HOPEX Business Data

User Guide

HOPEX V2
1st Edition (July 2016)

INTRODUCTION TO HOPEX LOGICAL DATA

The **HOPEX Logical Data** solution is integrated in the more global Information Architecture offer. This offer brings together the three solutions **HOPEX Information Architecture**, **HOPEX Logical Data** and **HOPEX Database Builder**. It is used to build the global architecture, from business data definition to database design. It ensures the traceability of data between three different levels: conceptual, logical and physical.

Business concepts are defined in the **HOPEX Information Architecture** solution.

These concepts can then be connected to logical data models created in **HOPEX Logical Data** to define the information system. Company organizers and architects can describe operations using **HOPEX Logical Data** by modeling data used when implementing business processes and applications. To this end, **HOPEX Logical Data** makes available a number of notations.

Using logical data models, **HOPEX Database Builder** builds corresponding physical models, that is, creates database tables, with its columns, indexes and keys as well as the relational diagram drawings.

The following points are covered here:

CONNECTING TO HOPEX LOGICAL DATA

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

Connecting to the solution

To connect to **HOPEX Logical Data**, see HOPEX Common Features, "HOPEX Desktop", "Accessing (HOPEX)".

HOPEX Logical Data Profiles

There are default user profiles with which rights and accesses are associated. These profiles define the objects displayed on the desktop, according to each business.

Two profiles are used to connect to **HOPEX Logical Data**:

- **Data architect** : he has read and write access to logical data. The data architect is responsible for defining the logical view (data models, packages, classes, etc.) of the system.
  During integration with **HOPEX Information Architecture**, the data architect has reading access to business information and ensures their connection with the objects in the logical view.
- **IA Functional Administrator**: the IA Functional Administrator is responsible for standard administration functions, such as user management. The IA Functional Administrator has rights on all objects in the repository.

Desktop Presentation

Once connected to **HOPEX Logical Data**, the "Information Architecture" desktop appears.

The "Information Architecture" desktop is the **HOPEX Logical Data** workspace, in common with the **HOPEX Information Architecture** and **HOPEX Database Builder** solutions.

The elements presented in the interface depend on solutions available to you as well as the profile with which you are connected. Thus the "Information Architecture" is based on three products that cover three modeling levels:

Modeling level	Associated product	Details
Business function	HOPEX Information Architecture	Definition of the business vocabulary Realization of concepts
Logical	HOPEX Logical Data	Modeling of logical data Realization of concepts by entities and classes
Physical	HOPEX Database Builder	Modeling/generating of physical data Synchronization of logical and physical models

Presentation of space common to all roles

All users have access to the "Information Architecture" desktop and access to the following panes:

- **Home, Dashboard**, **List of tasks** and **Reports** that are common to all **HOPEX** solution users.
- **Environment**: pane used to organize and access all library elements easily,

Data architect space

In addition to the panes offered in standard mode to all "Information Architecture" desktop users, the data architect has access to the **Logical data** pane that gives access to the following folders:

- **DataType Packages**,
- **Packages**,
- **Data areas**,
- **Data views**.

> ☛ *During integration with the **HOPEX Information Architecture solution**, the data architect also has access to the **Business information** pane.*

IA functional administrator workspace

In addition to the panes offered in standard mode to all "Information Architecture" desktop users, the IA Functional Administrator has access to the following navigation panes:

- **Logical data** that gives the IA Functional Administrator access to the it of logical data in the following folders:
  - **DataType Package**,
  - **Package**,
  - **Data areas**,
  - **Data views**,
  - **Databases**,
  - **Tools** from the logical view.
- **Administration**, particularly for user management.

> ☛ *During integration with the **HOPEX Information Architecture solution,** the data architect also has access to the **Business information** pane.*
> *During integration with **HOPEX Database Builder**, the IA Function Administrator also has access to the **Physical data** pane.*

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 navigation tree, expand the **Data Modeling** 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.

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 navigation tree, expand the **Data Modeling** folder.
3. Click **Data Notation**.
4. In the right part of the window, select the notations that you want to display.
5. Click **OK**.

ABOUT THIS GUIDE

Guide Structure

The **HOPEX Logical Data** guide covers the following points:

- "The data model", page 21
- "Other Notations", page 51
- "Attribute Types", page 95

Additional Resources

This guide is supplemented by:

- the **HOPEX Common Features Common Features** guide, which describes the Web interface and tools specific to MEGA solutions.

  ☛ *It can be useful to consult this guide for a general presentation of the interface.*

- the **HOPEX Power Supervisor** administration guide.

Conventions Used in the Guide

Styles and formatting

☛ *Remark on the preceding points.*

📖 *Definition of terms used.*

☺ *A tip that may simplify things.*

Compatibility with previous versions.

Things you must not do.

Very important remark to avoid errors during an operation.

Commands are presented as seen here: **File > Open**.

Names of products and technical modules are presented in bold as seen here: **HOPEX**.

DATA ARCHITECTURE

HOPEX Logical Data provides a set of tools for modeling logical data in Class/Association formalism.

With **HOPEX Logical Data** , you can define how the data of an organization will be used by the processes and the applications that it uses. Using data domain concepts and data views, you can detail a logical data structure in a particular use context.

During integration with **HOPEX Information Architecture**, it is possible to associate the entities and the classes with concepts created during the semantic analysis phase.

The following points are covered here:

ACCESSING LOGICAL DATA

Access to logical data in the repository is reserved for the data architect.

Display the navigation pane of logical data

To access repository logical data:

1. In the "Information Architecture" desktop, click on the navigation menu, then **Logical data**.
   The navigation pane shows the corresponding object types:
   - Data packages
   - Data models (if the corresponding option is checked)
   - Data areas
   - Data views

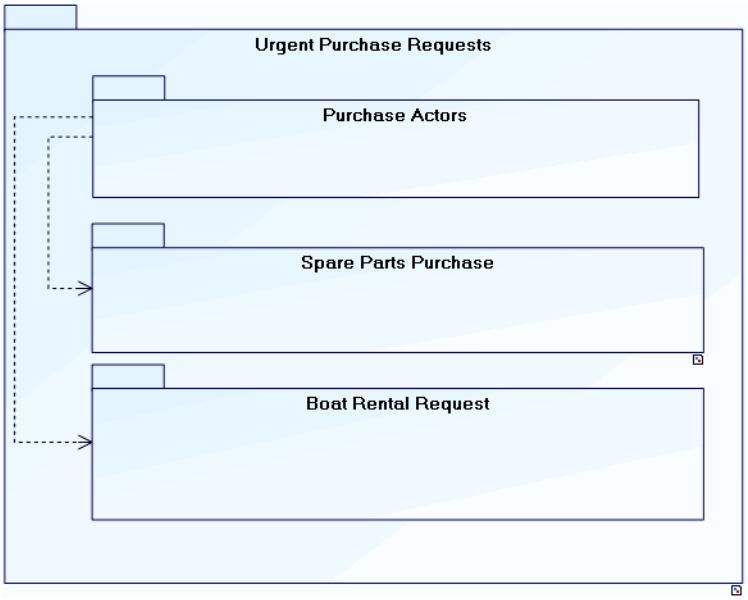
The different concepts in the logical view

The package notion

To clearly organize information relating to logical data, **HOPEX** uses *package* mechanisms.

> 🕮 *A package partitions the domain studied and the associated work. It enables grouping of various elements, in particular use cases and classes. A package can also contain other packages. Packages are interconnected through contractual reports defining their interface.*

The package allows you to classify elements referenced in a project. You can create sub-packages in a package to classify objects in finer detail, for example actors of a project




```
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 UML** guide.*

Class diagrams are used to represent graphically elements from a package.

Data models

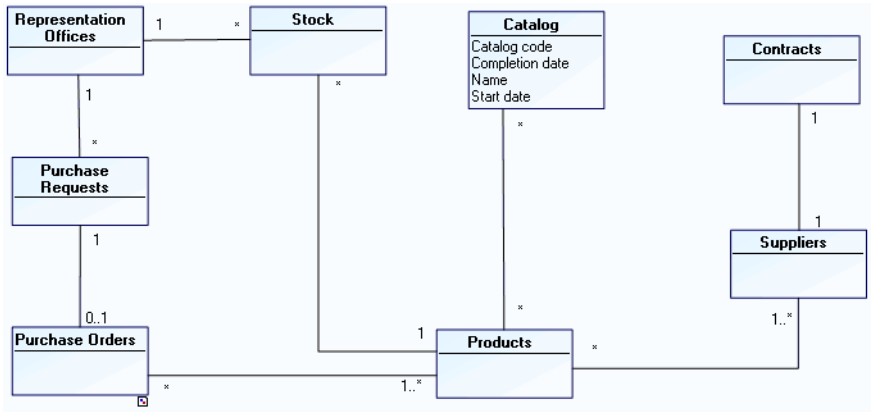
A data model is used to represent the static structure of a system, particularly the types of objects manipulated in the system, their internal structure, and the relationships between them.

Data diagrams are used to represent graphically elements from a data model.

For more details on creating and updating a data model, see "The data model", page 21.

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.

☛ *For more details on creating and updating a data model, see "Modeling data".*

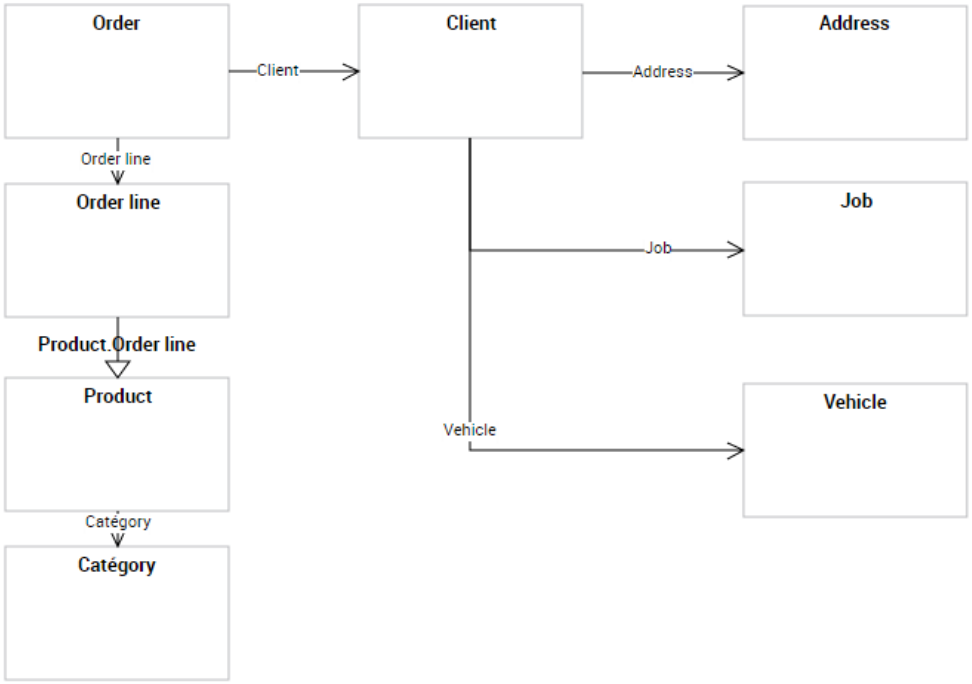
Data areas

A Data Area represents a restricted data structure dedicated to the description of a software Data Store (see "The data store concept", page 15 in SOIA). It is made of classes and/or data views and can be described in a Data Area Diagram.

For more details, see "Logical data areas", page 12.

Example

The following data area diagram represents a data structure relating to Orders; it describes classes and their relationships in a Whole/Part formalism.



To address these specific use cases, you can create Data Views in which you can see and modify the scope covered by the classes.

See "Logical data areas", page 12.

Logical data view

A data view represents the scope covered by an element of a data model or a data area. A data view is based on the selection of several classes connected in the specific context of the view. See "Data views", page 15.

LOGICAL DATA AREAS

A logical data area is used to define a logical data structure made up of classes and data views.

A logical data domain is owned by a package and references objects held in other packages.

You can define the access mode (creation, deletion, etc.) to the objects referenced by a data area by integrating them as components of the data area.

During integration with **HOPEX Database Builder**, a corresponding physical structure can be defined via a physical data area. It is made up of tables and table views.

Creating a logical data area

To create a logical data area:

1. Click on the navigation menu, then on **Logical Data**.
2. In the navigation pane, click **All logical data areas**.
   The list of logical data areas appears.
3. Click the **New** button.
   The creation dialog box opens.
4. Enter the name of the data area.
5. If appropriate, enter the package owner.
6. Click **OK**.
   The data area appears in the list.

The Data Area Diagram

A logical data area can be described by a data area diagram. See "Creating a data area diagram", page 13.

A data area diagram defines classes and their relationships in a Whole/Part formalism in connection with the subject of the data area (logical or physical) described.

You can connect one or more data area diagrams to a logical data area, according to what you want to describe.

The structure of a data area must be explicitly stated via the creation of its components (classes or class views) See "Creating a component", page 13.

Creating a data area diagram

To create a data area diagram from a logical data area:

- Right-click on the logical data area and select **New** > **Data Area Diagram**.

Adding an object

In the data area diagram, you can add a new object or connect an existing object.

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 "Creating a data view", page 15.

Creating a component

To define the structure of a logical data area, you must specify its components (classes or class views)

You can create a component from an object in the diagram or in the properties window of the data area.

To create a component from an object of the data area diagram:

- In the diagram, right-click the object in question and select **Add to (name of the data area)**.
  The name of the component created appears in the properties of the data area. By default it has the name of the object that it references.

Defining object access mode

On the component, you can define the access mode to the object referenced (creation, read-only, deletion, etc.).

To define the access mode to the object in the data area:

1. Open the properties data domain.
2. Click the drop-down list then **Components**.
   The list of components of the data area appears.
3. Select the component.

4. For each column (Creation, Read-only, etc.), select the corresponding check box.

DATA VIEWS

A data view enables representation of the scope covered by a data model element. A data view is based on the selection of several classes connected in the specific context of the view.

Creating a data view

Creating a data view consists of:

- defining the objects concerned by the view
- defining more precisely parts of these objects to be taken into account in the view. Using the objects contained in the view, you can create embedded components and referenced components.
  An embedded component allows you to use all the information that makes up the object of the view. A reference component references only the object of the view.

You can create a view:

- from the **Logical data** > **Data view** navigation pane
- in a data area diagram
- directly on an object.

View example

For Order management, you must retrieve the delivery address available for each Client. To take into account only this information, you will create a view of the Client class that uses only the delivery address.

Creating a data view directly on an object

To create a data view on an object:

1. Right-click the object concerned and select **New** > **Data view**.

   ```
   In our example, right-click the client class.
   ```

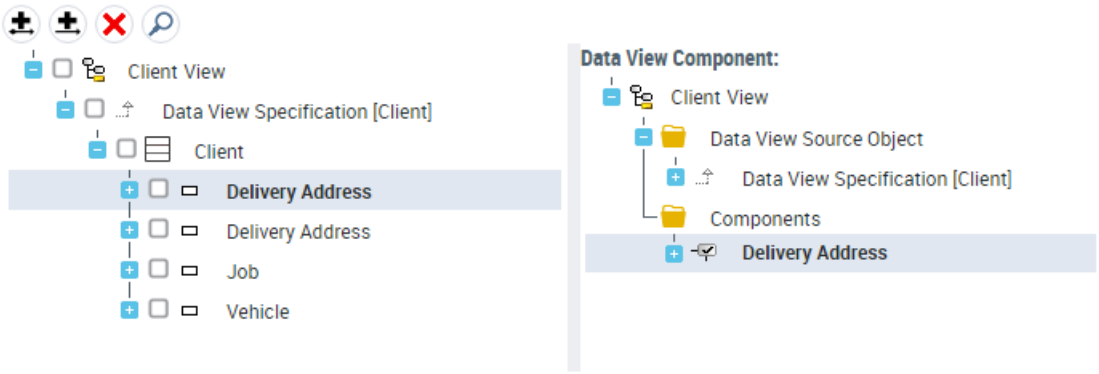
   The data view creation wizard opens.
2. Enter the name of the view.
3. If appropriate, enter the name of the owner.
4. Click **OK**.
   The editor view appears.
   To the left, the class concerned by the view (Client) appears.
   To the right, you will define the components of the view, in other words, the information to be displayed (here the delivery address of the Client).
5. To the left of the editor, select the element to take into account.

   ```
   In our example select "Delivery address".
   ```

6. Click **Add an embedded component** or **Add a reference component** as needed.

   ☛ *Unlike the reference component, the embedded component encompasses the entire perimeter of the selected object.*

   The component appears to the right, in the definition of the view.



CONNECT THE BUSINESS CONCEPTS TO THE LOGICAL DATA

The business concepts defined in **HOPEX Information Architecture** can be implemented in the IS using the UML method and formalism.

The "Concept realization" work therefore 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.

Creating a concept realization

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

To associate a logical view object to a dictionary element, you must create a realization.

To associate a class with a concept, for example:

1. In the Information Architecture desktop, click on the navigation menu, then **Logical data**.
2. In the navigation pane, expand the "**Packages**" folder.
3. Click **Package Hierarchy**.
   The list of packages in the repository appears in the edit window.
4. Expand the package folder that interests you.
5. Select the class that you want to connect to a concept and open its properties.
6. Select the **Characteristics >Realization** page.
7. Click **New**.
   The **Choice of MetaClass** dialog box opens.
8. Select "Realization of Concept" and click **OK**.
   The creation of page opens.
9. In the **Dictionary Entity Realized** field, click **Connect**.
10. In the selection window, select "Concept" and click **Find**.
11. Select the concept that interests you and click **Connect**.
12. In the concept realization creation window, click **OK**.
    The concept realization appears in the properties window with the name of the concept selected.

Realization report

You can use the realization report to view the realization (or implementation) coverage of dictionary elements with another architecture element.

To access the report:

1. Click the navigation menu, then **Reports**.
2. In the navigation pane, select **Realization report**.

Report parameters

This consists of defining report input data.

Parameters	Parameter type	Constraints
List of objects	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 mandatory.
Responsibles display mode	Show responsibles Show responsibles and percentages Hide responsibles	
Indicators display mode	Boolean	

Report example

The example below enables viewing of the coverage rate of objects specified as parameters.

Note that realizations of structural components of concepts specified as parameters are also displayed.

Covered
Covered by multiple objects
Not Covered

Consultant	Consulting Organization	Training	Training certification	Training Session
Realizer Consultant 100%	**Realizer** Consulting Compagny 100% **Realizer** IT Subsidary 100%	**Realizer** Training 100%		
	Consultant Consultant		Certified consultant	Trainer
	Delivered certification		Granted training	Delivered training
	Proposed training Delivered training Mission			End
				Training occurrence (EN)

THE DATA MODEL

To help you describe logical data, **HOPEX** offers a simple notation to represent all current cases based on the data model.

A data model is used to represent the static structure of a system, particularly the types of objects handled in the system, their internal structure, and the relationships between them.

With **HOPEX Database Builder**, data models can be mapped with physical models.

The following points are covered here:

DATA MODELING PRINCIPLES

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.

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

For the HOPEX Windows Front-End version, see the guide in .pdf format delivered on the Support site.

Creating a Data Model

Use of data models requires selection of an option. See "Data Modeling Options", page 4.

To create a data model:

1. On the desktop, click on the **Logical data** navigation pane.
2. Display the list of **Data Models**.
   The list of data models appears in the edit area.
3. In the edit area, click **New**.
   The data model mapping creation dialog box opens.
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.

To create a data diagram:

- 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 "Datatype Packages", page 96.

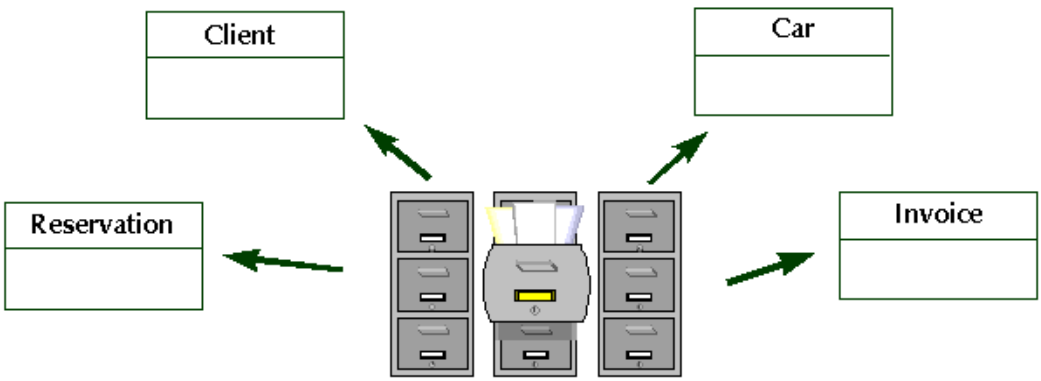
ENTITIES

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
```

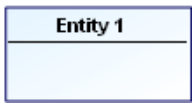
Creating an entity

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**.

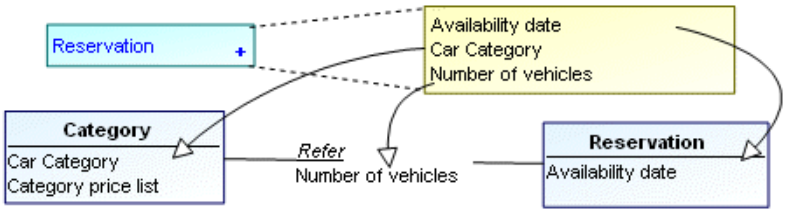
   ☛ *When the **OK** or **Create** buttons are grayed, this is because the requirements for the dialog box in which they appear have not been completed.*
4. Click **Add**.
   The entity appears in the diagram.



☺ *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.

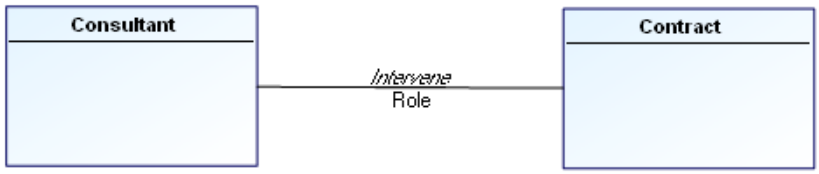
📖 *An attribute is the most basic data saved in the enterprise information system. An attribute is a property when it describes an entity or association, and an identifier when selected as a means of identification of each instance of an entity.*

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 "Attribute Types", page 95 for more details on data types that can be assigned to an attribute.*

Inherited attributes

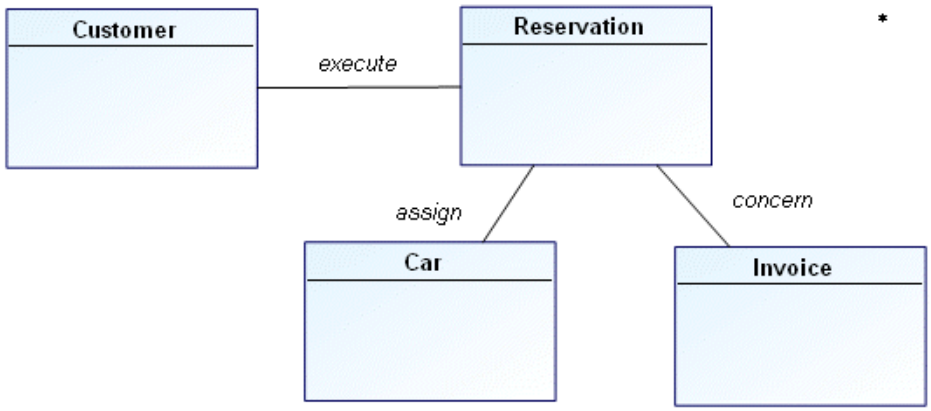
When a generalization exists between a general entity and a more specialized entity, the specialized entity inherits the attributes of the general entity.

See "Generalizations", page 40.

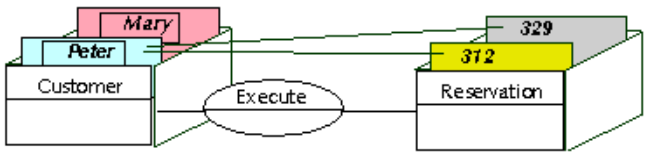
ASSOCIATIONS

📖 *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.



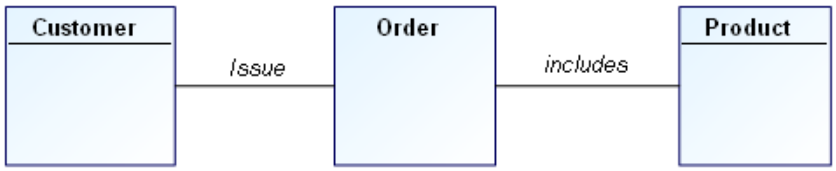
Peter and Mary are clients. Peter has made reservations numbers 312 and 329.

A data diagram should be able to store all situations in the context of the company, but these situations only.

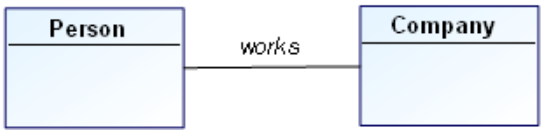
☛ *The diagram should not allow representing unrealistic or aberrant situations.*

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.

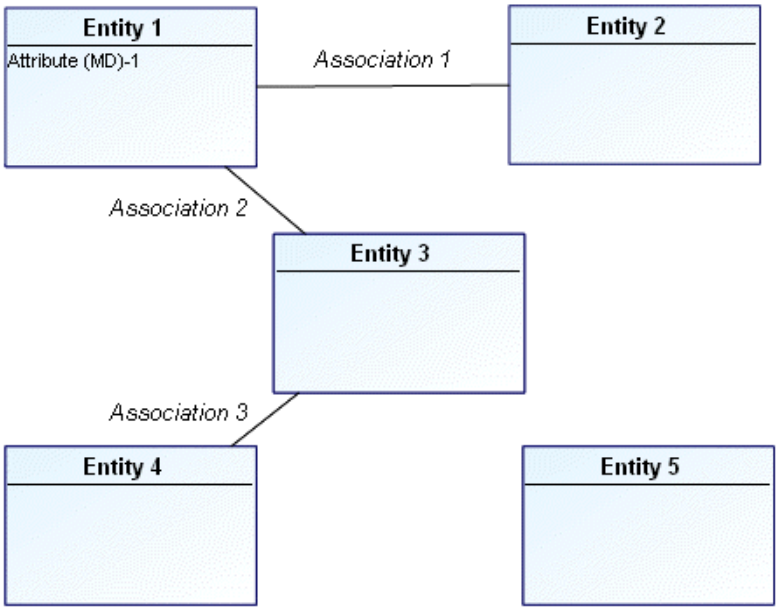
Creating an Association

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



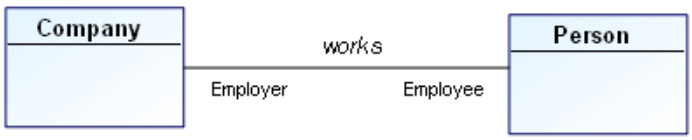
You can also delete an element or link you created in error by right-clicking it and selecting **Delete**.

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.

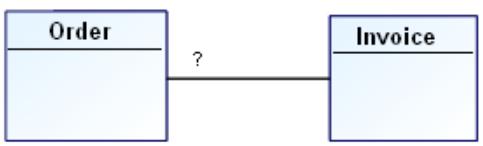
Multiplicity specifies the minimum and maximum number of instances of an entity that can be linked by the association to each instance of the other entity.

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.

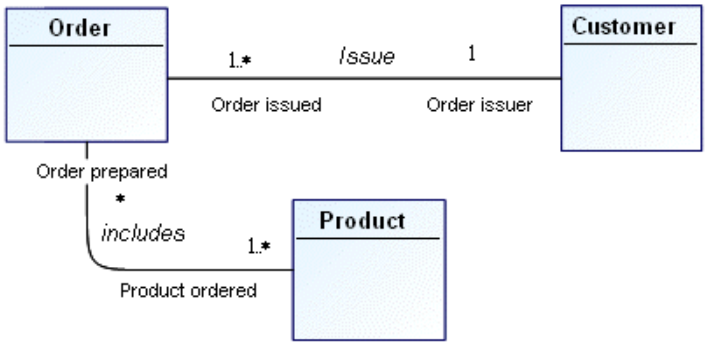
1	One and one only
0 / 1	Zero or one
M..N	From M to N (natural integer)
*	From zero to several
0..*	From zero to several
1..*	From one to several

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:

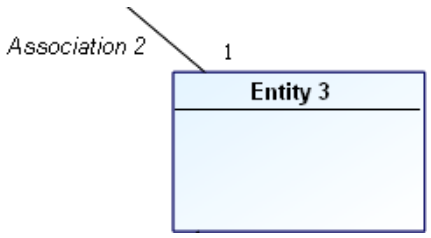


1..*	A client can issue one or more orders.
1	An order is issued by one and only one client.
1..*	An order contains one or more products.
*	A product can be contained in any number of orders, including no orders.
0 / 1	A person works for a company.
1..*	An alarm is triggered by one or more sensors.
1	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.



☛ *In MEGA 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.*

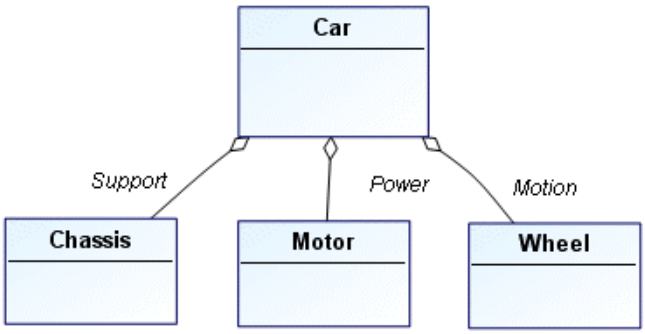
Other association characteristics

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.

> ☺ *In HOPEX Windows Front-End you can specify aggregation directly from role's pop-up menu.*

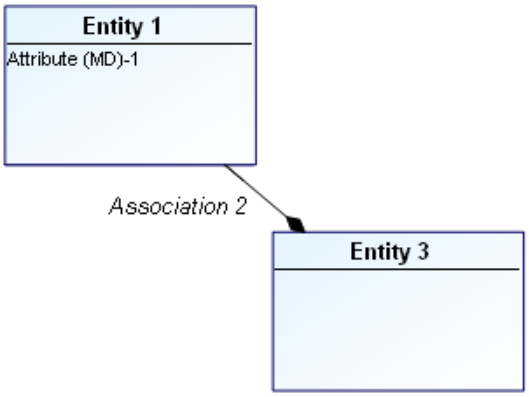
Composition

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.



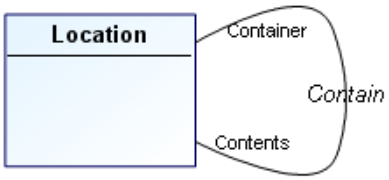
To specify composition of a role:

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".

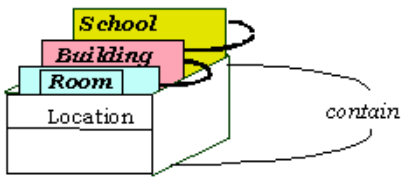
☺ *In HOPEX Windows Front-End you can specify composition directly from role's pop-up menu.*

Using reflexive associations

Certain associations use the same entity several times.



A classroom, a building, and a school are all locations.



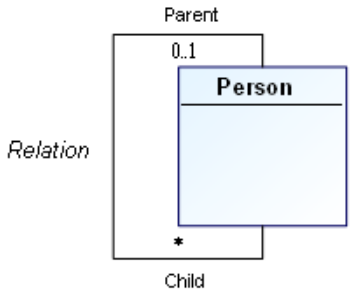
A classroom is contained in a building, which is contained in a school.

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.

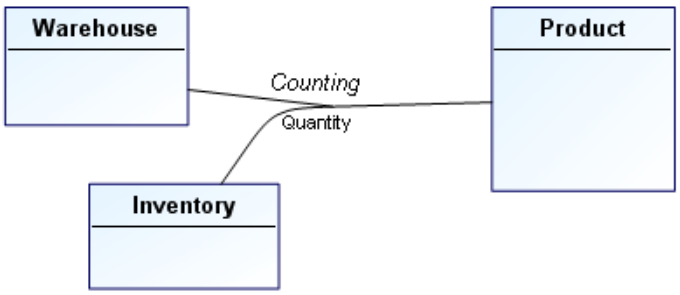
🕮 *A role enables indication of one of the entities concerned by the association. Indication of roles is particularly important in the case of an association between an entity and itself.*

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.

Displaying an 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.

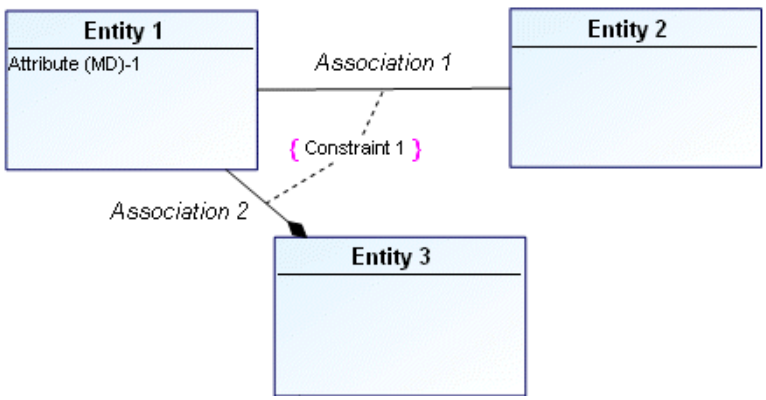
Most *constraints* involve associations between entities.

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.



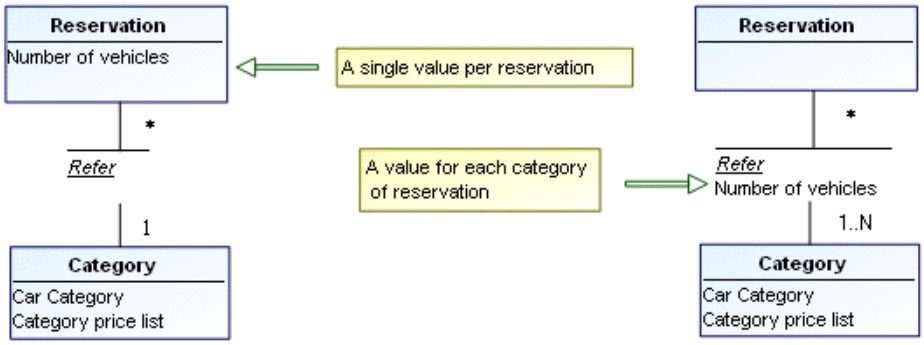
☺ *Save your work regularly using the **Save** button.*

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.

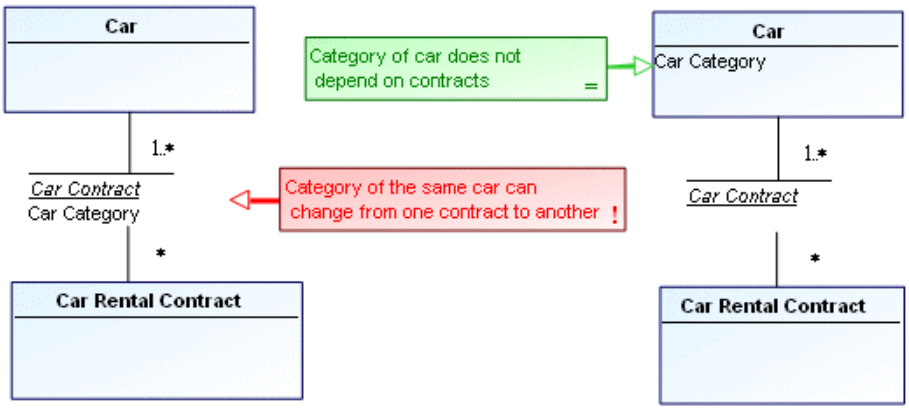


If the number of vehicles is an attribute of the "Reservation" entity, you can only indicate the total number of vehicles for a reservation. You must therefore make one reservation per category of rental vehicle (multiplicity of 1).

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.

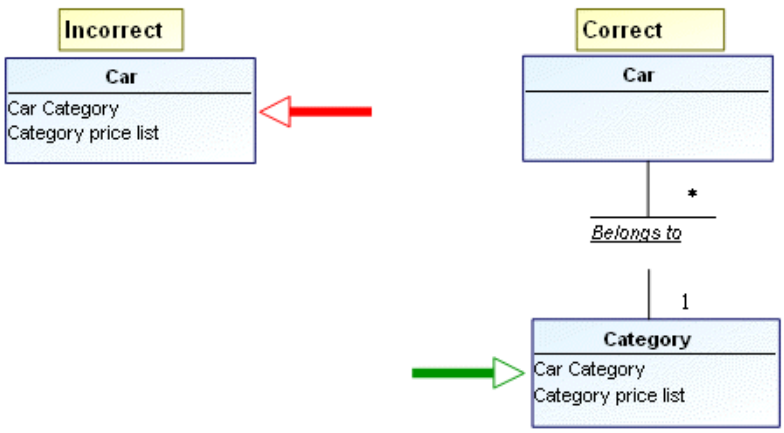


If the car category is an attribute of the "Car Contract" association, this assumes that the car category may change from one contract to the next, which would not be very honest.

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.*

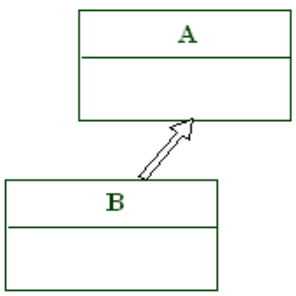
GENERALIZATIONS

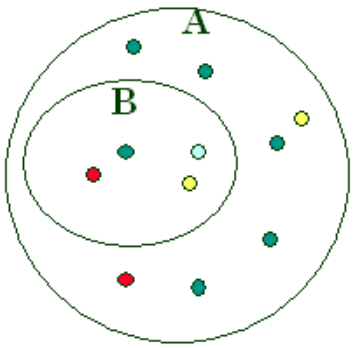
See:

- ✓ "What is a generalization?", page 40
- ✓ "Multiple sub-entities", page 42
- ✓ "Multiple inheritance", page 44
- ✓ "Creating a generalization", page 44
- ✓ "Discriminator", page 44

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.
```

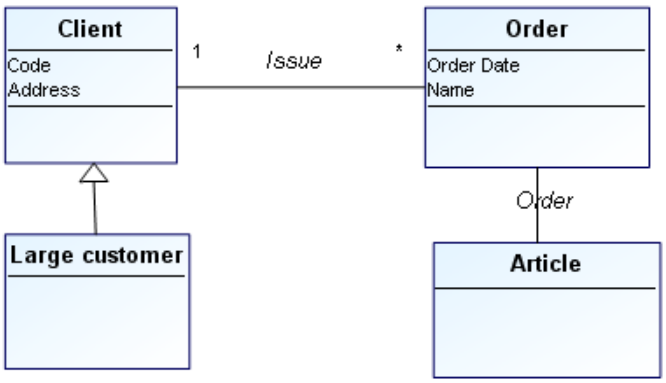
B being a subset of A, the instances of entity B "inherit" the characteristics of those in entity A.

It is therefore unnecessary to redescribe for entity B:

- Its attributes
- Its associations

Example:

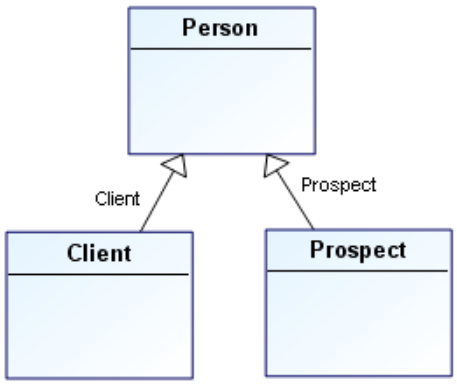
The "Large Client" entity, representing clients with a 12-month revenue exceeding $1 million, can be a specialization of the Client entity (origin).



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.

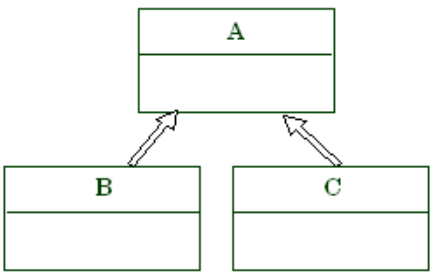
"polygon", "ellipse" and "circle" are sub-entities of the "shape" entity.

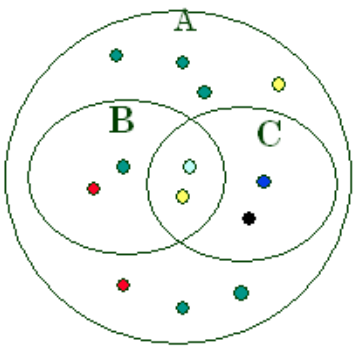
"oak", "elm" and "birch" are sub-entities of the "tree" entity.

"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

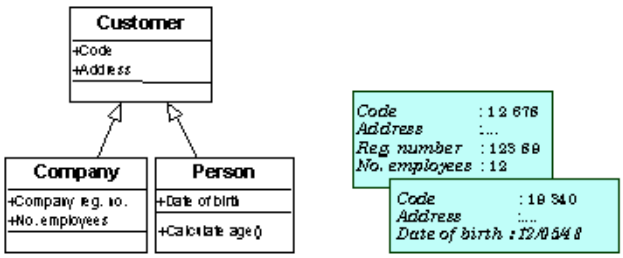




Several sub-entities of the same entity:

- are not necessarily exclusive.
- do not necessarily partition the set.

Advantages of sub-entities

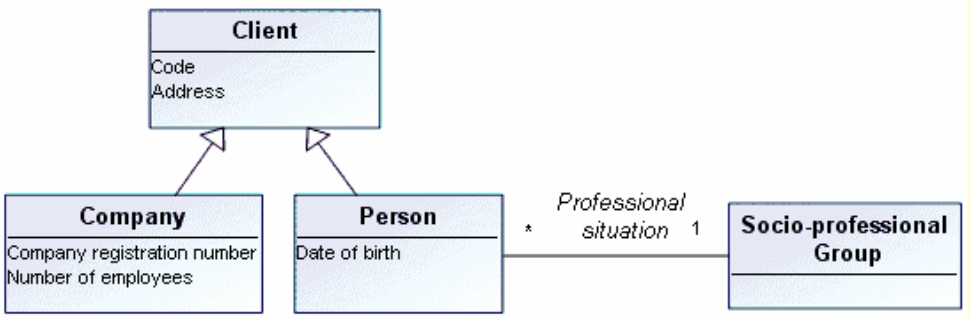


A sub-entity inherits all the attributes and associations of its super-entity, but can have attributes or associations that the super-entity does not have.

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.

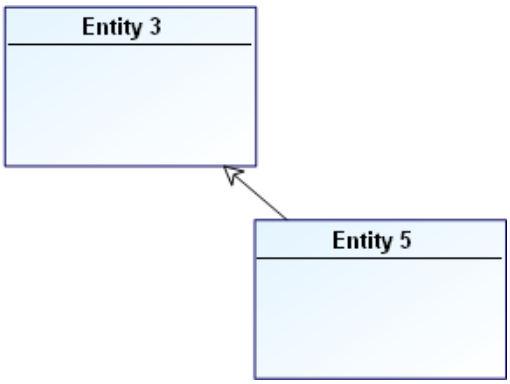
Multiple inheritance

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*:

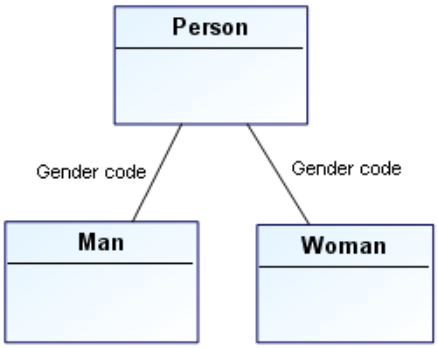
1. In the data diagram insert toolbar, click the **Generalization** button.
2. Click the sub-entity, in this example "Entity 5", and drag the mouse to the general entity, in this example "Entity 3", then release the button. The generalization is now indicated in the diagram by an arrow.



Discriminator

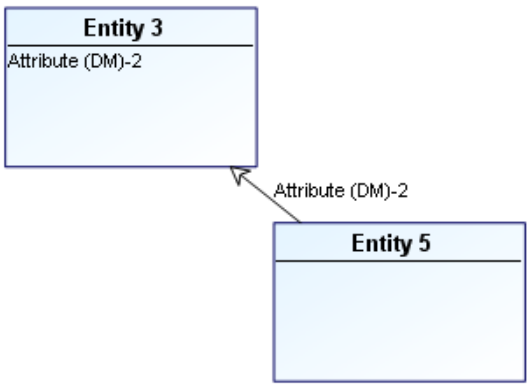
The discriminator is the general entity attribute whose value partitions the objects into the sub-entities associated with the generalization.

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)**.
4. Find ans select the discriminator among the super-entity attributes. Once selected, the discriminator is displayed on the generalization.



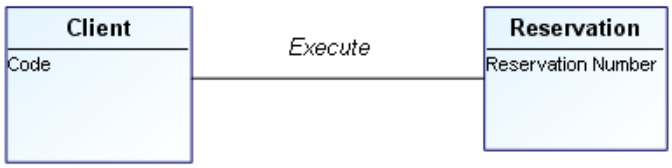
☛ *You can also indicate if the generalization is **Complete**: in this case all instances of the generic entity belong to at least one of the category entities of the generalization.*

IDENTIFIERS

Defining an 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:

1. 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.

> ☛ *In Windows Front-End, a candidate key comprising this attribute is then automatically created for this entity. The corresponding primary key will be created in the table generated for the entity at synchronization of the data model with the relational model using **HOPEX Database Builder**.*

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 function data, while assuring traceability. The MEGA Dictionary tool supplements this approach, enabling compilation of business function vocabulary structured as a dictionary.

For more detailed information, see HOPEX Common Features.

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 **Main Menu**
- A data model
- A data package
- A database

To run the mapping editor from the **Main Menu**:

1. Select **Main Menu** >**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.

> ☛ *In Windows Front-End, you can also create the mapping from the pop-up menu of the last object selected, by clicking **Map**.*

The mapping is created from the last object selected.

Deleting a mapping

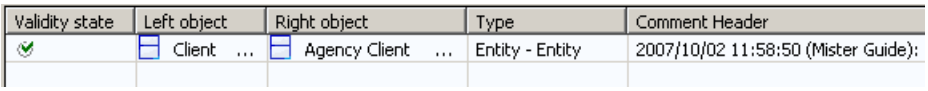
To delete a mapping on an object:

- Select the object in question and click the **Delete mapping item** button.

> ☛ *In Windows Front-End, you can also delete a mapping from the pop-up menu of the object selected, by clicking **Undo mapping**.*

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.



Mapping properties

To view mapping properties:

- 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.

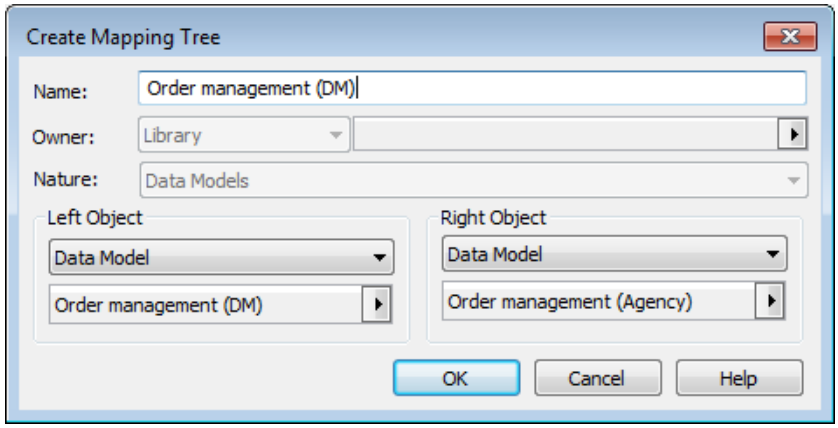
At logical level, the "Order Management (Agency)" data model presents an operational view of IS system data specific to each agency.

We find identical concepts in each of the models. These are however distinct objects.

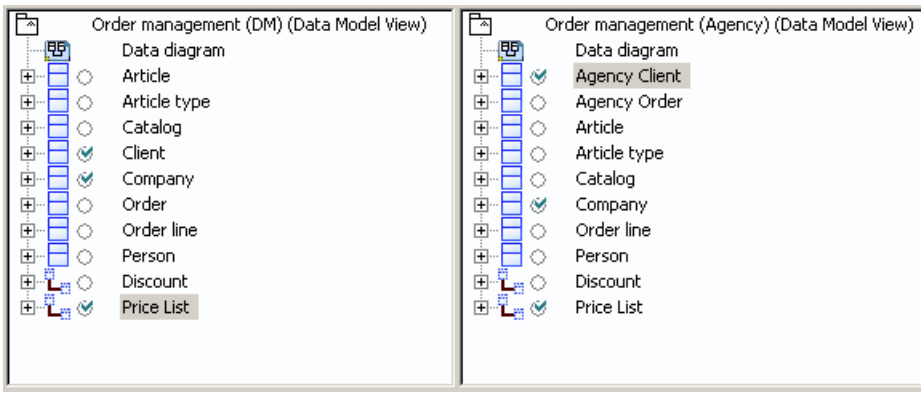
You can map the two data models to favor cohesion between the business function requirements and the systems that support them.

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.

OTHER NOTATIONS

This chapter presents the other notations available with the **HOPEX Logical Data product.**

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 navigation tree, expand the **Data Modeling** folder.
3. Click **Data Notation**.
4. In the right-hand side of the window, select the IDEF1X notation:
5. Click **OK**.

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 Logical Data**, 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.

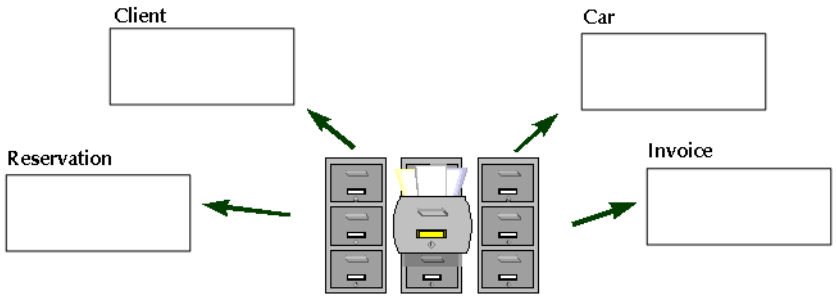
To create a data diagram:

- 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 "Attribute Types", page 95 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 "Categorization Relationships (Generalizations) - (IDEF1X)", page 63.

Specifying the entity identifier

To specify the entity identifier:

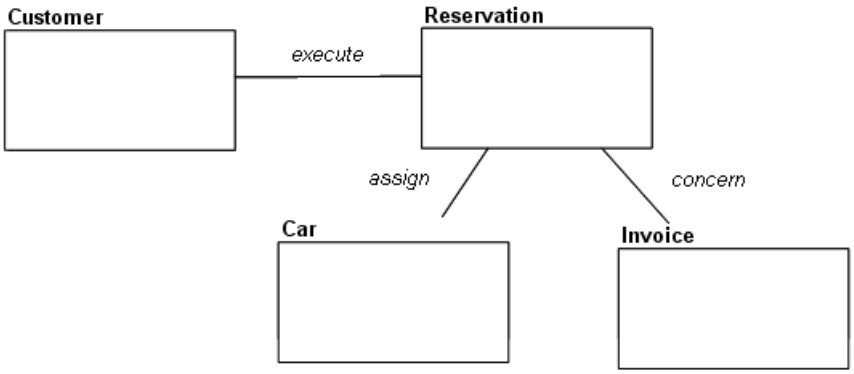
1. Open the properties dialog box of the entity.
2. Select the **Attributes** tab.
3. For the chosen attribute, select "Yes" in the **Identifier** column.

☛ *For more details, see "Defining an Entity Identifier", page 46.*

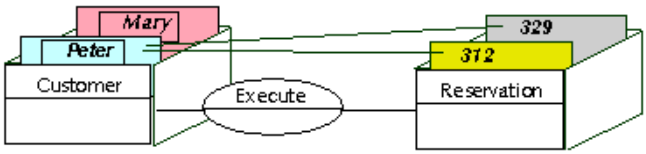
Associations (IDEF1X)

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.



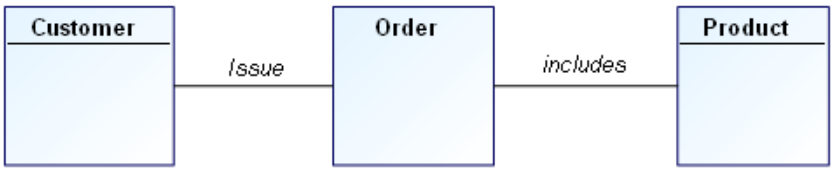
Peter and Mary are clients. Peter has made reservations numbers 312 and 329.

A data diagram should be able to store all situations in the context of the company, but these situations only.

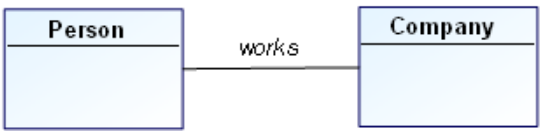
> ☛ *The diagram should not allow representing unrealistic or aberrant situations.*

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.

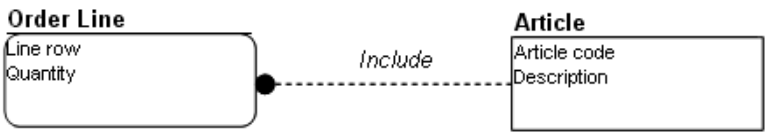
Mandatory non-identifying relationship

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

An optional 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 zero or one instance of the first entity. It is represented by a dashed line with a dot at the second entity end of the line and a small diamond at the other end.

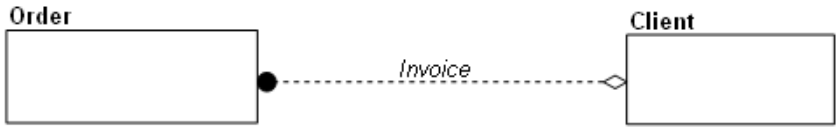
In an *optional non-identifying relationship*, each instance of the child entity is related to zero or one instances of the parent entity.

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.

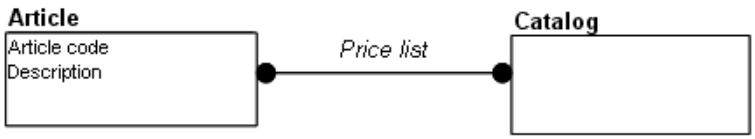
In the initial development of a model, it is often helpful to identify "non-specific relationships" between entities. These non-specific relationships are refined in later development phases of the model.

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.



non-specific relationship

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

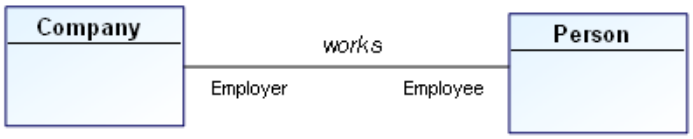
In the above example, an article can be discounted for zero, one or several clients and a client can have discounts for zero, one or several articles. In each case, the discount rate is indicated on the associative class.

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 MEGA 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.

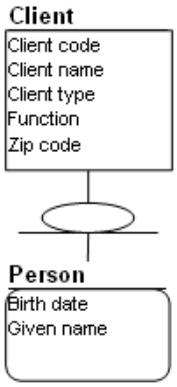
A "categorization relationship" or "generalization" is a relationship between one entity, referred to as the "generic entity", and another entity, referred to as a "category entity" or "specialized entity". Cardinality is not specified for the category entity since it is always zero or one.

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.



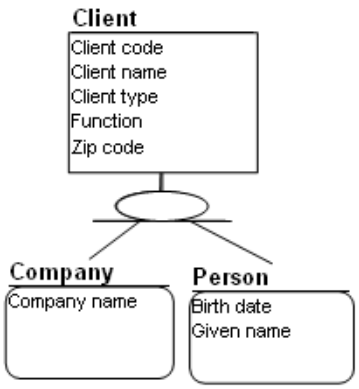
Categorization relationship

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 navigation tree, expand the **Data Modeling** folder.
3. Click **Data Notation**.
4. In the right-hand side of the window, select the I.E. notation:
5. Click **OK**.

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 diagram 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.

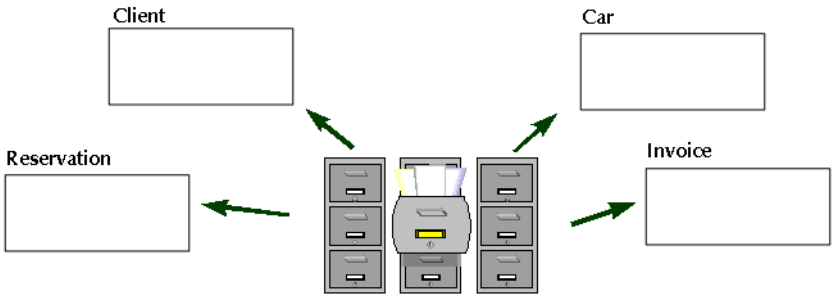
To create a data diagram:

- 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.
```

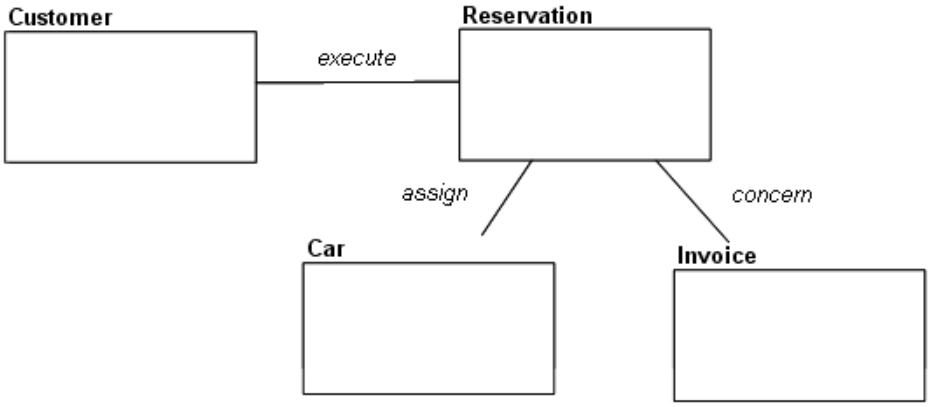
> 🕮 *A datatype is used to group characteristics shared by several attributes. Data types are implemented as classes.*
>
> ☛ *See "Attribute Types", page 95 for more details on data types that can be assigned to an attribute.*

Associations (I.E)

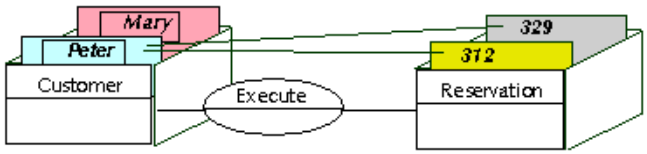
> 🕮 *An association is a meaningful link between two objects. Associations are used to capture data about the relationship between two objects.*

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 crows-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".

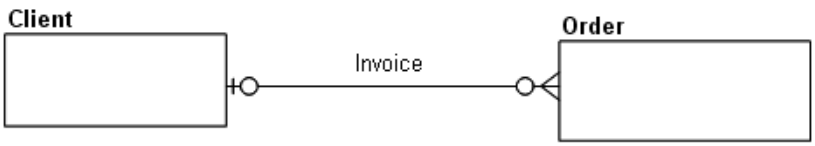
Mandatory relationship

 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

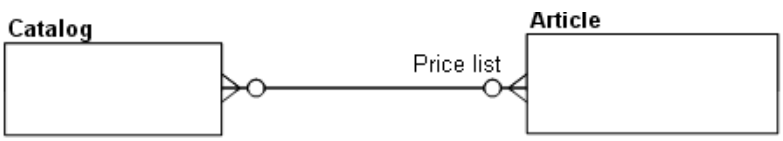
 An optional relationship means that each instance of the first entity is associated with zero or one instance of the second entity and that the second entity can be associated with zero, one or many instances of the first entity.



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.

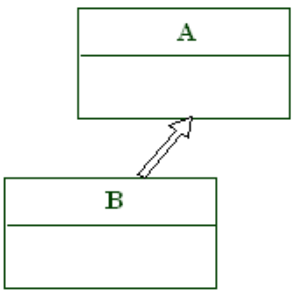
The representation of the association changes according to its new multiplicities.

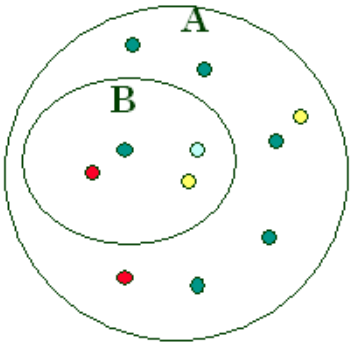
☛ *In MEGA Windows Front-End, multiplicity is also displayed in the role's pop-up menu. If the menu you see does not propose multiplicity, check that you clicked on that part of the line indicating the role and not the association.*

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.

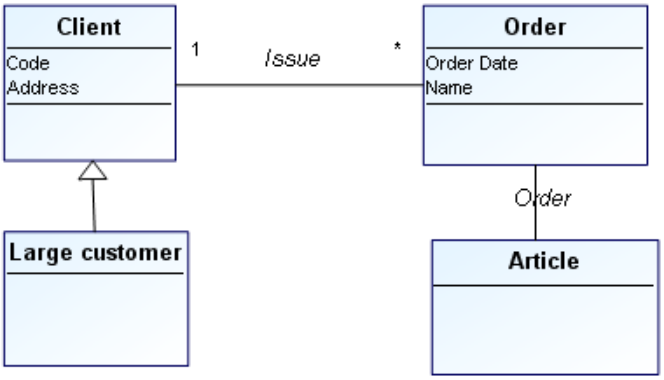
B being a subset of A, the instances of entity B "inherit" the characteristics of those in entity A.

It is therefore unnecessary to redescribe for entity B:

- Its attributes
- Its associations

Example:

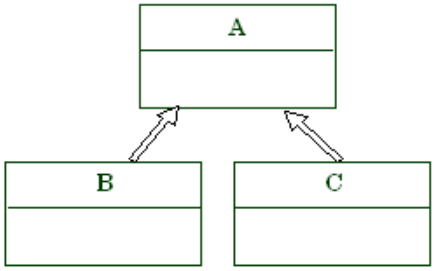
The "Large Client" entity, representing clients with a 12-month revenue exceeding $1 million, can be a subtype of the Client entity.

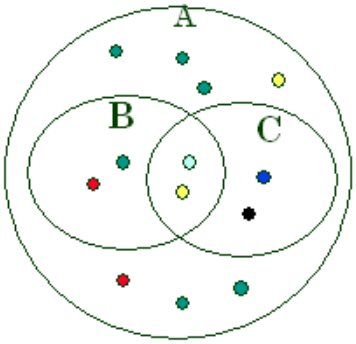


A subtype inherits all attributes, associations, roles and constraints of its supertype, but it can also have attributes, associations, roles or constraints that its supertype does not have.

In the above example, the attributes, associations, roles and constraints specified for "Client" are also valid for "Large Client".

Multiple Subtypes

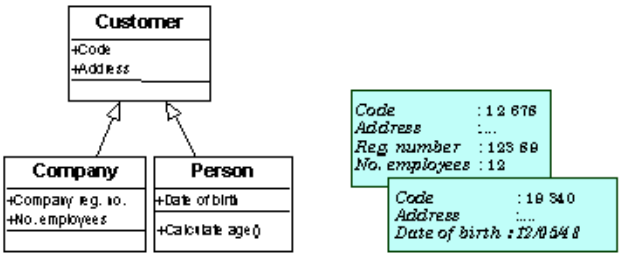




Several subtypes of the same entity:

- are not necessarily exclusive.
- do not necessarily partition the type.

Advantages of sub-types

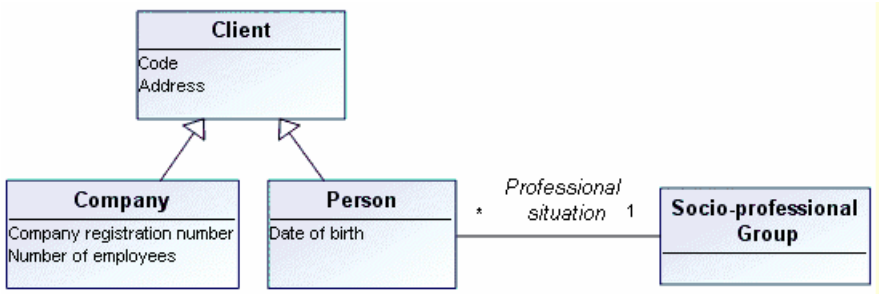


A subtype entity inherits all the attributes and associations of its supertype entity, but can have attributes or associations that the supertype entity does not have.

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.

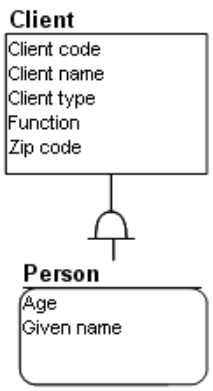
Multiple inheritance

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 navigation tree, expand the **Data Modeling** folder.
3. Click **Data Notation**.
4. In the right-hand side of the window, select the Merise notation:
5. Click **OK**.

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 Logical Data**, 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**, 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 (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.

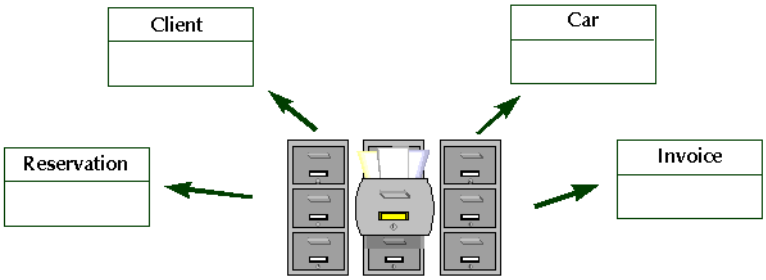
To create a data diagram:

- 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.

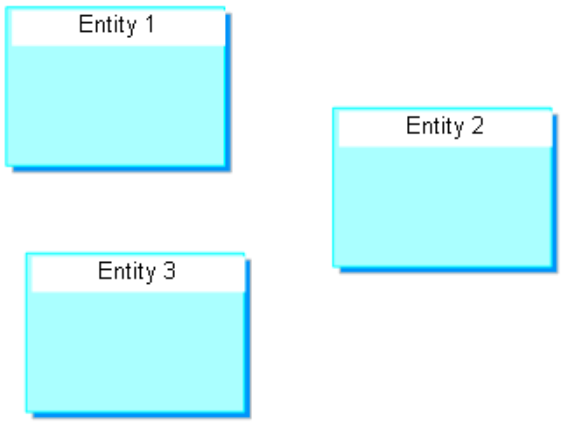


An entity represents a particular object class, of which all instances can be described in the same way.

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 "Defining an Entity Identifier", page 46.*

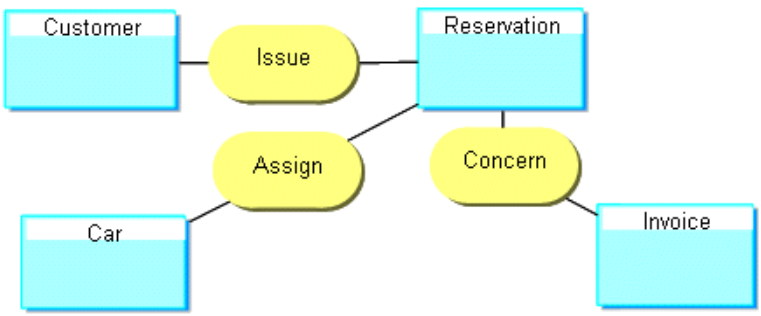
The associations (Merise)

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.

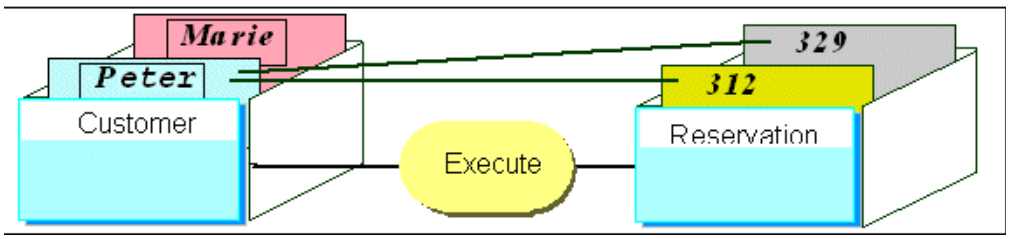
Examples of associations

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.



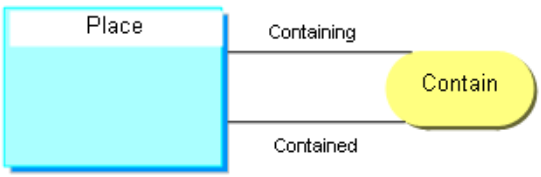
Peter and Mary are clients. Peter has made reservations numbers 312 and 329.

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.*

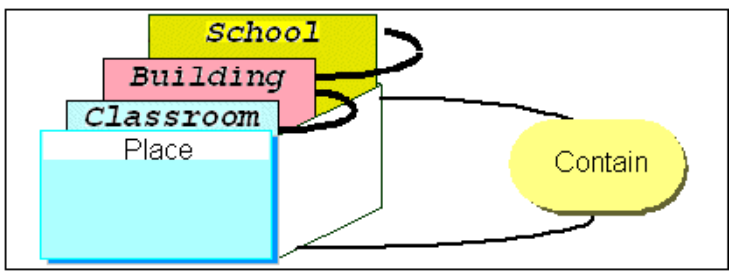
Reflexive relationships

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.

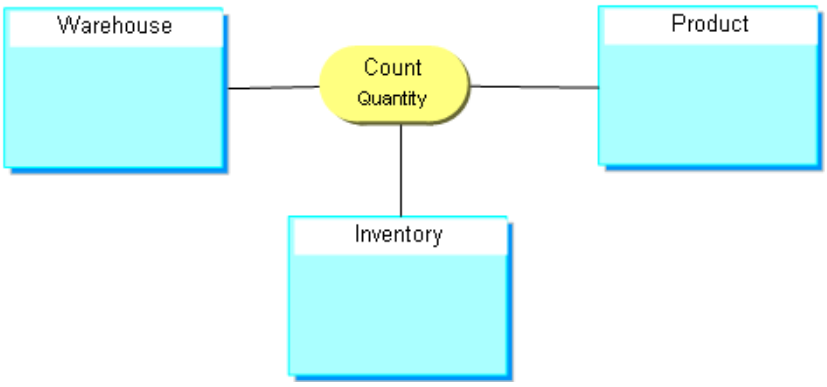
"n-ary" relationships

Certain associations associate more than two entities.

These associations are generally rare.

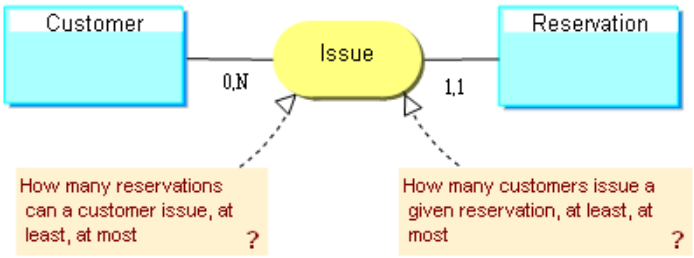
Example

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.

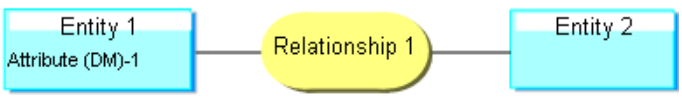
1.1 Each order has one and only one corresponding invoice.

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.*

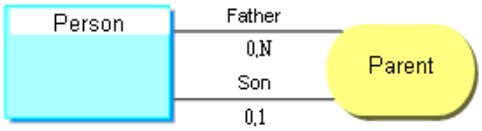
Reflexive relationships

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.

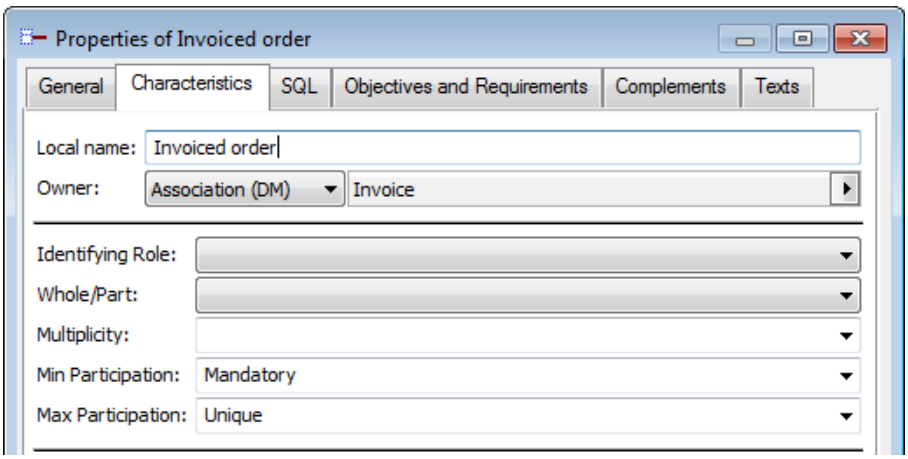
Example:

"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).



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.

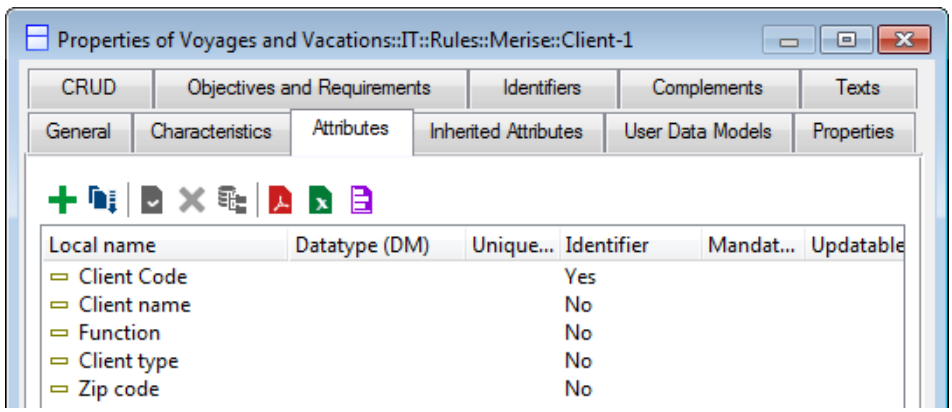
☛ *An identifier consists of one or several mandatory attributes or roles that enable unique identification of an entity.*

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:

- Click the New button and enter the name of the attribute.

You can specify its characteristics (see "Attribute Description", page 90 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.

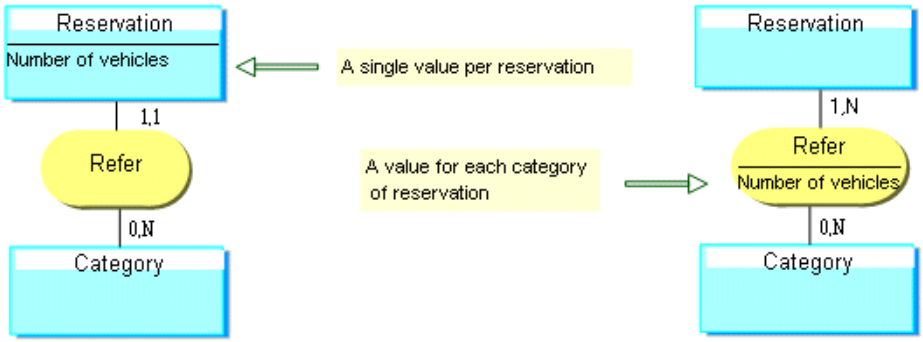
Normalization Rules (Merise)

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).

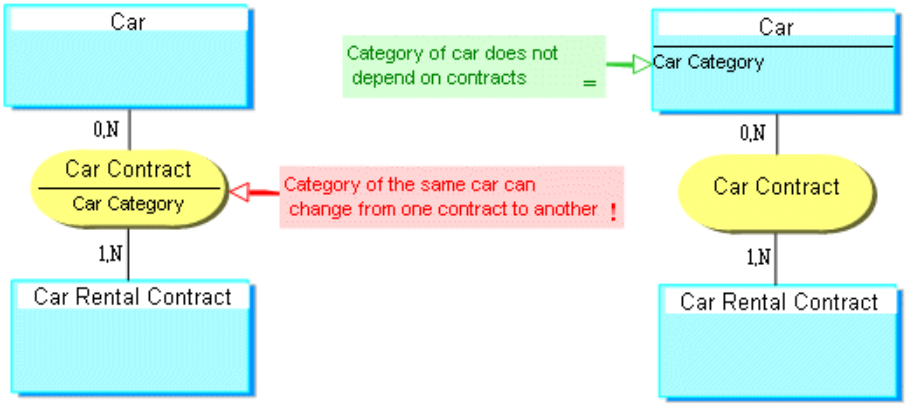


If the number of vehicles is an attribute of the "Reservation" entity, you can only indicate the total number of vehicles for a reservation. You must therefore make one reservation per category of rental vehicle (cardinalities1,1).

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.

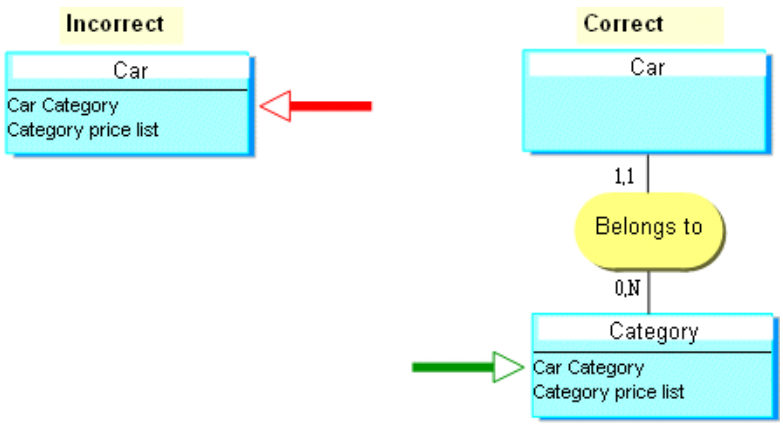


The fact that the car category is an attribute of the "Car Contract" association assumes that the car category may change from one contract to the next, which would not be very honest.

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:

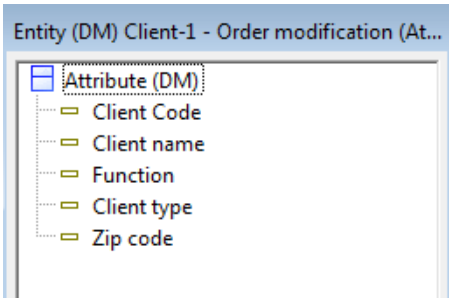
- Sub-type entities.
- Constraints that must be respected by data in documentary terms. These constraints are imposed by checks carried out during data update processing.

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.

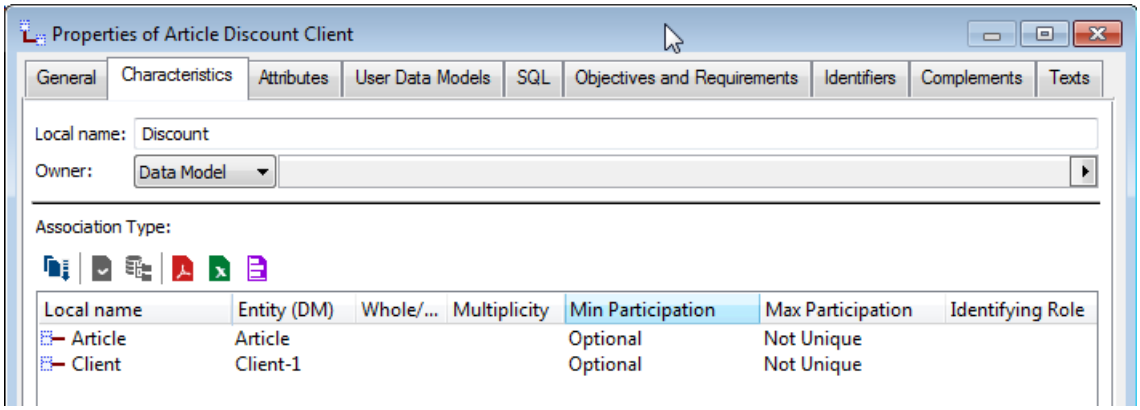
You can enter the attribute characteristics values in the corresponding fields.

- 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.



☛ *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:

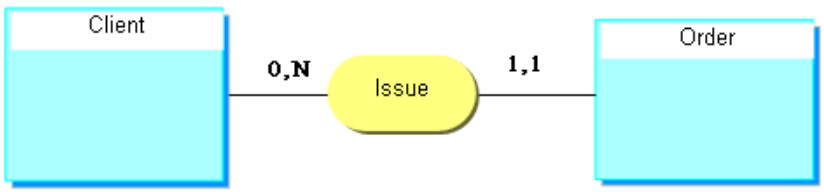
- 0 or 1 for minimum cardinality (optional or mandatory minimum participation).
- 1 or N for maximum cardinality (unique or not unique maximum participation).
  Different values are permitted.

  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.

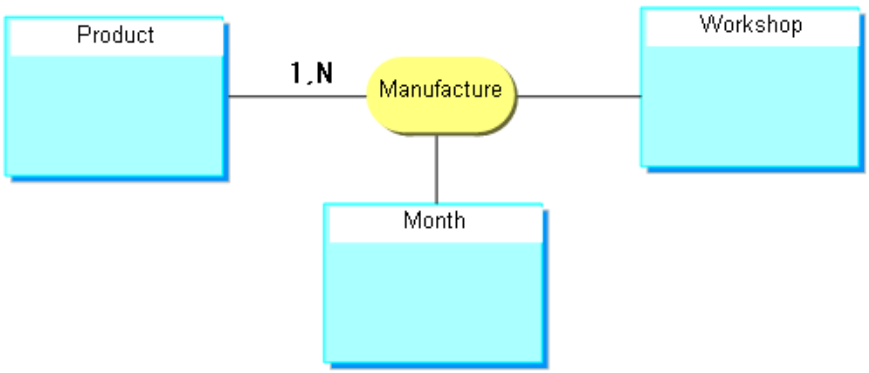
> ☛ *If expression of cardinalities is not sufficient to describe the link that exists between an entity and an association, for example when a cardinality depends on an organizational context, it is possible to use cardinality constraints, which enable more precise description.*

Examples



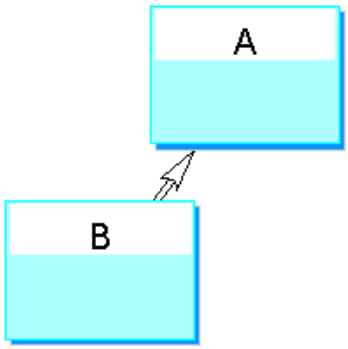
0,N : The client can issue no order, can issue a maximum of N orders (N indeterminate).

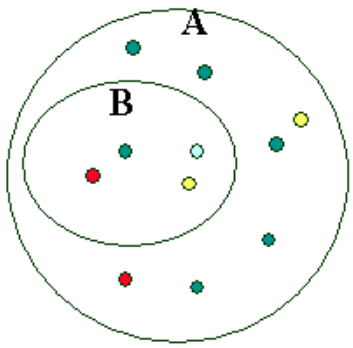
1.1, The order must be issued by one and only one client.



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.

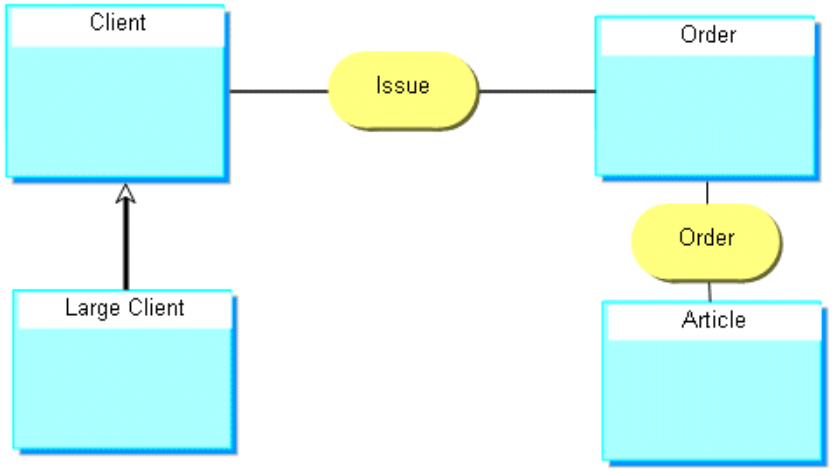
B being a subset of A, the instances of entity B "inherit" the characteristics of those in entity A.

It is therefore unnecessary to redescribe for entity B:

- its properties
- Its associations

Example

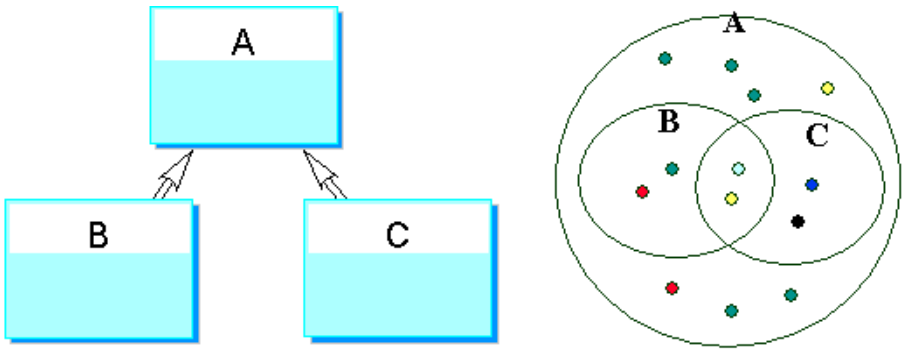
The "Large Client" entity, representing clients with a 12-month revenue exceeding $1 million, can be a subtype of the Client entity (origin).



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".

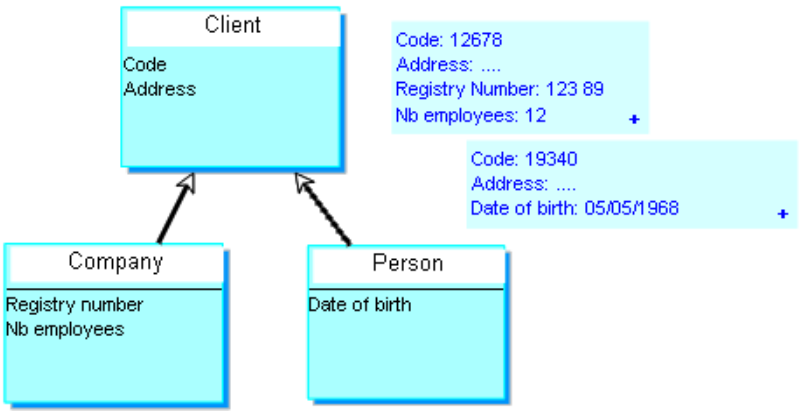
Multiple Subtypes



Several sub-types of the same entity

- are not necessarily exclusive.
- do not necessarily partition the type.

Advantages of sub-types

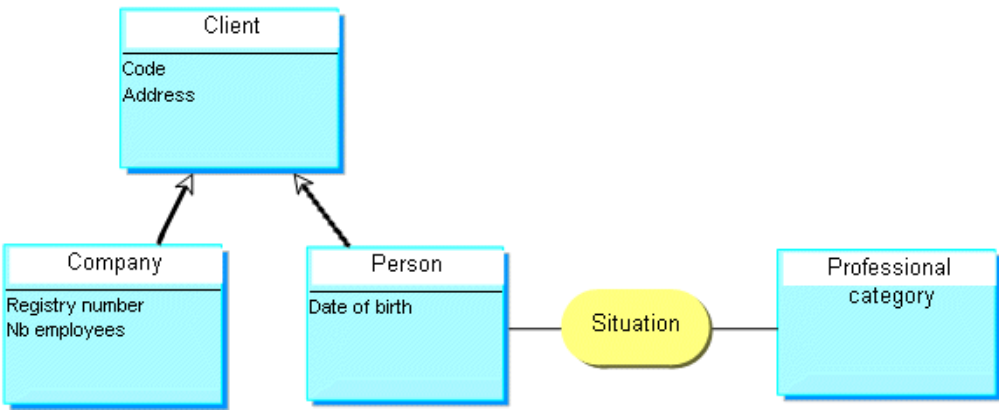


A sub-type entity can have specific properties. These only have meaning for that particular sub-type. In the above example:

- "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.

ATTRIBUTE TYPES

The following points are covered here:

DATATYPE PACKAGES

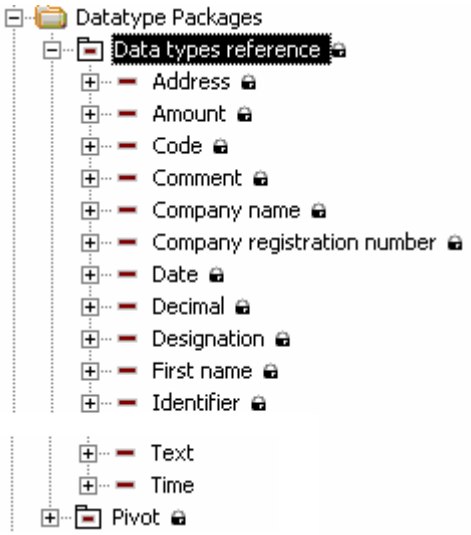
> 🕮 *A data model is used to represent the static structure of a system, particularly the types of objects manipulated in the system, their internal structure, and the relationships between them. A data model is a set of entities with their attributes, the associations existing between these entities, the constraints bearing on these entities and associations, etc.*
>
> 🕮 *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.*

A type is used to group characteristics shared by several attributes.

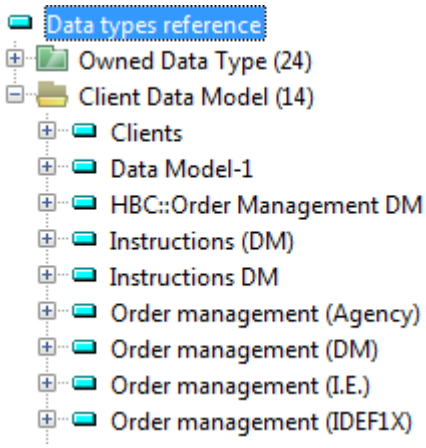
To type attributes of an entity, only datatypes defined for the data model that contains this entity are proposed.

When you create a data model, the "Datatype Reference" datatype package is automatically associated with it by default.

Ths "Datatype Reference" package owns datatypes "Address", "Code", "Date", etc.



Opening the explorer on this datatype package, you can see that is 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:

1. On the desktop, click on the **Logical data** navigation pane.
2. Display the list of **Data Model Packages**.
3. In the edit area, click **New**.
4. Enter the package name and click **OK**.

You can then add types to this package.

Creating a datatype

To create a datatype:

1. Right-click the package name and open its **Properties**.
   The package properties appear.
2. Click the drop-down list then **Data Types**.
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 model:

1. On the desktop, click the navigation menu, then **Logical data**.
2. In the logical data navigation pane, click **All data models**.
   The list of data models appears in the edit area.
3. Right-click the model concerned and open its **Properties**.
4. Click the drop-down list then **Data Type Packages Used**.
5. Click **Connect**.
   The query dialog box appears.
6. Click **Find**.
   The list of datatype packages appears.
7. Select the desired package and click **Connect**.

ASSIGNING TYPES TO ATTRIBUTES

When the datatype package has been referenced for the data model, the list of types it contains is available on each attribute of entities of the model. All that is required is to select the one that is suitable.

To define the type of an attribute:

1. Right-click the entity that contains the attribute.
2. Select **Properties**.
   The properties dialog box of the attribute opens.
3. Click the drop-down list then **Attributes**.
4. In the **Datatype (DM)** box corresponding to the attribute, select the desired type in the list.
5. Click **Apply**.