datatypes by declaring them as subclasses of the pivot types, as shown in the example below.



The datatypes defined as subclasses will automatically inherit the characteristics of their superclass. In particular, the datatype conversion rule for the superclass is applied to the subclass.

It is possible to specify a length and a number of decimal places for the subclass. These will be taken into account when generating the data types if they were not already defined for the superclass.

By a correspondence link

To create this link:

1. Open the properties dialog box of the class.
2. Click the drop-down list then **Generation** > **SQL**.

3. Indicate the **SQL Type** associated with the class.

☛ Only pivot types of the *Standard::Types::Pivot* package are proposed in the list.

4. You can also indicate the length and the number of decimal places to be applied.

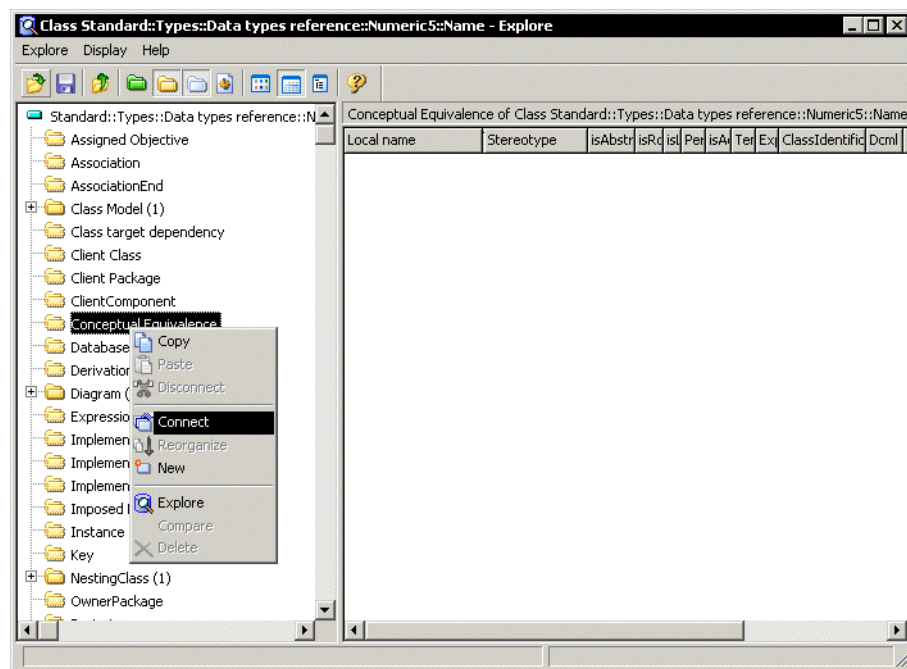
By an equivalence link

It is possible to define a new datatype by creating an equivalence link with a pivot type. To do this:

1. Create a new datatype
2. Right-click the type and select **Explorer**.
3. In the dialog box that appears, click the **Empty Collections** button.

☛ If necessary, click  to display the hidden commands.

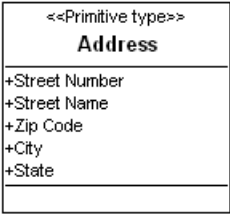
4. Right-click the "Conceptual Equivalence" folder and select **Connect**.



5. In the standard query dialog box, select the pivot type you wish to connect to your type.

By creating a compound datatype

You can define a compound datatype by assigning to it a list of attributes.

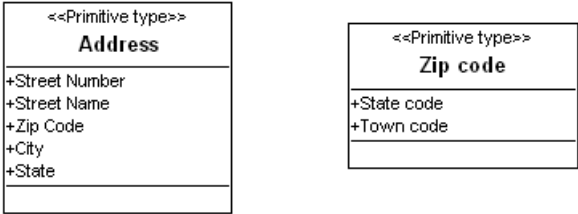


Here the "Address" type is composed of the number, street, zip code, city, and country.

The derivation of the "Address" attribute will produce these five columns.

It is possible to have several levels of compound types by assigning a compound type to an attribute of a compound type.

For example, the zip code can be broken down into the main five digits and the four-digit extension:



Mappings Between Pivot Types and Datatypes

Pivot types establish correspondence between the logical datatypes to which they are connected and the datatypes for which they have an equivalent in each target DBMS.

The equivalence links carry conditions that enable them to be distinguished one from the other.

To visualize correspondences between pivot types and the different DBMS datatypes, see [Pivot Types and Datatypes Correspondence Tables](#).

Example of correspondence between pivot types and Oracle 8 datatypes

Pivot to Datatype

Pivot	Condition	Datatype
P-AutoIdentifier		NUMBER
P-Binary		RAW(@L)
P-Boolean	L=2 or L ø	RAW(1)
	L>1	RAW(@L)
P-Byte		RAW(1)
P-Character	L=256 or L ø	CHAR(@L)
	L>2000	LONG
	255<L<2001	VARCHAR2(@L)
P-Currency		NUMBER(@L,@D)
P-Date		DATE
P-Datetime		DATE
P-Decimal		NUMBER(@L,@D)
P-Double		NUMBER(@L,@D)
P-Float		NUMBER(@L,@D)
P-Integer		NUMBER(@L)
P-Long Integer		NUMBER(@L)
P-Long Real		NUMBER(@L,@D)
P-Multimedia		LONG RAW
P-Numeric	L=0 or L ø	NUMBER
	L>0 and D ø	NUMBER(@L)
	L>0 and D not ø	NUMBER(@L,@D)
P-Real		NUMBER(@L,@D)
P-Smallint		NUMBER(@L)
P-String		LONG
P-Text		VARCHAR2(@L)
P-Time		DATE
P-Timestamp		ROWID
P-Tinyint		NUMBER(@L)

Pivot	Condition	Datatype
P-Varbinary		LONG RAW
P-Varchar	L>2000 or L=0 or L ø	LONG
	0<L<2001	VARCHAR2(@L)

Datatype to Pivot

Datatype	Condition	Pivot
CHAR(L)		P-Character
DATE		P-Date
LONG		P-String
LONG RAW		P-Multimedia
NUMBER		P-Numeric
NUMBER(L)		P-Numeric
NUMBER(L,D)		P-Numeric
RAW(1)		P-Boolean
RAW(L)		P-Boolean
ROWID		P-Timestamp
VARCHAR2(L)		P-Varchar

In this table, we can see that the "P-Numeric" type has three correspondences for type classes using three different conditions on the equivalence links.

if P_Numeric is assigned to an attribute and the length of this attribute is 10, then the column justified by this attribute via the synchronization will give Number(10).

The condition is written in VB Script language. The main elements of the condition are:

- **Sub ConditionInvoke (Column, ByRef bValid)**: the first line constitutes the signature of the function.
- **Column**: the column is given as an input parameter.
- **bValid** : is the return parameter. Its value is "True" if the condition is verified, "False" if not.

Example:

```
Sub ConditionInvoke (Column, ByRef bValid)
    bValid = False
    If (IsNumeric(Column.Length)) Then bValid = True
End Sub
```

The following can be specified in the condition:

- Presence of a number
- Presence of a decimal
- Range concerned (example: between 0 and 150 inclusive)

Creating New Datatypes

Each datatype is implemented in the form of a class; it is specific to a DBMS version. It is possible to use masks with datatypes.

Example for Oracle 10

Objective

In the ORACLE scripts, create a numeric datatype called Data8 with a length and a specified number of decimal places.

Steps

Steps are as follows:

1. Create a new datatype in **HOPEX**.
2. Connect the datatype to the target DBMS (in this case Oracle 10).
3. Connect the datatype to the corresponding type in the "Pivot" package.
4. Configure the conditions on each link in both directions (from datatype to pivot type and vice-versa).

➡ For more information on equivalence links and conditions, see [Determining Column Datatypes from Attribute Types](#).

Prerequisite Conditions

To see packages containing DBMS datatypes, you must import the corresponding module. See [Importing a DBMS Version](#).



It is recommended that a datatype be defined in only one DBMS version.

To import the module of datatypes:

1. From your HOPEX version, open the HAS Console.
2. From the navigation menu, click **Modules**.
The HAS console displays:
 - installed modules
 - the store where you can download modules
 - modules to be updated
3. In the store, search for the module to install and download it.

In addition, certain data is protected in **HOPEX**. To be able to modify objects contained in DBMS packages:

1. Open the administration desktop with the HOPEX Administrator profile.
2. In the upper right-hand corner of the desktop, click on the menu associated with the administrator's account and then click **Options**.
3. In the options window, expand the **Installation** folder.

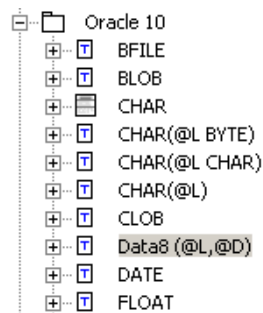
4. Click the **Customization** folder.
The list of options linked to customizing appears in the right pane of the window.
5. In the **Authorize HOPEX data modification** box, select "Authorize".
6. Click **OK**.

Creating a new datatype

To create a new datatype in **HOPEX Data Governance**:

1. Click the navigation menu then **Architecture > Hierarchy View**.
☛ In HOPEX Information Architecture, click the navigation menu then Data Architecture > Physical Data Assets.
2. Expand the **Data Dictionaries** folder.
3. Click the icon of the "Oracle 10" package and select **New > Class**.
The **Creation of Class** dialog box opens.
4. Name your class "Data8 (@L,@D)".
5. Open the properties dialog box of this new class.
6. In the **Characteristics** page, select "Expression" in the **Stereotype** field.
7. In the **Type expression** field which appears, a little lower, select the value "Data8 (@L,@D)".

You can see in the navigator that a new class "Data 8" has been created.



☛ This new class is automatically created for UML operating requirements..

Connecting the datatype to the pivot type

If you wish to obtain this datatype after synchronization, you must give it an equivalence at the logical level.

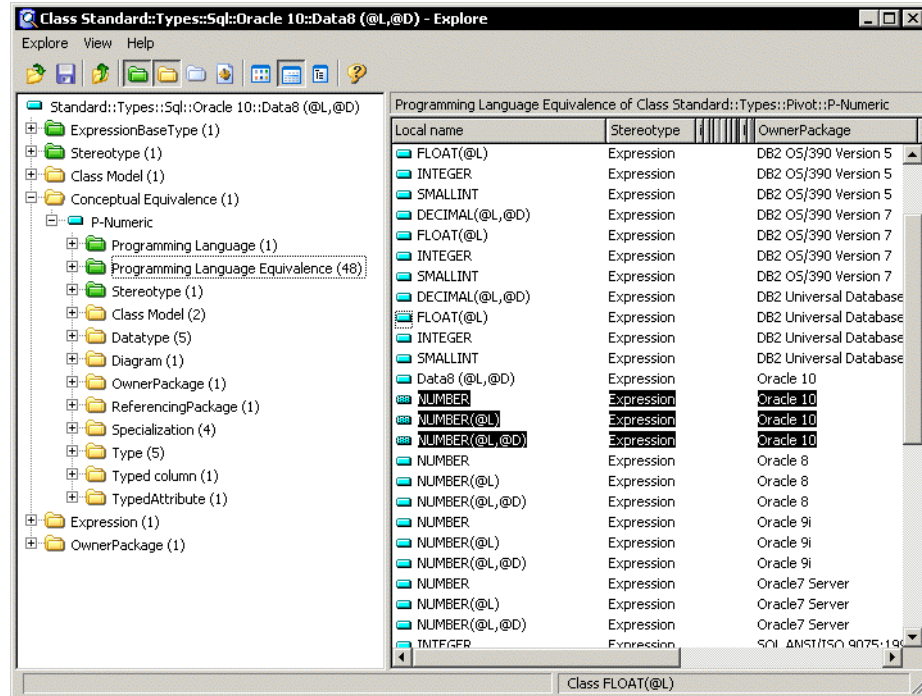
1. Open the properties dialog box of datatype "DATA8 (@L,@D)".
2. Click the drop-down list then click **Complements**.
3. Right-click the "Conceptual Equivalence" folder and select **Connect**.
4. In the query dialog box, select the class "P-Numeric".

Configuring conditions on links

To configure a condition on links:

1. Right-click the "Data8" class and select **Explorer**.
2. Expand the "Conceptual Equivalence" folder.

3. Select the green folder "Programming Language Equivalence".
You will see that there are three other cases of correspondence for Oracle 10.



Conditions on these correspondences must therefore be modified so that they will be coherent with the conditions placed on the new datatype.

4. Right-click the "NUMBER(@L,@D)" datatype and select **Properties**.
5. In the **Texts** page, select "Language equivalence condition", and modify the text as follows:

```
Sub ConditionInvoke (Column, ByRef bValid)
    bValid = False
    Dim IsNumericLength
    IsNumericLength = IsNumeric(Column.Length)
    Dim IsNumericDecimal
    IsNumericDecimal = IsNumeric(Column.Decimal)
    If (IsNumericLength and IsNumericDecimal) Then
        If (Column.Length <> 8) Then
            bValid = True
        End If
    End If
End Sub
```


6. In the same way, add the following text in the properties dialog box of new datatype "Data8".

```
Sub ConditionInvoke (Column, ByRef bValid)
    bValid = False
    Dim IsNumericLength
    IsNumericLength = IsNumeric(Column.Length)
    Dim IsNumericDecimal
    IsNumericDecimal = IsNumeric(Column.Decimal)
    If (IsNumericLength and IsNumericDecimal) Then
        If (Column.Length = 8) Then
            bValid = True
        End If
    End If
End Sub
```

Verifying datatypes

To verify datatypes:

1. Display the properties of a column concerned by conditions placed on the datatype.
2. Verify that the mask displayed in the **Datatype** column is "Data8 (%l, %d)" for this column.

Example for SQL Server 7

Objective

Reread SQL Server 7 columns containing a non-standard datatype.

Manipulations are the same as for Oracle (see [Example for Oracle 10](#)). On this occasion we shall not create a mask.

Creating a new datatype

To create a new datatype in **HOPEX Data Governance**:

1. On the desktop, click the navigation menu then **Architecture > Hierarchy View**.
2. Unfold the **Data Dictionaries** folder.
3. Click the icon of the "Oracle 10" package and select **New > Class**. The **Creation of Class** dialog box opens.
4. Name your class "TLongName".
5. Open the properties dialog box of this new class.
6. In the **Characteristics** page, select "Expression" in the **Stereotype** drop-down list, then click **OK**.

Connecting the datatype to the pivot type

To connect the datatype to the primitive type:

1. Open the properties dialog box of the "TLibelleLong" datatype.
2. Select the **Complements** page.

3. Right-click the "Conceptual Equivalence" folder and select **Connect**.
4. In the query dialog box, select the "P-Text" class.

Configuring conditions on links

To configure a condition on links:

1. Right-click the "TLibelleLong" class and select **Explorer**.
2. Select the green folder "Programming Language Equivalence".
You will see that there is another correspondence for SQL Server 7.
3. Open the properties dialog box of datatype "text".
4. In the **Texts** page, select "Language equivalence condition", and modify the text as follows:

```
Sub ConditionInvoke (Column, ByRef bValid)
    bValid = False
    Dim IsNumericLength
    IsNumericLength = IsNumeric(Column.Length)
    If (IsNumericLength) Then
        If (Column.Length > 255) Then
            bValid = True
        End If
    End If
End Sub
```

5. In the same way, add the following text in the properties dialog box of new datatype "TLongName".

```
Sub ConditionInvoke (Column, ByRef bValid)
    bValid = False
    Dim IsNumericLength
    IsNumericLength = IsNumeric(Column.Length)
    If (IsNumericLength) Then
        If (Column.Length <= 255) Then
            bValid = True
        End If
    End If
End Sub
```

DATABASE MODELING RULES

HOPEX Information Architecture provides rules that enable database modeling checks. The physical regulation contains the rules relating to the database relational diagram. It is used to check the corresponding relational diagram in a DBMS.

The physical regulation contains rules relating to technical specifications of the database DBMS. It is used to check consistency of physical parameters of the relational diagram specific to the DBMS.

Checking a database

You can run a check on the database or on a database object.

To check a database:

1. Right-click the name of the database.
2. Select **Administer > Check > Regulation with Propagation**.
When several regulations can apply to the check object, a dialog box asks you to select the required regulation.

The check applies to the database as well as the objects it owns.

Results appear in an HTML report.

For more details, see the **HOPEX Common Features** user guide, "Exploring the repository", "Tools for Checking Objects".



DATA CATEGORIZATION



You can classify repository data by category. A dedicated tree lists the various categories and associated data. Data thus classified can be used in the **HOPEX Privacy Management** solution specific to sensitive data and compliance with the RGPD.

DEFINING DATA CATEGORIES

The **HOPEX Data Governance** and **HOPEX Information Architecture** solutions provide a list of categories that is used to classify data. You can complete this list by importing the **Sample Privacy Management Content** module.

➡ For more information on importing a module, see [Importing a Module into HOPEX](#).

You can also create categories.

Examples of data classification:

- Sensitive data
- Reference data
- Confidential data
- etc.

Accessing Data Categories

To access the data categories in **HOPEX Data Governance**:

1. Click the navigation menu then **Architecture > Hierarchy View**.
2. In the edit area, unfold the **Data Categories** folder.

Creating a Data Category

To create a data category:

1. In the edit area, right-click the **Data Categories** folder.
 2. Click **New > Data Category**.
 3. In the creation wizard specify the name of the category and the owner if appropriate.
 4. Click **OK**.
- The category appears in the list. You can enter a description.

INDICATING THE CATEGORY OF A DATA ITEM

Defining a Category on a Business Data

A data item can belong to one or more data categories.

To indicate which category a data item belongs to:

1. Open the properties of the data item in question.
2. Click the **Characteristics** page.
3. In the **Data Categories** section, select the relevant category.

Associating a Data Category with a Regulation

You can indicate that a regulation relates to a category of data, for example, "sensitive" data. All data that falls into this category is subject to the regulation.

To define a data category on a regulation:

1. Click the navigation menu and then click **Compliance > Regulatory frameworks** or **Policy Frameworks**, depending on the type of regulation.
2. Open the regulation's properties.
3. Click the **Constrained Data Categories** page.
4. Click **New** to create a category or **Connect** to connect an existing category.

You can also define categories in **HOPEX** from data imported via an Excel file. See [Data Import and Export](#).



DATA RESPONSIBILITY



HOPEX Data Governance and **HOPEX Information Architecture** allow you to appoint people responsible (for design, quality, etc.) for the data in the repository.

- ✓ [Use of Data Responsibilities](#)
- ✓ [Defining Responsibilities for Data](#)

USE OF DATA RESPONSIBILITIES

Assigning responsible persons to data that flows within an organization offers several advantages:

- Track the progress of data design. See [Data Validation Workflow](#).
- Communicate and collaborate on data: requests to update or validate data can be sent to the designated managers. They then receive a notification and follow their tasks in **HOPEX**. See [Accessing Notifications](#).
- Ensure data traceability. See [Data Lineages](#).
- Evaluate the data, directly as an expert, or through an evaluation campaign, followed by action plans if necessary. See [Data Quality](#).

For this purpose, **HOPEX Data Governance** and **HOPEX Information Architecture** offer different default roles, such as Data Owner or Data Quality Manager.

Roles Applicable to Data

- **Chief Data Officer (DCO)**: responsible for data and its governance.
- **Data Owner**: this is the authority who decides on data access and use. The data owner can be the data designer, one of its users or a third party. The owner of the data may be the designer of the data, one of its users or a third party. Data stewards can ask data owners to check or complete the value of a field, for example to correct a data quality defect.
- **Data Designer**: the person responsible for designing an object (such as a package, data domain, database, etc.).
- **Data Engineer**: builds and maintains the tools and the infrastructures necessary for the analysis of data by data scientists. He/she creates solutions able to process large volumes of data while guaranteeing its security. He/she is the first link in the IT chain.
- **Data Scientist**: is responsible for bringing together the data designer (business and logical data) and the managers of the processes who use this data.
- **Data Quality Manager**: must ensure the relevant and usefulness of the enterprise data. To do so, he/she must implement data control procedures.
- **Data Steward** : guarantees the quality of data; this is the person who possesses the knowledge of the data and its metadata.

Accessing Notifications

To view your notifications in HOPEX Information Architecture:

1. Click the navigation menu then **Collaboration**.
2. Click the **My notifications** tile.

For more information on the platform's collaboration tools see [Communicating in HOPEX](#).

DEFINING RESPONSIBILITIES FOR DATA

When a data is created (business, logical or physical), a **Designer** and an **Owner** are automatically assigned to it. By default it is the person who creates the data, but it is possible to assign other people to it.

☛ On certain object types in **HOPEX Information Architecture**, such as concept information elements for example, assignments are not automatic and must be made explicitly if necessary.

Viewing the Persons in Charge of an Object

To access information concerning the assignment of an object:

1. Select the object in question and click the **Properties** button.
2. In the Properties window, select the **Assignment** page.

☛ Only assignable objects have an **Assignment** page.

Automatic Assignment of an Object

When an object is created, an assignment is also automatically created. The object is assigned to the person who created it with the **Data designer** and **Data Owner** role.

These objects are visible in the object lists of the user in question (for example **My concepts**, **My concept domains**, etc.).

Explicit Assignment of an Object

You can explicitly define the assignment of an object to an existing person.

To assign an object to a person:

1. Open the properties dialog box of the assignable object.
2. Select the **Assignments** page.
3. click **New**.
The **Create Assignment** dialog box opens.
4. Using the **Person or Person Group** field, click **Connect**.
A **Connection** window opens.
5. Find and select the person that interests you and click **Connect**.
6. In the **Create an assignment** window, select the **Business role** of the person that you have selected.

☛ For more details on the business roles used for assignments, see [Business Roles of HOPEX Data Governance](#).

7. Click **OK**.
A new assignment is added to the list of assignments associated with the object.

Consulting Data Governance reports

Data governance reports consist of a set of reports that list the number and the details of the data items concerned for each business role.

Business roles are the following:

- Chief Data Officer (DCO)
- Data owner
- Data Designer
- Data Scientist
- Data Quality Manager
- Data Steward

Each report lists, in a bar graph, the persons who hold the role in question, for example the persons who are data owners. It displays, for each of them, the details of the information for which they are responsible.

Under the report, additional details are displayed for the data manager:

- name
- email
- telephone number
- amount of information allocated

When you click on a bar in the graph, the list of information appears with the following details:

- data name
- comment
- design progress rate
- owner



USE OF DATA BY THE INFORMATION SYSTEM



HOPEX Data Governance allows you to define and visualize which data is used by which processes and applications.

DEFINING THE DATA USED BY APPLICATIONS

HOPEX Data Governance allows you to make an inventory of applications that use the data you have modeled.

Thanks to the HOPEX integrated platform you can use this inventory in the HOPEX solutions specific to the description of the application architecture, such as **HOPEX IT Architecture** or **HOPEX IT Portfolio Management**.

Dedicated reports allow you to visualize in which applications certain data is used. See [Data Usage Reports](#).

Creating the Inventory of Application Assets

The inventory of application assets is carried out by the IA functional administrator.

To create an application in **HOPEX Data Governance**:

1. Click the navigation menu then **Architecture > Data Usage**.
2. In the edit area, click the **Data-using Applications** tab.
3. Click the **Applications** sub-tab then click the **New** button.
4. Enter the name of the application and an owner if necessary.
5. Click **OK**.

In the same way you can create application systems, application services, micro-services and application flow scenarios.

For more details on objects of the application architecture, see [Modeling Applications and System Architectures](#).

Connecting Data to an Application

The Data Manager has the ability to define the data that is used by the applications.

To connect logical data to an application - or any other application asset object - you must create a data store on the application. The data store provides a mechanism for storing and consulting data through the application.

The data store references an application data area that represents the structure of the data that the application needs.

On the application, you can create internal data stores (used only within the described system) or external data stores.

These data stores can be logical (ER) or physical.

Creating a data store on an application

To create a data store on an application:

1. Open the properties of the application in question.

2. Click the **Structure** page.
3. Go to the section that corresponds to the type of component to be created:
 - Internal components for a local data store.
 - Boundary components for an external data store.
4. Select the type of data store: logical or physical.
5. Click the **New** button.
6. In the window that appears, select the data area referenced by the data store and click **OK**.
The new data store appears in the application properties.

Defining data access mode

In the data area properties you can define how to access the data (create, read, delete, etc.).

To access the properties of a data area:

1. In the application properties, click the data store icon then click **Properties**.
Properties of the corresponding data area appear.
2. Click the **Components** page.
3. Select the component and select the check boxes that correspond to the types of access in question (Creation, Read-only, etc.).

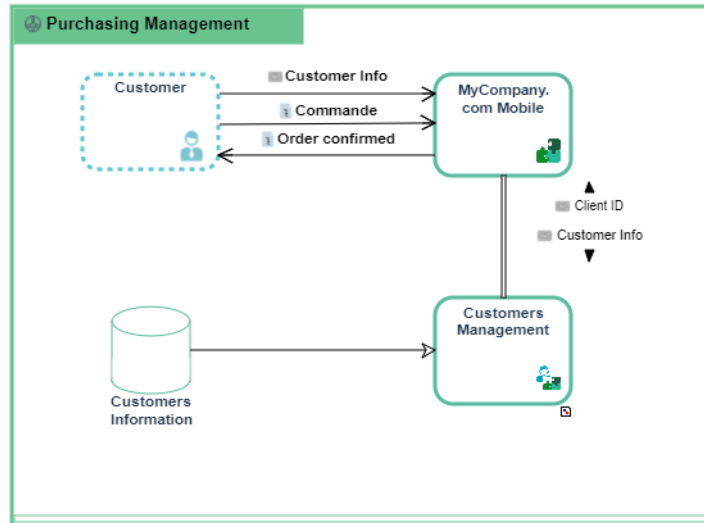
Connecting Data to an Application Flow Scenario

An application flow scenario presents the flows exchanged between the components of an application.

In the same way, an application system flow scenario represents the flows exchanged between the elements of the application system in a given context.

As for an application, connecting data to an application flow scenario involves creating a data store. The data store references an application data area that represents the data structure used in the scenario framework.

For example, below, the "Customer data" data store references the customer data structure (that can be comprised of the details of each client) used by the "Customer management" application service.



Creating a data store on an application flow scenario

To create a data store on an application flow scenario:

1. Click the navigation menu then **Architecture > Data Usage**.
2. In the edit area, click the **Data-using Applications** tab.
3. Click the **Scenarios** sub-tab.
4. Open the properties of the scenario in question.
5. Select the **Scenario** page.
6. Select the type of data store: internal or external.
7. Click the **New** button.
8. In the window that appears, select the data area referenced by the data store and click **OK**.

The new data store appears in the application properties.

DEFINING THE DATA USED BY PROCESSES

HOPEX Data Governance allows you to make an inventory of processes that use the data you have modeled.

Thanks to the HOPEX integrated platform you can use this inventory in the HOPEX solutions specific to the description of the processes, such as **HOPEX Business Architecture**.

Creating the Inventory of Processes

Inventory of processes is carried out by the IA functional administrator.

Process Types

Value chain

A value stream is a collection of Value Stages that creates an outcome for a customer, who may be the ultimate customer or an internal end-user in the value stream.

Value streams reference only business data areas.

Organizational Process

An organizational process is a set of operations performed by org-units within a company or organization, to produce a result. It is depicted as a sequence of operations, controlled by events and conditions.

Organizational processes can reference business data or logical data areas.

System process

A system process is the executable representation of a process. It defines a sequence of tasks.

You can connect the following data areas to it:

- Application data area
- Relational Schemas
- NoSQL data area
- File structure

See also:

[Logical and Application Data Domains](#)

[Physical Data Maps and Relational Schemas](#).

Creating a Process

To create a process with **HOPEX Data Governance**:

1. Click the navigation menu then **Architecture > Data Usage**.
2. In the edit area, click the **Data-using Processes** tab.
3. Select the type of process to be created, for example, System process.
4. Click the **New** button.
5. Enter the name of the process and an owner if necessary.
6. Click **OK**.

Connecting Data to a Process

The Data Manager has the ability to define the data that is used in the processes.

To connect logical data to a process you must create a data store on the process. The data store provides a mechanism for storing and consulting data through the process.

The data store references a data area that represents the structure of the data that the process needs.

Creating a data store

To create a data store on a process:

1. Open the properties of the process in question.
2. Click the **Data Store** page.
3. click **New**.
4. In the window that appears, select the data area referenced by the data store and click **Next**.
The new data store appears in the process properties.

Defining data access mode

In the data area properties you can define how to access the data (create, read, delete, etc.).

To access the properties of a data area:

1. In the process properties, click the data store icon then click **Properties**. Properties of the corresponding data area appear.
2. Click the **Components** page.
3. Select the component and select the check boxes that correspond to the types of access in question (Creation, Read-only, etc.).

See also: [Managing Data](#).

Connecting Data to a Content

The content corresponds to information exchanged between a process and its environment, via flows.

You can connect business or logical data to a flow; it is defined as components of the content.



SYNCHRONIZING LOGICAL AND PHYSICAL MODELS



Synchronization is a process that translates a class diagram expressed in classes/parts formalism to a physical model expressed in relational formalism, and vice-versa. It therefore ensures correspondence of objects in the two models.

 A compatibility option enables to synchronize a data model (entities/associations) and a physical model. See [Logical Formalism and Synchronization](#).

The procedure should be carried out periodically. Throughout the modeling project, these models are each subject to their particular changes. Synchronization intervenes when we wish to compare the two models and automatically restore canonical mappings that connect them.

The synchronization functionality is available with "Advanced" access to the **HOPEX** repository.

 **Synchronization of models of a database can be in one direction or another - either in physical > logical direction or in logical > physical direction, but not both at the same time. When synchronization direction has been determined, synchronization should not be reversed. Mapping justification between the logical and physical levels is not guaranteed if this rule is not followed.**

See also [Model Mapping](#).

- ✓ ["Logical to Physical" Synchronization Rules](#)
- ✓ [From the Logical Model to the Physical Model](#)
- ✓ [Reduced Synchronization \(Logical to physical mode\)](#)
- ✓ [Running Synchronization After Modifications](#)
- ✓ [From the Physical Model to the Logical Model](#)
- ✓ [Configuring Synchronization](#)
- ✓ [Diagram Synchronization](#)

Synchronization Display Options

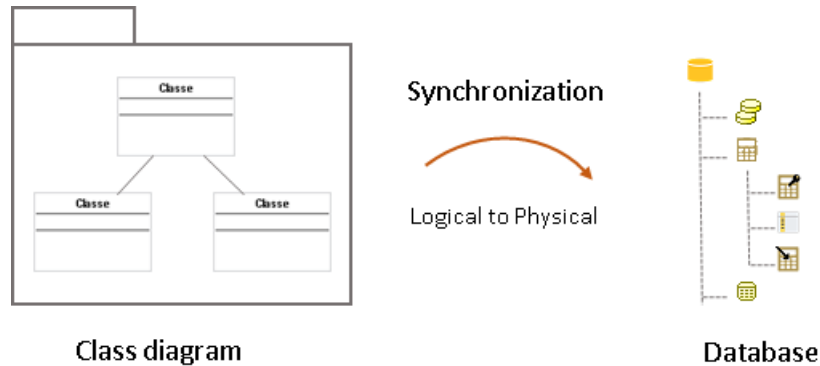
Certain synchronization options are filtered by default. To display these:

1. Click **Main Menu > Settings > Options**.
2. In the **Options** tree in the left pane, unfold the **HOPEX Solutions > Information Architecture** folder.
3. Click the **Database synchronization** sub-folder.
4. In the right pane, check the desired synchronization options:
 - Synchronization (Logical to Physical)
 - Synchronization (Physical to Logical)
 - Reduced synchronization (Logical to Physical)
 - Reduced synchronization (Physical to Logical)

Note that the UML notation is applied by default in the synchronization. See [Formalisms](#).

"LOGICAL TO PHYSICAL" SYNCHRONIZATION RULES

The following rules are applied for transforming class diagrams into relational formalism.



See also [Configuring Name Generation](#) and [Data Types and Column Datatypes](#).

Logical > physical synchronization: the application and logical data areas

In logical > physical mode, the synchronization of application and logical data areas gives rise to relational data areas.

See also [Physical Data Areas](#).

Logical to Physical Synchronization: the Entities (or Classes)

In Logical to Physical mode, classes and entities are processed in the same way by the synchronization tool.

🔧 **By default, the synchronization tool applies the class diagram logical formalism. See [Logical Formalism and Synchronization](#).**

General rule

- Any non-abstract entity of the model becomes a table.
- The entity identifier becomes the primary key for the table. If the identifier is implicit, a column is automatically created. See [Configuring Name Generation](#).
- Entity attributes become columns in the table.
- Mapping rules are applied to determine the column datatypes from the datatype (DM) of each attribute. The possible configurations depend on the DBMS.

➡ For more detailed information, see [Data Types and Column Datatypes](#).

Sub-entity

- The foreign key reflecting dependency between the sub-entity and its super-entity is created.

Abstract entity

An abstract entity does not produce a table at synchronization.

If constraint associations point to an abstract entity, the corresponding foreign keys are not created, but columns corresponding to the foreign keys are created to respect table integrity.

When a sub-entity is abstract, all columns and foreign keys of the corresponding table are taken by the table corresponding to the super-entity.

Conversely, when a super-entity is abstract, all columns and foreign keys of the corresponding table are taken by the table corresponding to the sub-entity.

To define an abstract entity:

1. Open the properties dialog box of the entity.
2. Click the **Characteristics** tab.
3. In the **Abstract** box, select "Yes".

Realized entity

An entity is said to be realized if it produces creation of a table at synchronization.

A "not realized" entity is treated as an abstract entity.

Unlike the "abstract" property which characterizes the entity in all use cases, the "realized" concept applies only in the context of database synchronization. See [Realized mode](#).

Logical to Physical Synchronization: the Associations

Synchronization of associations is available with the former UML formalism which takes into account associations but not parts. See [Logical Data Modeling Options](#).

Constraint associations (multiplicities: 0,1 or 1,1)

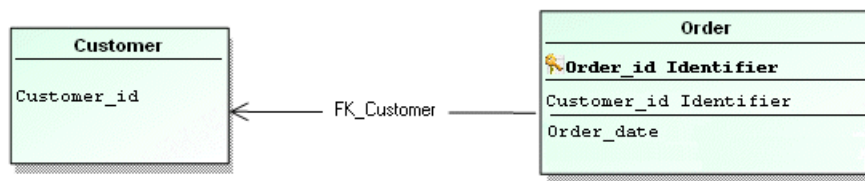
A constraint association is a binary association one role of which has maximum multiplicity 1. In this case, it is not necessary to create a table corresponding to the association. Simply add a column to the table that corresponds to the entity.

A constraint association (one of its maximum multiplicities is 1) does not result in a table. In the following example, an order has only one customer.



Synchronization of this data diagram produces one of the two following results:

The association does not result in a table.



- A column corresponding to the key for the "Customer" entity is created in the "Order" table.
- A column is also created for each attribute of the association.
- A foreign key "FK_Customer" is added to check the "Customer_ID" column in the "Order" table. It indicates that the possible values for the "Customer_ID" column in the "Order" table are the values that already exist in the "Customer ID" column of the "Customer" table.

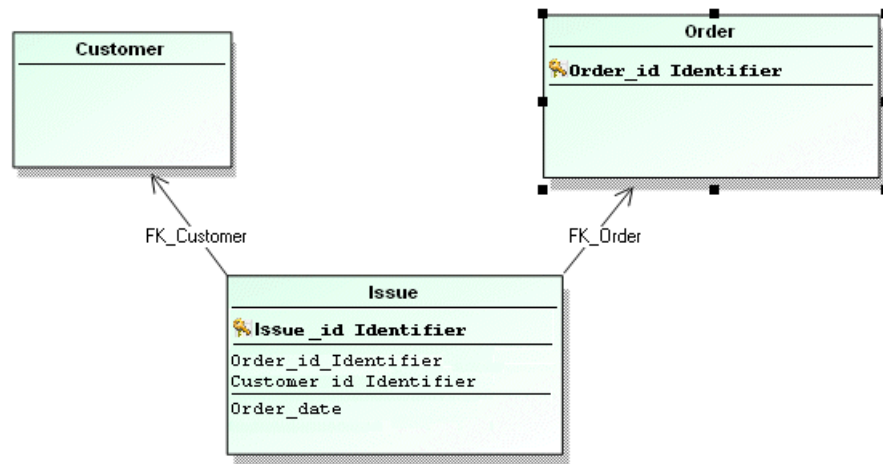
The foreign key is created from the entity identifier.

The association is transformed to a table

For the association to be transformed into a table:

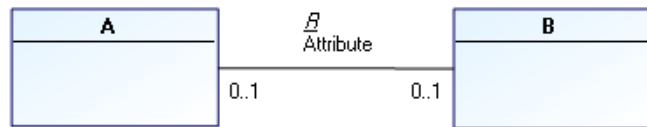
1. Open the properties dialog box of the association.
2. Click the **Characteristics** tab.

3. In the **Potential Mapping** field, select "Table".



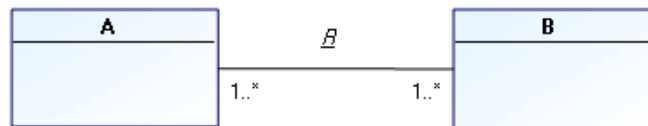
Constraint associations (multiplicities: 0,1 and 0,1)

In this particular case, the combination of multiplicities is ambiguous. There is nothing that can be used to decide which table should contain the column corresponding to the attribute.



Synchronization proposes a column in each table.

Deadlocks



The multiplicities 1..X, 1..X indicate that each of the two objects must be connected to at least one object of the other type in order for it to exist.

This poses problems when creating the first object of each type. In fact:

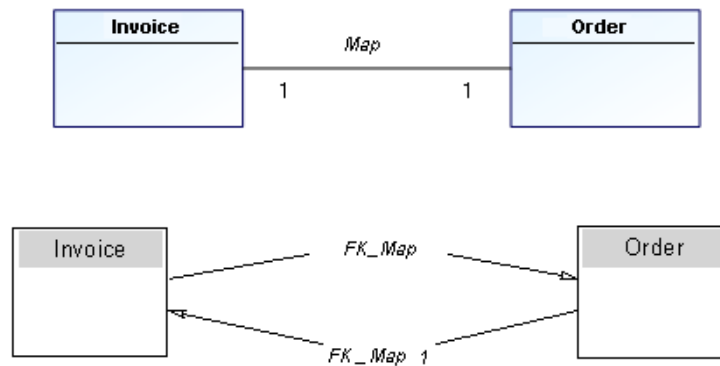
- An object of type A must exist in order to create an object of type B and then connect them.
- Conversely, an object of type B must exist in order to create an object of type A and then connect them.

This is the case for the following multiplicity combinations:

- Multiplicities of 1..*, 1..*
- Multiplicities of 1, 1..*

However, it is not physically impossible to resolve such cases, because the problem is limited to creating the first object of each type. In addition, no foreign key is generated for verifying data integrity in the first case, and only one in the second case, so there is no resulting deadlock situation.

Multiplicities 1, 1 generate several obligatory foreign keys that will become deadlocked:



This situation results in complete deadlock, because in order to meet the constraints, several tables must be created at the same time.

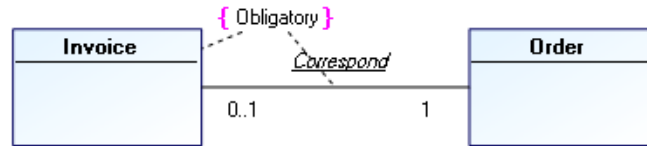
Certain DBMSs prohibit creation of tables of this type.

For correct synchronization, situations such as this should be avoided.

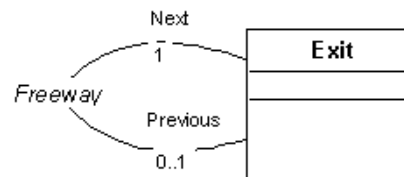
☛ **Select one of the foreign keys and assign it a multiplicity of 1, then assign 0..1 to the other. You can also pull the foreign key from the table after synchronization, but this is less convenient.**

You can still impose the minimum multiplicity of 1 by adding a constraint as shown below. This constraint will not be taken into account in the synchronization.

➡ See [Constraints](#) for further information.

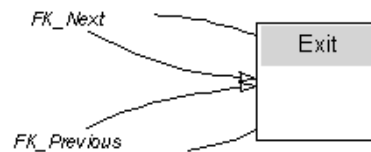


Here is another example using a reflexive link: Reflexive link 1, 1 or 0..1, 1



We want to show that each exit from a freeway comes before an exit and comes after an exit.

When modeled this way, all exits must be created at the same time, because each exit must have a previous exit already created.



To avoid this, set the multiplicities to 0..1 and 0..1.

Non-constraint association

An association where maximum multiplicities are not 1 will have a corresponding table:

- A column is created for each attribute of connected entities identifiers.
- The primary key for the table uses all these columns.
- A foreign key is also built for each connected entity.
- An additional column is created for each attribute of the association.

Before starting synchronization it is advisable to check validity of the data diagram and check that synchronization configuration is correct. See [Preparing Synchronization](#).

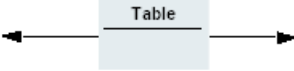
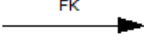
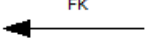
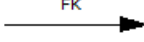
Association class

Associations connecting associative classes are not included in synchronization.

Logical to Physical Synchronization: the Parts (UML)

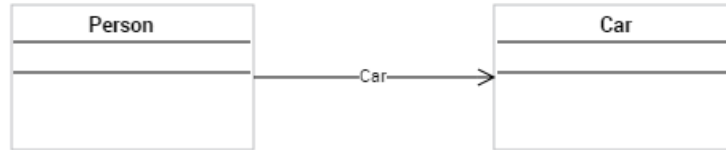
Synchronization of parts is available by default with the new UML formalism.

The result of the synchronization is determined by the combination of the **Whole/Part** link (None, Aggregation, Composition) and the **Multiplicity** defined on the part.

Multiplicity	Whole/Part link	
	Aggregation  Composition 	None
None (*) 2 / 6 1..*	The part gives rise to a foreign key to the owner class 	The part gives rise to a table between the two classes 
1 0 / 1	The part gives rise to a foreign key to the referenced class  and gives rise to a foreign key to the owner class 	The part gives rise to a foreign key to the referenced class 

Example 1: None / *

In the following example, the "Person" class references the "Car" class, without multiplicity constraint.

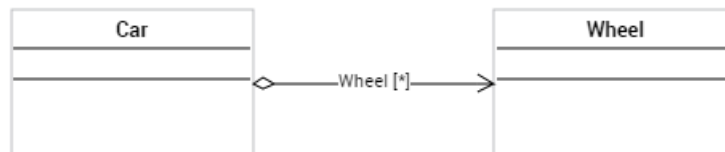


After synchronization, the "Car" part gives rise to a table:

- A column is created for each attribute of connected entities identifiers.
- The primary key for the table uses all these columns.
- A foreign key is also built for each connected entity.

Example 2: Aggregation / *

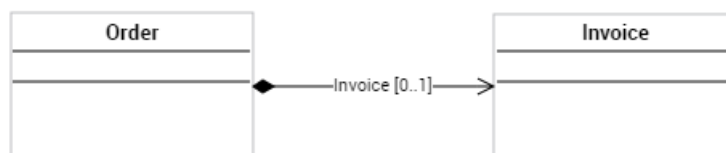
A car can have one or several wheels.



After synchronization, the part gives rise to a foreign key to the "Car" class.

Example 3: Composition / 0..1

An order contains an invoice.



After synchronization:

- A foreign key references the "Invoice" table in the "Order" table.
- A foreign key references the "Order" table in the "Invoice" table.

FROM THE LOGICAL MODEL TO THE PHYSICAL MODEL

This section explains how to synchronize the logical model of a database (represented by a data diagram) with the corresponding physical (relational) model.

Although synchronization of the relational model from the data diagram essentially concerns entities, it can also be carried out more generally from classes of a class diagram.

Synchronization in the reverse direction, from a physical model to a logical model, is also possible but not on the same database. When synchronization direction has been selected for an object, synchronization in the other direction will no longer be possible.

See [From the Physical Model to the Logical Model](#).

The points that follow present synchronization of a database. You can also run reduced synchronization, in other words on a specific object of the database. See [Reduced Synchronization \(Logical to physical mode\)](#).

Running Synchronization

Logical to Physical synchronization consists of building the physical model from the logical model, in other words of creating tables and columns corresponding to data diagram entities and attributes:

The synchronization tool is available in the **Tools > Data Synchronization** navigation pane. You can also open the synchronization tool directly from the database concerned.

To run a Logical to Physical synchronization on a database:

1. Click the icon of the database and select **Synchronize**.
The synchronization wizard opens.
2. Select the synchronization type "Logical to physical".

Step 1: Selecting the source objects to be synchronized

To define synchronization scope:

1. In the logical view tree, expand the list of objects contained in the database.

- By default, all objects are selected and therefore included in synchronization. To exclude an object from the synchronization, clear it from the **Scope** column. When an object is excluded, its mapping is also excluded.

DataBase Selection

Define the scope of your synchronization:

	Scope	Not Realiz...	Name	ExpressionType
Order Management DB	☒		HBC Group::Sales::...	
Order management (D...	☒		HBC Group::Sales::...	
Data diagram	☒		HBC Group::Sales::...	
Article	☒		HBC Group::Sales::...	
Article type	☒		HBC Group::Sales::...	
Catalog	☒		HBC Group::Sales::...	
Client	☒		HBC Group::Sales::...	
Company	☒		HBC Group::Sales::...	
Order	☒		HBC Group::Sales::...	
Order line	☒		HBC Group::Sales::...	
Person	☒		HBC Group::Sales::...	
Discount	☒		Article Discount Cli...	

Previous

- By default, all the objects are "realized", in other words, they give way to the creation of an object during synchronization. To specify that an object is "not realized", select it in the **Not realized** column. For more detailed information, see [Realized mode](#).
- When the list of objects has been defined, click the **Next** in the wizard.

Step 2: Synchronization options

From the synchronization options, you can:

- in a case where objects of the logical view have already been synchronized, synchronization will start from zero and will delete existing target objects.
 - ☛ **When models have already been synchronized, so that mappings are taken into account at a new synchronization, make sure the "Target object reinitialization" option is cleared.**
- Recalculate target object names: names of physical objects are recalculated as a function of those of the source objects. This means that any manual modification of physical object names is canceled.
- Take account of optimizations: all optimizations - including those not selected in the validation step (see step 4) are proposed.
- Take account of deletions: entities, associations and diagrams that have been deleted are included in the scope. Consequently, deletion of corresponding target objects and links is proposed.
See [Using Options](#).

Other options concern target object properties update. By default, synchronization updates all properties of each object concerned.

Scheduling

You can run synchronization:

- Immediately
 - As soon as possible (after publication of updates)
 - At a predefined date and time
- 】 When options have been specified, click **Next**.

Step 3: Protecting objects

Synchronization can impact all objects in an existing database.

- 】 To keep an object intact, select it in the **Frozen**.
- 】 Click **Next** to continue.

See [Protecting Objects](#).

Step 4: Validating results

The wizard displays the results that will produce synchronization validation.

Objects that will be automatically modified are indicated by a tick.

Icons preceding object names indicate actions that will be executed on the objects.

Actions can be creation , deletion  or update .

An arrow  preceding an object indicates that synchronization has an impact on sub-objects of the object in question.

- 1 Expand the object to view the modifications concerned.

Validating optimizations

Optimizations are customizations on objects thus removed from automatic synchronization processing.

Optimization examples:

A tick indicates objects that will be modified. If you do not wish to validate modifications relating to certain objects, you must clear the corresponding boxes. This optimization is kept at subsequent synchronizations.

In addition, **HOPEX** deduces optimizations following actions you may have carried out manually. If you have added a table in the physical view without having created the corresponding object in the logical view, synchronization does not select

deletion of this table.  Table1 

So that the object will be deleted, you must select the corresponding box.

- 1 When actions on target objects have been defined, you can click **Next**.

A report shows actions carried out.

You can close the wizard and view the results in the editor.

Using Options

The combination of the "Take account of optimizations" and "Take account of deletions" options varies depending on the scope of objects you wish to update.

Take account of optimizations

When this option is selected, synchronization proposes all creations, deletions and modifications, including optimizations not selected by default at the validation step.

When the option is cleared, only the modifications selected by default are proposed at the validation step.

This option enables filtering of the synchronization result to present only those modifications that have a real impact on target data.

Take account of deletions

When this option is selected, synchronization includes entities, associations and diagrams that have been deleted. Consequently, deletion of corresponding target objects and links is proposed.

When this option is cleared, the entities, associations and diagrams that have been deleted are not included. Consequently, the corresponding target objects are not modified.

 **This option only applies to entities, associations and diagrams. For other deleted object types (attributes, identifiers, etc.), the impact on target objects and links is conditioned not by this option, but by the object that contains them.**

This option enables limitation of impact of a synchronization strictly to the source scope defined by the user, excluding any object not explicitly declared in the scope. This option can be associated with synchronization scope for use case types.

Possible option combinations

1. "Take account of deletions" option selected and complete synchronization scope

This is a use case that favors complete synchronization between source and target. In this case, all objects have a valid mapping on completion of each synchronization wizard operation. This mode should be used when source and target should be totally consistent.

2. "Take account of deletions" option selected and partial synchronization scope

This use case enables working on the selected scope, while including impact of deleted objects. In particular it enables confirmation of deletions of target objects following the deletion of source objects of entity, association or diagram types: as these source objects have been deleted, it is theoretically not possible to include them in the synchronization scope. Selecting this option makes this choice possible. This mode should be favored when the scope is wide, and the few objects excluded from the scope are only excluded temporarily (for example, new objects for which we wish to delay the impact on the target).

3. "Take account of deletions" option selected and empty synchronization scope

This is a special mode enabling "cleanup" of target objects whose mapping is no longer valid, with no other impact.

4. "Take account of deletions" option not selected and partial synchronization scope

This use case enables working strictly on the selected scope, excluding any impact outside this scope. In particular it avoids deletions of target objects following the deletion of source objects of entity, association or diagram types: as these source objects have been deleted, it is theoretically not possible to exclude them in the synchronization scope. Clearing this option makes this choice possible. This mode should be favored for a specific synchronization on a restricted scope, which does not include total consistency of source and target. In addition, it is the fastest mode.

5. "Take account of deletions" option not selected and complete synchronization scope

This combination is in principle an infrequent use case. It corresponds to a work mode in which deletion of source objects has no effect on the target; in other words no target object created is ever deleted when this mode is activated.

6. "Take account of deletions" option not selected and empty synchronization scope

This combination has no effect.

Protecting Objects

You can protect an object so that synchronization will have no impact on it. This excludes the object from synchronization without it disappearing.

There are two object protection modes; one upstream and the other downstream.

Frozen mode

"Frozen" mode concerns the target object, that which results from synchronization.

When you freeze a relational model table, you also freeze all child objects of this table: no child object is created, modified or deleted by synchronization.

You can freeze objects:

- Before running synchronization, in the database editor.
- When running synchronization, in the options presented in the wizard.
See [Step 3: Protecting objects](#).

Realized mode

"Realized" mode concerns the source object of synchronization.

An object is said to be realized if it produces object creation at synchronization.

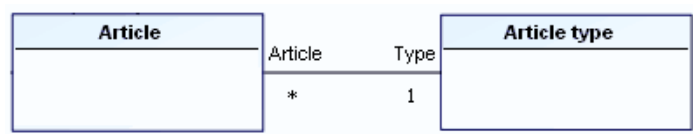
An object not realized does not produce object creation at synchronization but is treated as an abstract object. See [Abstract entity](#).

By default, all objects are realized.

You can exclude a source object from synchronization selecting the "Not realized" column on the object in question in the synchronization wizard. This action is available on entities, attributes and associations. The "realized" or "not realized" action is propagated to child objects.

Not realized entity example

The "Article" entity has an association to the "Article Type" entity.



The "Article Type" entity is said to be "not realized".

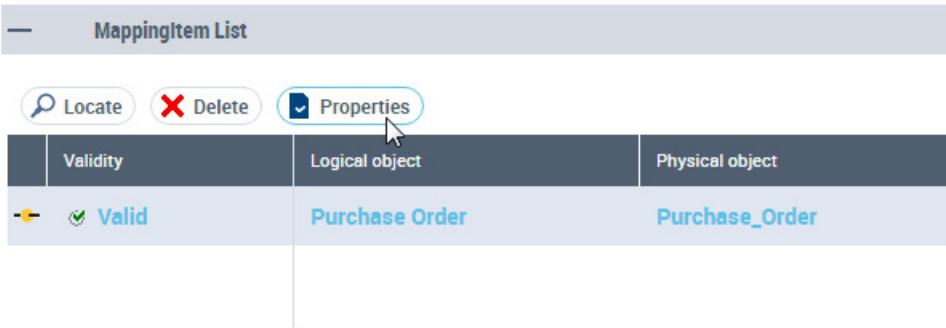
At synchronization, the "Article Type" entity does not produce creation of a table. In the "Article" table, the foreign key to "ArticleType" is not created; however the "Code_Type_Article" column is created. **Synchronization Results: Correspondences**

When synchronization is completed, the tables, columns, keys, and indexes of the physical diagram have been synchronized with the data diagram. They can now be viewed and the desired optimizations made.

Mapping characteristics

For more details on mapping:

- 1 In the mapping editor, select the object mapping element and click **Properties**.
- 2 In the window that opens, click the drop down list and select



Characteristics.

Synchronization scope

By default, all objects in synchronized models are included in the synchronization. You can however exclude an object from the synchronization.

See [Step 1: Selecting the source objects to be synchronized](#).

Synchronization state

You can protect an object so that it will not be modified at synchronization by specifying that it is "Frozen".

See also [Protecting Objects](#).

Synchronization direction

The synchronization direction of a mapping indicates which object is updated related to the other.

In certain cases, synchronization is possible in both directions (for example, when two objects that can be synchronized do not yet have a mapping). In other cases, it is only possible in one direction (for example, if one of the two objects is already synchronized) or impossible in both (because each object already has a mapping or because the object types concerned cannot be subject to synchronization). This indication is given in the **Synchronization** box.

Summing up:

- **Bidirectional: synchronization in both directions.**
- **From left to right:** synchronization is from left to right (the object on the right is synchronized with the object on the left).
- **From right to left: synchronization is from right to left.**
- **Never : no mapping is possible between the two objects.**

For more details on mapping, see [The Database Editor](#).

REDUCED SYNCHRONIZATION (LOGICAL TO PHYSICAL MODE)

The synchronization function enables synchronization of a logical model and a physical model in the database. In design phase, it is often useful to synchronize part of the current model without having to consider the complete database which can be extremely large. **HOPEX Information Architecture** enables limitation of synchronization scope to a set of objects, thus reducing synchronization processing time.

The points that follow detail "Reduced synchronization" mode in direction Logical > Physical directions, but it is also available in Physical > Logical direction.

Reduced Synchronization Source Objects

Reduced synchronization is synchronization applied to an object other than to the database. Reduced synchronization applies only to an object of which the database has already been synchronized.

Objects on which you can run reduced synchronization are:

- Class
- Association
- Package
- Table
- Table file
- Entity (DM)
- Association (DM)
- Data model

Reduced synchronization scope is determined by the object on which you run reduced synchronization.

The following cases illustrate reduced synchronization in the logical to physical direction.

Running from a data model

When you run a synchronization on a data model, by default all objects of the model are selected in the scope of the synchronization; they are all selected in the editor.

Running from a data model entity

When you run a synchronization from an entity (or another object) belonging to a data model, only those objects linked to this entity within the same model are selected by default.

Objects linked to the entity but belonging to another model are displayed in the editor (when they are connected to the target database) but not selected by default. You must select the associated check boxes to take them into account in the synchronization.

The data model of the synchronized entity is taken as scope only if the ownership link between data model and entity is clearly identified: this is the case when you select the entity in the navigation window, but not when you select it in a diagram.

Running on an entity outside context

When you run synchronization on an entity outside context, for example in an explorer window, all objects depending on the modified entity, whether or not included in different models (on condition that these models are connected to the target database) are selected by default in synchronization scope, since no particular model context is identified.

Reduced Synchronization Strategies

At synchronization from an object, the three strategies below can be applied.

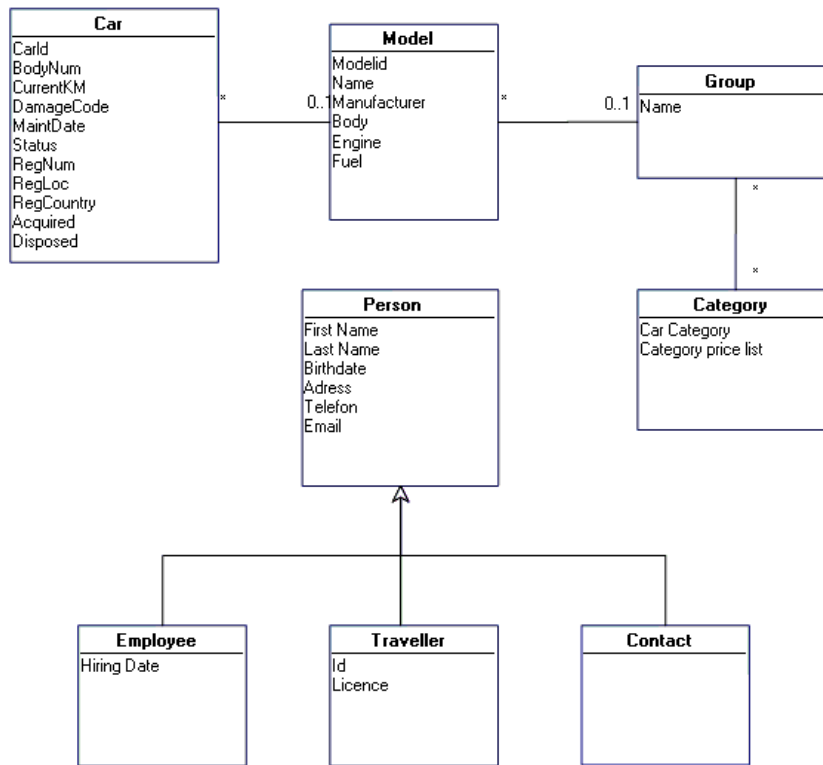
Impact of synchronized object on other objects

This strategy enables definition of synchronization scope from the source object, with the possibility of extending this to all objects dependent on the source object and likely to be affected by its modification.

Example

With this strategy in the model below, reduced synchronization of the "Model" entity allows inclusion of the "Car" entity, which has a constraint association to the "Model" entity.

Logical model



Reduced synchronization results

Indicate objects to dismiss:

	Name	ExpressionType		Scope	Name
	Car Rental				Car Rental
	Car Model				Car
	Car	Car Model::Car			Model
	Model	Car Model::Model			
	Car Rental	Car Rental			

Impact of other objects on synchronized object

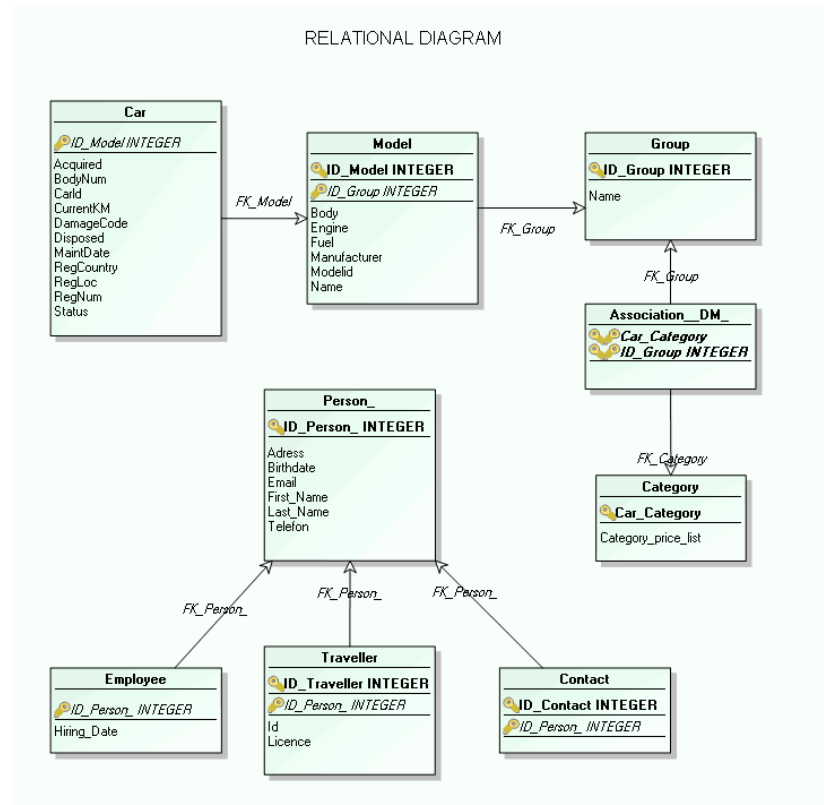
This strategy enables integration in reduced synchronization scope of objects on which the source object directly depends, for example all objects associated with the source entity which are necessary for update of the corresponding table.

Example

In the same example as before, taking the "Model" entity as source of reduced synchronization, the scope extends to the "Group" entity since tables corresponding to these two entities are linked by a foreign key: the "Group" entity can modify the "Model" table associated with the "Model" entity via the intermediary foreign key "FK_Group" (see diagram below).

The "Car" entity however is not taken into account in the scope since it cannot act on the "Model" table.

Physical model



Reduced synchronization results

Indicate objects to dismiss:

	Name	ExpressionType		Scope	Name
	Car Rental	Car Rental			Car Rental
	Car Model	Car Model			Group
	Group	Car Model::Group			Model
	Model	Car Model::Model			
	Car Rental	Car Rental			

All impacts

This strategy allows a combination of the two strategies described above. Scope of reduced synchronization is extended to objects required by the source object, and to all objects likely to be affected.

Example

Reduced synchronization results

Indicate objects to dismiss:

	Name	ExpressionType		Scope	Name
	Car Rental	Car Rental			Car Rental
	Car Model	Car Model			Car
	Car	Car Model::Car			Group
	Group	Car Model::Group			Model
	Model	Car Model::Model			
	Car Rental	Car Rental			

Running Reduced Synchronization

Before starting, check that option "Reduced synchronization (logical > physical)" is active:

- 1. On the desktop, click **Main Menu > Settings > Options**. The options window appears.
- 2. In the left pane of the window, expand the **HOPEX Solutions > Information Architecture** folder.
- 3. Click **Database Synchronization**.
- 4. In the right pane of the window, select **Activate reduced synchronization (logical > physical)**.

To start reduced synchronization:

1. Right-click the object to be synchronized.
2. Select **Synchronize**.
The synchronization wizard opens.
3. Select the synchronization type "logical to physical".
An entity can be used by several data models, therefore by several databases. When this is the case, you must select the database concerned.
4. Select the **Strategy**.
5. Click **Next**.
6. Select the objects to be synchronized
The scope selected by default depends on the context in which you select the object to be synchronized: if reduced synchronization is initialized from an entity in a diagram, the diagram model in question is selected. If the entity is selected outside its context, all models in which it appears are displayed in the editor.
 *Selected objects are not memorized and at a new synchronization default scope is again displayed.*
7. Click **Next**.
8. Define the **Synchronization Options**. All standard synchronization options are available with the exception of the "Reinitialize target objects" option.
9. Click **Next**.
The target objects protection option is displayed, you can view frozen objects. Protection of objects cannot be modified.
10. Validate results by clicking **Next**.
The synchronization report appears.
11. Click **OK** to close the synchronization wizard.
The mapping editor appears (unless the mapping option was cleared from the synchronization wizard).

Reduced synchronization options

Reduced synchronization presents the same options as total synchronization, with the exception of:

- Reinitialization of target objects
- Order

The reduced scope of reduced synchronization does not give a valid result for these two options

RUNNING SYNCHRONIZATION AFTER MODIFICATIONS

When a database has been synchronized and then manually modified, any additional specifications made directly in the database are retained, unless:

- Reinitialization is requested.
- Changes made in the data diagrams prevent this (addition or deletion of objects or links).

These changes include:

- Creation of entities, associations, attributes in the data diagram
- Deletion of entities, associations, attributes in the data diagram
- Modifying the characteristics of an attribute
- Modifying the name of an attribute, entity, or association
- Modifying the maximum multiplicity of an association
- Modifying the links of an association

Additional specifications made to the relational diagram may include:

- Deleting objects created by the synchronization
- Creating objects
- Modifying object characteristics created by the synchronization
- Modifying the order of columns in the tables, keys, or indexes

Synchronization after Modification of the Data Diagram

Newly created entities, associations, and attributes in the data diagram

The corresponding elements are created in the relational diagram, according to the rules used in the first synchronization.

Entities, associations, or attributes deleted from the data diagram

The corresponding elements are deleted in the database. For example, when an attribute is deleted in the data diagram, the corresponding column(s) are also deleted.

Modified attribute characteristics

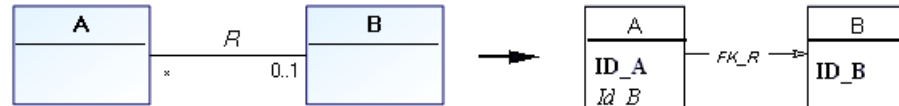
Modifications made to the characteristics of an attribute (type, length, decimal places, etc.) are reflected in the corresponding column in the relational diagram.

If the value of a characteristic of a column has been changed directly in the relational diagram, it will be preserved.

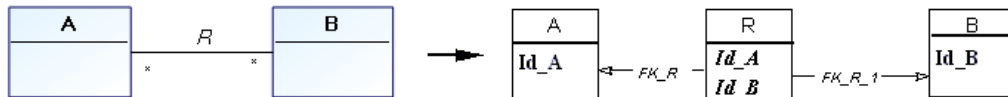
Modified name of an attribute, entity, or association

Modifications made to the name of an attribute, entity, or association are not reflected in the corresponding object in the relational diagram.

Modified maximum multiplicity of an association



If the maximum multiplicity of an association was 1, resulting in the creation of a migratory column, and has been changed to N, the migratory column is deleted and the table mapped by the maximum multiplicity of N is created.



Modified association links



Association R no longer concerns entity B, but does concern entity C.

In this case, the migratory column for B in A is no longer mapped.

- It is deleted.
- A migratory column from C is created.

Synchronization after Modifications to the Physical Diagram

Deleted table or column

If you delete objects from the database that were created by the synchronization (table, column, key, index...), these deletions are memorized and retained.

As long as the entity, association, or attribute that maps the table or column exists, the table or column is no longer recreated.

To recreate a column created by the synchronization and subsequently deleted:

1. Run database synchronization
2. At the results validation step, confirm the creation action (select the corresponding check box) proposed for this column.

Created objects

Objects (table, column, key, index) created in the relational diagram are retained.

However, deleting the objects they depend on may result in their deletion.

For example, a column is created in a table mapped by an entity. If the entity is then configured as “No table” or if the entity is disconnected from the datamodel, the corresponding table will disappear and the column with it.

☛ *Objects created in the relational diagram can be mapped manually. Objects created at synchronization are mapped automatically.*

Modified characteristics of objects created by synchronization

Modifications to the characteristics (SQL name, length, not null, datatypes) of objects created by synchronization are retained.

Modified order

Concerning modifications of order, processing depends on options defined in the synchronization wizard (see [Step 2: Synchronization options](#)).

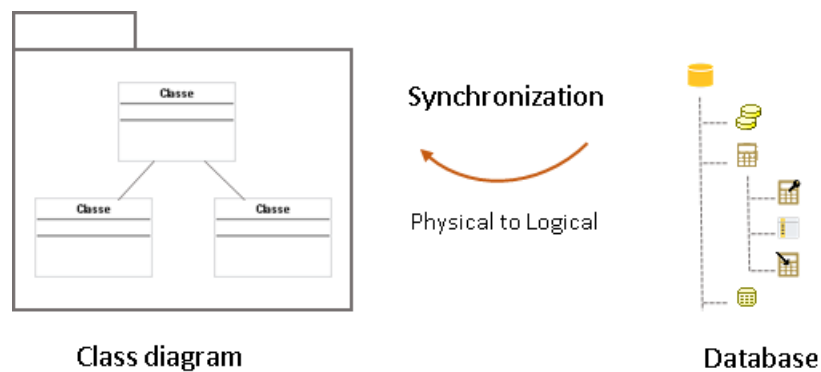
FROM THE PHYSICAL MODEL TO THE LOGICAL MODEL

This section describes how to synchronize the physical model model of a database with the corresponding conceptual model.

"Physical to Logical" Synchronization Rules

Synchronizing the logical model from the physical model enables creation of the database data diagram from its tables.

Rules used for this transformation are:



- A table of which the primary key is composed of a foreign key relating to the same columns becomes an entity. A generalization is created between this entity and the entity corresponding to the table to which the foreign key is pointed.

Physical level example:

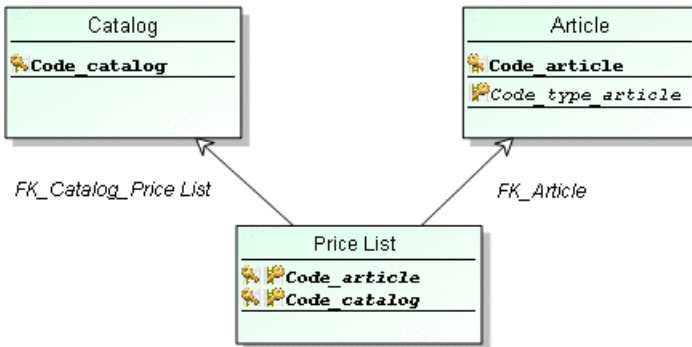


Result at the logical level:



- A table of which the primary key is composed of foreign keys only becomes an association of multiplicities (*..*). If columns do not belong to the primary key, an attribute connected to the association will be created for each of these columns.

Physical level example:



Result at the logical level:



- A table of which the primary key contains foreign keys and at least one column that is not a foreign key becomes an entity. An aggregated association is created between this entity and the entity corresponding to the table to which each of the foreign keys is pointed.

The candidate key of the entity is composed of the roles of aggregated associations and of the attribute(s) corresponding to the other columns of the primary key of the table.

Physical level example:



Logical level result:

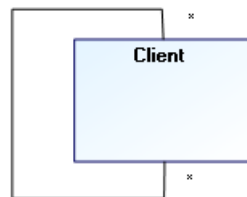


- A table of which the primary key is composed of foreign keys only pointing to the same table becomes a reflexive association of multiplicities (*..*).

Physical level example:



Logical level result:



- In other cases, each table becomes an entity and its columns the attributes of the entity.

- A foreign key becomes an association (0..1, *). If all the columns of the key are mandatory, its cardinalities become (1, *).

- Types of attributes are recalculated with the help of the conversion table specific to the target DBMS (see [Data Types and Column Datatypes](#)).

Running Synchronization

To start the synchronization:

1. Select the database concerned (in the list of databases or in a diagram for example).
2. Right-click the database and select **Synchronize**.
The synchronization wizard opens.
3. Select the synchronization type "physical to logical".

Step 1: Selecting objects to be synchronized

To define synchronization scope:

Scroll the list of objects contained in the database.

1. By default, all objects are selected and therefore included in synchronization. To exclude an object from the synchronization, clear it from the **Scope** column. When an object is excluded, its mapping is also excluded.
2. By default, all the objects are "realized", in other words, they give way to the creation of an object during synchronization. To specify that an object is "not realized", select it in the **Not realized** column. For more detailed information, see [Realized mode](#).
3. When the list of objects has been defined, click the **Next** in the wizard.

Step 2: Synchronization options

From the synchronization options, you can:

- Reinitialize target objects: synchronization starts from zero and deletes existing target objects.
- Recalculate target object names: names of data diagram objects are recalculated as a function of those of the relational diagram. This means that any manual modification of the data diagram is canceled.
- Take account of optimizations: all optimizations - including those not selected in the validation step (see [Validating optimizations](#)) are proposed.
- Take account of deletions: entities, associations and diagrams that have been deleted are included in the scope. Consequently, deletion of corresponding target objects and links is proposed.
See [Using Options](#).

You must also indicate the data model that will own the set of objects created by the synchronization.

Other options concern target object properties update. By default, synchronization updates all properties of each object concerned.

Scheduling

You can run synchronization:

- Immediately
- As soon as possible (after publication of updates)
- At a predefined date and time

】 When options have been specified, click **Next**.

Step 3: Protecting objects

Synchronization can impact all target objects.

- 】 To keep an object intact, select it in the **Frozen**.
- 】 Click **Next** to continue.

See [Protecting Objects](#).

Step 4: Validating results

The wizard displays the results that will produce synchronization validation.

- 】 To validate these results, click **Next**.

A report presents a list of processes that have been carried out.

Reduced synchronization

The above synchronization applies to a database but you can also run a Physical > Logical synchronization on a specific object of the database to reduce synchronization scope and processing time. See [Reduced Synchronization \(Logical to physical mode\)](#).

"Physical to Logical" Synchronization Results

Owner data model

A default data model owning the entities is created at the time of the Physical to Logical synchronization. Synchronized classes and associations are automatically connected to it. You can then distribute these entities and associations between the various data models of your study.

Data diagrams

A data diagram is created for each of the relational diagrams of the database. Classes and associations resulting from the tables of the corresponding relational diagram are connected to it.

Mappings

See At synchronization, the "Article Type" entity does not produce creation of a table. In the "Article" table, the foreign key to "ArticleType" is not created; however the "Code_Type_Article" column is created. See also Synchronization Results: Correspondences. See also Synchronization Results: Correspondences. At synchronization, the "Article Type" entity does not produce creation of a table. In the "Article" table, the foreign key to "ArticleType" is not created; however the "Code_Type_Article" column is created. Synchronization Results: Correspondences.

CONFIGURING SYNCHRONIZATION

This section describes the default options and parameters taken into account at synchronization.

Preparing Synchronization

To prepare synchronization:

1. Right-click the database and select **Properties**.
The properties window appears.
2. Click **Options > Standard**.
3. If you want the physical database name used at SQL script generation to be different from the database name, specify the target database name in the **SQL Name** text box.
4. If required, indicate a **Prefix** for the SQL Tables. This prefix will be added to the beginning of the name for each table generated.
*Additional parameters for configuration of synchronization and generation can be indicated in **Options**. These parameters vary as a function of the DBMS selected.*
5. Click again on the scroll-down list of the properties window and click **Characteristics**.
6. Select the **Target DBMS** and its version.
The type of the target DBMS determines:
 - For synchronization, the generation of column datatypes based on the type and length of attributes (see [Data Types and Column Datatypes](#)).
 - In generation, the syntax of the generated SQL commands.
7. Click **OK** to close the dialog box, saving the modifications.

Creation Options

On a database

It is possible to configure synchronization for each database in order to modify:

- Its creation options
- Processing of repository integrity (keys OnDelete or OnUpdate as a function of the possibilities offered by the DBMS target)

This configuration also concerns processing of the Not Null columns and the automatic creation of indexes on primary keys.

To configure the creation options for the database:

1. Right-click the database and select **Properties**.
The properties window appears.

- Click **Options > Synchronization**.
The corresponding options appear.

You can specify the following parameters:

Options - Synchronization

- Names of tables: 30,^DB^ROOT
- Names of columns: 30,^ROOT
- Names columns of FK: 30,^ROOT
- Names of PK: 30,PK_^ROOT
- Names of FK: 30,FK_^ROOT
- Names of PK index: 30,IDX_^ROOT
- Names of FK index: 30,IDX_^ROOT
- Index names: 30,IDX_^ROOT
- Names cols PK auto: 30,ID_^TBL
- Not Null Columns: Not Null
- OnUpdate:
- OnDelete:
- Not Null PkColumns: Not Null

- **Columns Not Null** activates/deactivates the WITH DEFAULT option for Not Null columns.
- **OnDelete**: key deletion default strategy. Possible values are:
 - **Restrict**: deletion is refused.
 - **Cascade**: deletion of a column is reflected in dependent tables.
 - **Set Null**: indicates "Null".
 - **Set Default**: gives the default value if this is specified. If no default value is specified, nothing occurs ("No action").
 - **No Action**: nothing occurs.
- **OnUpdate**: key update default strategy.
- **Names cols PK auto**: columns derived from the implicit identifier. See [General rule](#).

Parameters whose names begin with **Names of** indicate rules applied in name generation (see [Configuring Name Generation](#)).

On the DBMS

The default values for database synchronization and generation parameters are accessible in the properties dialog box of the DBMS used.

To display DBMS synchronization parameters:

1. In the edit area, click the **Main Menu** button then **Advanced Search**.
2. Select the object type "DBMS Version" and click on **Find**.
3. Right-click the target DBMS name and open its **Properties** dialog box.
4. Click **Options > Synchronization**.

By default, the parameters specified at the DBMS levels are valid for all new databases. When you modify synchronization parameters on a database, this database no longer takes account of DBMS parameters.

➡ For more information on synchronization configuration, see also [Running Synchronization](#).

Configuring Name Generation

Naming rules

The names of physical objects created at "logical to physical" synchronization are deduced from the **Local Name** of the logical objects from which they are derived.

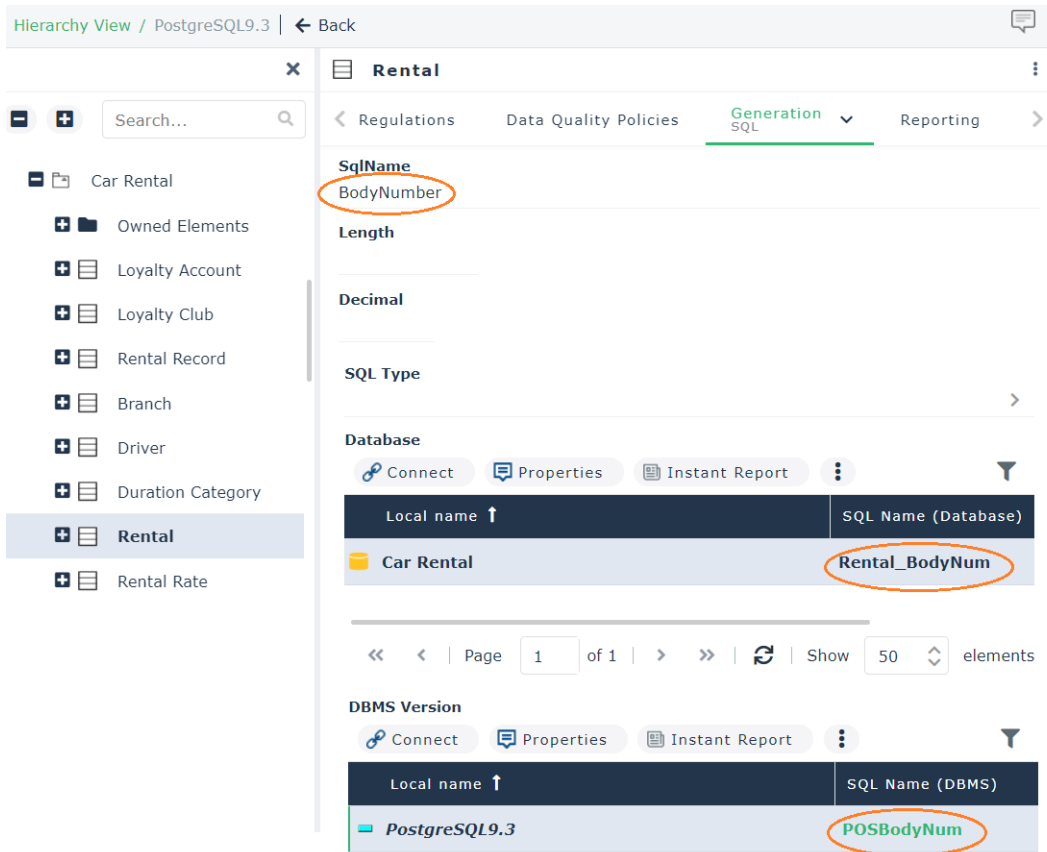
As logical object names (class, association, part, attribute, role) are not subject to any particular restrictions, transformation rules apply by default at their synchronization. These rules are accessible locally in the **Options > Standard** page of synchronized database properties, or globally in the target DBMS properties:

- **Identifier size**: maximum size of SQL identifier for this target DBMS
- **First character**: character set authorized for first character of SQL identifier
- **Authorized characters**: character set authorized for SQL identifier characters
- **Replacement character**: replacement character for unauthorized characters
- **Converted characters**: SQL identifier character set to be converted
- **Conversion characters**: character set corresponding to characters to be converted
- **Upper-case conversion**: conversion to upper-case of SQL identifiers

It is possible to indicate another name for each synchronized object using its **SQL Name**. The **SQL Name** replaces the **Local Name** at synchronization, while taking account of default transformation rules.

The **SQL Name** of logical objects is accessible in the **Generation > SQL** page (or **SQL** page) of their properties.

You can give a different name depending on the database and DBMS.



When fields **SQL Name (Database)** and **SQL Name (DBMS)** indicate different names, the name defined at database level takes precedence at synchronization.

By default, names of relational objects are generated according to the following masks:

Table	Database prefix + name* of entity or association
Column	name* of attribute
Primary key	"PK_" + name* of entity or association
Foreign key	"FK_" + name* of target entity or role if specified
Primary key index	"IDX_" + name* of entity or association
Foreign key index	"IDX_" + name* of target entity or role if specified

**Name calculated according to previously explained naming rules.*

These masks can be modified locally in each database, or globally for a given target DBMS.

Modifying a naming rule

To modify the mask of a naming rule:

1. Right-click the database and select **Properties**.
2. Click **Options > Synchronization**.
3. In the field of the rule in question, click the arrow.

Names of FK index

30,IDX_

Index names

30,IDX_^ROOT

Names cols PK auto

30.ID ^TBL

Modify

Reinitialize this parameter

Reinitialize all parameters

4. Click **Modify**.
The **Enter SQL Mask** window opens.

Entering the SQL mask

SQL masks define relational object naming rules at synchronization.

Example: In the DB_EMPLOYEES database, which has the prefix EMP, the mask ^DB_^ROOT generates the following for the table derived from the Customer entity: EMP_CUSTOMER

In the SQL mask entry dialog box, you can directly enter the **Mask** using syntax indicated below, but you can also use entry help proposed in the Component frame.

- 】 In the list in the **Component** box, select the elements that are to prefix the names. These elements are:

^ROOT	- For a table: name* of the class, association or part from which it is derived. - For a column: name* of the attribute or name of the identifier. - For a primary key: name* of the class, association or part from which it is derived. - For a foreign key: name* of the target class or the role if specified - For an FK column: name* of the attribute. - For an auto PK column: name* of the class identifier. - For an index on primary key: name* of the class, association or part. - For an index on foreign key: name* of the target class or the role if specified - For an index: name* of the attribute, role or class.
^DB	Database prefix
^EXT	- For a foreign key: name* of the association, part or generalization - For an index on foreign key: name* of the association, part or generalization
^TBL	Table local name or reference table local name
^TBO	- For a foreign key: name* of the class, association or part from which it is derived - For an index on foreign key: name* of the class, association or part from which it is derived
^TBR	Reference table name
^KEY	Foreign Key name
^CPT	Timestamp

**Name calculated according to previously explained naming rules.*

Size indicates total length limit of the generated name. It also applies to each of the elements used, which will be shortened to the number of characters indicated between brackets alongside the element concerned.

Definition of a Timestamp ("^CPT") enables automatic generation of an order number and indication of its length, (for example, ^CPT[^1^] will generate "1", "2", "3"; ^CPT[^3^] will generate "001", "002", "003").

The **Always** option indicates that timestamping begins from the first occurrence (CLI00, CLI01, etc., instead of CLI, CLI01).

You can also specify characters used as prefix and suffix of this timestamp.

Using the **Name Unicity** option ensures that the name of an object is not repeated in the database, repository or table in which this object appears. As such, if you apply the Name unicity option to a table, different objects of this table cannot have the same name.

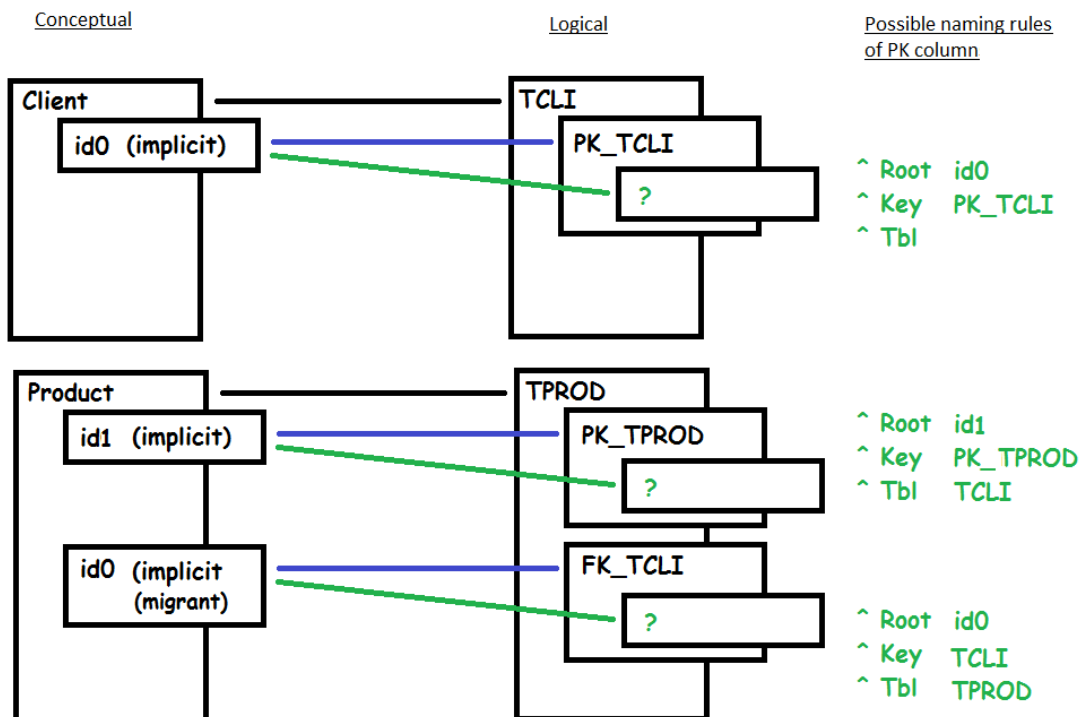
Configuring PK column names (implicit identifier)

At synchronization in Logical > Physical mode, the entity identifier becomes the primary key of the table. If the identifier is implicit, a column is automatically created. For more details, see ["Logical to Physical" Synchronization Rules](#).

By default, the name of a column derived from an implicit identifier is built using the ^TBL keyword which corresponds:

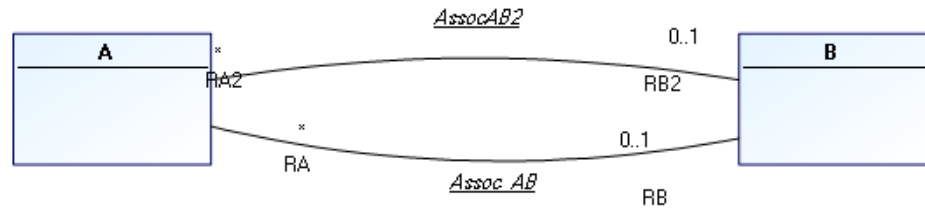
- to the name of the migrating table (in other words derived from a foreign key) if the identifier is migrating
- to the name of the table if the identifier is not migrating

You can modify construction rules and build the name of these columns with the ^KEY keyword corresponding to the name of the foreign key (without "FK_") if the identifier is migrating, as well as with the ^ROOT keyword corresponding to the name of the identifier (ID). See [Modifying a naming rule](#).

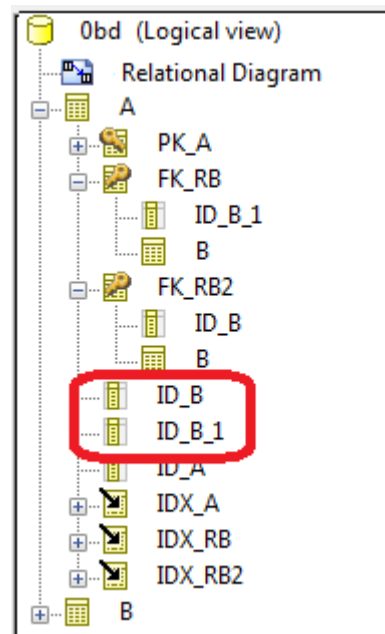


Example

When there are two constraint associations between two entities as below:



By default after synchronization, you obtain two columns with identical names, differentiated only by prefix "1".



You can modify the naming rule and build the name of these columns with the ^KEY keyword corresponding to the name of the foreign key (without " FK_ ").

The name of the foreign key being calculated on the name of the Role when it is specified, the names obtained for these two columns will be different.

In our example, if you replace "ID^TBL" par "ID^KEY" after synchronization you obtain:

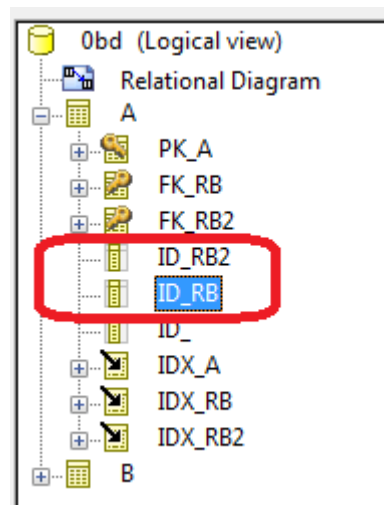


DIAGRAM SYNCHRONIZATION

Synchronization enables translation of a logical data model to a physical model, and vice versa. Before running synchronization, source diagrams (data diagrams and physical diagrams) must be saved and closed.

A first synchronization automatically creates the target model diagram. A new synchronization on previously synchronized models does not include automatic update of diagrams, in other words of graphical representation of models. Depending on the changes you have made, the synchronization wizard may propose update of the target diagram.

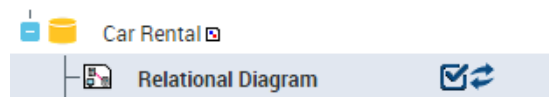
Case of Diagram Update at Synchronization

The synchronization wizard proposes diagram update in the following cases.

After source diagram modification

When you run synchronization after source diagram modification, by default the wizard activates target diagram update. Updating is indicated by display of two small blue arrows.

Below, at synchronization in logical to physical mode, modification of the logical diagram automatically produces a physical diagram update.



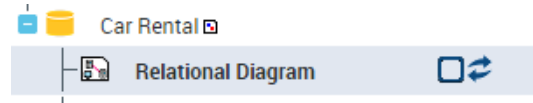
After target diagram modification

If you have modified the target diagram, for example the physical diagram, by default the synchronization does not propose update of this target diagram.

To display the case of target diagram update, you must select the synchronization option "Take account of optimizations". So that the update will be activated, you must select the check box in question.

After modification of both diagrams

If both diagrams - logical and physical - have been modified, target diagram update is proposed but not selected by default.



No modification detected

If neither of the two diagrams has been modified, the wizard does not propose update. It should be noted that any diagram modification must be saved before closing the diagram so that it will be taken into account by the synchronization wizard.

Particular case: an entity mapping with two tables

When an entity of the logical model is associated with two tables in the physical model (following merging of entities for example), the updated physical model displays only one of the tables.

MODEL MAPPING



In many modeling projects the problem arises of communication between teams of analysts and architects and the database development teams.

HOPEX Information Architecture offers two modeling levels:

- ✓ The logical level which describes data modeling in terms of entities and relationships and is intended for analysts and developers.
- ✓ The physical (or relational) level, which describes the database in terms of tables and interfaces with the DBMS. This level is intended for the designer and the database administrator.

By enabling change from one data model to another, the database editor favors consistency between the architecture of data and its support systems.

The following points are covered:

- ✓ [The Database Editor](#)
- ✓ [Mapping Details](#)

THE DATABASE EDITOR

The database editor allows you to synchronize the different views of a database manually; the logical model and the physical (or relational) model.

The synchronization wizard automatically maps the the two views. See [Synchronizing logical and physical models](#).

After synchronization, you can create or modify mappings in the editor manually, but this method no longer guarantees consistency of the two models. The denormalization wizard maintains this consistency. See [Denormalizing logical and physical models](#).

Run the editor on a database

To open the editor on a database:

- 1. Click the database icon and select **Mapping Editor**.
The mapping editor juxtaposes the logical view and the physical view of the database. When a mapping tree exists, it is automatically displayed. When a tree has not been created for the database, a window prompts you to create it.

Creating a Logical/Physical Mapping Tree

To create a mapping tree:

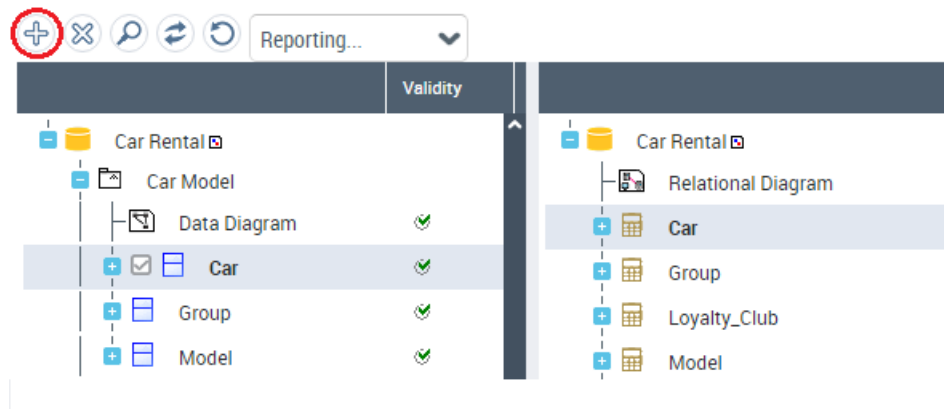
1. In the creation dialog box that opens, indicate the name of the new mapping tree.
2. In the **Nature** list box, select the nature of the tree: "Logical/Physical".
3. In the **Left Object** and **Right Object** frames, select the logical and physical models that you wish to align.
4. Click **OK**.
The editor displays the mapping tree juxtaposing the two models.

Creating a Mapping

To create a mapping between an entity and a table:

1. In the database editor, select the entity then the table.

2. Click **Create mapping item**.

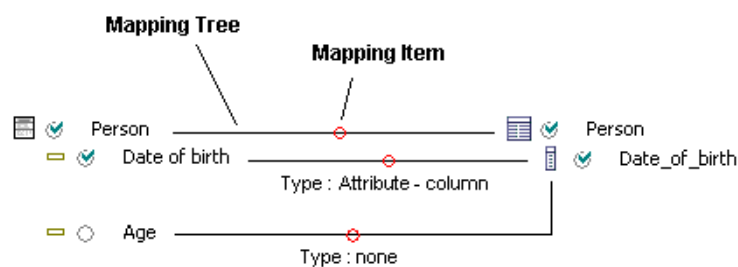


The mapping is created from the last object selected. Therefore, in order to create a mapping from the logical model to physical, in other words to define an object of the physical model from an object of the logical model, you must select the logical model object then the physical model object and create the mapping from the latter. If a mapping cannot be created, an error message appears (see [Synchronization direction](#)).

New mapping example

Consider the "Person" entity that contains the "Birth Date" attribute. In physical formalism it has as mapping the "Person" table which contains the "Birth_Date" column.

Suppose we add the "Age" attribute on the entity. This can be calculated from the birth date. So as not to create a column corresponding to this new attribute at synchronization, you can directly connect it to the "Birth_Date" column.



To create a mapping between the "Age" attribute and the "Birth_Date" column:

1. In the editor, on one side select the "Birth_Date" column, and on the other the "Age" attribute.
2. Click **Create mapping item**.

When the mapping has been created, a tick appears in front of the "Age" attribute.

Deleting a mapping

To delete a mapping on an object:

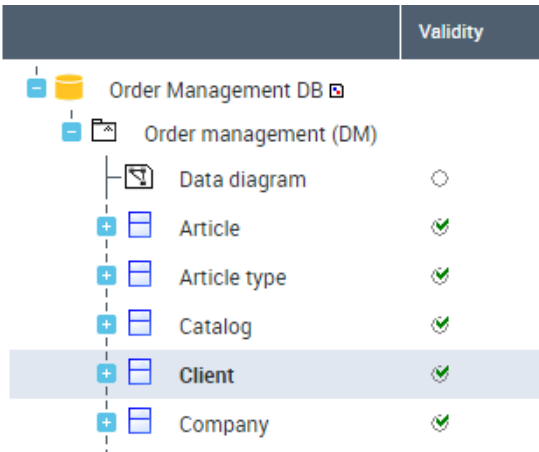
- 1 Select the object in question and click the **Delete mapping item** button.

MAPPING DETAILS

Objects with mappings are ticked green. When you select one of these objects, its mapping appears in the mapping properties dialog box located by default at the bottom of the database editor. It groups the names of objects connected in the two formalisms, the object types and comments where applicable.

Mapping example

The "Customer" table is selected in the logical view tree:



The mapping displays the following objects:

Validity	Logical object	Physical object	Type
Valid	Client	Client	Entity - Table

This means that the "Customer" table is derived from the entity of the same name.

Mapping Properties

To view mapping properties:

1. In the mapping editor, select the mapping item and click **Properties**.
2. In the window that opens, click the drop down list and select **Characteristics**.

See also [Mapping characteristics](#).

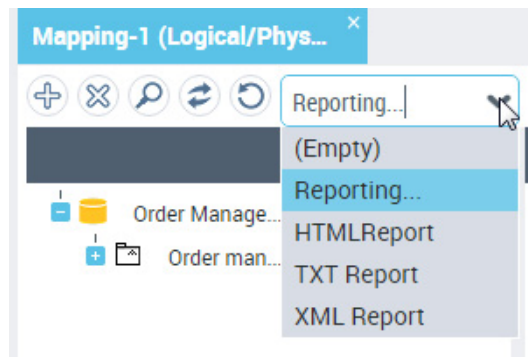
Mapping Report

In a document you can generate detail of mappings between the two database models. This can be an HTML, text or XML file.

Generating an HTML report

To generate the HTML report of a mapping tree:

- 1 In the database editor toolbar, click the drop-down list and select **HTML Report**.



The corresponding file opens. It presents detail of mappings of the physical view and the logical view in the form of a table.

Each row of the table presents an object of the model and its equivalent in the other model. In the middle of each row appears the status of the mapping between the two objects.

Description of Mapping View "Logical view"		
Object		Mapping
Order Management DB		
Relational Diagram	✓	Data diagram
Article	✓	Article
FK_Article_type	✓	ID_Article type3
	✓	Type
Article_type_code	✓	Article type code
Article_type_code_1	✓	Article type code

At the end of the document, you will also find a list of invalid mappings.

Object status

Indicators enable indication of status of synchronized objects. A filter bar allows you to show all or only certain of these indicators. This bar is available by selecting **View** > **Toolbar** in the editor.

Object status can be characterized as:

-  Valid
-  Invalid (when an object has kept a mapping to an object that no longer exists)
-  No mapping
-  Frozen (Protected)
-  Standard

Saving display of editor indicators

An option allows you to save the status of indicators in the mapping editor. It is specific to the user and the current mapping tree. A user who has selected the indicators display option will automatically find the status of objects in the previously created mapping tree.

The indicators display option of the editor is cleared by default. To activate it:

1. On the desktop, click **Main Menu** > **Settings** > **Options**.
2. In the left pane of the options window, select **Mapping Editor**.
3. In the right pane, select option **Save display of editor indicators**.
4. Click **OK**.

Mapping Source

When you select an object in the editor in one of the formalisms, you can display its mapping in the other formalism.

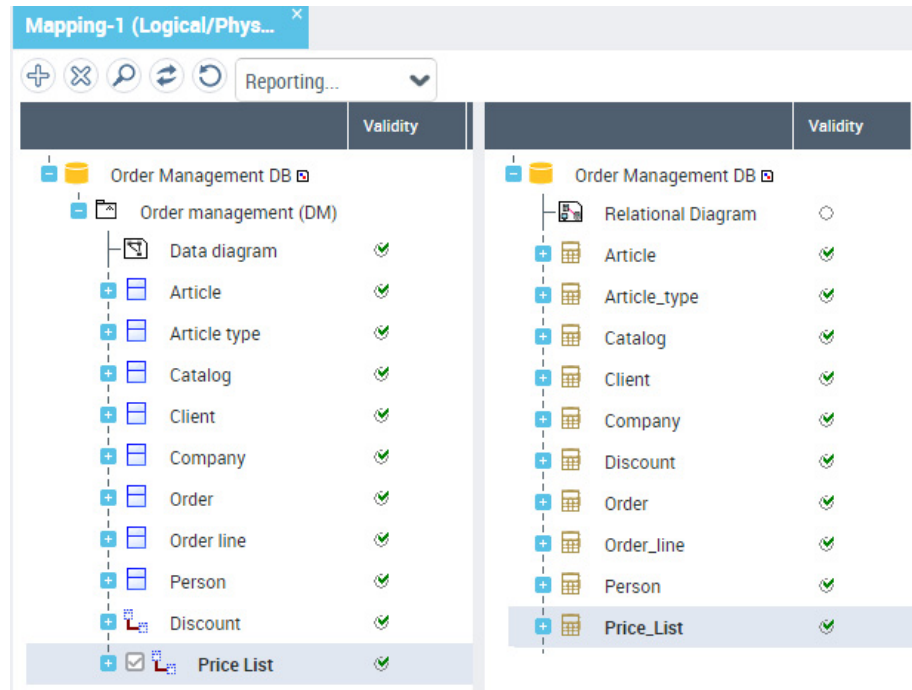
To display an object mapping:

1. In the mapping editor, select the object concerned and click **Find**.

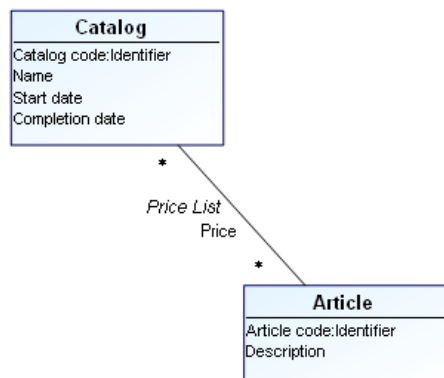
The editor displays the source object.

Mapping example

Consider the "Price List" table created in the logical model. In the pop-up menu of the table, select **Locate**. You will see that it corresponds to an association.



Opening the logical model diagram, you can see that this is the "Price List" association that connects the "Catalog" and "Article" entities.



At synchronization, this non-constraint association produces a table in which:

- A column is created for each connected entity.
- The primary key for the table uses all these columns.
- A foreign key is also built for each connected entity.

For more information on synchronization of associations, see [Logical to Physical Synchronization: the Associations](#).

Mapping Drawing

To view the mapping drawing:

- 1 In the mapping tree, select the objects concerned.

A window appears at the bottom of the editor showing a drawing of objects and their connecting links.



DENORMALIZING LOGICAL AND PHYSICAL MODELS



After having synchronized a data model and a physical model, you can develop them by modification in their diagrams. This method does not however guarantee consistency between the two models.

Denormalization wizards have been created to ensure such consistency. They allow you to modify definition of one model while maintaining consistency with the other.

The wizards also allow you to quickly carry out duplication or merging operations without mapping transfer and/or without source object deletion when you wish to transfer modifications in the relational model.

 **In HOPEX Information Architecture V2R1, the denormalization functionality is only available for data models, it does not apply to packages (UML notation).**

The following points are covered here:

- ✓ [Denormalization Principles](#)
- ✓ [Logical Denormalization](#)
- ✓ [Physical Denormalization](#)

DENORMALIZATION PRINCIPLES

Denormalization enables transformation or detailing of models as a function of specific requirements: choice of modeling, performance optimization, physical implementation level, redundancies, etc.

The denormalization tool is presented in the form of wizards enabling execution of these transformations.

A wizard applies to an object or group of objects. Depending on optimization type requested, denormalization creates new objects in a model starting from initial objects (source objects).

Denormalization: consistency of models

Denormalization changes the object of a model. When this model has been mapped with another (see [Synchronizing logical and physical models](#)), you can manage impact of this change on the other model.

At denormalization, you can therefore:

- Transfer or not transfer mappings with synchronized objects.
- Delete or keep source objects.

Transferring mappings

Transfer of mappings guarantees stability and consistency between two models. When denormalization creates a new object at the logical level, mapping with the physical object is transferred to the new logical object. When you clear this option, mappings with new objects are not created and the two models must therefore be resynchronized.



So that synchronization can validate changes resulting from denormalization, make sure the "Reinitialize target objects" option is cleared.

Deleting source objects

Source objects are deleted by default. Non-deletion of source objects allows you to keep initial objects after denormalization.

Synchronization and Denormalization

In data modeling, synchronization and denormalization are often combined to respond to particular use cases.

Example

A PAYMENT entity that you wish to represent in the database by three tables TRANSFER, CHECK and OTHER. To produce this modeling:

1. Create the PAYMENT entity.
2. Run synchronization (logical to physical) to obtain the PAYMENT table.
3. Run the physical wizard to horizontally partition the PAYMENT table.
4. Rename the three duplicates TRANSFER, CHECK and OTHER.

The three tables obtained in this way are now connected to the PAYMENT entity and will follow the developments of future synchronizations.

Combining denormalization and synchronization options

Impact of a denormalization on two synchronized models varies depending on the options selected.

Consider the example of a logical denormalization. Possible combinations are:

- Deletion of source objects + transfer of mappings: logical model source objects are deleted. The physical level is unchanged; mapping with logical objects resulting from denormalization is assured.
- Deletion of source objects + non-transfer of mappings: physical objects corresponding to logical objects resulting from denormalization are created. The physical objects corresponding to deleted logical objects are deleted.
- Non-deletion of source objects + transfer of mappings: logical model source objects are kept. The physical level is unchanged; mapping with objects resulting from denormalization is assured.
- Non-deletion of source objects + non-transfer of mappings: physical objects corresponding to objects resulting from denormalization are created. Existing physical objects are kept.

**Particular cases: ascending and descending merges:**

In the case of an ascending merge, the supertype entity plays a particular role. Denormalization keeps its mapping in all cases. Similarly, in the case of a descending merge, subtype entities play a particular role; their mapping is kept in all cases.

Denormalization: Use Case

Combination of denormalization options varies depending on design mode of your models.

1. Maintaining stability at the physical level when a modification is applied at the logical level.

Context: a synchronization has already been established between the logical and physical levels. The physical level is in production. A modification must be applied at the logical level, without impact on the physical level.

Recommended denormalization options: transfer of mappings, deletion of source objects.

This use case corresponds to a preventive maintenance-oriented work mode: modifications are carried out by anticipation on the logical level, knowing that the physical level must not be modified until further notice.

Result: after denormalization, mappings are re-established between target logical objects and physical objects. After synchronization, nothing changes at the physical level.

2. Developing the physical level when denormalization is applied at the logical level.

Context: a synchronization has already been established between the logical and physical levels. The physical level is not frozen and must develop as a function of the logical level.

Recommended denormalization options: non-transfer of mappings, deletion of source objects.

This use case favors developments at conceptual level, ignoring impact at the physical level.

Result: after denormalization, target logical objects are without mappings and physical objects corresponding to source logical objects are no longer synchronized. After synchronization, the physical level is updated: the physical objects corresponding to source logical objects (objects existing before denormalization are deleted; new physical objects corresponding to target logical objects (objects created by denormalization) are created.

3. Simplifying logical level development during the development phase.

Context: a synchronization has already been established between the logical and physical levels, or the new physical level has not yet been implemented.

Recommended denormalization options: non-transfer of mappings, non-deletion of source objects.

This use case corresponds to an "incremental" work mode: logical level source objects are unchanged. The model is supplemented by target objects resulting from denormalization. These target objects produce a new section at the physical level and the existing physical section remains stable.

Result: after denormalization, mappings are unchanged. After synchronization, new physical objects corresponding to new logical objects are created; physical objects corresponding to source logical objects are unchanged.

4. Favoring installation of multiple scenarios in development phase.

Context: a synchronization has already been established between the logical and physical levels, several modeling options temporarily coexist for a single physical level.

Recommended denormalization options: transfer of mappings, non-deletion of source objects.

This use case, to be used with care, enables keeping two modeling options at conceptual level that produce a common result at the physical level.

Result: after denormalization, physical objects remain connected to source logical objects and are also connected to target logical objects. After synchronization, objects at the physical level are unchanged.

LOGICAL DENORMALIZATION

Logical denormalization applies to data model entities (or classes) and attributes.

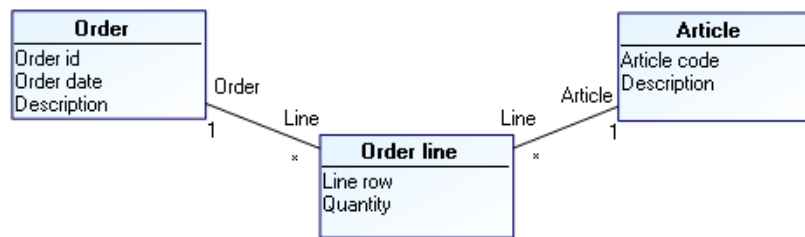
Running Logical Denormalization

To denormalize logical formalism:

1. Right-click the database with which the data model is associated and select **Denormalize**.
A wizard opens.
2. Select the type of denormalization concerned (logical) and follow the instructions of the wizard. See [Logical Denormalization Wizards](#).

Logical denormalization example

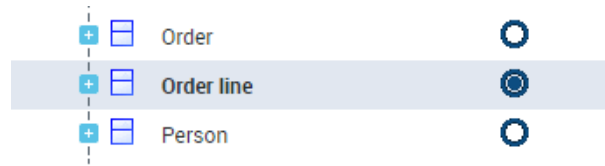
Suppose that you wish to transform the "Order line" entity to an association.



To transform this entity to an association:

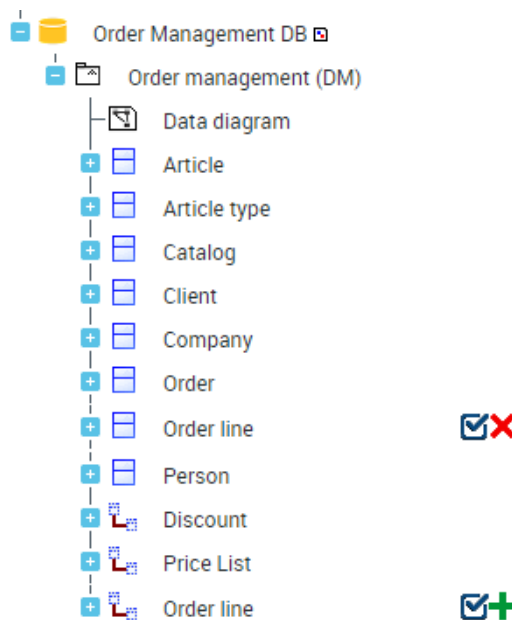
1. Right-click the "Order management" database which contains this entity and select **Denormalize**.
A wizard opens.
2. Select the **Logical** denormalization.
3. Click **Next**.
4. In the **Select the denormalization type field**, select "Transform an entity to an association".
5. Click **Next**.

6. In the editor tree, select the **Scope** column opposite the "Order line" entity check box.



You can select several entities at denormalization.

7. Click **Next**.
Denormalization options appear. Mapping transfer and source object deletion are activated by default. This means that the "Order line" entity will be deleted and the mapping link with the "Order line" table will be transferred to the association which replaces it.
8. Click **Next**.
The editor displays changes produced by this denormalization. You can see that the "Order line" entity will be deleted and the "Order line" association will be created.



When a selected entity cannot be transformed, the editor will indicate the reason.

You can refuse a modification by clearing the corresponding box.

9. Validate results by clicking **Next**.
This transformation is definitive and will be taken into account by the next synchronization.

On completion of denormalization, you can see that the "Order line" association that replaces the entity is now mapped with the "Order line" table.

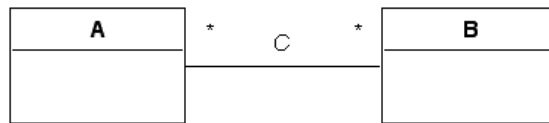
☞ If an object is protected, it is not possible to select it during denormalization.

Logical Denormalization Wizards

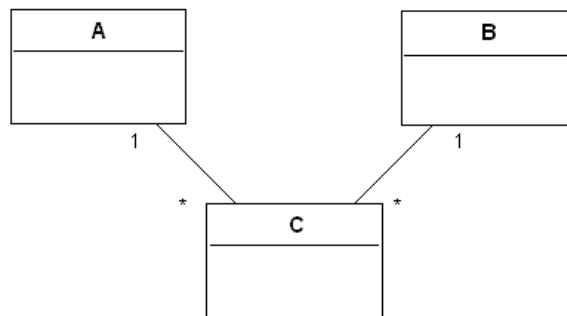
Transform association to entity

This denormalization enables transformation of an n-ary association, whatever its multiplicities, to an entity. An association of multiplicity '*',1' is created between this entity and the entities of the association.

Before



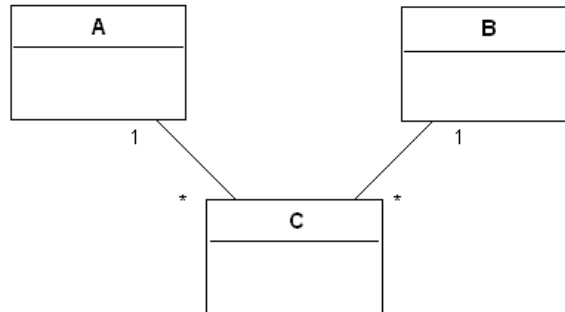
After



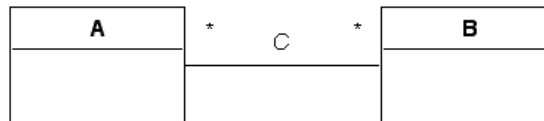
Transform entity to association

This denormalization enables transformation of an entity with n binary associations, of which opposite roles are of multiplicity '1', to an association. A new n-ary association of multiplicity '*' is created between these entities.

Before



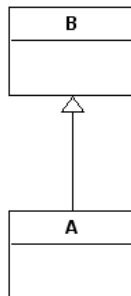
After



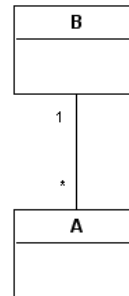
Transform generalization to association

This denormalization enables transformation of a generalization between two entities to an association. An association of multiplicity *,1 is created between the two entities and the generalization is deleted.

Before



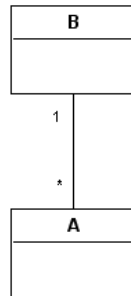
After



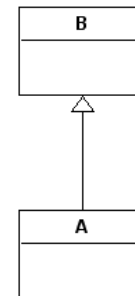
Transform association to generalization

This denormalization enables transformation of an association 1,* to a generalization. A generalization is created between the two entities and the association is deleted.

Before



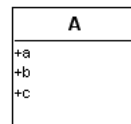
After



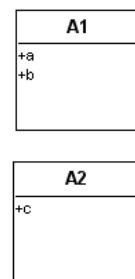
Vertical partition of an entity

This denormalization enables division of an entity into several entities. The attributes, associations and generalizations are shared between the entities.

Before



After



Horizontal partition of an entity

This denormalization enables duplication of an entity.

Before

A
+a
+b
+c

After

A1
+a
+b
+c

A2
+a
+b
+c

Horizontal partition and synchronization in Logical > Physical mode

Consider a "Catalog" entity. After horizontal partition, this entity gives two entities "Catalog-1" and "Catalog-2".

After synchronization of the Logical > Physical mode, the two entities have a table as mapping.

If you display properties of mappings, you will note that both are bidirectional, meaning that entities and table are updated in both directions of synchronization.

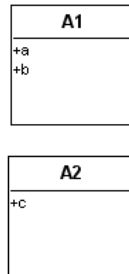
If you carry out logical modifications on these entities and then re-run synchronization, the editor displays a signal on the target table; it does not know which entity to take to carry out updates.

When you select an entity, this is kept as reference entity (for example Catalog-1). The other entity will be kept in one direction only, in other words Catalog-2 could be updated in the logical model but will have no impact on the table.

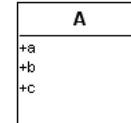
Merging of entities

This denormalization enables merging of entities.

Before



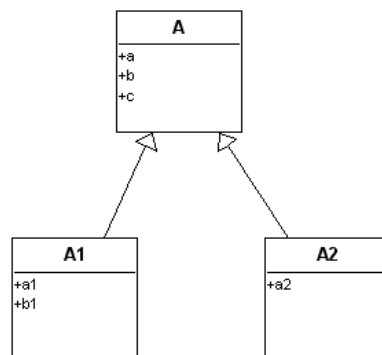
After



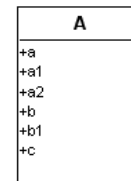
Merging of ascending entities

This denormalization enables merging of an entity with its parent entity: all attributes and links are transferred to the parent entity and the child entity is deleted.

Before



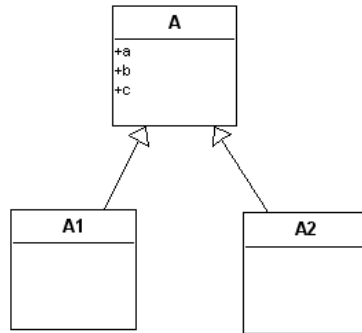
After



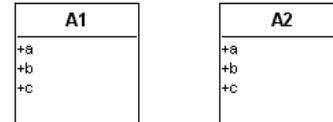
Merging of descending entities

This denormalization enables merging of an entity with its child entity: all attributes and links are transferred to the child entity and the parent entity is deleted.

Before



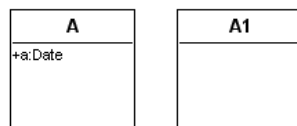
After



Copy/paste of attributes

This denormalization enables transfer of attributes of an entity or association to other entities or associations.

Before



After



PHYSICAL DENORMALIZATION

Physical denormalization applies to database objects represented by tables, columns, keys and indexes.

Running Physical Denormalization

To denormalize the physical formalism:

1. Right-click the database and select **Denormalize**.

A wizard presents all customizations possible on database objects.

Physical denormalization example

It is possible to arrange that a table is partitioned into two separate tables, either to separate columns of the two tables (vertical partition), or to duplicate information in a table in two others (horizontal partition).

Consider the example of an order entry. This order entry is represented by an entity at the logical level, and gives a table at the physical level. Suppose that order entry has to be managed differently at technical level, depending on whether it is a telephone or Internet order. This change can be integrated in the physical model without having to modify the logical model in parallel. To do this, we arrange that the entity representing order entry is partitioned into two tables; one for telephone orders, the other for Internet orders. By integrating this modification from the wizard, the partition becomes automatic at each synchronization, and the two models remain consistent.

To create a horizontal partition such as that described above:

1. Right-click the "Order management" database that contains this entity and select **Denormalize**.
A wizard opens.
2. Select the **Physical** denormalization.
3. Click **Next**.
4. In the **Select the denormalization type field**, select "Horizontal partition of table".
5. Click **Next**.
6. In the editor tree, select the table you wish to duplicate, in this case "Order".
7. Click **Next**.

Denormalization options appear. Mapping transfer and source object deletion are activated by default. This means that the "Order" table will be deleted and the link with the "Order" entity will be transferred to the two tables.

Specify the number of partitions, in other words the number of tables created. The tool creates two tables by default.

8. Click **Next**.
- The editor displays changes produced by this denormalization. You can see that the "Order" table will be deleted and that two new tables will be created.

	Scope	Name
 Order Management DB 		HBC Group::Sales::...
 Relational Diagram		Order Management...
 Article		HBC Group::Sales::...
 Article_type		HBC Group::Sales::...
 Catalog		HBC Group::Sales::...
 Client		HBC Group::Sales::...
 Company		HBC Group::Sales::...
 Discount		HBC Group::Sales::...
 Order	 	HBC Group::Sales::...
 Order_1	 	HBC Group::Sales::...
 Order_2	 	HBC Group::Sales::...
 Order_line		HBC Group::Sales::...

9. Validate results by clicking **Next**.
- This transformation is definitive and will be taken into account by the next synchronization.
- On completion of denormalization, you can see that the two new tables are now mapped with the "Order" entity.

 **Denormalization here applies to the physical model. The synchronization that will take this customization into account must therefore be run in the same direction, in other words from logical model to the physical. It is not valid in the other direction.**

List of Physical Denormalization Wizards

Vertical partition of a table

This denormalization enables division of a table into several tables. Columns are shared between the tables obtained.

Only columns that are not part of a key can be distributed between tables.

Before

A
a
b
c

After

A1
a
b

A2
c

Horizontal partition of a table

This denormalization enables duplication of a table. The two tables obtained contain all columns of the original table.

Before

A
a
b
c

After

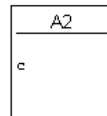
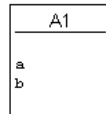
A1
a
b
c

A2
a
b
c

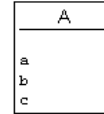
Merging of tables

This denormalization enables merging of tables.

Before



After

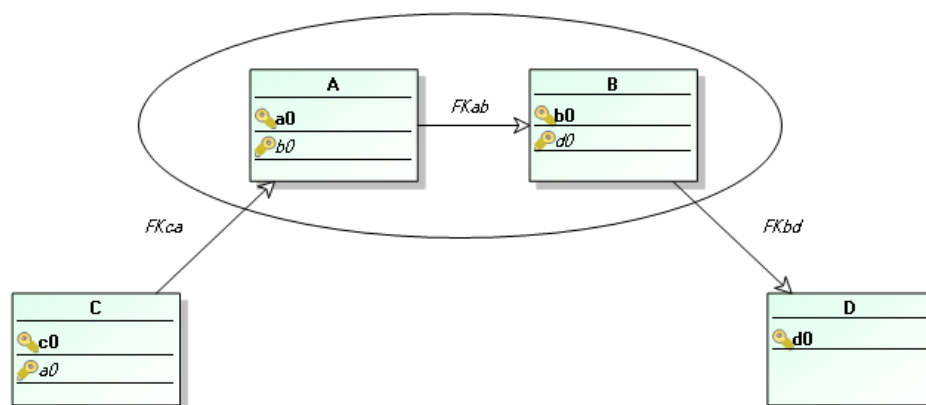


Primary keys option

When you run merging of tables, an option allows you to determine the primary key of the merge table: you can select one of the primary keys of the source tables or merge all primary keys of the source tables.

When you select a primary key for the merge table, only those foreign keys that reference this primary key are transferred. Foreign keys that reference primary keys that are not transferred are not taken into account.

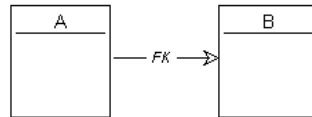
Therefore in the following example, if you merge tables A and B and keep the primary key of table A, the primary key of B disappears at merge. Nor is foreign key FK_{ab} transferred since it references primary key B. The other foreign keys, FK_{ca} and FK_{bd}, are transferred in the relational diagram.



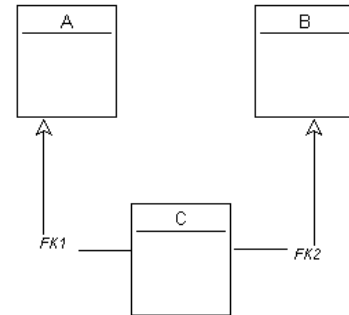
Transform foreign key to table

This denormalization enables transformation of a foreign key to a table. A new table and two new foreign keys are created. The original foreign key is deleted.

Before



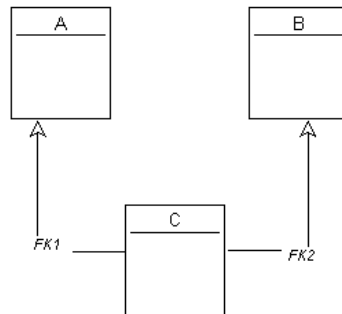
After



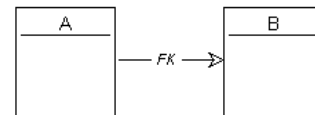
Transform table to foreign key

This denormalization enables transformation of a table to a foreign key. The table and its two foreign keys are deleted and a new foreign key is created.

Before



After



Copy/paste of columns

This denormalization enables transfer of columns of one table to another.

Before

A
a DATE

A1

After

A
a DATE

A1
a DATE



GENERATING SQL SCRIPTS



The SQL generation function produces SQL script files, which, from logical objects of your **HOPEX** repository (database, table, column, etc.) allow you to create, modify or update the corresponding objects in the target DBMS of your choice.

Generation takes into account parameters inherited from the target DBMS (specified for the database), parameters that you can customize at a global level (see [Configuring Database Generation](#)) or at a more detailed level, on a column or primary key for example.

For the main target DBMSs on the market, the database editor makes accessible a "physical view" that allows you to optimize the SQL grammar of generated scripts in order to integrate technical options specific to the selected DBMS, such as partitioning. See [Adding Physical Properties to Database Objects](#).

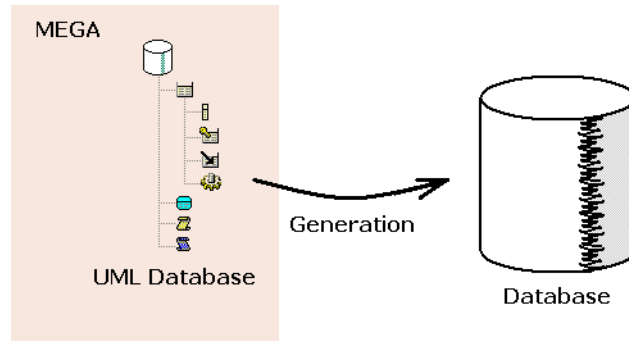
Finally, logical level can be completed by generation of physical objects specific to each database for a DBMS, such as logical views, stored procedures and triggers.

The different generation modes presented below take into account the constraints linked to database administration under different systems with maximum flexibility.

The points covered here are:

- ✓ [Running SQL Generation](#)
- ✓ [Incremental Generation](#)
- ✓ [Configuring SQL generation](#)
- ✓ [Supported Syntax](#)
- ✓ [Defining Database Views](#)
- ✓ [Defining Triggers for a Database](#)
- ✓ [Using Stored Procedures](#)
- ✓ [Adding Physical Properties to Database Objects](#)

RUNNING SQL GENERATION



SQL Generation Objects

Objects taken into account in generation are:

- Table
- Column
- Primary key
- Foreign key
- Indexes
- Data group
- Logical view
- Material view
- Trigger
- Stored procedure
- Job Title
- Synonym
- Sequence
- Cluster
- Partition

Scripts generated by **HOPEX** manage only the structure of relational objects, their content is not covered.

Start the generation wizard

Display of certain generation targets can be filtered. Before starting generation, check that the selected generation target is activated:

1. On the desktop, click **Main Menu > Settings > Options**.

2. In the left pane of the options window, expand the **HOPEX Solutions > Information Architecture** folders.
3. Click the **SQL Generation** folder.
This folder contains all supported SQL generators.
4. In the right part of the window pane select those that you want to display in **HOPEX Information Architecture**

For generation targets, see [Supported DBMS versions](#).

To start an SQL generation:

1. On the desktop, click the navigation menu, then **Data Architecture > Tools**.
2. In the edit area, click **SQL Code Generation**.
A wizard opens.
3. Define the **Generation scope**:
 - database for a complete generation
 - another SQL object for a partial generation.
4. For complete generation, select the database concerned and click **Next**.
5. Select the **Generation Mode**.
Four generation modes are possible:
 - "Creation": generates creation orders for all objects.
 - "Deletion": generates object deletion orders only.
 - "Replacement": starts deletion of objects, then recreation (to avoid creation of duplicates for example). Supposes that the target DBMS supports this generation mode.
 - "Modification": modifications only are taken into account. Unlike the other modes that act without taking account of what possibly exists, this mode enables connection to the DBMS server to obtain read-only access to the database already created. The wizard compares the **HOPEX** data file and the database information. After analysis of the two structures, the wizard generates the corresponding modification SQL orders. See [Incremental Generation](#).

 This mode is only available for the main DBMSs. See [Supported DBMS versions](#).
6. Click **Next**.
A dialog box then presents the objects generated.
7. Click **Next**.
Generation starts. A dialog box presents progress of the operation indicating the file(s) containing the result.
Click **Open** to see the list of the files generated.

When the extension used for the generated files is recognized by Windows, you need only double-click the file name to view it using the editor associated with the extension.

INCREMENTAL GENERATION

When a database has already been generated, you can subsequently reflect only changes to the database using "Modification" mode of SQL generation.

For a database, incremental generation allows you to:

- consult in an HTML report the differences between the database and its representation in **HOPEX**.
- produce SQL scripts enabling update of the target database from its description in **HOPEX**.

Incremental Generation Objects

Objects managed by incremental generation are the same as those of generation in "Creation" mode: table, column, primary key, foreign key, index, data group, logical view, material view, trigger, stored procedure, function, synonym, cluster, partition.

Scripts generated by **HOPEX** manage only the structure of relational objects, their content is not covered. Incremental generation options enable isolation of SQL orders that require particular precautions or additional processing.

Running Incremental Generation

Generation options

Incremental generation is done in a global file; it is carried out from the database and not from a particular modified object.

Before starting generation:

1. Right-click the database and select **Properties**.
The properties window of the database appears.
2. Click the drop-down list then **Options > Generation**.
3. In **Script Distribution**, select "A global file".
4. Click **OK**.

In options, you must also indicate incremental generation mode, which authorizes object deletion or not.

For more details on generation options, see [Configuring Database Generation](#).

Start the generation wizard

To run incremental generation:

1. Right-click the database and select **Generate the code**.

2. In the **Generation mode** field, select "Modification".
 - ☞ This "Modification" command is only available for the main DBMSs. See [Supported DBMS versions](#).
3. Select the **Data source**. Incremental generation can be carried out:
 - From an ODBC connection.
 - From an extraction file See [Extracting Database Schema Description from Data Sources](#).
4. Click **Next**.
5. When connected to the target DBMS, enter the name of the owner. This will enable you to filter the tables to be taken into account in the generation.
6. Click **Next**.

The result window presents two files, the SQL file and a "Report.htm" file. The latter file is a report of the generation. It is a dynamic file that presents initial content of the database and modifications carried out.

Origin (MEGA)		Target (Oracle 10)	Updating ac
TBSTABLE		TBSTABLE	
TBSPARTITION		TBSPARTITION	
TBSOVERFLOW		TBSOVERFLOW	
DEPARTEMENT_TEST5	⚠	DEPARTEMENT_TEST5	Insight
DEPARTEMENT_TEST4		DEPARTEMENT_TEST4	
DEPARTEMENT_TEST1	⚠	DEPARTEMENT_TEST1	Insight
DEPTNO		DEPTNO	
DOM		DOM	
NAME		NAME	
PAYS		PAYS	
REGION		REGION	
PARAMETERS		PARAMETRES	

Each row of the list describes:

- The **HOPEX** object. This can be empty if it has been deleted from the **HOPEX** repository.
- The DBMS update reference as compared to **HOPEX**. The various actions possible on objects are:
 - Creation ✖
 - Modification ↻
 - Deletion ✖
 - Replacement 📄
- A warning symbol ⚠ when the update of a DBMS object is not complete, or when this must be handled with care. When this icon is present, a block in the generated script details what cannot be updated.
- Object on DBMS side. This can be empty in the context of creation on the **HOPEX** side.
- The link to the object update script.

From each object you can access all its sub-objects by expanding the corresponding tree. You can also view the physical parameters. In a context of object modification, only the modified physical parameters are displayed.

There is also a report of the generation (.txt) in the properties dialog box of the generated database.

CONFIGURING SQL GENERATION

Configuring the DBMS Version

Supported DBMS versions

This list is given as a guide only. It is not intended to be an exhaustive list, and may change as new DBMS versions are released.

Product	Editor	Supported Versions
SQL ANSI	ISO 9075	1992
DB2	IBM	OS 390 V5 / OS 390 V7 / OS 390 V8 UDB V5 / UDB V7 / UDB V8
Ingres II	Computer Associates	2.0
Dynamic Server	Informix	7.3
Oracle	Oracle	8 / 9i / 10 / 11
SQL Server	Microsoft	7 / 2000 / 2005 / 2008
Adaptative Server	Sybase	11 / 12.5
Teradata Database	Teradata	14
PostgreSQL	PostgreSQL Global Development Group	9.3

Modifying DBMS version properties

Generation configuration is carried out to check the size of the identifiers generated, the characters authorized, etc.

To configure generation options of a DBMS version:

1. Search for the DBMS in question using the search tool.
2. Open its properties.
3. In the properties window of the DBMS, click the drop down list and select **Options > Generation**.
4. Modify the parameters you wish to change.

☛ The parameters available vary as a function of the target DBMS. If the **Generation** subtab does not appear, select **Tools > Options** in the

HOPEX bar, expand "Data Modeling SQL Generation" and check that the generation option linked to the DBMS concerned is selected.

Configuring Database Generation

To configure generation of a database:

1. Open the database properties dialog box.
2. Click the drop-down list then **Options > Generation**.

As for configuration of a target, the parameters proposed vary as a function of the DBMS and an explanatory message indicates the use of each parameter.

You can specify the following parameters:

- The **Trigger Name** parameters define the names of three types of trigger.
- **Error Ref Value**: user error number for the current DBMS.
- **Val. Default Value**: activates/deactivates generation of DEFAULT orders for columns.
- **Quoted Identifier**: activates/deactivates generation of quotes around SQL identifiers (SQL name).
- **Qualifier** : enables prefix of object names. See [Prefixing Object Names](#).
- **Mode Generation Inc**: this parameter applies to incremental generation and can take values "Alter" and "Drop/Create".
 - "Alter" does not authorize deletion of objects (tables, indexes, etc.) at the level of generated scripts. Only those instructions that can be executed using the ALTER command are generated. For physical parameters, deletion is still authorized (this is the case notably for partitions).
 - "Drop/Create" authorizes object deletion. If an update cannot execute via the ALTER command, the object is deleted then recreated. By default, the parameter takes value "Alter".
- **Script Distribution**: indicates if the result of generation should be created in a unique file or in one file per object or in one file per object type.
- **SQL Script**: name of the file generated when this is a unique file. By default, this sub-folder is called REFEXT. You can customize at DBMS level. The arrow at extreme right of the field allows you to reinitialize the parameter.



You can also reinitialize all parameters of the object concerned. This action should be carried out with care.

- **Script Directory**: relative generation folder.
- The various **Ext** parameters allow you to specify extensions of each file generated for tables, datagroups, views, etc.

- **Conversion:** format of generated files (ANSI Windows or ASCII MS-DOS).
 - **CREATE CLUSTER:** activates/deactivates generation of CREATE CLUSTER orders.
 - **CREATE TABLE:** activates/deactivates generation of CREATE TABLE orders.
 - **CREATE TABLESPACE:** activates/deactivates generation of CREATE TABLESPACE orders.
 - **PRIMARY KEY:** activates/deactivates generation of PRIMARY KEY orders.
 - **FOREIGN KEY:** activates/deactivates generation of FOREIGN KEY orders.
 - **CREATE INDEX:** activates/deactivates generation of CREATE INDEX orders.
 - **CREATE PROCEDURE:** activates/deactivates generation of stored procedures.
 - **CREATE INDEX PK:** activates/deactivates generation of CREATE INDEX orders for primary key indexes.
 - **CREATE INDEX[UNIQUE]:** activates/deactivates generation of CREATE INDEX orders for unique indexes.
 - **CREATE VIEW:** activates/deactivates generation of logical views.
 - **CREATE SEQUENCE:** activates/deactivates generation of CREATE SEQUENCE orders.
 - **CREATE SYNONYM:** activates/deactivates generation of CREATE SYNONYM orders.
 - **CREATE TRIGGER:** activates/deactivates generation of triggers.
 - **Comments:** activates/deactivates generation of **HOPEX** comments in SQL script.
 - **UNIQUE:** activates/deactivates generation of UNIQUE orders.
 - **UNIQUE[PK]:** activates/deactivates generation of UNIQUE orders for primary keys.
 - **PRIMARY KEY syntax:** PRIMARY KEY orders are generated in CREATE TABLE order or in an ALTER TABLE order.
 - **Position FOREIGN KEY:** generation of FOREIGN KEY orders after each CREATE TABLE or grouped at end of script.
 - **COMMENT ON TABLE:** comments on tables (0:no comment, 1:one line, Total:all text)
 - **COMMENT ON COLUMN:** comments on columns (0:no comment, 1:one line, Total:all text)
- ☛ *Generation of comments is only possible for target systems that accept these (Oracle, DB2,...).*
- The various **Add-Ons** parameters activate/deactivate generation of add-ons on tables, datagroups, etc.
 - **Tbspace of Tables:** by default, tables are generated in the SYSTEM tablespace.
 - **Tbspace of Indexes:** by default, tables are generated in the SYSTEM tablespace.

Prefixing Object Names

There is a schema concept for most DBMSs, which enables definition of a logical grouping for objects.

Therefore at creation of a table for example, it can be automatically stored in a schema, and if this is not specified, a default schema can be automatically assigned.

In **HOPEX** there is not a schema concept, but a specific concept - the Qualifier - which enables prefix of database object names at generation. If for example you want objects to have names prefixed "MEGA", you must enter this value in the Qualifier field of the objects in question.

Inheritance

The Qualifier property can be defined at database level and on all other object types. There is an Inheritance system: if the Qualifier is not specified at the level of a table, by default it is the value entered on the database that is taken into account.

To prefix the name of an object at generation:

1. Open the properties dialog box of the object in question.
2. Click the drop-down list then **Options** > **Generation**.
3. In the **Qualifier** field, enter the value that will prefix the name of the object.

DBMSs concerned

The Qualifier is available for the following DBMSs:

- Oracle
- SQL Server
- DB2
- MySQL

SUPPORTED SYNTAX

CREATE TABLE Instruction

The CREATE TABLE instruction defines a table. The definition includes:

- Table name
- The names and attributes of its columns.
- The attributes of the table such as its primary and foreign keys.

The syntax is as follows:

```
CREATE TABLE table-name (coll-name coll-type [NOT NULL]
...
name-coln type-coln [NOT NULL])
```

For DB2, the syntax is as follows:

```
CREATE TABLE table-name (coll-name coll-type [NOT NULL]
...
name-coln type-coln [NOT NULL])
[in Tablespace <Name>]
```

For Oracle, the syntax is as follows:

```
CREATE TABLE table-name (coll-name coll-type [NOT NULL]
...
name-coln type-coln [NOT NULL])
[Tablespace <Name>]
```

- **table name** : "SQL" value for the table, or else defaults to the name of the table; unrecognized characters are replaced by "_"
- **col-name**: Value of the SQL Name attribute for the column, or by default the name of the column; unrecognized characters are replaced by "_"
- **col-type**
- **NOT NULL**: See [Managing NOT NULL](#).
- **Tablespace**: DB2 and Oracle: Name of the target tablespace for the tables

The PRIMARY KEY clause is generated within the CREATE TABLE command (see [PRIMARY KEY clause](#)).

Managing NOT NULL

Clauses NULL, NOT NULL and NOT NULL WITH DEFAULT are generated automatically on the columns of primary keys and on columns derived from obligatory attributes at time of synchronization.

These values can be initialized as "Null", "Not Null" or "Not Null with Default" as a function of configuration defined in the database Properties dialog box for **Not Null Columns**, in the **Synchronization** subtab of the **Options** tab.

The values proposed can then be modified on each column.

PRIMARY KEY clause

Defining a primary key

One or more of the columns in a table can be used to uniquely identify each row in the table. Values in these columns must be specified. They act as the primary key for the table.

A table must have only one primary key or none.

Each column name must identify a column in the table, and the column cannot be identified more than once.

Processing and generating SQL commands

After declaring the names of the columns in the table, if the **PRIMARY KEY** option is enabled, the name(s) of the columns in the primary key are declared as follows:

```
PRIMARY KEY (list of columns in the primary key)
```

The PRIMARY KEY clause is generated within the **CREATE TABLE** command.

Example 1	Example 2
The primary key "PK" has only one column, "PK-col". CREATE TABLE table-name (PK-col CHAR(9) NOT NULL, info1 CHAR(7), info2 CHAR(7), PRIMARY KEY (PK-col))	The primary key "PK1" has columns "PK11" and "PK12". CREATE TABLE table-name PK11 CHAR(9) NOT NULL, PK12 CHAR(9) NOT NULL, info1 CHAR(7), info2 CHAR(7), PRIMARY KEY (PK11, PK12))

For Oracle, the complete PRIMARY KEY clause is as follows:

```
CONSTRAINT PK_<key name> (list of columns in the primary  
key)
```

FOREIGN KEY clause

Database integrity can be ensured either by FOREIGN KEY clauses or by generated triggers, depending on the target DBMS. (In Oracle, it is ensured either with triggers, or with FOREIGN KEY clauses, depending on the database configuration.)

One or more columns in a table can refer to a primary key in this table or in another table. These columns form the foreign key. These columns do not need to have a value in each row.

The table containing the referenced primary key is a parent table. The table containing the foreign key is a dependent table.

Each column name must identify a single column in the table, and the same column cannot be identified more than once. If the same list of column names is specified in more than one FOREIGN KEY clause, these clauses cannot identify the same table.

The table name specified in the FOREIGN KEY clause must identify a parent table. A foreign key in a dependent table must have the same number of columns as the primary key for the parent table.

The number of foreign keys is unlimited.

Processing and generating SQL commands

After declaring the primary keys (PRIMARY KEY), the column name(s) for the foreign key(s) are declared for a table using FOREIGN KEY:

```
FOREIGN KEY (list of columns in the foreign key) REFERENCES
<Parent table name> [ON DELETE <Action>] [ON UPDATE
<Action>]
```

or:

```
ALTER TABLE tablename [ADD] FOREIGN KEY (list of columns for
the foreign key) REFERENCES <Parent table name > [ON DELETE
<Action>] [ON UPDATE <Action>]
```

For Oracle, the syntax is as follows:

```
CONSTRAINT FK_<name of the foreign key> (list if columns for
the foreign key) REFERENCES <Parent table name > [ON DELETE
<Action>] [ON UPDATE <Action>]
```

or:

```
ALTER TABLE...
```

Examples

The table "table1-name" has two foreign keys. These keys have no components.

```
CREATE TABLE table1-name
pk1 CHAR(9) NOT NULL,
pk2-rel12 CHAR(7) NOT NULL,
pk3-rel13 CHAR(7) NOT NULL,
info1 CHAR(7),
info2 CHAR(7),
PRIMARY KEY (pk1))
ALTER TABLE table1-name ADD FOREIGN KEY(cp2-rel12)
REFERENCES table2-name
ALTER TABLE table2-name ADD FOREIGN KEY(cp3-rel13)
REFERENCES table3-name
```

The table "table1-name" has a foreign key "fk2" which has two components, "fk21" and "fk22". The foreign key "fk2" has no reference (it is therefore a component of the primary key of another table).

```
CREATE TABLE table1-name
pk1 CHAR(9) NOT NULL,
fk21 CHAR(7) NOT NULL,
fk22 CHAR(7) NOT NULL,
info1 CHAR(7),
PRIMARY KEY (pk1)
ALTER TABLE table1-name ADD FOREIGN KEY (fk21, fk22)
REFERENCES table2-name
```

The table "table1-name" has a foreign key, "fk2". The foreign key "fk2" is equivalent to the primary key "pk2" which has two components, "pk21" and "pk22". The columns identified by "pk21" and "pk22" are "NOT NULL".

```
CREATE TABLE table1-name
pk1 CHAR(9) NOT NULL,
pk21 CHAR(7) NOT NULL,
pk22 CHAR(7) NOT NULL,
info1 CHAR(7),
info2 (CHAR7),
PRIMARY KEY (pk1)
ALTER TABLE table1-name ADD FOREIGN KEY (pk21, pk22)
REFERENCES table2-name
```

UNIQUE clause

A UNIQUE clause is generated for each unique index in the table, unless this index corresponds to the primary key.

Processing and generating SQL commands

For each unique index, the following clause is generated:

```
UNIQUE (col1,...,coln)
```

(col1,...n,coln) represent the columns used by the index.

CREATE INDEX Instruction (Oracle, Sybase, SQL Server)

Definition of an index

An index is a set of columns in a table for which a direct access is defined.

For Sybase and SQL Server, the value of the index-type attribute for the index determines what type of index is generated: UNIQUE, CLUSTERED, or UNIQUE CLUSTERED.

Processing and generating SQL commands

For each index a clause is generated as a function of the target DBMS.

For Oracle:

```
CREATE INDEX (column1,..., columnN) [TABLESPACE  
(TbSpaceName)
```

(column1, ..., columnN) represent the columns in the index; TbSpaceName is the name of the tablespace for the indexes (see [Configuring SQL generation](#)).

For Sybase and SQL Server:

```
CREATE [UNIQUE] [CLUSTERED] INDEX (IndexName) (TableName)  
(column1,..., columnN)
```

(column1, ..., columnN) represent the columns in the index; TbSpaceName is the name of the tablespace for the indexes (see [Configuring SQL generation](#)).

CREATE VIEW Clause

A view is defined for a database. It can include one or more tables.

```
CREATE VIEW view-name  
AS  
SELECT  
(column-name,column-name,...)
```

You can enhance this definition in the view specification (see [Defining Database Views](#)).

DEFINING DATABASE VIEWS

A physical view is a virtual table, of which structure and content are deduced from one or several other tables with an SQL query.

Database *views* are created in a tree format, which automatically generates part of the view definition. The user can then add to it as desired.

Creating Database Views

To create a physical view from a database:

1. On the desktop, click the navigation menu then **Data Architecture > Physical data Assets**.
2. In the edit area click **Physical views**.
The list of physical views appears in the edit area.
3. Click **New**.
The view creation wizard opens.
4. In the **Owner** field, select the database concerned.
5. In the **Physical View Component** field, click **New**.
6. Select the tables concerned by the view
7. Click **OK**.
The physical view editor appears.

The left tree displays tables to which the physical view relates, with their columns. The right tree displays the tables and columns that constitutes the view. By default these ones have the same names as the source tables and columns. You can rename them.



Add a table or a column to a view

To add a table to a view:

1. In the right-hand part of the editor, right-click the **Table** folder and select **Table view**.

Select the desired table and click **OK**.

To add a column to a view:

1. In the right-hand part of the editor, right-click the **Column** folder and select **Column view**.
2. Select the desired column and click **OK**.

SQL Definition

The right side of the dialog box, labeled **SQL Definition**, shows the SQL code that would be generated to define the view. The code is initially calculated based on the definition indicated in the tree.

You can modify this code, in particular by using joints. You can also directly enter modifications in the SQL frame.

View joints

By default, the edition of logical views window proposes the foreign keys of the selected tables where these exist.

It is thus possible to complete specification of a view by associating with it foreign keys, potential sources of joints.

To associate a foreign key with the view:

1. Select the foreign key in the tree on the left and drag it into the SQL definition field.

User mode

You can modify the view code by typing directly in the SQL definition field:

1. Click the **Save** button so that the **SQL Definition** will be saved in the repository as is.

After you have modified the definition, you can restore the definition as determined by the tree:

1. Click the **Initialize the SQL definition**  button.
A message warns you that the previously saved definition will be reinitialized. %In other words, any user additions made to the definition will be lost.
1. Click **OK** to confirm.

Fields

Field categories correspond to object types used in the declarative tree: table, view, column and foreign key column. Fields displayed in the SQL definition correspond to elements declared in the tree.

The foreign key type does not produce a field category: usable fields are derived from key columns and not from the keys themselves.

The **Field properties**  button displays properties of the object corresponding to the selected field.

If an object is added to the tree, a corresponding field becomes available for insertion.

If an object is renamed in the tree or in the repository, its references remain valid and the fields are displayed in the text with the new name.

If an object is deleted in the tree or in the repository, its references become invalid and are indicated as such in the fields.

Defining a Data Group

A data group - or tablespace - is a group, in the same physical pages of the database, of the rows of several tables to optimize queries, joints in particular. Example: Tablespace in DB2, Cluster in Oracle etc.

To define *data groups* in the database:

1. Open the database properties dialog box.
2. Click the drop-down list then **Components**.
3. The **Data groups** section displays the list of data groups
4. Click the **New** button.
5. In the dialog box that opens, indicate the **Name** of the data group.
6. Click **OK**.

Then open the properties window of the data group to define the tables and indexes that it includes.

To specify the tables and indexes included in the data group:

1. Select the data group and click **Properties**.
The data group properties dialog box appears.
2. Click the drop-down list then **Tables**.
3. Click the **Connect** button.
4. In the list of database tables presented, select the tables to be included.
 Click **Disconnect** to remove a table from the list in the case of error.
5. Carry out the same operations for the indexes of the data group.
6. Click **OK**.

DEFINING TRIGGERS FOR A DATABASE

A trigger is processing recorded in a database, which automatically triggers on updating a table.

Creating Triggers

Triggers are defined at the level of database tables.

It should be noted that triggers are defined as a function of the target DBMS; this is why it is important to check that the target DBMS is correct before creating triggers.

If the target DBMS is later modified, triggers created for the DBMS are not deleted but deactivated.

To create a trigger in **HOPEX Data Governance**:

1. On the desktop, click the navigation menu then **Architecture > Hierarchy View**.

*In HOPEX Information Architecture, click the navigation menu then **Data Architecture > Logical Data Assets**.*

2. Expand the folder of the database then the table concerned.
3. Right-click the **Trigger** folder and select **New > Trigger**. The dialog box for creating a trigger opens.
4. Specify the name of the trigger and actions triggered. See [Trigger triggering](#).

Trigger triggering

Triggering can occur following one of the three following actions:

- At **Creation** of a row in the table.
- At **Deletion** of a row.
- At **Modification** of the table or of a particular column.

In addition, you can choose to run it **Before** or **After** these actions, on the entire table, or on each row concerned.

References

The "Reference of old rows" and "Reference of new rows" fields create in the trigger code references to lines inserted, deleted or updated.

The name indicated in the "Reference of old rows" field corresponds to the line that existed before the update.

The name in the "Reference of new rows" field indicates the line after the update.

In the case of insertion, only the new line is valid.

In the case of deletion, only the old line is valid.

SQL Definition

The **SQL Definition** option in the properties window of the trigger presents the trigger code.

To display the trigger code:

1. Right-click the trigger and select **Properties**.
The properties window of the trigger appears.
2. Click the drop-down list then click **Texts**.
3. Select the **SQL Definition** tab.

Repository Integrity

Repository integrity is managed by creation of foreign keys on a database.

It groups all constraints allowing a check of the impact of modification of a table in tables connected to it.

It could be that the existence of keys in certain DBMSs does not involve a systematic check. It could also be that you wish to customize constraints to be applied on a particular table.

This is why you can generate in triggers the code that corresponds to repository integrity management.

To generate repository integrity of a table:

1. Right-click the database and select **Trigger Initialization**.
The trigger generation dialog box opens.
2. Select one of the options offered:
 - Generate Trigger by type
 - Generate Trigger by repository integrity
3. Select the tables of the database.
4. Click **Next**.

Triggers are automatically created for the selected tables.

When generation has been completed, the triggers appear under the **Trigger** folder available under each table. There are three trigger types:

- An update trigger (U_ followed by table name), which enables specification of the action to be carried out in case of modification of a line of the table that is part of the foreign key.
- A delete trigger (D_), which specifies the action to be carried out in case of deletion.
- An insert trigger (I_), which specifies the action to be carried out in case of insertion.

These triggers are only valid for a target DBMS. When you change DBMS, you must regenerate the triggers.

USING STORED PROCEDURES

HOPEX Information Architecture allows you to create *stored procedures*.

A stored procedure combines a procedural language and SQL requests within a program. It enables execution of a particular task on a database. It is recorded in a database and can be called from a program external to the database or from a trigger.

A stored procedure can be implemented in two ways; either as a procedure or as a function.



A procedure is a set of instructions executing a sub-program.



A function is a procedure returning a value on completion of execution.

To create a procedure stored on a database:

1. Open the database properties dialog box.
2. Click the drop-down list then **Components**.
The **Stored Procedures** section displays the list of stored procedures.
3. Click the **New** button.
4. In the window that opens, specify the name of the procedure and its nature (Procedure or Function).
5. Click **OK**.
The stored procedure appears. Open its properties to define its code.

Example of stored procedure for Oracle

This is an example of a stored procedure updating the unit price of a part as a function of the part identifier:

```
CREATE PROCEDURE update_part_unitprice (part_id IN INTEGER,
new_price IN NUMBER)
IS
  Invalid_part EXCEPTION;
BEGIN
  -- HERE'S AN UPDATE STATEMENT TO UPDATE A DATABASE RECORD
  UPDATE sales.parts
    SET unit_price = new_price
    WHERE id = part-id;
  -- HERE'S AN ERROR-CHECKING STATEMENT
  If SQL%NOTFOUND THEN
    RAISE invalid_part;
  END IF;
EXCEPTION
  -- HERE'S AN ERROR-HANDLING ROUTINE
  WHEN invalid_part THEN
    raise_application_error(-20000, 'Invalid Part ID');
```

```
END update_part_unitprice;
```

ADDING PHYSICAL PROPERTIES TO DATABASE OBJECTS

When your database has been defined in a relational diagram, you can generate the corresponding SQL scripts for the different DBMSs.

The physical data navigation pane allows you to complete database physical modeling by specifying parameters specific to each DBMS and therefore to produce complete SQL scripts.

In **HOPEX**, you can also import physical parameters defined on reverse engineered objects. See: [Physical Properties Reverse Engineering](#).

You can adapt the same logical model to several DBMSs. It is not necessary to duplicate objects.

Target DBMSs

To define a target DBMS on a database:

1. Open the properties dialog box of the database concerned.
2. Click the **Characteristics** page.
3. Specify the **target DBMS** field in the corresponding field.

See also [Importing a DBMS Version](#).

Creating Physical Properties

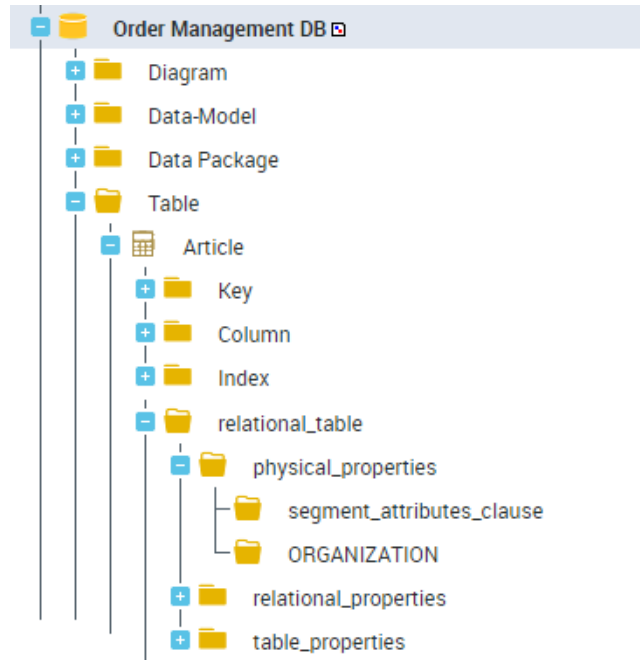
To create physical properties on objects of a database:

1. On the desktop, click the navigation menu then **Data Architecture** > **Physical data**.
2. In the edit area click **Database physical hierarchy**.
The list of repository databases appears in the edit area.
3. Expand the folder and the sub-folders of the databases concerned.

Parameters are presented in tree form, conforming to SQL grammar of the DBMS considered (refer to DBMS SQL documentation).

Two folder types are presented in the tree:

- Navigation folders.
- Parameter groups that you must instance.



Each parameter group, represented by an "SQL clause" object, has a properties page enabling value definition.

SQL clauses defined in this way are accessible just like repository standard objects. For example it is possible to query SQL objects that have a given parameter value.

By default, clauses cannot be reused from one object to another. It is however possible to define a clause for one object and connect it to other objects. In this case, any modification of the clause affects all objects that use it.

Objects containing physical parameters

Not all objects in **HOPEX** support physical parameters. These concern only:

- Data groups
- Tables
- Indexes
- Clusters

Creating a new clause

To define object parameters:

1. Right-click the corresponding parameter group and select **New > SQL clause**.

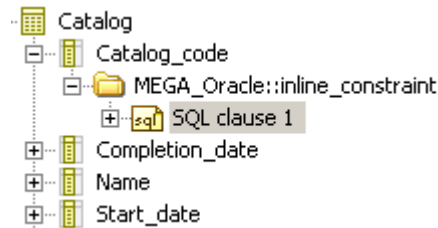
2. Open the properties window of the clause and specify the value of the parameter to be defined.

Connecting a clause

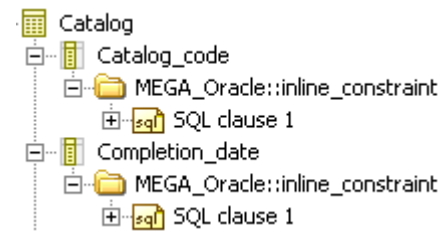
You can assign the same clause to several objects, on condition that you connect the correct clause type.

Consider the "Order Management" database with Oracle 9i as DBMS.

On the column "Code_catalogue", create "Clause 1" of type "inline_constraint".



You can connect "Clause 1" to another column. Being the same type of clause, this is copied on the new column with no problem.



On the other hand, if you connect "Clause 1" to an object of type different from that initially defined on "Clause 1" - for example "Storage_clause" - then "Clause 1" changes type to take that of the last element connected. In other words, "Clause 1" that was type "inline_constraint" takes type "storage_clause". This change is reflected on the start columns to which "Clause 1" was connected.

Naming clauses

Standard case

By default, the clause takes the name of the clause type to which it is attached. When you attach a clause to another type, the name automatically adapts.

Specific naming

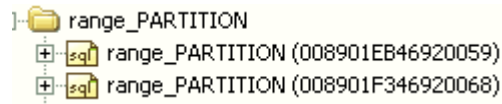
A specific name can be given to a clause. In this case, the clause name becomes static and will not be modified at change of clause type.

☛ You can return to dynamic mode by overloading the name empty.

Specific naming is essential when a clause is used in different contexts (generic clause).

Multiple clauses

For a given level, several clauses can be attached to the same clause type. To distinguish different clauses, the clause name comprises the name of the clause type followed by its hexaIdAbs.



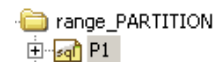
Naming from a property

It is possible to modify standard behavior by defining an automatic naming rule for an SQL clause type. This configuration is carried out at the clause type _settings property level. In the example below, the configuration on clause type "range_PARTITION" for Oracle 9i indicates that the name of SQL clauses of this type will be built from the value of the PARTITION property.

```
[NameIdentification]
AttrForNameIdentification=A3F2DE8C417E06C9
```

When the parameter has been executed, names of SQL clauses are automatically calculated from values of the PARTITION property.

Property	Value
Key compression	
PARTITION	P1
Value list	

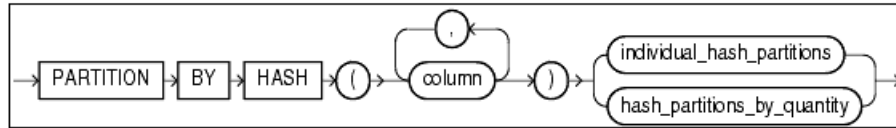


The name of SQL clauses is not taken into account in SQL generation. In the example provided, it is the value of the PARTITION property that feeds the generated SQL scripts.

Physical Model Customization Example

You can partition a table to simplify data access or to manage the information blocks differently.

Suppose you wish to partition the "Order Line" table of the "Order Management" database using the Oracle by hash method. This method enables dynamic calculation of table partitioning.



Hash partitioning instruction syntax

Check that the database has Oracle 9i as target DBMS.

- 1. Open its properties dialog box and click the **Characteristics** page. The DBMS name is indicated in the **Target DBMS** box.

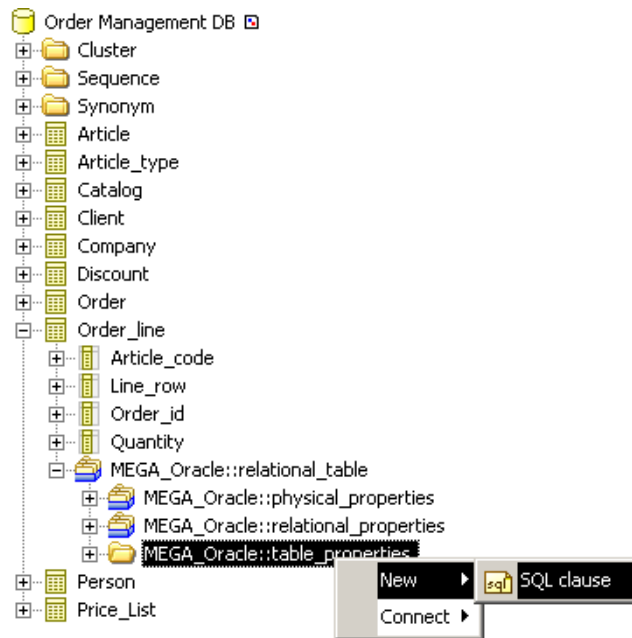
Display the physical properties of the "Order Management" database:

1. On the desktop, click the navigation menu then **Data Architecture > Physical data**.
2. In the edit area click **Database physical hierarchy**. The list of repository databases appears in the edit area.
3. Expand the folder and the sub-folders of the "Order Management" database to display the parameters linked to the Oracle grammar.

To partition the "Order line" table:

1. Expand the "Order_line" table.
2. Expand the "MEGA_Oracle::relational_table" parameter group.

3. Right-click the "MEGA_Oracle::table_properties" clause type and select **New > SQL clause**.



4. Name the clause "Order_line/Table_properties".
It appears in the navigation tree.
5. Under this clause, expand the "MEGA_Oracle::table_partitioning_clauses" parameter group. It contains the different partitioning types that can be produced in Oracle.
6. On the "MEGA_Oracle::hash_partitioning" folder, create the clause "Order_line/hash_partitioning".
7. Open its properties page.
8. In the **PARTITION BY HASH** page, indicate the columns on which breakdown applies. To do this, connect the columns concerned by partitioning.
9. Close the properties dialog box.
10. Under the clause "Order_line/hash_partitioning" two clause types are available:
 - **individual_hash_partitions**: enables naming of each partition.
 - **hash_partitions_by_quantity**: enables definition of the number of partitions you wish to create.
11. Create the clause "Order_line/Hash_partition_by_quantity".
12. Open its properties page.
13. Select the **PARTITIONS** page.
14. In the **Hash partition quantity** field, indicate the number of partitions.
These partitions are represented by data groups.
15. In the **STORE IN** field, connect the data groups.

To obtain the script corresponding to this partitioning, right-click the "Order line" table and select **Generate the code**.

```
SPOOL \_MEGASQL.LST;
PROMPT ----- ;
PROMPT  Compte-Rendu de génération ;
PROMPT ----- ;
/* -----
/* Begin of generation Oracle 9i for database Order_Management_DB the June 14, 2006 at 14: 0:49 */
/* -----
CREATE TABLE "Order_line"
(
  "Article_code" CHAR(6),
  "Line_row" NUMBER(7),
  "Order_id" CHAR(5),
  "Quantity" NUMBER(7),
  CONSTRAINT "FK_Order_line" PRIMARY KEY
  (
    "Line_row",
    "Order_id"
  )
)
PARTITION BY HASH ("Line_row","No_commande")
PARTITIONS 5 STORE IN ("tbs_1","tbs_2","tbs_3","tbs_4") TABLESPACE "SYSTEM";
/* -----
/* Foreign Key FK_Order */
/* -----
ALTER TABLE "Order_line"
ADD CONSTRAINT "FK_Order" FOREIGN KEY
(
  "Order_id"
) REFERENCES "Order";
/* -----
/* Foreign Key FK_Order_line_ */
/* -----
ALTER TABLE "Order_line"
ADD CONSTRAINT "FK_Order_line_" FOREIGN KEY
(
  "Article_code"
) REFERENCES "Article";
SPOOL OFF;
DEFINE _EDITOR = "notepad.exe";
EDIT \_MEGASQL.LST
EXIT;
/* -----
/* End of generation Oracle 9i for database Order_Management_DB the June 14, 2006 at 14: 0:51 */
/* -----
```

Generating the SQL File

When object customization has been completed, you can generate the corresponding script file to consult the results, without having to regenerate the entire database.

For example, to generate the SQL file of an index:

- Right-click the index and select **Generate the code**.

See also [Generating SQL scripts](#).



REVERSE ENGINEER TABLES



Reverse engineering enables you to take existing databases and create the corresponding tables and columns in the **HOPEX** repository.

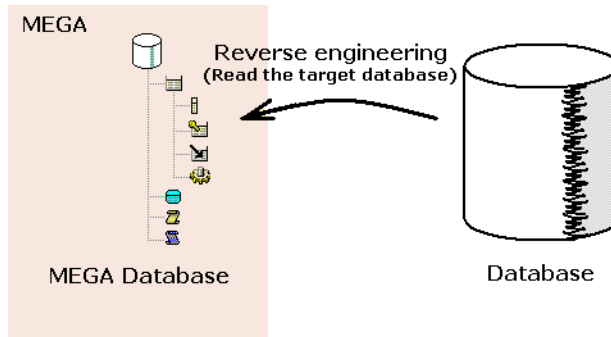
Reverse engineering can be done from a previous database extraction. See [Extracting Database Schema Description from Data Sources](#).

The tables and columns are integrated in a database where they can be easily maintained and documented.

The following points are covered here:

- ✓ [Running Reverse Engineering](#)
- ✓ Recognizing Datatypes by ODBC
- ✓ [Physical Properties Reverse Engineering](#)
- ✓ [Extracting Database Schema Description from Data Sources](#)

RUNNING REVERSE ENGINEERING



To run reverse engineering:

1. In the navigation menu click **Tools > Reverse Engineer a Database**. A window opens.
2. Select the source database.
3. Click **Next**.
4. Select the **Type** of data source: Extraction file obtained using the data extraction utility (see [Extracting Database Schema Description from Data Sources](#)).
5. Indicate the location of the file.
6. Define the **Options**:
If the database has already been reverse engineered, it is possible to specify the extent of the **Reinitialization**, which can concern only **Additions**, **Deletions** or **Modifications**.
The **Simulation** option is provided so you can simulate the operation and generate a report indicating the impact on the repository.
7. Click the **Next** button to run reverse engineering. Messages inform you of progress.
8. On completion of processing, a report displays the details of the execution.
At the end of reverse engineering, the database tables and columns are created. However, the relational diagram must be created in order to view the database in a graphical model.

☺ If non-standard datatypes were created in the DBMS, they must be added to the configuration if they are to be recognized by **HOPEX**. See [Recognizing Datatypes by ODBC](#).

RECOGNIZING DATATYPES BY ODBC

When reading a database, HOPEX asks the database (Oracle in our example) what are the datatypes, through the intermediary of ODBC.

For example, if Oracle contains a datatype "SQL_LONGVARCHAR", HOPEX will know that this is a "P-Text" datatype.

To do this, HOPEX uses the "xmdb.xdr" file, located in the "Mega\Mega_Std" folder.

The part of the "xmdb.xdr" file that concerns correspondence of datatypes starts with the following line:

```
<!-- Mapping of standard Datatypes -->
```

Each line is presented as follows:

```
<!-- #DataType(5B845EAE3BAF0020) = "SQL_LONGVARCHAR"  
Standard::Types::Pivot::P-Text -->
```

- The figure between brackets (here 5B845EAE3BAF0020) is the absolute identifier of the datatype.
- The text between quotation marks (here SQL_LONGVARCHAR) is the name of the ODBC datatype.
- The path that follows (here Standard::Types::Pivot::) specifies in which HOPEX package the corresponding data type is located.
- Finally, the HOPEX datatype (here P-Text) is indicated.

To recognize the datatypes the software uses, in order, the three following lists:

- **List of datatypes added by the user.**
When you create a datatype manually, in order that it will be recognized you must add a rule to the list.
- **List of datatypes of the ODBC driver used.**

Example: here there is one list of datatypes for Ingres and two lists for Oracle for different drivers (Oracle driver and Microsoft driver)

```
<!-- Declaration Rule Parser Datatypes for Ingres -->
<!-- #DataType(13ED147938B5008F) = "SQL_CHAR,SQL_INGRESC"
Standard::Types::Pivot::P-Character -->
<!-- #DataType(5B845EAE3BAF0020) =
"SQL_VARCHAR,SQL_INGRESTEXT" Standard::Types::Pivot::P-Text
-->
<!-- #DataType(13ED1A3D38B501E0) =
"SQL_DECIMAL,SQL_INGRESMONEY" Standard::Types::Pivot::P-
Currency -->
<!-- #DataType(13ED1A3038B501D5) =
"SQL_TIMESTAMP,SQL_INGRESDATE" Standard::Types::Pivot::P-
Date -->
<!-- Declaration Rule Parser Datatypes for Oracle with
Oracle Driver -->
<!-- #DataType(B55EB70C3471008F) =
"SQL_TIMESTAMP,SQL_Oracle8DATE" Standard::Types::Pivot::P-
Datetime -->
<!-- #DataType(45F44A3B3BAF003D) =
"SQL_VARBINARY,SQL_Oracle8RAW" Standard::Types::Pivot::P-
Binary -->
<!-- #DataType(B55EB71E347100BC) =
"SQL_VARCHAR,SQL_Oracle8ROWID" Standard::Types::Pivot::P-
Timestamp -->
<!-- #DataType(B55EB6F134710044) =
"SQL_DECIMAL,SQL_Oracle8DECIMAL" Standard::Types::Pivot::P-
Numeric -->
<!-- #DataType(B55EB6F134710044) =
"SQL_TINYINT,SQL_Oracle8NUMBER(3,0)"
Standard::Types::Pivot::P-Numeric -->
<!-- #DataType(B55EB6F134710044) =
"SQL_INTEGER,SQL_Oracle8DECIMAL" Standard::Types::Pivot::P-
Numeric -->
<!-- Declaration Rule Parser Datatypes for Oracle with
Microsoft Driver -->
<!-- #DataType(B55EB70C3471008F) =
"SQL_TIMESTAMP,SQL_OracleDATE" Standard::Types::Pivot::P-
Datetime -->
<!-- #DataType(45F44A3B3BAF003D) =
"SQL_VARBINARY,SQL_OracleRAW" Standard::Types::Pivot::P-
Binary -->
<!-- #DataType(B55EB71E347100BC) =
"SQL_CHAR,SQL_OracleROWID" Standard::Types::Pivot::P-
```

```
Timestamp -->
<!-- #DataType(B55EB6F134710044) =
"SQL_DECIMAL,SQL_OracleNUMBER" Standard::Types::Pivot::P-
Numeric -->
```

- **List of standard datatypes.**

Datatype Recognition Problems

It is necessary to add a correspondence between datatypes in the "xmdb.xdr" file in the following two cases:

- **The datatype is incorrectly recognized.**
The user can confirm this by noting for example that an "Integer" datatype in Oracle becomes a "P-Number" datatype in MEGA, instead of "P-Integer".
- **The datatype is not recognized.**
When the datatype is not recognized, a message of the following type appears at the bottom of the dialog box when reading the target database:
DataType 'SQL_Data8' not recognized. See Help for how to configure correspondence of Datatypes.

To add a datatype correspondence:

1. Open the extraction results file.
2. Locate in this file the correspondence between the datatypes.
For example, in the expression:

```
<COLDATATYPE
Name="COLDATATYPE.JAVA036HTTP036DEPLOYMENT036DIGEST036.IS09
5DOC095ROOT" DataType="SQL_DECIMAL"
```

the ODBC datatype is "SQL_DECIMAL" and the HOPEX datatype is "SQL_Oracle8DECIMAL".

3. Add in any location in the "xmdb.xdr" file a line indicating the correspondence between datatypes constructed as above.
Example:

```
<!-- #DataType(B55EB6F134710044) = "SQL_Data8"
Standard::Types::Pivot::P-Text -->
```

Here we have defined the "SQL_Data8" datatype as a "P-Text" datatype.

PHYSICAL PROPERTIES REVERSE ENGINEERING

Reverse engineering also enables creation in **HOPEX** of physical properties on objects of a database.

Physical properties are parameters enabling expression, for a relational object (table, index, etc.), of the way in which information will be stored within a database. These parameters are specific to each DBMS and can evolve depending on versions of the same DBMS.

See also [Adding Physical Properties to Database Objects](#).

Default Values

Certain DBMS properties are automatically reversed in **HOPEX** even if they have not been explicitly specified.

So as not to recover these values by default, **HOPEX** provides for each DBMS and for each DBMS version a generic clause that contains the list of these default values and treating them specifically.

At reverse engineering, DBMS properties with values equal to values defined in the generic clause are not imported.

You can activate the generic clause by importing into **HOPEX** the .mol file associated with each DBMS in the Mega_Std folder of your installation.

Eliminating Redundant and Transverse Values

At reverse engineering of objects in **HOPEX**, certain physical properties that have not been clearly determined are automatically regenerated by the DBMS via an inheritance mechanism.

With Oracle for example, the value of property PCTFREE in a table, if it has not been specified, is directly inherited from that of its attached tablespace. Such a value is called transverse, since it is derived from an inheritance between two distinct object types. A value is said to be redundant if inheritance is derived from objects of the same type.

At reverse engineering, **HOPEX** does not recover transverse and redundant values.

Only the management of redundant values can be customized.

Specific Cases

Physical properties of tablespaces

In certain cases, reverse engineering of object physical properties requires an ODBC connection with DBMS Administrator rights. For example with Oracle, reverse engineering of the physical properties of tablespaces requires that you use a "System" account.

Clusters Reverse Engineering

Reverse engineering of a cluster in Oracle is carried out correctly if the connecting user verifies one of the two following conditions:

- The user is owner of the cluster
- The user has Administrator profile

If the cluster is not accessible, it is not reversed.

When the user sees the cluster but is neither owner nor administrator, the cluster is reversed, but the link between columns of the cluster and columns of the attached tables is not reversed in **HOPEX**.

☛ *From a technical viewpoint, for an administrator user, reverse engineering depends on the view oracle sys.all_clu_columns (relationship between cluster columns and table columns). This view enables reversing only of information relating to objects of which the user is owner.*

EXTRACTING DATABASE SCHEMA DESCRIPTION FROM DATA SOURCES

HOPEX Data Source Extractor is an application that uses ODBC APIs to extract the schema definition from a database. This description, obtained in structured format, can then be used for reverse engineering purposes in **HOPEX** or for generation in modification mode.

The extraction tool is available in 64 bit-version.

It can be deployed separately from HOPEX.

Required Data Source Configuration

To use **HOPEX Data Source Extractor**, you must have the **ODBC Data Sources (64 bits)** tool. This Microsoft tool is installed with Windows and is accessible from the **Start** menu.

Downloading HOPEX Data Source Extractor

To install the tool, make sure you have installation rights on your computer.

HOPEX Data Source Extractor is available on the MEGA HOPEX Store. To download it:

1. Visit the HOPEX Store at: <https://store.mega.com/modules>.
2. Select the **HOPEX Data Source Extractor** module.
3. Unzip the file on a computer that has access to the relevant database.

Starting Data Extraction

For data extraction, it is necessary to define an ODBC data source with the **ODBC Data Source Administrator** tool:

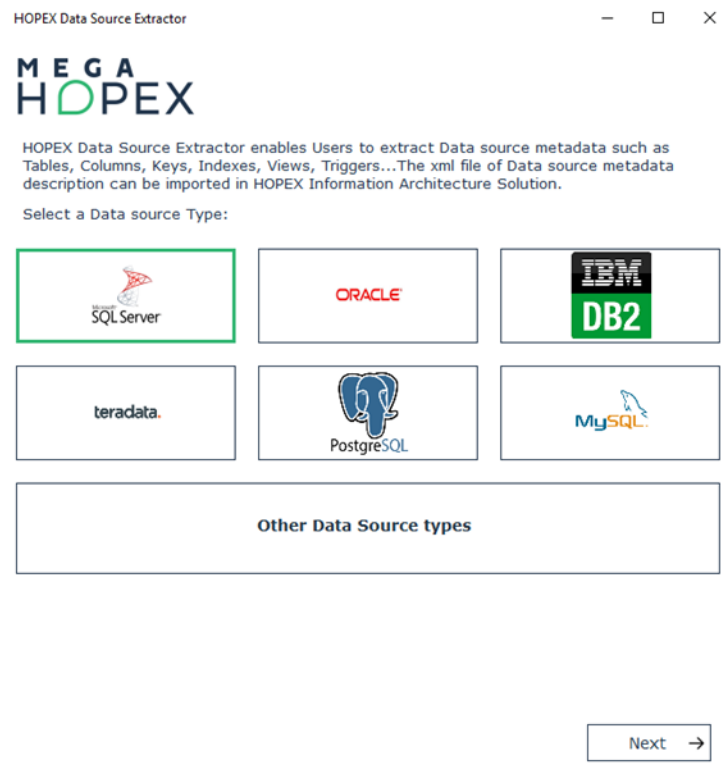
1. Launch the **ODBC Data Source Administrator** tool.
2. Click the **Drivers** tab.
3. Select the driver and click **OK**.

The database driver must have a conformance level of 1 or higher. The object extraction field depends on the driver used in the ODBC data source definition.

To extract the description of a database:

1. Run the HOPEX Data Source Extractor utility (in the directory indicated when downloading the module).
A wizard appears.

2. Select the type of data source from which you want to extract the schema description.
The main supported DBMS are presented, you can display the other data source types by clicking on **Other Data Source Types**.



3. Click **Next**.

The list of corresponding data sources appears.

☛ *This list is empty if these connections are not defined or could not be established.*

The screenshot shows a window titled "HOPEX Data Source Extractor". At the top left is the "MEGA HOPEX" logo. Below the logo, the text reads: "Select a Data Source from the list, or use ODBC Data Source Administrator button to Add a new Data Source". Underneath, it says "Select Data Source :". A table lists four data sources, all using the "ODBC Driver 17 for SQL Server". At the bottom left is a button labeled "ODBC Data Source Administrator". At the bottom right are two buttons: "back" with a left arrow and "Next" with a right arrow.

Name	Driver
test sql server 2017	ODBC Driver 17 for SQL Server
Employee	ODBC Driver 17 for SQL Server
Product	ODBC Driver 17 for SQL Server
Mega NAF	ODBC Driver 17 for SQL Server

4. Select a data source and click **Next**.

The connection windows appears.

5. If they are not already defined at the data source level, specify a **User ID**, a **Password**, and a **Server Name**. If other parameters are required by the ODBC driver, you will be prompted for them when the connection is established.

HOPEX Data Source Extractor

MEGA
HOPEX

SQL Server

Selected Data Source	test sql server 2017
User ID	
Password	
Server Name	

← back Next →

- Click **Next** to confirm the connection.
Once the connection has been established, select the desired extraction options.

☞ If some of the options remain disabled, this is because the driver does not support them.

😊 To obtain information on the ODBC protocol used, click the **ODBC information** button.

HOPEX Data Source Extractor

MEGA
HOPEX

Please select options for metadata description and define location of extraction file. The preview helps you to select the scope of objects to extract otherwise all objects of Data source will be extracted by default.

Selected Data Source: Developer
Driver Name: ODBC Driver 17 for SQL Server

SQL Server

Options

- Tables ☒
- Primary Key ☒
- Foreign Key ☒
- Indexes ☒
- Primary Indexes ☒
- [Show more](#)
- Synonyms ☐
- Tablespace ☐
- Physical Views ☐
- Procedures ☐
- Triggers ☐
- Sequences ☐
- Clusters ☐
- Physical Parameters ☐
- [ODBC Information](#)

Owner: **ALL**

Location File: [Browse](#)

[Preview](#)

Name	Comment

[back](#) [Extract](#)

- Select the elements to be extracted in addition to the tables and columns. By default, all of these elements are selected.
All the accessible tables are displayed, whether or not you are the owner.
Synonym tables can also appear if you select the corresponding check box.

📖 A synonym is an alternative name given to an object (table, view, stored procedure, synonym and sequence). A synonym can be defined to indicate an object in another database.

To view only those tables belonging to a specific **Owner**, select the appropriate owner from the drop-down list. It may take a few seconds for the list of owners and their tables to appear. Table extraction takes a few minutes.

The following elements are included in the extraction:

- Primary keys (**Primary keys**).
- Foreign keys (**Foreign keys**).
- Index (**Index**): these are indexes that do not use primary keys.
- Primary index (**Primary index**): these are indexes that do not use primary keys.

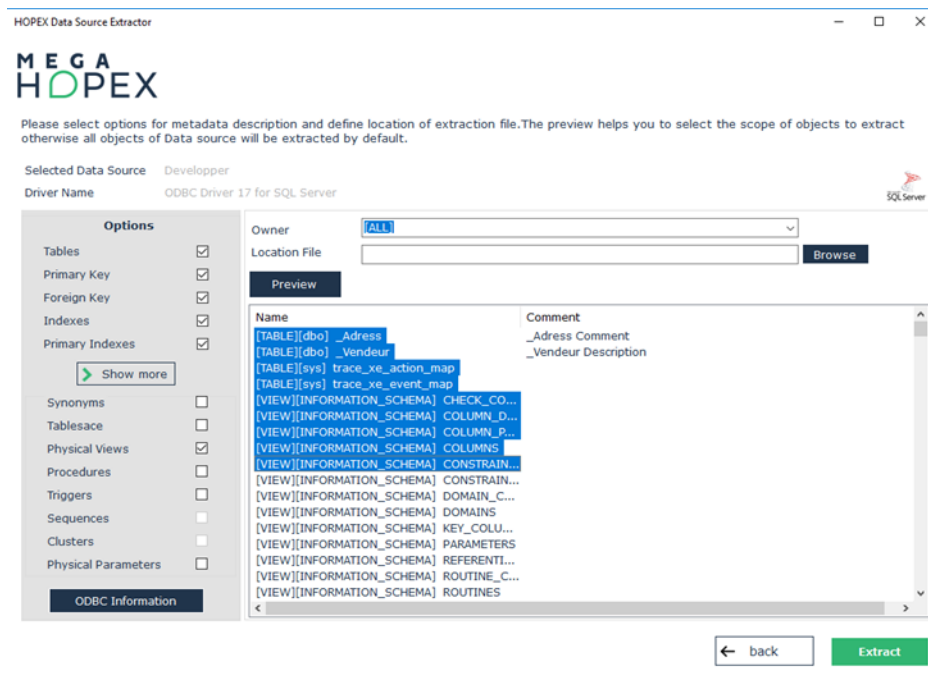
⚠ **Not all drivers support the ODBC primitives used to extract these elements; if this is your case, a message will indicate this in the report file. In addition, some DBMS do not handle the corresponding concepts, which are then ignored.**

The **Destination file** field specifies the name and path of the extraction file; use the **Browse** button to specify its location.

8. After selecting the extraction options, click the **Extract** button to begin the extraction.

A message reports the number of tables extracted. You can select the **Warnings** button to view the report file.

You can view the list of accessible tables by clicking the **List Tables** button, and then select specific tables from the list for extraction (all tables are selected by default).



9. On completion of extraction, click **Open file** to consult the result. The report file is available at {Current User Data}/Local/Mega.

⚠ *If the extraction is incomplete, it is advisable to use another driver with a compliance level higher than 1.*

The result file can be used for the reverse engineering (see [Reverse engineer tables](#)). It contains a database description in the form of **HOPEX** objects.

When the extraction operation has been completed:

- » Click **Close** to disconnect from the data source.
- » Click **Back** to perform a new extraction.

Extraction Report File

The file that reports on the table extraction is created by the ODBC extraction utility. It is called <FIC>_CRD.TXT where <FIC> represents the first three characters of the name of the results file.

It contains a list of the tables that were reread.

Example:

```
=====
Data Source Extracting: DATASOURCE
=====
Table: OWNER.TABLENAME1
Table: OWNER.TABLENAME2
(cont.)
=====
End of extraction
=====
```

Extraction Results File

The extraction results file contains the description of the tables and columns that results from the read. This file has extension ".xml".

Example of extraction file:

← → ↺ ↻ ⓘ File | C:/Temp/tmp.xml

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<?xml version="2.0">
  <DATABASE NameId="Database" DatabaseName="Airport Services">
    <QUALIFIER NameId="QUALIFIER.SQLPrmSpecific032Qualifier" QualifierValue=""> </QUALIFIER>
    <TABLE NameId="TABLE.Account" DBBName="Account" Order="9999" Comment="">
      <TBLCOL NameId="TBLCOL.Account.Account095Number" Order="1" DBBName="Account_Number" Comment="" Decimale="NULL" Length="10" NotNull="M" Default="">
        <COLDATATYPE NameId="COLDATATYPE.Account.Account095Number" DataType="SQL_CHAR" TypeTech="SQL_SQL Server 2008nchar"/>
      </TBLCOL>
      <TBLCOL NameId="TBLCOL.Account.Status" Order="2" DBBName="Status" Comment="" Decimale="NULL" Length="10" NotNull="N" DefaultValue="">
        <COLDATATYPE NameId="COLDATATYPE.Account.Status" DataType="SQL_CHAR" TypeTech="SQL_SQL Server 2008nchar"/>
      </TBLCOL>
      <TBLCOL NameId="TBLCOL.Account.Customer" Order="3" DBBName="Customer" Comment="" Decimale="NULL" Length="10" NotNull="N" DefaultValue="">
        <COLDATATYPE NameId="COLDATATYPE.Account.Customer" DataType="SQL_CHAR" TypeTech="SQL_SQL Server 2008nchar"/>
      </TBLCOL>
      <PK NameId="PK.Account.PK095Account" DBBName="PK_Account" KeyType="P" Order="1">
        <PKCOL NameId="PKCOL.Account.PK095Account.Account095Number" Order="1" Reference="TBLCOL.Account.Account095Number"/>
      </PK>
      <INDEX NameId="INDEX.Account.PK095Account" DBBName="PK_Account" Type="U" Clustered="Y" Sort="NULL">
        <IDXCOL NameId="IDXCOL.Account.PK095Account.Account095Number" Order="1" Sort="A" Reference="TBLCOL.Account.Account095Number"/>
      </INDEX>
    </TABLE>
    <TABLE NameId="TABLE.Customer" DBBName="Customer" Order="9999" Comment="">
      ...
    </TABLE>
    <TABLE NameId="TABLE.Purchase" DBBName="Purchase" Order="9999" Comment="">
      ...
    </TABLE>
  </DATABASE>
</xml>
```

Customizing ODBC Extraction

When the extraction is incomplete or does not correspond to your needs, you can customize the extraction with the Odwdbex.ini file. This configuration depends on the ODBC driver you are using.

You can customize the extraction in a number of different ways by using:

- ODBC standard APIs available for the main concepts (Table, Column, Key, Index, etc.)
- **HOPEX** queries delivered as a replacement for ODBC standard APIs
- customized queries.

By default, ODBC standard APIs are used for the main concepts and **HOPEX** queries for the other concepts. For some ODBC drivers, **HOPEX** queries are used for the main concepts as the result obtained by the ODBC standard is incomplete.

Using the Odwdbex.ini file and customized queries

To customize extraction:

1. Contact your data administrator to obtain the customized queries corresponding to your ODBC driver used to select objects (Eg: primary keys, foreign keys, sequences, etc).
2. In the "All users" folder in Windows, create a file named Odwdbex.ini (example: C:\Documents and Settings\All Users\ApplicationData\Mega\Odwdbex.ini).

3. Edit the file and add the queries for the concepts whose behavior you want to manage. The concepts that are not cited here remain unchanged.

```
[<DBMS Name>]
PRIMARY KEYS="Custom query"
FOREIGN KEYS="Custom query"
TBLCOLUMNS="Custom query"
...
```

The <DBMS Name> value depends on the ODBC utility. To obtain the appropriate value:

1. Run the **HOPEX** extraction tool (mgwdbx32.exe).
2. In the **Data Source** menu, select the data source.
3. Then click **System > ODBC Information**.
4. Read the "DBMS Name".

You can edit the Odwdbex.ini file by selecting **System > Edit Odwdbex.ini** in the **HOPEX** extraction tool. Check that the file is archived.

For more the format of queries, see [Select Clause Formats](#).

Using ODBC standard APIs

To force the use of ODBC APIs:

1. Edit the Odwdbex.ini file.
2. At the level of each concept concerned, modify the extraction strategy using the keyword: USE_DRIVER_ODBC.

Example in the ODWDBEX.INI file:

```
[<DBMS Name>]
INDEXES=USE_DRIVER_ODBC
```

Select Clause Formats



It is important to use the indicated syntax and in particular not to omit any of the "1"s. Note that the clauses must fit on a single line in the ODWDBEX.INI file.

Primary Keys

```
SELECT
1,
TABLE_OWNER,
TABLE_NAME,
COLUMN_NAME,
KEY_SEQUENCE,
PK_NAME
FROM ...
WHERE ...
```

- TABLE_OWNER: owner of the primary key table
- TABLE_NAME: name of the primary key table
- COLUMN_NAME: name of the primary key column
- KEY_SEQUENCE: Number of the column in the key sequence (starts at 1)
- PK_NAME: Name of the primary key; "1" if this name is not supported by the DBMS.

Foreign Keys

```
SELECT
1,
PKTABLE_OWNER,
PKTABLE_NAME,
1,
1,
FKTABLE_OWNER,
FKTABLE_NAME,
FKCOLUMN_NAME,
KEY_SEQ,
UPDATE_RULE,
DELETE_RULE,
FK_NAME
FROM ...
WHERE ...
```

- PKTABLE_OWNER: name of the owner of the primary key table (reference table)
- PKTABLE_NAME: name of the primary key table
- FKTABLE_OWNER: name of the owner of the foreign key table
- FKTABLE_NAME: name of the foreign key table
- FKCOLUMN_NAME: name of the foreign key column
- KEY_SEQ: number of the column in the key sequence (starts at 1)
- UPDATE_RULE : R: Restrict, C: Cascade
- DELETE_RULE : R: Restrict, C: Cascade
- FK_NAME: name of the foreign key; "1" if this name is not supported by the DBMS.

Indexes

```
SELECT
1,
TABLE_OWNER,
TABLE_NAME,
NON_UNIQUE,
1,
INDEX_NAME,
TYPE,
SEQ_IN_INDEX,
COLUMN_NAME,
COLLATION
FROM ...
WHERE ...
```

- TABLE_OWNER: name of the owner of the table concerned by the statistic or the index
- TABLE_NAME: name of the index table
- NON_UNIQUE: the indexes must have a unique value
- INDEX_NAME: name of the index
- TYPE : Index type
- SEQ_IN_INDEX: number of the column in the key sequence (starts at 1)
- COLUMN_NAME: name of the column
- COLLATION: column sort; "A" increasing order, "D" decreasing order

Columns

```

SELECT
1,
COLUMN_OWNER,
TABLE_NAME,
COLUMN_NAME,
DataType ODBC,
DataType Name,
Detail,
Length,
Scale,
1,
NULLABLE,
COMMENT,
DEFAULT_VALUE,
1,
1,
1,
Order
WHERE [Joint between <MEGA:OWNER><MEGA:OBJECT_NAME>]

```

- <MEGA:OWNER> is replaced by the user, the Schema or "".
- <MEGA:OBJECT_NAME>] is replaced by the name of the table.
- COLUMN_OWNER: name of the column, string with 128 characters.
- TABLE_NAME: name of the table, string with 128 characters.
- DataType ODBC: data type in the form of an integer. This value is the value of ODBC data types therefore comprised of the following:

```

# -1 (SQL_LONGVARCHAR)
# -2 (SQL_BINARY
# -3 (SQL_VARBINARY)
# -4 (SQL_LONGVARBINARY)
# -5 (SQL_BIGINT)
# -6 (SQL_TINYINT)
# -7 (SQL_BIT)
# 0 (SQL_UNKNOWN_TYPE)
# 1 (SQL_CHAR)
# 2 (SQL_NUMERIC)
# 3 (SQL_DECIMAL)
# 4 (SQL_INTEGER)
# 5 (SQL_SMALLINT)
# 6 (SQL_FLOAT)

```

```
# 7 (SQL_REAL)
# 8 (SQL_DOUBLE)
# 9 (SQL_DATE)
# 10 (SQL_TIME)
# 11 (SQL_TIMESTAMP)
# 12 (SQL_VARCHAR)
```

- **DataType Name:** name of the data type, string of 128 characters. It is built as follows: "SQL_<DbmsName><String>"
- **Precision:** length in MEGA if "Length" is empty.
- **Length:** length in MEGA if greater than 0.
- **Scale:** integer
- **NULLABLE:** integer specifying if the column can be NULL. ODBC values possible: 0 (SQL_NO_NULLS), 1 (SQL_NULLABLE) or 3 (SQL_NULL_WITH_DEFAULT).
- **COMMENT:** column comments, string with 1257 characters.
- **DEFAULT_VALUE:** default value of the column, string with 1257 characters.

PIVOT TYPES AND DATATYPES CORRESPONDENCE TABLES



The following tables show correspondence between pivot types and the different supported DBMSs and their versions.

- ✓ [DB2 OS/390 Version 5](#)
- ✓ [DB2 OS/390 Version 7](#)
- ✓ [DB2 for z/OS Version 8](#)
- ✓ [DB2 Universal Database Version 5](#)
- ✓ [DB2 Universal Database Version 7](#)
- ✓ [DB2 Universal Database Version 8](#)
- ✓ [DB2 Universal Database Version 9](#)
- ✓ [DB2 Version 9 For OS](#)
- ✓ [Informix Dynamic Server 7.3](#)
- ✓ [Ingres II 2.0](#)
- ✓ [MySQL 4.1](#)
- ✓ [MySQL 5.0](#)
- ✓ [Oracle 10](#)
- ✓ [Oracle 11](#)
- ✓ [Oracle 8](#)
- ✓ [Oracle 9i](#)
- ✓ [PostgreSQL9.3](#)
- ✓ [SQL ANSI/ISO 9075:1992](#)
- ✓ [SQL Server 2000](#)
- ✓ [SQL Server 2005](#)
- ✓ [SQL Server 2008](#)
- ✓ [SQL Server 7](#)
- ✓ [Sybase Adaptive Server 11](#)
- ✓ [Sybase Adaptive Server 12.5](#)
- ✓ [Teradata Database](#)

DB2 OS/390 VERSION 5

Pivot --> Datatype (DB2 OS/390 Version 5)

Pivot	Condition	Datatype
P-AutoIdentifier		INTEGER
P-Binary		CHAR(@L) FOR BIT DATA
P-Boolean		CHAR(@L) FOR BIT DATA
P-Byte		CHAR (1) FOR BIT DATA
P-Character	Not Unicode and (L=0 or L ø)	CHAR
	Not Unicode and 0<L<256	CHAR(@L)
	Not Unicode and (L=256 or L ø)	GRAPHIC(@L)
	Not Unicode and L>255	VARCHAR(@L)
	Not Unicode and L>255	VARGRAPHIC(@L)
P-Currency		DECIMAL(@L,@D)
P-Date		DATE
P-Datetime		TIMESTAMP
P-Decimal	L=0 or L ø	DECIMAL
	L>0 and D ø	DECIMAL(@L)
	L>0 and D not ø	DECIMAL(@L,@D)
P-Double		DOUBLE
P-Float	L=0 or L ø	FLOAT
	L<>0	FLOAT(@L)
P-Integer		INTEGER
P-Long Integer		INTEGER
P-Long Real		REAL

Pivot --> Datatype (DB2 OS/390 Version 5)

Pivot	Condition	Datatype
P-Multimedia		LONG VARCHAR FOR BIT DATA
P-Numeric	L not $\emptyset$ and D not $\emptyset$	DECIMAL(@L,@D)
	L>9 and D $\emptyset$	FLOAT(@L)
	4<L<10 and D $\emptyset$	INTEGER
	L=5 or L $\emptyset$	SMALLINT
P-Real		REAL
P-Smallint		SMALLINT
P-String		LONG VARCHAR
P-Text		LONG VARCHAR
P-Time		TIME
P-Timestamp		TIMESTAMP
P-Tinyint		SMALLINT
P-Varbinary		VARCHAR(@L) FOR BIT DATA
P-Varchar	Not Unicode and (L=0 or L $\emptyset$)	VARCHAR
	Not Unicode and L<>0	VARCHAR(@L)
	Unicode	VARGRAPHIC(@L)

Datatype --> Pivot (DB2 OS/390 Version 5)

Datatype	Condition	Pivot
CHAR		P-Character
CHAR (1) FOR BIT DATA		P-Byte
CHAR(L)		P-Character
CHAR(L) FOR BIT DATA		P-Boolean
DATE		P-Date
DECIMAL		P-Decimal

Datatype --> Pivot (DB2 OS/390 Version 5)

Datatype	Condition	Pivot
DECIMAL(L)		P-Decimal
DECIMAL(L,D)		P-Decimal
DOUBLE		P-Double
FLOAT		P-Float
FLOAT(L)		P-Float
INTEGER		P-Integer
LONG VARCHAR		P-Text
LONG VARCHAR FOR BIT DATA		P-Multimedia
REAL		P-Real
SMALLINT		P-Smallint
TIME		P-Time
TIMESTAMP		P-Timestamp
VARCHAR		P-Varchar
VARCHAR(L)		P-Varchar
VARCHAR(L) FOR BIT DATA		P-Varbinary

DB2 OS/390 VERSION 7

Pivot --> Datatype (DB2 OS/390 Version 7)

Pivot	Condition	Datatype
P-AutoIdentifier		INTEGER
P-Binary		CHAR(@L) FOR BIT DATA
P-Boolean		CHAR(@L) FOR BIT DATA
P-Byte		CHAR (1) FOR BIT DATA
P-Character	Not Unicode and (L=0 or L ø)	CHAR
	Not Unicode and 0<L<256	CHAR(@L)
	Not Unicode and (L=256 or L ø)	GRAPHIC(@L)
	Not Unicode and L>255	VARCHAR(@L)
	Not Unicode and L>255	VARGRAPHIC(@L)
P-Currency		DECIMAL(@L, @D)
P-Date		DATE
P-Datetime		TIMESTAMP
P-Decimal	L=0 or L ø	DECIMAL
	L>0 and D ø	DECIMAL(@L)
	L>0 and D not ø	DECIMAL(@L, @D)
P-Double		DOUBLE
P-Float	L=0 or L ø	FLOAT
	L<>0	FLOAT(@L)
P-Integer		INTEGER
P-Long Integer		INTEGER
P-Long Real		REAL

Pivot --> Datatype (DB2 OS/390 Version 7)

Pivot	Condition	Datatype
P-Multimedia		LONG VARCHAR FOR BIT DATA
P-Numeric	L not ø and D not ø	DECIMAL(@L, @D)
	L>9 and D ø	FLOAT(@L)
	4<L<10 and D ø	INTEGER
	(L<5 and D ø) or L ø	SMALLINT
P-Real		REAL
P-Smallint		SMALLINT
P-String		LONG VARCHAR
P-Text		LONG VARCHAR
P-Time		TIME
P-Timestamp		TIMESTAMP
P-Tinyint		SMALLINT
P-Varbinary		VARCHAR(@L) FOR BIT DATA
P-Varchar	Not Unicode and (L=0 or L ø)	VARCHAR
	Not Unicode and L<>0	VARCHAR(@L)
	Unicode	VARGRAPHIC(@L)

Datatype --> Pivot (DB2 OS/390 Version 7)

Datatype	Condition	Pivot
CHAR		P-Character
CHAR (1) FOR BIT DATA		P-Byte
CHAR(L)		P-Character
CHAR(L) FOR BIT DATA		P-Boolean
DATE		P-Date
DECIMAL		P-Decimal

Datatype --> Pivot (DB2 OS/390 Version 7)

Datatype	Condition	Pivot
DECIMAL(L)		P-Decimal
DECIMAL(L, D)		P-Decimal
DOUBLE		P-Double
FLOAT		P-Float
FLOAT(L)		P-Float
INTEGER		P-Integer
LONG VARCHAR		P-Text
LONG VARCHAR FOR BIT DATA		P-Multimedia
REAL		P-Real
SMALLINT		P-Smallint
TIME		P-Time
TIMESTAMP		P-Datetime
VARCHAR(L)		P-Varchar
VARCHAR(L) FOR BIT DATA		P-Varbinary

DB2 FOR Z/OS VERSION 8

Pivot --> Datatype (DB2 for z/OS Version 8)

Pivot	Condition	Datatype
P-AutoIdentifier		INTEGER
P-Binary		CHAR(@L) FOR BIT DATA
P-Boolean		CHAR(@L) FOR BIT DATA
P-Byte		CHAR (1) FOR BIT DATA
P-Character	Not Unicode and (L=0 or L ø)	CHAR
	Not Unicode and 0<L<256	CHAR(@L)
	Not Unicode and (L=256 or L ø)	GRAPHIC(@L)
	Not Unicode and L>255	VARCHAR(@L)
	Not Unicode and L>255	VARGRAPHIC(@L)
P-Currency		DECIMAL(@L, @D)
P-Date		DATE
P-Datetime		TIMESTAMP
P-Decimal	L=0 or L ø	DECIMAL
	L>0 and D ø	DECIMAL(@L)
	L>0 and D not ø	DECIMAL(@L, @D)
P-Double		DOUBLE
P-Float	L=0 or L ø	FLOAT
	L<>0	FLOAT(@L)
P-Integer		INTEGER
P-Long Integer		INTEGER
P-Long Real		REAL

Pivot --> Datatype (DB2 for z/OS Version 8)

Pivot	Condition	Datatype
P-Multimedia		LONG VARCHAR FOR BIT DATA
P-Numeric	L not $\emptyset$ and D not $\emptyset$	DECIMAL(@L, @D)
	L>9 and D $\emptyset$	FLOAT(@L)
	4<L<10 and D $\emptyset$	INTEGER
	(L<5 and D $\emptyset$) or L $\emptyset$	SMALLINT
P-Real		REAL
P-Smallint		SMALLINT
P-String		LONG VARCHAR
P-Text		LONG VARCHAR
P-Time		TIME
P-Timestamp		TIMESTAMP
P-Tinyint		SMALLINT
P-Varbinary		VARCHAR(@L) FOR BIT DATA
P-Varchar	Not Unicode and (L=0 or L $\emptyset$)	VARCHAR
	Not Unicode and L<>0	VARCHAR(@L)
	Unicode	VARGRAPHIC(@L)

Datatype --> Pivot (DB2 for z/OS Version 8)

Datatype	Condition	Pivot
CHAR		P-Character
CHAR (1) FOR BIT DATA		P-Byte
CHAR(L)		P-Character
CHAR(L) FOR BIT DATA		P-Boolean
DATE		P-Date
DECIMAL		P-Decimal

Datatype --> Pivot (DB2 for z/OS Version 8)

Datatype	Condition	Pivot
DECIMAL(L)		P-Decimal
DECIMAL(L, D)		P-Decimal
DOUBLE		P-Double
FLOAT		P-Float
FLOAT(L)		P-Float
INTEGER		P-Integer
LONG VARCHAR		P-Text
LONG VARCHAR FOR BIT DATA		P-Multimedia
REAL		P-Real
SMALLINT		P-Smallint
TIME		P-Time
TIMESTAMP		P-Datetime
VARCHAR(L)		P-Varchar
VARCHAR(L) FOR BIT DATA		P-Varbinary

DB2 UNIVERSAL DATABASE VERSION 5

Pivot --> Datatype (DB2 Universal Database Version 5)

Pivot	Condition	Datatype
P-AutoIdentifier		INTEGER
P-Binary		CHAR(@L) FOR BIT DATA
P-Boolean		CHAR(@L) FOR BIT DATA
P-Byte		CHAR (1) FOR BIT DATA
P-Character	Not Unicode and (L=0 or L ø)	CHAR
	Not Unicode and 0<L<255	CHAR(@L)
	Not Unicode and (L=256 or L ø)	GRAPHIC(@L)
	Not Unicode and L>254	VARCHAR(@L)
	Not Unicode and L>255	VARGRAPHIC(@L)
P-Currency		DECIMAL(@L,@D)
P-Date		DATE
P-Datetime		TIMESTAMP
P-Decimal	L=0 or L ø	DECIMAL
	L>0 and D ø	DECIMAL(@L)
	L>0 and D not ø	DECIMAL(@L,@D)
P-Double		DOUBLE
P-Float	L=0 or L ø	FLOAT
	L<>0	FLOAT(@L)
P-Integer		INTEGER
P-Long Integer		BIGINT
P-Long Real		REAL

Pivot --> Datatype (DB2 Universal Database Version 5)

Pivot	Condition	Datatype
P-Multimedia		BLOB(@L)
P-Numeric	L not ø and D not ø	DECIMAL(@L,@D)
	L>9 and D ø	FLOAT(@L)
	4<L<10 and D ø	INTEGER
	(L<5 and D not ø) or L ø	SMALLINT
P-Real		REAL
P-Smallint		SMALLINT
P-String		LONG VARCHAR
P-Text		LONG VARCHAR
P-Time		TIME
P-Timestamp		TIMESTAMP
P-Tinyint		SMALLINT
P-Varbinary		VARCHAR(@L) FOR BIT DATA
P-Varchar	Not Unicode and (L=0 or L ø)	VARCHAR
	Not Unicode and L<>0	VARCHAR(@L)
	Unicode	VARGRAPHIC(@L)

Datatype --> Pivot (DB2 Universal Database Version 5)

Datatype	Condition	Pivot
BIGINT		P-Long Integer
BLOB(L)		P-Multimedia
CHAR		P-Character
CHAR (1) FOR BIT DATA		P-Byte
CHAR(L)		P-Character
CHAR(L) FOR BIT DATA		P-Boolean
DATE		P-Date

Datatype --> Pivot (DB2 Universal Database Version 5)

Datatype	Condition	Pivot
DECIMAL		P-Decimal
DECIMAL(L)		P-Decimal
DECIMAL(L,D)		P-Decimal
DOUBLE		P-Double
FLOAT		P-Float
FLOAT(L)		P-Float
INTEGER		P-Integer
LONG VARCHAR		P-Text
REAL		P-Real
SMALLINT		P-Smallint
TIME		P-Time
TIMESTAMP		P-Timestamp
VARCHAR		P-Varchar
VARCHAR(L)		P-Varchar
VARCHAR(L) FOR BIT DATA		P-Varbinary

DB2 UNIVERSAL DATABASE VERSION 7

Pivot --> Datatype (DB2 Universal Database Version 7)

Pivot	Condition	Datatype
P-AutoIdentifier		INTEGER
P-Binary		CHAR(@L) FOR BIT DATA
P-Boolean		CHAR(@L) FOR BIT DATA
P-Byte		CHAR (1) FOR BIT DATA
P-Character	Not Unicode and (L=0 or L ø)	CHAR
	Not Unicode and 0<L<255	CHAR(@L)
	Not Unicode and (L=256 or L ø)	GRAPHIC(@L)
	Not Unicode and L>254	VARCHAR(@L)
	Not Unicode and L>255	VARGRAPHIC(@L)
P-Currency		DECIMAL(@L,@D)
P-Date		DATE
P-Datetime		TIMESTAMP
P-Decimal	L=0 or L ø	DECIMAL
	L>0 and D ø	DECIMAL(@L)
	L>0 and L not ø	DECIMAL(@L,@D)
P-Double		DOUBLE
P-Float	L=0 or L ø	FLOAT
	L<>0	FLOAT(@L)
P-Integer		INTEGER
P-Long Integer		BIGINT
P-Long Real		REAL

Pivot --> Datatype (DB2 Universal Database Version 7)

Pivot	Condition	Datatype
P-Multimedia		BLOB(@L)
P-Numeric	L not ø and D not ø	DECIMAL(@L,@D)
	L>9 and D ø	FLOAT(@L)
	4<L<10 and D ø	INTEGER
	L=5 or L ø	SMALLINT
P-Real		REAL
P-Smallint		SMALLINT
P-String		LONG VARCHAR
P-Text		LONG VARCHAR
P-Time		TIME
P-Timestamp		TIMESTAMP
P-Tinyint		SMALLINT
P-Varbinary		VARCHAR(@L) FOR BIT DATA
P-Varchar	Not Unicode and (L=0 or L ø)	VARCHAR
	Not Unicode and L<>0	VARCHAR(@L)
	Unicode	VARGRAPHIC(@L)

Datatype --> Pivot (DB2 Universal Database Version 7)

Datatype	Condition	Pivot
BIGINT		P-Long Integer
BLOB(L)		P-Multimedia
CHAR		P-Character
CHAR (1) FOR BIT DATA		P-Byte
CHAR(L)		P-Character
CHAR(L) FOR BIT DATA		P-Boolean
DATE		P-Date

Datatype --> Pivot (DB2 Universal Database Version 7)

Datatype	Condition	Pivot
DECIMAL		P-Decimal
DECIMAL(L)		P-Decimal
DECIMAL(L,D)		P-Decimal
DOUBLE		P-Double
FLOAT		P-Float
FLOAT(L)		P-Float
INTEGER		P-Integer
LONG VARCHAR		P-Text
REAL		P-Real
SMALLINT		P-Smallint
TIME		P-Time
TIMESTAMP		P-Timestamp
VARCHAR		P-Varchar
VARCHAR(L)		P-Varchar
VARCHAR(L) FOR BIT DATA		P-Varbinary

DB2 UNIVERSAL DATABASE VERSION 8

Pivot --> Datatype (DB2 Universal Database Version 8)

Pivot	Condition	Datatype
P-AutoIdentifier		INTEGER
P-Byte		CHAR (1) FOR BIT DATA
P-Character	Not Unicode and (L=0 or L ø)	CHAR
	Not Unicode and 0<L<255	CHAR(@L)
	Not Unicode and (L=256 or L ø)	GRAPHIC(@L)
	Not Unicode and L>254	VARCHAR(@L)
	Not Unicode and L>255	VARGRAPHIC(@L)
P-Currency		DECIMAL(@L,@D)
P-Date		DATE
P-Datetime		TIMESTAMP
P-Decimal	L=0 or L ø	DECIMAL
	L>0 and D ø	DECIMAL(@L)
	L>0 and L not ø	DECIMAL(@L,@D)
P-Double		DOUBLE
P-Float	L=0 or L ø	FLOAT
	L<>0	FLOAT(@L)
P-Integer		INTEGER
P-Long Integer		BIGINT
P-Long Real		REAL

Pivot --> Datatype (DB2 Universal Database Version 8)

Pivot	Condition	Datatype
P-Numeric	L not ø and D not ø	DECIMAL(@L,@D)
	L>9 and D ø	FLOAT(@L)
	4<L<10 and D ø	INTEGER
	L=5 or L ø	SMALLINT
P-Real		REAL
P-Smallint		SMALLINT
P-String		LONG VARCHAR
P-Text		LONG VARCHAR
P-Time		TIME
P-Timestamp		TIMESTAMP
P-Tinyint		SMALLINT
P-Varbinary		VARCHAR(@L) FOR BIT DATA
P-Varchar	Not Unicode and (L=0 or L ø)	VARCHAR
	Not Unicode and L<>0	VARCHAR(@L)
	Unicode	VARGRAPHIC(@L)

Datatype --> Pivot (DB2 Universal Database Version 8)

Datatype	Condition	Pivot
BIGINT		P-Long Integer
CHAR		P-Character
CHAR (1) FOR BIT DATA		P-Byte
CHAR(L)		P-Character
DATE		P-Date
DECIMAL		P-Decimal
DECIMAL(L)		P-Decimal
DECIMAL(L,D)		P-Decimal

Datatype --> Pivot (DB2 Universal Database Version 8)

Datatype	Condition	Pivot
DOUBLE		P-Double
FLOAT		P-Float
FLOAT(L)		P-Float
INTEGER		P-Integer
LONG VARCHAR		P-Text
REAL		P-Real
SMALLINT		P-Smallint
SMALLINT		P-Tinyint
TIME		P-Time
TIMESTAMP		P-Timestamp
VARCHAR		P-Varchar
VARCHAR(L)		P-Varchar
VARCHAR(L) FOR BIT DATA		P-Varbinary

DB2 UNIVERSAL DATABASE VERSION 9

Pivot --> Datatype (DB2 Universal Database Version 9)

Pivot	Condition	Datatype
P-AutoIdentifier		INTEGER
P-Binary		CHAR(@L) FOR BIT DATA
P-Boolean		CHAR(@L) FOR BIT DATA
P-Byte		CHAR (1) FOR BIT DATA
P-Character	Not Unicode and (L=0 or L ø)	CHAR
	Not Unicode and 0<L<255	CHAR(@L)
	Not Unicode and (L=256 or L ø)	GRAPHIC(@L)
	Not Unicode and L>254	VARCHAR(@L)
	Not Unicode and L>255	VARGRAPHIC(@L)
P-Currency		DECIMAL(@L,@D)
P-Date		DATE
P-Datetime		TIMESTAMP
P-Decimal	L=0 or L ø	DECIMAL
	L>0 and D ø	DECIMAL(@L)
	L>0 and L not ø	DECIMAL(@L,@D)
P-Double		DOUBLE
P-Float	L=0 or L ø	FLOAT
	L<>0	FLOAT(@L)
P-Integer		INTEGER
P-Long Integer		BIGINT
P-Long Real		REAL

Pivot --> Datatype (DB2 Universal Database Version 9)

Pivot	Condition	Datatype
P-Multimedia		BLOB(@L)
P-Numeric	L not ø and D not ø	DECIMAL(@L,@D)
	L>9 and D ø	FLOAT(@L)
	4<L<10 and D ø	INTEGER
	L=5 or L ø	SMALLINT
P-Real		REAL
P-Smallint		SMALLINT
P-String		LONG VARCHAR
P-Text		LONG VARCHAR
P-Time		TIME
P-Timestamp		TIMESTAMP
P-Tinyint		SMALLINT
P-Varbinary		VARCHAR(@L) FOR BIT DATA
P-Varchar	Not Unicode and (L=0 or L ø)	VARCHAR
	Not Unicode and L<>0	VARCHAR(@L)
	Unicode	VARGRAPHIC(@L)

Datatype --> Pivot (DB2 Universal Database Version 9)

Datatype	Condition	Pivot
BIGINT		P-Long Integer
BLOB(L)		P-Multimedia
CHAR		P-Character
CHAR (1) FOR BIT DATA		P-Byte
CHAR(L)		P-Character
CHAR(L) FOR BIT DATA		P-Boolean
CLOB(L)		CLOB

Datatype --> Pivot (DB2 Universal Database Version 9)

Datatype	Condition	Pivot
DATE		P-Date
DECIMAL		P-Decimal
DECIMAL(L)		P-Decimal
DECIMAL(L,D)		P-Decimal
DOUBLE		P-Double
FLOAT		P-Float
FLOAT(L)		P-Float
INTEGER		P-Integer
LONG VARCHAR		P-Text
REAL		P-Real
SMALLINT		P-Smallint
SMALLINT		P-Tinyint
TIME		P-Time
TIMESTAMP		P-Timestamp
VARCHAR		P-Varchar
VARCHAR(L)		P-Varchar
VARCHAR(L) FOR BIT DATA		P-Varbinary

DB2 VERSION 9 FOR OS

Pivot --> Datatype (DB2 Version 9 For OS)

Pivot	Condition	Datatype
P-AutoIdentifier		INTEGER
P-Binary		CHAR(@L) FOR BIT DATA
P-Boolean		CHAR(@L) FOR BIT DATA
P-Byte		CHAR (1) FOR BIT DATA
P-Character	Not Unicode and (L=0 or L ø)	CHAR
	Not Unicode and 0<L<256	CHAR(@L)
	Not Unicode and (L=256 or L ø)	GRAPHIC(@L)
	Not Unicode and L>255	VARCHAR(@L)
	Not Unicode and L>255	VARGRAPHIC(@L)
P-Currency		DECIMAL(@L, @D)
P-Date		DATE
P-Datetime		TIMESTAMP
P-Decimal	L=0 or L ø	DECIMAL
	L>0 and D ø	DECIMAL(@L)
	L>0 and D not ø	DECIMAL(@L, @D)
P-Double		DOUBLE
P-Float	L=0 or L ø	FLOAT
	L<>0	FLOAT(@L)
P-Integer		INTEGER
P-Long Integer		INTEGER
P-Long Real		REAL

Pivot --> Datatype (DB2 Version 9 For OS)

Pivot	Condition	Datatype
P-Multimedia		BLOB
		BLOB(@L)
		CLOB
		CLOB FOR MIXED DATA
		CLOB(@L)
		CLOB(@L) FOR MIXED DATA
		LONG VARCHAR FOR BIT DATA
P-Numeric	L not ø and D not ø	DECIMAL(@L, @D)
	L>9 and D ø	FLOAT(@L)
	4<L<10 and D ø	INTEGER
	(L<5 and D ø) or L ø	SMALLINT
P-Real		REAL
P-Smallint		SMALLINT
P-String		LONG VARCHAR
P-Text		LONG VARCHAR
P-Time		TIME
P-Timestamp		TIMESTAMP
P-Tinyint		SMALLINT
P-Varbinary	L <=1024	VARCHAR(@L) FOR BIT DATA
	L>1024	XML
P-Varchar	Not Unicode and (L=0 or L ø)	VARCHAR
	Not Unicode and L<>0	VARCHAR(@L)
	Unicode	VARGRAPHIC(@L)

Datatype --> Pivot (DB2 Version 9 For OS)

Datatype	Condition	Pivot
BLOB		P-Multimedia
BLOB(L)		P-Multimedia
CHAR		P-Character
CHAR (1) FOR BIT DATA		P-Byte
CHAR(L)		P-Character
CHAR(L) FOR BIT DATA		P-Boolean
CLOB		P-Multimedia
CLOB FOR MIXED DATA		P-Multimedia
CLOB(L)		P-Multimedia
CLOB(L) FOR MIXED DATA		P-Multimedia
DATE		P-Date
DECIMAL		P-Decimal
DECIMAL(L)		P-Decimal
DECIMAL(L, D)		P-Decimal
DOUBLE		P-Double
FLOAT		P-Float
FLOAT(L)		P-Float
INTEGER		P-Integer
LONG VARCHAR		P-Text
LONG VARCHAR FOR BIT DATA		P-Multimedia
REAL		P-Real
SMALLINT		P-Smallint
TIME		P-Time
TIMESTAMP		P-Datetime
VARCHAR(L)		P-Varchar

Datatype --> Pivot (DB2 Version 9 For OS)

Datatype	Condition	Pivot
VARCHAR(L) FOR BIT DATA		P-Varbinary
XML		P-Varbinary

INFORMIX DYNAMIC SERVER 7.3

Pivot --> Datatype (Informix Dynamic Server 7.3)

Pivot	Condition	Datatype
P-AutoIdentifier		SERIAL
P-Binary		BYTE
P-Boolean		SMALLINT
P-Byte		BYTE
P-Character		CHAR(@L)
P-Currency		MONEY(@L,@D)
P-Date		DATE
P-Datetime		DATETIME
P-Decimal	D ø	DECIMAL(@L)
	D not ø	DECIMAL(@L,@D)
P-Double		FLOAT
P-Float	L=0 or L ø	FLOAT
	L > 0	SMALLFLOAT(@L)
P-Integer		INTEGER
P-Long Integer		INTEGER
P-Long Real		FLOAT
P-Multimedia		BYTE
P-Numeric	L > 9	DECIMAL(@L)
	4 < L < 10	INTEGER
	L = 5 or L ø	SMALLINT
P-Real		FLOAT
P-Smallint		SMALLINT
P-String		VARCHAR(@L)
P-Text		VARCHAR(@L)

Pivot --> Datatype (Informix Dynamic Server 7.3)

Pivot	Condition	Datatype
P-Time		DATETIME
P-Timestamp		SERIAL
P-Tinyint		SMALLINT
P-Varchar	D ø	VARCHAR(@L)
	D not ø	VARCHAR(@L,@D)

Datatype --> Pivot (Informix Dynamic Server 7.3)

Datatype	Condition	Pivot
BYTE		P-Multimedia
CHAR(L)		P-Character
DATE		P-Date
DATETIME		P-Datetime
DECIMAL(L)		P-Decimal
DECIMAL(L,D)		P-Decimal
FLOAT		P-Float
INTEGER		P-Integer
MONEY(L,D)		P-Currency
SERIAL		P-Timestamp
SMALLFLOAT(L)		P-Float
SMALLINT		P-Boolean
VARCHAR(L)		P-Varchar
VARCHAR(L,D)		P-Varchar

INGRES II 2.0

Pivot --> Datatype (Ingres II 2.0)

Pivot	Condition	Datatype
P-AutoIdentifier		integer
P-Binary		long byte(@L)
P-Boolean		integer1
P-Byte		byte(@L)
P-Character		char(@L)
P-Currency		money(@L,@D)
P-Date		date
P-Datetime		date
P-Decimal		decimal(@L,@D)
P-Double		integer1
P-Float		float4
P-Integer		integer
P-Long Integer		integer
P-Long Real		float
P-Multimedia	L=2001 or L ø	byte(@L)
	L>2000	long byte(@L)
P-Numeric	L>9	float
	4<L<10	integer
	L=3 or L ø	integer1
	2<L<5	smallint
P-Real		float
P-Smallint		smallint
P-String		long varchar(@L)
P-Text		text(@L)

Pivot --> Datatype (Ingres II 2.0)

Pivot	Condition	Datatype
P-Time		date
P-Timestamp		date
P-Tinyint		integer1
P-Varbinary		byte varying(@L)
P-Varchar		varchar(@L)

Datatype --> Pivot (Ingres II 2.0)

Datatype	Condition	Pivot
byte varying(L)		P-Varbinary
byte(L)		P-Multimedia
char(L)		P-Character
date		P-Datetime
decimal(L,D)		P-Decimal
float		P-Real
float4		P-Float
integer		P-Integer
integer1		P-Tinyint
long byte(L)		P-Binary
long varchar(L)		P-String
money(L,D)		P-Currency
smallint		P-Smallint
text(L)		P-Text
varchar(L)		P-Varchar

MySQL 4.1

Pivot --> Datatype (MySQL 4.1)

Pivot	Condition	Datatype
	$L > 0$ and D not $\emptyset$	REAL (@L,@D) UNSIGNED
	$L = 0$ or L $\emptyset$	REAL UNSIGNED
	$L > 0$ and D not $\emptyset$	REAL (@L,@D) UNSIGNED ZEROFILL
	$L = 0$ or L $\emptyset$	REAL UNSIGNED ZEROFILL
P-AutoIdentifier		INTEGER
P-Binary	L $\emptyset$ or $L \leq 0$	BINARY
	L is numeric and $L \neq 0$	BINARY (@L)
P-Boolean		BOOLEAN
P-Byte	L $\emptyset$ or $L \leq 0$	BIT
	L is numeric and $L \neq 0$	BIT (@L)
P-Character	Not Unicode and $L = 0$	CHAR
	Not Unicode and $L \neq 0$	CHAR (@L)
	Unicode and $L \neq 0$	CHAR (@L) UNICODE
	Unicode and $L = 0$	CHAR UNICODE
P-Character Ascii	L $\emptyset$ or $L \leq 0$	CHAR ASCII
	L is numeric and $L \neq 0$	CHAR(@L) ASCII
P-Character Binary	Not Unicode and $L \neq 0$	CHAR (@L) BINARY
	Unicode and $L \neq 0$	CHAR (@L) UNICODE BINARY
	Not Unicode and $L = 0$	CHAR BINARY
	Unicode and $L = 0$	CHAR UNICODE BINARY

Pivot --> Datatype (MySQL 4.1)

Pivot	Condition	Datatype
P-Character Unicode	L is numeric and $L > 0$	CHAR (@L) UNICODE
	L $\emptyset$ or $L \leq 0$	CHAR UNICODE
P-Character Unicode Binary	L is numeric and $L > 0$	CHAR (@L) UNICODE BINARY
	L $\emptyset$ or $L \leq 0$	CHAR UNICODE BINARY
P-Currency		DECIMAL(@L,@D)
P-Date		DATE
P-Datetime		DATETIME
P-Decimal	$L = 0$ or L $\emptyset$	DECIMAL
	$L > 0$ and D $\emptyset$	DECIMAL (@L)
	$L > 0$ and D not $\emptyset$	DECIMAL(@L,@D)
P-Decimal Unsigned	$L > 0$ and D $\emptyset$	DECIMAL (@L) UNSIGNED
	$L > 0$ and D not $\emptyset$	DECIMAL (@L,@D) UNSIGNED
	$L = 0$ or L $\emptyset$	DECIMAL UNSIGNED
P-Decimal Unsigned Zero-fill	$L > 0$ and D $\emptyset$	DECIMAL (@L) UNSIGNED ZEROFILL
	$L > 0$ and D not $\emptyset$	DECIMAL (@L,@D) UNSIGNED ZEROFILL
	$L = 0$ or L $\emptyset$	DECIMAL UNSIGNED ZEROFILL
P-Double	L $\emptyset$ or $L \leq 0$	DOUBLE PRECISION
	$L > 0$ or $D > 0$	DOUBLE PRECISION (@L,@D)
P-Double Unsigned	$L > 0$ and D not $\emptyset$	DOUBLE PRECISION (@L,@D) UNSIGNED
	$L = 0$ or L $\emptyset$	DOUBLE PRECISION UNSIGNED

Pivot --> Datatype (MySQL 4.1)

Pivot	Condition	Datatype
P-Double Unsigned Zero-fill	$L > 0$ and $D \neq \emptyset$	DOUBLE PRECISION (@L,@D) UNSIGNED ZEROFILL
	$L = 0$ or $L \neq \emptyset$	DOUBLE PRECISION UNSIGNED ZEROFILL
P-Float	$L = 0$ or $L \neq \emptyset$	FLOAT
	$L > 0$ and $D \neq \emptyset$	FLOAT (@L)
	$L > 0$ and $D \neq \emptyset$	FLOAT (@L,@D)
P-Float Unsigned	$L > 0$ and $D \neq \emptyset$	FLOAT (@L) UNSIGNED
	$L > 0$ and $D \neq \emptyset$	FLOAT (@L,@D) UNSIGNED
	$L = 0$ or $L \neq \emptyset$	FLOAT UNSIGNED
P-Float Unsigned Zerofill	$L > 0$ and $D \neq \emptyset$	FLOAT (@L) UNSIGNED ZEROFILL
	$L > 0$ and $D \neq \emptyset$	FLOAT (@L,@D) UNSIGNED ZEROFILL
	$L = 0$ or $L \neq \emptyset$	FLOAT UNSIGNED ZEROFILL
P-Integer	$L \neq \emptyset$ or $L \leq 0$	INTEGER
	L is numeric and $L \neq 0$	INTEGER (@L)
P-Integer Unsigned	L is numeric and $L \neq 0$	INTEGER (@L) UNSIGNED
	$L \neq \emptyset$ or $L \leq 0$	INTEGER UNSIGNED
P-Integer Unsigned Zerofill	L is numeric and $L \neq 0$	INTEGER (@L) UNSIGNED ZEROFILL
	$L \neq \emptyset$ or $L \leq 0$	INTEGER UNSIGNED ZEROFILL
P-Long Integer	$L \neq \emptyset$ or $L \leq 0$	BIGINT
	L is numeric and $L \neq 0$	BIGINT (@L)

Pivot --> Datatype (MySQL 4.1)

Pivot	Condition	Datatype
P-Long Integer Unsigned	L is numeric and $L > 0$	BIGINT (@L) UNSIGNED
	L $\emptyset$ or $L \leq 0$	BIGINT UNSIGNED
P-Long Integer Unsigned Zerofill	L is numeric and $L > 0$	BIGINT (@L) UNSIGNED ZEROFILL
	L $\emptyset$ or $L \leq 0$	BIGINT UNSIGNED ZEROFILL
P-Longblob		LONGBLOB
P-Longtext		LONGTEXT
P-Mediumblob		MEDIUMBLOB
P-Mediumint	L $\emptyset$ or $L \leq 0$	MEDIUMINT
	L is numeric and $L > 0$	MEDIUMINT (@L)
P-Mediumint Unsigned	L is numeric and $L > 0$	MEDIUMINT (@L) UNSIGNED
	L $\emptyset$ or $L \leq 0$	MEDIUMINT UNSIGNED
P-Mediumint Unsigned Zerofill	L is numeric and $L > 0$	MEDIUMINT (@L) UNSIGNED ZEROFILL
	L $\emptyset$ or $L \leq 0$	MEDIUMINT UNSIGNED ZEROFILL
P-Mediumtext		MEDIUMTEXT
P-Multimedia		BLOB
P-National Varchar	L $\emptyset$ or $L < 0$	NATIONAL VARCHAR
	L is numeric and $L \geq 0$	NATIONAL VARCHAR (@L)
P-National Varchar Binary	L is numeric and $L \geq 0$	NATIONAL VARCHAR (@L) BINARY
	L $\emptyset$ or $L < 0$	NATIONAL VARCHAR BINARY

Pivot --> Datatype (MySQL 4.1)

Pivot	Condition	Datatype
P-Numeric	$L=0$ or $L \emptyset$	NUMERIC
	$L>0$ and $D \emptyset$	NUMERIC (@L)
	$L>0$ and D not $\emptyset$	NUMERIC (@L,@D)
P-Real	$L \emptyset$ or $L \leq 0$	REAL
	$L>0$ and D not $\emptyset$	REAL (@L,@D)
P-Smallint		SMALLINT
	L is numeric and $L \neq 0$	SMALLINT (@L)
P-Smallint Unsigned	L is numeric and $L \neq 0$	SMALLINT (@L) UNSIGNED
	$L \emptyset$ or $L \leq 0$	SMALLINT UNSIGNED
P-Smallint Unsigned Zero-fill	L is numeric and $L \neq 0$	SMALLINT (@L) UNSIGNED ZEROFILL
	$L \emptyset$ or $L \leq 0$	SMALLINT UNSIGNED ZEROFILL
P-String		VARCHAR(@L)
P-Text		TEXT
P-Time		TIME
P-Timestamp		TIMESTAMP
P-Tinyblob		TINYBLOB
P-Tinyint	$L \emptyset$ or $L \leq 0$	TINYINT
	L is numeric and $L \neq 0$	TINYINT (@L)
P-Tinyint Unsigned	L is numeric and $L \neq 0$	TINYINT (@L) UNSIGNED
	$L \emptyset$ or $L \leq 0$	TINYINT UNSIGNED
P-Tinyint Unsigned Zero-fill	L is numeric and $L \neq 0$	TINYINT (@L) UNSIGNED ZEROFILL
	$L \emptyset$ or $L \leq 0$	TINYINT UNSIGNED ZEROFILL
P-Tinytext		TINYTEXT

Pivot --> Datatype (MySQL 4.1)

Pivot	Condition	Datatype
P-Varbinary	L ø or L<0	VARBINARY
	L is numeric and L>=0	VARBINARY (@L)
P-Varchar	L ø or L<0	VARCHAR
	L is numeric and L>=0	VARCHAR(@L)
P-Varchar Binary	L is numeric and L>=0	VARCHAR (@L) BINARY
	L ø or L<0	VARCHAR BINARY
P-Wide Character	L ø or L<=0	NATIONAL CHAR
	L is numeric and L<>0	NATIONAL CHAR (@L)
P-Wide Character Binary	L is numeric and L<>0	NATIONAL CHAR (@L) BINARY
	L ø or L<=0	NATIONAL CHAR BINARY
P-Year	L ø or L<=0	YEAR
	L is numeric and L<>0	YEAR (@L)

Datatype --> Pivot (MySQL 4.1)

Datatype	Condition	Pivot
BIGINT		P-Long Integer
BIGINT (L)		P-Long Integer
BIGINT (L) UNSIGNED		P-Long Integer Unsigned
BIGINT (L) UNSIGNED ZEROFILL		P-Long Integer Unsigned Zerofill
BIGINT UNSIGNED		P-Long Integer Unsigned
BIGINT UNSIGNED ZEROFILL		P-Long Integer Unsigned Zerofill
BINARY		P-Binary
BINARY (L)		P-Binary
BLOB		P-Multimedia

Datatype --> Pivot (MySQL 4.1)

Datatype	Condition	Pivot
BOOLEAN		P-Boolean
CHAR (L)		P-Character
CHAR (L) BINARY		P-Character Binary
CHAR (L) UNICODE		P-Character Unicode
CHAR (L) UNICODE BINARY		P-Character Unicode Binary
CHAR ASCII		P-Character Ascii
CHAR BINARY		P-Character Binary
CHAR UNICODE		P-Character Unicode
CHAR UNICODE BINARY		P-Character Unicode Binary
CHAR(L) ASCII		P-Character Ascii
DATE		P-Date
DATETIME		P-Datetime
DECIMAL (L)		P-Decimal
DECIMAL (L) UNSIGNED		P-Decimal Unsigned
DECIMAL (L) UNSIGNED ZEROFILL		P-Decimal Unsigned Zero-fill
DECIMAL (L,D) UNSIGNED		P-Decimal Unsigned
DECIMAL (L,D) UNSIGNED ZEROFILL		P-Decimal Unsigned Zero-fill
DECIMAL UNSIGNED		P-Decimal Unsigned
DECIMAL UNSIGNED ZEROFILL		P-Decimal Unsigned Zero-fill
DOUBLE PRECISION		P-Double
DOUBLE PRECISION (L,D)		P-Double

Datatype --> Pivot (MySQL 4.1)

Datatype	Condition	Pivot
DOUBLE PRECISION (L,D) UNSIGNED		P-Double Unsigned
DOUBLE PRECISION (L,D) UNSIGNED ZERO-FILL		P-Double Unsigned Zero-fill
DOUBLE PRECISION UNSIGNED		P-Double Unsigned
DOUBLE PRECISION UNSIGNED ZEROFILL		P-Double Unsigned Zero-fill
FLOAT		P-Float
FLOAT (L)		P-Float
FLOAT (L) UNSIGNED		P-Float Unsigned
FLOAT (L) UNSIGNED ZEROFILL		P-Float Unsigned Zerofill
FLOAT (L,D)		P-Float
FLOAT (L,D) UNSIGNED		P-Float Unsigned
FLOAT (L,D) UNSIGNED ZEROFILL		P-Float Unsigned Zerofill
FLOAT UNSIGNED		P-Float Unsigned
FLOAT UNSIGNED ZERO-FILL		P-Float Unsigned Zerofill
INTEGER		P-Integer
INTEGER (L)		P-Integer
INTEGER (L) UNSIGNED		P-Integer Unsigned
INTEGER (L) UNSIGNED ZEROFILL		P-Integer Unsigned Zerofill
INTEGER UNSIGNED		P-Integer Unsigned
INTEGER UNSIGNED ZEROFILL		P-Integer Unsigned Zerofill
LOBLOB		P-Longblob

Datatype --> Pivot (MySQL 4.1)

Datatype	Condition	Pivot
LONGTEXT		P-Longtext
MEDIUMBLOB		P-Mediumblob
MEDIUMINT		P-Mediumint
MEDIUMINT (L)		P-Mediumint
MEDIUMINT (L) UNSIGNED		P-Mediumint Unsigned
MEDIUMINT (L) UNSIGNED ZEROFILL		P-Mediumint Unsigned Zerofill
MEDIUMINT UNSIGNED		P-Mediumint Unsigned
MEDIUMINT UNSIGNED ZEROFILL		P-Mediumint Unsigned Zerofill
MEDIUMTEXT		P-Mediumtext
NATIONAL CHAR		P-Wide Character
NATIONAL CHAR (L)		P-Wide Character
NATIONAL CHAR (L) BINARY		P-Wide Character Binary
NATIONAL CHAR BINARY		P-Wide Character Binary
NATIONAL VARCHAR		P-National Varchar
NATIONAL VARCHAR (L)		P-National Varchar
NATIONAL VARCHAR (L) BINARY		P-National Varchar Binary
NATIONAL VARCHAR BINARY		P-National Varchar Binary
NUMERIC		P-Numeric
NUMERIC (L)		P-Numeric
NUMERIC (L,D)		P-Numeric
REAL		P-Real

Datatype --> Pivot (MySQL 4.1)

Datatype	Condition	Pivot
REAL (L,D)		P-Real
REAL (L,D) UNSIGNED		
REAL (L,D) UNSIGNED ZEROFILL		
REAL UNSIGNED		
REAL UNSIGNED ZEROFILL		
SMALLINT		P-Smallint
SMALLINT (L)		P-Smallint
SMALLINT (L) UNSIGNED		P-Smallint Unsigned
SMALLINT (L) UNSIGNED ZEROFILL		P-Smallint Unsigned Zero-fill
SMALLINT UNSIGNED		P-Smallint Unsigned
SMALLINT UNSIGNED ZEROFILL		P-Smallint Unsigned Zero-fill
TEXT		P-Text
TIME		P-Time
TIMESTAMP		P-Timestamp
TINYBLOB		P-Tinyblob
TINYINT		P-Tinyint
TINYINT (L)		P-Tinyint
TINYINT (L) UNSIGNED		P-Tinyint Unsigned
TINYINT (L) UNSIGNED ZEROFILL		P-Tinyint Unsigned Zero-fill
TINYINT UNSIGNED		P-Tinyint Unsigned
TINYINT UNSIGNED ZEROFILL		P-Tinyint Unsigned Zero-fill
TINYTEXT		P-Tinytext
VARBINARY		P-Varbinary

Datatype --> Pivot (MySQL 4.1)

Datatype	Condition	Pivot
VARBINARY (L)		P-Varbinary
VARCHAR		P-Varchar
VARCHAR (L) BINARY		P-Varchar Binary
VARCHAR BINARY		P-Varchar Binary
VARCHAR(L)		P-Varchar
YEAR		P-Year
YEAR (L)		P-Year

MySQL 5.0

Pivot --> Datatype (MySQL 5.0)

Pivot	Condition	Datatype
	L>0 and D not ø	REAL (@L,@D) UNSIGNED
	L=0 or L ø	REAL UNSIGNED
	L>0 and D not ø	REAL (@L,@D) UNSIGNED ZEROFILL
	L=0 or L ø	REAL UNSIGNED ZEROFILL
P-AutoIdentifier		INTEGER
P-Binary	L ø or L<=0	BINARY
	L is numeric and L<>0	BINARY (@L)
P-Boolean		BOOLEAN
P-Byte	L ø or L<=0	BIT
	L is numeric and L<>0	BIT (@L)
P-Character	Not Unicode and L=0	CHAR
	Unicode and L<>0	CHAR (@L) UNICODE
	Unicode and L=0	CHAR UNICODE
	Not Unicode and L<>0	CHAR(@L)
P-Character Ascii	L ø or L<=0	CHAR ASCII
	L is numeric and L<>0	CHAR(@L) ASCII
P-Character Binary	Not Unicode and L<>0	CHAR (@L) BINARY
	Unicode and L<>0	CHAR (@L) UNICODE BINARY
	Not Unicode and L=0	CHAR BINARY
	Unicode and L=0	CHAR UNICODE BINARY

Pivot --> Datatype (MySQL 5.0)

Pivot	Condition	Datatype
P-Character Unicode	L is numeric and $L \neq 0$	CHAR (@L) UNICODE
	L $\emptyset$ or $L \leq 0$	CHAR UNICODE
P-Character Unicode Binary	L is numeric and $L \neq 0$	CHAR (@L) UNICODE BINARY
	L $\emptyset$ or $L \leq 0$	CHAR UNICODE BINARY
P-Currency		DECIMAL(@L,@D)
P-Date		DATE
P-Datetime		DATETIME
P-Decimal	$L=0$ or L $\emptyset$	DECIMAL
	$L > 0$ and D $\emptyset$	DECIMAL (@L)
	$L > 0$ and D not $\emptyset$	DECIMAL(@L,@D)
P-Decimal Unsigned	$L > 0$ and D $\emptyset$	DECIMAL (@L) UNSIGNED
	$L > 0$ and D not $\emptyset$	DECIMAL (@L,@D) UNSIGNED
	$L=0$ or L $\emptyset$	DECIMAL UNSIGNED
P-Decimal Unsigned Zero-fill	$L > 0$ and D $\emptyset$	DECIMAL (@L) UNSIGNED ZEROFILL
	$L > 0$ and D not $\emptyset$	DECIMAL (@L,@D) UNSIGNED ZEROFILL
	$L=0$ or L $\emptyset$	DECIMAL UNSIGNED ZEROFILL
P-Double	$L=0$ or L $\emptyset$	DOUBLE PRECISION
	$L > 0$ and D not $\emptyset$	DOUBLE PRECISION (@L,@D)
P-Double Unsigned	$L > 0$ and D not $\emptyset$	DOUBLE PRECISION (@L,@D) UNSIGNED
	$L=0$ or L $\emptyset$	DOUBLE PRECISION UNSIGNED

Pivot --> Datatype (MySQL 5.0)

Pivot	Condition	Datatype
P-Double Unsigned Zero-fill	$L > 0$ and $D \neq \emptyset$	DOUBLE PRECISION (@L,@D) UNSIGNED ZEROFILL
	$L = 0$ or $L \neq \emptyset$	DOUBLE PRECISION UNSIGNED ZEROFILL
P-Float	$L = 0$ or $L \neq \emptyset$	FLOAT
	$L > 0$ and $D \neq \emptyset$	FLOAT (@L)
	$L > 0$ and $D \neq \emptyset$	FLOAT (@L,@D)
P-Float Unsigned	$L > 0$ and $D \neq \emptyset$	FLOAT (@L) UNSIGNED
	$L > 0$ and $D \neq \emptyset$	FLOAT (@L,@D) UNSIGNED
	$L = 0$ or $L \neq \emptyset$	FLOAT UNSIGNED
P-Float Unsigned Zerofill	$L > 0$ and $D \neq \emptyset$	FLOAT (@L) UNSIGNED ZEROFILL
	$L > 0$ and $D \neq \emptyset$	FLOAT (@L,@D) UNSIGNED ZEROFILL
	$L = 0$ or $L \neq \emptyset$	FLOAT UNSIGNED ZEROFILL
P-Integer	$L \neq \emptyset$ or $L \leq 0$	INTEGER
	L is numeric and $L \neq 0$	INTEGER (@L)
P-Integer Unsigned	L is numeric and $L \neq 0$	INTEGER (@L) UNSIGNED
	$L \neq \emptyset$ or $L \leq 0$	INTEGER UNSIGNED
P-Integer Unsigned Zerofill	L is numeric and $L \neq 0$	INTEGER (@L) UNSIGNED ZEROFILL
	$L \neq \emptyset$ or $L \leq 0$	INTEGER UNSIGNED ZEROFILL
P-Long Integer	$L \neq \emptyset$ or $L \leq 0$	BIGINT
	L is numeric and $L \neq 0$	BIGINT (@L)

Pivot --> Datatype (MySQL 5.0)

Pivot	Condition	Datatype
P-Long Integer Unsigned	L is numeric and $L > 0$	BIGINT (@L) UNSIGNED
	L $\emptyset$ or $L \leq 0$	BIGINT UNSIGNED
P-Long Integer Unsigned Zerofill	L is numeric and $L > 0$	BIGINT (@L) UNSIGNED ZEROFILL
	L $\emptyset$ or $L \leq 0$	BIGINT UNSIGNED ZEROFILL
P-Longblob		LONGBLOB
P-Longtext		LONGTEXT
P-Mediumblob		MEDIUMBLOB
P-Mediumint	L $\emptyset$ or $L \leq 0$	MEDIUMINT
	L is numeric and $L > 0$	MEDIUMINT (@L)
P-Mediumint Unsigned	L is numeric and $L > 0$	MEDIUMINT (@L) UNSIGNED
	L $\emptyset$ or $L \leq 0$	MEDIUMINT UNSIGNED
P-Mediumint Unsigned Zerofill	L is numeric and $L > 0$	MEDIUMINT (@L) UNSIGNED ZEROFILL
	L $\emptyset$ or $L \leq 0$	MEDIUMINT UNSIGNED ZEROFILL
P-Mediumtext		MEDIUMTEXT
P-Multimedia		BLOB
P-National Varchar	L $\emptyset$ or $L < 0$	NATIONAL VARCHAR
	L is numeric and $L > 0$	NATIONAL VARCHAR (@L)
P-National Varchar Binary	L is numeric and $L > 0$	NATIONAL VARCHAR (@L) BINARY
	L $\emptyset$ or $L < 0$	NATIONAL VARCHAR BINARY

Pivot --> Datatype (MySQL 5.0)

Pivot	Condition	Datatype
P-Numeric	$L=0$ or $L \emptyset$	NUMERIC
	$L>0$ and $D \emptyset$	NUMERIC (@L)
	$L>0$ and D not $\emptyset$	NUMERIC (@L,@D)
P-Real	$L=0$ or $L \emptyset$	REAL
	$L>0$ and D not $\emptyset$	REAL (@L,@D)
P-Smallint	$L \emptyset$ or $L \leq 0$	SMALLINT
	L is numeric and $L \neq 0$	SMALLINT (@L)
P-Smallint Unsigned	L is numeric and $L \neq 0$	SMALLINT (@L) UNSIGNED
	$L \emptyset$ or $L \leq 0$	SMALLINT UNSIGNED
P-Smallint Unsigned Zero-fill	L is numeric and $L \neq 0$	SMALLINT (@L) UNSIGNED ZEROFILL
	$L \emptyset$ or $L \leq 0$	SMALLINT UNSIGNED ZEROFILL
P-String		VARCHAR(@L)
P-Text		TEXT
P-Time		TIME
P-Timestamp		TIMESTAMP
P-Tinyblob		TINYBLOB
P-Tinyint	$L \emptyset$ or $L \leq 0$	TINYINT
	L is numeric and $L \neq 0$	TINYINT (@L)
P-Tinyint Unsigned	L is numeric and $L \neq 0$	TINYINT (@L) UNSIGNED
	$L \emptyset$ or $L \leq 0$	TINYINT UNSIGNED
P-Tinyint Unsigned Zero-fill	L is numeric and $L \neq 0$	TINYINT (@L) UNSIGNED ZEROFILL
	$L \emptyset$ or $L \leq 0$	TINYINT UNSIGNED ZEROFILL
P-Tinytext		TINYTEXT

Pivot --> Datatype (MySQL 5.0)

Pivot	Condition	Datatype
P-Varbinary	L ø or L<0	VARBINARY
	L is numeric and L>=0	VARBINARY (@L)
P-Varchar	L ø or L<=0	VARCHAR
	L is numeric and L=0	VARCHAR(@L)
P-Varchar Binary	L is numeric and L<>0	VARCHAR (@L) BINARY
	L ø or L<0	VARCHAR BINARY
P-Wide Character	L ø or L<=0	NATIONAL CHAR
	L is numeric and L<>0	NATIONAL CHAR (@L)
P-Wide Character Binary	L is numeric and L<>0	NATIONAL CHAR (@L) BINARY
	L ø or L<=0	NATIONAL CHAR BINARY
P-Year	L ø or L<=0	YEAR
	L is numeric and L<>0	YEAR(@L)

Datatype --> Pivot (MySQL 5.0)

Datatype	Condition	Pivot
BIGINT		P-Long Integer
BIGINT (L)		P-Long Integer
BIGINT (L) UNSIGNED		P-Long Integer Unsigned
BIGINT (L) UNSIGNED ZEROFILL		P-Long Integer Unsigned Zerofill
BIGINT UNSIGNED		P-Long Integer Unsigned
BIGINT UNSIGNED ZEROFILL		P-Long Integer Unsigned Zerofill
BINARY		P-Binary
BINARY (L)		P-Binary
BIT		P-Byte

Datatype --> Pivot (MySQL 5.0)

Datatype	Condition	Pivot
BIT (L)		P-Byte
BLOB		P-Multimedia
BOOLEAN		P-Boolean
CHAR		P-Character
CHAR (L) BINARY		P-Character Binary
CHAR (L) UNICODE		P-Character Unicode
CHAR (L) UNICODE BINARY		P-Character Unicode Binary
CHAR ASCII		P-Character Ascii
CHAR BINARY		P-Character Binary
CHAR UNICODE		P-Character Unicode
CHAR UNICODE BINARY		P-Character Unicode Binary
CHAR(L)		P-Character
CHAR(L) ASCII		P-Character Ascii
DATE		P-Date
DATETIME		P-Datetime
DECIMAL		P-Decimal
DECIMAL (L)		P-Decimal
DECIMAL (L) UNSIGNED		P-Decimal Unsigned
DECIMAL (L) UNSIGNED ZEROFILL		P-Decimal Unsigned Zero-fill
DECIMAL (L,D) UNSIGNED		P-Decimal Unsigned
DECIMAL (L,D) UNSIGNED ZEROFILL		P-Decimal Unsigned Zero-fill
DECIMAL UNSIGNED		P-Decimal Unsigned
DECIMAL UNSIGNED ZEROFILL		P-Decimal Unsigned Zero-fill

Datatype --> Pivot (MySQL 5.0)

Datatype	Condition	Pivot
DECIMAL(L,D)		P-Decimal
DOUBLE PRECISION		P-Double
DOUBLE PRECISION (L,D)		P-Double
DOUBLE PRECISION (L,D) UNSIGNED		P-Double Unsigned
DOUBLE PRECISION (L,D) UNSIGNED ZERO-FILL		P-Double Unsigned Zero-fill
DOUBLE PRECISION UNSIGNED		P-Double Unsigned
DOUBLE PRECISION UNSIGNED ZEROFILL		P-Double Unsigned Zero-fill
FLOAT		P-Float
FLOAT (L)		P-Float
FLOAT (L) UNSIGNED		P-Float Unsigned
FLOAT (L) UNSIGNED ZEROFILL		P-Float Unsigned Zerofill
FLOAT (L,D)		P-Float
FLOAT (L,D) UNSIGNED		P-Float Unsigned
FLOAT (L,D) UNSIGNED ZEROFILL		P-Float Unsigned Zerofill
FLOAT UNSIGNED		P-Float Unsigned
FLOAT UNSIGNED ZEROFILL		P-Float Unsigned Zerofill
INTEGER		P-Integer
INTEGER (L)		P-Integer
INTEGER (L) UNSIGNED		P-Integer Unsigned
INTEGER (L) UNSIGNED ZEROFILL		P-Integer Unsigned Zerofill

Datatype --> Pivot (MySQL 5.0)

Datatype	Condition	Pivot
INTEGER UNSIGNED		P-Integer Unsigned
INTEGER UNSIGNED ZEROFILL		P-Integer Unsigned Zerofill
LOBLOB		P-Longblob
LONGTEXT		P-Longtext
MEDIUMBLOB		P-Mediumblob
MEDIUMINT		P-Mediumint
MEDIUMINT (L)		P-Mediumint
MEDIUMINT (L) UNSIGNED		P-Mediumint Unsigned
MEDIUMINT (L) UNSIGNED ZEROFILL		P-Mediumint Unsigned Zerofill
MEDIUMINT UNSIGNED		P-Mediumint Unsigned
MEDIUMINT UNSIGNED ZEROFILL		P-Mediumint Unsigned Zerofill
MEDIUMTEXT		P-Mediumtext
NATIONAL CHAR		P-Wide Character
NATIONAL CHAR (L)		P-Wide Character
NATIONAL CHAR (L) BINARY		P-Wide Character Binary
NATIONAL CHAR BINARY		P-Wide Character Binary
NATIONAL VARCHAR		P-National Varchar
NATIONAL VARCHAR (L)		P-National Varchar
NATIONAL VARCHAR (L) BINARY		P-National Varchar Binary
NATIONAL VARCHAR BINARY		P-National Varchar Binary
NUMERIC		P-Numeric

Datatype --> Pivot (MySQL 5.0)

Datatype	Condition	Pivot
NUMERIC (L)		P-Numeric
NUMERIC (L,D)		P-Numeric
REAL		P-Real
REAL (L,D)		P-Real
REAL (L,D) UNSIGNED		
REAL (L,D) UNSIGNED ZEROFILL		
REAL UNSIGNED		
REAL UNSIGNED ZEROFILL		
SMALLINT		P-Smallint
SMALLINT (L)		P-Smallint
SMALLINT (L) UNSIGNED		P-Smallint Unsigned
SMALLINT (L) UNSIGNED ZEROFILL		P-Smallint Unsigned Zero-fill
SMALLINT UNSIGNED		P-Smallint Unsigned
SMALLINT UNSIGNED ZEROFILL		P-Smallint Unsigned Zero-fill
TEXT		P-Text
TIME		P-Time
TIMESTAMP		P-Timestamp
TINYBLOB		P-Tinyblob
TINYINT		P-Tinyint
TINYINT (L)		P-Tinyint
TINYINT (L) UNSIGNED		P-Tinyint Unsigned
TINYINT (L) UNSIGNED ZEROFILL		P-Tinyint Unsigned Zero-fill
TINYINT UNSIGNED		P-Tinyint Unsigned

Datatype --> Pivot (MySQL 5.0)

Datatype	Condition	Pivot
TINYINT UNSIGNED ZEROFILL		P-Tinyint Unsigned Zero-fill
TINYTEXT		P-Tinytext
VARBINARY		P-Varbinary
VARBINARY (L)		P-Varbinary
VARCHAR		P-Varchar
VARCHAR (L) BINARY		P-Varchar Binary
VARCHAR BINARY		P-Varchar Binary
VARCHAR(L)		P-Varchar
YEAR		P-Year
YEAR(L)		P-Year

ORACLE 10

Pivot --> Datatype (Oracle 10)

Pivot	Condition	Datatype
P-AutoIdentifier		NUMBER
P-Binary		RAW(@L)
P-Boolean	L=2 or L ø	RAW(1)
	L>1	RAW(@L)
P-Byte		RAW(1)
P-Character	Not Unicode and (L<2001 or L ø)	CHAR(@L)
	L>4000	LONG
	Not Unicode and (L=2001 or L ø)	NCHAR(@L)
	Unicode and 2000<L<4001	NVARCHAR2(@L)
	Not Unicode and 2000<L<4001	VARCHAR2(@L)
P-Currency		NUMBER(@L,@D)
P-Date		DATE
P-Datetime		DATE
P-Decimal		NUMBER(@L,@D)
P-Double		NUMBER(@L,@D)
P-Float	L=0 or L>126 or L ø	FLOAT
	0<L<127	FLOAT(@L)
P-Integer		NUMBER(@L)
P-Long Integer		NUMBER(@L)
P-Long Real		NUMBER(@L,@D)
P-Multimedia		LONG RAW

Pivot --> Datatype (Oracle 10)

Pivot	Condition	Datatype
P-Numeric	L=0 or L ø	NUMBER
	L>0 and D ø	NUMBER(@L)
	L>0 and D not ø	NUMBER(@L,@D)
P-Real		NUMBER(@L,@D)
P-Smallint		NUMBER(@L)
P-String		LONG
P-Text	Unicode	NVARCHAR2(@L)
	Not Unicode	VARCHAR2(@L)
P-Time		DATE
P-Timestamp	L>9 or L ø	TIMESTAMP
	L<10	TIMESTAMP(@L)
P-Tinyint		NUMBER(@L)
P-Varbinary		LONG RAW
P-Varchar	L>4000 or L=0 or L ø	LONG
	Unicode and 0<L<4001	NVARCHAR2(@L)
	Not Unicode and 0<L<4001	VARCHAR2(@L)

Datatype --> Pivot (Oracle 10)

Datatype	Condition	Pivot
CHAR(L)		P-Character
DATE		P-Date
FLOAT		P-Float
FLOAT(L)		P-Float
LONG		P-String
LONG RAW		P-Multimedia
NUMBER		P-Numeric

Datatype --> Pivot (Oracle 10)

Datatype	Condition	Pivot
NUMBER(L)		P-Numeric
NUMBER(L,D)		P-Numeric
RAW(1)		P-Boolean
RAW(L)		P-Boolean
TIMESTAMP		P-Timestamp
TIMESTAMP(L)		P-Timestamp
VARCHAR2(L)		P-Varchar

ORACLE 11

Pivot --> Datatype (Oracle 11)

Pivot	Condition	Datatype
P-AutoIdentifier		NUMBER
P-Binary		RAW(@L)
P-Boolean	L=2 or L ø	RAW(1)
	L>1	RAW(@L)
P-Byte		RAW(1)
P-Character	Not Unicode and (L<2001 or L ø)	CHAR(@L)
	L>4000	LONG
	Not Unicode and (L=2001 or L ø)	NCHAR(@L)
	Unicode and 2000<L<4001	NVARCHAR2(@L)
	Not Unicode and 2000<L<4001	VARCHAR2(@L)
P-Currency		NUMBER(@L,@D)
P-Date		DATE
P-Datetime		DATE
P-Decimal		NUMBER(@L,@D)
P-Double		NUMBER(@L,@D)
P-Float	0<L<127	FLOAT(@L)
	L=0 or L>126 or L ø	FLOAT
P-Integer		NUMBER(@L)
P-Long Integer		NUMBER(@L)
P-Long Real		NUMBER(@L,@D)

Pivot --> Datatype (Oracle 11)

Pivot	Condition	Datatype
P-Numeric	L=0 or L ø	NUMBER
	L>0 and D ø	NUMBER(@L)
	L>0 and D not ø	NUMBER(@L,@D)
P-Real		NUMBER(@L,@D)
P-Smallint		NUMBER(@L)
P-String		LONG
P-Text	Unicode	NVARCHAR2(@L)
	Not Unicode	VARCHAR2(@L)
P-Time		DATE
P-Timestamp	L<10	TIMESTAMP(@L)
	L>9 or L ø	TIMESTAMP
P-Tinyint		NUMBER(@L)
P-Varchar	L>4000 or L=0 or L ø	LONG
	Unicode and 0<L<4001	NVARCHAR2(@L)
	Not Unicode and 0<L<4001	VARCHAR2(@L)

Datatype --> Pivot (Oracle 11)

Datatype	Condition	Pivot
CHAR(L)		P-Character
DATE		P-Date
FLOAT		P-Float
FLOAT(L)		P-Float
LONG		P-String
NUMBER		P-Numeric
NUMBER(L)		P-Numeric
NUMBER(L,D)		P-Numeric

Datatype --> Pivot (Oracle 11)

Datatype	Condition	Pivot
RAW(1)		P-Boolean
RAW(L)		P-Boolean
TIMESTAMP		P-Timestamp
TIMESTAMP(L)		P-Timestamp
VARCHAR2(L)		P-Varchar

ORACLE 8

Pivot --> Datatype (Oracle 8)

Pivot	Condition	Datatype
P-AutoIdentifier		NUMBER
P-Binary		RAW(@L)
P-Boolean	L=2 or L ø	RAW(1)
	L>1	RAW(@L)
P-Byte		RAW(1)
P-Character	L=2001 or L ø	CHAR(@L)
	L>4000	LONG
	2000<L<4001	VARCHAR2(@L)
P-Currency		NUMBER(@L,@D)
P-Date		DATE
P-Datetime		DATE
P-Decimal		NUMBER(@L,@D)
P-Double		NUMBER(@L,@D)
P-Float		NUMBER(@L,@D)
P-Integer		NUMBER(@L)
P-Long Integer		NUMBER(@L)
P-Long Real		NUMBER(@L,@D)
P-Multimedia		LONG RAW
P-Numeric	L=0 or L ø	NUMBER
	L>0 and D ø	NUMBER(@L)
	L>0 and D not ø	NUMBER(@L,@D)
P-Real		NUMBER(@L,@D)
P-Smallint		NUMBER(@L)
P-String		LONG

Pivot --> Datatype (Oracle 8)

Pivot	Condition	Datatype
P-Text		VARCHAR2(@L)
P-Time		DATE
P-Timestamp		ROWID
P-Tinyint		NUMBER(@L)
P-Varbinary		LONG RAW
P-Varchar	L>4000 or L=0 or L ø	LONG
	0<L<4001	VARCHAR2(@L)

Datatype --> Pivot (Oracle 8)

Datatype	Condition	Pivot
CHAR(L)		P-Character
DATE		P-Date
LONG		P-String
LONG RAW		P-Multimedia
NUMBER		P-Numeric
NUMBER(L)		P-Numeric
NUMBER(L,D)		P-Numeric
RAW(1)		P-Boolean
RAW(L)		P-Boolean
ROWID		P-Timestamp
VARCHAR2(L)		P-Varchar

ORACLE 9i

Pivot --> Datatype (Oracle 9i)

Pivot	Condition	Datatype
P-AutoIdentifier		NUMBER
P-Binary		RAW(@L)
P-Boolean	L=2 or L ø	RAW(1)
	L>1	RAW(@L)
P-Byte		RAW(1)
P-Character	Not Unicode and (L<2001 or L ø)	CHAR(@L)
	L>4000	LONG
	Not Unicode and (L=2001 or L ø)	NCHAR(@L)
	Unicode and 2000<L<4001	NVARCHAR2(@L)
	Not Unicode and 2000<L<4001	VARCHAR2(@L)
P-Currency		NUMBER(@L,@D)
P-Date		DATE
P-Datetime		DATE
P-Decimal		NUMBER(@L,@D)
P-Double		NUMBER(@L,@D)
P-Float	L=0 or L>126 or L ø	FLOAT
	0<L<127	FLOAT(@L)
P-Integer		NUMBER(@L)
P-Long Integer		NUMBER(@L)
P-Long Real		NUMBER(@L,@D)
P-Multimedia		LONG RAW
P-Numeric	L=0 or L ø	NUMBER
	L>0 and D ø	NUMBER(@L)
	L>0 and D not ø	NUMBER(@L,@D)

Pivot --> Datatype (Oracle 9i)

Pivot	Condition	Datatype
P-Real		NUMBER(@L,@D)
P-Smallint		NUMBER(@L)
P-String		LONG
P-Text	Unicode	NVARCHAR2(@L)
	Not Unicode	VARCHAR2(@L)
P-Time		DATE
P-Timestamp	L>9 or L ø	TIMESTAMP
	L<10	TIMESTAMP(@L)
P-Tinyint		NUMBER(@L)
P-Varbinary		LONG RAW
P-Varchar	L>4000 or L=0 or L ø	LONG
	Unicode and 0<L<4001	NVARCHAR2(@L)
	Not Unicode and 0<L<4001	VARCHAR2(@L)

Datatype --> Pivot (Oracle 9i)

Datatype	Condition	Pivot
CHAR(L)		P-Character
DATE		P-Date
FLOAT		P-Float
FLOAT(L)		P-Float
LONG		P-String
LONG RAW		P-Multimedia
NUMBER		P-Numeric
NUMBER(L)		P-Numeric
NUMBER(L,D)		P-Numeric
RAW(1)		P-Boolean
RAW(L)		P-Boolean

Datatype --> Pivot (Oracle 9i)

Datatype	Condition	Pivot
TIMESTAMP		P-Timestamp
VARCHAR2(L)		P-Varchar

POSTGRESQL9.3

Pivot --> Datatype (PostgreSQL9.3)

Pivot	Condition	Datatype
P-Boolean		boolean
P-Byte	L=0 or L ø	bit
	Valid	bit(@L)
P-Character	L=0 or L ø	char
	Valid	char(@L)
P-Currency		money
P-Date		date
P-Decimal	L=0 or L ø	decimal
	L>=1 and D ø	decimal(@L)
	L>=1	decimal(@L,@D)
P-Double		double precision
P-Integer		integer
P-Long Integer		bigint
P-Numeric	L=0 or L ø	numeric
	L>=1 and D ø	numeric(@L)
	L>=1	numeric(@L,@D)
P-Real		real
P-Smallint		smallint
P-Text		text
P-Time	L=0 or L ø	time
	Valid	time(@L)
P-Timestamp	L=0 or L ø	timestamp
	L <> 0	timestamp(@L)

Pivot --> Datatype (PostgreSQL9.3)

Pivot	Condition	Datatype
P-Varchar	L=0 or L ∅	varchar
	Valid	varchar(@L)

Datatype --> Pivot (PostgreSQL9.3)

Datatype	Condition	Pivot
bigint		P-Long Integer
bit		P-Byte
bit(L)		P-Byte
boolean		P-Boolean
char		P-Character
char(L)		P-Character
date		P-Date
decimal		P-Decimal
decimal(L)		P-Decimal
decimal(L,D)		P-Decimal
double precision		P-Double
integer		P-Integer
money		P-Currency
numeric		P-Numeric
numeric(L)		P-Numeric
numeric(L,D)		P-Numeric
real		P-Real
smallint		P-Smallint
text		P-Text
time		P-Time
time(L)		P-Time
timestamp		P-Timestamp

Datatype --> Pivot (PostgreSQL9.3)

Datatype	Condition	Pivot
timestamp(L)		P-Timestamp
varchar		P-Varchar
varchar(L)		P-Varchar

SQL ANSI/ISO 9075:1992

Pivot --> Datatype (SQL ANSI/ISO 9075:1992)

Pivot	Condition	Datatype
P-AutoIdentifier		INTEGER
P-Binary		BIT VARYING(@L)
P-Boolean		BIT(@L)
P-Byte		BIT(@L)
P-Character		CHAR(@L)
P-Currency		DECIMAL(@L,@D)
P-Date		DATE
P-Datetime		DATETIME
P-Decimal		DECIMAL(@L,@D)
P-Double		DOUBLE PRECISION
P-Float		FLOAT
P-Integer		INTEGER
P-Long Integer		INTEGER
P-Long Real		REAL
P-Multimedia		BIT VARYING(@L)
P-Numeric	L>4	INTEGER
	L=5 or L ø	SMALLINT
P-Real		REAL
P-Smallint		SMALLINT
P-String		VARCHAR(@L)
P-Text		VARCHAR(@L)
P-Time		TIME
P-Timestamp		DATETIME
P-Tinyint		SMALLINT

Pivot --> Datatype (SQL ANSI/ISO 9075:1992)

Pivot	Condition	Datatype
P-Varbinary		BIT VARYING(@L)
P-Varchar		VARCHAR(@L)

Datatype --> Pivot (SQL ANSI/ISO 9075:1992)

Datatype	Condition	Pivot
BIT VARYING(L)		P-Multimedia
BIT(L)		P-Boolean
CHAR(L)		P-Character
DATE		P-Date
DATETIME		P-Datetime
DECIMAL(L,D)		P-Currency
DOUBLE PRECISION		P-Double
FLOAT		P-Float
INTEGER		P-Integer
REAL		P-Real
SMALLINT		P-Smallint
TIME		P-Time
VARCHAR(L)		P-Varchar

SQL SERVER 2000

Pivot --> Datatype (SQL Server 2000)

Pivot	Condition	Datatype
P-AutoIdentifier		uniqueidentifier
P-Binary		binary(@L)
P-Boolean		bit
P-Byte		bit
P-Character	Not Unicode and (L<8001 or L ø)	char(@L)
	Not Unicode and (L=8001 or L ø)	nchar(@L)
	Not Unicode and L>8000	ntext
	Not Unicode and L>8000	text
P-Currency		money
P-Date		smalldatetime
P-Datetime		datetime
P-Decimal		decimal(@L,@D)
P-Double		numeric(@L,@D)
P-Float		float
P-Integer		int
P-Long Integer		bigint
P-Long Real		real
P-Multimedia		image
P-Numeric		numeric(@L,@D)
P-Real		real
P-Smallint		smallint

Pivot --> Datatype (SQL Server 2000)

Pivot	Condition	Datatype
P-String	Unicode	ntext
	Not Unicode	text
P-Text	Unicode	ntext
	Not Unicode	text
P-Time		datetime
P-Timestamp		timestamp
P-Tinyint		tinyint
P-Varbinary		varbinary(@L)
P-Varchar	Not Unicode and L>8000	ntext
	Not Unicode and (L=8001 or L ø)	nvarchar(@L)
	Not Unicode and L>8000	text
	Not Unicode and (L<8001 or L ø)	varchar(@L)
P-Wide Character		nchar(@L)
P-Wide String		nvarchar(@L)

Datatype --> Pivot (SQL Server 2000)

Datatype	Condition	Pivot
bigint		P-Long Integer
binary(L)		P-Binary
bit		P-Boolean
char(L)		P-Character
datetime		P-Datetime
decimal(L,D)		P-Decimal
float		P-Float
image		P-Multimedia
int		P-Integer

Datatype --> Pivot (SQL Server 2000)

Datatype	Condition	Pivot
money		P-Currency
nchar(L)		P-Wide Character
numeric(L,D)		P-Numeric
nvarchar(L)		P-Wide String
real		P-Real
smalldatetime		P-Date
smallint		P-Smallint
text		P-Text
timestamp		P-Timestamp
tinyint		P-Tinyint
uniqueidentifier		P-AutoIdentifier
varbinary(L)		P-Varbinary
varchar(L)		P-Varchar

SQL SERVER 2005

Pivot --> Datatype (SQL Server 2005)

Pivot	Condition	Datatype
P-AutoIdentifier		uniqueidentifier
P-Binary		binary(@L)
P-Boolean		bit
P-Byte		bit
P-Character	Not Unicode and (L<8001 or L ø)	char(@L)
	Not Unicode and (L=8001 or L ø)	nchar(@L)
	Not Unicode and L>8000	ntext
	Not Unicode and L>8000	text
P-Currency		money
		smallmoney
P-Date		smalldatetime
P-Datetime		datetime
P-Decimal		decimal(@L,@D)
P-Double		numeric(@L,@D)
P-Float		float
P-Integer		int
P-Long Integer		bigint
P-Long Real		real
P-Multimedia		image
P-Numeric		numeric(@L,@D)
P-Real		real
P-Smallint		smallint

Pivot --> Datatype (SQL Server 2005)

Pivot	Condition	Datatype
P-String	Unicode	ntext
	Not Unicode	text
P-Text	Unicode	ntext
	Not Unicode	text
P-Time		datetime
P-Timestamp		timestamp
P-Tinyint		tinyint
P-Varbinary		varbinary(@L)
P-Varchar	Not Unicode and L>8000	ntext
	Not Unicode and (L=8001 or L ø)	nvarchar(@L)
	Not Unicode and L>8000	text
	Not Unicode and (L<8001 or L ø)	varchar(@L)
P-Wide Character		nchar(@L)
P-Wide String		nvarchar(@L)

Datatype --> Pivot (SQL Server 2005)

Datatype	Condition	Pivot
bigint		P-Long Integer
binary(L)		P-Binary
bit		P-Boolean
char(L)		P-Character
datetime		P-Datetime
decimal(L,D)		P-Decimal
float		P-Float
image		P-Multimedia
int		P-Integer

Datatype --> Pivot (SQL Server 2005)

Datatype	Condition	Pivot
money		P-Currency
nchar(L)		P-Wide Character
numeric(L,D)		P-Numeric
nvarchar(L)		P-Wide String
real		P-Real
smalldatetime		P-Date
smallint		P-Smallint
smallmoney		P-Currency
text		P-Text
timestamp		P-Timestamp
tinyint		P-Tinyint
uniqueidentifier		P-AutoIdentifier
varbinary(L)		P-Varbinary
varchar(L)		P-Varchar

SQL SERVER 2008

Pivot --> Datatype (SQL Server 2008)

Pivot	Condition	Datatype
P-AutoIdentifier		uniqueidentifier
P-Binary		binary(@L)
P-Boolean		bit
P-Byte		bit
P-Character	Not Unicode and (L<8001 or L ø)	char(@L)
	Not Unicode and (L=8001 or L ø)	nchar(@L)
	Not Unicode and L>8000	ntext
	Not Unicode and L>8000	text
P-Currency	L not empty or L > 10	money
	L empty or L < 11	smallmoney
P-Date		smalldatetime
P-Datetime		datetime
P-Decimal		decimal(@L,@D)
P-Double		numeric(@L,@D)
P-Float	L empty	float
	L not empty	float(@L)
P-Integer		int
P-Long Integer		bigint
P-Long Real		real
P-Multimedia		image
P-Numeric		numeric(@L,@D)
P-Real		real

Pivot --> Datatype (SQL Server 2008)

Pivot	Condition	Datatype
P-Smallint		smallint
P-String	Unicode	ntext
	Not Unicode	text
P-Text	Unicode	ntext
	Not Unicode	text
P-Time		time
P-Timestamp		timestamp
P-Tinyint		tinyint
P-Varbinary		varbinary(@L)
P-Varchar	Not Unicode and L>8000	ntext
	Not Unicode and (L=8001 or L ø)	nvarchar(@L)
	Not Unicode and L>8000	text
	Not Unicode and (L<8001 or L ø)	varchar(@L)
P-Wide Character		nchar(@L)
P-Wide String		nvarchar(@L)

Datatype --> Pivot (SQL Server 2008)

Datatype	Condition	Pivot
bigint		P-Long Integer
binary(L)		P-Binary
bit		P-Boolean
char(L)		P-Character
datetime		P-Datetime
decimal(L,D)		P-Decimal
float		P-Float
float(L)		P-Float

Datatype --> Pivot (SQL Server 2008)

Datatype	Condition	Pivot
image		P-Multimedia
int		P-Integer
money		P-Currency
nchar(L)		P-Wide Character
numeric(L,D)		P-Numeric
nvarchar(L)		P-Wide String
real		P-Real
smalldatetime		P-Date
smallint		P-Smallint
smallmoney		P-Currency
text		P-Text
time		P-Time
timestamp		P-Timestamp
tinyint		P-Tinyint
uniqueidentifier		P-AutoIdentifier
varbinary(L)		P-Varbinary
varchar(L)		P-Varchar

SQL SERVER 7

Pivot --> Datatype (SQL Server 7)

Pivot	Condition	Datatype
P-AutoIdentifier		timestamp
P-Binary		binary(@L)
P-Boolean		bit
P-Byte		bit
P-Character	0<L<251	char(@L)
	L>250 or L=0 or L ø	text
P-Currency		money
P-Date		smalldatetime
P-Datetime		datetime
P-Decimal		decimal(@L,@D)
P-Double		numeric(@L,@D)
P-Float		float
P-Integer		int
P-Long Integer		int
P-Long Real		real
P-Multimedia		image
P-Numeric	L>9 and D ø	float
	4<L<10 and D ø	int
	L not ø and D not ø	numeric(@L,@D)
	2<L<5 and D ø	smallint
	(L<3 and D ø) or L ø	tinyint
P-Real		real
P-Smallint		smallint
P-String		char(@L)

Pivot --> Datatype (SQL Server 7)

Pivot	Condition	Datatype
P-Text		text
P-Time		datetime
P-Timestamp		timestamp
P-Tinyint		tinyint
P-Varbinary		varbinary(@L)
P-Varchar		varchar(@L)

Datatype --> Pivot (SQL Server 7)

Datatype	Condition	Pivot
binary(L)		P-Binary
bit		P-Boolean
char(L)		P-String
datetime		P-Datetime
decimal(L,D)		P-Decimal
float		P-Float
image		P-Multimedia
int		P-Integer
money		P-Currency
numeric(L,D)		P-Double
real		P-Real
smalldatetime		P-Date
smallint		P-Smallint
text		P-Character
timestamp		P-Timestamp
tinyint		P-Tinyint
varbinary(L)		P-Varbinary
varchar(L)		P-Varchar

SYBASE ADAPTIVE SERVER 11

Pivot --> Datatype (Sybase Adaptive Server 11)

Pivot	Condition	Datatype
P-AutoIdentifier		timestamp
P-Binary		binary(@L)
P-Boolean		bit
P-Byte		bit
P-Character	Not Unicode and L<256	char(@L)
	Unicode and L<256	nChar(@L)
	L>255 or L ø	text
P-Currency		money
P-Date		smalldatetime
P-Datetime		datetime
P-Decimal	0<L<39 and D not ø	decimal(@L,@D)
	(L>9 and D ø) or (1<L<38 and D not ø)	float
	4<L<10 and D ø	int
	2<L<5 and D ø	smallint
	(L<3 and D ø) or L ø	tinyint
P-Double		double precision
P-Float		float
P-Integer		int
P-Long Integer		int
P-Long Real		real
P-Multimedia		image

Pivot --> Datatype (Sybase Adaptive Server 11)

Pivot	Condition	Datatype
P-Numeric	(L>9 and D ø) or (1<L<38 and D not ø)	float
	4<L<10 and D ø	int
	0<L<39 and D not ø	numeric(@L,@D)
	2<L<5 and D ø	smallint
	(L<3 and D ø) or L ø	tinyint
P-Real		real
P-Smallint		smallint
P-String	Not Unicode	char(@L)
	Unicode	nChar(@L)
P-Text		text
P-Time		datetime
P-Timestamp		timestamp
P-Tinyint		tinyint
P-Varbinary		varbinary(@L)
P-Varchar	Unicode	nVarChar(@L)
	Not Unicode	varchar(@L)

Datatype --> Pivot (Sybase Adaptive Server 11)

Datatype	Condition	Pivot
binary(L)		P-Binary
bit		P-Boolean
char(L)		P-String
datetime		P-Datetime
decimal(L,D)		P-Decimal
double precision		P-Double
float		P-Float

Datatype --> Pivot (Sybase Adaptive Server 11)

Datatype	Condition	Pivot
image		P-Multimedia
int		P-Integer
money		P-Currency
numeric(L,D)		P-Numeric
real		P-Real
smalldatetime		P-Date
smallint		P-Smallint
text		P-Character
timestamp		P-Timestamp
tinyint		P-Tinyint
varbinary(L)		P-Varbinary
varchar(L)		P-Varchar

SYBASE ADAPTIVE SERVER 12.5

Pivot --> Datatype (Sybase Adaptive Server 12.5)

Pivot	Condition	Datatype
P-AutoIdentifier		timestamp
P-Binary	L=0 or L $\emptyset$	binary
	L > 0	binary(@L)
P-Boolean		bit
P-Byte		bit
P-Character	Not Unicode and (L=0 or L $\emptyset$)	char
	Not Unicode and 0<L<256	char(@L)
	L>255	text
	Not Unicode and (L=256 or L $\emptyset$)	unichar(@L)
P-Currency		money
P-Date		smalldatetime
P-Datetime		datetime
P-Decimal	0<L<39 and D not $\emptyset$	decimal(@L,@D)
	(L>9 and D $\emptyset$) or ((L<1 or L>38) and D not $\emptyset$)	float
	4<L<10 and D $\emptyset$	int
	2<L<5 and D $\emptyset$	smallint
	L $\emptyset$ or (L<3 and D $\emptyset$)	tinyint
P-Double		double precision
P-Float		float
P-Integer		int
P-Long Integer		int
P-Long Real		real

Pivot --> Datatype (Sybase Adaptive Server 12.5)

Pivot	Condition	Datatype
P-Multimedia		image
P-Numeric	(L>9 and D $\emptyset$) or ((L<1 or L>38) and D not $\emptyset$)	float
	4<L<10 and D $\emptyset$	int
	0<L<39 and D not $\emptyset$	numeric(@L,@D)
	2<L<5 and D $\emptyset$	smallint
	L $\emptyset$ or (L<3 and D $\emptyset$)	tinyint
P-Real		real
P-Smallint		smallint
P-String	Not Unicode and (L=0 or L $\emptyset$)	char
	Not Unicode and (0<L<256)	char(@L)
	L>255	text
	Not Unicode and (L=256 or L $\emptyset$)	unichar(@L)
P-Text		text
P-Time		datetime
P-Timestamp		timestamp
P-Tinyint		tinyint
P-Varbinary	L=0 or L $\emptyset$	varbinary
	L > 0	varbinary(@L)
P-Varchar	L>255	text
	Not Unicode and (L=256 or L $\emptyset$)	univarchar(@L)
	Not Unicode and (L=0 or L $\emptyset$)	varchar
	Not Unicode and 0<L<256	varchar(@L)

Datatype --> Pivot (Sybase Adaptive Server 12.5)

Datatype	Condition	Pivot
binary		P-Binary
binary(L)		P-Binary
bit		P-Boolean
char		P-Character
char(L)		P-Character
datetime		P-Datetime
decimal(L,D)		P-Decimal
double precision		P-Double
float		P-Float
image		P-Multimedia
int		P-Integer
money		P-Currency
numeric(L,D)		P-Numeric
real		P-Real
smalldatetime		P-Date
smallint		P-Smallint
text		P-Text
timestamp		P-Timestamp
tinyint		P-Tinyint
varbinary		P-Varbinary
varbinary(L)		P-Varbinary
varchar		P-Varchar
varchar(L)		P-Varchar

TERADATA DATABASE

Pivot --> Datatype (Teradata Database 14)

Pivot	Condition	Datatype
P-AutoIdentifier		NUMBER
P-Boolean		BYTEINT
P-Byte	L=0 or L ø	BYTE
	L>0	BYTE(@L)
P-Character	L=0 or L ø	CHAR
	L>0	CHAR(@L)
P-Date		DATE
P-Datetime		DATE
P-Decimal	L=0 or L ø and D=0 Or D ø	DECIMAL
	L>0 and D=0 Or D ø	DECIMAL(@L)
P-Double		NUMBER(@L,@D)
P-Float		FLOAT
P-Integer		INTEGER
P-Long Integer		BIGINT
P-Long Real		FLOAT
P-Multimedia	L=0 or L ø	BLOB
	L>0	BLOB(@L)
P-Numeric	L=0 or L ø and D=0 Or D ø	NUMBER
	L ø and D > 0	NUMBER(*,@D)
	L>0 and D=0 Or D ø	NUMBER(@L)
	L>0 and D>0	NUMBER(@L,@D)
P-Real		FLOAT
P-Smallint		SMALLINT
P-String		VARCHAR(@L)

Pivot --> Datatype (Teradata Database 14)

Pivot	Condition	Datatype
P-Text		VARCHAR(@L)
P-Time		TIME
	D > 0	TIME(@D)
P-Timestamp		TIMESTAMP
	D > 0	TIMESTAMP(@D)
P-Tinyint		SMALLINT
P-Varbinary		VARBYTE(@L)
P-Varchar		VARCHAR(@L)

Datatype --> Pivot (Teradata Database 14)

Datatype	Condition	Pivot
BIGINT		P-Long Integer
BLOB		P-Multimedia
BLOB(L)		P-Multimedia
BYTE		P-Byte
BYTE(L)		P-Byte
BYTEINT		P-Boolean
CHAR		P-Character
CHAR(L)		P-Character
DATE		P-Date
DECIMAL		P-Decimal
DECIMAL(L)		P-Decimal
FLOAT		P-Real
INTEGER		P-Integer

Datatype --> Pivot (Teradata Database 14)

Datatype	Condition	Pivot
NUMBER		P-Numeric
NUMBER(*,D)		P-Numeric
NUMBER(L)		P-Numeric
NUMBER(L,D)		P-Numeric
SMALLINT		P-Smallint
TIME		P-Time
TIME(D)		P-Time
TIMESTAMP		P-Timestamp
TIMESTAMP(D)		P-Timestamp
VARBYTE(L)		P-Varbinary
VARCHAR(L)		P-Varchar







Compliance

RULES AND REGULATIONS



Regulatory frameworks encourage organizations to apply a governance policy in order for data to be compliant with the expectations of authorities, partners and clients.

The functional administrator can define in **HOPEX Data Governance** regulatory frameworks and business policies with which the organization must comply.

The **HOPEX Data Governance** solution offers tools to define regulatory frameworks that put restrictions on organizations. In particular, it provides the contents of the BCBS 239 and Solvency II regulations, which you can import into the repository. Once the regulations have been defined, you can assess the level of compliance of your data with these directives.

CREATING AN INVENTORY OF REGULATORY FRAMEWORKS (EXTERNAL REGULATIONS)

A regulatory framework is an authority document falling under any of following categories: regulations (rules of law that, if not followed, can result in penalties), guidelines, standards, best practices.

HOPEX Data Governance provides by default the BCBS 239 banking regulatory framework as well as the Solvability II regulation, in the form of modules. You can define other regulatory frameworks in your repository.

Importing the Module of Regulatory Frameworks

The libraries of regulations to be imported are delivered in the form of modules that can be downloaded into the HOPEX HAS console.

To download the module of a regulation:

- 1 From your HOPEX version, open the HAS Administration Console and download the module. See [Importing a Module into HOPEX](#).

Structure of a Regulatory Framework

A regulatory framework defines the content of a regulation. It includes a set of sections, and under each section, sub-sections and articles.

You can define the objects in your organization that are constrained by the regulation as well as the business rules that ensure the implementation of the regulation within the company.

Because a regulation can contain many chapters, you can indicate precisely to which section or article objects are subject and for which business rules to associate.

➡ For more details on business rules, see [Ensure Compliance with Data: Create Business Rules](#).

		Actions
 	EU General Data Protection Regulation (GDPR)	
 	Chapter II. Principles	   
 	Chapter III. Rights of the data subject	        
 	§ 1. Transparency and modalities	   
 	Art. 12. Transparent information, communication and mo...	   
 	Constrained Assets	
 	Implementations	
 	§ 2. Information and access to personal data	   

Creating a Regulatory Framework

To create a regulatory framework in **HOPEX Data Governance**:

1. In the navigation menu, click **Compliance > Regulatory frameworks**.
2. In the edit area, click the **Regulatory Frameworks** tab.
3. Click **New**.
4. In the dialog box that appears, specify:
 - the name of the regulatory framework
 - its description
5. Click **OK**.
The regulatory framework appears in the list.

Defining the objects in your organization constrained by a regulatory framework

You can define which objects a regulation constrains in the properties of the object in question. For example, to define the regulations associated with a concept, see: [Regulations](#).

See also: [Analyzing Information Constrained by a Regulation.](#)

Viewing the objects constrained by a regulatory framework

To view the objects constrained by a regulatory framework:

1. Select the regulatory framework.
Buttons appear to its right.
2. Click **Properties**.



3. In the regulatory framework properties, click the **Subjected Elements** page.

Adding a Section to the Regulatory Framework

To add a section to the regulatory framework:

1. Click the regulatory framework icon then **New > Regulation Section**.
The creation window for a section opens.
2. Specify:
 - (optional) the section code
 - the title
 - ☛ *The name of the section is calculated automatically using the code of the title.*
 - the description, which corresponds to the text of the section.
 - ☛ *When the section is composed of other sections and articles, the texts can appear in them.*
 - (optional) the subjected elements: these concern the objects that are constrained by the section. It can be business data, physical data, application data, etc.
 - (optional) implementation: this concerns the business rules that ensure the effective implementation of the regulation within the enterprise.
 - ☛ See [Ensure Compliance with Data: Create Business Rules](#).
3. Click **OK**.
The section appears in the navigation tree.

Defining the objects constrained by a section of a regulatory framework

To view the objects constrained by a regulatory framework:

1. Select the regulatory framework.
Buttons appear to its right.
2. Click **Properties**.

3. In the section properties, click the **Subjected Elements** page.

Adding an Article to a Section

To add an article to a section:

1. Click the section icon then **New > Regulation Section**.
The window for creating an article appears. You can define the following information:
 - the article code
 - the title

 *The name of the article is calculated automatically using the code of the title.*
 - the description, which corresponds to the text of the article.
 - (optional) the concept definitions that can be associated with key words of the text.
 - (optional) the subjected elements: these concern the objects that are constrained by the article. It can be business data, physical data, application data, etc.
 - (optional) implementations: this concerns the business rules that ensure the effective implementation of the regulation within the enterprise.

 See [Ensure Compliance with Data: Create Business Rules](#).
2. Click **OK**.
The article appears in the navigation tree under the section.

Adding a Definition to an Article

You can connect to the article definitions of concepts that exist in your repository that support the understanding of the article.

To create the definition of an article:

1. In the framework regulatory tree, select the article.
Buttons appear to its right.
2. Click **Properties**.
3. In its properties window, click the **Characteristics** page.
4. In the **Definition** section, click **New**.
The definition creation window opens.
5. Select the **Object type** that carries the definition, for example, a concept, and select from the **Short name** list the object in question..
6. Click **OK**.

See also:

With a data catalog you can specify which sections and articles of a regulatory framework apply to an organization. See [Regulations Associated with the Catalog](#).

Creating the Inventory of Control Directives

Control directives are an interpretation of the law and contribute to the enforcement of any regulation article your organization has to comply with.

In **HOPEX Data Governance** you can view the control directives in different ways:

- within the control frameworks that contain them
- in a tree structure that lists the control directives
- within the regulatory frameworks in which they appear
- in the UCF documents. See [Importing UCF Documents](#).

Implementation controls may be associated with the control directives. See: [Defining Data Quality Controls](#).

Accessing the list of control directives

To access control directives from a tree:

1. In the navigation menu, click **Compliance > Regulatory frameworks**.
2. In the edit area, click the **By Control Directives** tab.

First-level Implied control directives appear in this list only if one of their children is mandated.

Some columns indicate:

- whether the control directive constrains the organization
- the number of implementing controls associated

Supported and supporting directives

All control directives are displayed in a tree that enables to view their inter-relations.

The screenshot shows the HOPEX Data Governance interface. On the left, a tree of control directives is displayed. The tree is titled "Establish and maintain a Customer Information Management program." and lists several directives. A "Supported directive" is highlighted in green, and its "Supporting directives" are listed below it. A "Directive" is also highlighted in green. The interface includes a filter box and a table of directives with columns for Name and Enforcement Level.

Supported directive

Directive

Supporting directives

Establish and maintain a Customer Information Management program.

Characteristics

Name

Establish and maintain a Customer Information Management program.

Control Nature

Preventive

Enforcement Level

Implied Control

External Identifier(s)

84

Regulation Articles

Directives

Supporting Directive

Supported Directive

Name

Enforcement Level

Privacy protection for information and data

Mandated Control

Having several control directives in the **Supported Directive** tab of a control directive is highly beneficial. This means that efforts spent implementing one mandated control directive also contributes partly to the implementation of its supported control directives.

Enforcement level of control directives

There are three enforcement levels for each control directive:

- **mandated**

 a mandated directive is directly associated to a regulation article. It implements a regulatory framework.

- **implied**

 An implied directive is a non-mandated control directive that is a parent of a mandated directive. It indicates that one of the control directives contained within its supporting hierarchy is mandated.

- **implementation**

 An implementation directive is a non-mandated directive that is a child of a mandated directive. It provides details regarding how to carry out the mandated directive and facilitates its implementation.

Enforcement level for control directives	
Implied control directive	- Is not mandated - Contains at least a mandated control directive in its hierarchy - Allows to display the mandated control directives within the UCF hierarchy - Is supported by a mandated control directive
Implementation control directive	- Is not mandated - Appears under a mandated control directive (is supporting a mandated control directive)
Mandated control directive	- Is supporting an implied control directive - Can be supported by implementation control directives - Can support or be supported by other mandated directives

Associating a control directive with an article of regulatory framework

From an article, you can create a control directive or connect an existing control directive.

To create a control directive from an article:

3. In the framework regulatory tree, select the article.
Buttons appear to its right.
4. Click the **New > Control Directive** button.
The window for creating a control directive appears.
5. Enter the name of the control directive.
6. Click **OK**.

To connect an existing control directive to an article:

1. In the framework regulatory tree, select the article.
Buttons appear to its right.
2. Click the **Connect > Control Directive** button.
The search window appears.

3. Enter the name of the control directive or click directly on the **Find** button.
The list of control directives appears.
4. Select the control directive in question.
5. Click **Connect**.

Viewing articles associated with a control directive

To view articles associated with a control directive:

1. Open the properties of the control directive.
2. Click the **Characteristics** page.
3. Expand the **Fulfilled Regulation Articles**.

CREATING AN INVENTORY OF THE POLICY FRAMEWORKS (INTERNAL REGULATIONS)

A business policy is a directive whose purpose is to govern or to guide the enterprise.

In **HOPEX Data Governance**, a *Policy framework* defines a set of business policies.

It is composed of sections and sub-sections that represent categories of business policies. Under these sections you can define the business policies, the assets constrained by the policies in question and their implementation.

Example

A company's internal regulations are defined in a business policy framework.

This framework has different sections that represent policy categories: security, reception, cafeteria use, etc.

Each section has business policies: locker closing, table cleaning, etc.

Creating a Policy Framework

To create a policy framework:

1. In the navigation menu click **Compliance > Policy Frameworks**.
 2. In the edit area, click the **Policy Frameworks** tab.
 3. Click **New**.
 4. In the window that opens, enter the name and the description of the policy framework.
 5. Click **OK**.
- The framework appears in the list.

Adding a Section to the Policy Framework

To add a section to the policy framework:

1. In the navigation tree, click the icon of the framework then **New > Policy Framework Section**.

2. Specify:
 - the title
 - The description
 - (optional) the objects constrained by the section. It can be business data, physical data, application data, etc.
 - (optional) implementation: these are the business rules that implement the policies of the section. See [Ensure Compliance with Data: Create Business Rules](#).
3. Click **OK**.
The section appears in the navigation tree.

Adding a Business Policy

A business policy is a directive whose purpose is to govern or guide the company. A business policy serves as the basis for defining business rules and governing corporate processes. A business policy is always under the control of the company. It allows to control, guide and formalize the strategies and tactics of the company.

To add a business policy to a section:

1. In the navigation tree, click the icon of the section concerned then **New > Business Policy**.
2. Specify:
 - the name of the section
 - the title
 - the comment
 - (optional) the concept definitions that can be associated with key words of the text.
 - (optional) the objects constrained by the business policy
 - (optional) implementation: this concerns the business rules that ensure the effective implementation of the business policy. See [Ensure Compliance with Data: Create Business Rules](#).

Analyzing Information Constrained by a Regulation

To display information constrained by regulatory frameworks or business policy frameworks:

1. Click the navigation menu, then **Reports > Policies Reports**.
2. In the edit area, click the **Regulatory Framework Report** tile.
3. In the **Regulatory Framework List** box, click **Connect**.
4. Search and select the regulatory frameworks to be analyzed.
5. Refresh the report.

IMPORTING UCF DOCUMENTS

UCF (Unified Compliance Framework) is the largest library of regulatory content available today. It contains:

- Authority Documents
- Citations
- UCF Controls

HOPEX Data Governance provides an integration that allows you to access the Unified Compliance Framework (UCF) library of compliance documents and controls.

HOPEX Information Architecture combined with the Unified Compliance Framework (UCF) automates IT compliance initiatives by providing a simple way to manage and monitor compliance with a wide range of IT regulations and standards. UCF helps implement the appropriate controls for each regulation, while identifying controls common to multiple regulatory documents. This allows companies to avoid duplication of effort and reduce the time and cost of compliance.

HOPEX builds an inventory of relevant regulations by importing available UCF policies, requirements, and controls and keeps them up to date. It also provides out-of-the-box reports and dashboards to monitor compliance level, remediation plans, and their impact on the IT landscape.

For more details on UCF, see [About Unified Compliance Framework](#).

Using UCF Import

UCF Import Prerequisites

The OA Functional Administrator can download UCF content (authority documents, citations and controls) and update it.

To be able to import this content to **HOPEX UCF**, you must have:

- **HOPEX Data Governance AND HOPEX UCF**
- a UCF account and API key
- a Shared List with the Authority Documents you want to import.

☛ For more information, see [Unified Compliance Framework](#).

- parameterized UCF options in **HOPEX UCF**

☛ In the UCF Common Controls Framework, information is generally available in English.

If you want to use **HOPEX UCF** with **HOPEX** user data language other than English, you must:

- set up your data language of interest (example: if you want to use **HOPEX** with French as data language, make sure to set up French as data).
- import UCF data
- repeat the operation (change data language + proceed to import) as many times as desired languages.

Parameterizing UCF Import

To parameterize UCF import:

1. In the **Main menu**, select **Settings > Options**.
2. In the Options window, expand **Tools > Data Exchange > Import > UCF Common Controls Hub Integration**.
3. Select the **Activate UCF Import** check box.
4. Enter the URL corresponding to UCF API.

`https://api.unifiedcompliance.com/`

5. Enter your **UCF API Authentication Key**.

 To retrieve your API authentication key in your Unified Compliance Framework workspace:

- go to **Settings > API Manager > API Keys**.
 - **Create Credentials** and copy paste your API Key.
6. Click **OK**.

Importing Data from the Common Controls Hub

In **HOPEX Data Governance**, the functional administrator can import relevant data from the UCF Common Controls Hub (Authority Documents, Citations and Controls).

To import UCF data:

1. In the navigation menu, click **Compliance > Regulatory frameworks**.
2. In the edit area, click **Import UCF Content**.
3. Click **Next**.
4. Select the Shared List from your Common Controls Hub.
5. Click **Next**.
6. Select the Authority Document(s) you wish to import into **HOPEX**.

UCF Import - Regulatory Frameworks			
The Authority Documents contained in the previously selected Shared List are displayed below. Please select the Authority Document(s) you wish to import into HOPEX.			
☐	Name ↑	Already Present in HOPEX?	Latest Available UCF Update
☐	AICPA Reporting on Controls at a Service Organization SOC-2	No	9/9/2019
☐	Basel II	No	4/2/2020
☐	California Consumer Privacy Act of 2018	No	9/23/2019
☐	EU General Data Protection Regulation (GDPR)	No	9/11/2019

 If you update an already imported Authority Document, it may be useful to compare the columns **Latest available UCF updates** and **Last imported UCF update**.

7. Click **Next**.

Defining the Applicable Regulatory Content

Regulatory content relevance

All the articles/sections of an imported regulatory framework are not applicable to your organization.

Compliance officers can inspect the imported regulatory frameworks and specify which ones are applicable.

Only the applicable content will appear in **HOPEX** libraries for your stakeholders.

 The regulatory content you directly create in **HOPEX** is automatically considered as compliant (applicable).

Reviewing regulatory frameworks after UCF import

Once the UCF data has been imported, the tree structure appears in **Compliance > By Control Directives > UCF Import**.

It displays regulatory frameworks (UCF Authority Documents) and features regulation articles (Citations) along their enforcing control directives (UCF Controls). It is based on the supported/supporting structure originally defined by UCF.

From this tree you can:

- review the newly imported regulatory frameworks and their content.
- Indicate which pieces of regulatory content are deemed relevant to your organization.

Selecting relevant content for your organization

To declare regulatory content as relevant:

- 1 Expand the tree if necessary and select the check-box corresponding to the regulatory frameworks/articles/sections you must comply with.

	Applicable	Regulatory Children
 CobiT		40
  AC1 Source Data Preparation and Authorisation. Ensure ...	☒	0
  AC2 Source Data Collection and Entry. Establish that dat...	☒	0
  AC3 Accuracy, Completeness and Authenticity Checks. E...	☒	0
  AC4 Processing Integrity and Validity. Maintain the integ...	☒	0
  AC5 Output Review, Reconciliation and Error Handling. E...	☐	0
  AC6 Transaction Authentication and Integrity. Before pa...	☐	0
  PO1. Define a Strategic IT Plan	☐	1

 The grey square  means that the regulatory content below has been partially selected only.

Data corresponding to the regulatory content you have selected become available to Internal Controllers in the Control Framework libraries.

ENSURE COMPLIANCE WITH DATA: CREATE BUSINESS RULES

A business rule constitutes an action to ensure that an internal or external regulation is complied with within the organization.

For example, the banking regulation concerning money laundering requires that each banking establishment has up-to-date knowledge of its clients. To apply this obligation, the bank creates a business rule that consists of systematically analyzing the revenue and assets of its client via the transmission of a detailed questionnaire.

HOPEX Data Governance identifies three types of enterprise rule:

- business rule: the definition references business information (eg. concept)
- operational rule: the definition references logical data item (class, entity, data view)
- system rule: references a logical or physical data item (class, entity, table, data view, physical view)

Defining a Business Rule

To create a rule:

1. Click the navigation menu then **Compliance > Operational Assurance**.
2. In the edit area, click **Business Rules**.
3. Select the type of rule that you want to create: business, operational or system.
4. Click the **New** button.
5. In the window that opens, specify the name of the rule and the owner if appropriate.
6. Click **OK**.

To define content of the rule:

1. Open the properties of the rule.
2. In the **Characteristics** page, specify the text of the rule and any **definitions**.
3. In the **Application** page of the rule specify the means that are used to compliance with regulation selected.

For example, the "APIX" application applies the system rule "RS #1" which ensures implementation of section 3 of BCBS 230.

Rules Report

A report allows you to visualize the content of a rule.

To generate this report:

1. Click the navigation menu then **Reports > Policies Report > Rules Report**.
The report opens in the edit area.
2. Click **Connect**.
3. Select the type of rule (business, operational or system) and the rule you want to analyze.
4. Click **Connect**.
5. Refresh the report.

For each rule selected as input, the report shows:

- its description (comment)
- the regulation or policy schema it implements
- the object that applies it (process for a business rule, application for a system rule...)
-



DATA QUALITY



Data quality involves preparing data to meet the regulations and the needs of its business users to which the company is subject.

To do this, the company must define rules that establish the quality criteria and the objectives to be achieved for each of these criteria.

HOPEX Data Governance allows you to set up this data quality policy but also to build an evaluation and control repository to ensure that the data reaches the desired quality level.

- ✓ [Defining a Data Quality Policy](#)
- ✓ [Assess Data Quality](#)

DEFINING A DATA QUALITY POLICY

Creating a Data Quality Policy

Data quality policies are guidelines that describe what criteria data must meet to be useful and reliable within an organization.

A data quality policy is part of a Policy framework. It describes:

- the objects to which it applies
- the dimensions that constitute the quality criteria to be respected



A policy framework is an internal document issued by the organization. It defines a set of business policies. See [Creating an Inventory of the Policy Frameworks \(Internal Regulations\)](#).

Controls can be put in place to ensure that the quality policy is properly implemented. See [Defining Data Quality Controls](#).

Example

A school policy requires parents of new students to complete a form that includes a medical questionnaire and emergency contact information, as well as confirmation of the student's name, address and date of birth.

A data quality policy is defined: it must respect the "Completeness" dimension, with a threshold whose value must be equal to 100%.

To create a data quality policy in **HOPEX Data Governance**:

1. Click the navigation menu then **Compliance > Data Quality Policies**.
2. In the edit area, click **New**.
3. Specify:
 - the name of the data quality policy
 - a description
 - the subject, which represents the constrained asset
 - the dimension



A dimension of data quality is a measurable characteristic of the data. For more details, see [Quality Dimensions](#).

4. Click **OK**.
The data quality policy appears in the edit area.

Defining Data Quality Controls

Implementation controls may be associated with the control directives.



Control directives are an interpretation of the law and contribute to the enforcement of any regulation article your organization has to comply with.

To create a data quality control:

1. Click the navigation menu then **Compliance > Operational Assurance**.
2. In the edit area click **Data Controls**.
3. Click **New**.
4. Enter the name and an owner if necessary.
5. Click **OK**.
The control appears in the edit area.
6. Hover the cursor over the control and click the **Properties**  button to define its characteristics:
 - **Code**: enables unique identification of the control.
 - **Nature**: corrective, detective, preventive
 - **Frequency**: daily, weekly, monthly, at each transaction, bi-monthly
 - **Control directive** associated with the control



Control directives are an interpretation of the law and contribute to the enforcement of any regulation article your organization has to comply with.

- **Regulation articles** of a the regulatory framework concerned
- **Data Quality Measure**



Data Quality Measure is what should be enforced to ensure the quality of Data.

- **Execution Control**

Identifying Data Quality Issues

When data quality controls are found to be unsatisfactory, you can specify quality issues and implement action plans to address them.

To define a data quality issue:

1. Click the navigation menu then **Compliance > Operational Assurance**.
2. In the edit area, click the **Data Issues** tab.
The list of data issues appears.
3. Click **New**.
4. Name the issue and click **OK**.
The issue appears in the edit area.

Addressing the Data Quality Issues Encountered: Action Plans

You can set up action plans to improve a quality control that has been considered unsatisfactory.

To create an action plan:

1. Click the navigation menu then **Compliance > Operational Assurance**.
2. In the edit area, click the **Action Plans** tab.
The edit area displays:
 - All action plans
 - Delayed action plans
3. In the **All Action Plans** tab, click **New**.
4. Indicate the name of the action plan and possibly a comment as well as the start and end dates.
5. Click **OK**.
The action plan appears in the list.
6. Hover the cursor over the action plan and click the **Properties**  button to define its characteristics.

Properties of an action plan in HOPEX Data Governance

Main characteristics

You can specify the following information:

- **Name:** action plan name.
- **Owner:** this field is specified by default by the user who created the action plan.
- **Owner Entity:** entity responsible for action plan implementation.
- **Approver:** user responsible for validation of the action plan when all actions are completed.
- **Means:** text description of means required/desired for action plan execution.
- **Priority:** enables indication of a level. Priority can be:
 - "Low"
 - "Medium"
 - "High"
 - "Critical"
- **Origin:** enables definition of the context of carrying out the action plan:
 - "Audit"
 - "Compliance"
 - "Event"
 - "Risk"
 - "RFC"
 - "Other".
- **Category:** the action plan can for example be connected to:
 - risk impact reduction
 - project management
 - process improvement
 - control performance improvement
 - etc.
- *Other values are available.*
- **Nature:** enables definition of whether the action plan is:
 - Corrective
 - Preventive
- **Comment:** supplements information on the action plan and its characteristics.
- **Steering Calendar:** used for sending reminders to the person responsible for an action plan so that they can indicate action plan progress.
 -  *A steering calendar for monthly reminder of progress is supplied by default.*
- **Action Plan Status :** indicates at which stage of the workflow the action plan is located.

Financial assertion

- **Forecast Cost:** action plan cost estimate.
- **Forecast Cost (Man-Days):** estimate in man-days of action plan implementation workload.

Responsibilities

The user defined as action plan **Responsible** is responsible for definition of actions to be carried out and their execution.

This section is defined by the action plan creator or the action plan approver.

Success factors

In the **Success Factors** section, you can specify in text the success indicators enabling assessment of success of the action plan.

Scope?

To position an action plan in its environment, you can associate objects with the action plan in the **Scope** section.

You can connect objects of the following types:

- controls
- applications
- risks
- entities
- processes
- problems

Milestones

Milestones are key dates of the action plan.

☛ *The planned end date is mandatory.*

Actions

The owner of the action plan must define actions enabling execution of the action plan. The owner can create actions and assign these.



An action is included in an action plan and represents a transformation or processing in an organization or system.

7. In the **Actions** section, click **New**.
8. Fill in the columns associated with the action.

☛ *For more details on actions, see [Managing Actions](#).*

Attachments

You can attach business documents to an action plan:

☛ *For more details on the use of business documents, see the **HOPEX Common Features** guide.*

ASSESS DATA QUALITY

An evaluation can be carried out directly on data or remotely via questionnaires.

HOPEX Data Governance provides by default a data assessment template that focuses on the following dimensions:

- completeness
- accuracy
- consistency
- validity
- uniqueness
- freshness

Quality Dimensions

A dimension of data quality is a measurable characteristic of the data. The data quality dimensions establish data quality requirements.

HOPEX Data Governance delivers a number of predefined dimensions. To display the list of predefined dimensions:

1. In the navigation menu, click **Compliance > Data Quality Policies**.
2. In the edit area, click the **Data Quality Dimensions** tab.
3. Click the **Add predefined dimensions** button.

Dimension	Description																
Accessibility	Reflects the ease of access and use of the data, at a practical level (quick, without outside intervention).																
Timeless	Indicates the extent to which the data represents reality as of the required time. Timeliness of data implies that the data has been updated as necessary to remain relevant.																
Consistency	Identifies the level of consistency in the data, the lack of difference when comparing two or more representations of a thing to a definition. Example Below is an inconsistency in the data format. <table><tr><th>Order Number</th><th>Client Id</th><th>ShipDate</th><th>Total</th></tr><tr><td>1000</td><td>1</td><td>1/12/2018</td><td>100\$</td></tr><tr><td>1001</td><td>2</td><td>1/12/2018</td><td>200£</td></tr><tr><td></td><td></td><td></td><td></td></tr></table>	Order Number	Client Id	ShipDate	Total	1000	1	1/12/2018	100\$	1001	2	1/12/2018	200£				
Order Number	Client Id	ShipDate	Total														
1000	1	1/12/2018	100\$														
1001	2	1/12/2018	200£														

Dimension	Description																				
Completeness	Identifies the level of completeness of data and missing properties. Example: Below some columns have no value (in red) and others are truncated (Dupont@Samp.gm) <table><tr><th>First Name</th><th>Last Name</th><th>Billing Address</th><th>Shipping Address</th><th>Email</th></tr><tr><td>Dupont</td><td></td><td>9 rue Rene Coty Paris 75002</td><td>NULL</td><td>Dupont@Sample.gm</td></tr><tr><td>Durand</td><td>Robin</td><td>344 rue de Rivoli 75001</td><td>NULL</td><td>Durand@Sample.com</td></tr><tr><td></td><td></td><td></td><td></td><td></td></tr></table>	First Name	Last Name	Billing Address	Shipping Address	Email	Dupont		9 rue Rene Coty Paris 75002	NULL	Dupont@Sample.gm	Durand	Robin	344 rue de Rivoli 75001	NULL	Durand@Sample.com					
First Name	Last Name	Billing Address	Shipping Address	Email																	
Dupont		9 rue Rene Coty Paris 75002	NULL	Dupont@Sample.gm																	
Durand	Robin	344 rue de Rivoli 75001	NULL	Durand@Sample.com																	
Confidence	Are data governance, data protection and data security in place? What is the reputation of the data, and is it verified or verifiable?																				
Accuracy	Identifies the level of accurate, reliable data. Example: Below, for Dupont, the position and the department are reversed. For Durand, the item displays a typographical error For Rene, the department displays an erroneous value. <table><tr><th>First Name</th><th>Position</th><th>Department</th><th>Email</th></tr><tr><td>Dupont</td><td>Product Management</td><td>Business Analyst</td><td>Dupont@Sample.gmail</td></tr><tr><td>Durand</td><td>Sftware Engineer</td><td>Product Development</td><td>Durand@Sample.com</td></tr><tr><td>René</td><td>Test Analyst</td><td>xxùpoi*£</td><td>Rene@Sample.com</td></tr></table>	First Name	Position	Department	Email	Dupont	Product Management	Business Analyst	Dupont@Sample.gmail	Durand	Sftware Engineer	Product Development	Durand@Sample.com	René	Test Analyst	xxùpoi*£	Rene@Sample.com				
First Name	Position	Department	Email																		
Dupont	Product Management	Business Analyst	Dupont@Sample.gmail																		
Durand	Sftware Engineer	Product Development	Durand@Sample.com																		
René	Test Analyst	xxùpoi*£	Rene@Sample.com																		
Freshness	This criterion assesses whether the information is available at the required time. The freshness of the data is essential to have a good view of a situation at a given time and to make decisions about the data. Freshness is important in two ways: a short delay between the data collection and its analysis and a short delay between the reporting and the resulting optimization or correction action.																				
Relevancy	Relevance of data refers to the extent to which the data meets the needs of users. Information needs may change and is important that reviews take place to ensure data collected is still relevant for decision makers.																				
Data Security	Data security covers the notion of empowerment (authorization of access to sensitive data), measures taken against the loss of information; controlling the risk of sensitive information leaks.																				
Traceability	Traceability makes it possible to follow the progress of information from its collection to its return, including its processing. Very often it is associated with the history of a process or a product.																				

Dimension	Description																
Uniqueness	This criterion assesses the level of uniqueness of the data. Example: The "Client" table must not contain the same occurrence twice, each record must be unique.																
Usability	Is the data understandable, simple, relevant, accessible, maintainable and at the right level of precision?																
Value	The value of the data reflects its worth: is there a good cost/benefit ratio for the data? Are they being used optimally? Do they jeopardize the safety or privacy of individuals or the legal responsibilities of the company? Do they support or contradict the company's brand image or message?																
Validity	Identifies the level of valid data. Data are valid when they conform to the syntax (format, type) of their definition. Example: The value of the "Available units" field on Prod1 should not be negative. A withdrawal date is set to Prod2 but the field "Available units" does not display a null value. <table><tr><th>Product Code</th><th>Name</th><th>Units Available</th><th>Retire Date</th></tr><tr><td>1000</td><td>Prod1</td><td>-10</td><td>12/4/2020</td></tr><tr><td>1001</td><td>Prod2</td><td>100</td><td>31/12/2017</td></tr><tr><td></td><td></td><td></td><td>-</td></tr></table>	Product Code	Name	Units Available	Retire Date	1000	Prod1	-10	12/4/2020	1001	Prod2	100	31/12/2017				-
Product Code	Name	Units Available	Retire Date														
1000	Prod1	-10	12/4/2020														
1001	Prod2	100	31/12/2017														
			-														
Reasonability	Reasonability asks whether Data are in the correct ranges, for example check the max and mins, the Distribution and the outliers. HOPEX provides an Excel template that allows you to evaluate the data criteria in a file and import them into your repository.																

Evaluation Objects

An evaluation can deal with the following objects:

- concept, concept view
- class, data view
- table, physical view
- all data areas (business, logical and physical)

Direct Assessment

To directly assess a data item:

1. Open the properties of the data item in question.

2. Select the **Evaluation** page.
3. Click **Evaluate**.
4. On the page that appears, select a value for each question. See [Quality Dimensions](#).
5. Click **OK**.

Assessment By Campaign

The functional administrator can create evaluation campaigns or sessions for data. On creation of a campaign, questionnaires are sent to designated respondents to obtain qualitative estimations on the objects for which they are responsible. For more details on campaigns and sessions, see [Managing Assessment Campaigns](#).

Prerequisites for Data Evaluation

Before starting a data evaluation campaign, you must first prepare the work environment. Ensure that you have defined respondents for the data, and specify for each one the entity to which he/she is attached as well as an email.

Creating assessment campaigns

To create an evaluation campaign with the template provided as standard:

1. Click the navigation menu, then **Campaign Management > List of Campaigns**.
2. In the edit area click **New**.
The campaign creation page appears.
3. Enter the name of the campaign.
4. Select **Evaluation Template** "Data Quality Evaluation"..
5. Modify the **Calendar** if required.

 *The calendar serves to initialize the begin and end dates of the evaluation campaign.*

6. Specify the **Begin Date** and the **End Date**.
7. Click **Next**.
8. In the **Scope Selection** window, select the objects that define the evaluation context.
The context encompasses the elements of the branch that extends from the object in question up to the root.

 *If you deselect a node of a branch, only the child elements of this branch are deselected.*

9. Click **Next**.

10. In the preview window, click **Refresh the Report**.
Elements that will be assessed appear.
In particular, you can view:
 - evaluated characteristics (defined in the evaluation template)
 - evaluated objects
 - the context objects
 - evaluation nodes which correspond to objects placed in their context objects, associated with respondents.
 - respondents
11. Click **OK**.

For more information on campaigns, see [Managing Assessment Campaigns](#).

See also:

- ✓ [Importing Data Assessments](#)

Data Quality Report

The quality report provides a summary of the quality of the data of an information area (application, business, logical, physical area) using the data evaluation results.

➡ See [Quality Dimensions](#).

To generate the data quality report on an information area:

1. In the navigation menu, click **Reports > Data Assurance Report**.
2. In the edit area, click the **Data Quality Report** tile.
The report settings appear.
3. Click **Connect** to define the information area(s) to be analyzed.
4. In the search window, select the area type (business, logical, etc.) then the information area in question.
5. Click **Connect** then refresh report.



DATA REPORTS AND ANALYSIS TOOLS



HOPEX Data Governance offers different types of reports designed to analyze the business data defined in the repository.

➤ *For more details on operation of reports, see the HOPEX Common Features guide, "Generating Reports".*

➤ *Reports on the diagrams available in standard mode with **HOPEX** are also accessible with **HOPEX Data Governance**.*

ANALYSIS REPORT

The analysis reports are dynamic reports that enable repository data analysis according to different perspectives.

Accessing HOPEX Data Governance Reports

To access **HOPEX Data Governance** reports:

- 1 Click the navigation menu, then **Reports**.

Some analysis reports are also embedded in the repository objects. These reports are available in the properties of these objects, in the **Reports** page.

For example, the reports displayed under **Reports** > **Description Reports** > **Data Domain Map** are also accessible in the properties of a data domain map.

Description Reports

The View Report

See [The View Report](#).

Glossary Report

HOPEX Data Governance provides a ready-to-use glossary report to automatically build the business glossary of terms derived from a set of Business dictionaries. For each term, the glossary displays a list of associated definitions with their text, synonyms and components list.

Report parameters

This consists of defining report input data.

Parameters	Parameter type	Comment
List of libraries	Library	Not mandatory
List of business dictionaries	business dictionary	Mandatory if no library
Example option	yes or no	Used to display the business information examples
Show the component type icon	yes or no	

Report example

The example below shows the terms from the "Media Library" dictionary.

 Media Library Vocabulary	
(piece of) work	 Concept 1. An artistic creation, such as a painting, sculpture, or literary or musical composition; a work of art. (Synonyms) Sample - Oakland City::Oakland City - Publishing Editor::Publishing Secteur Vocabulary::Work (Hypernyms) Artistic Product (Hyponyms) Literary composition, Musical composition, Sample - Oakland City::Oakland City - Publishing Editor::Publishing Secteur Vocabulary::Literary work, Sample - Oakland City::Oakland City - Publishing Editor::Publishing Secteur Vocabulary::Musical work, Sample - Oakland City::Oakland City - Publishing Editor::Publishing Secteur Vocabulary::Cinematographic work (Concept Type) Type of Work (Defined Architecture Products)  Work (Component 1) Work Title (Type) (Presence) Always (Cardinality) 1 (Component 2) Publication event: Date where a Published work is offered for distribution. (Type) Publication event (Presence) Always (Cardinality) 1
Artistic Product	 Concept 1. The abstraction of Body of work and Work. (Hyponyms) Oeuvre, Body of work, Works, (piece of) work, Sample - Oakland City::Oakland City - Publishing Editor::Publishing Secteur Vocabulary::Work (Component 1) Author: An author is the originator of any created Artistic Product. (Type) Person (Presence) Always (Cardinality) *

Data Domain Map

On a data map, two report templates allow you to visualize the hierarchy of the domains that make it up.

Data Domain Tree Map Report

In this report, filters allow you to view:

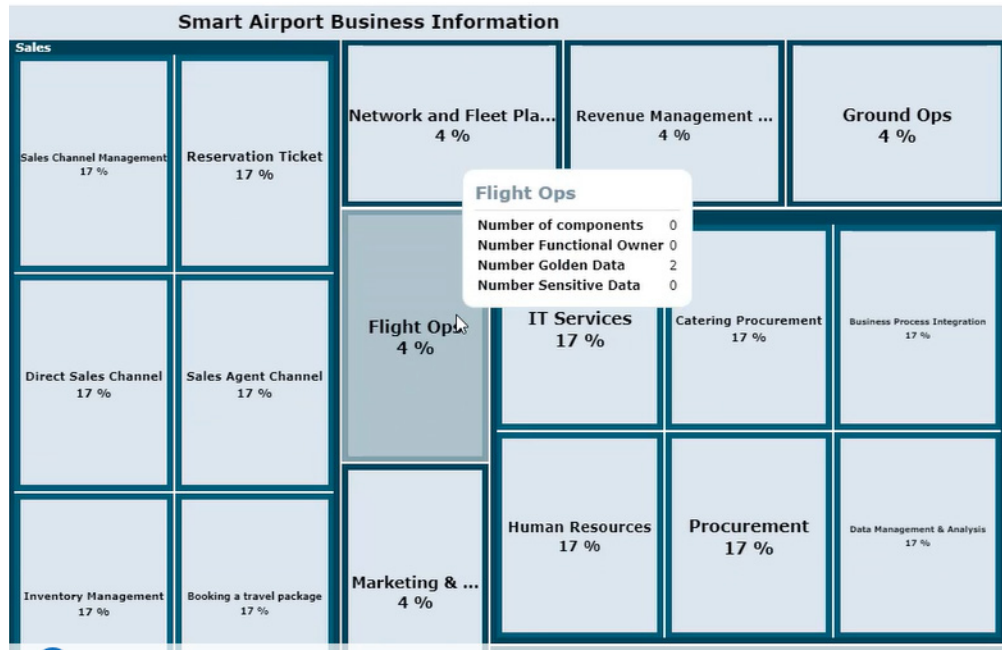
- the number of components in each domain
- the number of functional owners



The functional owner has a responsibility for the use of a data in a domain. You can specify the functional owners of the components of a domain in the properties of the domain in question, under the

Components section. See [Managing the components of a business information area](#).

- the number of reference data
- the number of data declared as sensitive.



Data Map Breakdown Report

This report also displays the domains that make up the map. The following information is available for each domain:

- the number of sensitive data
- the number of reference data
- average data quality

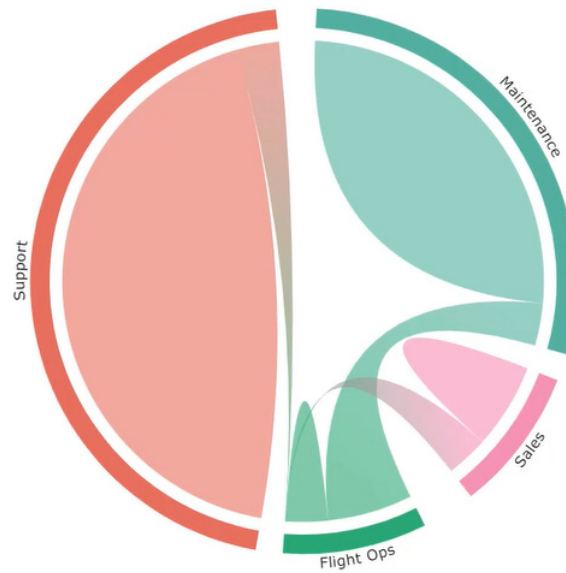
Relization Chart

See [Displaying the Realization Chart](#).

Data Domain Dependencies

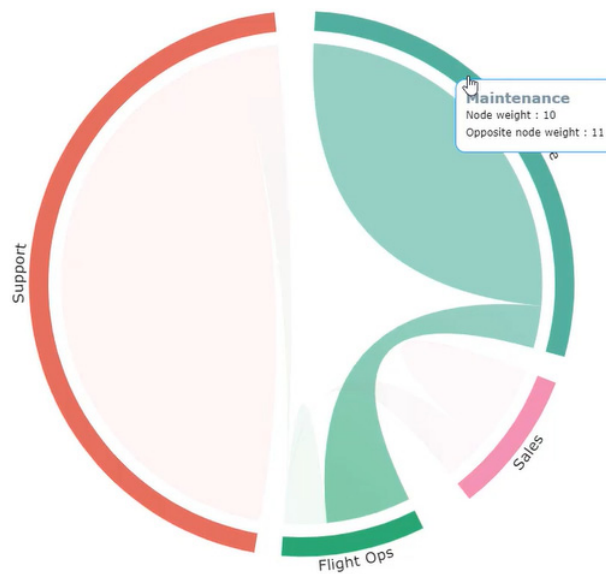
Based on a data map, this report presents the structural dependencies of the data used between data areas in the form of a string graph.

It is possible to dive into a data domain and view the dependencies between sub-domains where they exist.



By selecting a domain, its links with the other domains are highlighted.

Below you can see the links between the "Maintenance" data domain and the "Flight Ops" data domain.



Data Categories Dendrogram

The Data Category Dendrogram report allows you to view where data in a category is used in the application inventory.

To launch this report:

1. Click the navigation menu then click **Reports > Description Reports**.
2. In the edit area, click the **Data Categories Dendrogram** tile.
3. Select the relevant data categories.
4. Refresh the report.

Word Cloud Reports

Amount of Information in Information Areas

This report template relates to areas of an information container.

In the generated report, the size of the area name is proportional to the number of information that compose the area.

Extent of the Description of the Information

This report template relates to information containers (business dictionary, data dictionary, database) and display the corresponding elements: concepts, classes, tables, etc.

In the report, the size of the element name is proportional to the number of information that characterize it (for example attributes and relations that characterize a class).

Use of Information in Data Area

This report template relates to information containers (business dictionary, data dictionary, database) and display the corresponding elements: concepts, classes, tables, etc.

In the report, the size of the element name is proportional to its use in the information areas.

Data Usage Reports

Use of information held by a container

For the object selected as input (eg: a package), the report displays:

- the information that it holds (eg: classes or data views)
- the areas that use this information, with which access rights
- the applications that use the areas (via data stores that are used to declare them on the applications, application systems, application

service or micro-service), and through which components (in read-only or read/write).

A report template presents this information in the form of a dendrogram, another report template in the form of a table.

Report parameters

This consists of defining report input data.

Parameters	Object types
Subject	Business dictionary Database Package Data Model Catalog of NoSQL data building block Data type packages

Use of information in a domain

For the object selected as input (an area), the report displays:

- the information used (eg: classes or data views in the case of an application data area), with which access rights.
- in which applications the selected area is used (via data stores that are used to declare them on the applications, application systems, application service or micro-service), and through which components (in read-only or read/write).

Report parameters

This consists of defining report input data.

Parameters	Object types
Subject	Application data area Logical Data Area concept domain File structure Relational data area NoSql data area

A report template presents this information in the form of a dendrogram, another report template in the form of a table.

Use of information of an information map

The input parameter is an information map that can group together one or more area(s).

This report displays

- the areas and the data that it uses, with the access rights for this data.
- in which systems (applications, application systems, application services or micro-service) the areas are used and with which components of these systems, specifying the access mode (read-only or read/write).

A report template presents this information in the form of a dendrogram, another report template in the form of a table.

Use of information

The root object of the report consists of an item of information (concept, class, data view, table, etc.). For this root, the report displays:

- the areas that use this information, with which access rights
- in which systems (applications, application systems or micro-service) these areas are used and with which components of these systems, specifying the access mode (read-only or read/write).

Report parameters

This consists of defining report input data.

Parameters	Object types
Subject	Class Concept State concept Event concept Concept type Entity Period type Representation type Table Datatype Concept view Data view Physical view

Use of information of the domains of a container

For the object selected as input (a container), the report displays:

- the areas owned
- the domains used by the areas, with which access rights
- the systems that use these areas and the access mode of the components of these systems (read-only or read/write)

Report parameters

This consists of defining report input data.

Parameter	Object types
Subject	Business dictionary Database Package Data Model Catalog of NoSQL building blocks Data type package

Regulatory Reports

Regulatory Framework Report

This report displays the list of information constrained by a regulatory framework.

Constrained objects are those for which a regulatory framework has been associated in the **Regulations** page of their properties (for example, a concept).

 You can see which objects are constrained by a regulatory framework in the properties of that regulatory framework, on the **Subjected Elements** page.

The report displays the articles and sections of the regulation and highlights those that are constraints on the data.

Parameters

Regulatory Framework

Basel II x

Refresh the report

Regulation	Regulation Explanation	Constrained Data	Definition of Constrained Data	Data
§ 115. The organization should implement robust procedures and policies to control residual risks.	Residual risk is the amount of risk that remains in the process after all the risks have been calculated, accounted, and hedged. During an investment or a business process, there are a lot of risks involved, and the entity takes into consideration all such risks	Potential loss	gradual decline in amount or activity	Data Data
§ 163. The organization should have implemented processes to ensure it complies with all internal policies, procedures, and controls.	Internal policy means a Department statement that implements, interprets, or prescribes procedure or practice requirements for the internal operations of the Department	Customer Management		Data Data
§ 165, § 443, § 620(f), § 744. The internal auditing process should include an independent review of the risk measurement system. Internal audit must review the organization's rating system and operations at least annually. The internal audit should review the organization's adherence to applicable minimum requirements and document all of its findings. Internal or external auditors should regularly review the internal assessment process and the validity of the assessments.				

Report parameters

This consists of defining report input data.

Parameter	Object type
Regulatory Framework List	Regulatory Framework

Rules Report

This type of report allows you to select a set of rules (business, operational or system) and view in tabular form the regulations they implement.

See [Ensure Compliance with Data: Create Business Rules](#).

Data Catalog Reports

Data Catalog Report

The data catalog report is used to analyze the content of one or more data catalog(s). It presents the following chapters:

- Organizations at stake: entities that use or consume the data in the catalog.
- Regulatory Frameworks and Business Policy Frameworks
- Categories/Sections
- Data lineage. See also [Data Lineages](#).

Data Quality Report

This report presents the quality of information area data along dimensions such as consistency, accuracy, etc.

See also: [Quality Dimensions](#).

REPORT DATASETS

A Report DataSet is a data table created using repository objects, on which instant reports can be generated.

HOPEX Data Governance provides different types of Report DataSets.

Creating a Report DataSets

To create a Report DataSet

1. Click the navigation menu, then **Reports**.
2. In the navigation pane, click **Other Reports**.
3. In the edit area, click **My Report DataSets**.
4. click **New**.
5. Specify:
 - the name of the report
 - the holder (optional)
 - the definition of the Report DataSet, on which the report is based
6. Click **OK**.

Example of a Report Dataset

Definition of terms used

This type of Report DataSet has a list of terms as input parameter.

Using selected terms, you can, for example, create a "matrix" type instant report that presents the list of concepts that use the terms in question.

The screenshot displays two panels from a software application. The left panel, titled 'Matrice', shows configuration options for a matrix report. It includes dropdowns for 'Ligne:' (set to 'Concept'), 'Colonne:' (set to 'Term'), and 'Cell Display:' (set to 'Value'). Below these are options for 'Appliquer le calcul sur:' (set to 'Concept') and 'Calculer:' (set to 'Count'). A central area shows a matrix with rows for concepts: 'Book subscription (EN)', 'Work (EN)', 'compte cible', and 'Virement Planifié'. The columns represent terms, with 'Virement Planifié' having a value of 1 in the 'compte cible' row. The right panel, titled 'Propriétés de Report DataSet-3', shows the 'Paramètres' section with a 'Term List' containing the same four concepts. Below this is the 'Report DataSet' section with buttons for 'Rafraîchir', 'PDF', 'Excel', and 'Rapport instantané'. A table at the bottom lists the terms and concepts included in the report.

To create an instant report for this type of Report DataSet:

1. Create a "Term definition" Report DataSet type.
➡ See above [Creating a Report DataSets](#).
2. Open the properties of the Report DataSet.
3. Display the **Data** page.
4. (If needed) In the **Parameters** section, click **Add** and select the input parameters, here the terms.
5. (If needed) In the **Report DataSet** section, click **Refresh**.
6. In the **Report DataSet** section, click **Instant Report**.
7. Select the required instant report, here a matrix.
8. Click **OK**.

Business dictionary x Concept Matrix

A concept can be referenced by one or more business dictionaries.

This Report DataSet has a list of business dictionaries as input. It is used to create a "Matrix" type business report that lists the concepts referenced in the selected business dictionaries.

To create an instant report for this type of Report DataSet:

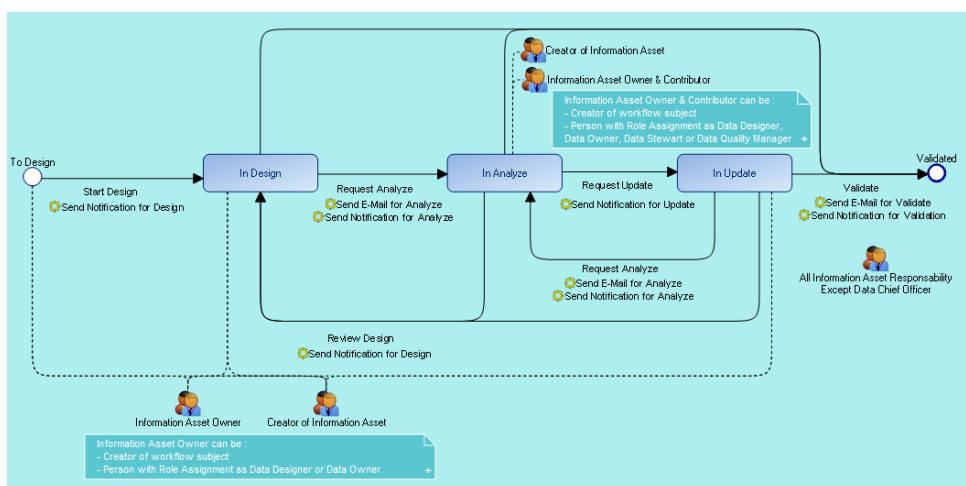
1. Create a "Business Dictionary Matrix x Concepts" type Report DataSet.
➡ See above [Creating a Report DataSets](#).
2. Open the properties of the Report DataSet.
3. Display the **Data** page.

4. (If needed) In the **Parameters** section, click **Add** and select the input parameters, here the terms.
5. (If needed) In the **Report DataSet** section, click **Refresh**.
6. In the **Report DataSet** section, click **Instant Report**.
7. Select the required instant report, here a matrix.
8. Click **OK**.



The workflow definition is provided by default on the data lineage, data domain and data map, for the business, logical and physical layers.

By default, the instantiation of this workflow is automatic when a data lineage is created. It is optional and not automatic for data domains and maps. You can select and decide on which object you want to create a workflow instance.



Workflow instantiation can be performed by the data designer or the owner of the object data. In addition, all other persons assigned as Data Asset Manager, Data Steward, Data Quality Manager and Data Chef Officer can use the "Data Contributor" profile to connect to HOPEX data and trigger the validation steps.

For each transition triggered, a notification or e-mail is sent to the assigned person on the subject of the workflow.

For more details on workflows, see [Using Workflows](#).

For more details on business roles, see [Business Roles of HOPEX Data Governance](#).

Generating a workflow report

Workflow reports are available on the different types of objects (data lineage, data domains, etc.); they display the number of objects located at each step of the workflow (number of objects in design, in analyze, etc.).

To generate a workflow report:

1. Click the navigation menu then click **Dashboard**.
2. Click + to add a report.
3. Expand the **Information Architecture** folder, then the **Design Status** sub-folder.
4. Select the report concerned.
The report appears in your dashboard.

DATA IMPORT AND EXPORT



HOPEX Data Governance and **HOPEX Information Architecture** provide Excel file templates so that you can import into the HOPEX repository existing business and logical data, from which you can build business dictionaries automatically. You can also use these templates to export data from the HOPEX repository.

A template dedicated to data quality allows to collect measures of data quality criteria (Completeness, Unicity, Timeless, etc.).

- ✓ [Importing Business Data from an Excel File](#)
- ✓ [Importing Logical Data from an Excel File](#)
- ✓ [Importing Data Assessments](#)

IMPORTING BUSINESS DATA FROM AN EXCEL FILE

HOPEX Data Governance and **HOPEX Information Architecture** provide an Excel file template so that you can import a set of existing business information into the HOPEX repository.

You can use the template to simply import a list of terms and their definitions, in order to generate a glossary, or for a more detailed description of business information, with the ability to define the relationships between concepts, their synonyms, etc.

You can also use this template to export business data from the repository.

Downloading the Excel File Template

To download the Excel template associated with the business data:

1. On the **HOPEX Information Architecture** desktop, click **Main menu > Export > Excel**.
2. In the wizard that appears, check the option "From a template".
3. Click **Next**.
4. In the **Predefined Template File** field, select "Concept Template".
5. Click twice on **Next** and save the created file. It contains the structure provided by the model.

Content of the Excel Template

The template contains the following sheets that interact with each other:

- Business Dictionary
- Data Category
- Term
- Concept
- Synonym
- Hyperonym
- Component
- State concept

Term Sheet

The **Term** sheet allows you to import a set of terms with their name, language and definition.

It contains the following columns:

Term_Ident

This property identifies the term. You must define this property when several terms have the same Term_name.

Term_Name

This property defines the name of the term.

Business Dictionary

In this column you must indicate the name of the business dictionary in which the imported Terms, Concepts and other business objects will be created.

It is not possible to specify different business dictionary names in the same Excel file. It is also important to enter the same name in all object lines of the sheets to be imported.

Language

This property indicates the abbreviation of the language associated with the term, for example FR, EN, etc. This abbreviation is used to identify the language of the object in HOPEX.

When you click in the corresponding column, a list of languages is proposed.

Text Definition

This property contains the definition of terms and is used to create in HOPEX the concepts that correspond to the terms entered in the sheet.

The **Text Definition** property is to be completed when only the **Term** sheet is used or when the term concerned has only one definition. If the term has several definitions, they must be declared in the **Concept** sheet. Thus each of the concepts refers to the associated term (via its identifier declared in the **Term** sheet) and carries its definition in the **Text Definition** property of the **Concept** sheet (see below the example of the Concept sheet).

Example of Term sheet:

B	C	D	E	
Term_Name	Term_Ident	Business Dictionary	Language	Text Definition
#0000000040000063	9B089CFF5C947952	C69DCA055C931225	C69DC9445C9311DC	869F9ABB5CA43A1E
Order	T0	Import Business Glossary	EN	
club	T1	Import Business Glossary	EN	
society	T2	Import Business Glossary	EN	

Concept Sheet

The **Concept** sheet allows you to link concepts to the terms defined in the **Term** sheet, and to give their definition.

Concept_Ident

This property identifies the concept that corresponds to the term.

Term_Ident

This property identifies the term from which the concept is derived and which is defined in the **Term** sheet. You must define this property when several terms have the same Term_name.

Term_Name

This property defines the name of the term.

Text Definition

This property contains the definition of the term from which the concept is derived.

Example:

The sheet below gives all the concepts and definitions associated with the term "Order" (defined in the terms sheet with the identifier "T0"):

B	C	D	E	
Concept	Term_Name	Term_Ident	Business Dictionary	Text Definition
EE9B089CF7C014905C	F7C014905C	F7C0132E5C	C69DCA055C931225	43584EB15CB8739F
C0		T0	Import Business Glossary	(often plural) a command given by a superior (e.g., a military or law
C1		T0	Import Business Glossary	a degree in a continuum of size or quantity; "it was on the order of
C2		T0	Import Business Glossary	established customary state (especially of society); "order ruled in
C3		T0	Import Business Glossary	logical or comprehensible arrangement of separate elements; "we
C4		T0	Import Business Glossary	a condition of regular or proper arrangement; "he put his desk in or
C5		T0	Import Business Glossary	a legally binding command or decision entered on the court record
C6		T0	Import Business Glossary	a commercial document used to request someone to supply someth

Data Categories

This property indicates the category of the data. It is possible to indicate several categories in the cell by performing a carriage return between each category. If the category does not exist, it is created.

Information Item Sheet (Concept Properties)

The **Information Item** sheet contains the information items (properties) of the concepts associated with the terms.

Information item_Ident

This property identifies the information item.

Term_Name

This property is optional; it is used for information purposes. It gives the name of the term.

Term_Ident

This property identifies the term associated with the information item, which is defined in the **Term** sheet. You must define this property when several terms have the same Term_name.

Business Dictionary

In this column you must indicate the name of the business dictionary in which the imported information items will be created.

It is not possible to specify different business dictionary names in the same Excel file. It is also important to enter the same name in all object lines of the sheets to be imported.

Text Definition

This property contains the definition of the term from which the concept is derived.

Data Categories

This property indicates the category of the data. It is possible to indicate several categories in the cell by performing a carriage return between each category. If the category does not exist, it is created.

Synonym Sheet

The **Synonym** sheet allows you to link terms defined in the **Term** sheet to definitions entered in the **Concept** sheet, by designating the terms as synonyms of the definition (the definition being carried by a concept).

Concept_Ident

This property identifies the concept that corresponds to the term.

Concept_Name

This property is optional; it is used for information purposes. It gives the name of the concept.

Term_Ident

This property identifies the term associated with the synonym, which is defined in the **Term** sheet.

Term_Name

This property is optional; it is used for information purposes. It gives the name of the term.

Component sheet

The **Component** sheet is used to define the relationships between concepts.

Concept_Component_Ident

This property identifies the concept components.

Term_Ident

This property is to be defined if the component is to designate the term. By default the component name is initialized from the referenced concept.

Owner_Concept_Ident

This property is mandatory; it identifies the owner concept.

Referenced_Concept_Ident

This property is mandatory; it identifies the referenced concept.

State Concept sheet

The **State Concept** sheet allows you to import a set of concept states associated with terms.

Concept_State_Ident

This property identifies the concept state to be imported.

Term_Ident

This property identifies the associated term, which is defined in the **Term** sheet.

Concept_State_Name

This property is optional; it is used for information purposes. The name of the concept state is derived from the corresponding term.

Text Definition

This property contains the definition of the associated term.

StateOf_Ident

This property identifies the concept of the state. If this property is null, the concept state is created without a concept.

See also: [Importing Logical Data from an Excel File](#).

IMPORTING LOGICAL DATA FROM AN EXCEL FILE

HOPEX Data Governance and **HOPEX Information Architecture** provide an Excel file template so that you can import a set of existing logical information into the HOPEX repository. You can also use this template to export data from the repository.

Downloading the Excel File Template

To download the Excel template associated with the logical data:

1. On the **HOPEX Information Architecture** desktop, click **Main menu** > **Export** > **Excel**.
2. In the wizard that appears, check the option "From a template".
3. Click **Next**.
4. In the **Predefined Template File** field, select "Data Excel Template".
5. Click twice on **Next** and save the created file. It contains the structure provided by the model.

Content of the Excel Template

The template contains the following sheets:

Data Dictionary sheet

The **Data Dictionary** sheet allows you to import a set of data dictionaries with their name and owner.

Data Dictionary Short Name

Name of the data dictionary. This property is mandatory.

Data Dictionary ID

The ID is used to identify the business dictionary, in the event that several dictionaries have the same name. This property is optional.

Data Dictionary Owner Name

Name of the holding data dictionary. This property is optional.

Data Dictionary Owner ID

The ID is used to identify the holding business dictionary, in the event that several have the same name. This property is optional.

Comment

Comment of the data dictionary. This property is optional.

Data Type sheet

The **Data Type** sheet defines the data types to be imported.

Data Type Short Name

Name of the data dictionary. This property is mandatory.

Data Type ID

The ID is used to identify the data type, in the event that several have the same name. This property is optional.

Data Type Package Name

Name of the package that holds the data type. This property is optional.

In the case of a hierarchy in detention, the syntax of the name is as follows: <Name of the holding package 1>::<Name of the holding package 2>

Example:

```
Standard::Types::Data Types Reference
```

Data Type Package ID

The ID is used to identify the data type package, in the event that several have the same name.

Length

Length of the data type.

Decimal

"Decimal" value.

Data Type Type

You can associate a standard type with the type of data to be imported, for example "P-Character".

Comment

Comment of the data type.

Data Type Component sheet

The **Data Type Component** sheet defines the attributes of the data types defined in the **Data Type** sheet.

Attribute Short Name

Name of the data type attribute. This property is mandatory.

Example: "Number".

Owner Data Type Name

Name of the holding data type. This property is mandatory if the ID of the holding data type is not specified.

Example: "Address".

Owner Data Type ID

ID of the holding data type. This property is mandatory if the ID of the data type is not specified.

Data Type Name

Name of the attribute data type.

Example: "Number".

Data Type ID

ID of the attribute data type.

Length

Length of the attribute data type.

Decimal

"Decimal" value.

Comment

Comment of the data type attribute.

Class sheet

The **Class** sheet enables to define classes to be imported. Enter the name of the class. Enter the class ID when different classes have the same name.

Class Short Name

Name of the class. This property is mandatory.

Class ID

The ID is used to identify the class, in the event that several have the same name. This property is optional.

Data Dictionary Name

Name of the data dictionary that holds the class. This property is optional.

Data Category

You can give the class a classification, e.g. "sensitive data", "reference data", and so on.

Comment

Comment of the class.

Attribute sheet

The **Attribute** sheet defines the attributes of the classes defined in the **Class** sheet.

Leave the first column blank if you want to create new occurrences, or fill it in with the Absolute Hopex ID if you want to update existing data.

Short Name

Name of attribute.

Class Name

Name of the holding class. The name or the ID of the class is mandatory.

Class ID

ID of the holding class. The name or the ID of the class is mandatory.

Length

Length of the attribute.

Decimal

"Decimal" value.

Comment

Comment of the attribute.

Relationship sheet

Part Name

Name of the part.

Owner Class Name

Name of the holding class. The name or the ID of the class is mandatory.

Owner Class ID

ID of the holding class. The name or the ID of the class is mandatory.

Referenced Class Name

Name of the referenced class. The name or the ID of the referenced class is mandatory.

Referenced Class ID

ID of the referenced class. The name or the ID of the class is mandatory.

Multiplicity

Multiplicity of the part.

Generalization sheet

Generalization Name

Name of the part.

Super Class Name

Name of the general class. The name or the ID of the class is mandatory.

Super Class ID

ID of the general class. The name or the ID of the class is mandatory.

Sub Class Name

Name of the sub- class. The name or the ID of the class is mandatory.

Sub Class ID

ID of the sub- class. The name or the ID of the class is mandatory.

IMPORTING DATA ASSESSMENTS

HOPEX provides an Excel template to import data quality criteria values into your repository.

This Excel template is mapped to the data quality assessment template implemented in the direct assessment tool of **HOPEX Data Governance** and **HOPEX Information Architecture**.

It can be used for different types of objects that have direct evaluation, e.g. concepts, classes, tables, etc.

Import Example

Below is an example of importing the different quality values of a concept.

Values defined in the Excel file

	A	B	C	D	E	F	G	H	I	J
1	Hopex Identifier	Object Type	Object name	Completeness	Uniqueness	Timeliness	Validity	Accuracy	Consistency	Assessor Name
2		Concept	Artifact	Very Low	Low	Medium	Very High	High	Medium	NOURY Sam
3										
4										

Result in HOPEX Information Architecture

Artifact									
Assessment									
Direct Assessment									
+ Evaluate									
	Assessment Mode	Assessor	Evaluation Date	Completeness	Uniqueness	Timeliness	Validity	Accuracy	Consistency
	Artifact (Concept)	NOURY Samir	8/21/2020	Very Low	Low	Medium	Very High	High	Medium

Content of the Excel Template

Columns	Description
HOPEX Identifier	_IdAbs of an object assessed. When this property is specified, the Object Type and Name columns are not required.
Object Type	This property indicates the type of object assessed: "Concept", "Concept view", "Class", "Data view", "Table". This property must be defined when the HOPEX identifier is not specified.
Object Name	This property indicates the long name of the assessed object. It must be defined when the HOPEX identifier is not specified.
Completeness	Corresponds to the evaluation criteria "Completeness".
Uniqueness	Corresponds to the evaluation criteria "Unicity".
Timeliness	Corresponds to the evaluation criteria "Timeless".
Validity	Corresponds to the evaluation criteria "Validity".
Accuracy	Corresponds to the evaluation criteria "Accuracy".
Consistency	Corresponds to the evaluation criteria "Consistency".
Assessor Name	Assessor name. If no assessor is indicated, the name of the current user is taken by default in HOPEX.
Evaluation Date	Object assessment date. For the same object, measurements can be made at different times.

Downloading the Excel Template

The **Data Quality Excel Template** is available in the MEGA HOPEX Store. To download it:

1. Go to the HOPEX Store, at the following address:<https://community.mega.com/t5/HOPEX-Store/bd-p/hopex-store>.
2. Click **Data Quality Excel Template**.
3. Download the file.

Importing an Excel File of Data Assessments

To import data assessments from an Excel file:

1. Click the **Main Menu** then **Import > Excel (*.xls; *.xlsx)**.
2. To the right of the **Excel Import File** field, click **Browse**.
3. Select the file containing the assessments.

4. Click **Import**.

Appendix – Data Discovery Connectors

Contents

INTRODUCTION	17
1.1. Prerequisite for using HOPEX Data Discovery	17
1.2. Connection issues	18
2. ACCESS 	19
2.1. Access Version Support	19
2.2. Connection Options	19
2.2.1. Database	19
2.2.2. Logging	19
2.2.3. Schema	19
2.2.4. Miscellaneous	20
3. AMAZON DYNAMODB 	21
3.1. Amazon DynamoDB Version Support	21
3.2. Connection Options	21
3.2.1. AWS Authentication	21
3.2.2. SSO	22
3.2.3. SSL	22
3.2.4. Firewall	23
3.2.5. Proxy	23
3.2.6. Logging	24
3.2.7. Schema	24
3.2.8. Caching	25
3.2.9. Miscellaneous	25
4. AZURE ANALYSIS SERVICES 	29
4.1. Connection Options	29
4.1.1. Authentication	29
4.1.2. Azure Authentication	29
4.1.3. OAuth	30
4.1.4. SSL	31
4.1.5. Firewall	31
4.1.6. Proxy	32
4.1.7. Logging	32
4.1.8. Schema	33
4.1.9. Caching	34
4.1.10. Miscellaneous	34
5. AZURE DATA CATALOG 	37
5.1. Connection Options	37
5.1.1. Authentication	37
5.1.2. Azure Authentication	37
5.1.3. OAuth	37
5.1.4. SSL	38
5.1.5. Firewall	39
5.1.6. Proxy	39

5.1.7.	Logging.....	40
5.1.8.	Schema	40
5.1.9.	Caching	41
5.1.10.	Miscellaneous.....	41



6. AZURE SYNAPSE 43

6.1.	Connection Options.....	43
6.1.1.	Authentication	43
6.1.2.	Bulk	43
6.1.3.	Azure Authentication.....	44
6.1.4.	OAuth.....	44
6.1.5.	SSL.....	45
6.1.6.	Firewall	45
6.1.7.	Logging.....	45
6.1.8.	Schema	46
6.1.9.	Miscellaneous.....	46



7. AZURE TABLE 48

7.1.	Connection Options.....	48
7.1.1.	Authentication	48
7.1.2.	SSL.....	48
7.1.3.	Firewall	48
7.1.4.	Proxy	49
7.1.5.	Logging.....	50
7.1.6.	Schema	50
7.1.7.	Caching	51
7.1.8.	Miscellaneous.....	51



8. BIGQUERY 53

8.1.	Connection Options.....	53
8.1.1.	Authentication	53
8.1.2.	BigQuery	53
8.1.3.	Storage API	54
8.1.4.	Uploading	54
8.1.5.	OAuth.....	55
8.1.6.	JWT OAuth	55
8.1.7.	SSL.....	56
8.1.8.	Firewall	56
8.1.9.	Proxy	57
8.1.10.	Logging.....	57
8.1.11.	Schema	58
8.1.12.	Caching	58
8.1.13.	Miscellaneous.....	59



9. CASSANDRA 62

9.1.	Cassandra Version Support	62
9.2.	Connection Options.....	62
9.2.1.	Authentication	62

9.2.2.	Kerberos	63
9.2.3.	SSL.....	63
9.2.4.	SSH.....	64
9.2.5.	Firewall	65
9.2.6.	Logging.....	65
9.2.7.	Schema	66
9.2.8.	Caching	66
9.2.9.	Miscellaneous.....	67

10. CLOUDANT

10.1.	Connection Options.....	70
10.1.1.	Authentication.....	70
10.1.2.	OAuth.....	70
10.1.3.	SSL.....	71
10.1.4.	Firewall	71
10.1.5.	Proxy	71
10.1.6.	Logging.....	72
10.1.7.	Schema	72
10.1.8.	Caching	73
10.1.9.	Miscellaneous.....	74

11. COCKROACHDB

11.1.	CockroachDB Version Support	76
11.2.	Connection Options.....	76
11.2.1.	Authentication.....	76
11.2.2.	SSL.....	76
11.2.3.	Firewall	77
11.2.4.	Logging.....	77
11.2.5.	Schema	78
11.2.6.	Miscellaneous.....	78

12. CONFLUENCE

12.1.	Connection Options.....	80
12.1.1.	Authentication.....	80
12.1.2.	SSO.....	80
12.1.3.	OAuth.....	81
12.1.4.	SSL.....	82
12.1.5.	Firewall	82
12.1.6.	Proxy	82
12.1.7.	Logging.....	83
12.1.8.	Schema	84
12.1.9.	Caching	84
12.1.10.	Miscellaneous.....	85

13. COSMOS DB

13.1.	Connection Options.....	87
13.1.1.	Authentication.....	87
13.1.2.	Azure Authentication.....	87

13.1.3.	OAuth.....	88
13.1.4.	SSL.....	89
13.1.5.	Firewall	89
13.1.6.	Proxy	90
13.1.7.	Logging.....	90
13.1.8.	Schema	91
13.1.9.	Caching	91
13.1.10.	Miscellaneous.....	92

14. COUCHBASE 95

14.1.	Connection Options.....	95
14.1.1.	Authentication.....	95
14.1.2.	SSL.....	96
14.1.3.	Firewall	96
14.1.4.	Proxy	97
14.1.5.	Logging.....	97
14.1.6.	Schema	98
14.1.7.	Caching	99
14.1.8.	Miscellaneous.....	100

15. CSV 103

15.1.	Connection Options.....	103
15.1.1.	Authentication.....	103
15.1.2.	Connection.....	103
15.1.3.	AWS Authentication	104
15.1.4.	Azure Authentication.....	105
15.1.5.	SSO.....	105
15.1.6.	OAuth.....	105
15.1.7.	JWT OAuth	107
15.1.8.	Kerberos	107
15.1.9.	SSL.....	108
15.1.10.	SSH.....	109
15.1.11.	Firewall	109
15.1.12.	Proxy	110
15.1.13.	Logging.....	110
15.1.14.	Schema	111
15.1.15.	Caching	112
15.1.16.	Data Formatting.....	112
15.1.17.	Miscellaneous.....	114

16. DATABRICKS 117

16.1.	Connection Options.....	117
16.1.1.	Authentication.....	117
16.1.2.	AWS Authentication	117
16.1.3.	Azure Authentication.....	118
16.1.4.	AzureServicePrincipal Authentication	118
16.1.5.	SSL.....	119
16.1.6.	Firewall	119

16.1.7.	Proxy	120
16.1.8.	Logging.....	120
16.1.9.	Schema	121
16.1.10.	Caching	121
16.1.11.	Databricks	122
16.1.12.	Miscellaneous	123

17. ELASTICSEARCH 125

17.1.	Elasticsearch Version Support	125
17.2.	Connection Options	125
17.2.1.	Authentication	125
17.2.2.	Connection.....	125
17.2.3.	AWS Authentication	126
17.2.4.	Kerberos	126
17.2.5.	SSL.....	127
17.2.6.	Firewall	127
17.2.7.	Proxy	128
17.2.8.	Logging.....	128
17.2.9.	Schema	129
17.2.10.	Caching	129
17.2.11.	Miscellaneous	130

18. EXCEL 132

18.1.	Excel Version Support.....	132
18.2.	Connection Options	132
18.2.1.	Authentication	132
18.2.2.	Connection.....	132
18.2.3.	AWS Authentication	133
18.2.4.	Azure Authentication.....	134
18.2.5.	SSO.....	134
18.2.6.	Data	135
18.2.7.	OAuth.....	135
18.2.8.	JWT OAuth.....	137
18.2.9.	Kerberos	137
18.2.10.	SSL.....	138
18.2.11.	SSH.....	138
18.2.12.	Firewall	139
18.2.13.	Proxy	139
18.2.14.	Logging.....	140
18.2.15.	Schema	140
18.2.16.	Caching	141
18.2.17.	Miscellaneous	141

19. FINANCIALFORCE 144

19.1.	Connection Options	144
19.1.1.	Authentication	144
19.1.2.	Connection.....	144
19.1.3.	SSO.....	144
19.1.4.	BulkAPI.....	145

19.1.5.	OAuth.....	145
19.1.6.	SSL.....	146
19.1.7.	Firewall	147
19.1.8.	Proxy	147
19.1.9.	Logging.....	148
19.1.10.	Schema	148
19.1.11.	Caching	149
19.1.12.	Miscellaneous.....	149



20. EXCEL ONLINE 152

20.1.	Connection Options.....	152
20.1.1.	Authentication.....	152
20.1.2.	Connection.....	152
20.1.3.	Azure Authentication.....	153
20.1.4.	OAuth.....	153
20.1.5.	SSL.....	154
20.1.6.	Firewall	154
20.1.7.	Proxy	155
20.1.8.	Logging.....	155
20.1.9.	Schema	156
20.1.10.	Caching	156
20.1.11.	Miscellaneous.....	157



21. GOOGLE DATA CATALOG 160

21.1.	Connection Options.....	160
21.1.1.	Authentication.....	160
21.1.2.	OAuth.....	160
21.1.3.	JWT OAuth	161
21.1.4.	SSL.....	162
21.1.5.	Firewall	162
21.1.6.	Proxy	162
21.1.7.	Logging.....	163
21.1.8.	Schema	164
21.1.9.	Caching	164
21.1.10.	Miscellaneous.....	165



22. GOOGLE SPANNER 167

22.1.	Connection Options.....	167
22.1.1.	Authentication.....	167
22.1.2.	OAuth.....	167
22.1.3.	JWT OAuth	168
22.1.4.	SSL.....	169
22.1.5.	Firewall	169
22.1.6.	Proxy	169
22.1.7.	Logging.....	170
22.1.8.	Schema	171
22.1.9.	Caching	171
22.1.10.	Miscellaneous.....	172

23.	HBASE 	174
23.1.	Apache HBase Version Support	174
23.2.	Connection Options	174
23.2.1.	Authentication	174
23.2.2.	Kerberos	174
23.2.3.	SSL	175
23.2.4.	Firewall	175
23.2.5.	Proxy	176
23.2.6.	Logging	176
23.2.7.	Schema	177
23.2.8.	Caching	178
23.2.9.	Miscellaneous	178
24.	HIVE 	181
24.1.	Apache Hive Version Support	181
24.2.	Connection Options	181
24.2.1.	Authentication	181
24.2.2.	Kerberos	181
24.2.3.	SSL	182
24.2.4.	SSH	183
24.2.5.	Firewall	183
24.2.6.	Proxy	184
24.2.7.	Logging	185
24.2.8.	Schema	185
24.2.9.	Caching	186
24.2.10.	Miscellaneous	186
25.	IBM DB2 	189
25.1.	Connection Options	189
25.1.1.	Authentication	189
25.1.2.	SSL	189
25.1.3.	SSH	190
25.1.4.	Firewall	190
25.1.5.	Logging	191
25.1.6.	Schema	191
25.1.7.	Miscellaneous	192
26.	IBM DB2 FOR Z/OS	194
26.1.	Prerequisite	194
26.2.	Connection Options	195
27.	IBM INFORMIX 	198
27.1.	Informix Version Support	198
27.2.	Connection Options	198
27.2.1.	Authentication	198
27.2.2.	SSL	199
27.2.3.	SSH	199
27.2.4.	Firewall	200
27.2.5.	Logging	200

27.2.6.	Schema	201
27.2.7.	Miscellaneous	201

28. IMPALA

28.1.	Connection Options	203
28.1.1.	Authentication	203
28.1.2.	Kerberos	203
28.1.3.	SSL	204
28.1.4.	Firewall	205
28.1.5.	Proxy	205
28.1.6.	Logging	206
28.1.7.	Schema	206
28.1.8.	Caching	207
28.1.9.	Miscellaneous	207

29. JIRA

29.1.	Connection Options	210
29.1.1.	Authentication	210
29.1.2.	SSO	210
29.1.3.	OAuth	211
29.1.4.	SSL	212
29.1.5.	Firewall	212
29.1.6.	Proxy	213
29.1.7.	Logging	213
29.1.8.	Schema	214
29.1.9.	Caching	214
29.1.10.	Miscellaneous	215

30. JSON ^{JSON}

30.1.	Connection Options	218
30.1.1.	Authentication	218
30.1.2.	Connection	218
30.1.3.	AWS Authentication	219
30.1.4.	Azure Authentication	220
30.1.5.	SSO	220
30.1.6.	OAuth	221
30.1.7.	JWT OAuth	222
30.1.8.	Kerberos	223
30.1.9.	SSL	224
30.1.10.	SSH	224
30.1.11.	Firewall	225
30.1.12.	Proxy	225
30.1.13.	Logging	226
30.1.14.	Schema	226
30.1.15.	Caching	227
30.1.16.	Miscellaneous	228

31. KAFKA

31.1.	Connection Options	231

31.1.1.	Authentication	231
31.1.2.	Connection.....	231
31.1.3.	Kerberos	232
31.1.4.	SSL.....	232
31.1.5.	SchemaRegistry	233
31.1.6.	Firewall	234
31.1.7.	Proxy	234
31.1.8.	Logging.....	235
31.1.9.	Schema	236
31.1.10.	Caching	236
31.1.11.	Miscellaneous	237

32. MARIADB 240

32.1.	MariaDB Version Support.....	240
32.2.	Connection Options	240
32.2.1.	Authentication	240
32.2.2.	SSL.....	240
32.2.3.	SSH.....	241
32.2.4.	Firewall	242
32.2.5.	Logging.....	242
32.2.6.	Schema	243
32.2.7.	Miscellaneous	243

33. MONGODB 245

33.1.	MongoDB Version Support.....	245
33.2.	Connection Options	245
33.2.1.	Authentication	245
33.2.2.	Kerberos	246
33.2.3.	SSL.....	246
33.2.4.	SSH.....	247
33.2.5.	Firewall	247
33.2.6.	Logging.....	248
33.2.7.	Schema	248
33.2.8.	Caching	249
33.2.9.	Miscellaneous	249

34. MYSQL 252

34.1.	MySQL Version Support.....	252
34.2.	Connection Options	252
34.2.1.	Authentication	252
34.2.2.	Azure Authentication.....	253
34.2.3.	OAuth.....	253
34.2.4.	SSL.....	254
34.2.5.	SSH.....	254
34.2.6.	Firewall	255
34.2.7.	Proxy	255
34.2.8.	Logging.....	256
34.2.9.	Schema	256
34.2.10.	Miscellaneous	257

35.	SAS DATA SETS 	259
35.1.	Connection Options	259
35.1.1.	Authentication	259
35.1.2.	Connection.....	259
35.1.3.	AWS Authentication	260
35.1.4.	Azure Authentication.....	261
35.1.5.	SSO.....	261
35.1.6.	OAuth.....	262
35.1.7.	JWT OAuth.....	263
35.1.8.	Kerberos	263
35.1.9.	SSL.....	264
35.1.10.	SSH.....	264
35.1.11.	Firewall	265
35.1.12.	Proxy	265
35.1.13.	Logging.....	266
35.1.14.	Schema	266
35.1.15.	Caching	267
35.1.16.	Miscellaneous	267
36.	NEO4J 	270
36.1.	Neo4j Version Support	270
36.2.	Connection Options	270
36.2.1.	Authentication	270
36.2.2.	Connection.....	270
36.2.3.	SSL.....	270
36.2.4.	Firewall	271
36.2.5.	Proxy	271
36.2.6.	Logging.....	272
36.2.7.	Schema	272
36.2.8.	Caching	273
36.2.9.	Miscellaneous	273
37.	ORACLE 	275
37.1.	Prerequisite	275
37.2.	Oracle OCI Version Support.....	276
37.3.	Connection Options	276
37.3.1.	Authentication	276
37.3.2.	Logging.....	276
37.3.3.	Schema	277
37.3.4.	Miscellaneous	277
38.	PARQUET 	280
38.1.	Connection Options	280
38.1.1.	Authentication	280
38.1.2.	Connection.....	280
38.1.3.	AWS Authentication	281
38.1.4.	Azure Authentication.....	281
38.1.5.	SSO.....	282

38.1.6.	OAuth.....	282
38.1.7.	JWT OAuth	284
38.1.8.	SSL.....	284
38.1.9.	SSH.....	285
38.1.10.	Firewall	285
38.1.11.	Proxy	285
38.1.12.	Logging.....	286
38.1.13.	Schema	287
38.1.14.	Caching	287
38.1.15.	Miscellaneous	288

39. **POSTGRES**SQL 291

39.1.	PostgreSQL Version Support.....	291
39.2.	Connection Options	291
39.2.1.	Authentication	291
39.2.2.	AWS Authentication	292
39.2.3.	Azure Authentication.....	292
39.2.4.	OAuth.....	292
39.2.5.	Kerberos	293
39.2.6.	SSL.....	294
39.2.7.	SSH.....	294
39.2.8.	Firewall	295
39.2.9.	Logging.....	295
39.2.10.	Schema	296
39.2.11.	Miscellaneous	296

40. **REDIS** 298

40.1.	Redis Version Support	298
40.2.	Connection Options	298
40.2.1.	Authentication	298
40.2.2.	Connection.....	298
40.2.3.	SSL.....	299
40.2.4.	SSH.....	299
40.2.5.	Firewall	300
40.2.6.	Logging.....	300
40.2.7.	Schema	301
40.2.8.	Caching	301
40.2.9.	Miscellaneous	302

41. **REDSHIFT** 304

41.1.	Amazon Redshift Version Support.....	304
41.2.	Connection Options	304
41.2.1.	Authentication	304
41.2.2.	AWS Authentication	305
41.2.3.	SSO.....	305
41.2.4.	SSL.....	306
41.2.5.	SSH.....	306
41.2.6.	Firewall	307
41.2.7.	Proxy	307

41.2.8.	Logging.....	308
41.2.9.	Schema	308
41.2.10.	Miscellaneous	309

42. SALESFORCE 311

42.1.	Connection Options.....	311
42.1.1.	Authentication	311
42.1.2.	Connection.....	311
42.1.3.	SSO.....	312
42.1.4.	BulkAPI.....	312
42.1.5.	OAuth.....	313
42.1.6.	JWT OAuth	314
42.1.7.	SSL.....	314
42.1.8.	Firewall	314
42.1.9.	Proxy	315
42.1.10.	Logging.....	315
42.1.11.	Schema	316
42.1.12.	Caching	316
42.1.13.	Miscellaneous	317

43. SAP 320

43.1.	SAP ERP Version Support.....	320
43.2.	Connection Options	320
43.2.1.	Authentication	320
43.2.2.	Security	321
43.2.3.	SSL.....	321
43.2.4.	Firewall	321
43.2.5.	Proxy	322
43.2.6.	Logging.....	322
43.2.7.	Schema	323
43.2.8.	Caching	323
43.2.9.	Miscellaneous	324

44. SAP HANA 327

44.1.	Connection Options.....	327
44.1.1.	Authentication	327
44.1.2.	SSL.....	327
44.1.3.	SSH.....	328
44.1.4.	Firewall	329
44.1.5.	Logging.....	329
44.1.6.	Schema	330
44.1.7.	Miscellaneous	330

45. SHAREPOINT 332

45.1.	SharePoint Version Support	332
45.2.	Connection Options	332
45.2.1.	Authentication	332
45.2.2.	Azure Authentication.....	332

45.2.3.	SSO.....	333
45.2.4.	OAuth.....	333
45.2.5.	JWT OAuth.....	334
45.2.6.	Kerberos	334
45.2.7.	SSL.....	335
45.2.8.	Firewall	335
45.2.9.	Proxy.....	336
45.2.10.	Logging.....	336
45.2.11.	Schema	337
45.2.12.	Caching	337
45.2.13.	Miscellaneous.....	338

46. SNOWFLAKE 341

46.1.	Snowflake Version Support	341
46.2.	Connection Options.....	341
46.2.1.	Authentication.....	341
46.2.2.	Connection.....	342
46.2.3.	Azure Authentication.....	342
46.2.4.	SSO.....	342
46.2.5.	KeyPairAuth	343
46.2.6.	OAuth.....	343
46.2.7.	SSL.....	344
46.2.8.	Firewall	344
46.2.9.	Proxy	345
46.2.10.	Logging.....	345
46.2.11.	Schema	346
46.2.12.	Caching	346
46.2.13.	Miscellaneous.....	347

47. SPARK 350

47.1.	Spark SQL Version Support.....	350
47.2.	Connection Options.....	350
47.2.1.	Authentication.....	350
47.2.2.	Kerberos	351
47.2.3.	SSL.....	351
47.2.4.	Firewall	352
47.2.5.	Proxy	352
47.2.6.	Logging.....	353
47.2.7.	Schema	353
47.2.8.	Caching	354
47.2.9.	Miscellaneous.....	354

48. SQL SERVER 357

48.1.	SQL Server Version Support.....	357
48.2.	Connection Options.....	357
48.2.1.	Authentication	357
48.2.2.	Azure Authentication.....	358
48.2.3.	OAuth.....	358
48.2.4.	Kerberos	358

48.2.5.	SSL.....	359
48.2.6.	SSH.....	359
48.2.7.	Firewall	360
48.2.8.	Proxy	361
48.2.9.	Logging.....	361
48.2.10.	Schema	362
48.2.11.	Miscellaneous	362
48.3.	Description Extraction	364

49. SQL SERVER ANALYSIS SERVICES 365

49.1.	Connection Options	365
49.1.1.	Authentication	365
49.1.2.	Kerberos	365
49.1.3.	SSL.....	366
49.1.4.	Firewall	366
49.1.5.	Proxy	367
49.1.6.	Logging.....	367
49.1.7.	Schema	368
49.1.8.	Caching	368
49.1.9.	Miscellaneous	369



50. SYBASE 371

50.1.	Sybase Version Support	371
50.2.	Connection Options	371
50.2.1.	Authentication	371
50.2.2.	Kerberos	372
50.2.3.	SSL.....	372
50.2.4.	Firewall	373
50.2.5.	Logging.....	373
50.2.6.	Schema	374
50.2.7.	Miscellaneous	374



51. TERADATA 376

51.1.	Prerequisite	376
51.2.	Connection Options	377
51.2.1.	Authentication	377
51.2.2.	Connection.....	377
51.2.3.	Firewall	378
51.2.4.	Logging.....	378
51.2.5.	Schema	379
51.2.6.	Miscellaneous	379



52. XML 384

52.1.	Connection Options	384
52.1.1.	Authentication	384
52.1.2.	Connection.....	384
52.1.3.	AWS Authentication	385

52.1.4.	Azure Authentication.....	386
52.1.5.	SSO.....	386
52.1.6.	OAuth.....	387
52.1.7.	JWT OAuth	388
52.1.8.	Kerberos	389
52.1.9.	SSL.....	389
52.1.10.	SSH.....	390
52.1.11.	Firewall	390
52.1.12.	Proxy	391
52.1.13.	Logging.....	391
52.1.14.	Schema	392
52.1.15.	Caching	393
52.1.16.	Miscellaneous.....	393

INTRODUCTION

The HOPEX Data Discovery tool allows you to browse different data sources and define the metadata to import into your data catalogs.

Each data source has connection parameters. This document presents the connection string options of JDBC driver for each data source.

1.1. Prerequisite for using HOPEX Data Discovery

The HOPEX Data Discovery tool is available as a module, downloadable from the HOPEX HAS console.

For more details see the documentation [HOPEX Data Governance > Data Catalogs > Data Discovery](#).

In stand-alone mode, the workstation on which the module is installed must have .Net Core 6 and Microsoft Edge WebView2 installed.

After deploying the tool, you can connect to a data source: depending on the connector, see the list of supported versions and the description of the connection options.

After selecting the connector, to ensure the connection of HOPEX Data Discovery with the data source, follow these steps.

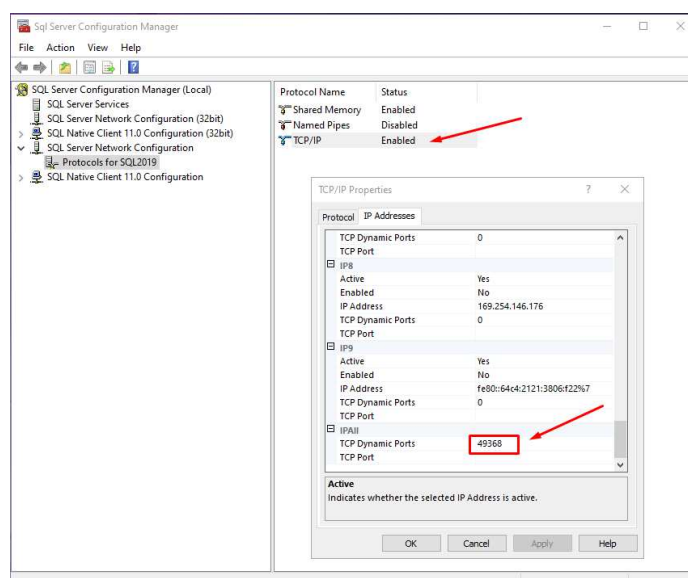
Connection to the data source

1. Make sure that the workstation where HOPEX Data Discovery is deployed is allowed to access the server where the data source is deployed (firewall, etc.).
2. Indicate the port/server/user names and the password.
3. Depending on the authentication system (for example Kerberos), check with the IT manager that the connection ticket is up to date.
4. For some connectors it is necessary to enter the name of the database you want to extract.

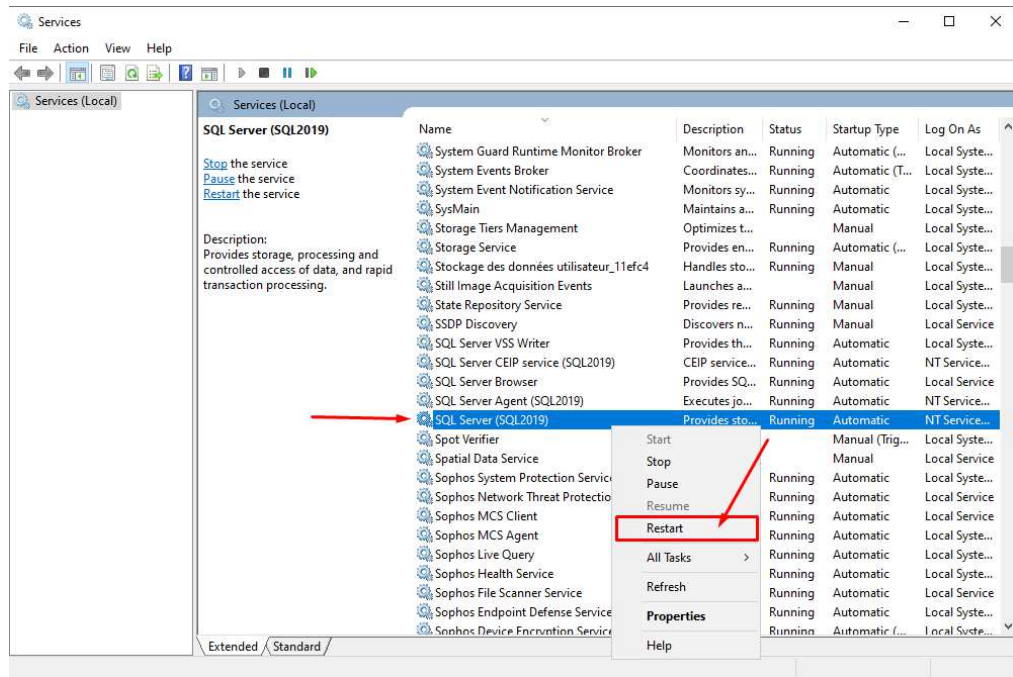
Network configuration

1. Check that the port used to connect to the data source is the correct one and that TCP/IP is enabled.

Example for SQL Server



2. After modifications restart the services.



1.2. Connection issues

If the connection is not established, provide MEGA Support with the Log file of connection errors.

To get the log file of connection errors:

1. Create a "DataDiscoveryLog.Txt" file and save it, in the "c:\temp" folder for example.
2. In the connection property "Logfile", specify the path to the file.
3. Add the property "Verbosity", with the value "5".

The syntax is as follows:

logfile="c:\tempDataDiscoveryLog.txt";verbosity="5".

Make sure there are no spaces in the path or in the file name.

In addition to the connection log file:

- Online mode: get all the logs from HOPEX.
- Stand-alone mode: get the log from HAS Light

2. ACCESS

2.1. Access Version Support

The driver may work with Access 1997 (.mdb, r/o), Access 2000 (.mdb), Access 2003 (.mdb), Access 2007 (.accdb), Access 2010 (.accdb), Access 2013 (.accdb) database files.

2.2. Connection Options

2.2.1. Database

Property	Description
DataSource	The full path and name of the MS Access database file.

2.2.2. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

2.2.3. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.

BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.

2.2.4. Miscellaneous

Property	Description
BatchSize	The maximum size of each batch operation to submit.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the Access when the connection is opened.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.
Readonly	You can use this property to enforce read-only access to Access from the provider.
RTK	The runtime key used for licensing.
SupportEnhancedSQL	This property enhances SQL functionality beyond what can be supported through the API directly, by enabling in-memory client-side processing.
UseConnectionPooling	This property enables connection pooling.

For more details, see <https://cdn.cdata.com/help/DCG/jdbc/>.

3. AMAZON DYNAMODB

3.1. Amazon DynamoDB Version Support

The driver uses the current version of the Amazon DynamoDB REST API, version 2012-08-10, to enable read/write access to DynamoDB instances.

3.2. Connection Options

3.2.1. AWS Authentication

Property	Description
AuthScheme	The scheme used for authentication. Accepted entries are: Auto, , AwsRootKeys , AwsIAMRoles , AwsEC2Roles , AwsMFA , ADFS, Okta, PingFederate , AwsCredentialsFile.
Domain	Your AWS domain name. You can optionally choose to associate your domain name with AWS.
AWSAccessKey	Your AWS account access key. This value is accessible from your AWS security credentials page.
AWSSecretKey	Your AWS account secret key. This value is accessible from your AWS security credentials page.
AWSRoleARN	The Amazon Resource Name of the role to use when authenticating.
AWSRegion	The hosting region for your Amazon Web Services.
AWSCredentialsFile	The path to the AWS Credentials File to be used for authentication.
AWSCredentialsFileProfile	The name of the profile to be used from the supplied AWSCredentialsFile.
AWSSessionToken	Your AWS session token.

AWSExternalId	A unique identifier that might be required when you assume a role in another account.
MFASerialNumber	The serial number of the MFA device if one is being used.
MFAToken	The temporary token available from your MFA device.
CredentialsLocation	The location of the settings file where MFA credentials are saved.
TemporaryTokenDuration	The amount of time (in seconds) a temporary token will last.

3.2.2. SSO

Property	Description
User	The IDP user used to authenticate the IDP via SSO.
Password	The password used to authenticate the IDP user via SSO.
SSOLoginURL	The identity provider's login URL.
SSOProperties	Additional properties required to connect to the identity provider in a semicolon-separated list.
SSOExchangeUrl	The url used for consuming the SAML response and exchanging it with Amazon DynamoDB specific credentials.

3.2.3. SSL

Property	Description
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

3.2.4. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

3.2.5. Proxy

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.
ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.
ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

3.2.6. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

3.2.7. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.

3.2.8. Caching

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .
CacheDriver	The database driver to be used to cache data.
CacheConnection	The connection string for the cache database. This property is always used in conjunction with CacheProvider . Setting both properties will override the value set for CacheLocation for caching data.
CacheLocation	Specifies the path to the cache when caching to a file.
CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache .
Offline	Use offline mode to get the data from the cache instead of the live source.
CacheMetadata	This property determines whether or not to cache the table metadata to a file store.

3.2.9. Miscellaneous

Property	Description
BatchSize	The maximum size of each batch operation to submit.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the Amazon DynamoDB when the connection is opened.

FlattenArrays	By default, nested arrays are returned as strings of JSON. The FlattenArrays property can be used to flatten the elements of nested arrays into columns of their own. Set FlattenArrays to the number of elements you want to return from nested arrays.
FlattenObjects	Set FlattenObjects to true to flatten object properties into columns of their own. Otherwise, objects nested in arrays are returned as strings of JSON.
FlexibleSchema	Set FlexibleSchema to true to scan for additional metadata on the query result set. Otherwise, the metadata will remain the same.
GenerateSchemaFiles	Indicates the user preference as to when schemas should be generated and saved.
IgnoreTypes	Removes support for the specified types. For example, Time. These types will then be reported as strings instead.
MaximumRequestRetries	The maximum number of times to retry a request.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.
Pagesize	The maximum number of results to return per page from Amazon DynamoDB.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.
PseudoColumns	This property indicates whether or not to include pseudo columns as columns to the table.

ReadOnly	You can use this property to enforce read-only access to Amazon DynamoDB from the provider.
RetryWaitTime	The minimum number of milliseconds the provider will wait to retry a request.
RowScanDepth	The maximum number of rows to scan to look for the columns available in a table.
RTK	The runtime key used for licensing.
SeparatorCharacter	The character or characters used to denote hierarchy.
SupportEnhancedSQL	This property enhances SQL functionality beyond what can be supported through the API directly, by enabling in-memory client-side processing.
ThreadCount	The number of threads to use when selecting data via a parallel scan. Setting ThreadCount to 1 will disable parallel scans.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
TypeDetectionScheme	Determines how to determine the data type of columns.
UseBatchWriteItemOperation	When enabled the provider will use BatchWriteItem operation for handling updates and inserts. By default, the provider uses ExecuteStatement/BatchExecuteStatement operation. You need to enable BatchWriteItem only when inserting/updating binary/binary-set data. ExecuteStatement/BatchExecuteStatement doesn't support manipulating binary fields.
UseConnectionPooling	This property enables connection pooling.
UseConsistentReads	Whether to always use Consistent Reads or not when querying DynamoDb.
UseSimpleNames	Boolean determining if simple names should be used for tables and columns.

For more details, see <https://cdn.cdata.com/help/DDG/jdbc/>.

4. AZURE ANALYSIS SERVICES

4.1. Connection Options

4.1.1. Authentication

Property	Description
AuthScheme	The type of authentication to use when connecting to Azure Analysis Services.
URL	The URL used to connect to the Azure Analysis Services.
User	The Azure Analysis Services user account used to authenticate.
Password	The password used to authenticate the user.

4.1.2. Azure Authentication

Property	Description
AzureTenant	The Microsoft Online tenant being used to access data. If not specified, your default tenant will be used.
AzureEnvironment	The Azure Environment to use when establishing a connection.

4.1.3. OAuth

Property	Description
InitiateOAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.
OAuthClientId	The client ID assigned when you register your application with an OAuth authorization server.
OAuthClientSecret	The client secret assigned when you register your application with an OAuth authorization server.
OAuthAccessToken	The access token for connecting using OAuth.
OAuthSettingsLocation	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH. Alternatively, this can be held in memory by specifying a value starting with memory://.
CallbackURL	The OAuth callback URL to return to when authenticating. This value must match the callback URL you specify in your app settings.
OAuthVerifier	The verifier code returned from the OAuth authorization URL.
OAuthRefreshToken	The OAuth refresh token for the corresponding OAuth access token.
OAuthExpiresIn	The lifetime in seconds of the OAuth AccessToken.
OAuthTokenTimestamp	The Unix epoch timestamp in milliseconds when the current Access Token was created.

4.1.4. SSL

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.
SSLClientCertSubject	The subject of the TLS/SSL client certificate.
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

4.1.5. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

4.1.6. Proxy

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.
ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.
ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

4.1.7. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.

LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

4.1.8. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.
Catalog	The Analysis Services catalog to use. This may also be known as a Database from within Analysis Services.
IncludeJoinColumns	Set this to true to include extra join columns on each table.

4.1.9. Caching

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .
CacheDriver	The database driver to be used to cache data.
CacheConnection	The connection string for the cache database. This property is always used in conjunction with CacheProvider . Setting both properties will override the value set for CacheLocation for caching data.
CacheLocation	Specifies the path to the cache when caching to a file.
CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache .
Offline	Use offline mode to get the data from the cache instead of the live source.
CacheMetadata	This property determines whether or not to cache the table metadata to a file store.

4.1.10. Miscellaneous

Property	Description
BatchSize	The maximum size of each batch operation to submit.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.

ConnectOnOpen	This property species whether to connect to the Azure Analysis Services when the connection is opened.
CustomHeaders	Other headers as determined by the user (optional).
ExtraProperties	Additional properties to submit on each MDX request to Azure Analysis Services.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.
ResponseRowLimit	The number of response rows to allow before erroring. Set to 0 for now limit.
RTK	The runtime key used for licensing.
SplitMeasures	Set this to true to split Measures table into individual tables.
SupportEnhancedSQL	This property enhances SQL functionality beyond what can be supported through the API directly, by enabling in-memory client-side processing.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
UseConnectionPooling	This property enables connection pooling.

UseMDX	Set this to true to pass MDX queries to Azure Analysis Services as-is.

For more details, see <https://cdn.cdata.com/help/OAG/jdbc/>.

5. AZURE DATA CATALOG

5.1. Connection Options

5.1.1. Authentication

Property	Description
AuthScheme	The type of authentication to use when connecting to Azure Data Catalog.
CatalogName	The name of the catalog to connect to.

5.1.2. Azure Authentication

Property	Description
AzureTenant	The Microsoft Online tenant being used to access data. If not specified, your default tenant will be used.
AzureEnvironment	The Azure Environment to use when establishing a connection.

5.1.3. OAuth

Property	Description
InitiateOAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.
OAuthClientId	The client ID assigned when you register your application with an OAuth authorization server.

OAuthClientSecret	The client secret assigned when you register your application with an OAuth authorization server.
OAuthAccessToken	The access token for connecting using OAuth.
OAuthSettingsLocation	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH. Alternatively, this can be held in memory by specifying a value starting with memory://.
CallbackURL	The OAuth callback URL to return to when authenticating. This value must match the callback URL you specify in your app settings.
OAuthVerifier	The verifier code returned from the OAuth authorization URL.
OAuthRefreshToken	The OAuth refresh token for the corresponding OAuth access token.
OAuthExpiresIn	The lifetime in seconds of the OAuth AccessToken.
OAuthTokenTimestamp	The Unix epoch timestamp in milliseconds when the current Access Token was created.

5.1.4. SSL

Property	Description
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

5.1.5. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

5.1.6. Proxy

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.
ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.

ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

5.1.7. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

5.1.8. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.

Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.

5.1.9. Caching

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .
CacheDriver	The database driver to be used to cache data.
CacheConnection	The connection string for the cache database. This property is always used in conjunction with CacheProvider . Setting both properties will override the value set for CacheLocation for caching data.
CacheLocation	Specifies the path to the cache when caching to a file.
CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache .
Offline	Use offline mode to get the data from the cache instead of the live source.
CacheMetadata	This property determines whether or not to cache the table metadata to a file store.

5.1.10. Miscellaneous

Property	Description
BatchSize	The maximum size of each batch operation to submit.

ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the Azure Data Catalog when the connection is opened.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.
Pagesize	The maximum number of results to return per page from Azure Data Catalog.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.
PseudoColumns	This property indicates whether or not to include pseudo columns as columns to the table.
RTK	The runtime key used for licensing.
SupportEnhancedSQL	This property enhances SQL functionality beyond what can be supported through the API directly, by enabling in-memory client-side processing.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
UseConnectionPooling	This property enables connection pooling.

For more details, see <https://cdn.cdata.com/help/GNG/jdbc/>.

6. AZURE SYNAPSE

6.1. Connection Options

6.1.1. Authentication

Property	Description
AuthScheme	The scheme used for authentication. Accepted entries are Password, AzureAD, AzureMSI.
Server	The name of the server running Synapse.
Port	The port of the Synapse.
User	The Azure Synapse user account used to authenticate.
Database	The name of the Synapse database.
Password	The password used to authenticate the user.
Encrypt	This field sets whether SSL is enabled.

6.1.2. Bulk

Property	Description
BatchMode	The Batch Mode of Azure Synapse bulkInsert.
StorageAccountLocation	The Storage Account Location for staging your data.
AzureSASToken	The string value of the Azure Blob shared access signature.

6.1.3. Azure Authentication

Property	Description
AzureTenant	The Microsoft Online tenant being used to access data. If not specified, your default tenant will be used.

6.1.4. OAuth

Property	Description
InitiateOAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.
OAuthClientId	The client ID assigned when you register your application with an OAuth authorization server.
OAuthClientSecret	The client secret assigned when you register your application with an OAuth authorization server.
OAuthAccessToken	The access token for connecting using OAuth.
OAuthSettingsLocation	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH. Alternatively, this can be held in memory by specifying a value starting with memory://.
CallbackURL	The OAuth callback URL to return to when authenticating. This value must match the callback URL you specify in your app settings.
OAuthGrantType	The grant type for the OAuth flow.
OAuthVerifier	The verifier code returned from the OAuth authorization URL.
OAuthExpiresIn	The lifetime in seconds of the OAuth AccessToken.
OAuthTokenTimestamp	The Unix epoch timestamp in milliseconds when the current Access Token was created.

6.1.5. SSL

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.
SSLClientCertSubject	The subject of the TLS/SSL client certificate.
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

6.1.6. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

6.1.7. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.

Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

6.1.8. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.

6.1.9. Miscellaneous

Property	Description
CustomizeDateFormat	Configure the Date format.
ApplicationIntent	Expresses the client application's request to be directed either to a read-write or read-only version of an availability group database.
BatchSize	The maximum size of each batch operation to submit.

ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the Azure Synapse when the connection is opened.
EnableTransaction	Determines whether transactions are enabled.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.
QueryPassthrough	This option passes the query to the Azure Synapse server as is.
Readonly	You can use this property to enforce read-only access to Azure Synapse from the provider.
RTK	The runtime key used for licensing.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
UseConnectionPooling	This property enables connection pooling.

For more details, see <https://cdn.cdata.com/help/HEG/jdbc/>.

7. AZURE TABLE

7.1. Connection Options

7.1.1. Authentication

Property	Description
Account	The Windows Azure Storage account name.
UseSSL	This field sets whether SSL is enabled. The default is true.
AccessKey	The key for the storage account.
AuthScheme	The scheme used for authentication. Accepted entries are AccessToken and SharedAccessSignature.
Backend	The backend where data is stored.
SharedAccessSignature	A shared access key signature that may be used for authentication.

7.1.2. SSL

Property	Description
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

7.1.3. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.

FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

7.1.4. Proxy

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.
ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.
ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

7.1.5. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

7.1.6. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.

7.1.7. Caching

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .
CacheDriver	The database driver to be used to cache data.
CacheConnection	The connection string for the cache database. This property is always used in conjunction with CacheProvider . Setting both properties will override the value set for CacheLocation for caching data.
CacheLocation	Specifies the path to the cache when caching to a file.
CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache .
Offline	Use offline mode to get the data from the cache instead of the live source.
CacheMetadata	This property determines whether or not to cache the table metadata to a file store.

7.1.8. Miscellaneous

Property	Description
BatchSize	The maximum size of each batch operation to submit.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the Azure Table Storage when the connection is opened.

GenerateSchemaFiles	Indicates the user preference as to when schemas should be generated and saved.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.
PseudoColumns	This property indicates whether or not to include pseudo columns as columns to the table.
Readonly	You can use this property to enforce read-only access to Azure Table Storage from the provider.
RowScanDepth	The maximum number of rows to scan to look for the columns available in a table.
RTK	The runtime key used for licensing.
SupportEnhancedSQL	This property enhances SQL functionality beyond what can be supported through the API directly, by enabling in-memory client-side processing.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
TypeDetectionScheme	Determines how to determine the data type of columns.
UseConnectionPooling	This property enables connection pooling.

For more details, see <https://cdn.cdata.com/help/CAG/jdbc/>.

8. BIGQUERY

8.1. Connection Options

8.1.1. Authentication

Property	Description
AuthScheme	The type of authentication to use when connecting to Google BigQuery.
ProjectId	The ProjectId used to resolve unqualified tables.
DatasetId	The DatasetId of the dataset you wish to connect to and view tables of.
BillingProjectId	The ProjectId of the billing project for executing jobs.

8.1.2. BigQuery

Property	Description
AllowLargeResultSets	Whether or not to allow large datasets to be stored in temporary tables for large datasets.
DestinationTable	This property determines where query results are stored in Google BigQuery.
UseQueryCache	Specifies whether to use Google BigQuery's built-in query cache.
PageSize	The number of results to return per page from Google BigQuery.
PollingInterval	This determines how long to wait in seconds, between checks to see if a job has completed.
AllowUpdatesWithoutKey	Whether or not to allow update without primary keys.

FilterColumns	Please set `AllowUpdatesWithoutKey` to true before you could use this property.
UseLegacySQL	Specifies whether to use BigQuery's legacy SQL dialect for this query. By default, Standard SQL will be used.

8.1.3. Storage API

Property	Description
UseStorageAPI	Specifies whether to use BigQuery's Storage API for bulk data reads.
UseArrowFormat	Specifies whether to use the Arrow format with BigQuery's Storage API.
StorageThreshold	The minimum number of rows a query must return to invoke the Storage API.
StoragePageSize	Specifies the page size to use for Storage API queries.

8.1.4. Uploading

Property	Description
InsertMode	Specifies what kind of method to use when inserting data. By default streaming inserts are used.
WaitForBatchResults	Whether to wait for the job to complete when using the bulk upload API. Only active when InsertMode is set to Upload.
GCSBucket	Specifies the name of a GCS bucket to upload bulk data for staging.
GCSBucketFolder	Specifies the name of the folder in GCSBucket to upload bulk data for staging.
TempTableDataset	The prefix of the dataset that will contain temporary tables when performing bulk UPDATE or DELETE operations.

8.1.5. OAuth

Property	Description
InitiateOAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.
OAuthClientId	The client ID assigned when you register your application with an OAuth authorization server.
OAuthClientSecret	The client secret assigned when you register your application with an OAuth authorization server.
OAuthAccessToken	The access token for connecting using OAuth.
OAuthSettingsLocation	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH. Alternatively, this can be held in memory by specifying a value starting with memory://.
Scope	Identifies the Google API access that your application is requesting. Scopes are space-delimited.
OAuthVerifier	The verifier code returned from the OAuth authorization URL.
OAuthRefreshToken	The OAuth refresh token for the corresponding OAuth access token.
OAuthExpiresIn	The lifetime in seconds of the OAuth AccessToken.
OAuthTokenTimestamp	The Unix epoch timestamp in milliseconds when the current Access Token was created.

8.1.6. JWT OAuth

Property	Description
OAuthJWTCert	The JWT Certificate store.

OAuthJWTCertType	The type of key store containing the JWT Certificate.
OAuthJWTCertPassword	The password for the OAuth JWT certificate.
OAuthJWTCertSubject	The subject of the OAuth JWT certificate.
OAuthJWTIssuer	The issuer of the Java Web Token.
OAuthJWTSubject	The user subject for which the application is requesting delegated access.

8.1.7. SSL

Property	Description
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

8.1.8. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

8.1.9. Proxy

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.
ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.
ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

8.1.10. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).

MaxLogFileCount	A string specifying the maximum file count of log files.

8.1.11. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.
RefreshViewSchemas	Allows the provider to determine up-to-date view schemas automatically.
ShowTableDescriptions	Controls whether table descriptions are returned via the platform metadata APIs and sys_tables / sys_views.
PrimaryKeyIdentifiers	Set this property to define primary keys.
AllowedTableTypes	Specifies what kinds of tables will be visible.

8.1.12. Caching

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .
CacheDriver	The database driver to be used to cache data.

CacheConnection	The connection string for the cache database. This property is always used in conjunction with CacheProvider . Setting both properties will override the value set for CacheLocation for caching data.
CacheLocation	Specifies the path to the cache when caching to a file.
CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache .
Offline	Use offline mode to get the data from the cache instead of the live source.
CacheMetadata	This property determines whether or not to cache the table metadata to a file store.

8.1.13. Miscellaneous

Property	Description
StorageTimeout	How long a Storage API connection must remain idle before the provider reconnects.
AllowAggregateParameters	Allows raw aggregates to be used in parameters when QueryPassthrough is enabled.
AuditLimit	The maximum number of rows which will be stored within an audit table.
AuditMode	What provider actions should be recorded to audit tables.
BatchSize	The maximum size of each batch operation to submit.
BigQueryOptions	A comma separated list of Google BigQuery options.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the Google BigQuery when the connection is opened.

GenerateSchemaFiles	Indicates the user preference as to when schemas should be generated and saved.
MaximumBillingTier	The MaximumBillingTier is a positive integer that serves as a multiplier of the basic price per TB. For example, if you set MaximumBillingTier to 2, the maximum cost for that query will be 2x basic price per TB.
MaximumBytesBilled	Limits how many bytes BigQuery will allow a job to consume before it is cancelled.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.
PseudoColumns	This property indicates whether or not to include pseudo columns as columns to the table.
QueryPassthrough	This option passes the query to the Google BigQuery server as is.
Readonly	You can use this property to enforce read-only access to Google BigQuery from the provider.
RTK	The runtime key used for licensing.
TableSamplePercent	This determines what percent of a table is sampled with the TABLESAMPLE operator.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
UseConnectionPooling	This property enables connection pooling.

For more details, see <https://cdn.cdata.com/help/DBG/jdbc/>.

9. CASSANDRA

9.1. Cassandra Version Support

The driver supports CQL versions 2.0, 3.0 and 4.0.

9.2. Connection Options

9.2.1. Authentication

Property	Description
AuthScheme	The scheme used for authentication. Accepted entries are Basic, DSE, Kerberos, and LDAP.
Server	The host name or IP address of the server hosting the Cassandra database.
Port	The port for the Cassandra database.
LDAPServer	The host name or IP address of the LDAP server.
User	The Cassandra user account used to authenticate.
Password	The password used to authenticate the user.
LDAPPort	The port for the LDAP server.
Database	The name of the Cassandra keyspace.
DefaultLDAPUser	The default LDAP user used to connect to and communicate with the server, it must be set if the LDAP server do not allow anonymous bind.
LDAPPassword	The password of the default LDAP user. It must be set if the LDAP server do not allow anonymous bind.
SearchBase	The search base for your LDAPServer, used to look up users.

SearchFilter	The search filter for looking up usernames in LDAP. The default setting is (uid=), When using Active Directory set the filter to (sAMAccountName=).
UseSSL	This field sets whether SSL is enabled.

9.2.2. Kerberos

Property	Description
KerberosKDC	The Kerberos Key Distribution Center (KDC) service used to authenticate the user.
KerberosRealm	The Kerberos Realm used to authenticate the user with.
KerberosSPN	The service principal name (SPN) for the Kerberos Domain Controller.
KerberosKeytabFile	The Keytab file containing your pairs of Kerberos principals and encrypted keys.
KerberosServiceRealm	The Kerberos realm of the service.
KerberosServiceKDC	The Kerberos KDC of the service.
KerberosTicketCache	The full file path to an MIT Kerberos credential cache file.

9.2.3. SSL

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.

SSLClientCertSubject	The subject of the TLS/SSL client certificate.
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

9.2.4. SSH

Property	Description
SSHAuthMode	The authentication method to be used to log on to an SFTP server.
SSHClientCert	A certificate to be used for authenticating the user.
SSHClientCertPassword	The password of the SSHClientCert certificate if it has one.
SSHClientCertSubject	The subject of the SSH client certificate.
SSHClientCertType	The type of SSHClientCert certificate.
SSHServer	The SSH server.
SSHPort	The SSH port.
SSHUser	The SSH user.
SSHPassword	The SSH password.
SSHServerFingerprint	The SSH server fingerprint.
UseSSH	Whether to tunnel the Cassandra connection over SSH. Use SSH.

9.2.5. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

9.2.6. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

9.2.7. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.

9.2.8. Caching

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .
CacheDriver	The database driver to be used to cache data.
CacheConnection	The connection string for the cache database. This property is always used in conjunction with CacheProvider . Setting both properties will override the value set for CacheLocation for caching data.
CacheLocation	Specifies the path to the cache when caching to a file.
CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache .
Offline	Use offline mode to get the data from the cache instead of the live source.

CacheMetadata

This property determines whether or not to cache the table metadata to a file store.

9.2.9. Miscellaneous

Property	Description
AggregationsSupported	Whether or not to support aggregations in the Cassandra server. Note that in queries to the provider, you must use single quotes to define strings.
AllowFiltering	When true, slow-performing queries are processed on the server.
BatchSize	The maximum size of each batch operation to submit.
CaseSensitivity	Enable case sensitivity to the CQL sending to the server, if set to True, the identifiers in the CQL will be enclosed in double quotation marks.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the Cassandra when the connection is opened.
ConsistencyLevel	The consistency level determines how many of the replicas of the data you are interacting with need to respond for the query to be considered a success.
FlattenArrays	By default, nested arrays are returned as strings of JSON. The FlattenArrays property can be used to flatten the elements of nested arrays into columns of their own. Set FlattenArrays to the number of elements you want to return from nested arrays.
FlattenObjects	Set FlattenObjects to true to flatten object properties into columns of their own. Otherwise, objects nested in arrays are returned as strings of JSON.

MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
NullToUnset	Use unset instead of NULL in CQL query when performing INSERT operations.
Other	These hidden properties are used only in specific use cases.
Pagesize	The maximum number of results to return per page from Cassandra.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.
PseudoColumns	This property indicates whether or not to include pseudo columns as columns to the table.
QueryPassthrough	This option passes the query to the Cassandra server as is.
Readonly	You can use this property to enforce read-only access to Cassandra from the provider.
RowScanDepth	The maximum number of rows to scan to look for the columns available in a table.
RTK	The runtime key used for licensing.
SupportEnhancedSQL	This property enhances SQL functionality beyond what can be supported through the API directly, by enabling in-memory client-side processing.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
UseConnectionPooling	This property enables connection pooling.

UseJsonFormat	Whether to submit and return the JSON encoding for CQL data types.
VarintToString	Map Cassandra VARINT to String value.

For more details, see <https://cdn.cdata.com/help/RCG/jdbc/>.

10. CLOUDANT

10.1. Connection Options

10.1.1. Authentication

Property	Description
AuthScheme	The type of authentication to use when connecting to Cloudant.
URL	The URL used to connect to the Cloudant.
User	The Cloudant user account used to authenticate.
Password	The password used to authenticate the user.
ApiKey	The API Key used to identify the user to IBM Cloud.

10.1.2. OAuth

Property	Description
InitiateOAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.
OAuthAccessToken	The access token for connecting using OAuth.
OAuthSettingsLocation	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH. Alternatively, this can be held in memory by specifying a value starting with memory://.
OAuthRefreshToken	The OAuth refresh token for the corresponding OAuth access token.
OAuthExpiresIn	The lifetime in seconds of the OAuth AccessToken.

OAuthTokenTimestamp	The Unix epoch timestamp in milliseconds when the current Access Token was created.

10.1.3. SSL

Property	Description
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

10.1.4. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

10.1.5. Proxy

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.

ProxyPort	The TCP port the ProxyServer proxy is running on.
ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.
ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

10.1.6. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

10.1.7. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.

BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.
ListViews	Whether to list views from Cloudant or not.

10.1.8. Caching

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .
CacheDriver	The database driver to be used to cache data.
CacheConnection	The connection string for the cache database. This property is always used in conjunction with CacheProvider . Setting both properties will override the value set for CacheLocation for caching data.
CacheLocation	Specifies the path to the cache when caching to a file.
CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache .
Offline	Use offline mode to get the data from the cache instead of the live source.
CacheMetadata	This property determines whether or not to cache the table metadata to a file store.

10.1.9. Miscellaneous

Property	Description
BatchSize	The maximum size of each batch operation to submit.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the Cloudant when the connection is opened.
FlattenArrays	Set FlattenArrays to the number of array elements to flatten into columns. Otherwise, arrays are returned as JSON strings.
FlattenObjects	Set FlattenObjects to true to flatten object properties into columns of their own. Otherwise, embedded objects as raw JSON strings.
FlexibleSchema	Set FlexibleSchema to true to scan for additional metadata on the query result set. Otherwise, the metadata will remain the same.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.
Pagesize	The maximum number of results to return per page from Cloudant.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.

PseudoColumns	This property indicates whether or not to include pseudo columns as columns to the table.
Readonly	You can use this property to enforce read-only access to Cloudant from the provider.
RowScanDepth	The maximum number of rows to scan to look for the columns available in a table.
RTK	The runtime key used for licensing.
SeparatorCharacter	The character or characters used to denote hierarchy.
SupportEnhancedSQL	This property enhances SQL functionality beyond what can be supported through the API directly, by enabling in-memory client-side processing.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
TypeDetectionScheme	Option for how the provider will scan the data to determine the fields and data types in each document collection.
UseConnectionPooling	This property enables connection pooling.

For more details, see <https://cdn.cdata.com/help/EWG/jdbc/>.

11. COCKROACHDB

11.1. CockroachDB Version Support

The driver enables standards-based access to Cockroach DB version 2.0 and later.

11.2. Connection Options

11.2.1. Authentication

Property	Description
Server	The host name or IP address of the server.
Port	The port number of the CockroachDB server.
Database	The name of the Cockroach database.
User	The CockroachDB user account used to authenticate.
Password	The password used to authenticate the user.
UseSSL	Whether SSL is enabled.

11.2.2. SSL

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.
SSLClientCertSubject	The subject of the TLS/SSL client certificate.

SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

11.2.3. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

11.2.4. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

11.2.5. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.

11.2.6. Miscellaneous

Property	Description
AllowPreparedStatement	Allow the preparation of a query statement before its execution.
BatchSize	The maximum size of each batch operation to submit.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the CockroachDB when the connection is opened.
FetchResultSetMetadata	This field sets whether the provider is getting detailed information about resultset columns from the server.

MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.
QueryPassthrough	This option passes the query to the CockroachDB server as is.
Readonly	You can use this property to enforce read-only access to CockroachDB from the provider.
RTK	The runtime key used for licensing.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
TimeZone	The time zone for the current session.
UseConnectionPooling	This property enables connection pooling.

For more details, see <https://cdn.cdata.com/help/HJG/jdbc/>.

12. CONFLUENCE

12.1. Connection Options

12.1.1. Authentication

Property	Description
AuthScheme	The type of authentication to use when connecting to Confluence.
URL	The URL to your JIRA endpoint.
CloudId	The Cloud Id for the Atlassian site that was authorized.
User	The Confluence user account used to authenticate.
Password	The password used to authenticate the user.
APIToken	APIToken of the currently authenticated user.
Timezone	Specify the timezone of the Confluence instance in order to use the datetime filters accordingly and retrieve the results according to your timezone. An example of a timezone would be America/New_York.

12.1.2. SSO

Property	Description
SSOLoginURL	The identity provider's login URL.
SSOProperties	Additional properties required to connect to the identity provider in a semicolon-separated list.
SSOExchangeUrl	The url used for consuming the SAML response and exchanging it with Confluence specific credentials.
SSOAppName	App Name used with SSO for IdPs that require it.

SSOAppPassword

App Password used with SSO for IdPs that require it.

12.1.3. OAuth

Property	Description
InitiateOAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.
OAuthVersion	The version of OAuth being used.
OAuthClientId	The client ID assigned when you register your application with an OAuth authorization server.
OAuthClientSecret	The client secret assigned when you register your application with an OAuth authorization server.
OAuthAccessToken	The access token for connecting using OAuth.
OAuthSettingsLocation	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH. Alternatively, this can be held in memory by specifying a value starting with memory://.
CertificateStore	The certificate store used for Confluence authentication.
CertificateStorePassword	The password of the certificate store used with Confluence authentication.
CertificateSubject	The subject of the certificate used with Confluence Private Application authentication.
CertificateStoreType	The type of certificate store used with Confluence Private Application authentication.
OAuthVerifier	The verifier code returned from the OAuth authorization URL.
OAuthExpiresIn	The lifetime in seconds of the OAuth AccessToken.

OAuthTokenTimestamp	The Unix epoch timestamp in milliseconds when the current Access Token was created.

12.1.4. SSL

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.
SSLClientCertSubject	The subject of the TLS/SSL client certificate.
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

12.1.5. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

12.1.6. Proxy

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.
ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.
ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

12.1.7. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).

MaxLogFileCount	A string specifying the maximum file count of log files.

12.1.8. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.

12.1.9. Caching

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .
CacheDriver	The database driver to be used to cache data.
CacheConnection	The connection string for the cache database. This property is always used in conjunction with CacheProvider . Setting both properties will override the value set for CacheLocation for caching data.
CacheLocation	Specifies the path to the cache when caching to a file.

CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache .
Offline	Use offline mode to get the data from the cache instead of the live source.
CacheMetadata	This property determines whether or not to cache the table metadata to a file store.

12.1.10. Miscellaneous

Property	Description
BatchSize	The maximum size of each batch operation to submit.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the Confluence when the connection is opened.
IncludeArchivedSpaces	Whether to include content from archived spaces in the result. This defaults to false.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.
Pagesize	The maximum number of results to return per page from Confluence.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.

PoolWaitTime	The max seconds to wait for an available connection.
PseudoColumns	This property indicates whether or not to include pseudo columns as columns to the table.
RTK	The runtime key used for licensing.
SpaceKey	By specifying the SpaceKey, the search results will only display contents from this specific space.
SupportEnhancedSQL	This property enhances SQL functionality beyond what can be supported through the API directly, by enabling in-memory client-side processing.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
UseConnectionPooling	This property enables connection pooling.

For more details, see <https://cdn.cdata.com/help/UGG/jdbc/>.

13. COSMOS DB

13.1. Connection Options

13.1.1. Authentication

Property	Description
AuthScheme	The type of authentication to use when connecting to Cosmos DB.
AccountEndpoint	The value should be the Cosmos DB account URL from the Keys blade of the Cosmos DB account.
AccountKey	A master key token or a resource token for connecting to the Cosmos DB REST API.
TokenType	Denotes the type of token: master or resource.

13.1.2. Azure Authentication

Property	Description
AzureTenant	The Microsoft Online tenant being used to access data. If not specified, your default tenant will be used.
AzureEnvironment	The Azure Environment to use when establishing a connection.

13.1.3. OAuth

Property	Description
InitiateOAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.
OAuthClientId	The client ID assigned when you register your application with an OAuth authorization server.
OAuthClientSecret	The client secret assigned when you register your application with an OAuth authorization server.
OAuthAccessToken	The access token for connecting using OAuth.
OAuthSettingsLocation	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH. Alternatively, this can be held in memory by specifying a value starting with memory://.
CallbackURL	The OAuth callback URL to return to when authenticating. This value must match the callback URL you specify in your app settings.
OAuthGrantType	The grant type for the OAuth flow.
OAuthVerifier	The verifier code returned from the OAuth authorization URL.
OAuthRefreshToken	The OAuth refresh token for the corresponding OAuth access token.
OAuthExpiresIn	The lifetime in seconds of the OAuth AccessToken.
OAuthTokenTimestamp	The Unix epoch timestamp in milliseconds when the current Access Token was created.

13.1.4. SSL

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.
SSLClientCertSubject	The subject of the TLS/SSL client certificate.
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

13.1.5. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

13.1.6. Proxy

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.
ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.
ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

13.1.7. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.

MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

13.1.8. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.
Schema	Specify the Cosmos DB database you want to work with.

13.1.9. Caching

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .
CacheDriver	The database driver to be used to cache data.
CacheConnection	The connection string for the cache database. This property is always used in conjunction with CacheProvider . Setting both properties will override the value set for CacheLocation for caching data.

CacheLocation	Specifies the path to the cache when caching to a file.
CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache .
Offline	Use offline mode to get the data from the cache instead of the live source.
CacheMetadata	This property determines whether or not to cache the table metadata to a file store.

13.1.10. Miscellaneous

Property	Description
BatchSize	The maximum size of each batch operation to submit.
CalculateAggregates	Specifies whether will return the calculated value of the aggregates or grouped by partiton range.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the Cosmos DB when the connection is opened.
ConsistencyLevel	Denotes the type of token: master or resource.
FlattenArrays	By default, nested arrays are returned as strings of JSON. The FlattenArrays property can be used to flatten the elements of nested arrays into columns of their own. Set FlattenArrays to the number of elements you want to return from nested arrays.
FlattenObjects	Set FlattenObjects to true to flatten object properties into columns of their own. Otherwise, objects nested in arrays are returned as strings of JSON.
GenerateSchemaFiles	Indicates the user preference as to when schemas should be generated and saved.

MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
MaxThreads	Specifies the maximum number of concurrent requests for Batch CUD (Create, Update, Delete) operations.
MultiThreadCount	Aggregate queries in partitioned collections will require parallel requests for different partition ranges. Set this to the number of parallel request to be issued in the same time.
Other	These hidden properties are used only in specific use cases.
Pagesize	The maximum number of results to return per page from Cosmos DB.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.
PseudoColumns	This property indicates whether or not to include pseudo columns as columns to the table.
Readonly	You can use this property to enforce read-only access to Cosmos DB from the provider.
RowScanDepth	The maximum number of rows to scan to look for the columns available in a table.
RTK	The runtime key used for licensing.
SeparatorCharacter	The character or characters used to denote hierarchy.
SetPartitionKeyAsPK	Whether or not to use the collection's Partition Key field as part of composite Primary Key for the corresponding exposed table.

SupportEnhancedSQL	This property enhances SQL functionality beyond what can be supported through the API directly, by enabling in-memory client-side processing.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
TypeDetectionScheme	Comma-separated options for how the provider will scan the data to determine the fields and datatypes in each document collection.
UseConnectionPooling	This property enables connection pooling.
UseRidAsPk	Set this property to false to switch using the id column as primary key instead the default _rid.
WriteThroughputBudget	Defines the Requests Units (RU) budget per Second that the Batch CUD (Create, Update, Delete) operations should not exceed.

For more details, see <https://cdn.cdata.com/help/EHG/jdbc/>.

14. COUCHBASE

14.1. Connection Options

14.1.1. Authentication

Property	Description
AuthScheme	The type of authentication to use when connecting to Couchbase.
User	The Couchbase user account used to authenticate.
Password	The password used to authenticate the user.
CredentialsFile	Use this property if you need to provide credentials for multiple users or buckets. This file takes priority over other forms of authentication.
Server	The address of the Couchbase server or servers to which you are connecting.
CouchbaseService	Determines the Couchbase service to connect to. Default is N1QL. Available options are N1QL and Analytics.
ConnectionMode	Determines how to connect to the Couchbase server. Must be either Direct or Cloud.
DNSServer	Determines what DNS server to use when retrieving Couchbase Cloud information.
N1QLPort	The port for connecting to the Couchbase N1QL Endpoint.
AnalyticsPort	The port for connecting to the Couchbase Analytics Endpoint.
WebConsolePort	The port for connecting to the Couchbase Web Console.

14.1.2. SSL

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.
SSLClientCertSubject	The subject of the TLS/SSL client certificate.
UseSSL	Whether to negotiate TLS/SSL when connecting to the Couchbase server.
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

14.1.3. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

14.1.4. Proxy

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.
ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.
ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

14.1.5. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.

MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

14.1.6. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.
Dataverse	Which Analytics dataverse to scan when discovering tables.
TypeDetectionScheme	Determines how the provider builds tables and columns from the buckets found in Couchbase.
InferNumSampleValues	The maximum number of values for every field to scan before determining its data type. Applies to Automatic Schema Discovery when TypeDetectionScheme is set to INFER.
InferSampleSize	The maximum number of documents to scan for the columns available in the bucket. Applies to Automatic Schema Discovery when TypeDetectionScheme is set to INFER.
InferSimilarityMetric	Specifies the similarity degree where different schemas will be considered to be the same flavor. Applies to Automatic

	Schema Discovery when TypeDetectionScheme is set to INFER.
FlexibleSchemas	Whether the provider allows queries to use columns that it has not discovered.
ExposeTTL	Specifies whether document TTL information should be exposed.
NumericStrings	Whether to allow string values to be treated as numbers.
IgnoreChildAggregates	Whether the provider exposes aggregate columns that are also available as child tables. Ignored if TableSupport is not set to Full.
TableSupport	How much effort the provider will put into discovering tables on the Couchbase server.
NewChildJoinsMode	Determines the kind of child table model the provider exposes.

14.1.7. Caching

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .
CacheDriver	The database driver to be used to cache data.
CacheConnection	The connection string for the cache database. This property is always used in conjunction with CacheProvider . Setting both properties will override the value set for CacheLocation for caching data.
CacheLocation	Specifies the path to the cache when caching to a file.
CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache .

Offline	Use offline mode to get the data from the cache instead of the live source.
CacheMetadata	This property determines whether or not to cache the table metadata to a file store.

14.1.8. Miscellaneous

Property	Description
AllowJSONParameters	Allows raw JSON to be used in parameters when QueryPassthrough is enabled.
BatchSize	The maximum size of each batch operation to submit.
ChildSeparator	The character or characters used to denote child tables.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the Couchbase when the connection is opened.
CreateTableRamQuota	The default RAM quota, in megabytes, to use when inserting buckets via the CREATE TABLE syntax.
DataverseSeparator	The character or characters used to denote Analytics dataverses and scopes/collections.
FlattenArrays	The number of elements to expose as columns from nested arrays. Ignored if IgnoreChildAggregates is enabled.
FlattenObjects	Set FlattenObjects to true to flatten object properties into columns of their own. Otherwise, objects nested in arrays are returned as strings of JSON.
FlavorSeparator	The character or characters used to denote flavors.
GenerateSchemaFiles	Indicates the user preference as to when schemas should be generated and saved.

InsertNullValues	Determines whether an INSERT should include fields that have NULL values.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.
Pagesize	The maximum number of results to return per page from Couchbase.
PeriodsSeparator	The character or characters used to denote hierarchy.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.
PseudoColumns	This property indicates whether or not to include pseudo columns as columns to the table.
QueryExecutionTimeout	This sets the server-side timeout for the query, which governs how long Couchbase will execute the query before returning a timeout error.
QueryPassthrough	This option passes the query to the Couchbase server as is.
Readonly	You can use this property to enforce read-only access to Couchbase from the provider.
RowScanDepth	The maximum number of rows to scan to look for the columns available in a table.
RTK	The runtime key used for licensing.
StrictComparison	Adjusts how precisely to translate filters on SQL input queries into Couchbase queries. This can be set to a

	comma-separated list of values, where each value can be one of: date, number, boolean, or string.
SupportEnhancedSQL	This property enhances SQL functionality beyond what can be supported through the API directly, by enabling in-memory client-side processing.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
TransactionDurability	Specifies how a document must be stored for a transaction to succeed. Specifies whether to use N1QL transactions when executing queries.
TransactionTimeout	This sets the amount of time a transaction may execute before it is timed out by Couchbase.
UpdateNullValues	Determines whether an UPDATE writes NULL values as NULL, or removes them.
UseCollectionsForDDL	Whether to assume that CREATE TABLE statements use collections instead of flavors. Only takes effect when connecting to Couchbase v7+ and GenerateSchemaFiles is set to OnCreate.
UseConnectionPooling	This property enables connection pooling.
UseTransactions	Specifies whether to use N1QL transactions when executing queries.
ValidateJSONParameters	Allows the provider to validate that string parameters are valid JSON before sending the query to Couchbase.

For more details, see <https://cdn.cdata.com/help/CKG/jdbc/>.

15. CSV

15.1. Connection Options

15.1.1. Authentication

Property	Description
AuthScheme	The type of authentication to use when connecting to remote services.
AccessKey	Your account access key. This value is accessible from your security credentials page.
SecretKey	Your account secret key. This value is accessible from your security credentials page.
ApiKey	The API Key used to identify the user to IBM Cloud.
User	The CSV user account used to authenticate.
Password	The password used to authenticate the user.
SharePointEdition	The edition of SharePoint being used. Set either SharePointOnline or SharePointOnPremise.

15.1.2. Connection

Property	Description
URI	The Uniform Resource Identifier (URI) for the CSV resource location.
Region	The hosting region for your S3-like Web Services.
ProjectId	The id of the project where your Google Cloud Storage instance resides.

OracleNamespace	The Oracle Cloud Object Storage namespace to use.
StorageBaseURL	The URL of a cloud storage service provider.
SimpleUploadLimit	This setting specifies the threshold, in bytes, above which the provider will choose to perform a multipart upload rather than uploading everything in one request.
UseVirtualHosting	If true (default), buckets will be referenced in the request using the <code>hosted-style</code> request: <code>http://yourbucket.s3.amazonaws.com/yourobject</code> . If set to false, the bean will use the <code>path-style</code> request: <code>http://s3.amazonaws.com/yourbucket/yourobject</code> . Note that this property will be set to false, in case of an S3 based custom service when the CustomURL is specified.

15.1.3. AWS Authentication

Property	Description
AWSAccessKey	Your AWS account access key. This value is accessible from your AWS security credentials page.
AWSSecretKey	Your AWS account secret key. This value is accessible from your AWS security credentials page.
AWSRoleARN	The Amazon Resource Name of the role to use when authenticating.
AWSPrincipalArn	The ARN of the SAML Identity provider in your AWS account.
AWSRegion	The hosting region for your Amazon Web Services.
AWSSessionToken	Your AWS session token.
MFASerialNumber	The serial number of the MFA device if one is being used.
MFAToken	The temporary token available from your MFA device.

15.1.4. Azure Authentication

Property	Description
AzureStorageAccount	The name of your Azure storage account.
AzureAccessKey	The storage key associated with your CSV account.
AzureSharedAccessSignature	A shared access key signature that may be used for authentication.
AzureTenant	The Microsoft Online tenant being used to access data. If not specified, your default tenant will be used.
AzureEnvironment	The Azure Environment to use when establishing a connection.

15.1.5. SSO

Property	Description
SSOLoginURL	The identity provider's login URL.
SSOProperties	Additional properties required to connect to the identity provider in a semicolon-separated list.

15.1.6. OAuth

Property	Description
InitiateOAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.
OAuthVersion	The version of OAuth being used.

OAuthClientId	The client ID assigned when you register your application with an OAuth authorization server.
OAuthClientSecret	The client secret assigned when you register your application with an OAuth authorization server.
OAuthAccessToken	The access token for connecting using OAuth.
OAuthAccessTokenSecret	The OAuth access token secret for connecting using OAuth.
OAuthSettingsLocation	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH. Alternatively, this can be held in memory by specifying a value starting with memory://.
CallbackURL	The OAuth callback URL to return to when authenticating. This value must match the callback URL you specify in your app settings.
OAuthGrantType	The grant type for the OAuth flow.
OAuthPasswordGrantMode	How to pass Client ID and Secret with OAuthGrantType is set to Password.
OAuthIncludeCallbackURL	Whether to include the callback URL in an access token request.
OAuthAuthorizationURL	The authorization URL for the OAuth service.
OAuthAccessTokenURL	The URL to retrieve the OAuth access token from.
OAuthRefreshTokenURL	The URL to refresh the OAuth token from.
OAuthRequestTokenURL	The URL the service provides to retrieve request tokens from. This is required in OAuth 1.0.
OAuthVerifier	The verifier code returned from the OAuth authorization URL.
AuthToken	The authentication token used to request and obtain the OAuth Access Token.

AuthKey	The authentication secret used to request and obtain the OAuth Access Token.
OAuthParams	A comma-separated list of other parameters to submit in the request for the OAuth access token in the format paramname=value.
OAuthRefreshToken	The OAuth refresh token for the corresponding OAuth access token.
OAuthExpiresIn	The lifetime in seconds of the OAuth AccessToken.
OAuthTokenTimestamp	The Unix epoch timestamp in milliseconds when the current Access Token was created.

15.1.7. JWT OAuth

Property	Description
OAuthJWTCert	The JWT Certificate store.
OAuthJWTCertType	The type of key store containing the JWT Certificate.
OAuthJWTCertPassword	The password for the OAuth JWT certificate.
OAuthJWTCertSubject	The subject of the OAuth JWT certificate.
OAuthJWTIssuer	The issuer of the Java Web Token.
OAuthJWTSubject	The user subject for which the application is requesting delegated access.

15.1.8. Kerberos

Property	Description
KerberosKDC	The Kerberos Key Distribution Center (KDC) service used to authenticate the user.

KerberosRealm	The Kerberos Realm used to authenticate the user with.
KerberosSPN	The service principal name (SPN) for the Kerberos Domain Controller.
KerberosKeytabFile	The Keytab file containing your pairs of Kerberos principals and encrypted keys.
KerberosServiceRealm	The Kerberos realm of the service.
KerberosServiceKDC	The Kerberos KDC of the service.
KerberosTicketCache	The full file path to an MIT Kerberos credential cache file.

15.1.9. SSL

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.
SSLClientCertSubject	The subject of the TLS/SSL client certificate.
SSLMode	The authentication mechanism to be used when connecting to the FTP or FTPS server.
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

15.1.10. SSH

Property	Description
SSHAuthMode	The authentication method to be used to log on to an SFTP server.
SSHClientCert	A certificate to be used for authenticating the user.
SSHClientCertPassword	The password of the SSHClientCert certificate if it has one.
SSHClientCertType	The type of SSHClientCert certificate.

15.1.11. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

15.1.12. Proxy

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.
ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.
ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

15.1.13. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.

MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

15.1.14. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.
SchemaIniLocation	A path to the directory that contains the schema.ini file.
AggregateFiles	When set to true, the provider will aggregate all of the files located in the URI directory into a single table called AggregatedFiles .
MetadataDiscoveryURI	Used together with AggregateFiles , this property specifies a specific file to read the schema of the AggregatedFiles result set.
TypeDetectionScheme	Determines how to determine the data types of columns.
ColumnCount	The number of columns to detect when dynamically determining columns for the table.
RowScanDepth	The number of rows to scan when dynamically determining columns for the table.

15.1.15. Caching

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .
CacheDriver	The database driver to be used to cache data.
CacheConnection	The connection string for the cache database. This property is always used in conjunction with CacheProvider . Setting both properties will override the value set for CacheLocation for caching data.
CacheLocation	Specifies the path to the cache when caching to a file.
CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache .
Offline	Use offline mode to get the data from the cache instead of the live source.
CacheMetadata	This property determines whether or not to cache the table metadata to a file store.

15.1.16. Data Formatting

Property	Description
IncludeColumnHeaders	Whether to get column names from the first line of the specified files.
FMT	The format to be used to parse all text files.
ExtendedProperties	The Microsoft Jet OLE DB 4.0-compatible extended properties for text files.

RowDelimiter	The character which will be used to detect the end of a CSV row.
SkipTop	Skips the amount of rows specified starting from the top.
IgnoreBlankRows	Indicates whether to skip the empty rows.
IncludeEmptyHeaders	Whether to include empty value for a column name within column header.
SkipHeaderComments	If set to true, skips rows at the top of the file beginning with #.
Charset	Specifies the session character set for encoding and decoding character data transferred to and from the CSV file. The default value is UTF-8.
QuoteEscapeCharacter	Determines the character which will be used to escape quotes.
QuoteCharacter	Determines the character which will be used to quote values.
TrimSpaces	Set to True if you want the provider to trim preceeding and trailing spaces in a cell containing a quoted value.
PushEmptyValuesAsNull	Indicates whether to read the empty values as empty or as null.
NullValues	A comma separated list which will be replaced with nulls if there are found in the CSV file.
PathSeparator	Determines the character which will be used to replace the file separator.
IgnoreIncompleteRows	Indicates whether to ignore incomplete rows.
MaxCellLength	Max length a cell can hold, after passing this number the cell content will get truncated. If value is -1 then we don't limit the cell content length.

15.1.17. Miscellaneous

Property	Description
BatchSize	The maximum size of each batch operation to submit.
ClientCulture	This property can be used to specify the format of data (e.g., currency values) that is accepted by the client application. This property can be used when the client application does not support the machine's culture settings. For example, Microsoft Access requires 'en-US'.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the CSV when the connection is opened.
Culture	This setting can be used to specify culture settings that determine how the provider interprets certain data types that are passed into the provider. For example, setting Culture='de-DE' will output German formats even on an American machine.
CustomHeaders	Other headers as determined by the user (optional).
CustomUrlParams	The custom query string to be included in the request.
DirectoryRetrievalDepth	Limit the subfolders recursively scanned when IncludeSubdirectories is enabled.
ExcludeFileExtensions	Set to true if file extensions should be excluded from table names.
ExcludeFiles	Comma-separated list of file extensions to exclude from the set of the files modeled as tables.

GenerateSchemaFiles	Indicates the user preference as to when schemas should be generated and saved.
IncludeDropboxTeamResources	Indicates if you want to include Dropbox team files and folders.
IncludeFiles	Comma-separated list of file extensions to include into the set of the files modeled as tables.
IncludeSubdirectories	Whether to read files from nested folders. In the case of a name collision, table names are prefixed by the underscore-separated folder names.
InsertMode	Specifies the mode for inserting data into CSV files.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.
Pagesize	The maximum number of results to return per page from CSV.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.
PseudoColumns	This property indicates whether or not to include pseudo columns as columns to the table.
Readonly	You can use this property to enforce read-only access to CSV from the provider.
RTK	The runtime key used for licensing.

SupportEnhancedSQL	This property enhances SQL functionality beyond what can be supported through the API directly, by enabling in-memory client-side processing.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
TruncateOnInserts	Set to True if you want the provider to truncate on every (batch) insert.
UseConnectionPooling	This property enables connection pooling.
UseRowNumbers	Set this to true if you are deleting or updating in CSV and you do not want to specify a custom schema. This will create a new column with the name RowNumber which will be used as key for that table.

For more details, see <https://cdn.cdata.com/help/RVG/jdbc/>.

16. DATABRICKS

16.1. Connection Options

16.1.1. [Authentication](#)

Property	Description
AuthScheme	The authentication scheme used. Accepted entries are PersonalAccessToken, AzureServicePrincipal.
Server	The host name or IP address of the server hosting the Databricks database.
ProtocolVersion	The Protocol Version used to authenticate with Databricks.
Database	The name of the Databricks database.
HTTPPath	The path component of the URL endpoint.
Token	The token used to access the Databricks server.

16.1.2. [AWS Authentication](#)

Property	Description
AWSAccessKey	Your AWS account access key. This value is accessible from your AWS security credentials page.
AWSSecretKey	Your AWS account secret key. This value is accessible from your AWS security credentials page.
AWSRegion	The hosting region for your Amazon Web Services.
AWSS3Bucket	The name of your AWS S3 bucket.

16.1.3. [Azure Authentication](#)

Property	Description
AzureStorageAccount	The name of your Azure storage account.
AzureAccessKey	The storage key associated with your Databricks account.
AzureBlobContainer	The name of your Azure Blob storage container.

16.1.4. [AzureServicePrincipal Authentication](#)

Property	Description
AzureTenantId	The Tenant id of your Microsoft Azure Active Directory.
AzureClientId	The application(client) id of your Microsoft Azure Active Directory application.
AzureClientSecret	The application(client) secret of your Microsoft Azure Active Directory application.
AzureSubscriptionId	The Subscription id of your Azure Databricks Service Workspace.
AzureResourceGroup	The Resource Group name of your Azure Databricks Service Workspace.
AzureWorkspace	The name of your Azure Databricks Service Workspace.

16.1.5. [SSL](#)

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.
SSLClientCertSubject	The subject of the TLS/SSL client certificate.
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

16.1.6. [Firewall](#)

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

16.1.7. [Proxy](#)

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.
ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.
ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

16.1.8. [Logging](#)

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.

LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

16.1.9. [Schema](#)

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.
PrimaryKeyIdentifiers	Set this property to define primary keys.

16.1.10. [Caching](#)

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .
CacheDriver	The database driver used to cache data.

CacheConnection	The connection string for the cache database. This property is always used in conjunction with CacheProvider . Setting both properties will override the value set for CacheLocation for caching data.
CacheLocation	Specifies the path to the cache when caching to a file.
CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache .
Offline	Use offline mode to get the data from the cache instead of the live source.
CacheMetadata	This property determines whether or not to cache the table metadata to a file store.

16.1.11. [Databricks](#)

Property	Description
CloudStorageType	Determine which cloud storage service will be used.
StoreTableInCloud	This option specifies whether Databricks server will create and save tables in cloud storage.
QueryTableDetails	Specifies whether to use DESCRIBE FORMATTED ... to query detailed table information, the query will take a long time if set it to True.
UseUploadApi	This option specifies whether the Databricks Upload API will be used when executing Bulk Insert operations.
UseCloudFetch	This option specifies whether to use CloudWatch to improve query efficiency when the data volume of the table is large.
UseLegacyDataModel	This option specifies whether to support Unity Catalog.

16.1.12. Miscellaneous

Property	Description
ApplicationName	The application name connection string property expresses the HTTP User-Agent.
AsyncQueryTimeout	The timeout for asynchronous requests issued by the provider to download large result sets.
BatchSize	The maximum size of each batch operation to submit.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property specifies whether to connect to the Databricks when the connection is opened.
DescribeCommand	The describe command used to communicate with the Hive server. Accepted entries are DESCRIBE and DESC.
DetectView	Specifies whether to use DESCRIBE FORMATTED ... to detect the specified table is view or not.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.
PseudoColumns	This property indicates whether or not to include pseudo columns as columns to the table.

QueryPassthrough	This option passes the query to the Databricks server as is.
Readonly	You can use this property to enforce read-only access to Databricks from the provider.
RTK	The runtime key used for licensing.
ServerConfigurations	A name-value list of server configuration variables to override the server defaults.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
UseConnectionPooling	This property enables connection pooling.
UseDescTableQuery	This option specifies whether the columns will be retrieved using a DESC TABLE query or the GetColumns Thrift API. The GetColumns Thrift API works for the Apache Spark 3.0.0 or later.
UseInsertSelectSyntax	Specifies whether to use an INSERT INTO SELECT statement.
UserDefinedViews	A filepath pointing to the JSON configuration file containing your custom views.

For more details, see [CData JDBC Driver for Databricks - CData JDBC Driver for Databricks](#).

17. ELASTICSEARCH

17.1. Elasticsearch Version Support

The driver models Elasticsearch data as a read/write, relational database. The driver can connect to Elasticsearch v2.2.0 and above via the REST API.

17.2. Connection Options

17.2.1. Authentication

Property	Description
AuthScheme	The scheme used for authentication. Accepted entries are None, Basic, Negotiate (Kerberos), AwsRootKeys, AwsIAMRoles, and APIKey. None is the default.
User	The user who is authenticating to Elasticsearch.
Password	The password used to authenticate to Elasticsearch.
Server	The host name or IP address of the Elasticsearch REST server.
Port	The port for the Elasticsearch REST server.
APIKey	The APIKey used to authenticate to Elasticsearch.
APIKeyId	The APIKey Id to authenticate to Elasticsearch.

17.2.2. Connection

Property	Description
DataModel	Specifies the data model to use when parsing Elasticsearch documents and generating the database metadata.

17.2.3. AWS Authentication

Property	Description
AWSAccessKey	Your AWS account access key. This value is accessible from your AWS security credentials page.
AWSSecretKey	Your AWS account secret key. This value is accessible from your AWS security credentials page.
AWSRoleARN	The Amazon Resource Name of the role to use when authenticating.
AWSRegion	The hosting region for your Amazon Web Services.
AWSSessionToken	Your AWS session token.
TemporaryTokenDuration	The amount of time (in seconds) an AWS temporary token will last.

17.2.4. Kerberos

Property	Description
KerberosKDC	The Kerberos Key Distribution Center (KDC) service used to authenticate the user.
KerberosRealm	The Kerberos Realm used to authenticate the user with.
KerberosSPN	The service principal name (SPN) for the Kerberos Domain Controller.
KerberosKeytabFile	The Keytab file containing your pairs of Kerberos principals and encrypted keys.
KerberosServiceRealm	The Kerberos realm of the service.
KerberosServiceKDC	The Kerberos KDC of the service.

KerberosTicketCache	The full file path to an MIT Kerberos credential cache file.

17.2.5. SSL

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.
SSLClientCertSubject	The subject of the TLS/SSL client certificate.
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

17.2.6. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

17.2.7. Proxy

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.
ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.
ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

17.2.8. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).

MaxLogFileCount	A string specifying the maximum file count of log files.

17.2.9. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.
FlattenObjects	Set FlattenObjects to true to flatten object properties into columns of their own. Otherwise, objects nested in arrays are returned as strings of JSON.
FlattenArrays	Set FlattenArrays to the number of nested array elements you want to return as table columns. By default, nested arrays are returned as strings of JSON.

17.2.10. Caching

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .
CacheDriver	The database driver to be used to cache data.

CacheConnection	The connection string for the cache database. This property is always used in conjunction with CacheProvider . Setting both properties will override the value set for CacheLocation for caching data.
CacheLocation	Specifies the path to the cache when caching to a file.
CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache .
Offline	Use offline mode to get the data from the cache instead of the live source.
CacheMetadata	This property determines whether or not to cache the table metadata to a file store.

17.2.11. Miscellaneous

Property	Description
BatchSize	The maximum size of each batch operation to submit.
ClientSideEvaluation	Set ClientSideEvaluation to true to perform Evaluation client side on nested objects.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the Elasticsearch when the connection is opened.
GenerateSchemaFiles	Indicates the user preference as to when schemas should be generated and saved.
MaxResults	The maximum number of total results to return from Elasticsearch when using the default Search API.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.

Other	These hidden properties are used only in specific use cases.
PageSize	The number of results to return per request from Elasticsearch.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.
PseudoColumns	This property indicates whether or not to include pseudo columns as columns to the table.
QueryPassthrough	This option allows you to pass exact queries to Elasticsearch.
Readonly	You can use this property to enforce read-only access to Elasticsearch from the provider.
RowScanDepth	The maximum number of rows to scan when generating table metadata. Set this property to gain more control over how the provider detects arrays.
RTK	The runtime key used for licensing.
ScrollDuration	Specifies the time unit to use when retrieving results via the Scroll API.
SupportEnhancedSQL	This property enhances SQL functionality beyond what can be supported through the API directly, by enabling in-memory client-side processing.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
UseConnectionPooling	This property enables connection pooling.

For more details, see <https://cdn.cdata.com/help/EEG/jdbc/>.

18. EXCEL

18.1. Excel Version Support

The driver supports the Office Open XML spreadsheets used by Excel 2007 and later.

18.2. Connection Options

18.2.1. Authentication

Property	Description
AuthScheme	The type of authentication to use when connecting to remote services.
User	The Excel user account used to authenticate.
Password	The password used to authenticate the user.
SharePointEdition	The edition of SharePoint being used. Set either SharePointOnline or SharePointOnPremise.

18.2.2. Connection

Property	Description
URI	The Uniform Resource Identifier (URI) for the Excel resource location.
DefineTables	Map Excel ranges to table names.
Header	Indicates whether the first row should be used as a column header.
Orientation	Indicates whether the data in Excel is laid out horizontally or vertically.

ExcelFile	The location of an Excel file.
BufferChanges	Indicates whether to hold changes to the data in memory until the connection is closed.
Region	The hosting region for your S3-like Web Services.
ProjectId	The id of the project where your Google Cloud Storage instance resides.
OracleNamespace	The Oracle Cloud Object Storage namespace to use.
StorageBaseURL	The URL of a cloud storage service provider.
UseVirtualHosting	If true (default), buckets will be referenced in the request using the <code>hosted-style</code> request: <code>http://yourbucket.s3.amazonaws.com/yourobject</code> . If set to false, the bean will use the <code>path-style</code> request: <code>http://s3.amazonaws.com/yourbucket/yourobject</code> . Note that this property will be set to false, in case of an S3 based custom service when the CustomURL is specified.

18.2.3. AWS Authentication

Property	Description
AWSAccessKey	Your AWS account access key. This value is accessible from your AWS security credentials page.
AWSSecretKey	Your AWS account secret key. This value is accessible from your AWS security credentials page.
AWSRoleARN	The Amazon Resource Name of the role to use when authenticating.
AWSPrincipalArn	The ARN of the SAML Identity provider in your AWS account.
AWSRegion	The hosting region for your Amazon Web Services.
AWSSessionToken	Your AWS session token.

MFASerialNumber	The serial number of the MFA device if one is being used.
MFAToken	The temporary token available from your MFA device.

18.2.4. Azure Authentication

Property	Description
AzureStorageAccount	The name of your Azure storage account.
AzureAccessKey	The storage key associated with your Excel account.
AzureSharedAccessSignature	A shared access key signature that may be used for authentication.
AzureTenant	The Microsoft Online tenant being used to access data. If not specified, your default tenant will be used.
AzureEnvironment	The Azure Environment to use when establishing a connection.

18.2.5. SSO

Property	Description
SSOLoginURL	The identity provider's login URL.
SSOProperties	Additional properties required to connect to the identity provider in a semicolon-separated list.

18.2.6. Data

Property	Description
AllowFormula	Whether or not to treat values starting with an equals (=) sign as formulas during inserts and updates.
EmptyValueMode	Indicates whether to read the empty values as empty or as null.
NullValues	A comma separated list which will be replaced with nulls if there are found in the CSV file.
IgnoreCalcError	Indicates whether to ignore errors that occurred during the calculation.
NullValueMode	Indicates whether to read empty cells as null or as empty.
HasCrossSheetReferences	Indicates how cross sheet references are handled.
Recalculate	Indicates whether to recalculate all formulas when data is read.
ShowEmptyRows	Indicates whether or not the empty rows should be pushed.
TypeDetectionScheme	Determines how the provider detects the data types of columns.
UseFastInsert	Indicates whether to use the fastInsert when executing Insert Statement.

18.2.7. OAuth

Property	Description
InitiateOAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.

OAuthVersion	The version of OAuth being used.
OAuthClientId	The client ID assigned when you register your application with an OAuth authorization server.
OAuthClientSecret	The client secret assigned when you register your application with an OAuth authorization server.
OAuthAccessToken	The access token for connecting using OAuth.
OAuthAccessTokenSecret	The OAuth access token secret for connecting using OAuth.
OAuthSettingsLocation	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH. Alternatively, this can be held in memory by specifying a value starting with memory://.
CallbackURL	The OAuth callback URL to return to when authenticating. This value must match the callback URL you specify in your app settings.
OAuthGrantType	The grant type for the OAuth flow.
OAuthPasswordGrantMode	How to pass Client ID and Secret with OAuthGrantType is set to Password.
OAuthIncludeCallbackURL	Whether to include the callback URL in an access token request.
OAuthAuthorizationURL	The authorization URL for the OAuth service.
OAuthAccessTokenURL	The URL to retrieve the OAuth access token from.
OAuthRefreshTokenURL	The URL to refresh the OAuth token from.
OAuthRequestTokenURL	The URL the service provides to retrieve request tokens from. This is required in OAuth 1.0.
OAuthVerifier	The verifier code returned from the OAuth authorization URL.
AuthToken	The authentication token used to request and obtain the OAuth Access Token.

AuthKey	The authentication secret used to request and obtain the OAuth Access Token.
OAuthParams	A comma-separated list of other parameters to submit in the request for the OAuth access token in the format paramname=value.
OAuthRefreshToken	The OAuth refresh token for the corresponding OAuth access token.
OAuthExpiresIn	The lifetime in seconds of the OAuth AccessToken.
OAuthTokenTimestamp	The Unix epoch timestamp in milliseconds when the current Access Token was created.

18.2.8. JWT OAuth

Property	Description
OAuthJWTCert	The JWT Certificate store.
OAuthJWTCertType	The type of key store containing the JWT Certificate.
OAuthJWTCertPassword	The password for the OAuth JWT certificate.
OAuthJWTCertSubject	The subject of the OAuth JWT certificate.
OAuthJWTIssuer	The issuer of the Java Web Token.
OAuthJWTSubject	The user subject for which the application is requesting delegated access.

18.2.9. Kerberos

Property	Description
KerberosKeytabFile	The Keytab file containing your pairs of Kerberos principals and encrypted keys.

18.2.10. SSL

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.
SSLClientCertSubject	The subject of the TLS/SSL client certificate.
SSLMode	The authentication mechanism to be used when connecting to the FTP or FTPS server.
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

18.2.11. SSH

Property	Description
SSHAuthMode	The authentication method to be used to log on to an SFTP server.
SSHClientCert	A certificate to be used for authenticating the user.
SSHClientCertPassword	The password of the SSHClientCert certificate if it has one.
SSHClientCertType	The type of SSHClientCert certificate.

18.2.12. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

18.2.13. Proxy

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.
ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.
ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.

ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

18.2.14. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

18.2.15. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.

18.2.16. Caching

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .
CacheDriver	The database driver to be used to cache data.
CacheConnection	The connection string for the cache database. This property is always used in conjunction with CacheProvider . Setting both properties will override the value set for CacheLocation for caching data.
CacheLocation	Specifies the path to the cache when caching to a file.
CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache .
Offline	Use offline mode to get the data from the cache instead of the live source.
CacheMetadata	This property determines whether or not to cache the table metadata to a file store.

18.2.17. Miscellaneous

Property	Description
BatchSize	The maximum size of each batch operation to submit.
Charset	Specifies the session character set for encoding and decoding character data transferred to and from the Excel file. The default value is UTF-8.
ClientCulture	This property can be used to specify the format of data (e.g., currency values) that is accepted by the

	client application. This property can be used when the client application does not support the machine's culture settings. For example, Microsoft Access requires 'en-US'.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the Excel when the connection is opened.
Culture	This setting can be used to specify culture settings that determine how the provider interprets certain data types that are passed into the provider. For example, setting Culture='de-DE' will output German formats even on an American machine.
CustomHeaders	Other headers as determined by the user (optional).
CustomUrlParams	The custom query string to be included in the request.
IncludeDropboxTeamResources	Indicates if you want to include Dropbox team files and folders.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.

PseudoColumns	This property indicates whether or not to include pseudo columns as columns to the table.
Readonly	You can use this property to enforce read-only access to Excel from the provider.
RowScanDepth	The maximum number of rows to scan to look for the columns available in a table.
RTK	The runtime key used for licensing.
SupportEnhancedSQL	This property enhances SQL functionality beyond what can be supported through the API directly, by enabling in-memory client-side processing.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
UseConnectionPooling	This property enables connection pooling.

For more details, see <https://cdn.cdata.com/help/RXG/jdbc/>.

19. FINANCIALFORCE

19.1. Connection Options

19.1.1. Authentication

Property	Description
AuthScheme	The type of authentication to use when connecting to FinancialForce.
User	The FinancialForce user account used to authenticate.
Password	The password used to authenticate the user.
SecurityToken	The security token used to authenticate access to the FinancialForce account.
UseSandbox	A boolean determining if the connection should be made to a FinancialForce sandbox account.

19.1.2. Connection

Property	Description
APIVersion	The version of the FinancialForce API used.
LoginURL	URL to the FinancialForce server used for logging in.

19.1.3. SSO

Property	Description
SSOLoginURL	The identity provider's login URL.

SSOProperties	Additional properties required to connect to the identity provider in a semicolon-separated list.
SSOExchangeUrl	The url used for consuming the SAML response and exchanging it with FinancialForce specific credentials.

19.1.4. BulkAPI

Property	Description
UseBulkAPI	Whether to use the synchronous SOAP API or the asynchronous Bulk API.
BulkAPIVersion	The version of the bulk API to use for processing queries.
BulkAPIConcurrencyMode	The concurrency mode for processing bulk rows with BULK API v1.
BulkPollingInterval	The time interval in milliseconds between requests that check the availability of the bulk query response. The default value is 500 ms.
BulkQueryTimeout	The timeout in minutes for which the provider will wait for a bulk query response. The default value is 25 minutes.
WaitForBulkResults	Whether to wait for bulk results when using the asynchronous API. Only active when UseBulkAPI is true.

19.1.5. OAuth

Property	Description
InitiateOAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.

OAuthClientId	The client ID assigned when you register your application with an OAuth authorization server.
OAuthClientSecret	The client secret assigned when you register your application with an OAuth authorization server.
OAuthAccessToken	The access token for connecting using OAuth.
OAuthSettingsLocation	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH. Alternatively, this can be held in memory by specifying a value starting with memory://.
CallbackURL	The OAuth callback URL to return to when authenticating. This value must match the callback URL you specify in your app settings.
OAuthServerURL	The server URL to use if authenticating with OAuth.
OAuthGrantType	The grant type for the OAuth flow.
OAuthVerifier	The verifier code returned from the OAuth authorization URL.
OAuthRefreshToken	The OAuth refresh token for the corresponding OAuth access token.
OAuthExpiresIn	The lifetime in seconds of the OAuth AccessToken.
OAuthTokenTimestamp	The Unix epoch timestamp in milliseconds when the current Access Token was created.

19.1.6. SSL

Property	Description
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

19.1.7. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

19.1.8. Proxy

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.
ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.

ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

19.1.9. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

19.1.10. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.

Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.

19.1.11. Caching

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .
CacheDriver	The database driver to be used to cache data.
CacheConnection	The connection string for the cache database. This property is always used in conjunction with CacheProvider . Setting both properties will override the value set for CacheLocation for caching data.
CacheLocation	Specifies the path to the cache when caching to a file.
CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache .
Offline	Use offline mode to get the data from the cache instead of the live source.
CacheMetadata	This property determines whether or not to cache the table metadata to a file store.

19.1.12. Miscellaneous

Property	Description
AllOrNone	A boolean indicating if you would like all inserts, updates, or deletes to fail in a request if even a single record fails.

ArchiveMode	Boolean indicating whether to include deleted and archived records with a standard SELECT query.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the FinancialForce when the connection is opened.
ContinueOnAlterException	Whether you want to continue after a ALTER statement has failed.
FilterScope	Optional scope to limit the records returned from queries.
IncludeMetadataDescription	Set this property to a value other than NONE if you want to retrieve the descriptions for columns, tables or both of them from the Metadata API.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.

PseudoColumns	This property indicates whether or not to include pseudo columns as columns to the table.
Readonly	You can use this property to enforce read-only access to FinancialForce from the provider.
RTK	The runtime key used for licensing.
ServerSideAggregation	A boolean determining if server side aggregation should be used.
SessionTimeout	The time in minutes for which a FinancialForce login session is reused.
SkipFormulaFields	Set to true if formula fields should be skipped when listing columns.
SupportEnhancedSQL	This property enhances SQL functionality beyond what can be supported through the API directly, by enabling in-memory client-side processing.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
UseConnectionPooling	This property enables connection pooling.
UseDisplayNames	Boolean determining if the display names for the columns should be used instead of the API names.

For more details, see <https://cdn.cdata.com/help/HFG/jdbc/>.

20. EXCEL ONLINE



20.1. Connection Options

20.1.1. Authentication

Property	Description
AuthScheme	The type of authentication to use when connecting to Excel Online.
UseSandbox	A boolean indicating if you are using a sandbox account. The provider makes requests to the production environment by default.

20.1.2. Connection

Property	Description
DefineTables	Assign table names to ranges.
Workbook	The name or Id of the workbook.
Drive	The Id of the drive.
IncludeSharePointSites	Whether to retrieve drives for all SharePoint sites when querying Drives view. If 'true' the provider will retrieve all Site IDs recursively and for each of them issue a separate call to get their drives. Therefore, be aware that setting this property to 'true' may decrease performance for Drives view.

20.1.3. Azure Authentication

Property	Description
AzureTenant	The Microsoft Online tenant being used to access data. If not specified, your default tenant will be used.
AzureEnvironment	The Azure Environment to use when establishing a connection.

20.1.4. OAuth

Property	Description
InitiateOAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.
OAuthClientId	The client ID assigned when you register your application with an OAuth authorization server.
OAuthClientSecret	The client secret assigned when you register your application with an OAuth authorization server.
OAuthAccessToken	The access token for connecting using OAuth.
OAuthSettingsLocation	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH. Alternatively, this can be held in memory by specifying a value starting with memory://.
CallbackURL	The OAuth callback URL to return to when authenticating. This value must match the callback URL you specify in your app settings.
OAuthGrantType	The grant type for the OAuth flow.

OAuthVerifier	The verifier code returned from the OAuth authorization URL.
OAuthRefreshToken	The OAuth refresh token for the corresponding OAuth access token.
OAuthExpiresIn	The lifetime in seconds of the OAuth AccessToken.
OAuthTokenTimestamp	The Unix epoch timestamp in milliseconds when the current Access Token was created.

20.1.5. SSL

Property	Description
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

20.1.6. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

20.1.7. Proxy

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.
ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.
ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

20.1.8. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.

LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

20.1.9. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.

20.1.10. Caching

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .
CacheDriver	The database driver to be used to cache data.
CacheConnection	The connection string for the cache database. This property is always used in conjunction with

	CacheProvider . Setting both properties will override the value set for CacheLocation for caching data.
CacheLocation	Specifies the path to the cache when caching to a file.
CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache .
Offline	Use offline mode to get the data from the cache instead of the live source.
CacheMetadata	This property determines whether or not to cache the table metadata to a file store.

20.1.11. Miscellaneous

Property	Description
AllowFormula	Whether or not to treat values starting with an equals (=) sign as formulas during inserts and updates.
BatchSize	The maximum size of each batch operation to submit.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the Excel Online when the connection is opened.
Header	Indicates whether or not the provider should detect column names from the first row.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.

Other	These hidden properties are used only in specific use cases.
Pagesize	The maximum number of results to return per page from Excel Online when UsePagination is True.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.
PseudoColumns	This property indicates whether or not to include pseudo columns as columns to the table.
Readonly	You can use this property to enforce read-only access to Excel Online from the provider.
RowScanDepth	The maximum number of rows to scan to look for the columns available in a table.
RTK	The runtime key used for licensing.
SharepointURL	The base URL of your Sharepoint Server.
ShowSharedDocuments	Whether or not to show shared documents.
SupportEnhancedSQL	This property enhances SQL functionality beyond what can be supported through the API directly, by enabling in-memory client-side processing.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
TypeDetectionScheme	Determines how to determine the data types of columns.
UseConnectionPooling	This property enables connection pooling.

UsePagination	Whether or not the CData ADO.NET Provider for Excel Online should use client side paging.
UseSimpleNames	Boolean determining if simple names should be used for tables and columns.
ValueInputOption	Determines how the provider parses values the user submits to Excel Online.
ValueRenderOption	Determines how the provider renders values in the output.

For more details, see <https://cdn.cdata.com/help/FXG/jdbc/>.

21. GOOGLE DATA CATALOG

21.1. Connection Options

21.1.1. Authentication

Property	Description
AuthScheme	The type of authentication to use when connecting to Google Data Catalog.
OrganizationId	The ID associated with the Google Cloud Platform organization resource to connect to.
ProjectId	The ID associated with the Google Cloud Platform project resource you would like to connect to.

21.1.2. OAuth

Property	Description
InitiateOAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.
OAuthClientId	The client ID assigned when you register your application with an OAuth authorization server.
OAuthClientSecret	The client secret assigned when you register your application with an OAuth authorization server.
OAuthAccessToken	The access token for connecting using OAuth.
OAuthSettingsLocation	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH. Alternatively, this can be held in memory by specifying a value starting with memory://.

CallbackURL	The OAuth callback URL to return to when authenticating. This value must match the callback URL you specify in your app settings.
OAuthVerifier	The verifier code returned from the OAuth authorization URL.
OAuthRefreshToken	The OAuth refresh token for the corresponding OAuth access token.
OAuthExpiresIn	The lifetime in seconds of the OAuth AccessToken.
OAuthTokenTimestamp	The Unix epoch timestamp in milliseconds when the current Access Token was created.

21.1.3. JWT OAuth

Property	Description
OAuthJWTCert	The JWT Certificate store.
OAuthJWTCertType	The type of key store containing the JWT Certificate.
OAuthJWTCertPassword	The password for the OAuth JWT certificate.
OAuthJWTCertSubject	The subject of the OAuth JWT certificate.
OAuthJWTIssuer	The issuer of the Java Web Token.
OAuthJWTSubject	The user subject for which the application is requesting delegated access.

21.1.4. SSL

Property	Description
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

21.1.5. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

21.1.6. Proxy

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.

ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.
ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

21.1.7. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

21.1.8. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.

21.1.9. Caching

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .
CacheDriver	The database driver to be used to cache data.
CacheConnection	The connection string for the cache database. This property is always used in conjunction with CacheProvider . Setting both properties will override the value set for CacheLocation for caching data.
CacheLocation	Specifies the path to the cache when caching to a file.
CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache .

Offline	Use offline mode to get the data from the cache instead of the live source.
CacheMetadata	This property determines whether or not to cache the table metadata to a file store.

21.1.10. Miscellaneous

Property	Description
BatchSize	The maximum size of each batch operation to submit.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the Google Data Catalog when the connection is opened.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.
Pagesize	The maximum number of results to return per page from Google Data Catalog.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.

PseudoColumns	This property indicates whether or not to include pseudo columns as columns to the table.
RTK	The runtime key used for licensing.
SupportEnhancedSQL	This property enhances SQL functionality beyond what can be supported through the API directly, by enabling in-memory client-side processing.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
UseConnectionPooling	This property enables connection pooling.

For more details, see <https://cdn.cdata.com/help/HGG/jdbc/>.

22. GOOGLE SPANNER

22.1. Connection Options

22.1.1. Authentication

Property	Description
AuthScheme	The type of authentication to use when connecting to Google Spanner.
InstancelId	The id of the Google Spanner instance to which you are connecting.
Database	The name of the Google Spanner database to connect to.
ProjectId	The id of the project where your Google Spanner instance resides.
UseGrpcEndpoint	Whether to use the Grpc endpoint or not.

22.1.2. OAuth

Property	Description
InitiateOAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.
OAuthClientId	The client ID assigned when you register your application with an OAuth authorization server.
OAuthClientSecret	The client secret assigned when you register your application with an OAuth authorization server.
OAuthAccessToken	The access token for connecting using OAuth.

OAuthSettingsLocation	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH. Alternatively, this can be held in memory by specifying a value starting with memory://.
OAuthVerifier	The verifier code returned from the OAuth authorization URL.
OAuthRefreshToken	The OAuth refresh token for the corresponding OAuth access token.
OAuthExpiresIn	The lifetime in seconds of the OAuth AccessToken.
OAuthTokenTimestamp	The Unix epoch timestamp in milliseconds when the current Access Token was created.

22.1.3. JWT OAuth

Property	Description
OAuthJWTCert	The JWT Certificate store.
OAuthJWTCertType	The type of key store containing the JWT Certificate.
OAuthJWTCertPassword	The password for the OAuth JWT certificate.
OAuthJWTCertSubject	The subject of the OAuth JWT certificate.
OAuthJWTIssuer	The issuer of the Java Web Token.
OAuthJWTSubject	The user subject for which the application is requesting delegated access.

22.1.4. SSL

Property	Description
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

22.1.5. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

22.1.6. Proxy

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.

ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.
ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

22.1.7. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

22.1.8. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.
Schema	The schema which will be used by default.

22.1.9. Caching

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .
CacheDriver	The database driver to be used to cache data.
CacheConnection	The connection string for the cache database. This property is always used in conjunction with CacheProvider . Setting both properties will override the value set for CacheLocation for caching data.
CacheLocation	Specifies the path to the cache when caching to a file.

CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache .
Offline	Use offline mode to get the data from the cache instead of the live source.
CacheMetadata	This property determines whether or not to cache the table metadata to a file store.

22.1.10. Miscellaneous

Property	Description
BatchSize	The maximum size of each batch operation to submit.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the Google Spanner when the connection is opened.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.

PseudoColumns	This property indicates whether or not to include pseudo columns as columns to the table.
QueryPassthrough	This option passes the query to the Google Spanner server as is.
Readonly	You can use this property to enforce read-only access to Google Spanner from the provider.
RTK	The runtime key used for licensing.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
UseConnectionPooling	This property enables connection pooling.

For more details, see <https://cdn.cdata.com/help/OGG/jdbc/>.

23. HBASE

23.1. Apache HBase Version Support

The driver models HBase data as a read/write, relational database. The driver can connect to HBase REST Server 0.0.3, available in HBase version 0.98 and above.

23.2. Connection Options

23.2.1. Authentication

Property	Description
AuthScheme	The scheme used for authentication. Accepted entries are None, Basic, and Negotiate (Kerberos). None is the default.
Server	The host name or IP address of the Apache HBase REST server.
Port	The port for the Apache HBase REST server.
User	The user who is authenticating to Apache HBase.
Password	The password used to authenticate to Apache HBase.

23.2.2. Kerberos

Property	Description
KerberosKDC	The Kerberos Key Distribution Center (KDC) service used to authenticate the user.
KerberosRealm	The Kerberos Realm used to authenticate the user with.
KerberosSPN	The service principal name (SPN) for the Kerberos Domain Controller.

KerberosKeytabFile	The Keytab file containing your pairs of Kerberos principals and encrypted keys.
KerberosServiceRealm	The Kerberos realm of the service.
KerberosServiceKDC	The Kerberos KDC of the service.
KerberosTicketCache	The full file path to an MIT Kerberos credential cache file.

23.2.3. SSL

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.
SSLClientCertSubject	The subject of the TLS/SSL client certificate.
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

23.2.4. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.

FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

23.2.5. Proxy

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.
ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.
ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

23.2.6. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

23.2.7. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.

23.2.8. Caching

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .
CacheDriver	The database driver to be used to cache data.
CacheConnection	The connection string for the cache database. This property is always used in conjunction with CacheProvider . Setting both properties will override the value set for CacheLocation for caching data.
CacheLocation	Specifies the path to the cache when caching to a file.
CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache .
Offline	Use offline mode to get the data from the cache instead of the live source.
CacheMetadata	This property determines whether or not to cache the table metadata to a file store.

23.2.9. Miscellaneous

Property	Description
BatchSize	The maximum size of each batch operation to submit.
ColumnEncoding	The Encoding used for the column values. Accepted entries are None, UTF8, and BASE64. None is the default.

ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the Apache HBase when the connection is opened.
DatetimeFormat	The format used when inserting datetime values into the database.
IncludeDisabledTables	Specifies whether to retrieve the disabled table.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.
Pagesize	The maximum number of results to return per page from Apache HBase.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.
PseudoColumns	This property indicates whether or not to include pseudo columns as columns to the table.
Readonly	You can use this property to enforce read-only access to Apache HBase from the provider.

ResetTableMetadataOnDelete	Specifies whether to reset the catch table after executing the Delete Statement.
RetrieveSelectedColumnsOnly	Specifies whether to retrieve selected columns only when executing a SELECT statement.
RowScanDepth	The maximum number of rows to scan to look for the columns available in a table.
RTK	The runtime key used for licensing.
SupportEnhancedSQL	This property enhances SQL functionality beyond what can be supported through the API directly, by enabling in-memory client-side processing.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
TypeDetectionScheme	Determines how to determine the data type of columns.
UseConnectionPooling	This property enables connection pooling.
UseSQLFiltering	Specifies whether to use the ScannerFilter when executing the Select Statement.

For more details, see <https://cdn.cdata.com/help/RHG/jdbc/>.

24. HIVE

24.1. Apache Hive Version Support

The driver models Apache Hive instances as relational databases. Hive versions 0.11.0 and above are supported.

24.2. Connection Options

24.2.1. Authentication

Property	Description
AuthScheme	The authentication scheme used. Accepted entries are Plain, LDAP, NoSasl, and Kerberos.
Server	The host name or IP address of the server hosting HiveServer2.
Port	The port for the connection to the HiveServer2 instance.
User	The username used to authenticate with Hive.
Password	The password used to authenticate with Hive.
ProtocolVersion	The Protocol Version used to authenticate with Hive.
TransportMode	The transport mode to use to communicate with the Hive server. Accepted entries are BINARY and HTTP.
Database	The name of the Hive database to use by default.
ImpersonationProxyUser	The proxy user of the Hive user impersonation.

24.2.2. Kerberos

Property	Description
KerberosKDC	The Kerberos Key Distribution Center (KDC) service used to authenticate the user.
KerberosRealm	The Kerberos Realm used to authenticate the user with.
SaslQop	Quality of protection for the SASL framework. The level of quality is negotiated between the client and server during authentication. Used by Kerberos authentication with TCP transport.
KerberosSPN	The service principal name (SPN) for the Kerberos Domain Controller.
KerberosKeytabFile	The Keytab file containing your pairs of Kerberos principals and encrypted keys.
KerberosServiceRealm	The Kerberos realm of the service.
KerberosServiceKDC	The Kerberos KDC of the service.
KerberosTicketCache	The full file path to an MIT Kerberos credential cache file.

24.2.3. SSL

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.
SSLClientCertSubject	The subject of the TLS/SSL client certificate.

SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

24.2.4. SSH

Property	Description
SSHAuthMode	The authentication method to be used to log on to an SFTP server.
SSHClientCert	A certificate to be used for authenticating the user.
SSHClientCertPassword	The password of the SSHClientCert certificate if it has one.
SSHClientCertSubject	The subject of the SSH client certificate.
SSHClientCertType	The type of SSHClientCert certificate.
SSHServer	The SSH server.
SSHPort	The SSH port.
SSHUser	The SSH user.
SSHPassword	The SSH password.
SSHServerFingerprint	The SSH server fingerprint.
UseSSH	Whether to tunnel the Apache Hive connection over SSH. Use SSH.

24.2.5. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.

FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

24.2.6. Proxy

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.
ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.
ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

24.2.7. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

24.2.8. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.

24.2.9. Caching

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .
CacheDriver	The database driver to be used to cache data.
CacheConnection	The connection string for the cache database. This property is always used in conjunction with CacheProvider . Setting both properties will override the value set for CacheLocation for caching data.
CacheLocation	Specifies the path to the cache when caching to a file.
CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache .
Offline	Use offline mode to get the data from the cache instead of the live source.
CacheMetadata	This property determines whether or not to cache the table metadata to a file store.

24.2.10. Miscellaneous

Property	Description
AsyncQueryTimeout	The timeout for asynchronous requests issued by the provider to download large result sets.
BatchSize	The maximum size of each batch operation to submit.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.

ConnectOnOpen	This property species whether to connect to the Apache Hive when the connection is opened.
FailoverHosts	This property allows you to specify a list of failover hosts in addition to the one configured in Server and Port .
HTTPPath	The path component of the URL endpoint when using HTTP TransportMode.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.
Pagesize	The maximum number of results to return per page from Apache Hive.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.
PseudoColumns	This property indicates whether or not to include pseudo columns as columns to the table.
QueryPassthrough	This option passes the query to the Apache Hive server as is.
Readonly	You can use this property to enforce read-only access to Apache Hive from the provider.
RTK	The runtime key used for licensing.

ServerConfigurations	A list of server configuration variables to override the server defaults.
SupportEnhancedSQL	This property enhances SQL functionality beyond what can be supported through the API directly, by enabling in-memory client-side processing.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
UseConnectionPooling	This property enables connection pooling.
UseDescTableQuery	This option specifies whether the columns will be retrieved using a DESC TABLE query or the GetColumns Thrift API.
UseShowDatabasesQuery	This option specifies whether the schemas will be retrieved using a SHOW DATABASES query or the GetSchemas Thrift API.
UseShowTablesQuery	This option specifies whether the tables will be retrieved using a SHOW TABLES query or the GetTables Thrift API.
UseSSL	Specifies whether to use SSL Encryption when connecting to Hive.
UseZookeeperDiscovery	Specifies whether to use ZooKeeper Service Discovery.
ZookeeperNamespace	The namespace configured on ZooKeeper for the Hive Server 2 znodes.

For more details, see <https://cdn.cdata.com/help/FIG/jdbc/>.

25. IBM DB2 DB2

Note: this driver does not support DB2 for z/OS. See [IBM DB2 for z/OS](#).

25.1. Connection Options

25.1.1. Authentication

Property	Description
AuthScheme	The authentication mechanism that the provider will use to authenticate with DB2.
Server	The name of the DB2 server.
Port	The port used to connect to the server hosting the DB2 database.
Database	The name of the DB2 database.
User	A database user.
Password	The user's password.
UseSSL	This field sets whether SSL is enabled.
AlternateServers	This property allows you to specify multiple servers in addition to the one configured in <code>Server</code> and <code>Port</code> . Specify both a server name and port; separate servers with a comma.

25.1.2. SSL

Property	Description
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

25.1.3. SSH

Property	Description
SSHAuthMode	The authentication method to be used to log on to an SFTP server.
SSHClientCert	A certificate to be used for authenticating the user.
SSHClientCertPassword	The password of the SSHClientCert certificate if it has one.
SSHClientCertSubject	The subject of the SSH client certificate.
SSHClientCertType	The type of SSHClientCert certificate.
SSHServer	The SSH server.
SSHPort	The SSH port.
SSHUser	The SSH user.
SSHPassword	The SSH password.
SSHServerFingerprint	The SSH server fingerprint.
UseSSH	Whether to tunnel the DB2 connection over SSH. Use SSH.

25.1.4. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.

FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

25.1.5. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

25.1.6. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.

Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.

25.1.7. Miscellaneous

Property	Description
AllowPreparedStatement	Prepare a query statement before its execution.
BatchSize	The maximum size of each batch operation to submit.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the DB2 when the connection is opened.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.

PoolWaitTime	The max seconds to wait for an available connection.
QueryPassthrough	This option passes the query to the DB2 server as is.
Readonly	You can use this property to enforce read-only access to DB2 from the provider.
RTK	The runtime key used for licensing.
SwitchMode	This property allows you to specify a switching mode to select a server from AlternateServers as the active server.
SwitchStrategy	This property allows you to specify a switching strategy to select a server from AlternateServers as the active server.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
UseConnectionPooling	This property enables connection pooling.

For more details, see <https://cdn.cdata.com/help/EDG/jdbc/>.

26. IBM DB2 FOR Z/OS

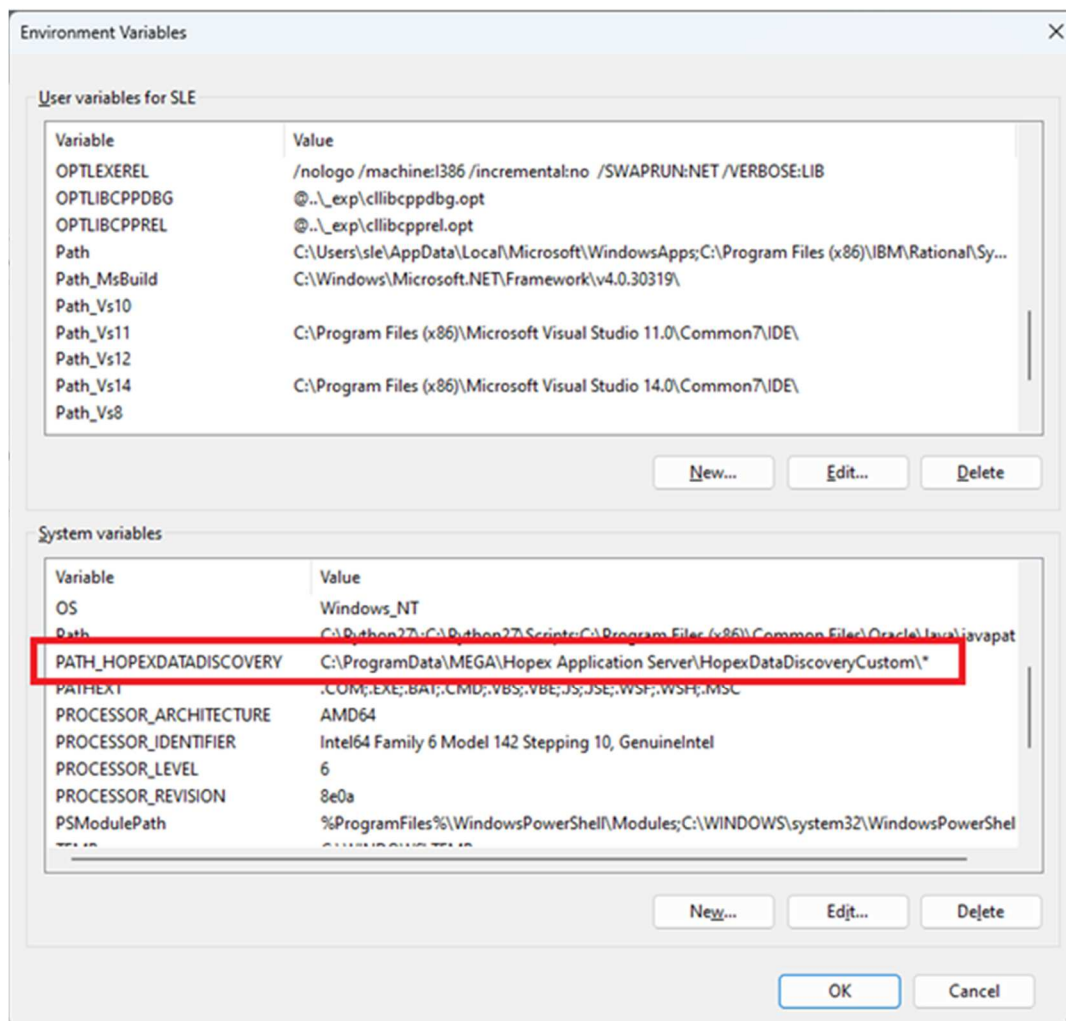
26.1. Prerequisite

1. Download the Db2 for z/OS driver: <https://www.ibm.com/support/pages/db2-jdbc-driver-versions-and-downloads>.

2. Place the Db2 for z/OS driver in a directory accessible from the HOPEX Data Discovery Module, but not in the module directory.

Example: "C:\ProgramData\MEGA\Hopex Application Server\HopexDataDiscoveryCustom"

3. Create the environment variable "PATH_HOPEXDATADISCOVERY" if it doesn't already exist and set it to the directory mentioned in step 2.



26.2. Connection Options

Property	Description
Database	A name for the data source. It's the Db2 location name that is defined during installation. All characters in the Db2 location name must be uppercase characters.
Server	The domain name or IP address of the data source
Port	The TCP/IP server port number that is assigned to the data source. This is an integer between 0 and 65535.
User	The user name for the connection
Password	The user's password for the connection.
DBNAME	For a table, or a view of tables, the name of the database that contains the table space that is named in TSNAME. For a created temporary table, an alias, or a view of a view, the value is DSNDB06.
specialRegisters	A list of special register settings for the JDBC connection. You can specify one or more special register name and value pairs. Special register name and value pairs must be delimited by commas (,).
globalSessionVariables	A list of session variable settings for the JDBC connection. You can specify one or more session variable name and value pairs. Session variable settings apply only to Db2 for z/OS Version 11 or later data servers. Session variable name and value pairs must be delimited by commas (,). The last pair must end with a semicolon (;).

queryAcceleration

Changes the value of the CURRENT QUERY ACCELERATION special register.

Possible values are:

NONE : Specifies that no query acceleration is done.

ENABLE : Specifies that queries are accelerated only if Db2 determines that it is advantageous to do so. If there is an accelerator failure while a query is running, or the accelerator returns an error, Db2 returns a negative SQLCODE to the application.

ENABLE WITH FAILBACK : Specifies that queries are accelerated only if Db2 determines that it is advantageous to do so. If the accelerator returns an error during the PREPARE or first OPEN for the query, Db2 executes the query without the accelerator. If the accelerator returns an error during a FETCH or a subsequent OPEN, Db2 returns the error to the user, and does not execute the query.

ELIGIBLE : Specifies that queries are accelerated if they are eligible for acceleration. Db2 does not use cost information to determine whether to accelerate the queries. Queries that are not eligible for acceleration are executed by Db2. If there is an accelerator failure while a query is running, or the accelerator returns an error, Db2 returns a negative SQLCODE to the application.

ALL : Specifies that queries are accelerated if they are eligible for acceleration. Db2 does not use cost information to determine whether to accelerate the queries. Queries that are not eligible for acceleration are not executed by Db2, and an SQL error is returned. If there is an accelerator failure while a query is running, or the accelerator returns an error, Db2 returns a negative SQLCODE to the application.

retrieveMessagesFromServerOnGetMessage

During a connection to a data server, set the retrieveMessagesFromServerOnGetMessage property to true if you want full message text from an SQLException.getMessage call. Default is false.

`emulateParameterMetaDataForZCalls`

This parameter enables the use of the method `PreparedStatement.getParameterMetaData()` against a DB2 for z/OS data source. Default is 0.

27. IBM INFORMIX

27.1. Informix Version Support

Compatible with IBM Informix 9.70 and newer. The driver supports industry-standard Distributed Relational Database Architecture (DRDA) connectivity only.

27.2. Connection Options

27.2.1. Authentication

Property	Description
AuthScheme	The authentication mechanism that the provider will use to authenticate with Informix.
Server	The name of the DB2 server.
Port	The port used to connect to the server hosting the database.
User	A database user.
Password	The user's password.
Database	The name of the DB2 database.
UseSSL	This field sets whether SSL is enabled.
AlternateServers	This property allows you to specify multiple servers in addition to the one configured in <code>Server</code> and <code>Port</code> . Specify both a server name and port; separate servers with a comma.

27.2.2. SSL

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.
SSLClientCertSubject	The subject of the TLS/SSL client certificate.
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

27.2.3. SSH

Property	Description
SSHAuthMode	The authentication method to be used to log on to an SFTP server.
SSHClientCert	A certificate to be used for authenticating the user.
SSHClientCertPassword	The password of the SSHClientCert certificate if it has one.
SSHClientCertSubject	The subject of the SSH client certificate.
SSHClientCertType	The type of SSHClientCert certificate.
SSHServer	The SSH server.
SSHPort	The SSH port.

SSHUser	The SSH user.
SSHPassword	The SSH password.
SSHServerFingerprint	The SSH server fingerprint.
UseSSH	Whether to tunnel the Informix connection over SSH. Use SSH.

27.2.4. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

27.2.5. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.

MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

27.2.6. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.

27.2.7. Miscellaneous

Property	Description
BatchSize	The maximum size of each batch operation to submit.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the Informix when the connection is opened.

MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.
QueryPassthrough	This option passes the query to the Informix server as is.
Readonly	You can use this property to enforce read-only access to Informix from the provider.
RTK	The runtime key used for licensing.
SessionVariables	A semicolon-separated list of SET ENVIRONMENT commands to set on the current connection.
SwitchMode	This property allows you to specify a switching mode to select a server from AlternateServers as the active server.
SwitchStrategy	This property allows you to specify a switching strategy to select a server from AlternateServers as the active server.
Timeout	A timeout for connections in seconds.
UseConnectionPooling	This property enables connection pooling.

For more details, see <https://cdn.cdata.com/help/JIG/jdbc/default.htm>.

28. IMPALA

28.1. Connection Options

28.1.1. Authentication

Property	Description
AuthScheme	The authentication scheme used. Accepted entries are NoSasl, LDAP and Kerberos.
Server	The name of the server running SQL Server.
Port	The port for the connection to the Impala Server instance.
User	The username used to authenticate with Impala.
Password	The password used to authenticate with Impala.
ProtocolVersion	The Thrift protocol version to use when connecting to the Impala server.
Database	The name of the Impala database to use by default.
TransportMode	The transport mode to use to communicate with the Impala server. Accepted entries are BINARY and HTTP.

28.1.2. Kerberos

Property	Description
KerberosKDC	The Kerberos Key Distribution Center (KDC) service used to authenticate the user.
KerberosRealm	The Kerberos Realm used to authenticate the user with.

KerberosSPN	The service principal name (SPN) for the Kerberos Domain Controller.
KerberosKeytabFile	The Keytab file containing your pairs of Kerberos principals and encrypted keys.
KerberosServiceRealm	The Kerberos realm of the service.
KerberosServiceKDC	The Kerberos KDC of the service.
KerberosTicketCache	The full file path to an MIT Kerberos credential cache file.

28.1.3. SSL

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.
SSLClientCertSubject	The subject of the TLS/SSL client certificate.
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

28.1.4. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

28.1.5. Proxy

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.
ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.

ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.
ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

28.1.6. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

28.1.7. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.

[Views](#)

Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.

28.1.8. Caching

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .
CacheDriver	The database driver used to cache data.
CacheConnection	The connection string for the cache database. This property is always used in conjunction with CacheProvider . Setting both properties will override the value set for CacheLocation for caching data.
CacheLocation	Specifies the path to the cache when caching to a file.
CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache .
Offline	Use offline mode to get the data from the cache instead of the live source.
CacheMetadata	This property determines whether or not to cache the table metadata to a file store.

28.1.9. Miscellaneous

Property	Description
BatchSize	The maximum size of each batch operation to submit.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.

ConnectOnOpen	This property specifies whether to connect to the Apache Impala when the connection is opened.
HTTPPath	The path component of the URL endpoint when using HTTP TransportMode.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.
Pagesize	The maximum number of results to return per page from Apache Impala.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.
PseudoColumns	This property indicates whether or not to include pseudo columns as columns to the table.
QueryPassthrough	This option passes the query to the Apache Impala server as is.
Readonly	You can use this property to enforce read-only access to Apache Impala from the provider.
RTK	The runtime key used for licensing.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
UseConnectionPooling	This property enables connection pooling.
UserDefinedViews	A filepath pointing to the JSON configuration file containing your custom views.

UseSSL

Specifies whether to use SSL Encryption when connecting to Impala.

For more details, see <https://cdn.cdata.com/help/GIH/jdbc/default.htm>.

29. JIRA

29.1. Connection Options

29.1.1. Authentication

Property	Description
AuthScheme	The type of authentication to use when connecting to Jira Service Desk.
URL	The URL to your JIRA Service Desk endpoint.
User	The Jira Service Desk user account used to authenticate.
Password	The password used to authenticate the user.
APIToken	APIToken of the currently authenticated user.

29.1.2. SSO

Property	Description
SSOLoginURL	The identity provider's login URL.
SSOProperties	Additional properties required to connect to the identity provider in a semicolon-separated list.
SSOExchangeUrl	The url used for consuming the SAML response and exchanging it with Jira Service Desk specific credentials.
SSOAppName	App Name used with SSO for IdPs that require it.
SSOAppPassword	App Password used with SSO for IdPs that require it.

29.1.3. OAuth

Property	Description
InitiateOAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.
OAuthVersion	The version of OAuth being used.
OAuthClientId	The client ID assigned when you register your application with an OAuth authorization server.
OAuthClientSecret	The client secret assigned when you register your application with an OAuth authorization server.
OAuthVerifier	The verifier code returned from the OAuth authorization URL.
AuthToken	The authentication token used to request and obtain the OAuth Access Token.
AuthKey	The authentication secret used to request and obtain the OAuth Access Token.
CertificateStoreType	The type of certificate store used with Jira Service Desk Private Application authentication.
CertificateStore	The certificate store used for JIRA Service Desk authentication.
CertificateStorePassword	The password of the certificate store used with Jira Service Desk authentication.
CertificateSubject	The subject of the certificate used with Jira Service Desk Private Application authentication.

29.1.4. SSL

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.
SSLClientCertSubject	The subject of the TLS/SSL client certificate.
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

29.1.5. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

29.1.6. Proxy

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.
ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.
ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

29.1.7. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.

Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

29.1.8. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.

29.1.9. Caching

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .

CacheProvider	The name of the provider to be used to cache data.
CacheConnection	The connection string for the cache database. This property is always used in conjunction with CacheProvider . Setting both properties will override the value set for CacheLocation for caching data.
CacheLocation	Specifies the path to the cache when caching to a file.
CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache .
Offline	Use offline mode to get the data from the cache instead of the live source.
CacheMetadata	This property determines whether or not to cache the table metadata to a file store.

29.1.10. Miscellaneous

Property	Description
DefaultDomain	This property is used for the Oracle Database Gateway for ODBC.
EnableForeignKeyDetection	Whether to detect the foreign keys in ODBC.
IncludeCustomFields	A boolean indicating if you would like to include custom fields in the column listing.
IncludeDualTable	Set this property to mock the Oracle DUAL table for better compatibility with Oracle database.
LimitKeySize	The maximum length of a primary key column.
MapBigintToVarchar	This property controls whether or not the bigint type maps to SQL_VARCHAR instead

	of SQL_BIGINT. This property is false by default.
MapToInt	This property controls whether or not the long type maps to SQL_INTEGER instead of SQL_BIGINT. This property is false by default.
MapToLongVarchar	This property controls whether or not a column is returned as SQL_LONGVARCHAR.
MapToWVarchar	This property controls whether or not string types map to SQL_WVARCHAR instead of SQL_VARCHAR. This property is set by default.
MaximumColumnSize	The maximum column size.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.
PseudoColumns	This property indicates whether or not to include pseudo columns as columns to the table.
Readonly	You can use this property to enforce read-only access to Jira Service Desk from the provider.
RequestLanguage	Use the requestLanguage to have column names translated in a specific language.
RTK	The runtime key used for licensing.
ServiceDeskID	Service Desk ID of the currently authenticated user.
SupportEnhancedSQL	This property enhances SQL functionality beyond what can be supported through

	the API directly, by enabling in-memory client-side processing.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
UpperCaseIdentifiers	This property reports all identifiers in uppercase. This is the default for Oracle databases and thus allows better integration with Oracle tools such as the Oracle Database Gateway.

For more details, see <https://cdn.cdata.com/help/GKG/odbc/>.

30. JSON {JSON}

30.1. Connection Options

30.1.1. Authentication

Property	Description
AuthScheme	The type of authentication to use when connecting to remote services.
AccessKey	Your account access key. This value is accessible from your security credentials page.
SecretKey	Your account secret key. This value is accessible from your security credentials page.
ApiKey	The API Key used to identify the user to IBM Cloud.
User	The JSON user account used to authenticate.
Password	The password used to authenticate the user.
SharePointEdition	The edition of SharePoint being used. Set either SharePointOnline or SharePointOnPremise.

30.1.2. Connection

Property	Description
URI	The Uniform Resource Identifier (URI) for the JSON resource location.
JSONPath	The JSONPath of an array element that defines the separation of rows.
DataModel	Specifies the data model to use when parsing JSON documents and generating the database metadata.

JSONFormat	Specifies the format of the JSON document.
Region	The hosting region for your S3-like Web Services.
ProjectId	The id of the project where your Google Cloud Storage instance resides.
OracleNamespace	The Oracle Cloud Object Storage namespace to use.
StorageBaseURL	The URL of a cloud storage service provider.
SimpleUploadLimit	This setting specifies the threshold, in bytes, above which the provider will choose to perform a multipart upload rather than uploading everything in one request.
UseVirtualHosting	If true (default), buckets will be referenced in the request using the hosted-style request: <code>http://yourbucket.s3.amazonaws.com/yourobject</code> . If set to false, the bean will use the path-style request: <code>http://s3.amazonaws.com/yourbucket/yourobject</code> . Note that this property will be set to false, in case of an S3 based custom service when the CustomURL is specified.

30.1.3. AWS Authentication

Property	Description
AWSAccessKey	Your AWS account access key. This value is accessible from your AWS security credentials page.
AWSSecretKey	Your AWS account secret key. This value is accessible from your AWS security credentials page.
AWSRoleARN	The Amazon Resource Name of the role to use when authenticating.
AWSPrincipalArn	The ARN of the SAML Identity provider in your AWS account.

AWSRegion	The hosting region for your Amazon Web Services.
AWSSessionToken	Your AWS session token.
MFASerialNumber	The serial number of the MFA device if one is being used.
MFAToken	The temporary token available from your MFA device.

30.1.4. Azure Authentication

Property	Description
AzureStorageAccount	The name of your Azure storage account.
AzureAccessKey	The storage key associated with your JSON account.
AzureSharedAccessSignature	A shared access key signature that may be used for authentication.
AzureTenant	The Microsoft Online tenant being used to access data. If not specified, your default tenant will be used.
AzureEnvironment	The Azure Environment to use when establishing a connection.

30.1.5. SSO

Property	Description
SSOLoginURL	The identity provider's login URL.
SSOProperties	Additional properties required to connect to the identity provider in a semicolon-separated list.

30.1.6. OAuth

Property	Description
InitiateOAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.
OAuthVersion	The version of OAuth being used.
OAuthClientId	The client ID assigned when you register your application with an OAuth authorization server.
OAuthClientSecret	The client secret assigned when you register your application with an OAuth authorization server.
OAuthAccessToken	The access token for connecting using OAuth.
OAuthAccessTokenSecret	The OAuth access token secret for connecting using OAuth.
OAuthSettingsLocation	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH. Alternatively, this can be held in memory by specifying a value starting with memory://.
CallbackURL	The OAuth callback URL to return to when authenticating. This value must match the callback URL you specify in your app settings.
OAuthGrantType	The grant type for the OAuth flow.
OAuthPasswordGrantMode	How to pass Client ID and Secret with OAuthGrantType is set to Password.
OAuthIncludeCallbackURL	Whether to include the callback URL in an access token request.

OAuthAuthorizationURL	The authorization URL for the OAuth service.
OAuthAccessTokenURL	The URL to retrieve the OAuth access token from.
OAuthRefreshTokenURL	The URL to refresh the OAuth token from.
OAuthRequestTokenURL	The URL the service provides to retrieve request tokens from. This is required in OAuth 1.0.
OAuthVerifier	The verifier code returned from the OAuth authorization URL.
AuthToken	The authentication token used to request and obtain the OAuth Access Token.
AuthKey	The authentication secret used to request and obtain the OAuth Access Token.
OAuthParams	A comma-separated list of other parameters to submit in the request for the OAuth access token in the format paramname=value.
OAuthRefreshToken	The OAuth refresh token for the corresponding OAuth access token.
OAuthExpiresIn	The lifetime in seconds of the OAuth AccessToken.
OAuthTokenTimestamp	The Unix epoch timestamp in milliseconds when the current Access Token was created.

30.1.7. JWT OAuth

Property	Description
OAuthJWTCert	The JWT Certificate store.

OAuthJWTCertType	The type of key store containing the JWT Certificate.
OAuthJWTCertPassword	The password for the OAuth JWT certificate.
OAuthJWTCertSubject	The subject of the OAuth JWT certificate.
OAuthJWTIssuer	The issuer of the Java Web Token.
OAuthJWTSubject	The user subject for which the application is requesting delegated access.

30.1.8. Kerberos

Property	Description
KerberosKDC	The Kerberos Key Distribution Center (KDC) service used to authenticate the user.
KerberosRealm	The Kerberos Realm used to authenticate the user with.
KerberosSPN	The service principal name (SPN) for the Kerberos Domain Controller.
KerberosKeytabFile	The Keytab file containing your pairs of Kerberos principals and encrypted keys.
KerberosServiceRealm	The Kerberos realm of the service.
KerberosServiceKDC	The Kerberos KDC of the service.
KerberosTicketCache	The full file path to an MIT Kerberos credential cache file.

30.1.9. SSL

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.
SSLClientCertSubject	The subject of the TLS/SSL client certificate.
SSLMode	The authentication mechanism to be used when connecting to the FTP or FTPS server.
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

30.1.10. SSH

Property	Description
SSHAuthMode	The authentication method to be used to log on to an SFTP server.
SSHClientCert	A certificate to be used for authenticating the user.
SSHClientCertPassword	The password of the SSHClientCert certificate if it has one.
SSHClientCertType	The type of SSHClientCert certificate.

30.1.11. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

30.1.12. Proxy

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.
ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.

ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

30.1.13. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

30.1.14. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.

Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.
FlattenArrays	By default, nested arrays are returned as strings of JSON. The FlattenArrays property can be used to flatten the elements of nested arrays into columns of their own. Set FlattenArrays to the number of elements you want to return from nested arrays.
FlattenObjects	Set FlattenObjects to true to flatten object properties into columns of their own. Otherwise, objects nested in arrays are returned as strings of JSON.
QualifyColumns	Controls whether the provider will use relative column names.

30.1.15. Caching

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .
CacheDriver	The database driver to be used to cache data.
CacheConnection	The connection string for the cache database. This property is always used in conjunction with CacheProvider . Setting both properties will override the value set for CacheLocation for caching data.
CacheLocation	Specifies the path to the cache when caching to a file.
CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache .
Offline	Use offline mode to get the data from the cache instead of the live source.

CacheMetadata

This property determines whether or not to cache the table metadata to a file store.

30.1.16. Miscellaneous

Property	Description
BackwardsCompatibilityMode	Set BackwardsCompatibilityMode to true to use the JSON functionality and features available in the 2017 version.
BatchSize	The maximum size of each batch operation to submit.
Charset	Specifies the session character set for encoding and decoding character data transferred to and from the JSON file. The default value is UTF-8.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the JSON when the connection is opened.
Culture	This setting can be used to specify culture settings that determine how the provider interprets certain data types that are passed into the provider. For example, setting Culture='de-DE' will output German formats even on an American machine.
CustomHeaders	Other headers as determined by the user (optional).
CustomUrlParams	The custom query string to be included in the request.

ExcludeFiles	Comma-separated list of file extensions to exclude from the set of the files modeled as tables.
FlattenRowLimit	The maximum number of rows that can result from a single flattened element.
GenerateSchemaFiles	Indicates the user preference as to when schemas should be generated and saved.
IncludeDropboxTeamResources	Indicates if you want to include Dropbox team files and folders.
IncludeFiles	Comma-separated list of file extensions to include into the set of the files modeled as tables.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
MetadataDiscoveryURI	Used when aggregating multiple files into one table, this property specifies a specific file to read to determined the aggregated table schema.
Other	These hidden properties are used only in specific use cases.
Pagesize	The maximum number of results to return per page from JSON.
PathSeparator	Determines the character which will be used to replace the file separator.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.

PoolWaitTime	The max seconds to wait for an available connection.
PseudoColumns	This property indicates whether or not to include pseudo columns as columns to the table.
Readonly	You can use this property to enforce read-only access to JSON from the provider.
RowScanDepth	The number of rows to scan when dynamically determining columns for the table.
RTK	The runtime key used for licensing.
SupportEnhancedSQL	This property enhances SQL functionality beyond what can be supported through the API directly, by enabling in-memory client-side processing.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
TypeDetectionScheme	Determines how to determine the data types of columns.
URISeparator	A delimiter used to separate different values in the URI property.
UseConnectionPooling	This property enables connection pooling.

For more details, see <https://cdn.cdata.com/help/DJG/jdbc/>.

31. KAFKA

31.1. Connection Options

31.1.1. Authentication

Property	Description
AuthScheme	The scheme used for authentication. Accepted entries are Auto,Plain,Scram,Kerberos.
User	The user who is authenticating to Apache Kafka.
Password	The password used to authenticate to Apache Kafka.
BootstrapServers	The address of the Apache Kafka BootstrapServers to which you are connecting to.
Topic	The name of the topic to produce into.
UseSSL	This field sets whether SSL is enabled.

31.1.2. Connection

Property	Description
ConsumerGroupld	Specifies which group the consumers created by the driver should belong to.
AutoCommit	Specifies if the Apache Kafka consumer should autocommit after each poll.

31.1.3. Kerberos

Property	Description
KerberosKeytabFile	The Keytab file containing your pairs of Kerberos principals and encrypted keys.
KerberosSPN	The service principal name (SPN) for the Kerberos Domain Controller.
KerberosServiceName	The Kerberos service name you will authenticate.
UseKerberosTicketCache	In case that you don't want to use a keytab file but instead a ticket cache with the logged in user.

31.1.4. SSL

Property	Description
SSLCert	Path to the Keystore location or to the client's certificate location.
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLCertPassword	The password of the Keystore or of the client's private key.
SSLClientCertPassword	The password for the TLS/SSL client certificate.

SSLClientCertSubject	The subject of the TLS/SSL client certificate.
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.
SSLIdentificationAlgorithm	The endpoint identification algorithm used by the Apache Kafka data provider client app to validate server host name.
SSLKey	Set either the password to decrypt the client private key or the path of the client's private key depending on the data provider edition.
TrustStorePassword	The password of the Truststore, 'ssl.truststore.password'.
TrustStorePath	Path to the Truststore or Certification Authorities (CA) location.

31.1.5. SchemaRegistry

Property	Description
RegistryVersion	Version of the schema read from RegistryUrl for the specified topic.
RegistryUrl	The server for the schema registry. When this property is specified, the driver will read Apache Avro schema from the server.
RegistryUser	Username to authorize with the server specified in RegistryUrl .
RegistryPassword	Password to authorize with the server specified in RegistryUrl .

RegistryType	Type of the schema specified for the a specific topic.
RegistryService	The Schema Registry service used for working with topic schemas.

31.1.6. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

31.1.7. Proxy

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.

ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.
ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

31.1.8. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

31.1.9. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.

31.1.10. Caching

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .
CacheDriver	The database driver to be used to cache data.
CacheConnection	The connection string for the cache database. This property is always used in conjunction with CacheProvider . Setting both properties will override the value set for CacheLocation for caching data.
CacheLocation	Specifies the path to the cache when caching to a file.

CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache .
Offline	Use offline mode to get the data from the cache instead of the live source.
CacheMetadata	This property determines whether or not to cache the table metadata to a file store.

31.1.11. Miscellaneous

Property	Description
AggregateMessages	Specifies whether or not to return the message as a whole string.
BatchSize	The maximum size of each batch operation to submit.
CompressionType	Data compression type. Batches of data will be compressed together.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the Apache Kafka when the connection is opened.
EnableIdempotence	If set to true, the Apache Kafka will ensure messages are delivered in the correct, and without duplicates.
FlattenArrays	By default, nested arrays won't show up if TypeDetectionScheme is set to SchemaRegistry. The FlattenArrays property can be used to flatten the elements of nested arrays into columns of their own. Set

	FlattenArrays to the number of elements you want to return from nested arrays.
GenerateSchemaFiles	Indicates the user preference as to when schemas should be generated and saved.
MaximumBatchSize	Specifies maximum batch size to gather before sending a request.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
MessageKeyColumn	If specified, the message key sent to Apache Kafka will be read from this column.
MessageKeyType	If MessageKeyColumn is specified, this property must be set to the expected type for the pertinent column.
OffsetResetStrategy	Specifies an offset for the consumer group.
Other	These hidden properties are used only in specific use cases.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.

ProduceMeta	Specifies whether or not to send a meta message while producing the outgoing message.
PseudoColumns	This property indicates whether or not to include pseudo columns as columns to the table.
ReadDuration	The duration which additional messages are allowed.
Readonly	You can use this property to enforce read-only access to Apache Kafka from the provider.
RowScanDepth	The maximum number of messages to scan for the columns available in the topic.
RTK	The runtime key used for licensing.
SerializationFormat	Specifies how to serialize/deserialize the incoming or outgoing message.
SupportEnhancedSQL	This property enhances SQL functionality beyond what can be supported through the API directly, by enabling in-memory client-side processing.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
TypeDetectionScheme	Comma-separated list of options specifying how the provider will scan the data to determine the fields and datatypes for the bucket.
UseConnectionPooling	This property enables connection pooling.

For more details, see <https://cdn.cdata.com/help/SKG/jdbc/>.

32. MARIADB

32.1. MariaDB Version Support

The driver enables connectivity to MariaDB Server through 5.0 to 10.5+.

32.2. Connection Options

32.2.1. Authentication

Property	Description
Server	The host name or IP address of the server.
Port	The port of the MariaDB server.
User	The MariaDB user account used to authenticate.
Password	The password used to authenticate the user.
Database	The name of the MariaDB database.
UseSSL	This field sets whether SSL is enabled.

32.2.2. SSL

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.

SSLClientCertSubject	The subject of the TLS/SSL client certificate.
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

32.2.3. SSH

Property	Description
SSHAuthMode	The authentication method to be used to log on to an SFTP server.
SSHClientCert	A certificate to be used for authenticating the user.
SSHClientCertPassword	The password of the SSHClientCert certificate if it has one.
SSHClientCertSubject	The subject of the SSH client certificate.
SSHClientCertType	The type of SSHClientCert certificate.
SSHServer	The SSH server.
SSHPort	The SSH port.
SSHUser	The SSH user.
SSHPassword	The SSH password.
SSHServerFingerprint	The SSH server fingerprint.
UseSSH	Whether to tunnel the MariaDB connection over SSH. Use SSH.

32.2.4. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

32.2.5. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

32.2.6. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.

32.2.7. Miscellaneous

Property	Description
AllowUserVariables	When set to True, user variables (prefixed by an @) can be used in SQL queries.
BatchSize	The maximum size of each batch operation to submit.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the MariaDB when the connection is opened.

MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.
QueryPassthrough	This option passes the query to the MariaDB server as is.
Readonly	You can use this property to enforce read-only access to MariaDB from the provider.
RTK	The runtime key used for licensing.
Timeout	The value in seconds until the connection timeout error is thrown.
UseConnectionPooling	This property enables connection pooling.
ZeroDatesToNull	Whether or not to return Date and DateTime values consisting of all zeros as NULL.

For more details, see <https://cdn.cdata.com/help/SRG/jdbc/>.

33. MONGODB

33.1. MongoDB Version Support

The driver models MongoDB instances as relational databases and supports MongoDB versions 2.6 through 5.0.

33.2. Connection Options

33.2.1. Authentication

Property	Description
AuthScheme	The authentication mechanism that MongoDB will use to authenticate the connection.
Server	The host name or IP address of the server hosting the MongoDB database.
Port	The port for the MongoDB database.
User	The MongoDB user account used to authenticate.
Password	The password used to authenticate the user.
Database	The name of the MongoDB database.
UseSSL	This field sets whether SSL is enabled.
AuthDatabase	The name of the MongoDB database for authentication.
ReplicaSet	This property allows you to specify multiple servers in addition to the one configured in Server and Port . Specify both a server name and port; separate servers with a comma.
DNSServer	Specify the DNS server when resolving MongoDB seed list.

33.2.2. Kerberos

Property	Description
KerberosKDC	The Kerberos Key Distribution Center (KDC) service used to authenticate the user.
KerberosRealm	The Kerberos Realm used to authenticate the user with.
KerberosSPN	The service principal name (SPN) for the Kerberos Domain Controller.
KerberosKeytabFile	The Keytab file containing your pairs of Kerberos principals and encrypted keys.
KerberosServiceRealm	The Kerberos realm of the service.
KerberosServiceKDC	The Kerberos KDC of the service.
KerberosTicketCache	The full file path to an MIT Kerberos credential cache file.

33.2.3. SSL

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.
SSLClientCertSubject	The subject of the TLS/SSL client certificate.
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

33.2.4. SSH

Property	Description
SSHAuthMode	The authentication method to be used to log on to an SFTP server.
SSHClientCert	A certificate to be used for authenticating the user.
SSHClientCertPassword	The password of the SSHClientCert certificate if it has one.
SSHClientCertSubject	The subject of the SSH client certificate.
SSHClientCertType	The type of SSHClientCert certificate.
SSHServer	The SSH server.
SSHPort	The SSH port.
SSHUser	The SSH user.
SSHPassword	The SSH password.
SSHServerFingerprint	The SSH server fingerprint.
UseSSH	Whether to tunnel the MongoDB connection over SSH. Use SSH.

33.2.5. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.

FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

33.2.6. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

33.2.7. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.

33.2.8. Caching

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .
CacheDriver	The database driver to be used to cache data.
CacheConnection	The connection string for the cache database. This property is always used in conjunction with CacheProvider . Setting both properties will override the value set for CacheLocation for caching data.
CacheLocation	Specifies the path to the cache when caching to a file.
CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache .
Offline	Use offline mode to get the data from the cache instead of the live source.
CacheMetadata	This property determines whether or not to cache the table metadata to a file store.

33.2.9. Miscellaneous

Property	Description
BatchSize	The maximum size of each batch operation to submit.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the MongoDB when the connection is opened.

DataModel	By default, the provider will not automatically discover the metadata for a child table as its own distinct table. To enable this functionality, set DataModel to Relational .
FlattenArrays	By default, nested arrays are returned as strings of JSON. The FlattenArrays property can be used to flatten the elements of nested arrays into columns of their own. Set FlattenArrays to the number of elements you want to return from nested arrays.
FlattenObjects	Set FlattenObjects to true to flatten object properties into columns of their own. Otherwise, objects nested in arrays are returned as strings of JSON.
GenerateSchemaFiles	Indicates the user preference as to when schemas should be generated and saved.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
NoCursorTimeout	The server normally times out idle cursors after an inactivity period (10 minutes) to prevent excess memory use. Set this option to prevent that.
Other	These hidden properties are used only in specific use cases.
Pagesize	The maximum number of results to return per page from MongoDB.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.
PseudoColumns	This property indicates whether or not to include pseudo columns as columns to the table.
QueryPassthrough	This option passes the query to MongoDB as-is.
Readonly	You can use this property to enforce read-only access to MongoDB from the provider.

ReadPreference	Set this to a strategy for reading from a replica set. Accepted values are primary, primaryPreferred, secondary, secondaryPreferred, and nearest.
RowScanDepth	The maximum number of rows to scan to look for the columns available in a table.
RTK	The runtime key used for licensing.
SlaveOK	This property sets whether the provider is allowed to read from secondary (slave) servers.
SupportEnhancedSQL	This property enhances SQL functionality beyond what can be supported through the API directly, by enabling in-memory client-side processing.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
TypeDetectionScheme	Comma-separated options for how the provider will scan the data to determine the fields and datatypes in each document collection.
UseConnectionPooling	This property enables connection pooling.
UseFindAPI	Execute MongoDB queries using <code>db.collection.find()</code> .
WriteConcern	Requests acknowledgment that the write operation has propagated to the specified number of mongod instances.
WriteConcernJournaled	It requires acknowledgment that the mongod instances, as specified in the WriteConcern , have written to the on-disk journal.
WriteConcernTimeout	This option specifies a time limit, in milliseconds, for the write concern.
WriteScheme	Sets whether the object type for inserted or updated objects is determined from the existing column metadata or the input value type.

For more details, see <https://cdn.cdata.com/help/DGG/jdbc/>.

34. **MYSQL**

34.1. MySQL Version Support

The driver enables connectivity to MySQL Server through 5.0 to 8.0.

34.2. Connection Options

34.2.1. Authentication

Property	Description
AuthScheme	The scheme used for authentication. Accepted entries are Password, AzureAD, AzurePassword, AzureMSI.
Server	The host name or IP address of the server. Supports cluster servers, for example '192.168.0.1,192.168.0.2'.
Port	The port of the MySQL server. Supports cluster servers, for example: '3306, 3307', the number of the port should match with Servers.
User	The MySQL user account used to authenticate.
Password	The password used to authenticate the user.
Database	The name of the MySQL database.
Domain	The name of the domain for a Windows (NTLM) security login.
NTLMVersion	The NTLM version.
IntegratedUser	The user that is authenticating to the Windows.
UseSSL	This field sets whether SSL is enabled.

34.2.2. Azure Authentication

Property	Description
AzureTenant	The Microsoft Online tenant being used to access data. If not specified, your default tenant will be used.

34.2.3. OAuth

Property	Description
InitiateOAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.
OAuthClientId	The client ID assigned when you register your application with an OAuth authorization server.
OAuthClientSecret	The client secret assigned when you register your application with an OAuth authorization server.
OAuthAccessToken	The access token for connecting using OAuth.
OAuthSettingsLocation	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH. Alternatively, this can be held in memory by specifying a value starting with memory://.
OAuthVerifier	The verifier code returned from the OAuth authorization URL.
OAuthExpiresIn	The lifetime in seconds of the OAuth AccessToken.
OAuthTokenTimestamp	The Unix epoch timestamp in milliseconds when the current Access Token was created.

34.2.4. SSL

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.
SSLClientCertSubject	The subject of the TLS/SSL client certificate.
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

34.2.5. SSH

Property	Description
SSHAuthMode	The authentication method to be used to log on to an SFTP server.
SSHClientCert	A certificate to be used for authenticating the user.
SSHClientCertPassword	The password of the SSHClientCert certificate if it has one.
SSHClientCertSubject	The subject of the SSH client certificate.
SSHClientCertType	The type of SSHClientCert certificate.
SSHServer	The SSH server.
SSHPort	The SSH port.
SSHUser	The SSH user.
SSHPassword	The SSH password.

SSHServerFingerprint	The SSH server fingerprint.
UseSSH	Whether to tunnel the MySQL connection over SSH. Use SSH.

34.2.6. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

34.2.7. Proxy

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.
ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.

ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

34.2.8. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

34.2.9. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.

34.2.10. Miscellaneous

Property	Description
AllowUserVariables	When set to True, user variables (prefixed by an @) can be used in SQL queries.
BatchSize	The maximum size of each batch operation to submit.
Characterset	The default client character set used by the provider. For example, 'utf8'.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the MySQL when the connection is opened.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.
QueryPassthrough	This option passes the query to the MySQL server as is.
Readonly	You can use this property to enforce read-only access to MySQL from the provider.
RTK	The runtime key used for licensing.
ServerTimeZone	Specify a specific server time zone id of current platform(.Net or Java) by user.

Timeout	The value in seconds until the connection timeout error is thrown.
UseConnectionPooling	This property enables connection pooling.
ZeroDatesToNull	Whether or not to return Date and DateTime values consisting of all zeros as NULL.

For more details, see <https://cdn.cdata.com/help/DMG/jdbc/>.

35. SAS DATA SETS

35.1. Connection Options

35.1.1. Authentication

Property	Description
AuthScheme	The type of authentication to use when connecting to remote services.
AccessKey	Your account access key. This value is accessible from your security credentials page.
SecretKey	Your account secret key. This value is accessible from your security credentials page.
ApiKey	The API Key used to identify the user to IBM Cloud.
User	The user account used to authenticate.
Password	The password used to authenticate the user.
SharePointEdition	The edition of SharePoint being used. Set either SharePointOnline or SharePointOnPremise.

35.1.2. Connection

Property	Description
ConnectionType	The type of connection to use.
URI	The Uniform Resource Identifier (URI) for the SAS resource location.

Region	The hosting region for your S3-like Web Services.
ProjectId	The Id of the project where your Google Cloud Storage instance resides.
OracleNamespace	The Oracle Cloud Object Storage namespace to use.
StorageBaseURL	The URL of a cloud storage service provider.
UseVirtualHosting	If true (default), buckets will be referenced in the request using the hosted-style request: <code>http://yourbucket.s3.amazonaws.com/yourobject</code> . If set to false, the bean will use the path-style request: <code>http://s3.amazonaws.com/yourbucket/yourobject</code> . Note that this property will be set to false, in case of an S3 based custom service when the CustomURL is specified.

35.1.3. AWS Authentication

Property	Description
AWSAccessKey	Your AWS account access key. This value is accessible from your AWS security credentials page.
AWSSecretKey	Your AWS account secret key. This value is accessible from your AWS security credentials page.
AWSRoleARN	The Amazon Resource Name of the role to use when authenticating.
AWSPrincipalARN	The ARN of the SAML Identity provider in your AWS account.
AWSRegion	The hosting region for your Amazon Web Services.
AWSCredentialsFile	The path to the AWS Credentials File to be used for authentication.

AWSCredentialsFileProfile	The name of the profile to be used from the supplied AWSCredentialsFile.
AWSSessionToken	Your AWS session token.
MFASerialNumber	The serial number of the MFA device if one is being used.
MFAToken	The temporary token available from your MFA device.
ServerSideEncryption	When activated, file uploads into Amazon S3 buckets will be server-side encrypted.

35.1.4. Azure Authentication

Property	Description
AzureStorageAccount	The name of your Azure storage account.
AzureAccessKey	The storage key associated with your SAS Data Sets account.
AzureSharedAccessSignature	A shared access key signature that may be used for authentication.
AzureTenant	The Microsoft Online tenant being used to access data. If not specified, your default tenant will be used.
AzureEnvironment	The Azure Environment to use when establishing a connection.

35.1.5. SSO

Property	Description
SSOLoginURL	The identity provider's login URL.
SSOProperties	Additional properties required to connect to the identity provider in a semicolon-separated list.

35.1.6. OAuth

Property	Description
InitiateOAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.
OAuthClientId	The client Id assigned when you register your application with an OAuth authorization server.
OAuthClientSecret	The client secret assigned when you register your application with an OAuth authorization server.
OAuthAccessToken	The access token for connecting using OAuth.
OAuthAccessTokenSecret	The OAuth access token secret for connecting using OAuth.
OAuthSettingsLocation	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH. Alternatively, this can be held in memory by specifying a value starting with memory://.
CallbackURL	The OAuth callback URL to return to when authenticating. This value must match the callback URL you specify in your app settings.
Scope	Specify scope to obtain the initial access and refresh token.
OAuthVerifier	The verifier code returned from the OAuth authorization URL.
OAuthRefreshToken	The OAuth refresh token for the corresponding OAuth access token.
OAuthExpiresIn	The lifetime in seconds of the OAuth AccessToken.
OAuthTokenTimestamp	The Unix epoch timestamp in milliseconds when the current Access Token was created.

35.1.7. JWT OAuth

Property	Description
<u>OAuthJWTCert</u>	The JWT Certificate store.
<u>OAuthJWTCertType</u>	The type of key store containing the JWT Certificate.
<u>OAuthJWTCertPassword</u>	The password for the OAuth JWT certificate.
<u>OAuthJWTCertSubject</u>	The subject of the OAuth JWT certificate.
<u>OAuthJWTIssuer</u>	The issuer of the Java Web Token.
<u>OAuthJWTSubject</u>	The user subject for which the application is requesting delegated access.

35.1.8. Kerberos

Property	Description
<u>KerberosKDC</u>	The Kerberos Key Distribution Center (KDC) service used to authenticate the user.
<u>KerberosRealm</u>	The Kerberos Realm used to authenticate the user.
<u>KerberosSPN</u>	The service principal name (SPN) for the Kerberos Domain Controller.
<u>KerberosKeytabFile</u>	The Keytab file containing your pairs of Kerberos principals and encrypted keys.
<u>KerberosServiceRealm</u>	The Kerberos realm of the service.
<u>KerberosServiceKDC</u>	The Kerberos KDC of the service.
<u>KerberosTicketCache</u>	The full file path to an MIT Kerberos credential cache file.

35.1.9. SSL

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.
SSLClientCertSubject	The subject of the TLS/SSL client certificate.
SSLMode	The authentication mechanism to be used when connecting to the FTP or FTPS server.
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

35.1.10. SSH

Property	Description
SSHAuthMode	The authentication method to be used to log on to an SFTP server.
SSHClientCert	A private key to be used for authenticating the user.
SSHClientCertPassword	The password of the SSHClientCert key if it has one.
SSHClientCertSubject	The subject of the SSH client certificate.
SSHClientCertType	The type of SSHClientCert private key.
SSHUser	The SSH user.
SSHPassword	The SSH password.

35.1.11. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

35.1.12. Proxy

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.
ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.
ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

35.1.13. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

35.1.14. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.

35.1.15. Caching

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .
CacheDriver	The database driver used to cache data.
CacheConnection	The connection string for the cache database. This property is always used in conjunction with CacheProvider . Setting both properties will override the value set for CacheLocation for caching data.
CacheLocation	Specifies the path to the cache when caching to a file.
CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache.
Offline	Use offline mode to get the data from the cache instead of the live source.
CacheMetadata	This property determines whether or not to cache the table metadata to a file store.

35.1.16. Miscellaneous

Property	Description
BatchSize	The maximum size of each batch operation to submit.
Charset	Specifies the session character set for encoding and decoding character data transferred to and from the SAS Data Sets file. The default value is UTF-8.

[ClientCulture](#)

This property can be used to specify the format of data (e.g., currency values) that is accepted by the client application. This property can be used when the client application does not support the machine's culture settings. For example, Microsoft Access requires 'en-US'.

[ConnectionLifeTime](#)

The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.

[ConnectOnOpen](#)

This property specifies whether to connect to the SAS Data Sets when the connection is opened.

[Culture](#)

This setting can be used to specify culture settings that determine how the provider interprets certain data types that are passed into the provider. For example, setting Culture='de-DE' will output German formats even on an American machine.

[DeleteDownloadedFiles](#)

When set to true, the provider will delete parsed .sas7bdat files downloaded from cloud sources.

[DirectoryRetrievalDepth](#)

Limit the subfolders recursively scanned when IncludeSubdirectories is enabled.

[ExcludeFiles](#)

Comma-separated list of file extensions to exclude from the set of the files modeled as tables.

[IncludeDropboxTeamResources](#)

Indicates if you want to include Dropbox team files and folders.

[IncludeFiles](#)

Comma-separated list of file extensions to include into the set of the files modeled as tables.

[IncludeSubdirectories](#)

Whether to read files from nested folders. In the case of a name collision, table names are prefixed by the underscore-separated folder names.

[MaxRows](#)

Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.

[Other](#)

These hidden properties are used only in specific use cases.

[PoolIdleTimeout](#)

The allowed idle time for a connection before it is closed.

[PoolMaxSize](#)

The maximum connections in the pool.

[PoolMinSize](#)

The minimum number of connections in the pool.

[PoolWaitTime](#)

The max seconds to wait for an available connection.

[PseudoColumns](#)

This property indicates whether or not to include pseudo columns as columns to the table.

[Readonly](#)

You can use this property to enforce read-only access to SAS Data Sets from the provider.

[RTK](#)

The runtime key used for licensing.

[TemporaryLocalFolder](#)

The path, or URI, to the folder that is used to temporarily download SAS file(s).

[Timeout](#)

The value in seconds until the timeout error is thrown, canceling the operation.

[UseConnectionPooling](#)

This property enables connection pooling.

[UserDefinedViews](#)

A filepath pointing to the JSON configuration file containing your custom views.

36. NEO4J

36.1. Neo4j Version Support

The driver leverages the Neo4j API. The driver supports Neo4j versions 4.2 and above.

36.2. Connection Options

36.2.1. Authentication

Property	Description
User	The Neo4j user account used to authenticate.
Password	The password used to authenticate the user.

36.2.2. Connection

Property	Description
Server	The server hosting the Neo4j instance.
Port	The port on which the Neo4j service is running.

36.2.3. SSL

Property	Description
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

36.2.4. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

36.2.5. Proxy

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.
ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.
ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

36.2.6. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

36.2.7. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.

36.2.8. Caching

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .
CacheDriver	The database driver used to cache data.
CacheConnection	The connection string for the cache database. This property is always used in conjunction with CacheProvider . Setting both properties will override the value set for CacheLocation for caching data.
CacheLocation	Specifies the path to the cache when caching to a file.
CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache .
Offline	Use offline mode to get the data from the cache instead of the live source.
CacheMetadata	This property determines whether or not to cache the table metadata to a file store.

36.2.9. Miscellaneous

Property	Description
Database	The default database queried for your Neo4j instance.
BatchSize	The maximum size of each batch operation to submit.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property specifies whether to connect to the Neo4j when the connection is opened.

<u>MaxRows</u>	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
<u>Other</u>	These hidden properties are used only in specific use cases.
<u>Pagesize</u>	The maximum number of results to return per page from Neo4j.
<u>PoolIdleTimeout</u>	The allowed idle time for a connection before it is closed.
<u>PoolMaxSize</u>	The maximum connections in the pool.
<u>PoolMinSize</u>	The minimum number of connections in the pool.
<u>PoolWaitTime</u>	The max seconds to wait for an available connection.
<u>PseudoColumns</u>	This property indicates whether or not to include pseudo columns as columns to the table.
<u>QueryPassthrough</u>	This option passes the query to the Neo4j server as is.
<u>RowScanDepth</u>	The maximum number of rows to scan to look for the columns available in a table.
<u>RTK</u>	The runtime key used for licensing.
<u>Timeout</u>	The value in seconds until the timeout error is thrown, canceling the operation.
<u>UseConnectionPooling</u>	This property enables connection pooling.
<u>UserDefinedViews</u>	A filepath pointing to the JSON configuration file containing your custom views.

For more details, see [CData JDBC Driver for Neo4j - CData JDBC Driver for Neo4j](#).

37. ORACLE

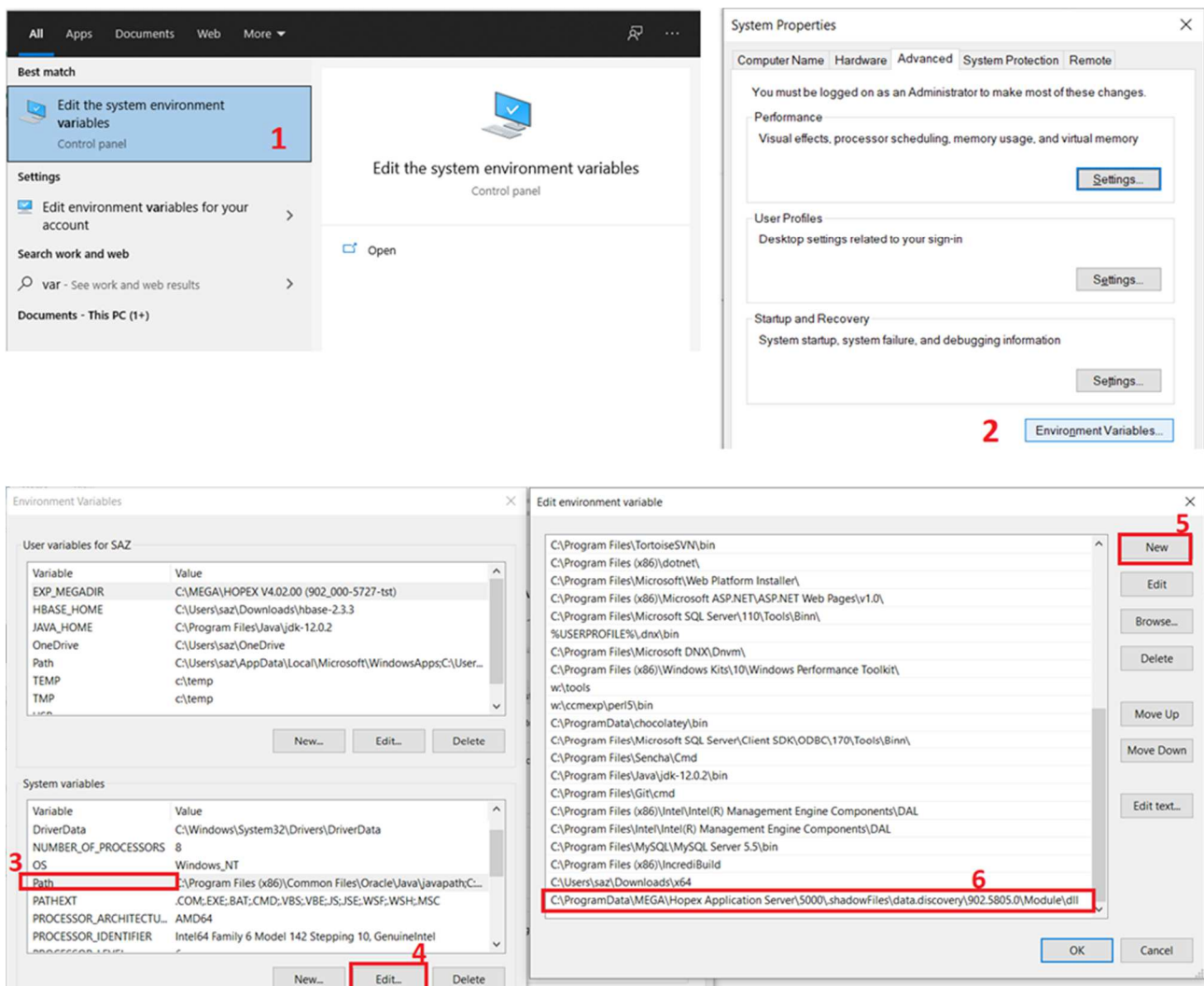
37.1. Prerequisite

After installing the HOPEX Discovery module, you must indicate the system where the supplied .dll files that allow connection to the Oracle server are located.

The path to these files must be added to the "PATH" variable in system variables of Windows. The path depends on system version:

32 bit: C:\ProgramData\MEGA\Hopex Application
Server\<ClusterName>\.shadowFiles\data.discovery\<Version Number>
\Module\Connectors\oracle\32

64 bit: C:\ProgramData\MEGA\Hopex Application
Server\<ClusterName>\.shadowFiles\data.discovery\<Version
Number>\Module\Connectors\oracle\64



You must restart your computer so that the addition of the .dll directory is taken into account by Windows.

37.2. Oracle OCI Version Support

The driver can connect Oracle Database 11.2 or later.

37.3. Connection Options

37.3.1. Authentication

Property	Description
Server	The host name or IP of the server hosting the Oracle database.
Port	The port used to connect to the server hosting the Oracle database.
ServiceName	The service name of the Oracle database.
User	The Oracle OCI user account used to authenticate.
Password	The password used to authenticate the user.
DataSource	Oracle Net Services Name, Connect Descriptor (known also as TNS Connect String), or an easy connect naming that identifies the database to which to connect.

37.3.2. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).

MaxLogFileCount	A string specifying the maximum file count of log files.

37.3.3. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.
IncludeSynonyms	Query meta data for synonyms as they are being the regular tables.
ShowMetadataDescriptions	Controls whether table and column descriptions are returned via the platform metadata APIs and sys_tables / sys_views / sys_tablecolumns.
UseDBAMetadataViews	Query meta data from DBA_.. system views instead of ALL_.. system views.

37.3.4. Miscellaneous

Property	Description
AllowPreparedStatement	Prepare a query statement before its execution.
BatchSize	The maximum size of each batch operation to submit.

ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the Oracle OCI when the connection is opened.
IncludeSynonyms	Query meta data for synonyms as they are being the regular tables.
MaxLobSize	The volume in numbers of bytes or UTF-8 chars which is allowed to query by non-parameterized SELECT query.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.
QueryPassthrough	This option passes the query to the Oracle OCI server as is.
Readonly	You can use this property to enforce read-only access to Oracle OCI from the provider.
ReconnectTimeout	The sleep time, in seconds, before retrying to reconnect to the server on a maximum idle time exceeded error.
ReconnectTries	The number of retry to connect server when a maximum idle time exceeded error is reported by server.
RTK	The runtime key used for licensing.
UseConnectionPooling	This property enables connection pooling.
UseDBAMetadataViews	Query meta data from DBA_.. system views instead of ALL_.. system views.

For more details, see <https://cdn.cdata.com/help/SOG/jdbc/>.

38. PARQUET

38.1. Connection Options

38.1.1. Authentication

Property	Description
AuthScheme	The type of authentication to use when connecting to remote services.
AccessKey	Your account access key. This value is accessible from your security credentials page.
SecretKey	Your account secret key. This value is accessible from your security credentials page.
ApiKey	The API Key used to identify the user to IBM Cloud.
SharePointEdition	The edition of SharePoint being used. Set either SharePointOnline or SharePointOnPremise.

38.1.2. Connection

Property	Description
URI	The Uniform Resource Identifier (URI) for the Parquet resource location.
DataModel	Specifies the data model to use when parsing Parquet documents and generating the database metadata.
Region	The hosting region for your S3-like Web Services.
ProjectId	The id of the project where your Google Cloud Storage instance resides.
OracleNamespace	The Oracle Cloud Object Storage namespace to use.

StorageBaseURL	The URL of a cloud storage service provider.
UseVirtualHosting	If true (default), buckets will be referenced in the request using the <code>hosted-style</code> request: <code>http://yourbucket.s3.amazonaws.com/yourobject</code> . If set to false, the bean will use the <code>path-style</code> request: <code>http://s3.amazonaws.com/yourbucket/yourobject</code> . Note that this property will be set to false, in case of an S3 based custom service when the CustomURL is specified.

38.1.3. AWS Authentication

Property	Description
AWSAccessKey	Your AWS account access key. This value is accessible from your AWS security credentials page.
AWSSecretKey	Your AWS account secret key. This value is accessible from your AWS security credentials page.
AWSRoleARN	The Amazon Resource Name of the role to use when authenticating.
AWSPrincipalArn	The ARN of the SAML Identity provider in your AWS account.
AWSRegion	The hosting region for your Amazon Web Services.
AWSSessionToken	Your AWS session token.
MFASerialNumber	The serial number of the MFA device if one is being used.
MFAToken	The temporary token available from your MFA device.

38.1.4. Azure Authentication

Property	Description
AzureStorageAccount	The name of your Azure storage account.

AzureAccessKey	The storage key associated with your Parquet account.
AzureSharedAccessSignature	A shared access key signature that may be used for authentication.
AzureTenant	The Microsoft Online tenant being used to access data. If not specified, your default tenant will be used.
AzureEnvironment	The Azure Environment to use when establishing a connection.

38.1.5. SSO

Property	Description
SSOLoginURL	The identity provider's login URL.
SSOProperties	Additional properties required to connect to the identity provider in a semicolon-separated list.

38.1.6. OAuth

Property	Description
InitiateOAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.
OAuthVersion	The version of OAuth being used.
OAuthClientId	The client ID assigned when you register your application with an OAuth authorization server.
OAuthClientSecret	The client secret assigned when you register your application with an OAuth authorization server.
OAuthAccessToken	The access token for connecting using OAuth.

OAuthAccessTokenSecret	The OAuth access token secret for connecting using OAuth.
OAuthSettingsLocation	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH. Alternatively, this can be held in memory by specifying a value starting with memory://.
CallbackURL	The OAuth callback URL to return to when authenticating. This value must match the callback URL you specify in your app settings.
OAuthGrantType	The grant type for the OAuth flow.
OAuthPasswordGrantMode	How to pass Client ID and Secret with OAuthGrantType is set to Password.
OAuthIncludeCallbackURL	Whether to include the callback URL in an access token request.
OAuthAuthorizationURL	The authorization URL for the OAuth service.
OAuthAccessTokenURL	The URL to retrieve the OAuth access token from.
OAuthRefreshTokenURL	The URL to refresh the OAuth token from.
OAuthRequestTokenURL	The URL the service provides to retrieve request tokens from. This is required in OAuth 1.0.
OAuthVerifier	The verifier code returned from the OAuth authorization URL.
AuthToken	The authentication token used to request and obtain the OAuth Access Token.
AuthKey	The authentication secret used to request and obtain the OAuth Access Token.
OAuthParams	A comma-separated list of other parameters to submit in the request for the OAuth access token in the format paramname=value.
OAuthRefreshToken	The OAuth refresh token for the corresponding OAuth access token.

OAuthExpiresIn	The lifetime in seconds of the OAuth AccessToken.
OAuthTokenTimestamp	The Unix epoch timestamp in milliseconds when the current Access Token was created.

38.1.7. JWT OAuth

Property	Description
OAuthJWTCert	The JWT Certificate store.
OAuthJWTCertType	The type of key store containing the JWT Certificate.
OAuthJWTCertPassword	The password for the OAuth JWT certificate.
OAuthJWTCertSubject	The subject of the OAuth JWT certificate.
OAuthJWTIssuer	The issuer of the Java Web Token.
OAuthJWTSubject	The user subject for which the application is requesting delegated access.

38.1.8. SSL

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.
SSLClientCertSubject	The subject of the TLS/SSL client certificate.
SSLMode	The authentication mechanism to be used when connecting to the FTP or FTPS server.

SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

38.1.9. SSH

Property	Description
SSHAuthMode	The authentication method to be used to log on to an SFTP server.
SSHClientCert	A certificate to be used for authenticating the user.
SSHClientCertPassword	The password of the SSHClientCert certificate if it has one.
SSHClientCertType	The type of SSHClientCert certificate.

38.1.10. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

38.1.11. Proxy

Property	Description

ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.
ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.
ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

38.1.12. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

38.1.13. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.
FlattenObjects	Set FlattenObjects to true to flatten object properties into columns of their own. Otherwise, objects nested in arrays are returned as strings of JSON.
FlattenArrays	By default, nested arrays are returned as strings. The FlattenArrays property can be used to flatten the elements of nested arrays into columns of their own. Set FlattenArrays to the number of elements you want to return from nested arrays.

38.1.14. Caching

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .
CacheDriver	The database driver to be used to cache data.
CacheConnection	The connection string for the cache database. This property is always used in conjunction with CacheProvider . Setting both

	properties will override the value set for CacheLocation for caching data.
CacheLocation	Specifies the path to the cache when caching to a file.
CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache .
Offline	Use offline mode to get the data from the cache instead of the live source.
CacheMetadata	This property determines whether or not to cache the table metadata to a file store.

38.1.15. Miscellaneous

Property	Description
AggregateFiles	When set to true, the provider will aggregate all the files in URI directory into a single result. With this option enabled, the AggregatedFiles will be exposed which can be used to query the dataset.
BatchSize	The maximum size of each batch operation to submit.
Charset	Specifies the session character set for encoding and decoding character data transferred to and from the Parquet file. The default value is UTF-8.
ClientCulture	This property can be used to specify the format of data (e.g., currency values) that is accepted by the client application. This property can be used when the client application does not support the machine's culture settings. For example, Microsoft Access requires 'en-US'.
Compression	Specifies which compression encoding to be used when creating .parquet files using Create Table Statement and Bulk Inserts.

ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the Parquet when the connection is opened.
Culture	This setting can be used to specify culture settings that determine how the provider interprets certain data types that are passed into the provider. For example, setting Culture='de-DE' will output German formats even on an American machine.
DeleteDownloadedFiles	When set to true, the provider will delete parsed .parquet files downloaded from cloud sources.
DirectoryRetrievalDepth	Limit the subfolders recursively scanned when IncludeSubdirectories is enabled.
EnableDictionary	When set to true, the provider will enable dictionary encoding when creating .parquet files using Create Table Statement and Bulk Inserts.
ExcludeFiles	Comma-separated list of file extensions to exclude from the set of the files modeled as tables.
IncludeDropboxTeamResources	Indicates if you want to include Dropbox team files and folders.
IncludeFiles	Comma-separated list of file extensions to include into the set of the files modeled as tables.
IncludeSubdirectories	Whether to read files from nested folders. In the case of a name collision, table names are prefixed by the underscore-separated folder names.
InsertMode	The behavior when using bulk inserts to create Parquet files.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.

MetadataDiscoveryURI	Used when aggregating multiple files into one table, this property specifies a specific file to read to determine the aggregated table schema.
Other	These hidden properties are used only in specific use cases.
PageSize	(Optional) PageSize value.
ParallelPagingSizeLimit	Parquet file size limit (MegaBytes) for which to use parallel paging.
PathSeparator	Determines the character which will be used to replace the file separator.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.
PseudoColumns	This property indicates whether or not to include pseudo columns as columns to the table.
RTK	The runtime key used for licensing.
SupportEnhancedSQL	This property enhances SQL functionality beyond what can be supported through the API directly, by enabling in-memory client-side processing.
TemporaryLocalFolder	The path, or URI, to the folder that is used to temporarily download parquet file(s).
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
UseConnectionPooling	This property enables connection pooling.

For more details, see <https://cdn.cdata.com/help/HIG/jdbc/>.

39. POSTGRESQL

39.1. PostgreSQL Version Support

The driver enables standards-based access to PostgreSQL databases version 7.4 and later.

39.2. Connection Options

39.2.1. Authentication

Property	Description
AuthScheme	The scheme used for authentication. Accepted entries are Password, AzureAD, AzurePassword, AzureMSI, AwsIAMRoles.
Server	The host name or IP address of the server.
Database	The name of the PostgreSQL database.
User	The PostgreSQL user account used to authenticate.
Password	The password used to authenticate the user.
Port	The port number of the PostgreSQL server.
UseSSL	This field sets whether SSL is enabled.
Visibility	Visibility restrictions used to filter exposed metadata for tables with privileges granted to them for current user. For example 'SELECT,INSERT' filter is restricting metadata visibility only for those tables which may be accessed by current user for SELECT and INSERT operations. Supported privilege values are SELECT, INSERT, UPDATE, DELETE, REFERENCES.

39.2.2. AWS Authentication

Property	Description
AWSAccessKey	Your AWS account access key. This value is accessible from your AWS security credentials page.
AWSSecretKey	Your AWS account secret key. This value is accessible from your AWS security credentials page.
AWSRoleARN	The Amazon Resource Name of the role to use when authenticating.

39.2.3. Azure Authentication

Property	Description
AzureTenant	The Microsoft Online tenant being used to access data. If not specified, your default tenant will be used.

39.2.4. OAuth

Property	Description
InitiateOAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.
OAuthClientId	The client ID assigned when you register your application with an OAuth authorization server.
OAuthClientSecret	The client secret assigned when you register your application with an OAuth authorization server.
OAuthAccessToken	The access token for connecting using OAuth.

OAuthSettingsLocation	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH. Alternatively, this can be held in memory by specifying a value starting with memory://.
CallbackURL	The OAuth callback URL to return to when authenticating. This value must match the callback URL you specify in your app settings.
OAuthGrantType	The grant type for the OAuth flow.
OAuthVerifier	The verifier code returned from the OAuth authorization URL.
OAuthRefreshToken	The OAuth refresh token for the corresponding OAuth access token.
OAuthExpiresIn	The lifetime in seconds of the OAuth AccessToken.
OAuthTokenTimestamp	The Unix epoch timestamp in milliseconds when the current Access Token was created.

39.2.5. Kerberos

Property	Description
KerberosKDC	The Kerberos Key Distribution Center (KDC) service used to authenticate the user.
KerberosRealm	The Kerberos Realm used to authenticate the user with.
KerberosSPN	The service principal name (SPN) for the Kerberos Domain Controller.
KerberosKeytabFile	The Keytab file containing your pairs of Kerberos principals and encrypted keys.
KerberosServiceRealm	The Kerberos realm of the service.
KerberosServiceKDC	The Kerberos KDC of the service.

KerberosTicketCache

The full file path to an MIT Kerberos credential cache file.

39.2.6. SSL

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.
SSLClientCertSubject	The subject of the TLS/SSL client certificate.
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

39.2.7. SSH

Property	Description
SSHAuthMode	The authentication method to be used to log on to an SFTP server.
SSHClientCert	A certificate to be used for authenticating the user.
SSHClientCertPassword	The password of the SSHClientCert certificate if it has one.
SSHClientCertSubject	The subject of the SSH client certificate.
SSHClientCertType	The type of SSHClientCert certificate.
SSHServer	The SSH server.
SSHPort	The SSH port.

SSHUser	The SSH user.
SSHPassword	The SSH password.
SSHServerFingerprint	The SSH server fingerprint.
UseSSH	Whether to tunnel the PostgreSQL connection over SSH. Use SSH.

39.2.8. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

39.2.9. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).

MaxLogFileCount	A string specifying the maximum file count of log files.

39.2.10. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.
IgnoredSchemas	Visibility restriction filter which is used to hide the list of schemas by metadata quering. For example, 'information_schema, pg_catalog'. Schema names are case sensitive.

39.2.11. Miscellaneous

Property	Description
AllowPreparedStatement	Prepare a query statement before its execution.
BatchSize	The maximum size of each batch operation to submit.
BrowsePartitions	By default the provider exposes super table and its partitions by metadata. You may hide sub partition by setting this property into false.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.

ConnectOnOpen	This property specifies whether to connect to the PostgreSQL when the connection is opened.
FetchResultSetMetadata	This field sets whether the provider is getting detailed information about resultset columns from the server.
IncludeTableTypes	If set to true, the provider will query for the types of individual tables and views.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.
QueryPassthrough	This option passes the query to the PostgreSQL server as is.
Readonly	You can use this property to enforce read-only access to PostgreSQL from the provider.
RTK	The runtime key used for licensing.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
TimeZone	Notifies the server about the time zone on the client with standard SET TIMEZONE query when connection is being opened.
UseConnectionPooling	This property enables connection pooling.

For more details, see <https://cdn.cdata.com/help/FPG/jdbc/>.

40. REDIS

40.1. Redis Version Support

The driver models Redis instances as relational databases. The driver leverages the Redis commands to enable bidirectional access to Redis and Redis Enterprise data through SQL. Redis 2.8.0 or above is required.

40.2. Connection Options

40.2.1. Authentication

Property	Description
AuthScheme	The authentication mechanism that the provider will use to authenticate with Redis.
Server	The host name or IP address of the server hosting the Redis instance.
Port	The port for the Redis database.
LogicalDatabase	The index of the Redis Logical Database.
Password	The password used to authenticate with Redis.
EnableCluster	This field sets whether the Redis Cluster Mode is enabled.
UseSSL	This field sets whether SSL is enabled.

40.2.2. Connection

Property	Description
DefineTables	Define the tables exposed by the provider using table names and Redis key patterns.

PatternSeparator	Define the table pattern's separator.
TablePattern	Define the tables exposed by the provider using Redis key patterns.

40.2.3. SSL

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.
SSLClientCertSubject	The subject of the TLS/SSL client certificate.
SSLStartMode	This property determines how the provider starts the SSL negotiation.
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

40.2.4. SSH

Property	Description
SSHAuthMode	The authentication method to be used to log on to an SFTP server.
SSHClientCert	A certificate to be used for authenticating the user.
SSHClientCertPassword	The password of the SSHClientCert certificate if it has one.
SSHClientCertSubject	The subject of the SSH client certificate.

SSHClientCertType	The type of SSHClientCert certificate.
SSHServer	The SSH server.
SSHPort	The SSH port.
SSHUser	The SSH user.
SSHPassword	The SSH password.
SSHServerFingerprint	The SSH server fingerprint.
UseSSH	Whether to tunnel the Redis connection over SSH. Use SSH.

40.2.5. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

40.2.6. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.

LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

40.2.7. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.

40.2.8. Caching

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .
CacheDriver	The database driver to be used to cache data.
CacheConnection	The connection string for the cache database. This property is always used in conjunction with CacheProvider . Setting both properties will override the value set for CacheLocation for caching data.

CacheLocation	Specifies the path to the cache when caching to a file.
CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache .
Offline	Use offline mode to get the data from the cache instead of the live source.
CacheMetadata	This property determines whether or not to cache the table metadata to a file store.

40.2.9. Miscellaneous

Property	Description
BatchSize	The maximum size of each batch operation to submit.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the Redis when the connection is opened.
IgnoreTypeErrors	Removes support for the specified data types and ignores casting exceptions for those types.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.
ParallelMode	This option sets whether the provider should use multiple connections when connecting to Redis.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.

PoolWaitTime	The max seconds to wait for an available connection.
PseudoColumns	This property indicates whether or not to include pseudo columns as columns to the table.
QueryTimeout	The timeout in seconds for which the provider will wait for the query response. The default value is -1, which indicates the provider should never time out.
Readonly	You can use this property to enforce read-only access to Redis from the provider.
RowScanDepth	The maximum number of rows to scan to look for the columns available in a table.
RTK	The runtime key used for licensing.
SupportEnhancedSQL	This property enhances SQL functionality beyond what can be supported through the API directly, by enabling in-memory client-side processing.
TableScanDepth	The maximum number of keys to scan when looking for tables available in your Redis database.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
UseConnectionPooling	This property enables connection pooling.

For more details, see <https://cdn.cdata.com/help/ElG/jdbc/>.

41. REDSHIFT

41.1. Amazon Redshift Version Support

The driver enables standards-based access to Amazon Redshift based on PostgreSQL 8.0.2 with Amazon Redshift SQL.

41.2. Connection Options

41.2.1. Authentication

Property	Description
AuthScheme	The type of authentication to use when connecting to Amazon Redshift.
Server	The host name or IP address of the Amazon Redshift cluster.
Port	The port number of the Amazon Redshift server.
Database	The name of the Amazon Redshift database.
User	The Amazon Redshift user account used to authenticate.
Password	The password used to authenticate the user.
UseSSL	This field sets whether SSL is enabled.
Visibility	Visibility restrictions used to filter exposed metadata for tables with privileges granted to them for current user. For example 'SELECT,INSERT' filter is restricting metadata visibility only for those tables which may be accessed by current user for SELECT and INSERT operations. Supported privilege values are SELECT, INSERT, UPDATE, DELETE, REFERENCES.

41.2.2. AWS Authentication

Property	Description
AwsAccessKey	Your AWS account access key or the access key for an authorized IAM user.
AwsSecretKey	Your AWS account secret key or the secret key for an authorized IAM user.
AutoCreate	Specify true to create a database user with the name specified for User if one does not exist while connecting with IAM credentials. See AuthScheme .
DbGroups	A comma-delimited list of the names of one or more existing database groups the database user joins for the current session when connecting with IAM credentials. See AuthScheme .
AWSRoleARN	The Amazon Resource Name of the role to use when authenticating.
AWSPrincipalArn	The ARN of the SAML Identity provider in your AWS account.
AWSRegion	The hosting region for your Amazon Web Services.
AWSSessionToken	Your AWS session token.

41.2.3. SSO

Property	Description
SSOLoginURL	The identity provider's login URL.
SSOProperties	Additional properties required to connect to the identity provider in a semicolon-separated list.
SSOExchangeUrl	The url used for consuming the SAML response and exchanging it with Amazon Redshift specific credentials.

41.2.4. SSL

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.
SSLClientCertSubject	The subject of the TLS/SSL client certificate.
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

41.2.5. SSH

Property	Description
SSHAuthMode	The authentication method to be used to log on to an SFTP server.
SSHClientCert	A certificate to be used for authenticating the user.
SSHClientCertPassword	The password of the SSHClientCert certificate if it has one.
SSHClientCertSubject	The subject of the SSH client certificate.
SSHClientCertType	The type of SSHClientCert certificate.
SSHServer	The SSH server.
SSHPort	The SSH port.
SSHUser	The SSH user.

SSHPassword	The SSH password.
SSHServerFingerprint	The SSH server fingerprint.
UseSSH	Whether to tunnel the Amazon Redshift connection over SSH. Use SSH.

41.2.6. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

41.2.7. Proxy

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.
ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.

ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.
ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

41.2.8. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

41.2.9. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.

Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.
IgnoredSchemas	Visibility restriction filter which is used to hide the list of schemas by metadata quering. For example, 'information_schema, pg_catalog'. Schema names are case sensitive.

41.2.10. Miscellaneous

Property	Description
AllowPreparedStatement	Prepare a query statement before its execution.
BatchSize	The maximum size of each batch operation to submit.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the Amazon Redshift when the connection is opened.
FetchResultSetMetadata	This field sets whether the provider is getting detailed information about resultset columns from the server.
IncludeTableTypes	If set to true, the provider will query for the types of individual tables and views.
InsertMode	Specifies what method to use when inserting bulk data. By default DML mode is used.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.

PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.
QueryPassthrough	This option passes the query to the Amazon Redshift server as is.
Readonly	You can use this property to enforce read-only access to Amazon Redshift from the provider.
RTK	The runtime key used for licensing.
S3Bucket	Specifies the name of AWS S3 bucket to upload bulk data for staging.
S3BucketFolder	Specifies the name of the folder in AWS S3 bucket to upload bulk data for staging. By default bulk data are staged in the root folder.
StripOutNulls	When set the null characters are stripped out from character values in bulk operations.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
TimeZone	Set time zone for the current session.
UseConnectionPooling	This property enables connection pooling.

For more details, see <https://cdn.cdata.com/help/FRG/jdbc/>.

42. SALESFORCE

42.1. Connection Options

42.1.1. Authentication

Property	Description
AuthScheme	The type of authentication to use when connecting to Salesforce.
User	The Salesforce user account used to authenticate.
Password	The password used to authenticate the user.
SecurityToken	The security token used to authenticate access to the Salesforce account.
UseSandbox	A boolean determining if the connection should be made to a Salesforce sandbox account.
CredentialsLocation	The location of the settings file where token retrieved with OKTA MFA is saved.

42.1.2. Connection

Property	Description
APIVersion	The version of the Salesforce API used.
LoginURL	URL to the Salesforce server used for logging in.

42.1.3. SSO

Property	Description
SSOLoginURL	The identity provider's login URL.
SSOProperties	Additional properties required to connect to the identity provider in a semicolon-separated list.
SSOExchangeUrl	The url used for consuming the SAML response and exchanging it with Salesforce specific credentials.

42.1.4. BulkAPI

Property	Description
UseBulkAPI	Whether to use the synchronous SOAP API or the asynchronous Bulk API.
BulkAPIConcurrencyMode	The concurrency mode for processing bulk rows with BULK API v1.
BulkPageSize	The number of records to retrieve before returning results to the user when UseBulkAPI=true.
BulkPollingInterval	The time interval in milliseconds between requests that check the availability of the bulk query response. The default value is 500 ms.
BulkQueryTimeout	The timeout in minutes for which the provider will wait for a bulk query response. The default value is 25 minutes.
WaitForBulkResults	Whether to wait for bulk results when using the asynchronous API. Only active when UseBulkAPI is true.

42.1.5. OAuth

Property	Description
InitiateOAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.
OAuthClientId	The client ID assigned when you register your application with an OAuth authorization server.
OAuthClientSecret	The client secret assigned when you register your application with an OAuth authorization server.
OAuthAccessToken	The access token for connecting using OAuth.
OAuthSettingsLocation	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH. Alternatively, this can be held in memory by specifying a value starting with memory://.
CallbackURL	The OAuth callback URL to return to when authenticating. This value must match the callback URL you specify in your app settings.
OAuthServerURL	The server URL to use if authenticating with OAuth.
OAuthGrantType	The grant type for the OAuth flow.
OAuthAuthorizationURL	The authorization URL for the OAuth service.
OAuthAccessTokenURL	The URL to retrieve the OAuth access token from.
OAuthVerifier	The verifier code returned from the OAuth authorization URL.
OAuthRefreshToken	The OAuth refresh token for the corresponding OAuth access token.
OAuthExpiresIn	The lifetime in seconds of the OAuth AccessToken.

OAuthTokenTimestamp	The Unix epoch timestamp in milliseconds when the current Access Token was created.

42.1.6. JWT OAuth

Property	Description
OAuthJWTCert	The JWT Certificate store.
OAuthJWTCertType	The type of key store containing the JWT Certificate.
OAuthJWTCertPassword	The password for the OAuth JWT certificate.
OAuthJWTCertSubject	The subject of the OAuth JWT certificate.
OAuthJWTIssuer	The issuer of the Java Web Token.
OAuthJWTSubject	The user subject for which the application is requesting delegated access.

42.1.7. SSL

Property	Description
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

42.1.8. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.

FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

42.1.9. Proxy

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.
ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.
ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

42.1.10. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.

Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

42.1.11. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.

42.1.12. Caching

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .
CacheDriver	The database driver to be used to cache data.

CacheConnection	The connection string for the cache database. This property is always used in conjunction with CacheProvider . Setting both properties will override the value set for CacheLocation for caching data.
CacheLocation	Specifies the path to the cache when caching to a file.
CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache .
Offline	Use offline mode to get the data from the cache instead of the live source.
CacheMetadata	This property determines whether or not to cache the table metadata to a file store.

42.1.13. Miscellaneous

Property	Description
AllOrNone	A boolean indicating if you would like all inserts, updates, or deletes to fail in a request if even a single record fails.
ArchiveMode	Boolean indicating whether to include deleted and archived records with a standard SELECT query.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the Salesforce when the connection is opened.
ContinueOnAlterException	Whether you want to continue after a ALTER statement has failed.
FilterScope	Optional scope to limit the records returned from queries.

IncludeMetadataDescription	Set this property to a value other than NONE if you want to retrieve the descriptions for columns, tables or both of them from the Metadata API.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.
PseudoColumns	This property indicates whether or not to include pseudo columns as columns to the table.
Readonly	You can use this property to enforce read-only access to Salesforce from the provider.
RTK	The runtime key used for licensing.
ServerSideAggregation	A boolean determining if server side aggregation should be used.
SessionTimeout	The time in minutes for which a Salesforce login session is reused.
SkipFormulaFields	Set to true if formula fields should be skipped when listing columns.
SupportEnhancedSQL	This property enhances SQL functionality beyond what can be supported through the API directly, by enabling in-memory client-side processing.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.

UseConnectionPooling	This property enables connection pooling.
UseDisplayNames	Boolean determining if the display names for the columns should be used instead of the API names.

For more details, see <https://cdn.cdata.com/help/RFG/jdbc/>.

43. SAP

43.1. SAP ERP Version Support

The driver enables relational access to tables, queries, and function modules in your SAP R/3, NetWeaver, or ERP/ECC 6.0+ system.

43.2. Connection Options

43.2.1. Authentication

Property	Description
ConnectionType	The type of connection you are making.
Host	Host name of the target system.
SystemNumber	The number by which the target system is defined. Used when setting the Host connection property.
User	The user that is authenticating to the SAP system.
Password	The password used to authenticate to the SAP system.
Client	The client authenticating to the SAP system.
X509Certificate	The X509 certificate used for login as an alternative to User , and Password .
MessageServer	The message server must be specified when connecting to an SAP system that uses load balancing.
Group	The Logon Group being used. This typically only needs to be specified when connecting to an SAP system that uses load balancing.
SystemId	The System Id or R3Name of the SAP System is a string with a maximum of three characters. It is often used in load balancing connections.

RFCURL	The URL of the SOAP interface to connect with SAP.
MessageServerService	The message server service you wish to connect to.

43.2.2. Security

Property	Description
SNCMode	A boolean determining if you are using SNC. Set this to true to use SNC.
SNCName	An optional input with the name of your SNC connection.
SNCQop	The quality of protection for your SNC connection.
SNCPartnerName	The application server's SNC name. This is a required input when using SNC.
SNCLibPath	An optional input detailing the path and file name of the external library.

43.2.3. SSL

Property	Description
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

43.2.4. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.

FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

43.2.5. Proxy

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.
ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.
ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

43.2.6. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.

Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

43.2.7. Schema

Property	Description
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.

43.2.8. Caching

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .
CacheDriver	The database driver to be used to cache data.
CacheConnection	The connection string for the cache database. This property is always used in conjunction with CacheProvider . Setting both properties will override the value set for CacheLocation for caching data.

CacheLocation	Specifies the path to the cache when caching to a file.
CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache .
Offline	Use offline mode to get the data from the cache instead of the live source.
CacheMetadata	This property determines whether or not to cache the table metadata to a file store.

43.2.9. Miscellaneous

Property	Description
BatchSize	The maximum size of each batch operation to submit.
Charset	The system code page used for Unicode to multibyte translations.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the SAP ERP when the connection is opened.
Destination	Reference to an existing destination that is specified in a local saprfc.ini file.
EndianType	The endian type for the SAP server. Enter either Big or Little.
GatewayHost	The gateway host you wish to connect to.
GatewayService	The gateway service you wish to connect to.
GenerateSchemaFiles	Determines how schema files should be generated.
InitialValueMode	How to treat initial values in SAP.
Language	The language value to be used when connecting to the SAP system.

Location	A path to the directory that contains the schema files defining tables, views, and stored procedures to work with your chosen data source. This must be set in order to add new tables to provider.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.
Pagesize	The number of results to return per page from SAP. Only used for SAP tables.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.
PseudoColumns	This property indicates whether or not to include pseudo columns as columns to the table.
QueryMode	Determines which SAP queries will be displayed as views, if any.
ReadTableFunction	The function to use for reading table data.
RTK	The runtime key used for licensing.
StoredProcedureFilter	A filter indicating which function modules to report as stored procedures.
SupportEnhancedSQL	This property enhances SQL functionality beyond what can be supported through the API directly, by enabling in-memory client-side processing.
TableMode	Determines which SAP tables will be displayed as views if any.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.

UseConnectionPooling	This property enables connection pooling.
UseLabels	Boolean determining if labels should be used for table and column names.
UseSimpleNames	Boolean determining if simple names should be used for tables and columns.
UseUnicodeRFC	A boolean indicating if you want to use RFC_GET_UNICODE_STRUCTURE to get structure information.

For more details, see <https://cdn.cdata.com/help/RYG/jdbc/>.

44. SAP HANA

44.1. Connection Options

44.1.1. Authentication

Property	Description
AuthScheme	The scheme used for authentication. Accepted entries are Password.
Server	The name of the server running SAP HANA database.
Port	The port of the SAP HANA database.
Database	The name of the SAP HANA database.
User	The SAP HANA user account used to authenticate.
Password	The password used to authenticate the user.
IncludeSystemObjects	Set IncludeSystemObjects to True to fetch Hana System schema and tables.
UseSSL	This field sets whether SSL is enabled.

44.1.2. SSL

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.

SSLClientCertSubject	The subject of the TLS/SSL client certificate.
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

44.1.3. SSH

Property	Description
SSHAuthMode	The authentication method to be used to log on to an SFTP server.
SSHClientCert	A certificate to be used for authenticating the user.
SSHClientCertPassword	The password of the SSHClientCert certificate if it has one.
SSHClientCertSubject	The subject of the SSH client certificate.
SSHClientCertType	The type of SSHClientCert certificate.
SSHServer	The SSH server.
SSHPort	The SSH port.
SSHUser	The SSH user.
SSHPassword	The SSH password.
SSHServerFingerprint	The SSH server fingerprint.
UseSSH	Whether to tunnel the SAP HANA connection over SSH. Use SSH.

44.1.4. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

44.1.5. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

44.1.6. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.

44.1.7. Miscellaneous

Property	Description
BatchSize	The maximum size of each batch operation to submit.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the SAP HANA when the connection is opened.
IncludeTableTypes	If set to true, the provider will report the types of individual tables and views.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.

PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.
QueryPassthrough	This option passes the query to the SAP HANA server as is.
Readonly	You can use this property to enforce read-only access to SAP HANA from the provider.
RTK	The runtime key used for licensing.
SessionVariables	A comma-separated list of session variables to set on the current connection.
Timeout	A timeout for the provider.
UseConnectionPooling	This property enables connection pooling.

For more details, see <https://cdn.cdata.com/help/SNG/jdbc/>.

45. SHAREPOINT

45.1. SharePoint Version Support

The driver supports all versions of SharePoint that support the SOAP API. This includes: Windows SharePoint Services 3.0, SharePoint Server 2007+ (2010, 2013, etc.), and SharePoint Online.

45.2. Connection Options

45.2.1. Authentication

Property	Description
AuthScheme	The scheme used for authenticating to SharePoint. For On-Premise instances, accepted entries are NTLM, Basic, Digest, Forms, None, and Negotiate. NTLM is the default. For Sharepoint Online, the accepted entries are OAuth and OAuthJWT.
URL	The base URL for the site.
SharePointEdition	The edition of SharePoint being used. Set either SharePoint Online or SharePoint On-Premise.
User	The SharePoint user account used to authenticate.
Password	The password used to authenticate the user.

45.2.2. Azure Authentication

Property	Description
AzureTenant	The Microsoft Online tenant being used to access data. If not specified, your default tentant will be used.
AzureEnvironment	The Azure Environment to use when establishing a connection.

45.2.3. SSO

Property	Description
SSOLoginURL	The identity provider's login URL.
SSODomain	The domain of the user when using single sign-on (SSO).
SSOProperties	Additional properties required to connect to the identity provider in a semicolon-separated list.

45.2.4. OAuth

Property	Description
InitiateOAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.
OAuthClientId	The client ID assigned when you register your application with an OAuth authorization server.
OAuthClientSecret	The client secret assigned when you register your application with an OAuth authorization server.
OAuthAccessToken	The access token for connecting using OAuth.
CallbackURL	The OAuth callback URL to return to when authenticating. This value must match the callback URL you specify in your Add-In settings.
OAuthSettingsLocation	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH. Alternatively, this can be held in memory by specifying a value starting with memory://.
Scope	The scope used for the OAuth flow to access data from the Application.
OAuthGrantType	The grant type for the OAuth flow.

OAuthVerifier	The verifier code returned from the OAuth authorization URL.
OAuthRefreshToken	The OAuth refresh token for the corresponding OAuth access token.
OAuthExpiresIn	The lifetime in seconds of the OAuth AccessToken.
OAuthTokenTimestamp	The Unix epoch timestamp in milliseconds when the current Access Token was created.

45.2.5. JWT OAuth

Property	Description
OAuthJWTCert	The JWT Certificate store.
OAuthJWTCertType	The type of key store containing the JWT Certificate.
OAuthJWTCertPassword	The password for the OAuth JWT certificate.
OAuthJWTCertSubject	The subject of the OAuth JWT certificate.
OAuthJWTIssuer	The issuer of the Java Web Token.
OAuthJWTSubject	The user subject for which the application is requesting delegated access.

45.2.6. Kerberos

Property	Description
KerberosKDC	The Kerberos Key Distribution Center (KDC) service used to authenticate the user.
KerberosRealm	The Kerberos Realm used to authenticate the user with.

KerberosSPN	The service principal name (SPN) for the Kerberos Domain Controller.
KerberosKeytabFile	The Keytab file containing your pairs of Kerberos principals and encrypted keys.
KerberosServiceRealm	The Kerberos realm of the service.
KerberosServiceKDC	The Kerberos KDC of the service.
KerberosTicketCache	The full file path to an MIT Kerberos credential cache file.

45.2.7. SSL

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.
SSLClientCertSubject	The subject of the TLS/SSL client certificate.
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

45.2.8. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.

FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

45.2.9. Proxy

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.
ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.
ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

45.2.10. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.

Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

45.2.11. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.
Schema	The type of schema to use.

45.2.12. Caching

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .
CacheDriver	The database driver to be used to cache data.

CacheConnection	The connection string for the cache database. This property is always used in conjunction with CacheProvider . Setting both properties will override the value set for CacheLocation for caching data.
CacheLocation	Specifies the path to the cache when caching to a file.
CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache .
Offline	Use offline mode to get the data from the cache instead of the live source.
CacheMetadata	This property determines whether or not to cache the table metadata to a file store.

45.2.13. Miscellaneous

Property	Description
BatchSize	The maximum size of each batch operation to submit.
CalculatedDataType	The data type to be used for calculated fields.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the SharePoint when the connection is opened.
ContinueOnError	Indicates whether or not to continue updating items in a batch after an error.
CreateIDColumns	Indicates whether or not to create supplemental ID columns for SharePoint columns that use values from information stored in other Lists.
FolderOption	An option to determine how to display folders in results. Enter either FilesOnly, FilesAndFolders, Recursive, or RecursiveAll.

IncludeLookupColumns	This option controls whether the driver returns the lookup columns defined on a table.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.
Pagesize	The maximum number of results to return per page from SharePoint.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.
PseudoColumns	This property indicates whether or not to include pseudo columns as columns to the table.
Readonly	You can use this property to enforce read-only access to SharePoint from the provider.
RTK	The runtime key used for licensing.
ShowHiddenColumns	Boolean determining if hidden columns should be shown or not. If false, all hidden columns will be removed from the column listing.
ShowPredefinedColumns	Boolean determining if predefined columns should be shown or not. If false, all columns derived from a base type will be removed from the column listing.
ShowVersionViews	Indicate whether to display the view of list versions. Such as ListA_Versions.
STSURL	The URL of the security token service (STS) when using single sign-on (SSO).

SupportEnhancedSQL	This property enhances SQL functionality beyond what can be supported through the API directly, by enabling in-memory client-side processing.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
UseConnectionPooling	This property enables connection pooling.
UseDisplayNames	Boolean determining if the display names for the columns should be used instead of the API names.
UseNTLMV1	Determines whether the driver will attempt to connect with NTLMv1 or NTLMv2 (default).
UseSimpleNames	Boolean determining if simple names should be used for tables and columns.

For more details, see <https://cdn.cdata.com/help/RSG/jdbc/>.

46. SNOWFLAKE

46.1. Snowflake Version Support

The driver enables standards-based access to all Snowflake editions.

46.2. Connection Options

46.2.1. Authentication

Property	Description
AuthScheme	The authentication scheme used. Accepted entries are Password, OKTA, PrivateKey, AzureAD, OAuth, or PingFederate.
Account	The Account provided for authentication with Snowflake database. This is usually derived from the URL automatically.
Warehouse	The name of the Snowflake warehouse.
User	The username provided for authentication with the Snowflake database.
Password	The user's password.
URL	The URL of Snowflake database.
MFAPasscode	Specifies the passcode to use for multi-factor authentication.
RoleName	The role of the Snowflake user: PUBLIC, SYSADMIN, or ACCOUNTADMIN.
CredentialsLocation	The location of the settings file where credentials are saved.

46.2.2. Connection

Property	Description
UseVirtualHosting	If true (default), buckets will be referenced in the request using the hosted-style request: <code>http://yourbucket.s3.amazonaws.com/yourobject</code> . If set to false, the bean will use the path-style request: <code>http://s3.amazonaws.com/yourbucket/yourobject</code> . Note that this property will be set to false, in case of an S3 based custom service when the CustomURL is specified.

46.2.3. Azure Authentication

Property	Description
AzureTenant	The Microsoft Online tenant being used to access data. If not specified, your default tenant will be used.

46.2.4. SSO

Property	Description
ProofKey	The ProofKey for authentication with Snowflake database. This is usually derived from GetSSOAuthorizationURL call.
ExternalToken	The External Token for authentication with the Snowflake database. This is usually derived from the external handler. For example, handle the callback URL from procedure GetSSOAuthorizationURL will get this token.
SSOProperties	Additional properties required to connect to the identity provider in a semicolon-separated list.

46.2.5. KeyPairAuth

Property	Description
PrivateKey	The private key provided for key pair authentication with Snowflake.
PrivateKeyPassword	The password for the private key specified in the PrivateKey property, if required.
PrivateKeyType	The type of key store containing the private key to use with key pair authentication.

46.2.6. OAuth

Property	Description
InitiateOAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.
OAuthClientId	The client ID assigned when you register your application with an OAuth authorization server.
OAuthClientSecret	The client secret assigned when you register your application with an OAuth authorization server.
OAuthAccessToken	The access token for connecting using OAuth.
CallbackURL	The OAuth callback URL to return to when authenticating. This value must match the callback URL you specify in your Add-In settings.
State	An optional value that has meaning for your OAuth App.
OAuthSettingsLocation	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH. Alternatively, this can be held in memory by specifying a value starting with memory://.

OAuthAuthenticator	This determines the authenticator that the OAuth application requests from Snowflake.
Scope	This determines the scopes that the OAuth application requests from Snowflake.
OAuthAuthorizationURL	The authorization URL for the OAuth service.
OAuthAccessTokenURL	The URL to retrieve the OAuth access token from.
OAuthVerifier	The verifier code returned from the OAuth authorization URL.
PKCEVerifier	A random value used as input for calling GetOAuthAccessToken in the PKCE flow.
OAuthRefreshToken	The OAuth refresh token for the corresponding OAuth access token.
OAuthExpiresIn	The lifetime in seconds of the OAuth AccessToken.
OAuthTokenTimestamp	The Unix epoch timestamp in milliseconds when the current Access Token was created.

46.2.7. SSL

Property	Description
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

46.2.8. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.

FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

46.2.9. Proxy

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.
ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.
ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

46.2.10. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.

LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

46.2.11. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.
Database	The name of the Snowflake database.
Schema	The schema of the Snowflake database.

46.2.12. Caching

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .
CacheDriver	The database driver to be used to cache data.

CacheConnection	The connection string for the cache database. This property is always used in conjunction with CacheProvider . Setting both properties will override the value set for CacheLocation for caching data.
CacheLocation	Specifies the path to the cache when caching to a file.
CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache.
Offline	Use offline mode to get the data from the cache instead of the live source.
CacheMetadata	This property determines whether or not to cache the table metadata to a file store.

46.2.13. Miscellaneous

Property	Description
AllowPreparedStatement	Prepare a query statement before its execution.
ApplicationName	The application name connection string property expresses the HTTP User-Agent.
AsyncQueryTimeout	The timeout for asynchronous requests issued by the provider to download large result sets.
BatchSize	The maximum size of each batch operation to submit.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the Snowflake when the connection is opened.
CustomStage	The name of a custom stage to use during bulk write operations.
EnableArrow	Whether to support Apache Arrow.

ExternalStageAWSAccessKey	Your AWS account access key. Only used when defining a CustomStage for bulk write operations.
ExternalStageAWSSecretKey	Your AWS account secret key. Only used when defining a CustomStage for bulk write operations.
ExternalStageAzureSASToken	The string value of the Azure Blob shared access signature.
IgnoreCase	Whether to ignore case in identifiers. Default: false.
IncludeTableTypes	If set to true, the provider will report the types of individual tables and views.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
MaxThreads	Specifies the number of concurrent requests.
MergeDelete	A boolean indicating whether batch DELETE statements should be converted to MERGE statements automatically. Only used when the DELETE statement's where clause contains a table's primary key field only and they are combined with AND logical operator.
MergeInsert	A boolean indicating whether INSERT statements should be converted to MERGE statements automatically. Only used when the INSERT contains a table's primary key field.
MergeUpdate	A boolean indicating whether batch UPDATE statements should be converted to MERGE statements automatically. Only used when the UPDATE statement's where clause contains a table's primary key field only and they are combined with AND logical operator.
Other	These hidden properties are used only in specific use cases.
Pagesize	The maximum number of results to return per page from Snowflake.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.

PoolWaitTime	The max seconds to wait for an available connection.
PseudoColumns	This property indicates whether or not to include pseudo columns as columns to the table.
QueryPassthrough	This option passes the query to the Snowflake server as is.
Readonly	You can use this property to enforce read-only access to Snowflake from the provider.
ReplaceInvalidUTF8Chars	Specifies whether to replace invalid UTF8 characters with a '?'. s
RetryOnS3Timeout	Whether or not to retry when network issues occur at during chunk downloading.
RTK	The runtime key used for licensing.
SessionParameters	The session parameters for Snowflake. For example: SessionParameters='QUERY_TAG=MyTag;QUOTED_IDENTIFIERS_IGNORE_CASE=True;';.
SupportEnhancedSQL	This property enhances SQL functionality beyond what can be supported through the API directly, by enabling in-memory client-side processing.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
UseAsyncQuery	This field sets whether async query is enabled.
UseConnectionPooling	This property enables connection pooling.

For more details, see <https://cdn.cdata.com/help/OWG/jdbc/>.

47. SPARK

47.1. Spark SQL Version Support

The driver leverages Spark Thrift to enable bidirectional SQL access to SparkSQL data. Spark version 1.6 and above are supported.

47.2. Connection Options

47.2.1. Authentication

Property	Description
AuthScheme	The authentication scheme used. Accepted entries are Plain, LDAP, NOSASL, and Kerberos.
Server	The host name or IP address of the server hosting the SparkSQL database.
Port	The port for the SparkSQL database.
User	The username used to authenticate with SparkSQL.
Password	The password used to authenticate with SparkSQL.
Database	The name of the SparkSQL database.
ProtocolVersion	The Protocol Version used to authenticate with SparkSQL.
ImpersonationProxyUser	The proxy user of the Hive user impersonation.
SaslQop	Quality of protection for the SASL framework. The level of quality is negotiated between the client and server during authentication. Used by Kerberos authentication with TCP transport.
TransportMode	The transport mode to use to communicate with the Hive server. Accepted entries are BINARY and HTTP.

47.2.2. Kerberos

Property	Description
KerberosKDC	The Kerberos Key Distribution Center (KDC) service used to authenticate the user.
KerberosRealm	The Kerberos Realm used to authenticate the user with.
KerberosSPN	The service principal name (SPN) for the Kerberos Domain Controller.
KerberosKeytabFile	The Keytab file containing your pairs of Kerberos principals and encrypted keys.
KerberosServiceRealm	The Kerberos realm of the service.
KerberosServiceKDC	The Kerberos KDC of the service.
KerberosTicketCache	The full file path to an MIT Kerberos credential cache file.

47.2.3. SSL

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.
SSLClientCertSubject	The subject of the TLS/SSL client certificate.
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

47.2.4. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

47.2.5. Proxy

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.
ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.
ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

47.2.6. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

47.2.7. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.

47.2.8. Caching

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .
CacheDriver	The database driver to be used to cache data.
CacheConnection	The connection string for the cache database. This property is always used in conjunction with CacheProvider . Setting both properties will override the value set for CacheLocation for caching data.
CacheLocation	Specifies the path to the cache when caching to a file.
CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache .
Offline	Use offline mode to get the data from the cache instead of the live source.
CacheMetadata	This property determines whether or not to cache the table metadata to a file store.

47.2.9. Miscellaneous

Property	Description
AsyncQueryTimeout	The timeout for asynchronous requests issued by the provider to download large result sets.
BatchSize	The maximum size of each batch operation to submit.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the Spark SQL when the connection is opened.

DescribeCommand	The describe command to determine which describe command will use to communicate with the Hive server. Accepted entries are DESCRIBE and DESC.
DetectView	Specifies whether to use DESCRIBE FORMATTED ... to detect the specified table is view or not.
HTTPPath	The path component of the URL endpoint when using HTTP TransportMode.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.
PseudoColumns	This property indicates whether or not to include pseudo columns as columns to the table.
QueryPassthrough	This option passes the query to the Spark SQL server as is.
Readonly	You can use this property to enforce read-only access to Spark SQL from the provider.
RTK	The runtime key used for licensing.
ServerConfigurations	A name-value list of server configuration variables to override the server defaults.
SupportEnhancedSQL	This property enhances SQL functionality beyond what can be supported through the API directly, by enabling in-memory client-side processing.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.

UseConnectionPooling	This property enables connection pooling.
UseDatabricksUploadApi	This option specifies whether the Databricks Upload API will be use when executing batch insert.
UseInsertSelectSyntax	Specifies whether to use an INSERT INTO SELECT statement.
UseSSL	Specifies whether to use SSL Encryption when connecting to Hive.

For more details, see <https://cdn.cdata.com/help/ESG/jdbc/>.

48. SQL SERVER

48.1. SQL Server Version Support

The driver enables connectivity to SQL Server through the TDS protocol. SQL Server versions 2008, 2012, 2014, 2016, and 2019 are supported.

Connections to Azure SQL Server and Azure Data Warehouse instances are supported as well.

48.2. Connection Options

48.2.1. Authentication

Property	Description
AuthScheme	The scheme used for authentication. Accepted entries are Password, NTLM, Kerberos, AzurePassword, AzureAD, AzureMSI, AzureServicePrincipal.
Server	The name of the server running SQL Server.
Port	The port of the MS SQL Server.
Database	The name of the SQL Server database.
User	The SQL Server user account used to authenticate.
Password	The password used to authenticate the user.
Domain	The name of the domain for a Windows (NTLM) security login.
IntegratedSecurity	Whether or not to authenticate with Windows Integrated Security.
NTLMVersion	The NTLM version.
Encrypt	This field sets whether SSL is enabled.

48.2.2. Azure Authentication

Property	Description
AzureTenant	The Microsoft Online tenant being used to access data. If not specified, your default tenant will be used.

48.2.3. OAuth

Property	Description
InitiateOAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.
OAuthClientId	The client ID assigned when you register your application with an OAuth authorization server.
OAuthClientSecret	The client secret assigned when you register your application with an OAuth authorization server.
CallbackURL	The OAuth callback URL to return to when authenticating. This value must match the callback URL you specify in your app settings.
OAuthGrantType	The grant type for the OAuth flow.
OAuthVerifier	The verifier code returned from the OAuth authorization URL.

48.2.4. Kerberos

Property	Description
KerberosKDC	The Kerberos Key Distribution Center (KDC) service used to authenticate the user.
KerberosRealm	The Kerberos Realm used to authenticate the user with.

KerberosSPN	The service principal name (SPN) for the Kerberos Domain Controller.
KerberosKeytabFile	The Keytab file containing your pairs of Kerberos principals and encrypted keys.
KerberosServiceRealm	The Kerberos realm of the service.
KerberosServiceKDC	The Kerberos KDC of the service.
KerberosTicketCache	The full file path to an MIT Kerberos credential cache file.

48.2.5. SSL

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.
SSLClientCertSubject	The subject of the TLS/SSL client certificate.
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

48.2.6. SSH

Property	Description
SSHAuthMode	The authentication method to be used to log on to an SFTP server.
SSHClientCert	A certificate to be used for authenticating the user.
SSHClientCertPassword	The password of the SSHClientCert certificate if it has one.

SSHClientCertSubject	The subject of the SSH client certificate.
SSHClientCertType	The type of SSHClientCert certificate.
SSHServer	The SSH server.
SSHPort	The SSH port.
SSHUser	The SSH user.
SSHPassword	The SSH password.
SSHServerFingerprint	The SSH server fingerprint.
UseSSH	Whether to tunnel the SQL Server connection over SSH. Use SSH.

48.2.7. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

48.2.8. Proxy

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.
ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.
ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

48.2.9. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).

MaxLogFileCount	A string specifying the maximum file count of log files.

48.2.10. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.

48.2.11. Miscellaneous

Property	Description
ApplicationIntent	The application intent connection string property expresses the client application's request to be directed either to a read-write or read-only version of an availability group database. To use read-only routing, a client must use an application intent of read-only in the connection string when connecting to the availability group listener. Without the read-only application intent, connections to the availability group listener are directed to the database on the primary replica.
ApplicationName	The application name connection string property expresses the HTTP User-Agent.
DefaultDomain	This property is used for the Oracle Database Gateway for ODBC.

EnableForeignKeyDetection	Whether to detect the foreign keys in ODBC.
IncludeDualTable	Set this property to mock the Oracle DUAL table for better compatibility with Oracle database.
LimitKeySize	The maximum length of a primary key column.
MapBigintToVarchar	This property controls whether or not the bigint type maps to SQL_VARCHAR instead of SQL_BIGINT. This property is false by default.
MapToInt	This property controls whether or not the long type maps to SQL_INTEGER instead of SQL_BIGINT. This property is false by default.
MapToLongVarchar	This property controls whether or not a column is returned as SQL_LONGVARCHAR.
MapToWVarchar	This property controls whether or not string types map to SQL_WVARCHAR instead of SQL_VARCHAR. This property is set by default.
MaximumColumnSize	The maximum column size.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.
QueryPassthrough	This option passes the query to the SQL Server server as is.
Readonly	You can use this property to enforce read-only access to SQL Server from the provider.
RTK	The runtime key used for licensing.
Timeout	A timeout for the provider.
UpperCaseIdentifiers	This property reports all identifiers in uppercase. This is the default for Oracle databases and thus allows better

integration with Oracle tools such as the Oracle Database Gateway.

For more details, see <https://cdn.cdata.com/help/RUG/odbc/>.

48.3. Description Extraction

To extract table and column descriptions for SQL Server, you must add the following parameters in the Options field of HOPEX Data Discovery:

ColumnExtendedProperty=MS_Description;

TableExtendedProperty=MS_Description;

Microsoft SQL Server

Server	vp-hadoop	i
Port	61299	i
User	sa	i
Password	*****	i
Azure Tenant		i
Options	ColumnExtendedProperty=MS_Description;TableExtendedProperty=MS_Description;	i

« Previous Test Connect Connect » Advanced

49. SQL SERVER ANALYSIS SERVICES

49.1. Connection Options

49.1.1. Authentication

Property	Description
AuthScheme	The scheme used for authentication. Accepted entries are NTLM, Basic, Digest, None, and Negotiate.
URL	The URL used to connect to the SQL Server Analysis Services.
User	The SQL Server Analysis Services user account used to authenticate.
Password	The password used to authenticate the user.

49.1.2. Kerberos

Property	Description
KerberosKDC	The Kerberos Key Distribution Center (KDC) service used to authenticate the user.
KerberosRealm	The Kerberos Realm used to authenticate the user with.
KerberosSPN	The service principal name (SPN) for the Kerberos Domain Controller.
KerberosKeytabFile	The Keytab file containing your pairs of Kerberos principals and encrypted keys.

49.1.3. SSL

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.
SSLClientCertSubject	The subject of the TLS/SSL client certificate.
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

49.1.4. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

49.1.5. Proxy

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.
ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.
ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

49.1.6. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).

MaxLogFileCount	A string specifying the maximum file count of log files.

49.1.7. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.

49.1.8. Caching

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .
CacheDriver	The database driver to be used to cache data.
CacheConnection	The connection string for the cache database. This property is always used in conjunction with CacheProvider . Setting both properties will override the value set for CacheLocation for caching data.
CacheLocation	Specifies the path to the cache when caching to a file.
CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache .

Offline	Use offline mode to get the data from the cache instead of the live source.
CacheMetadata	This property determines whether or not to cache the table metadata to a file store.

49.1.9. Miscellaneous

Property	Description
BatchSize	The maximum size of each batch operation to submit.
Catalog	The Analysis Services catalog to use. This may also be known as a Database from within Analysis Services.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the SQL Server Analysis Services when the connection is opened.
CustomHeaders	Other headers as determined by the user (optional).
ExtraProperties	Additional properties to submit on each MDX request to SQL Server Analysis Services.
IncludeJoinColumns	Set this to true to include extra join columns on each table.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
Other	These hidden properties are used only in specific use cases.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.

ResponseRowLimit	The number of response rows to allow before erroring. Set to 0 for no limit.
RTK	The runtime key used for licensing.
SplitMeasures	Set this to true to split Measures table into individual tables.
SupportEnhancedSQL	This property enhances SQL functionality beyond what can be supported through the API directly, by enabling in-memory client-side processing.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
UseConnectionPooling	This property enables connection pooling.
UseMDX	Set this to true to pass MDX queries to SQL Server Analysis Services as-is.

For more details, see <https://cdn.cdata.com/help/FYG/jdbc/>.

50. SYBASE

50.1. Sybase Version Support

The driver enables connectivity to Sybase Database (SAP Adaptive Server) through the TDS protocol. ASE Versions 12, 15, and 16 are supported. Connecting to Sybase SQL Anywhere or Sybase IQ are not supported in this driver.

50.2. Connection Options

50.2.1. Authentication

Property	Description
AuthScheme	The scheme used for authentication. Accepted entries are Password, Kerberos.
Server	The name of the server running Sybase Database.
Port	The port of the Sybase database.
Database	The name of the Sybase database.
User	The Sybase user account used to authenticate.
Password	The password used to authenticate the user.
Charset	Charset name to communicate with server.
UseSSL	This field sets whether SSL is enabled.
AlternateServers	This property allows you to specify multiple servers in addition to the one configured in Server and Port . Specify both a server name and port; separate servers with a comma.

50.2.2. Kerberos

Property	Description
KerberosKDC	The Kerberos Key Distribution Center (KDC) service used to authenticate the user.
KerberosRealm	The Kerberos Realm used to authenticate the user with.
KerberosSPN	The service principal name (SPN) for the Kerberos Domain Controller.
KerberosKeytabFile	The Keytab file containing your pairs of Kerberos principals and encrypted keys.
KerberosServiceRealm	The Kerberos realm of the service.
KerberosServiceKDC	The Kerberos KDC of the service.
KerberosTicketCache	The full file path to an MIT Kerberos credential cache file.

50.2.3. SSL

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.
SSLClientCertSubject	The subject of the TLS/SSL client certificate.
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

50.2.4. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

50.2.5. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

50.2.6. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.

50.2.7. Miscellaneous

Property	Description
BatchSize	The maximum size of each batch operation to submit.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the Sybase when the connection is opened.
EncryptPassword	This field sets whether password encryption is enabled.
IgnoreGroupNumber	Whether to return group number (like 'num') of procedure when list procedures.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.

MaxTDSPacketSize	The protocol sending and receiving buffer size. For the lower version, it's from 512 to 512. For the newer version, it's from 512 to 2048.
Other	These hidden properties are used only in specific use cases.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.
QueryPassthrough	This option passes the query to the Sybase server as is.
Readonly	You can use this property to enforce read-only access to Sybase from the provider.
RTK	The runtime key used for licensing.
SwitchMode	This property allows you to specify a switching mode to select a server from AlternateServers as the active server.
SwitchStrategy	This property allows you to specify a switching strategy to select a server from AlternateServers as the active server.
Timeout	A timeout for the provider.
UseConnectionPooling	This property enables connection pooling.

For more details, see <https://cdn.cdata.com/help/OYG/jdbc/>.

51. TERADATA

51.1. Prerequisite

The Teradata connector delivered by HOPEX is based on the native Teradata driver.

The native Teradata driver may already be installed with Teradata. If it is not the case, download it:

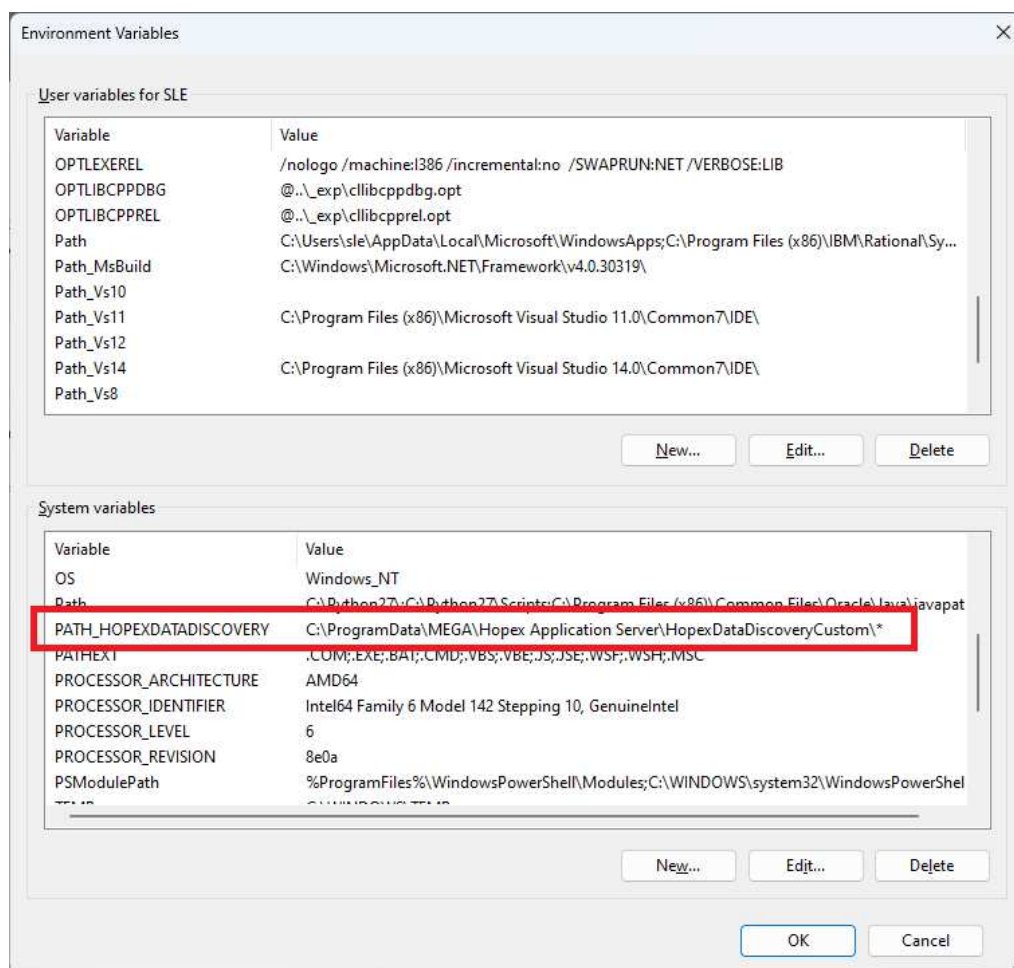
1> Download the terajdbc4.jar file from

<https://downloads.teradata.com/download/connectivity/jdbc-driver>

2> Put this file in a directory accessible from the HOPEX Data Discovery Module, but not the module directory.

For example: "C:\ProgramData\MEGA\Hopex Application Server\HopexDataDiscoveryCustom"

3> This directory must be filled in the environment variable "PATH_HOPEXDATADISCOVERY" which must be created if it does not exist:



51.2. Connection Options

51.2.1. Authentication

Property	Description
Port	Connects to the Teradata Database on the TCP/IP port specified. The default Teradata Database port is 1025.
User	The Teradata user account used to authenticate.
Password	The password used to authenticate the user.
Database	The database selected as the default database when a Teradata connection is opened.
DataSource	The Teradata server name or, equivalently, the DBC Name or TDPID.
Account	Specifies an account string to override the default account string defined for the Teradata Database user.
NewPassword	This connection parameter enables an application to change an expired password automatically.

51.2.2. Connection

Property	Description
ConnectFailureTTL	This option enables the CData ADO.NET Provider for Teradata remember the time of the last connection failure for each IP address/port combination. Also, the CData ADO.NET Provider for Teradata skips connection attempts to that IP address/port during subsequent logins for the number of seconds specified by the Connect Failure time-to-live (CONNECTFAILURETTL) value.
ConnectFunction	Specifies whether the Teradata Database should allocate a Logon Sequence Number (LSN) for this session or associate this session with an existing LSN.

51.2.3. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

51.2.4. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Log	Specifies the logging level (verbosity) for a connection. Logging is always enabled. The logging levels are listed in order from terse to verbose.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).
MaxLogFileCount	A string specifying the maximum file count of log files.

51.2.5. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.

51.2.6. Miscellaneous

Property	Description
BatchSize	The maximum size of each batch operation to submit.
Charset	Specifies the session character set for encoding and decoding character data transferred to and from the Teradata Database. The default value is ASCII.
ClientCharset	Specifies the Java character set for encoding and decoding character data transferred to and from the Teradata Database.
ColumnName	Controls the behavior of the ResultSetMetaData getColumnName and getColumnLabel methods.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the Teradata when the connection is opened.

Cop	Specifies whether COP Discovery is performed.
CopLast	Specifies how COP Discovery determines the last COP hostname.
Ddstats	Specify the value for DDSTATS.
DisableAutoCommitInBatch	Specifies whether or not disable the autocommit when executing the batch operation.
EncryptData	Specify the EncryptData value, ON or OFF.
ErrorQueryCount	Specifies the maximum number of times that JDBC FastLoad will attempt to query FastLoad Error Table 1 after a JDBC FastLoad operation.
ErrorQueryInterval	Specifies the number of milliseconds that JDBC FastLoad will wait in between attempts to query FastLoad Error Table 1 after a JDBC FastLoad operation.
ErrorTable1Suffix	Specifies the suffix for the name of FastLoad Error Table 1 created by JDBC FastLoad and JDBC FastLoad CSV.
ErrorTable2Suffix	Specifies the suffix for the name of FastLoad Error Table 2 created by JDBC FastLoad and JDBC FastLoad CSV.
ErrorTableDatabase	Specifies the database name for the FastLoad error tables created by JDBC FastLoad and JDBC FastLoad CSV.
FieldSep	Specifies a field separator for use with JDBC FastLoad CSV only. The default separator is ',' (comma).
FinalizeAutoClose	Specify the value for FinalizeAutoClose, ON or OFF.
GeturlCredentials	Specify the value for GeturlCredentials, ON or OFF.
Govern	Specify the value for GOVERN, ON or OFF.
LiteralUnderscore	Automatically escape LIKE-predicate patterns in DatabaseMetaData calls, such as schemPattern and tableNamePattern.
LobSupport	Specify the value for LobSupport, ON or OFF.

LobTempTable	Specifies the name of a table with the following columns: id integer, bval blob, cval clob.
LogData	Specifies additional data needed by a logon mechanism, such as a secure token, Distinguished Name, or a domain/realm name.
LogMech	Specifies the Logon Mechanism, which determines the connection's authentication and encryption capabilities.
LogonSequenceNumber	Specifies an existing Logon Sequence Number (LSN) to associate this session with.
MaxMessageBody	Specifies the maximum Response Message size in bytes.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
MaybeNull	Controls the behavior of the ResultSetMetaData.isNullable method.
Other	These hidden properties are used only in specific use cases.
Partition	Specifies the Teradata Database partition for the Connection.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.
PrepSupport	Specifies whether the Teradata Database performs a prepare operation when a PreparedStatement or CallableStatement is created.
QueryPassthrough	This option passes the query to the Teradata server as is.
Readonly	You can use this property to enforce read-only access to Teradata from the provider.

ReconnectCount	Enables Teradata Session Reconnect. Specifies the maximum number of times that the Teradata JDBC Driver will attempt to reconnect the session.
ReconnectInterval	Enables Teradata Session Reconnect. Specifies the number of seconds that the Teradata JDBC Driver will wait in between attempts to reconnect the session.
Redrive	Enables Teradata Session Reconnect, and also enables automatic redriving of SQL requests interrupted by database restart.
RTK	The runtime key used for licensing.
RunStartup	Specify the value for RunStartup, ON or OFF.
Sessions	Specifies the number of FastLoad or FastExport connections to be created, where $1 \leq \text{number of FastLoad or FastExport connections} \leq \text{number of AMPs}$.
SipSupport	Controls whether the Teradata Database and Teradata JDBC Driver use StatementInfo Parcel (SIP) to convey metadata.
SlobReceiveThreshold	Controls how small LOB values are received from the Teradata Database. Small LOB values are pre-fetched from the Teradata Database before the application explicitly reads data from Blob/Clob objects.
SlobTransmitThreshold	Controls how small LOB values are transmitted to the Teradata Database.
SpSpl	Specifies behavior for creating or replacing Teradata stored procedures.
StrictEncode	Specifies behavior for encoding character data to transmit to the Teradata Database.
Tcp	Specifies one or more TCP socket settings, separated by plus signs (" + ").
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.

TMode	Specifies the transaction mode for the connection.
TNano	Specifies the fractional seconds precision for all java.sql.Time values bound to a PreparedStatement or CallableStatement and transmitted to the Teradata Database as TIME or TIME WITH TIME ZONE values.
TrustedSql	Specify the value for TrustedSql.
TSNano	Specifies the fractional seconds precision for all java.sql.Timestamp values bound to a PreparedStatement or CallableStatement and transmitted to the Teradata Database as TIMESTAMP or TIMESTAMP WITH TIME ZONE values.
Type	Specifies the type of protocol to be used with the Teradata Database for SQL statements.
UpperCaseIdentifiers	This property reports all identifiers in uppercase. This is the default for Oracle databases and thus allows better integration with Oracle tools such as the Oracle Database Gateway.
UseConnectionPooling	This property enables connection pooling.
UseXViews	Specifies which Data Dictionary views should be queried to return result sets from DatabaseMetaData methods.

For more details, see <https://cdn.cdata.com/help/OTG/jdbc/>.

52. XML

52.1. Connection Options

52.1.1. Authentication

Property	Description
AuthScheme	The type of authentication to use when connecting to remote services.
AccessKey	Your account access key. This value is accessible from your security credentials page.
SecretKey	Your account secret key. This value is accessible from your security credentials page.
ApiKey	The API Key used to identify the user to IBM Cloud.
User	The XML user account used to authenticate.
Password	The password used to authenticate the user.
SharePointEdition	The edition of SharePoint being used. Set either SharePointOnline or SharePointOnPremise.

52.1.2. Connection

Property	Description
URI	The Uniform Resource Identifier (URI) for the XML resource location.
XPath	The XPath of an element that repeats at the same height within the XML document (used to split the document into multiple rows).

DataModel	Specifies the data model to use when parsing XML documents and generating the database metadata.
XMLFormat	Specifies the format of the XML document.
Region	The hosting region for your S3-like Web Services.
ProjectId	The id of the project where your Google Cloud Storage instance resides.
OracleNamespace	The Oracle Cloud Object Storage namespace to use.
StorageBaseURL	The URL of a cloud storage service provider.
SimpleUploadLimit	This setting specifies the threshold, in bytes, above which the provider will choose to perform a multipart upload rather than uploading everything in one request.
UseVirtualHosting	If true (default), buckets will be referenced in the request using the <code>hosted-style</code> request: <code>http://yourbucket.s3.amazonaws.com/yourobject</code> . If set to false, the bean will use the <code>path-style</code> request: <code>http://s3.amazonaws.com/yourbucket/yourobject</code> . Note that this property will be set to false, in case of an S3 based custom service when the CustomURL is specified.

52.1.3. AWS Authentication

Property	Description
AWSAccessKey	Your AWS account access key. This value is accessible from your AWS security credentials page.
AWSSecretKey	Your AWS account secret key. This value is accessible from your AWS security credentials page.
AWSRoleARN	The Amazon Resource Name of the role to use when authenticating.
AWSPrincipalArn	The ARN of the SAML Identity provider in your AWS account.

AWSRegion	The hosting region for your Amazon Web Services.
AWSSessionToken	Your AWS session token.
MFASerialNumber	The serial number of the MFA device if one is being used.
MFAToken	The temporary token available from your MFA device.

52.1.4. Azure Authentication

Property	Description
AzureStorageAccount	The name of your Azure storage account.
AzureAccessKey	The storage key associated with your XML account.
AzureSharedAccessSignature	A shared access key signature that may be used for authentication.
AzureTenant	The Microsoft Online tenant being used to access data. If not specified, your default tenant will be used.
AzureEnvironment	The Azure Environment to use when establishing a connection.

52.1.5. SSO

Property	Description
SSOLoginURL	The identity provider's login URL.
SSOProperties	Additional properties required to connect to the identity provider in a semicolon-separated list.

52.1.6. OAuth

Property	Description
InitiateOAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.
OAuthVersion	The version of OAuth being used.
OAuthClientId	The client ID assigned when you register your application with an OAuth authorization server.
OAuthClientSecret	The client secret assigned when you register your application with an OAuth authorization server.
OAuthAccessToken	The access token for connecting using OAuth.
OAuthAccessTokenSecret	The OAuth access token secret for connecting using OAuth.
OAuthSettingsLocation	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH. Alternatively, this can be held in memory by specifying a value starting with memory://.
CallbackURL	The OAuth callback URL to return to when authenticating. This value must match the callback URL you specify in your app settings.
OAuthGrantType	The grant type for the OAuth flow.
OAuthPasswordGrantMode	How to pass Client ID and Secret with OAuthGrantType is set to Password.
OAuthIncludeCallbackURL	Whether to include the callback URL in an access token request.
OAuthAuthorizationURL	The authorization URL for the OAuth service.
OAuthAccessTokenURL	The URL to retrieve the OAuth access token from.

OAuthRefreshTokenURL	The URL to refresh the OAuth token from.
OAuthRequestTokenURL	The URL the service provides to retrieve request tokens from. This is required in OAuth 1.0.
OAuthVerifier	The verifier code returned from the OAuth authorization URL.
AuthToken	The authentication token used to request and obtain the OAuth Access Token.
AuthKey	The authentication secret used to request and obtain the OAuth Access Token.
OAuthParams	A comma-separated list of other parameters to submit in the request for the OAuth access token in the format paramname=value.
OAuthRefreshToken	The OAuth refresh token for the corresponding OAuth access token.
OAuthExpiresIn	The lifetime in seconds of the OAuth AccessToken.
OAuthTokenTimestamp	The Unix epoch timestamp in milliseconds when the current Access Token was created.

52.1.7. JWT OAuth

Property	Description
OAuthJWTCert	The JWT Certificate store.
OAuthJWTCertType	The type of key store containing the JWT Certificate.
OAuthJWTCertPassword	The password for the OAuth JWT certificate.
OAuthJWTCertSubject	The subject of the OAuth JWT certificate.
OAuthJWTIssuer	The issuer of the Java Web Token.

OAuthJWTSubject	The user subject for which the application is requesting delegated access.

52.1.8. Kerberos

Property	Description
KerberosKDC	The Kerberos Key Distribution Center (KDC) service used to authenticate the user.
KerberosRealm	The Kerberos Realm used to authenticate the user with.
KerberosSPN	The service principal name (SPN) for the Kerberos Domain Controller.
KerberosKeytabFile	The Keytab file containing your pairs of Kerberos principals and encrypted keys.
KerberosServiceRealm	The Kerberos realm of the service.
KerberosServiceKDC	The Kerberos KDC of the service.
KerberosTicketCache	The full file path to an MIT Kerberos credential cache file.

52.1.9. SSL

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.
SSLClientCertSubject	The subject of the TLS/SSL client certificate.

SSLMode	The authentication mechanism to be used when connecting to the FTP or FTPS server.
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

52.1.10. SSH

Property	Description
SSHAuthMode	The authentication method to be used to log on to an SFTP server.
SSHClientCert	A certificate to be used for authenticating the user.
SSHClientCertPassword	The password of the SSHClientCert certificate if it has one.
SSHClientCertType	The type of SSHClientCert certificate.

52.1.11. Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

52.1.12. Proxy

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.
ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.
ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

52.1.13. Logging

Property	Description
Logfile	A filepath which designates the name and location of the log file.
Verbosity	The verbosity level that determines the amount of detail included in the log file.
LogModules	Core modules to be included in the log file.
MaxLogFileSize	A string specifying the maximum size in bytes for a log file (for example, 10 MB).

MaxLogFileCount	A string specifying the maximum file count of log files.

52.1.14. Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
BrowsableSchemas	This property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas=SchemaA,SchemaB,SchemaC.
Tables	This property restricts the tables reported to a subset of the available tables. For example, Tables=TableA,TableB,TableC.
Views	Restricts the views reported to a subset of the available tables. For example, Views=ViewA,ViewB,ViewC.
PushAttributes	Set PushAttributes to true to push any identified attributes as columns.
FlattenArrays	By default, nested arrays are returned as strings of XML. The FlattenArrays property can be used to flatten the elements of nested arrays into columns of their own. Set FlattenArrays to the number of elements you want to return from nested arrays.
FlattenObjects	Set FlattenObjects to true to flatten object properties into columns of their own. Otherwise, objects nested in arrays are returned as strings of XML.
QualifyColumns	Controls whether the provider will use relative column names.

52.1.15. Caching

Property	Description
AutoCache	Automatically caches the results of SELECT queries into a cache database specified by either CacheLocation or both of CacheConnection and CacheProvider .
CacheDriver	The database driver to be used to cache data.
CacheConnection	The connection string for the cache database. This property is always used in conjunction with CacheProvider . Setting both properties will override the value set for CacheLocation for caching data.
CacheLocation	Specifies the path to the cache when caching to a file.
CacheTolerance	The tolerance for stale data in the cache specified in seconds when using AutoCache .
Offline	Use offline mode to get the data from the cache instead of the live source.
CacheMetadata	This property determines whether or not to cache the table metadata to a file store.

52.1.16. Miscellaneous

Property	Description
BackwardsCompatibilityMode	Set BackwardsCompatibilityMode to true to use the XML functionality and features available in the 2017 version.
BatchSize	The maximum size of each batch operation to submit.

Charset	Specifies the session character set for encoding and decoding character data transferred to and from the XML file. The default value is UTF-8.
ClientCulture	This property can be used to specify the format of data (e.g., currency values) that is accepted by the client application. This property can be used when the client application does not support the machine's culture settings. For example, Microsoft Access requires 'en-US'.
ConnectionLifeTime	The maximum lifetime of a connection in seconds. Once the time has elapsed, the connection object is disposed.
ConnectOnOpen	This property species whether to connect to the XML when the connection is opened.
Culture	This setting can be used to specify culture settings that determine how the provider interprets certain data types that are passed into the provider. For example, setting Culture='de-DE' will output German formats even on an American machine.
CustomHeaders	Other headers as determined by the user (optional).
CustomUrlParams	The custom query string to be included in the request.
ExcludeFiles	Comma-separated list of file extensions to exclude from the set of the files modeled as tables.
FlattenRowLimit	The maximum number of rows that can result from a single flattened element.
GenerateSchemaFiles	Indicates the user preference as to when schemas should be generated and saved.
IncludeDropboxTeamResources	Indicates if you want to include Dropbox team files and folders.
IncludeFiles	Comma-separated list of file extensions to include into the set of the files modeled as tables.

MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
MetadataDiscoveryURI	Used when aggregating multiple files into one table, this property specifies a specific file to read to determined the aggregated table schema.
Other	These hidden properties are used only in specific use cases.
Pagesize	The maximum number of results to return per page from XML.
PathSeparator	Determines the character which will be used to replace the file separator.
PoolIdleTimeout	The allowed idle time for a connection before it is closed.
PoolMaxSize	The maximum connections in the pool.
PoolMinSize	The minimum number of connections in the pool.
PoolWaitTime	The max seconds to wait for an available connection.
PseudoColumns	This property indicates whether or not to include pseudo columns as columns to the table.
Readonly	You can use this property to enforce read-only access to XML from the provider.
RowScanDepth	The number of rows to scan when dynamically determining columns for the table.
RTK	The runtime key used for licensing.
SupportEnhancedSQL	This property enhances SQL functionality beyond what can be supported through the API directly, by enabling in-memory client-side processing.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.

TypeDetectionScheme	Determines how to determine the data types of columns.
URISeparator	A delimiter used to separate different values in the URI property.
UseConnectionPooling	This property enables connection pooling.

For more details, see <https://cdn.cdata.com/help/DVG/jdbc/>.