INTRODUCTION À HOPEX INFORMATION ARCHITECTURE

HOPEX Information Architecture permet de construire l'architecture globale des données, de la définition des données métier à la conception de bases de données. Elle assure la traçabilité des données entre les différents niveaux : conceptuel, logique et physique.

PÉRIMÈTRE DE LA SOLUTION HOPEX IA

La solution **HOPEX Information Architecture** couvre les trois niveaux de modélisation des données d'une organisation :

- Niveau métier (conceptuel): permet de définir les concepts de l'architecture métier. Ces concepts peuvent être mis en œuvre par des objets du niveau logique et être décrits par des modèles de données.
  Voir "HOPEX Information Architecture - Couche métier", page 5.
- Niveau logique : s'adresse à la maîtrise d'ouvrage qui souhaite élaborer des modèles généraux orientés métier. Il s'agit ici de modéliser les données d'un domaine, d'une application ou encore d'un processus. Il représente ce que l'on veut faire, à quoi l'on veut arriver, indépendamment des questions techniques liées à la mise en oeuvre. La représentation des données peut se faire dans un modèle de données ou un diagramme de classes.
  Voir : "HOPEX Information Architecture - Couche logique".
- Niveau physique : consiste à définir les modèles destinés à persister dans un SGBD. Il s'agit de spécifications détaillées en vue de la réalisation du schéma physique de la base. Il est représenté par le diagramme relationnel.
  Le niveau physique définit également la façon selon laquelle sont stockées les données et les méthodes pour y accéder. Il permet l'exploitation des données par les SGBD.
  Voir "HOPEX Information Architecture - Couche physique".

Niveau de modélisation	Détails
Métier	Définition du vocabulaire métier Réalisation des concepts
Logique	Modélisation des données logiques Réalisation des concepts par les entités et classes
Physique	Modélisation / génération des données physiques Synchronisation des modèles logiques et physiques

NOUVEAUTÉS DE LA VERSION HOPEX IA V2R1

Synchronisation des parties

Le nouveau standard de modélisation basé sur le concept de "Partie", qui remplace les associations pour connecter des classes, est pris en compte dans le processus de synchronisation.

Voir "Formalisme logique et synchronisation", page 2.

LE BUREAU INFORMATION ARCHITECTURE

Se connecter à HOPEX Information Architecture

Pour se connecter à **HOPEX Information Architecture**, voir HOPEX Common Features, "Le bureau HOPEX Web Front-End".

> ☛ *Pour plus de détails sur l'utilisation de la plateforme Web des solutions HOPEX, voir le guide **HOPEX Common Features**.*

Les menus et commandes disponibles dans **HOPEX Information Architecture** dépendent du profil avec lequel vous êtes connecté.

Voir "Les profils de HOPEX Information Architecture", page 6.

Afficher l'environnement de travail d'une entreprise

Un référentiel peut être partitionné en *Entreprises*.

Une entreprise est un projet d'entreprise, un effort mis en œuvre par une ou plusieurs organisations, ayant pour finalité la mise à disposition de biens et de services en adéquation avec la mission de l'entreprise et son environnement. L'entreprise établit des buts de l'entreprise à atteindre, ainsi que les plans d'action stratégiques visant à l'accomplissement de ces buts. Elle est constituée de phases de transformation dans lesquels sont définis les capacités ou livrables à atteindre.

Lorsqu'elles sont associées à un environnement de travail, les entreprises constituent des points d'entrée dans IA ; l'environnement offre un accès privilégié aux objets détenus et utilisés par l'entreprise en question.

Créer une entreprise et son environnement de travail

La création des entreprises et des environnements de travail est réalisée par l'administrateur fonctionnel IA.

Pour créer une entreprise dans **HOPEX Information Architecture** :

1. Cliquez sur le menu de navigation puis sur **Environnement**.
2. Dans le volet de navigation cliquez sur **Navigation standard**.
3. Dans la zone d'édition cliquez sur la tuile **Entreprises**.
4. Cliquez sur **Nouveau**.
5. Dans l'assistant de création, indiquez le nom de l'entreprise.
6. Pour créer en même temps l'environnement de l'entreprise, sélectionnez le type d'environnement "Information Architecture".
7. Cliquez sur **OK**.

Si aucun environnement n'a été créé en même temps que l'entreprise, vous pouvez le créer ultérieurement.

Pour assigner un environnement de travail à une entreprise existante dans **HOPEX Information Architecture** :

1. Sélectionnez le projet ou l'entreprise en question pour afficher ses propriétés.

   ☛ *Cliquez sur le bouton Propriétés de la zone d'édition si les propriétés ne s'affichent pas.*
2. Sélectionnez la page **Assignation de l'environnement de travail**.
3. Cliquez sur **Nouveau**.
4. Renommez si besoin le nouvel environnement et sélectionnez le type "Information Architecture".
5. Cliquez sur **OK**.

   ☛ *Vous pouvez aussi créer l'environnement de travail d'une entreprise lors de sa création.*

Afficher l'environnement de travail d'une entreprise

Pour afficher l'environnement de travail d'un projet ou d'une entreprise :

1. Cliquez sur le **Menu principal** et sélectionnez **Changer d'environnement de travail**.
2. Sélectionnez le projet ou l'entreprise sur lequel vous souhaitez travailler.

Le bureau IA affiche les objets spécifiques au projet ou à l'entreprise sélectionnée. Pour chaque type d'objet, par exemple les concepts sous le volet **Information métier**, la zone d'édition présente les objets détenus par l'entreprise ou le projet ainsi que les objets importés, autrement dit utilisés mais non détenus par l'entreprise ou le projet.

Par défaut les étapes de l'environnement de travail sont visibles par tous les utilisateurs. Vous pouvez définir plus précisément les participants au projet ou à l'entreprise.

Voir aussi :"Utiliser les entreprises".

LES PROFILS DE HOPEX INFORMATION ARCHITECTURE

Dans **HOPEX Information Architecture**, il existe, par défaut, des profils auxquels sont associés des droits et accès.

L'Architecte d'information métier

L'**Architecte d'information métier** est un représentant du métier de l'entreprise. Il est chargé de structurer les informations métier de l'entreprise afin d'en faciliter la gestion et l'accès. Il est de charge de concevoir le vocabulaire d'entreprise en modélisant les informations, leur détails et les mises en relations ainsi que les différents domaines de connaissance

L'**Architecte d'information métier** est responsable de l'exécution des tâches suivantes :

- Identification des domaines de connaissance,
- Création et définition des domaines d'information métier,
- Création, définition et classification des concepts et des types de concept,
- Création des termes,
- Construction des diagrammes d'architecture de l'information,
- Création des vues de concept,
- Création des rapports qui facilitent l'accès à l'information.

☛ *Pour plus de détails sur les activités de l'Architecte d'information métier, voir "Décrire l'architecture d'information métier", page 25.*

Architecte de données

L'**Architecte de données** est un intervenant SI, il a accès en lecture et écriture aux données logiques de l'entreprise. Il a pour responsabilité de modéliser toutes les données logiques (classes, associations, attributs, etc.) ainsi que les domaines de données permettant d'utiliser ces informations dans les cartographies de processus ou d'application.

Il est responsable de l'exécution des tâches suivantes :

- Définition des données logiques,
- Création des réalisations qui relient les données logiques aux concepts métier.

Architecte de base de données

L'**Architecte de base de données** a la responsabilité de la conception des bases de données. Pour chaque version du SGBD cible, il utilise les données logiques et produit la vue physique via les outils de synchronisation.

Administrateur de base de données

L'**Administrateur de base de données** peut se connecter au bureau pour consulter les bases de données qui lui sont assignées et générer les fichiers SQL correspondants.

Administrateur fonctionnel IA

L'**Administrateur fonctionnel IA** a la charge de gérer toutes les tâches d'administration du produit. Il possède des droits sur tous les objets.

- Il gère la création des utilisateurs et leur assignation aux profils.
- Il prépare l'environnement de travail et crée les éléments nécessaires à la gestion de l'information.
- Il peut intervenir sur :
  - les domaines de connaissance,
  - les domaines d'information métier,
  - les concepts, type de concept et vues de concept,
  - les diagrammes d'architecture de l'information,
  - les rapports,
  - tous les composants du référentiel.

LES RÔLES MÉTIER DE HOPEX INFORMATION ARCHITECTURE

Dans **HOPEX Information Architecture**, les objets peuvent être assignés à des personnes avec les rôles suivants :

- **Concepteur de données** (Data Designer) : permet de spécifier la personne responsable de la conception d'un objet (tel qu'un paquetage, un domaine de données, une bases de données, etc.).
- **Scientifique de données** (Data Scientist) :il est en charge du rapprochement entre le concepteur des données (métier et logiques) et les responsables des processus qui utilisent ces données.
- **Administrateur de bases de données** (Database Administrator) : peut être assigné à des bases de données.

☛ *Pour plus de détails sur les assignations, voir "Gérer les assignations des données métier", page 17.*

HOPEX Information Architecture

Guide d’utilisation

HOPEX V2R1

HOPEX INFORMATION ARCHITECTURE - COUCHE MÉTIER

HOPEX Information Architecture offre une solution de gestion et de partage du vocabulaire spécifique à votre entreprise. Cette application permet de recenser, définir, classer et organiser les concepts métier afin d'établir un lien pertinent avec les objets techniques mis en œuvre au niveau du système d'information.

Au niveau métier, **HOPEX Information Architecture** propose aux intervenants métier un outillage pour décrire les concepts qu'ils manipulent ainsi que les liens qui régissent leur organisation. Pour ce faire, **MEGA** s'est appuyé sur les principes largement répandus du web sémantique, ainsi que des cadres ontologiques tels qu'IDEAS ou la norme ISO 15926 (type de haut niveau, cycle de vie et événements).

Au niveau de l'architecture des SI, **HOPEX Information Architecture** offre les outils pour établir des correspondances entre vos modèles de données, basés sur le formalisme UML, et les informations décrites au niveau métier.

Voici les points abordés dans **HOPEX Information Architecture** - Couche métier :

- "IA métier : synthèse des objets utilisés", page 2
- "Décrire l'architecture d'information métier", page 25
- "Rapports sur les données métier", page 74

Pour plus de détails sur l'interface et les fonctionnalités **HOPEX** en général, voir :

- "Le bureau HOPEX Web Front-End", page 31

Le processus de gestion du vocabulaire

L'approche embarquée dans **HOPEX Information Architecture** part des concepts métier élémentaires et va jusqu'à la classification de ces concepts en prenant en compte des concepts temporels liés au cycle de vie des objets.

```
Par exemple : commande passée, commande payée, commande
livrée.
```

Cette approche incrémentale permet aux entreprises de construire progressivement des glossaires complets et adaptés au contexte de leur organisations.

Afin qu'intervenants métier et intervenants SI partagent un vocabulaire commun, **HOPEX Information Architecture** repose sur deux fonctions majeures :

- L'analyse et l'organisation des concepts métier,
- La mise en relation des concepts métier avec les éléments de l'architecture du système d'information.

L'analyse et l'organisation des concepts métier

Ce travail est réalisé par un utilisateur métier. Il consiste à décrire l'ensemble des concepts métier en s'appuyant sur un modèle sémantique simple articulé autour des notions de concept, d'événement et d'état.

- Un concept représente un objet métier, il est caractérisé par :
  - son périmètre, c'est-à-dire ses relations avec les autres concepts

    `Par exemple, un ouvrage est caractérisé par son auteur, son titre, sa date d'édition, etc.`
  - ses liens d'héritage avec d'autres concepts

    `Par exemple, un abonnement est un abonnement livre ou un abonnement média.`
  - ses occurrences,

    `Par exemple, Alexandre Dumas est une occurrence d'Auteur.`
- Un Etat permet d'identifier une évolution temporelle d'un concept

  `Par exemple, un ouvrage est disponible ou emprunté.`
- Un Evénement, représente un fait marquant qui modifie l'état d'un ou de plusieurs concepts.

  `Par exemple, la parution d'un ouvrage.`

HOPEX Information Architecture propose le profil standard "Architecte de données métier" pour assurer le travail d'analyse et d'organisation des concepts métier.

La réalisation des concepts

Les concepts métier sont, en général, mis en œuvre dans le SI avec le support de la méthode et du formalisme UML.

Le travail de "réalisation des concepts" consiste donc à rapprocher les éléments des modèles de données avec les concepts métier afin de :

- définir de manière plus précise les objets manipulés au niveau de l'architecture du SI,
- assurer un meilleur partage du vocabulaire et une meilleure communication globale entre intervenants métier et intervenants SI.

HOPEX Information Architecture propose le profil standard "Architecte de données" pour assurer le travail de "réalisation des concepts".

Voir "Relier les concepts métier aux données logiques", page 17.

A PROPOS DE CE GUIDE

Ce guide vous présente comment tirer parti de **HOPEX Information Architecture** pour construire une architecture efficace de votre information métier.

Structure du guide

Le guide **HOPEX Information Architecture - Couche métier** est composé des chapitres suivants :

- "IA métier : synthèse des objets utilisés", page 2, présente les objets de la suite **HOPEX** qui supportent **HOPEX Information Architecture** ainsi que les diagrammes spécifiques associés ;
- "Décrire l'architecture d'information métier", page 25, présente les fonctionnalités proposées par **HOPEX Information Architecture** aux gestionnaires des données métier pour organiser l'information de l'entreprise ;
- "Rapports sur les données métier", page 74, présente les rapports proposés par **HOPEX Information Architecture** pour une améliorer l'organisation des informations de l'entreprise et leur communication.

Ressources complémentaires

Ce guide est complété par :

- le guide **HOPEX Common Features**, qui décrit les facilités spécifiques aux solutions MEGA.

  ☛ *Il peut être utile de consulter ce guide pour une présentation générale de l'interface.*

- le guide d'administration **HOPEX Administration - Supervisor**.
- le guide **HOPEX Logical Data** présente la gestion des données logiques et les fonctionnalités proposées par **HOPEX Information Architecture** aux gestionnaires des données pour relier les données applicatives aux concepts métiers.

Conventions utilisées dans le guide

Styles et mises en forme

☛ *Remarque sur les points qui précèdent.*

📖 *Définition des termes employés.*

☺ *Astuce qui peut faciliter la vie de l'utilisateur.*

Compatibilité avec les versions précédentes.

Ce qu'il faut éviter de faire.

Remarque très importante à prendre en compte pour ne pas commettre d'erreurs durant une manipulation.

Les commandes sont présentées ainsi : **Fichier > Ouvrir**.

Les noms de produits et de modules techniques sont présentés ainsi : **HOPEX**.

IA MÉTIER : SYNTHÈSE DES OBJETS UTILISÉS

HOPEX Information Architecture permet de définir, structurer et organiser le vocabulaire métier de votre entreprise. A partir d'un modèle sémantique de base, l'application vous offre des menus, des commandes et des diagrammes qui facilitent la construction de votre architecture d'information.

Les notions de base utilisées par **HOPEX Information Architecture** sont introduites dans ce chapitre.

LES CONCEPTS

Avec **HOPEX Information Architecture**, vous pouvez construire un dictionnaire qui décrit et définit les éléments de votre vocabulaire métier.

Le composant de base d'un glossaire est le **Concept**.

> 🕮 *Un concept représente la détermination de ce qu'est un être, une chose ou un mot, par ses propriétés et caractéristiques essentielles ou ses qualités propres.*

Le mot qui est associé à un **Concept** et qui dépend de la langue est un **Terme**.

> 🕮 *Un terme est un mot ou groupe de mots considéré dans sa valeur de désignation, en particulier dans un vocabulaire spécialisé.*

Un terme est donc spécifique à une langue et il n'est pas traduisible. En revanche, il permet de créer et de visualiser les concepts dans la langue choisie par l'utilisateur.

Concept et terme

Dans des langues différentes, un même terme peut représenter des concepts différents.

```
Par exemple : le terme "car" en anglais représente une
voiture (véhicule léger), alors que le même terme en
français représente un véhicule de transport collectif.
```

Les termes ne sont pas traduisibles, donc plusieurs objets de type **Terme** peuvent porter le même nom dans des langues différentes.

En revanche, dans une même langue, un même terme peut représenter plusieurs concepts et le sens qui est donné à ce terme dépend de son contexte d'utilisation.

```
Par exemple, le mot "ring" représente en anglais une
sonnerie de téléphone et une bague.
```

Donc, pour une même langue, un objet de type **Terme** peut être relié à plusieurs concepts. Chacun des concepts donne une définition spécifique de ce terme, dans son domaine de connaissance.

> 🕮 *Un domaine d'information métier est un sous-ensemble d'éléments d'un domaine de connaissance qui permet de réduire le champ d'une étude.*

Par conséquent, avec **HOPEX**, un concept porte le nom du terme qui lui est associé dans la langue choisie par l'utilisateur. Pour changer le nom d'un concept, dans une langue donnée, il faut changer le nom du terme associé.

> ☛ *Pour plus de détails, voir "Utiliser le glossaire dans un contexte multilingue", page 16.*

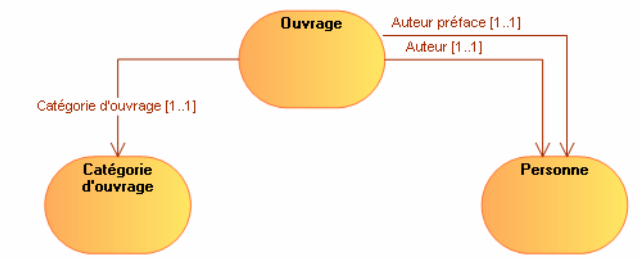
Les liens entre les concepts

Pour définir la sémantique d'un concept, vous pouvez tracer plusieurs types de liens entre les concepts : des liens de définition ou des liens dépendance.

Les liens de définition

Les liens de définition permettent de caractériser un concept.

```
Par exemple, un ouvrage est défini par sa catégorie
d'ouvrage (oeuvre littéraire ou musicale), son auteur,
l'auteur de sa préface.
```



Un lien de définition est décrit par un **Composant structurel de concept** qui peut, éventuellement, être associé à un terme.

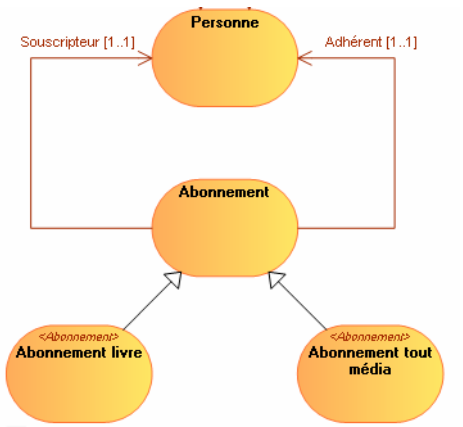
> 🕮 *Un composant structurel de concept permet de représenter une relation de dépendance entre deux concepts. Cette relation est orientée.*
>
> ☛ *Pour plus de détails, voir "Décrire les composants structurels d'un concept", page 45.*

Les liens de dépendance

Certains concepts métier sont des déclinaisons d'autres concepts : ils héritent des mêmes composants structurels de concept.

```
Par exemple, le concept "Abonnement" est décliné en
"Abonnement livre" et "Abonnement tout média". Ces deux
```

types d'abonnement héritent des liens "Souscripteur" et "Adhérent" spécifiés au niveau du concept "Abonnement".



Cette relation est décrite par une **Variation**.

> *Une variation décrit comment un concept peut être varié sous une autre forme. La variante est un objet quasi-similaire à l'objet varié mais avec des propriétés ou des relations qui peuvent différer.*
>
> ☛ *Pour plus de détails sur les variations, voir le guide **HOPEX Common Features**, chapitre "Manipuler les objets du référentiel", "Les variations d'objets".*

Il est également possible de créer une **Variation** entre deux **Composants structurels de concept**.

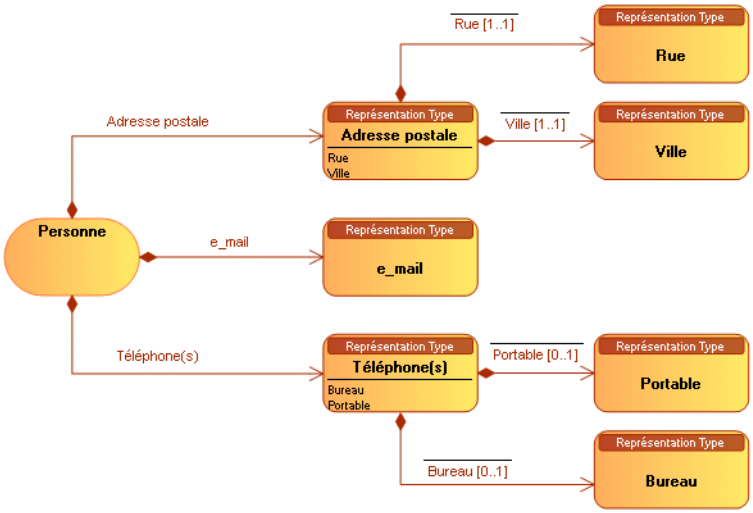
Par exemple, le "Souscripteur" est aussi un "Adhérent".

> ☛ *Pour plus de détails, voir "Décrire les variations d'un concept", page 48.*

Propriétés des concepts : les représentations types

Afin de décrire les caractéristiques attachées à un concept, vous pouvez lier un concept à des représentations types.

Par exemple, une personne est associée à une adresse postale obligatoire et unique, une adresse e_mail éventuellement et un ou plusieurs numéros de téléphone.



Le lien entre un concept et une représentation type est décrit par une **Représentation de concept** qui peut, éventuellement, être associée à un terme.

Un composant de représentation type permet de spécifier la relation entre deux représentations type.

Une représentation de concept permet de spécifier la relation entre un concept et une représentations type.

Pour plus de détails, voir "Utiliser les représentations types", page 47.

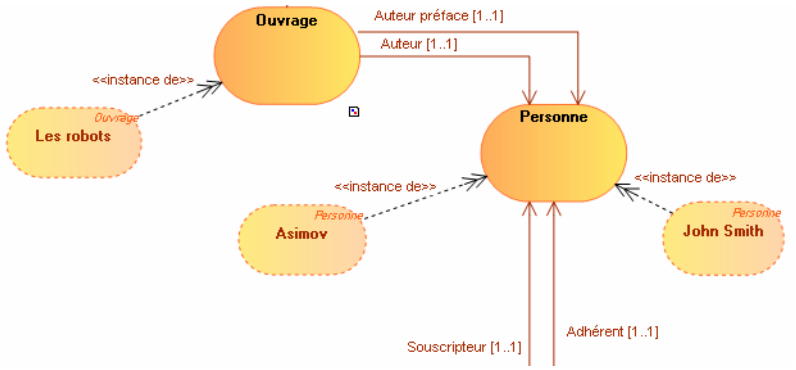
Instances de concept : les individus

Afin de valider le modèle sémantique construit à partir des concepts, **HOPEX Information Architecture** vous permet d'introduire des instances de concept, c'est à dire des objets réels.

Vous pouvez ainsi construire votre modèle sémantique selon deux approches : soit en partant des objets réels pour en déduire les concepts, soit en partant des concepts pour introduire ensuite les objets réels.

Par exemple, le "Asimov" est une instance de "Personne" et "Les Robots" est une instance d'ouvrage.

John Smith est également une instance de "Personne" mais dans la catégorie abonné.



Une instance de concept est un **Individu**.

Un individu représente l'occurrence d'un concept.

La relation entre un concept et ses instances est décrite par une **Classification d'individu**.

Une classification d'individu permet de relier un individu au concept qui le caractérise.

Vous pouvez relier deux individus par une relation de type **Composant d'entité du dictionnaire**.

Un composant d'entité permet de relier un individu à un élément de dictionnaire.

Il devient alors possible de préciser que "Asimov" est l'auteur de l'ouvrage "Les Robots".

☛ *Il n'y a pas de possibilité de décrire des variations entre des individus ou entre des classification d'individu.*

☛ *Pour plus de détails, voir "Décrire les individus", page 53.*

Le cycle de vie d'un concept ou d'un individu

Afin de prendre en compte l'évolution dans le temps des concepts métier, **HOPEX Information Architecture** a introduit deux concepts particuliers :

- Le **Concept d'état** qui permet d'identifier une évolution temporelle d'un concept,

  Un concept d'état est une situation au cours de la vie d'un concept durant laquelle il satisfait à certaines conditions, exerce une certaine activité ou attend un événement de concept. Un concept d'état représente un intervalle de temps dont les bornes sont deux

événements de concept. Un état de concept est une phase par laquelle passe le concept au cours de son cycle de vie.

- Le **Concept événement** qui représente un fait marquant qui modifie l'état d'un ou de plusieurs concepts.

  Un concept événement représente un fait se produisant durant la vie d'un concept, par exemple un changement de saison. Un concept événement permet de marquer l'impact sur un concept d'un phénomène interne ou externe au concept. On peut distinguer les événements de début de concept, les événements de fin de concept et les événements intermédiaires de concept.

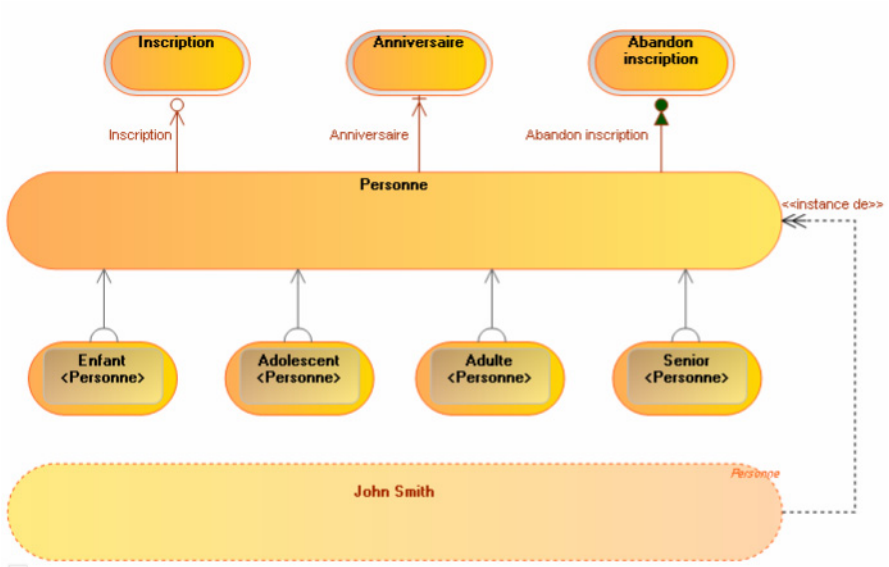
Les **Concepts d'état** et les **Concepts événement** sont des concepts à part entière qui peuvent être décrits comme n'importe quel concept.

Le cycle de vie d'un concept

Un même concept métier peut prendre plusieurs états.

Par exemple, un même Abonné peut passer d'un état de "Enfant" à l'état "Adolescent" puis à l'état "Adulte" et enfin "Senior".

Le passage d'un état à un autre peut être lié à un événement, un "Anniversaire", par exemple.



La relation entre un concept et son **Concept d'état** est décrite par un **Etat du dictionnaire de**.

Un état du dictionnaire de permet de relier un concept à un état de concept et de spécifier la nature de l'état.

La relation entre un concept et son **Concept événement** est décrite par :

- un **Événement de début**,
- un **Événement de fin**,
- ou un **Événement intermédiaire**.

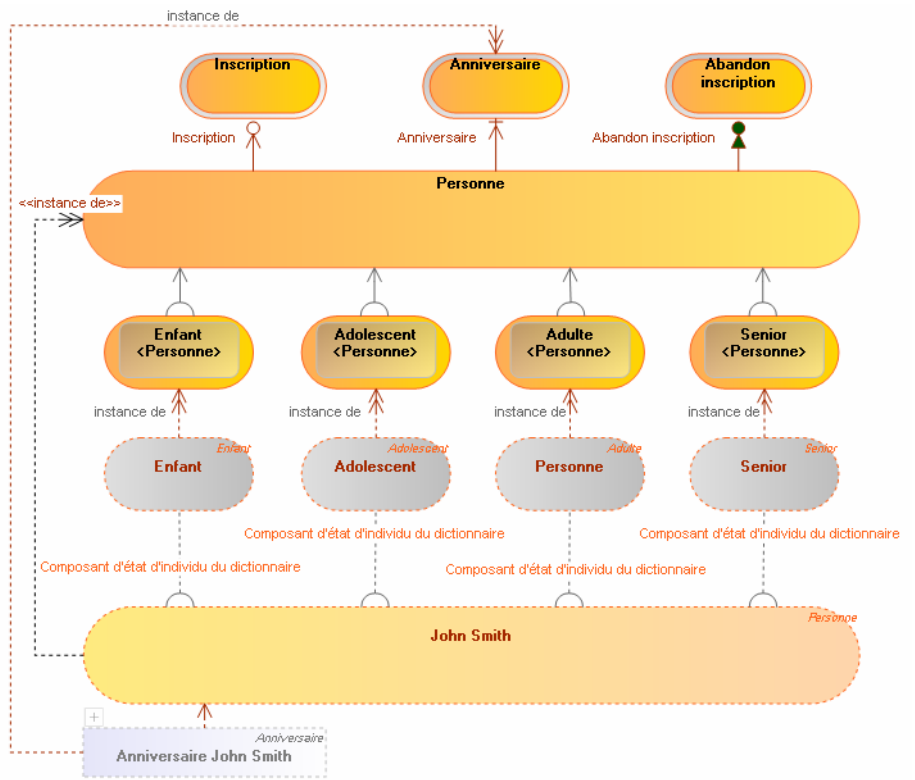
☛ *Pour plus de détails, voir "Décrire les états d'un concept ou d'un individu", page 57.*

Le cycle de vie d'un individu

☛ *Pour plus de détails, voir "Décrire les états et les événements d'un individu", page 63.*

Si un concept est associé à des états et à des événements, les instances de ce concept peuvent également être associées à des événements et des états.

Par exemple, "John Smith" est une "Personne" qui peut passer d'un état à autre le jour de son anniversaire.



Pour représenter la notion d'état d'individu, **HOPEX Information Architecture** propose l'**Etat d'individu**.

Un état d'individu est une instance d'un état du concept auquel l'individu est relié. Il représente un état de l'individu au cours de son cycle de vie.

La relation entre un individu et son **Etat d'individu** est décrite par un **Composant d'état d'individu**.

> 🕮 *Un composant d'état d'individu de permet de relier un individu à un état d'individu.*

Par ailleurs, le passage d'un état d'individu à un autre peut être conditionné par un **Événement d'individu**.

> 🕮 *Un événement d'individu représente un fait se produisant durant la vie de l'individu. C'est une instanciation d'un concept événement du concept auquel l'individu est relié.*

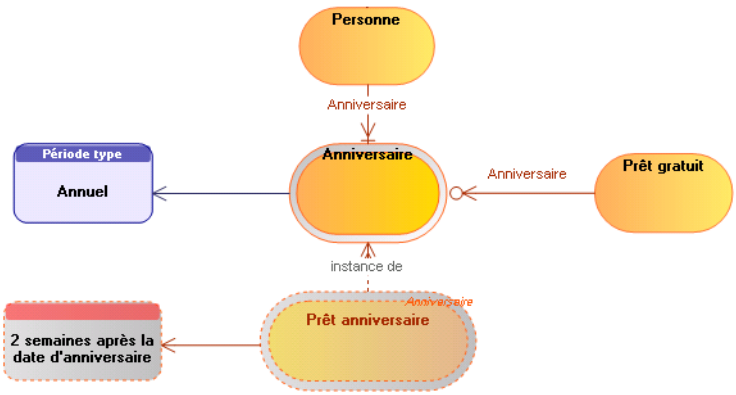
La relation entre un individu et son **Événement d'individu** est décrite par **Composant d'entité**.

> 🕮 *Un composant d'entité permet de relier un individu à un élément de dictionnaire.*

Les périodes

Les **Périodes** permettent d'apporter des précisons temporelles sur les événements.

```
Par exemple, un prêt gratuit peut être proposé aux abonnés à
chaque anniversaire. Ce prêt annuel est valide pendant une
durée de deux semaines.
```



Une **période type** est reliée à un **Concept événement**.

> 🕮 *Un concept événement représente un fait se produisant durant la vie d'un concept, par exemple un changement de saison. Un concept événement permet de marquer l'impact sur un concept d'un phénomène interne ou externe au concept. On peut distinguer les événements de début de concept, les événements de fin de concept et les événements intermédiaires de concept.*

La **période** est reliée à un **Événement d'individu**.

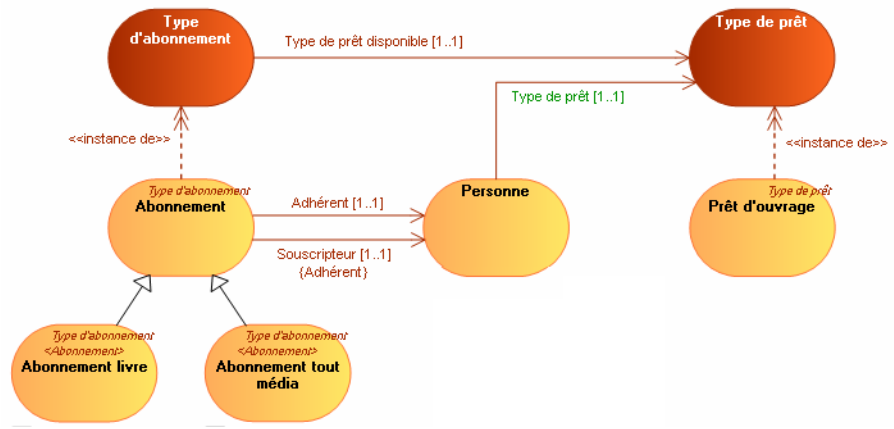
> 🕮 *Un événement d'individu représente un fait se produisant durant la vie de l'individu. C'est une instanciation d'un concept événement du concept auquel l'individu est relié.*

☛ *Pour plus de détails, voir "Utiliser les périodes", page 68.*

La classification des concepts et la notion de type de concept

Un concept type permet de classifier les concepts. Les relations entre les concepts type sont représentées par des composants de concept type.

Par exemple, les "Abonnement" peuvent être classés par "Type d'abonnement". Un "Type d'abonnement" étant caractérisé par un "Type de prêt".



HOPEX Information Architecture fournit les outils pour établir les relations suivantes :

- la relation entre deux **Concept type** est décrite par un **Composant de concept type**.

  Par exemple, un "Type d'abonnement" est caractérisé par un "Type de prêt disponible".

  Un composant de concept type permet de spécifier la relation entre deux concepts type.

- la relation d'appartenance d'un concept à un **Concept type** est décrite par une **Classification de concept**.

  Par exemple, tous les "Abonnements" doivent correspondre à un "Type d'abonnement".

  Une classification de concept permet de relier un concept au concept type qui le caractérise.

- la relation entre un concept et un **Concept type** qui permet de caractériser le concept est décrite par un **Super-composant de concept**.

  Par exemple, chaque "Personne" adhérentes pourrait être caractérisée par un "Type de prêt".

  Un super-composant de concept permet de relier un concept à un concept type pour caractériser une propriété du concept.

La vue de concept

Afin d'obtenir un aperçu conceptualisé de vos objets métier, **HOPEX Information Architecture** propose la notion de **Vue de concept**.

> 🕮 *Une vue de concept permet de représenter le périmètre sémantique couvert par un objet métier. Une vue de concept est construite à partir d'une sélection de plusieurs concepts reliés dans le contexte spécifique de la vue.*

A partir d'un concept de départ, lié à l'objet métier que vous souhaitez décrire, vous parcourez les liens sémantiques qui le définissent. Vous identifiez ainsi plusieurs concepts qui définissent l'objet décrit dans un contexte précis.

> ☛ *Vous pouvez construire différentes vues pour un même objet métier.*
>
> ☛ *Pour plus de détails, voir "Définir des vues de concept", page 75.*

La réalisation des éléments de dictionnaire

Afin d'assurer la cohérence entre les objets qui constituent votre référentiel organisationnel et technique, d'une part, et les concepts métier qui constituent votre dictionnaire, d'autre part, **HOPEX Information Architecture** propose la notion de **Réalisation**.

> 🕮 *Une réalisation de concept relie un objet technique ou organisationnel du référentiel à un élément de dictionnaire.*

Pour plus de détails sur les réalisations, voir la documentation HOPEX IA - Couche logique.

Pour plus de détails sur la génération du dictionnaire, voir "Rapport de glossaire", page 74.

PRÉSENTATION DES DIAGRAMMES DE MODÉLISATION DES CONCEPTS

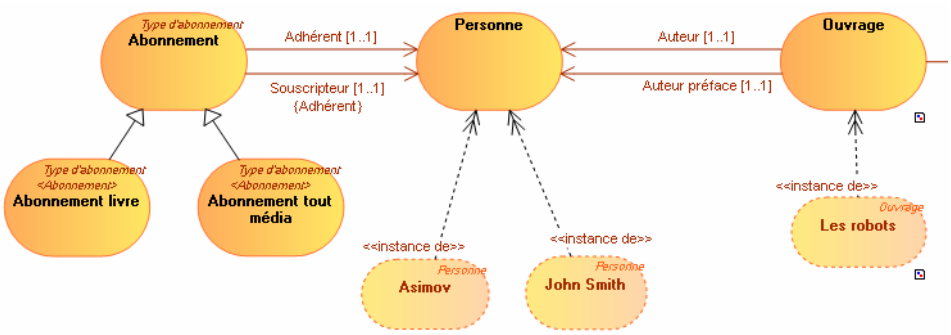
Pour la définition des données métier, **HOPEX Information Architecture** fournit différents types de diagramme.

Le diagramme de concepts

Un domaine d'information métier fournit une vue partielle des modèles ontologiques de l'information métier. Il est décrit par un diagramme de concepts qui présente les concepts, leurs composants, les sur-types et leurs liens.

Le sens des liens fournit un mécanisme naturel de lecture et de déduction du périmètre de "l'objet métier".

Le diagramme du domaine d'information métier suivant présente une vue partielle du domaine de connaissance "Médiathèque".

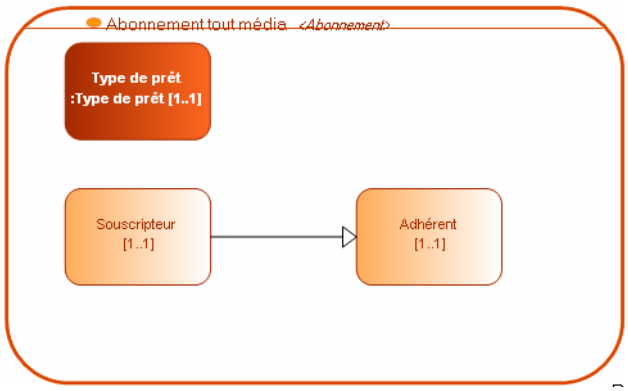


Exemple de domaine d'information métier avec les vues standards

☛ *Pour plus de détails, voir "Construire un diagramme de concepts", page 31.*

Le diagramme de structure de concept

Le contenu des objets métier peut être représenté dans un "Diagramme de structure de concept", qu'il est possible d'initialiser à partir des éléments du graphe de concept.



Exemple de diagramme de structure de concept

☛ *Pour plus de détails, voir "Le diagramme de structure de concept", page 51.*

Le diagramme de structure de concept type

Les types de concept peuvent être représentés dans un "Diagramme de structure de type concept", qu'il est possible d'initialiser à partir des éléments du graphe de concept.

☛ *Pour plus de détails, voir "Le diagramme de structure de concept type", page 74.*

Le diagramme de structure de concept d'état

Les concepts d'état peuvent être représentés dans un "Diagramme de structure de concept d'état", qu'il est possible d'initialiser à partir des éléments du graphe de concept.

☛ *Pour plus de détails, voir "Le diagramme de structure de concept d'état", page 62.*

Le diagramme de structure d'individu

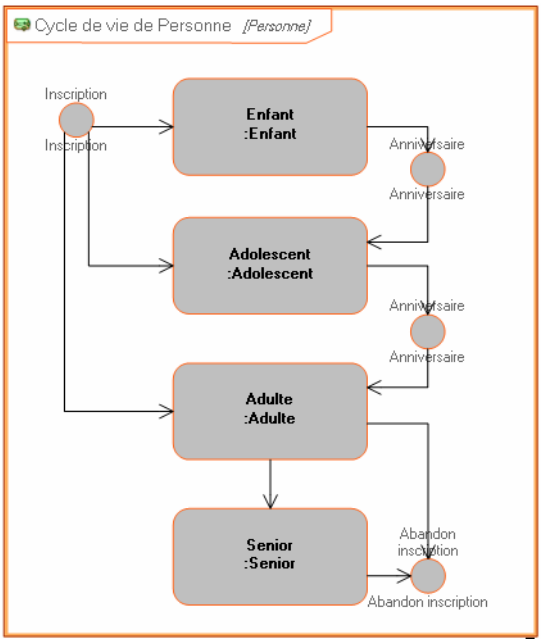
Le diagramme de structure d'individu décrit la structure interne de l'instance de concept et les liens entre tous les composants. Il est possible d'initialiser ce diagramme depuis les éléments du graphe de concept.

☛ *Pour plus de détails, voir "Le diagramme de structure d'individu", page 55.*

Le diagramme de structure de cycle de vie de concept

Le diagramme de structure de cycle de vie de concept permet de décrire l'enchaînement des concepts d'état qui s'opèrent au cours du cycle de vie d'un concept. Chaque concept d'état, que l'on peut considérer comme un point dans le temps, est suivi d'autres concepts d'état.

Le passage d'un état à un autre est modélisé par une transition.



Exemple de diagramme de structure de cycle de vie de concept

☛ *Pour plus de détails, voir "Le diagramme de structure de cycle de vie de concept", page 65.*

GÉNÉRER UN GLOSSAIRE

HOPEX Information Architecture propose un rapport de glossaire prêt à l'emploi pour construire automatiquement le glossaire métier de termes issus d'un ensemble de domaines de connaissance. Pour chaque terme, le glossaire affiche une liste des définitions associées avec leur texte, synonyme et liste de composants.

Lancement d'un rapport de glossaire

Voir "Rapport de glossaire", page 74.

Utiliser le glossaire dans un contexte multilingue

Pour plus de détails, voir "Utiliser HOPEX dans un contexte multilingue" dans le guide HOPEX Common Features.

GÉRER LES ASSIGNATIONS DES DONNÉES MÉTIER

Lors de sa création, un **Domaine de connaissance** ou un **Domaine d'information métier** est assigné automatiquement à une personne. Les autres types d'objet de **HOPEX Information Architecture** peuvent être assignés de manière explicite.

Les objets assignables sont :

- Les domaines de connaissance
- Les concepts
- Les types de concept
- Les concepts d'état
- Les concepts événement
- Les cycles de vie de concept
- Les vues de concept
- Les représentations type

Accéder à l'assignation d'un objet

Pour accéder aux informations qui concernent l'assignation d'un objet :

1. Cliquez sur le menu de navigation puis sur **Information métier** pour accéder à la liste des objets métier.
2. Dans la zone d'édition, cliquez sur le type d'objet, par exemple concepts.
3. Visualisez l'objet en question et dans la colonne **Action** associée, cliquez sur le bouton **Propriétés**.
4. Dans la fenêtre des propriétés de l'objet, sélectionnez la page **Assignation**.

☛ *Seuls les objets assignables ont une page **Assignation**.*

Assignation automatique d'un objet

Lors de la création d'un objet de type **Domaine de connaissance** ou **Domaine d'information métier**, une assignation est automatiquement créée. L'objet est assigné à la personne qui l'a créé avec le rôle **Concepteur de données**.

☛ *Les objets de type **Domaine de connaissance**, ou **Domaine d'information métier**, apparaissent dans les listes **Mes domaines de connaissance**, ou **Mes domaines d'information métier**, de la personne qui les a créés.*

Si un objet de type **Concept**, **Type de concept** ou **Vue de concept** est créé à partir d'un objet de type **Domaine de connaissance** ou **Domaine d'information métier**, l'objet créé est assigné de manière calculée à la personne qui a créé le **Domaine de connaissance**, ou **Domaine d'information métier**. L'objet créé apparaît automatiquement dans les listes **Mes concepts** (ou **Mes types de concept** ou **Mes Vues de concept**) de la personne qui a créé le **Domaine de connaissance**, ou **Domaine d'information métier**.

Assignation explicite d'un objet

Vous pouvez définir explicitement l'assignation d'un objet à une personne existante.

Pour assigner un objet à une personne :

1. Ouvrez la fenêtre de propriétés de l'objet assignable.
2. Sélectionnez la page **Assignation**.
3. Cliquez sur **Nouveau**.
   Une fenêtre **Création d'une assignation** s'ouvre.
4. A partir du champ **Personne ou Groupe de personnes**, cliquez sur **Relier**.
   Une fenêtre de **Connexion** s'ouvre.
5. Recherchez et sélectionnez la personne qui vous intéresse et cliquez sur **Relier**.
6. Dans la fenêtre de **Création d'une assignation**, sélectionnez le **Rôle métier** de la personne que vous venez d'assigner.

   ☛ *Pour plus de détails sur les rôles métier utilisés pour les assignations, voir "Les rôles métier de HOPEX Information Architecture", page 8.*
7. Cliquez sur **OK**.
   Une nouvelle assignation est ajoutée à la liste des assignations associées à l'objet.

Les objets de type **Concept**, **Type de concept** et **Vue de concept** apparaissent dans les listes **Mes concepts**, **Mes types de concept** et **Mes Vues de concept** des personnes auxquelles ils ont été assignés.

DÉCRIRE L'ARCHITECTURE D'INFORMATION MÉTIER

HOPEX Information Architecture permet de décrire l'architecture des informations métier de votre entreprise selon une démarche dont les différentes étapes sont décrites dans ce chapitre.

DÉFINIR UN DOMAINE DE CONNAISSANCE

Un domaine de connaissance représente l'ensemble de ce qui constitue l'objet d'un art, d'une science, d'une discipline ou du champ d'une étude.

Un domaine d'information métier est un sous-ensemble d'éléments d'un domaine de connaissance qui permet de réduire le champ d'une étude.

Le domaine de connaissance constitue l'élément essentiel de la construction de votre architecture d'information.

Les dictionnaires métier peuvent être créés avec le profil **Architecte d'informations métier**.

Les éléments d'un domaine de connaissance

HOPEX Information Architecture vous permet de mettre facilement à jour vos domaines de connaissance à partir des éléments de dictionnaire qui existent déjà : *Terme*, *Concept*, *Concept d'état*, *Concept événement* ou *Vue de concept*.

> ☛ *La liste des éléments d'un domaine de connaissance est accessible à partir de sa fenêtre de propriétés dans l'onglet **Caractéristiques**, sections **Périmètre** et **Instance de périmètre**.*

Un domaine de connaissance vous permet de décrire tous les éléments qui définissent votre architecture d'information :

- Les concepts

  > 🕮 *Un concept événement représente un fait se produisant durant la vie d'un concept, par exemple un changement de saison. Un concept événement permet de marquer l'impact sur un concept d'un phénomène interne ou externe au concept. On peut distinguer les événements de début de concept, les événements de fin de concept et les événements intermédiaires de concept.*
  >
  > ☛ *Pour plus de détails, voir "Décrire un concept", page 39*

- Les variations de concept

  > 🕮 *Une variation décrit comment un concept peut être varié sous une autre forme. La variante est un objet quasi-similaire à l'objet varié mais avec des propriétés ou des relations qui peuvent différer.*
  >
  > ☛ *Pour plus de détails, voir "Décrire les composants d'un concept", page 45*

- Les concepts type

  > 🕮 *Un concept type permet de classifier les concepts. Les relations entre les concepts type sont représentées par des composants de concept type.*
  >
  > ☛ *Pour plus de détails, voir "Décrire un concept type", page 66*

- Les concepts d'état

  > 🕮 *Un concept d'état est une situation au cours de la vie d'un concept durant laquelle il satisfait à certaines conditions, exerce une certaine activité ou attend un événement de concept. Un concept d'état représente un intervalle de temps dont les bornes sont deux*

événements de concept. Un état de concept est une phase par laquelle passe le concept au cours de son cycle de vie.

☛ *Pour plus de détails, voir "Décrire les états d'un concept ou d'un individu", page 53*

- Les concepts événement

  Un concept événement représente un fait se produisant durant la vie d'un concept, par exemple un changement de saison. Un concept événement permet de marquer l'impact sur un concept d'un phénomène interne ou externe au concept. On peut distinguer les événements de début de concept, les événements de fin de concept et les événements intermédiaires de concept.

  ☛ *Pour plus de détails, voir "Décrire les concepts événement", page 56*

- Les individus

  Un individu représente l'occurrence d'un concept.

  ☛ *Pour plus de détails, voir "Décrire les individus", page 49*

- Les états d'individu

  Un état d'individu est une instance d'un état du concept auquel l'individu est relié. Il représente un état de l'individu au cours de son cycle de vie.

  ☛ *Pour plus de détails, voir "Décrire les états d'un concept ou d'un individu", page 53*

Un domaine de connaissance peut être complètement, ou partiellement, décrit par un diagramme de concepts.

☛ *Pour plus de détails sur les graphes de concepts, voir "Définir un domaine d'information métier", page 30*

Créer un domaine de connaissance

Pour créer un *domaine de connaissance* avec **HOPEX Web Front-End** :

1. Cliquez sur le menu de navigation puis sur **Information métier**.
2. Dans la zone d'édition, cliquez sur **Tous les domaines de connaissance**.
3. Cliquez sur **Nouveau**.
   La fenêtre **Création d'un domaine de connaissance** apparaît.
4. Saisissez le **Nom** de votre nouveau domaine et cliquez sur **OK**.
   Le nouveau domaine de connaissance apparaît dans la liste.

☛ *Pour créer un domaine de connaissance avec **HOPEX Windows Front-End**, dans le bureau **Information Architecture**, cliquez sur l'onglet **Bibliothèque/IA** puis sur le volet de navigation **Information Architecture** et dépliez le dossier **Domaines de connaissance** pour obtenir la liste des domaines de connaissance existants dans le référentiel.*

Accéder aux concepts d'un domaine de connaissance

Pour accéder aux concepts d'un domaine de connaissance dans **HOPEX Web Front-End** :

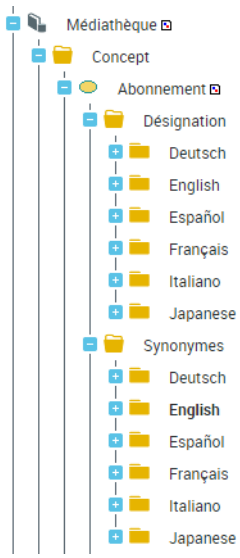
1. Cliquez sur le menu de navigation puis sur **Information métier**.

2. Dans le volet de navigation, cliquez sur **Domaines de connaissance** puis sur **Hiérarchie des informations métier**.
   La liste des domaines de connaissance présents dans le référentiel apparaît.
3. Dépliez le dossier du domaine de connaissance qui vous intéresse.
4. Dépliez le dossier **Concepts**.
   La liste des concepts du domaine de connaissance apparaît.

> ☛ *Les concepts portent le nom du terme associé au concept dans la langue des données. Pour plus de détails, voir "Utiliser le glossaire dans un contexte multilingue", page 16.*

Si vous dépliez le dossier associé à un concept, les termes et synonymes sont accessibles, dans toutes les langues disponibles pour votre environnement **HOPEX**.

> ☛ *Le nombre de langues proposées, à partir des dossiers, dépend de votre environnement **HOPEX**. Pour configurer la liste des langues, voir le guide **HOPEX Power Supervisor**, chapitre "Gérer les options", "Gérer les langues", "Installer des langues supplémentaires.*



Accéder aux termes d'un domaine de connaissance

Pour accéder aux termes d'un domaine de connaissance :

1. Dépliez le dossier du domaine de connaissance qui vous intéresse.
2. Dépliez le dossier **Termes**.
   L'ensemble des termes du domaine apparaît sans distinction de langue.

Créer une carte d'information métier

Une carte d'information métier est un outil d'urbanisation des informations métier. Elle permet de représenter les domaines d'information métier d'un domaine de connaissance et leurs liens de dépendance.

Vous pouvez créer une carte d'information métier à partir d'un projet ou une entreprise afin de cibler le périmètre étudié.

Pour créer la carte d'information métier d'un domaine de connaissance :

1. Cliquez sur le menu de navigation puis sur **Information métier**.
2. Dans le volet de navigation, cliquez sur **Carte d'information métier**.
3. Affichez toutes les cartes d'information métier.
4. Cliquez sur **Nouveau**.
5. Dans la fenêtre qui apparaît, indiquez le nom de la carte et le domaine de connaissance détenteur.
6. Cliquez sur **OK**.

Pour créer le diagramme de la carte d'information métier :

1. Faites un clic droit sur la carte d'information métier et sélectionnez **Nouveau** > C**arte d'information métier**.
   Le diagramme apparaît dans la zone d'édition.

Les composants d'une carte d'information métier

Dans une carte d'information métier vous pouvez ajouter des composants internes et externes.

Les composants internes sont les domaines d'information métier qui font partie du périmètre de la carte d'information métier (qu'ils appartiennent ou non au domaine de connaissance détenteur).

Les composants externes sont ceux qui sont utilisés dans la carte mais qui ne font pas partie du périmètre étudié.

DÉFINIR UN DOMAINE D'INFORMATION MÉTIER

Un domaine d'information métier est un sous-ensemble d'éléments d'un domaine de connaissance qui permet de réduire le champ d'une étude. Il est décrit par un diagramme de concepts.

Les domaines d'information métier peuvent être créés avec le profil **Architecte d'informations métier**.

Créer un domaine d'information métier

Pour créer un domaine d'information métier avec **HOPEX Information Architecture** :

1. Cliquez sur le menu de navigation puis sur **Information métier**.
2. Dans la zone d'édition, cliquez sur **Domaine d'information métier**.
3. Affichez tous les domaines d'information métier.
4. Cliquez sur **Nouveau**.
   La fenêtre de création apparaît.
5. Saisissez le nom du domaine d'information métier.
6. Recherchez et reliez le domaine de connaissance détenteur.
7. Cliquez sur **OK**.

Créer un diagramme de structure d'un domaine d'information métier

Le diagramme de structure définit les sous-domaines du domaine d'information métier et leurs relations.

Pour créer le diagramme de structure d'un domaine d'information métier :

- Faites un clic droit sur le domaine d'information métier et sélectionnez **Nouveau** > **Diagramme de structure du domaine d'information métier**.
  Le diagramme de structure associé au domaine d'information métier s'ouvre dans la fenêtre d'édition.

Construire un diagramme de concepts

Un diagramme de concepts est une représentation graphique des concepts qui sont utilisés dans le contexte d'un domaine d'information métier, ainsi que des liens qui existent entre ces concepts.

Un domaine d'information métier peut être décrit par plusieurs diagrammes de concepts.

Un objet conceptuel appartient au domaine de connaissance à partir duquel il a été créé mais il peut être utilisé/référencé par un domaine d'information métier d'un domaine de connaissance différent.

Voir aussi "Définir un domaine d'information métier", page 30.

Créer un diagramme de concepts d'un domaine d'information métier

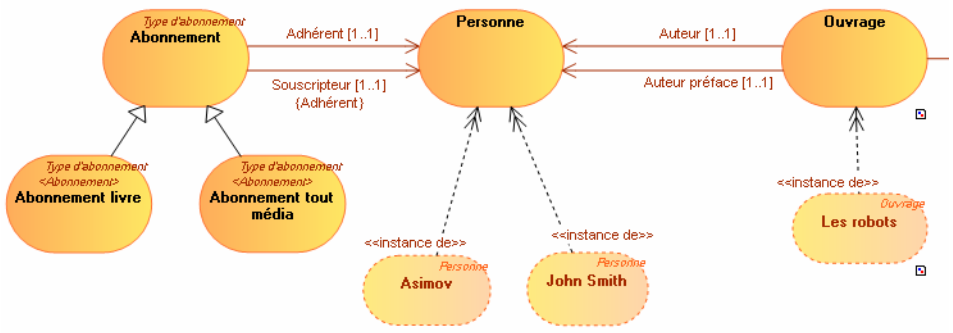
Pour créer le diagramme de concepts d'un domaine d'information métier :

- Faites un clic droit sur le domaine d'information métier et sélectionnez **Nouveau** > **Diagramme de concepts du domaine d'information métier**.
  Le diagramme de concepts s'ouvre dans la fenêtre d'édition.

Les composants d'un diagramme de concepts

Un diagramme de concepts décrit l'architecture de l'information. Par défaut, vous ne visualisez dans le diagramme de concepts que des concepts, des variations et des individus.

```
Le diagramme de concepts suivant décrit partiellement le
domaine de connaissance "Médiathèque".
```



Graphe de concept avec les vues standards

Activer la fenêtre de vue

La fenêtre **Vues et détails** présente une liste de vues (types d'objets à afficher) plus étendue.

Pour activer la fenêtre **Vues et détails** :

1. Dans un diagramme, cliquez sur **Vues et détails**.
   La liste des vues (types d'objets à afficher) apparaît.
2. Sélectionnez ou dé-sélectionnez les vues que vous voulez afficher ou non.

Les vues disponibles pour un domaine d'information métier sont :

- Les concepts,
- Les concepts type,
- Les concepts d'état,
- Les concepts événement,
- Les individus,
- Les états d'individu,
- Les événements d'individu,
- Les vues de concept.

> 🕮 *Une vue de concept permet de représenter le périmètre sémantique couvert par un objet métier. Une vue de concept est construite à partir d'une sélection de plusieurs concepts reliés dans le contexte spécifique de la vue.*
>
> ☛ *Pour plus de détails, voir "Définir des vues de concept", page 71*

Ajouter un élément dans un diagramme de concepts

Pour ajouter, par exemple, un concept existant dans un domaine d'information métier :

1. Dans la barre d'objets du diagramme de concept, cliquez sur le bouton

   Concept.
2. Cliquez dans le diagramme.
   Une fenêtre d'ajout d'un concept s'ouvre et vous demande de choisir le concept.
3. Sélectionnez le concept qui vous intéresse.
4. Cliquez sur **Ajouter**.
   Le concept apparaît dans le diagramme.

> ☛ *Pour plus de détails sur la création de concept, voir "Décrire un concept", page 39*

Utiliser la barre d'insertion de l'objet

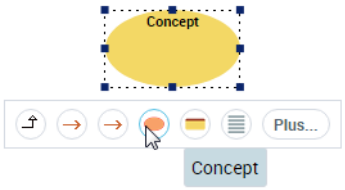
La barre d'insertion disponible sur chaque objet facilite la création d'objets en proposant une aide à la sélection des objets. Elle propose uniquement les objets que vous pouvez relier à l'objet courant.

> ☛ *Cette fonctionnalité est disponible avec **HOPEX Web Front-End** seulement.*

Pour créer, par exemple, un concept à partir d'un concept du diagramme :

1. Cliquez sur le concept du diagramme qui vous intéresse.
   La barre qui contient les objets que vous pouvez insérer à ce stade apparaît.
2. Cliquez sur l'icône qui représente l'objet que vous souhaitez créer.

   Par exemple : **Concept** .



La fenêtre de Choix de lien apparaît.

3. Dans la fenêtre de choix de lien, sélectionnez le type de lien désiré.

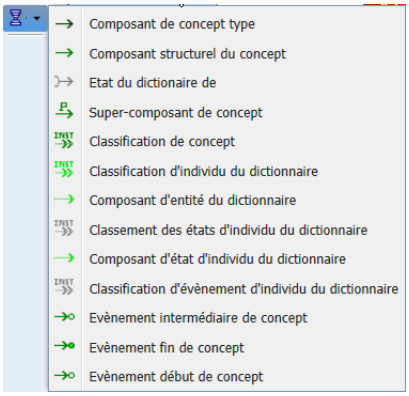
   Par exemple : **Composant structurel de concept**.

4. Cliquez dans le graphe à l'endroit où vous souhaitez poser l'objet.
   L'objet est créé, ainsi que son lien à l'objet de départ.

Synthèse des liens entre les objets

Dans chaque graphe de concept les relations entre les concepts, concepts type et individus de concept sont représentées par des liens.

Le sens des liens fournit un mécanisme naturel de lecture et de déduction du périmètre définissant "l'objet métier".



☛ *Pour plus de détails sur l'accès aux propriétés des liens d'un graphe de concept, voir "Accéder aux propriétés d'un lien dans un diagramme de concepts", page 34.*

Type de lien	Définition et Commentaire
Composant de concept type	*Un composant de concept type permet de spécifier la relation entre deux concepts type.*
Composant structurel de concept	*Un composant structurel de concept permet de représenter une relation de dépendance entre deux concepts. Cette relation est orientée.*
Etat du dictionnaire de	*Un état du dictionnaire de permet de relier un concept à un état de concept et de spécifier la nature de l'état.* Avec la vue "Concept d'état»
Super-composant de concept	*Un super-composant de concept permet de relier un concept à un concept type pour caractériser une propriété du concept.*
Classification de concept	*Une classification de concept permet de relier un concept au concept type qui le caractérise.*
Classification d'individu	*Une classification d'individu permet de relier un individu au concept qui le caractérise.*
Composant d'entité du dictionnaire	*Un composant d'entité permet de relier un individu à un élément de dictionnaire.*
Classification d'état d'individu	*Une classification d'état d'individu permet de relier un état d'individu au concept d'état qui le caractérise.* Ce lien est proposé avec la vue "Etat d'individu".
Composant d'état d'individu	*Un composant d'état d'individu de permet de relier un individu à un état d'individu.* Ce lien est proposé avec la vue "Etat d'individu".
Classification d'événement d'individu	*Une classification d'événement d'individu permet de relier un individu au concept d'événement qui le caractérise.* Ce lien est proposé avec la vue "Etat d'individu".
Événement intermédiaire de concept	*Un concept événement représente un fait se produisant durant la vie d'un concept, par exemple un changement de saison. Un concept événement permet de marquer l'impact sur un concept d'un phénomène interne ou externe au concept. On peut distinguer les événements de début de concept, les événements de fin de concept et les événements intermédiaires de concept.* Ces liens sont proposés avec la vue "Concept événement".
Événement fin de concept	
Événement début de concept	

Accéder aux propriétés d'un lien dans un diagramme de concepts

Dans un diagramme de concepts, les liens sont orientés et donnent accès à la fois aux propriétés du lien et à l'objet cible du lien.

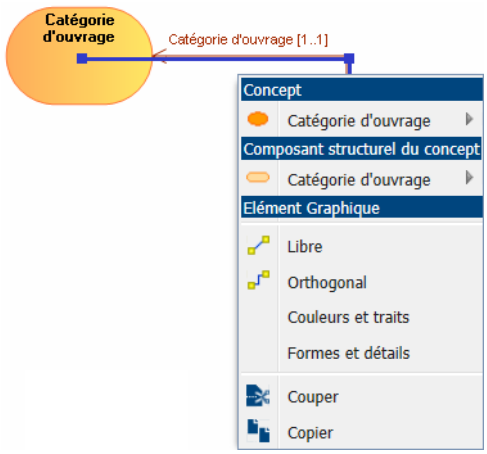
☛ *Pour plus de détails sur la liste des liens disponibles dans un domaine d'information métier, voir "Synthèse des liens entre les objets", page 33.*

Le menu contextuel d'un lien de type **Composant structurel de concept**, par exemple, présente :

- les commandes propres au type d'objet utilisé par le composant

  `par exemple` **`Concept`**
- les commandes relatives au composant lui-même

  `par exemple` **`Composant structurel de concept`**
- les commandes relatives au graphisme.



Pour accéder aux propriétés d'un lien de type "composant" :

`Par exemple` **`Composant structurel de concept`**

1. Faites un clic droit que le lien pour ouvrir son menu contextuel.
2. Sélectionnez le lien puis cliquez sur **Propriétés**.
   La fenêtre de propriétés du lien s'ouvre.

L'onglet **Caractéristiques** de la fenêtre de propriétés du lien présente plusieurs informations.

- Le **Nom Local** du lien qui correspond par défaut à l'élément de dictionnaire cible ou au terme associé au lien.

  ☛ *Pour plus de détails sur l'association d'un terme à un lien, voir "Décrire les composants d'un concept", page 45.*

- Le **Concept composé** ciblé par le lien.
- Le **Détenteur** qui est l'élément de dictionnaire à l'origine du lien.
- La **Multiplicité minimale** qui est le nombre d'éléments origine qui peuvent accéder au même éléments cible.

  ```
  Par exemple, combien d'"Ouvrages" peuvent appartenir à une
  même "Catégorie d'ouvrage".
  ```

- La **Multiplicité maximale** qui est le nombre d'éléments cibles qui peuvent être liés à un même élément origine.

  ```
  Par exemple, un "Ouvrages" ne peut appartenir qu'à une seule
  "Catégorie d'ouvrage".
  ```

- La case **Type abstrait du dictionnaire** qui précise le caractère concret ou abstrait d'un concept.
- La **Propriété du périmètre du dictionnaire** qui peut prendre les valeurs suivantes :
  - "Référencement" : pour signifier que le concept cible est seulement référencé par le lien,
  - "Embarqué" : pour signifier que le concept cible a son existence propre, mais fait partie intégrante du concept qui est à l'origine du lien
  - "Composite" : pour signifier que le concept cible est un composant du concept qui est à l'origine du lien, si le concept cible est détruit, le composite l'est également.
- La **Désignation** du lien et le champ **Texte de la définition** qui permettent d'associer un terme et une définition au lien.

  ☛ *Pour plus de détails sur l'association d'un terme à un lien, voir "Décrire les composants d'un concept", page 45.*

- Les **Sur-types** qui permettent d'accéder aux propriétés d'un lien héritées d'un type de concept.

  ☛ *Pour plus de détails, voir "Décrire un concept type", page 66.*

- La **Réalisation** qui permet d'associer cet élément de dictionnaire à des éléments de l'architecture applicative.

  ☛ *Pour plus de détails, voir le guide **HOPEX Logical Data**.*

- Les **Synonymes** qui permettent de spécifier une liste de synonymes.

  ☛ *Pour plus de détails, voir "Décrire les composants d'un concept", page 45 et "Concept et terme", page 3.*

Pour plus de détails sur la définition des concepts, voir "Décrire un concept", page 39.

Gérer les composants d'un domaine d'information métier

HOPEX Information Architecture vous permet de mettre à jour vos domaines de connaissance à partir des *Domaines d'information métier* et des éléments de dictionnaire qui existent déjà : *Terme*, *Concept*, *Concept d'état*, *Concept événement* ou *Vue de concept*.

Les composants d'un domaine d'information métier

La liste des éléments d'un domaine d'information métier qui appartiennent réellement au domaine d'information sont accessibles à partir de la fenêtre de propriétés du domaine, dans la page **Composant**.

Créer un composant de domaine d'information métier

Pour définir qu'un concept existant est un composant de domaine d'information métier :

1. Ouvrez la fenêtre de propriétés du domaine d'information métier.
2. Sélectionnez la page **Composant**.
3. Cliquez sur **Nouveau**.
   L'assistant de création d'une information métier apparaît.
4. En face du champ **Information métier mémorisée** cliquez sur **Relier**.
   Une fenêtre de connexion s'ouvre.
5. Sélectionnez le concept qui vous intéresse et cliquez sur **Relier**.
6. Dans la fenêtre de création d'une information métier, cliquez sur **OK**.
   Les concept est ajouté à la liste des composants du domaine d'information métier.

Relier ou supprimer un composant à partir d'un diagramme de concepts (HOPEX Web Front-End)

Pour relier un élément de dictionnaire à la liste des composants d'un domaine d'information métier :

1. Ouvrez le diagramme de concepts associé au domaine d'information métier.
2. Ajoutez l'élément de dictionnaire qui vous intéresse dans le diagramme.
3. Faites un clic droit sur cet élément pour faire apparaître son menu contextuel.
4. Sélectionnez **Ajouter à "Nom du domaine d'information métier courant"**.
   L'élément est ajouté à la liste des éléments du domaine d'information métier, dans la page **Composant** de la fenêtre de propriétés du domaine.
   La forme change dans le diagramme.

Pour supprimer un élément de dictionnaire d'un domaine d'information métier :

1. Faites un clic droit sur l'élément de dictionnaire qui vous intéresse pour faire apparaître son menu contextuel.
2. Sélectionnez **Supprimer de "Nom du domaine d'information métier courant"**.
   L'élément est supprimé de la liste de composants du domaine d'information métier.

Définir le CRUD sur les composants d'un domaine d'information métier

Il est possible de spécifier les droits d'accès à chacun des types de composants d'un domaine d'information métier. Pour ce faire, il faut cocher ou décocher les cases de chaque colonne associées aux actions : Create, Read, Update Delete.

Le contenu de la colonne **Accès au données** est calculé automatiquement en fonction des actions cochées. Ce résultat apparaît sur la forme de l'objet dans le diagramme de concepts associé au domaine d'information métier.

☛ *Pour plus de détails sur les composants d'un domaine d'information métier, voir "Les composants d'un domaine d'information métier", page 37.*

DÉCRIRE UN CONCEPT

Un concept représente la détermination de ce qu'est un être, une chose ou un mot, par ses propriétés et caractéristiques essentielles ou ses qualités propres.

Le concept est l'élément de dictionnaire de base proposé par **HOPEX Information Architecture**.

Accéder à la liste des concepts

En Web Front-End

Pour accéder à l'ensemble des concepts de votre référentiel :

1. Cliquez sur le menu de navigation puis sur **Information métier**.
2. Dans le volet **Information métier**, cliquez sur **Concepts > Tous les Concepts**.
   La liste des concepts s'affiche.

   ☛ *Pour plus de détails sur l'utilisation de la liste des concepts du référentiel, voir "Définir un domaine de connaissance", page 26.En*

En Windows Front-End

Pour accéder à l'ensemble des concepts de votre référentiel :

3. Sélectionnez l'onglet **Bibliothèque/IA > Eléments de dictionnaire > Gestion des domaines de connaissance.**
   La liste des éléments de dictionnaire apparaît à gauche de la zone d'édition et l'arbre des domaines de connaissance apparaît à droite de la zone d'édition.
4. Dans la partie gauche de la zone d'édition, sélectionnez l'onglet **Concept.**

Créer un concept

Pour créer un *concept* à partir d'un domaine de connaissance :

1. Dans le volet **Information métier**, cliquez sur **Domaines de connaissance > Hiérarchie des informations métier**.
   L'arbre des domaines de connaissance existants dans le référentiel apparaît.
2. Faites un clic droit sur le domaine de connaissance qui vous intéresse et cliquez sur **Nouveau > Concept**.
   L'assistant de création d'un concept apparaît.
3. Renseignez le **Nom local** et cliquez sur **Suivant**.

4. Dans la section **Terme**, le tableau **Termes existants** dresse la liste des termes portant le même nom que le nouveau concept.

   📖 *Un terme est un mot ou groupe de mots considéré dans sa valeur de désignation, en particulier dans un vocabulaire spécialisé.*

   ☛ *Si un terme a déjà été créé avec le même nom que le nouveau concept, ce terme est automatiquement relié et il apparaît dans la section **Terme**.*

5. Dans le champ **Texte de la définition**, saisissez le texte de la définition du concept et cliquez sur **OK**.
   Le nom du nouveau concept apparaît dans l'arborescence sous le domaine de connaissance.

   ☛ *Un nouveau terme portant le même nom que le concept est également créé.*

Les propriétés d'un concept

Les caractéristiques d'un concept

L'onglet **Caractéristiques** de la fenêtre de propriétés d'un concept permet d'accéder aux principales caractéristiques du concept.

Le concept est décrit par :

- la case **Concept abstrait** qui précise le caractère concret ou abstrait d'un concept,
- sa **Désignation** qui est représentée par un ou plusieurs termes,

  ☛ *Pour modifier le nom d'un concept dans une langue donnée, vous devez accéder aux propriétés du concept et modifier le nom du terme qui lui est associé dans la langue concernée. Pour plus de détails, voir "Concept et terme", page 3.*
- le **Texte de sa définition**,
- La section **Synonymes** permet de spécifier une liste des concepts qui lui sont synonymes,

  ```
  Par exemple, dans le domaine financier, le terme "Avance"
  est reconnu comme synonyme du terme "Acompte".
  ```

  📖 *Un synonyme est un terme qui est interchangeable avec un autre dans le contexte d'un concept de ce terme avec le même sens, ou presque.*
- La section **Sur-types** permet d'accéder aux types de concept qui classifient le concept courant,

  ☛ *Pour plus de détails, voir "Décrire un concept type", page 66.*
- La section **Réalisation** permet d'associer au concept un élément de l'architecture applicative,

  ☛ *Pour plus de détails, voir le guide **HOPEX Logical Data**.*

Les liens entre un concept et les autres éléments de dictionnaire

Outre les caractéristiques de terminologie, un concept est caractérisé par ses relations avec les autres éléments de dictionnaire.

- L'onglet **Variation** présente les concepts dont les propriétés sont héritées par le concept décrit, pour plus de détails voir "Décrire les variations d'un concept", page 44.
- L'onglet **Composant** présente :
  - la liste des composants structurels de concept détenus, pour plus de détails voir "Décrire les composants d'un concept", page 45.
  - la liste des Super composants de concept, pour plus de détails voir "Décrire le super composant d'un concept", page 42.

> ☛ *Les Concepts d'état reliés à un concept ne sont pas présentés dans la fenêtre de propriétés, pour plus de détails voir "Décrire les états d'un concept ou d'un individu", page 53.*

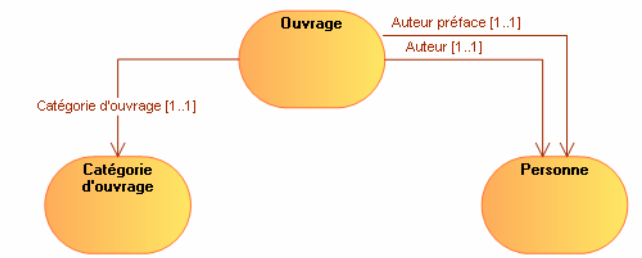
Décrire les composants structurels d'un concept

Avec **HOPEX Information Architecture**, il est possible de relier un concept à un autre concept pour le caractériser.

```
Par exemple, le concept d'"Ouvrage" est relié au concept de
"Personne" pour caractériser l'"Auteur" d'un ouvrage.
```

Cette relation est décrite par un **Composant structurel de concept** qui peut, éventuellement, être associé à un terme.

> *Un composant structurel de concept permet de représenter une relation de dépendance entre deux concepts. Cette relation est orientée.*



Accéder aux composants structurels d'un concept

Pour accéder aux composants structurels d'un concept :

1. Ouvrez la fenêtre de propriétés d'un concept.
2. Sélectionnez l'onglet **Composants**.
3. Dépliez la section **Composants Structurels**.
   La liste des composants structurels détenus apparaît.

> ☛ *Vous pouvez également consulter la liste des composants structurels d'un concept à partir de son diagramme de structure de concept. Pour plus de détails, voir "Le diagramme de structure de concept", page 47.*

Créer un composant structurel de concept à partir d'un diagramme

La procédure de création de composant structurel de concept "Auteur" entre les concepts "Ouvrage" et "Personne" est décrite à titre d'exemple.

Pour créer un composant structurel de concept entre deux concepts d'un domaine d'information métier :

1. Dans le graphe de concept associé au domaine d'information métier, cliquez sur le concept qui détient le lien.

   ☛ *Si vous êtes en **HOPEX Windows Front-End**, passez la souris sur le concept qui détient le lien et cliquez sur le signe +.*

2. Sélectionnez **Composant structurel de concept**.
3. Cliquez sur le concept cible.
   L'assistant de création d'un composant structurel de concept apparaît.
4. Renseignez le **Nom local**, par exemple "Auteur".
5. Etant donné que le terme "Auteur" doit être créé, cochez la case "Création avec terme".
   La section **Terme** apparaît dans la fenêtre de création.

   📖 *Un terme est un mot ou groupe de mots considéré dans sa valeur de désignation, en particulier dans un vocabulaire spécialisé.*

6. Dans le champ **Texte de la définition**, saisissez le texte de la définition du composant structurel de concept et cliquez sur **OK**.
   Le composant structurel de concept apparaît dans le graphe.

   ☛ *Un nouveau terme portant le même nom que le composant structurel de concept est également créé.*

Vous pouvez également créer un composant structurel de concept dans un diagramme de structure de concept. Dans ce cas là, vous devez préciser le concept cible dans l'assistant de création d'un composant structurel de concept.

☛ *Pour plus de détails, voir "Le diagramme de structure de concept", page 47.*

Décrire le super composant d'un concept

De même qu'un **Concept** peut être caractérisé par un lien vers un autre concept, un concept peut être caractérisé par un lien vers un **Concept type**.

📖 *Un concept type permet de classifier les concepts. Les relations entre les concepts type sont représentées par des composants de concept type.*

Par exemple, chaque "Personne" adhérente pourrait être caractérisée par un "Type de prêt".

☛ *Pour plus de détails, voir "Décrire un concept type", page 66.*

La relation entre un **Concept** et un **Concept type** est décrite par un **Super-composant de concept**.

📖 *Un super-composant de concept permet de relier un concept à un concept type pour caractériser une propriété du concept.*

Pour créer un **Super-composant de concept** entre un concept et un concept type dans un diagramme de domaine d'information métier :

1. Dans la barre d'insertion, cliquez sur le bouton **Lien**.
2. Cliquez sur le concept qui détient le lien.

   `Par exemple, "Personne"`
3. Cliquez sur le concept type cible.

   `Par exemple, "Type de prêt".`

   L'assistant de création d'un Super-composant de concept apparaît.
4. Renseignez le **Nom local**.
5. Si aucun terme n'est à créer, cochez la case "Création sans terme".
6. Cliquez sur **OK**.

   Le Super-composant de concept apparaît dans le diagramme.

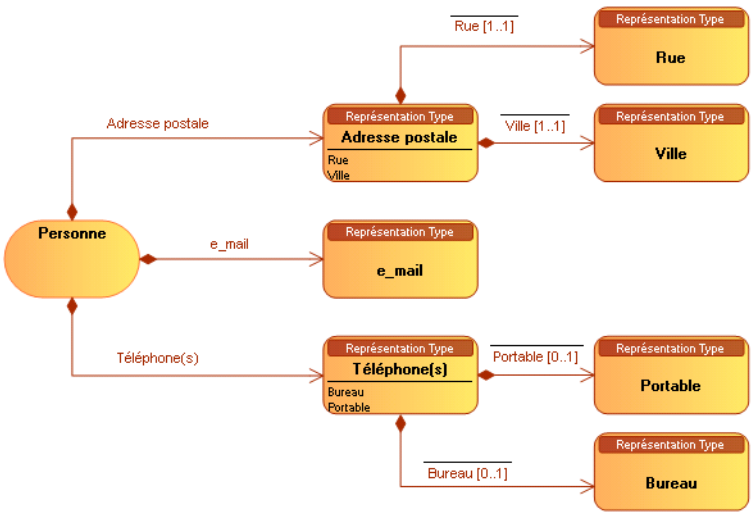
Utiliser les représentations types

Afin de décrire les éléments concrets attachés à un concept, **HOPEX Information Architecture** vous permet de lier un concept à des **représentations types**.

`Par exemple, une personne est associée à une adresse postale obligatoire et unique, une adresse e_mail éventuellement et un ou plusieurs numéros de téléphone.`

Une **représentation type** peut, elle-même, être reliée à d'autres représentations types.

`Par exemple, l'adresse postale est définie à partir du nom de la rue et du nom de la ville.`



Relier une représentation type à un concept

Avec **HOPEX Information Architecture**, le lien entre un concept et une représentation type est décrit par une **Représentation de concept** qui peut, éventuellement, être associé à un terme.

> 🕮 *Une représentation de concept permet de spécifier la relation entre un concept et une représentations type.*

Relier deux représentations types

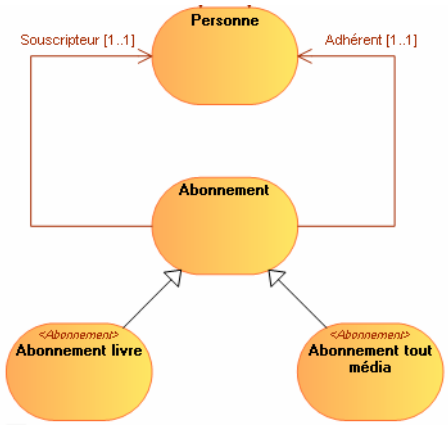
Avec **HOPEX Information Architecture**, le lien entre un concept et une représentation type est décrit par un **composant de représentation type** qui peut, éventuellement, être associé à un terme.

> 🕮 *Un composant de représentation type permet de spécifier la relation entre deux représentations type.*

Décrire les variations d'un concept

Certains concepts métier sont des déclinaisons d'autres concepts : ils sont définis par les mêmes concepts.

> Par exemple, le concepts "Abonnement" est décliné en "Abonnement livre" et "Abonnement tout média". Ces deux types d'abonnement héritent des liens "Souscripteur" et "Adhérent" du spécifié au niveau du concept "Abonnement".



Cette relation est décrite par une **Variation**.

> 🕮 *Une variation décrit comment un concept peut être varié sous une autre forme. La variante est un objet quasi-similaire à l'objet varié mais avec des propriétés ou des relations qui peuvent différer.*
>
> ☛ *Pour plus de détails sur les variations et les substitutions, voir le guide **HOPEX Common Features**, chapitre "Manipuler les objets du référentiel", "Les variations d'objets".*

Accéder aux variations d'un concept

Pour accéder aux variations d'un concept :

1. Ouvrez la fenêtre de propriétés d'un concept.
2. Sélectionnez l'onglet **Variation**.
   La liste des variations associées au concept apparaît.

Créer une variation d'un concept à partir d'un diagramme de concept

Vous pouvez spécifier qu'un concept hérite des caractéristiques définies pour un autre concept.

```
Par exemple, le concept "Abonnement livre" hérite du
concept "Abonnement".
```

Pour spécifier qu'un concept est une variation d'un autre concept à partir d'un domaine d'information métier :

1. Dans la barre d'insertion, cliquez sur le bouton **Variation**.
2. Cliquez sur le concept à varier, et faites glisser la souris jusqu'au nouveau concept, avant de relâcher votre pression.
3. Renseignez le **Nom** et cliquez sur **Ajouter**.
   Un lien fléché du concept à varier vers le concept racine apparaît.

   ☛ *La variation est représentée par un lien mais il s'agit d'un objet **HOPEX**.*

   L'assistant de création d'une variation apparaît.

Créer une variation d'un composant structurel de concept

Il est également possible de créer une **Variation** entre deux **Composants structurels de concept**.

```
Par exemple, le "Souscripteur" est aussi un "Adhérent".
```

Pour définir une variation entre deux composants structurels de concept, il faut qu'ils soient reliés aux mêmes concepts, soit directement soit à travers des variations.

Pour créer une variation entre deux composants structurels de concept :

1. Ouvrez la fenêtre de propriétés du composant structurel de concept à varier.
2. Sélectionnez l'onglet **Variation**.
3. Cliquez sur le bouton **Nouveau**.
   L'assistant de création d'une variation s'ouvre.
4. Sélectionnez les options :
   - "Initialisation des attributs"
   - "Initialisation des diagrammes" pour que la variation apparaisse dans les diagrammes.
5. Cliquez sur le bouton **OK**.
   La variations est crée.

   ☛ *Une variation entre deux **Composants structurels de concept** est représentée graphiquement dans un diagramme de structure de concept. Pour plus de détails, voir "Le diagramme de structure de concept", page 47.*

Créer une substitution d'un composant structurel de concept

Si, au contraire d'une variation, un lien est une redéfinition d'un autre lien, vous allez créer une **substitution**.

> 🕮 *Une substitution détermine quel élément peut être utilisé à la place d'un autre ou est effectivement remplacé par un élément existant dans un context donné (par exemple dans le cadre d'une variation). Contrairement à la variation, une substitution n'induit pas d'héritage mais une équivalence fonctionnelle.*
>
> ☛ *Pour plus de détails sur les variations et les substitutions, voir le guide **HOPEX Common Features**, chapitre "Manipuler les objets du référentiel", "Les variations d'objets".*

Pour définir une substitution entre deux composants structurels de concept, il faut qu'ils soient reliés aux mêmes concepts, soit directement soit à travers des variations.

Pour créer une substitution entre deux composants structurels de concept à partir d'un diagramme de structure de concept :

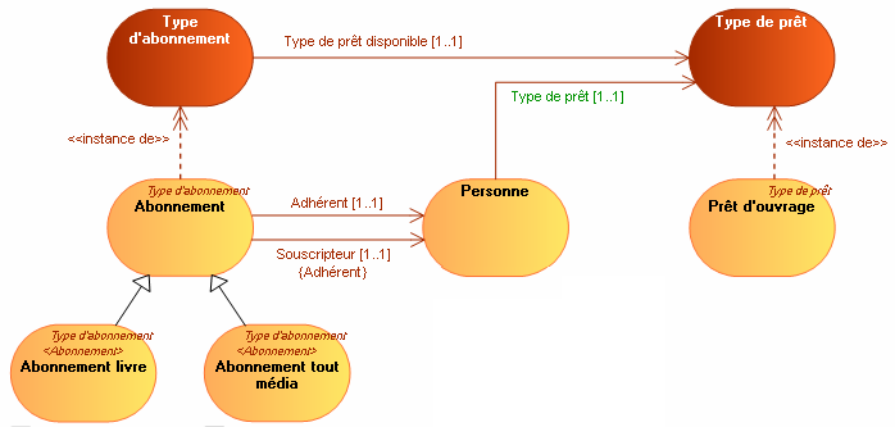
1. Dans la barre d'insertion, cliquez sur le bouton **Substitution**.
2. Cliquez sur le composants structurels à substituer, et faites glisser la souris jusqu'au composants structurels substituant, avant de relâcher votre pression.
3. Renseignez le **Nom** et cliquez sur **Ajouter**.
   Un lien en pointillé et fléché du composant structurel à substituer vers le composant structurel substituant apparaît.

   > ☛ *La substitution est représentée par un lien mais il s'agit d'un objet **HOPEX**.*

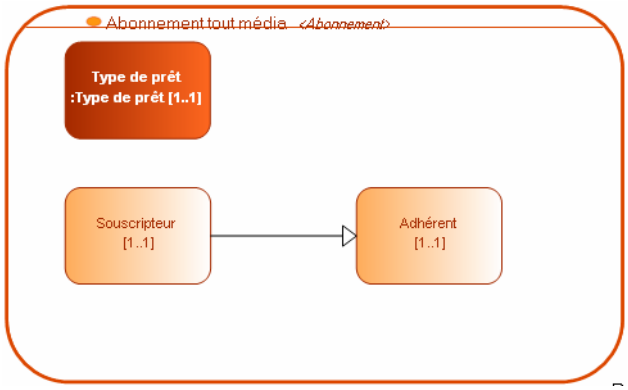
Le diagramme de structure de concept

Dans **HOPEX Information Architecture**, un diagramme de structure de concept rassemble l'ensemble des informations relatives au concept. Ce diagramme est initialisé à partir de des éléments du graphe de concept.

Par exemple, les "Abonnement" peuvent être classés par "Type d'abonnement".



Un "Type d'abonnement" est caractérisé par un "Type de prêt".



Ce diagramme est composé de :

- *variation* entre les composants,

  Par exemple, les "Abonnement" peuvent être classés par "Type d'abonnement". Un "Type d'abonnement" étant caractérisé par un "Type de prêt".

  Une variation décrit comment un concept peut être varié sous une autre forme. La variante est un objet quasi-similaire à l'objet varié mais avec des propriétés ou des relations qui peuvent différer.

  ☛ *Pour plus de détails, voir "Créer une variation d'un composant structurel de concept", page 45.*

- *substitution* entre les composants,

  Une substitution détermine quel élément peut être utilisé à la place d'un autre ou est effectivement remplacé par un élément existant dans un context donné (par exemple dans le cadre d'une variation). Contrairement à la variation, une substitution n'induit pas d'héritage mais une équivalence fonctionnelle.

  ☛ *Pour plus de détails, voir "Créer une substitution d'un composant structurel de concept", page 46.*

- *Composant structurel de concept* qui décrit la relation entre deux **Concepts**,

  Par exemple, un "Type d'abonnement" est caractérisé par un "Type de prêt disponible".

  Un composant structurel de concept permet de représenter une relation de dépendance entre deux concepts. Cette relation est orientée.

  ☛ *Pour plus de détails, voir "Décrire les composants d'un concept", page 45.*

- *Super-composant de concept* qui permet de caractériser le concept à partir d'un **Concepts type**,

  Par exemple, chaque "Personne" adhérentes pourrait être caractérisée par un "Type de prêt".

  Un super-composant de concept permet de relier un concept à un concept type pour caractériser une propriété du concept.

  ☛ *Pour plus de détails, voir "Décrire les variations d'un concept type", page 69.*

- *Evénement de début*, *Evénement intermédiaire* et *Evénement de fin* qui permettent de définir les événements qui contribuent au changement d'état d'un concept,

  Par exemple, le changement de état d'un adhérent peut être provoqué par son anniversaire.

  Un concept événement représente un fait se produisant durant la vie d'un concept, par exemple un changement de saison. Un concept événement permet de marquer l'impact sur un concept d'un phénomène interne ou externe au concept. On peut distinguer les événements de début de concept, les événements de fin de concept et les événements intermédiaires de concept.

  ☛ *Pour plus de détails, voir "Décrire les concepts d'état", page 53.*

DÉCRIRE LES INDIVIDUS

HOPEX Information Architecture fait la distinction entre un concept et les occurrences qui le caractérisent.

📖 *Un individu représente l'occurrence d'un concept.*

Les facilités qui permettent de gérer les individus sont décrites ici :

Accéder à la liste des individus

Pour accéder à l'ensemble des individus de votre référentiel avec **HOPEX Web Front-End** :

1. Dans le volet **Information métier**, cliquez sur **Domaines de connaissance > Hiérarchie des informations métier**.
   L'arbre des Domaines de connaissance s'affiche.

   ☛ *Pour accéder aux individus d'un domaine de connaissance avec **HOPEX Windows Front-End**, ans le bureau **Information Architecture**, cliquez sur l'onglet **Bibliothèque/IA** puis sur le volet de navigation **Information Architecture**.*
2. Dépliez le dossier **Domaines de connaissance**.
3. Dépliez le dossier du domaine de connaissance qui vous intéresse.
4. Dépliez le dossier **Individus**.
   La liste des individus du domaine de connaissance apparaît.

Créer un individu à partir d'un domaine de connaissance

Pour créer un individu à partir d'un domaine de connaissance :

1. Faites un clic droit sur le domaine de connaissance qui vous intéresse et cliquez sur **Nouveau > Individu**.
   L'assistant de création d'un individu apparaît.
2. Renseignez le **Nom local** et cliquez sur **Suivant**.
3. Dans la section **Classification d'individu**, vous pouvez cliquer sur le bouton **Nouveau** pour préciser le concept auquel l'individu est relié.

   ☛ *Pour plus de détails, voir "Créer une classification d'individu", page 50.*
4. Cliquez sur **OK**.
   Le nom du nouvel individu apparaît dans l'arborescence sous le domaine de connaissance.

Les propriétés d'un individu

La fenêtre de propriétés d'un individu présente dans l'onglet **Caractéristiques** les éléments suivants :

- Son **Nom Local**.
- Les classifications d'individu qui apparaissent dans la section **Classification**.

  Une classification d'individu permet de relier un individu au concept qui le caractérise.

  Pour plus de détails, voir "Créer une classification d'individu", page 50.

Les autres onglets de la fenêtre de propriétés d'un individu sont :

- La section **Etat d'individu** qui permet de présenter les différents états d'un individu.

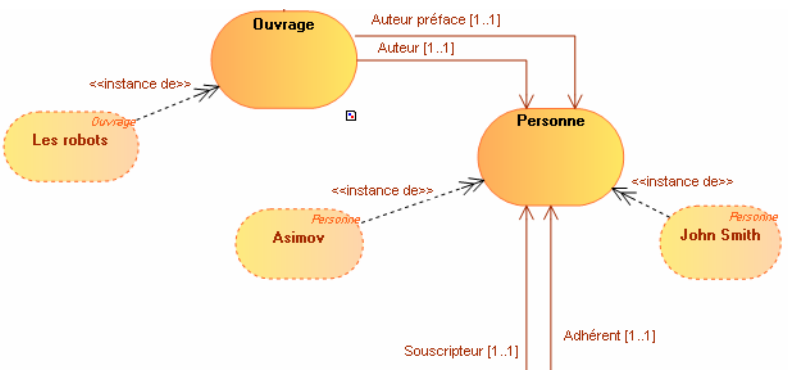
  Pour plus de détails, voir "Décrire les états et les événements d'un individu", page 59.
- L'onglet **Composant** qui présente les individus qui définissent à l'individu décrit.

  Pour plus de détails, voir "Créer un composant d'entité de dictionnaire", page 51.

Créer une classification d'individu

Une classification d'individu permet de relier un individu au concept qui le caractérise.

```
Par exemple, l'individu "Asimov" est une instance de
"Personne" et "Les Robots" est une instance d'"Ouvrage".
```



Pour créer une classification d'individu :

1. Ouvrez la fenêtre de propriétés de l'individu qui porte la relation.

   ```
   Par exemple, l'individu "Asimov".
   ```
2. Sélectionnez l'onglet **Caractéristique.**

3. Dans la section **Classification** cliquez sur le bouton **Nouveau**.
   L'assistant de création d'une classification d'individu s'ouvre.
4. A gauche du champ **Elément caractérisant**, cliquez sur le bouton **Relier**.
   L'assistant de recherche s'ouvre s'ouvre.
5. Sélectionnez le concept que vous voulez relier.

   `Par exemple, le concept "Personne".`
6. Cliquez sur le bouton **OK**.
   La classification d'individu en individu est créée.

Créer un composant d'entité de dictionnaire

Un composant d'entité permet de relier un individu à un élément de dictionnaire.

HOPEX Information Architecture permet également de relier deux individus en individus par une relation de type **Composant d'entité du dictionnaire**.

`Par exemple, vous pouvez préciser que "Asimov" est l'auteur de l'ouvrage "Les Robots".`

Pour créer un composant d'entité du dictionnaire entre deux individus :

1. Ouvrez la fenêtre de propriétés de l'individu qui porte la relation.

   `Par exemple, l'individu "Asimov".`
2. Sélectionnez l'onglet **Composant.**
3. Cliquez sur le bouton **Nouveau**.
   L'assistant de création d'un composant d'entité du dictionnaire s'ouvre.
4. A gauche du champ **Elément caractérisant**, cliquez sur le bouton **Relier**.
   L'assistant de recherche s'ouvre s'ouvre.
5. Sélectionnez l'individu que vous voulez relier.

   `Par exemple, l'individu "Les Robots".`
6. Cliquez sur le bouton **OK**.
   Le composant d'entité est créé. Il apparaîtra dans le diagramme de structure d'individu de l'objet décrit.

☛ *Pour plus de détails, voir "Le diagramme de structure d'individu", page 51.*

Le diagramme de structure d'individu

Avec **HOPEX Information Architecture**, un diagramme de structure d'individu décrit la structure interne de l'instance de concept et les liens entre ses composants. Ce diagramme est initialisé à partir de des éléments du graphe de concept.

Ce diagramme est composé de *composants d'entité du dictionnaire* qui permettent de relier deux individus.

```
Il devient alors possible de préciser que "Asimov" est
l'auteur de l'ouvrage "Les Robots".
```

📖 *Un composant d'entité permet de relier un individu à un élément de dictionnaire.*

☛ *Pour plus de détails, voir "Créer un composant d'entité de dictionnaire", page 51.*

DÉCRIRE LES ÉTATS D'UN CONCEPT OU D'UN INDIVIDU

Un objet métier peut avoir un cycle de vie au cours duquel il prend des états différents en fonction d'événements. Si un concept est lié à un objet métier, d'autres concepts peuvent être liés aux différents états de l'objet métier ainsi qu'aux événements qui sont à l'origine des changements d'état. Avec **HOPEX Information Architecture**, il est possible d'associer un cycle de vie à un concept ainsi que des concepts état et des concepts événement.

Enfin, les individus peuvent également être liés à des états d'individu et des événements d'individu qui sont des instanciations des concepts d'état et des concepts événement.

Les facilités proposées par **HOPEX Information Architecture** pour décrire l'évolution dans le temps d'un concept et des individus sont décrites ici :

Décrire les concepts d'état

Pour représenter la notion d'état d'un concept, **HOPEX Information Architecture** propose le **Concept d'état**.

> 🕮 *Un concept d'état est une situation au cours de la vie d'un concept durant laquelle il satisfait à certaines conditions, exerce une certaine activité ou attend un événement de concept. Un concept d'état représente un intervalle de temps dont les bornes sont deux événements de concept. Un état de concept est une phase par laquelle passe le concept au cours de son cycle de vie.*

Par exemple, un même Abonné peut passer d'un état de "Enfant" à l'état "Adolescent" puis à l'état "Adulte" et enfin "Senior".

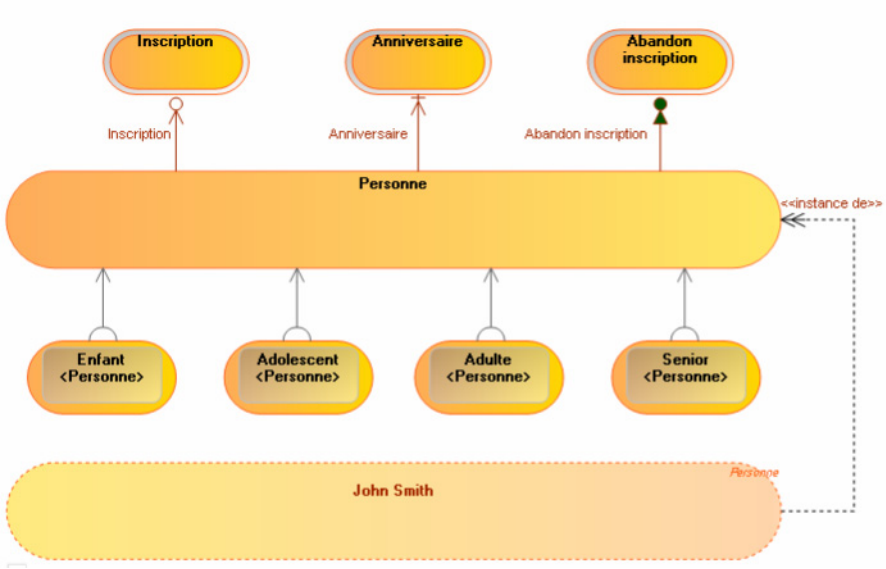
Le passage d'un concept d'état à un autre peut être conditionné par un **Concept événement**.

> 🕮 *Un concept événement représente un fait se produisant durant la vie d'un concept, par exemple un changement de saison. Un concept événement permet de marquer l'impact sur un concept d'un phénomène interne ou externe au concept. On peut distinguer les événements de*

début de concept, les événements de fin de concept et les événements intermédiaires de concept.

Par exemple, `le passage d'un état à un autre peut être lié à un événement, un "Anniversaire", par exemple.`

☛ *Pour plus de détails, voir "Décrire les concepts événement", page 56.*



Accéder à la liste des concepts d'état

Pour accéder aux concepts d'état d'un domaine de connaissance :

1. Depuis le volet de navigation **Information Architecture**, cliquez sur **Domaines de connaissance > Hiérarchie des informations métier**.
2. Dépliez le dossier **Domaines de connaissance** puis dépliez le dossier du domaine de connaissance qui vous intéresse.
3. Dépliez le dossier **Concepts d'état**.
   La liste des concepts d'état du domaine de connaissance apparaît.

Créer un concept d'état à partir d'un domaine de connaissance

Lors de la création d'un concept d'état, **HOPEX Information Architecture** construit également un **Etat du dictionnaire de** qui représente la relation entre un concept d'état et son concept.

🕮 *Un état du dictionnaire de permet de relier un concept à un état de concept et de spécifier la nature de l'état.*

Pour créer un concept d'état à partir d'un domaine de connaissance :

1. Faites un clic droit sur le domaine de connaissance qui vous intéresse et cliquez sur **Nouveau > Concept d'état**.
   L'assistant de création d'un concept d'état apparaît.

2. Renseignez le **Nom local** et cliquez sur **Suivant**.
3. Dans le champ **Individu type d'un état** précisez à quel concept est relié le concept d'état que vous êtes en train de créer.

   ☛ *Un **Etat du dictionnaire de** est automatiquement créé entre le concept et le concept d'état.*
4. Dans la section **Terme**, le tableau **Termes existants** dresse la liste des termes portant le même nom que le nouveau concept d'état.

   📖 *Un terme est un mot ou groupe de mots considéré dans sa valeur de désignation, en particulier dans un vocabulaire spécialisé.*

   ☛ *Si un terme a déjà été créé avec le même nom que le nouveau concept d'état, ce terme est automatiquement relié et il apparaît dans la section **Terme**.*
5. Dans le champ **Texte de la définition**, saisissez le texte de la définition du concept d'état et cliquez sur **OK**.
   Le nom du concept d'état apparaît dans l'arborescence sous le domaine de connaissance.

   ☛ *Vous pouvez également créer un concept d'état dans un domaine d'information métier.*

Les propriétés d'un concept d'état

Les caractéristiques d'un concept d'état

L'onglet **Caractéristiques** de la fenêtre de propriétés d'un concept d'état permet d'accéder à ses principales caractéristiques.

Avec **HOPEX Information Architecture** le concept d'état est décrit par :

- sa **Désignation** qui est représentée par un ou plusieurs termes,

  ☛ *Pour modifier le nom d'un concept dans une langue donnée, vous devez accéder aux propriétés du concept et modifier le nom du terme qui lui est associé dans la langue concernée. Pour plus de détails, voir "Concept et terme", page 3.*
- le **Texte de sa définition,**
- La section **Synonymes** permet de spécifier une liste des concepts qui lui sont synonymes,

  📖 *Un synonyme est un terme qui est interchangeable avec un autre dans le contexte d'un concept de ce terme avec le même sens, ou presque.*

  ☛ *Pour plus de détails, voir "Concept et terme", page 3.*
- La section **Réalisation** permet d'associer au concept un élément de l'architecture applicative.

  ☛ *Pour plus de détails, voir le guide **HOPEX Logical Data**.*

Les liens entre un concept d'état et les autres éléments de dictionnaire

Outre les caractéristiques de terminologie, un concept d'état est caractérisé par ses relations avec les autres éléments de dictionnaire.

- L'onglet **Sur-type** présente les concepts dont les propriétés sont héritées par le concept décrit, pour plus de détails voir "Décrire les variations d'un

concept", page 44

- L'onglet **Composant** présente :
  - la liste des composants structurels de concept détenus, pour plus de détails voir "Décrire les composants d'un concept", page 45
  - la liste des Super composants de concept, pour plus de détails voir "Décrire le super composant d'un concept", page 42.

> ☛ *Les concepts reliés à un concept d'état ne sont pas présentés dans la fenêtre de propriétés.*

Décrire les concepts événement

Un concept événement représente un fait se produisant durant la vie d'un concept, par exemple un changement de saison. Un concept événement permet de marquer l'impact sur un concept d'un phénomène interne ou externe au concept. On peut distinguer les événements de début de concept, les événements de fin de concept et les événements intermédiaires de concept.

Accéder à la liste des concepts événement

Pour accéder aux concepts événement d'un domaine de connaissance :

1. Depuis le volet de navigation **Information Architecture**, cliquez sur **Domaines de connaissance > Hiérarchie des informations métier**.
2. Dépliez le dossier **Domaines de connaissance** puis dépliez le dossier du domaine de connaissance qui vous intéresse.
3. Dépliez le dossier **Concepts événement**.
   La liste des concepts événement du domaine de connaissance apparaît.

> ☛ *En dépliant le dossier d'un concept, vous pouvez également accéder aux concepts événement qui lui sont attachés.*

Créer un concept événement à partir d'un domaine de connaissance

Pour créer un concept événement à partir d'un domaine de connaissance :

1. Faites un clic droit sur le domaine de connaissance qui vous intéresse et cliquez sur **Nouveau > Concept événement**.
   L'assistant de création d'un concept événement apparaît.
2. Renseignez le **Nom local** et cliquez sur **Suivant**.
3. Dans la section **Terme**, le tableau **Termes existants** dresse la liste des termes portant le même nom que le nouveau concept événement.

> 🕮 *Un terme est un mot ou groupe de mots considéré dans sa valeur de désignation, en particulier dans un vocabulaire spécialisé.*

> ☛ *Si un terme a déjà été créé avec le même nom que le nouveau concept événement, ce terme est automatiquement relié et il apparaît dans la section **Terme**.*

4. Dans le champ **Texte de la définition**, saisissez le texte de la définition du concept événement et cliquez sur **OK**.
   Le nom du concept événement apparaît dans l'arborescence sous le domaine de connaissance.

> ☛ *Vous pouvez également créer un concept événement dans un domaine d'information métier.*

Les propriétés d'un concept événement

L'onglet **Caractéristiques** de la fenêtre de propriétés d'un concept événement permet d'accéder à ses principales caractéristiques.

Avec **HOPEX Information Architecture** le concept événement est décrit par :

- sa **Désignation** qui est représentée par un ou plusieurs termes,

  ☛ *Pour modifier le nom d'un concept dans une langue donnée, vous devez accéder aux propriétés du concept et modifier le nom du terme qui lui est associé dans la langue concernée. Pour plus de détails, voir "Concept et terme", page 3.*
- le **Texte de sa définition,**
- La section **Synonymes** permet de spécifier une liste des concepts qui lui sont synonymes,

  Un synonyme est un terme qui est interchangeable avec un autre dans le contexte d'un concept de ce terme avec le même sens, ou presque.

  ☛ *Pour plus de détails, voir "Concept et terme", page 3.*
- La section **Réalisation** permet d'associer au concept un élément de l'architecture applicative.

  ☛ *Pour plus de détails, voir le guide **HOPEX Logical Data**.*

Relier un concept événement à son concept

La relation entre un concept et un concept événement est décrite par :

- un **Événement de début**,
- un **Événement de fin**,
- ou un **Événement intermédiaire**.

Pour relier un concept événement à son concept dans le diagramme associé à un domaine d'information métier :

1. Dans la barre d'insertion, cliquez sur le bouton **Lien**.
2. Cliquez sur le concept auquel le concept événement est attaché.

   `Par exemple, "Personne"`
3. Cliquez sur le concept événement à relier.

   `Par exemple, "Anniversaire".`

   Un assistant vous propose de choisir le type d'événement :
   - **Événement de début de concept**,
   - **Événement de fin de concept**,
   - **Événement intermédiaire de concept**.
4. Sélectionnez le type d'événement et cliquez sur **OK**.
   L'assistant de création du type d'événement de concept sélectionné s'ouvre.
5. Renseignez le **Nom local**.
6. Etant donné qu'aucun terme n'est à créer, cochez la case "Création sans terme".
7. Cliquez sur **OK**.
   Le lien entre le concept et le concept événement apparaît dans le diagramme avec une icône qui représente son type.

Le diagramme de structure de concept d'état

Dans **HOPEX Information Architecture**, un diagramme de structure de concept d'état rassemble l'ensemble des informations relatives au concept d'état décrit. Ce diagramme est initialisé à partir de des éléments du graphe de concept.

Par exemple,

Ce diagramme est composé de :

- *variation* entre les composants,

  Par exemple, les "Abonnement" peuvent être classés par "Type d'abonnement". Un "Type d'abonnement" étant caractérisé par un "Type de prêt".

  Une variation décrit comment un concept peut être varié sous une autre forme. La variante est un objet quasi-similaire à l'objet varié mais avec des propriétés ou des relations qui peuvent différer.

  ☛ *Pour plus de détails, voir "Créer une variation d'un composant structurel de concept", page 45.*

- *substitution* entre les composants,

  Une substitution détermine quel élément peut être utilisé à la place d'un autre ou est effectivement remplacé par un élément existant dans un context donné (par exemple dans le cadre d'une variation). Contrairement à la variation, une substitution n'induit pas d'héritage mais une équivalence fonctionnelle.

  ☛ *Pour plus de détails, voir "Créer une substitution d'un composant structurel de concept", page 46.*

- *Représentations de concept*,

  Une représentation de concept permet de spécifier la relation entre un concept et une représentations type.

  ☛ *Pour plus de détails, voir "Utiliser les représentations types", page 43.*

- *Composant structurel de concept* qui décrit la relation entre deux **Concepts**,

  Par exemple, un "Type d'abonnement" est caractérisé par un "Type de prêt disponible".

  Un composant structurel de concept permet de représenter une relation de dépendance entre deux concepts. Cette relation est orientée.

  ☛ *Pour plus de détails, voir "Décrire les composants d'un concept", page 45.*

- *Evénement de début*, *Evénement intermédiaire* et *Evénement de fin* qui permettent de définir les événements qui contribuent au changement d'état d'un concept,

  Par exemple, le changement de état d'un adhérent peut être provoqué par son anniversaire.

  Un concept événement représente un fait se produisant durant la vie d'un concept, par exemple un changement de saison. Un concept événement permet de marquer l'impact sur un concept d'un phénomène interne ou externe au concept. On peut distinguer les événements de début de concept, les événements de fin de concept et les événements intermédiaires de concept.

  ☛ *Pour plus de détails, voir "Décrire les concepts d'état", page 53.*

Décrire les états et les événements d'un individu

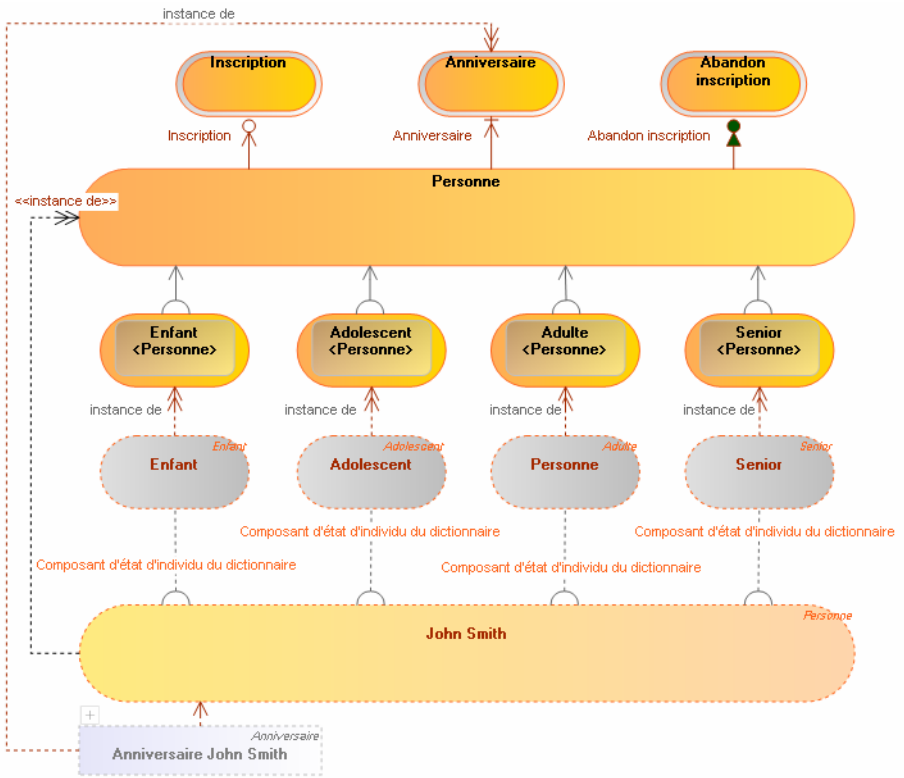
Si un concept est associé à des états, les occurrences de ce concept peuvent également être associées à des états. Ainsi **HOPEX Information Architecture** propose l'**Etat d'individu**.

> *Un état d'individu est une instance d'un état du concept auquel l'individu est relié. Il représente un état de l'individu au cours de son cycle de vie.*

Par ailleurs, le passage d'un état d'individu à un autre peut être conditionné par un **Événement d'individu**.

> *Un événement d'individu représente un fait se produisant durant la vie de l'individu. C'est une instanciation d'un concept événement du concept auquel l'individu est relié.*

Par exemple, "John Smith" est une "Personne" qui peut passer d'un état à autre le jour de son anniversaire.



La relation entre un individu et son **Etat d'individu** est décrite par un **Composant d'état d'individu**.

> *Un composant d'état d'individu de permet de relier un individu à un état d'individu.*

La relation entre un individu et son **Événement d'individu** est décrite par **Composant d'entité du dictionnaire**.

> 🕮 *Un composant d'entité permet de relier un individu à un élément de dictionnaire.*

Avec **HOPEX Information Architecture :**

- un état d'individu est une instance d'un concept d'état

  > 🕮 *Un concept d'état est une situation au cours de la vie d'un concept durant laquelle il satisfait à certaines conditions, exerce une certaine activité ou attend un événement de concept. Un concept d'état représente un intervalle de temps dont les bornes sont deux événements de concept. Un état de concept est une phase par laquelle passe le concept au cours de son cycle de vie.*

- un événement d'individu est une instance concept événement.

  > 🕮 *Un concept événement représente un fait se produisant durant la vie d'un concept, par exemple un changement de saison. Un concept événement permet de marquer l'impact sur un concept d'un phénomène interne ou externe au concept. On peut distinguer les événements de début de concept, les événements de fin de concept et les événements intermédiaires de concept.*

Accéder à la liste des états et des événements d'individu

Pour accéder aux états d'individu d'un domaine de connaissance :

1. Depuis le volet de navigation **Information Architecture**, cliquez sur **Domaines de connaissance > Hiérarchie des informations métier**
2. Dépliez le dossier **Domaines de connaissance**.
3. A partir du domaine de connaissance qui vous intéresse, dépliez le dossier **Etat d'individu**.
   La liste des états d'individu du domaine de connaissance apparaît.
4. Dépliez le dossier **Evénement d'individu**.
   La liste des événements d'individu du domaine de connaissance apparaît.

Créer un état d'individu à partir d'un domaine d'information métier

La relation entre un individu et son **Etat d'individu** est décrite par une **Composant d'état d'individu**.

> 🕮 *Un composant d'état d'individu de permet de relier un individu à un état d'individu.*

Si vous créez un état d'individu dans un diagramme, il est possible de créer automatiquement le composant d'état d'individu associé.

Pour créer un état d'individu à partir d'un graphe de concept :

1. Dans le diagramme, cliquez sur l'individu qui détient l'état d'individu.

   > ☛ *Si vous êtes en **HOPEX Windows Front-End**, passez la souris sur l'individu qui détient l'état d'individu et cliquez sur le signe* +.

2. Sélectionnez **Etat d'individu**.
3. Cliquez dans le diagramme.
   L'assistant de création d'un état d'individu apparaît.
4. Renseignez le **Nom local** et cliquez sur **Ajouter**.
   Le nouvel état d'individu apparaît dans le diagramme.

   > ☛ *Vous pouvez également créer un état d'individu à partir de son domaine de connaissance.*

Les propriétés d'un état d'individu

La fenêtre de propriétés d'un état d'individu présente dans l'onglet **Caractéristiques** les éléments suivants :

- Son **Nom Local**.
- Les classifications d'individu qui apparaissent dans la section **Classification**.

  📖 *Un composant d'état d'individu de permet de relier un individu à un état d'individu.*

  ☛ *Pour plus de détails, voir "Créer une classification d'individu", page 50.*
- L'onglet **Composant** qui présente les individus qui définissent l'individu décrit.

  ☛ *Pour plus de détails, voir "Créer un composant d'entité de dictionnaire", page 51.*

Créer un événement d'individu à partir d'un domaine d'information métier

Pour créer un événement d'individu à partir d'un domaine d'information métier :

1. Dans la barre d'insertion cliquez sur le bouton **Evénement d'individu** et cliquez dans le diagramme.
   L'assistant d'ajout d'un événement d'individu apparaît.
2. Renseignez le **Nom** et cliquez sur **Ajouter**.
   L'événement d'individu apparaît dans le diagramme.

Relier un événement d'individu à un individu

La relation entre un individu et son **Événement d'individu** est décrite par un **Composant d'entité du dictionnaire**.

📖 *Un composant d'entité permet de relier un individu à un élément de dictionnaire.*

Pour relier un concept événement à son concept dans le diagramme :

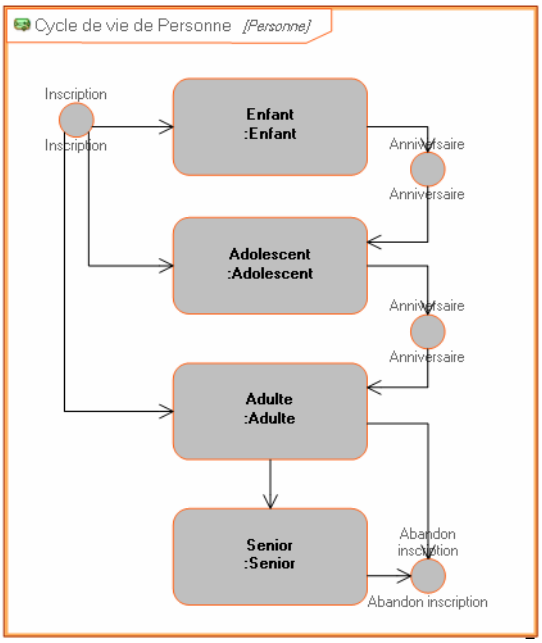
1. Dans la barre d'insertion, cliquez sur le bouton **Lien**.
2. Cliquez sur l'événement d'individu.
3. Cliquez sur l'individu.
   Le lien apparaît dans le diagramme.

Le diagramme de structure de cycle de vie de concept

Le diagramme de structure de cycle de vie de concept permet de décrire le cycle de vie d'un concept.

```
Par exemple, une "Personne", devient visible d'une
médiathèque après son "Inscription". Elle peut être inscrit
avec l'état de "Enfant", "Adolescent", "Adulte" ou
```

"Senior". Le passage d'un état à un autre peut être lié à un événement, un "Anniversaire", par exemple.



Un diagramme de structure de cycle de vie de concept est composé des éléments suivants :

- Les **Phases de cycle de vie de concept**, qui sont reliées aux concepts d'état du concept "Personne",

  Un concept d'état est une situation au cours de la vie d'un concept durant laquelle il satisfait à certaines conditions, exerce une certaine activité ou attend un événement de concept. Un concept d'état représente un intervalle de temps dont les bornes sont deux événements de concept. Un état de concept est une phase par laquelle passe le concept au cours de son cycle de vie.

  ☛ *Pour plus de détails sur les concepts d'état, voir "Décrire les concepts d'état", page 53*

- Les **Evénements de cycle de vie de concept**, qui sont reliées aux concepts événements du concept "Personne",

  Un concept événement représente un fait se produisant durant la vie d'un concept, par exemple un changement de saison. Un concept événement permet de marquer l'impact sur un concept d'un phénomène interne ou externe au concept. On peut distinguer les événements de début de concept, les événements de fin de concept et les événements intermédiaires de concept.

  ☛ *Pour plus de détails sur les concepts événements, voir "Décrire les concepts événement", page 56*

- Des **Transition de cycle de vie de concept** qui représente les enchaînements entre les états et les événements de concept.

Créer un cycle de vie de concept

Avec **HOPEX Information Architecture**, pour construire un diagramme de structure de cycle de vie de concept et décrire l'enchaînement des états qui définissent le cycle de vie d'un concept, il faut d'abord créer le **Cycle de vie du concept**.

Pour créer un cycle de vie de concept partir d'un domaine de connaissance :

1. Faites un clic droit sur le domaine de connaissance qui vous intéresse et cliquez sur **Nouveau > Élément de domaine de connaissance > Cycle de vie de concept** et cliquez sur **OK**.
   L'assistant de création d'un cycle de vie de concept apparaît.
2. Renseignez le **Nom local** et cliquez sur **Suivant**.
3. Dans le champ **Cycle de vie de**, précisez le concept sur lequel porte le cycle de vie.

   ```
   Par exemple, "Personne".
   ```

4. Dans la section **Terme**, le tableau **Termes existants** dresse la liste des termes portant le même nom que l'objet créé.

   ☛ *Si un terme a déjà été créé avec le même nom que le cycle de vie, ce terme est automatiquement relié au concept et il apparaît automatiquement dans la section **Terme**.*

5. Dans le champ **Texte de la définition**, saisissez le texte de la définition du concept d'état et cliquez sur **OK**.
   Le nom du nouveau cycle de vie de concept apparaît dans l'arborescence sous le domaine de connaissance.

Créer un diagramme de structure de cycle de vie de concept

Pour créer un diagramme de structure de cycle de vie de concept à partir d'un cycle de vie de concept :

- Faites un clic droit sur le cycle de vie de concept qui vous intéresse et cliquez sur **Nouveau > Diagramme de structure de cycle de vie de concept**.
  Le diagramme s'ouvre dans la fenêtre d'édition. Les concepts d'état associés au concept décrit sont positionnés dans le diagramme à travers des objets de type **Phases de cycle de vie de concept**.

Ajouter un événement de cycle de vie de concept

Pour ajouter un événement de cycle de vie de concept dans le diagramme de structure de cycle de vie de concept :

```
Par exemple, l'événement de cycle de vie de concept qui
correspond à l'"Inscription" de l'adhérent.
```

1. Dans la barre d'objets du diagramme, cliquez sur le bouton **Evénement de cycle de vie de concept**.
2. Cliquez dans le cadre du cycle de vie de concept.
   Une fenêtre de création d'un événement de cycle de vie de concept s'ouvre.

3. Dans le champ **Type composé**, précisez le nom du concept événement sur lequel porte l'objet créé.

   Par exemple, "Inscription".

   ☛ *Si une fenêtre de choix s'ouvre, sélectionnez l'objet qui vous intéresse.*
4. Renseignez le **Nom local**.
5. Etant donné qu'aucun terme n'est à créer, cochez la case "Création sans terme".
6. Cliquez sur **OK**.
   L'événement de cycle de vie de concept apparaît dans le diagramme.

Créer une Transition de cycle de vie de concept

Pour représenter l'enchaînement d'une phase vers un événement de cycle de vie de concept, vous devez créer une transition de cycle de vie de concept.

Pour créer une transition de cycle de vie de concept.

1. Dans la barre d'insertion, cliquez sur le bouton **Transition de cycle de vie de concept**.
2. Cliquez sur la phase (ou l'événement) de cycle de vie de concept déclenchante et, en maintenant le bouton gauche de la souris enfoncé, déplacez le curseur sur la phase (ou l'événement) déclenché.
3. Relâchez le bouton de la souris.
   Le lien apparaît dans le diagramme.

Utiliser les périodes

Une **Période** permet d'apporter des précisons temporelles sur un **événement d'individu**.

📖 *Un événement d'individu représente un fait se produisant durant la vie de l'individu. C'est une instanciation d'un concept événement du concept auquel l'individu est relié.*

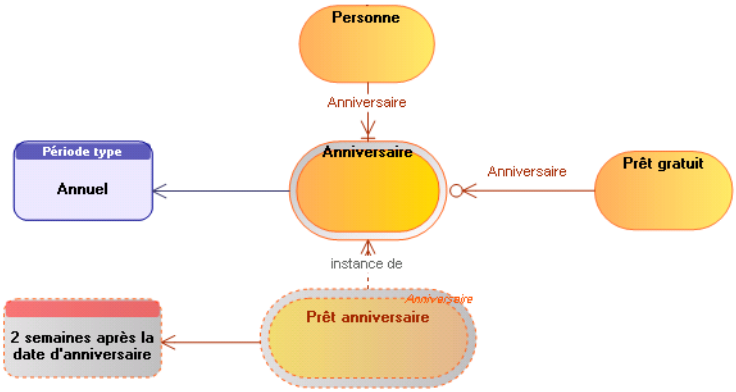
Par exemple, un prêt gratuit peut être proposé aux abonnés à chaque anniversaire. Ce prêt est valide pendant une durée de deux semaines après la date d'anniversaire.

Une **Période type** permet de préciser un **concept événement**.

📖 *Un concept événement représente un fait se produisant durant la vie d'un concept, par exemple un changement de saison. Un concept événement permet de marquer l'impact sur un concept d'un phénomène interne ou externe au concept. On peut distinguer les événements de*

début de concept, les événements de fin de concept et les événements intermédiaires de concept.

Par exemple, un prêt gratuit d'anniversaire est proposé tous les ans.



La relation entre une **période type** et un **événement d'individu** est décrite par une **Périodisation de Concept Événement**.

La relation entre une **période** et un **concept événement** est décrite par une **Périodisation d'événement**.

DÉCRIRE UN CONCEPT TYPE

Un concept type permet de classifier les concepts. Les relations entre les concepts type sont représentées par des composants de concept type.

Les facilités proposées par **HOPEX Information Architecture** pour utiliser les concepts type sont décrites ici :

Accéder à la liste des concepts type

Pour accéder à l'ensemble des concepts type de votre référentiel avec **HOPEX Web Front-End** :

- Dans le volet **Information métier**, cliquez sur **Concepts type > Tous les concepts type**.
  La liste des concepts type s'affiche.

  *Pour accéder à l'ensemble des concepts type que vous avez créés avec **HOPEX Web Front-End** : dans le volet **Information métier**, cliquez sur **Concepts type > Mes concepts type**.*

Pour accéder à l'ensemble des concepts type d'un domaine de connaissance avec **HOPEX Windows Front-End** :

1. Dans le bureau **Information Architecture**, cliquez sur l'onglet **Bibliothèque/IA** puis sur le volet de navigation **Information Architecture**.
2. Dépliez le dossier **Domaines de connaissance**.
3. Dépliez le dossier du domaine de connaissance qui vous intéresse.
4. Dépliez le dossier **Concept type**.
   La liste des concepts type du domaine de connaissance apparaît.

Pour accéder aux concepts type que vous avez créés avec **HOPEX Windows Front-End** :

1. Dans le bureau **Information Architecture**, cliquez sur l'onglet **Accueil** puis sur **Mon bureau** > **Mes Responsabilités**.
2. Dépliez le dossier **Mes types de concept**.
   La liste de vos concepts type apparaît.

Créer un nouveau concept type

Pour créer un concept type à partir d'un domaine de connaissance :

1. Faites un clic droit sur le domaine de connaissance qui vous intéresse et cliquez sur **Nouveau > Élément de domaine de connaissance > Concept type**.
   L'assistant de création d'un concept type apparaît.
2. Renseignez le **Nom local** et cliquez sur **Suivant**.
3. Dans la section **Terme**, le tableau **Termes existants** dresse la liste des termes portant le même nom que le nouveau concept type.

   📖 *Un terme est un mot ou groupe de mots considéré dans sa valeur de désignation, en particulier dans un vocabulaire spécialisé.*

   ☛ *Si un terme a déjà été créé avec le même nom que le nouveau concept type, ce terme est automatiquement relié et il apparaît dans la section **Terme**.*
4. Dans le champ **Texte de la définition**, saisissez le texte de la définition du concept type et cliquez sur **OK**.
   Le nom du nouveau concept type apparaît dans l'arborescence sous le domaine de connaissance.

   ☛ *Un nouveau terme portant le même nom que le concept type est également créé.*

Les propriétés d'un concept type

Les caractéristiques d'un concept type

L'onglet **Caractéristiques** de la fenêtre de propriétés d'un concept type permet d'accéder à ses principales caractéristiques.

Avec **HOPEX Information Architecture** le concept type est décrit par :

- sa **Désignation** qui est représentée par un ou plusieurs termes,

  ☛ *Pour modifier le nom d'un concept dans une langue donnée, vous devez accéder aux propriétés du concept et modifier le nom du terme qui lui est associé dans la langue concernée. Pour plus de détails, voir "Concept et terme", page 3.*
- le **Texte de sa définition,**
- La section **Synonymes** permet de spécifier une liste des concepts qui lui sont synonymes,

  📖 *Un synonyme est un terme qui est interchangeable avec un autre dans le contexte d'un concept de ce terme avec le même sens, ou presque.*

  ☛ *Pour plus de détails, voir "Concept et terme", page 3.*
- La section **Réalisation** permet d'associer au concept un élément de l'architecture applicative.

  ☛ *Pour plus de détails, voir le guide **HOPEX Logical Data**.*

Les liens entre un concept type et les autres éléments de dictionnaire

Outre les caractéristiques de terminologie, un concept est caractérisé par ses relations avec les autres éléments de dictionnaire.

- L'onglet **Composant** présente la liste des composants de concept type détenus, pour plus de détails voir "Décrire les composants d'un concept", page 45.
- L'onglet **Sur-types** présente les concepts type dont les propriétés sont héritées par le concept type décrit, pour plus de détails voir "Décrire les variations d'un concept type", page 69.

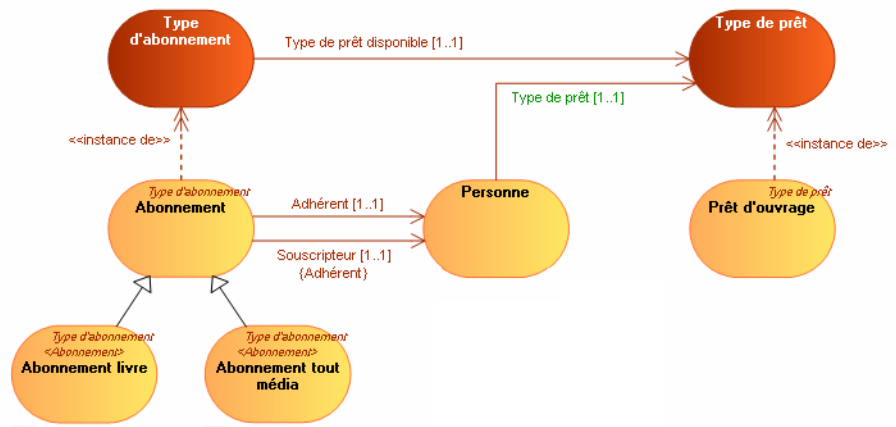
Décrire les composants d'un concept type

Avec **HOPEX Information Architecture**, il est possible de relier un concept type à un autre concept type pour le caractériser.

```
Par exemple, un "Type d'abonnement" est caractérisé par un
"Type de prêt".
```

Cette relation est décrite par un **Composant de concept type** qui peut, éventuellement, être associé à un terme.

Un composant de concept type permet de spécifier la relation entre deux concepts type.



Accéder aux composants de concept type

Pour accéder aux composants de concept type d'un concept type :

1. Ouvrez la fenêtre de propriétés du concept type.

2. Sélectionnez l'onglet **Composants**.
   La liste des composants de concept type associés au concept type apparaît.

   ☛ *Vous pouvez également consulter la liste des composants structurels d'un concept type à partir de son diagramme de structure de cycle de vie de concept. Pour plus de détails, voir "Le diagramme de structure de concept type", page 70.*

Créer un composant de concept type à partir d'un domaine d'information métier

Pour créer un composant de concept type entre deux concepts type dans un diagramme de domaine d'information métier :

1. Dans la barre d'insertion, cliquez sur le bouton **Lien**.
2. Cliquez sur le concept type qui détient le lien.

   `Par exemple, "Type d'abonnement"`
3. Cliquez sur le concept type cible.

   `Par exemple, "Type de prêt".`

   L'assistant de création d'un composant de concept type apparaît.
4. Renseignez le **Nom local**.
5. Etant donné qu'aucun terme n'est à créer, cochez la case "Création sans terme".
6. Cliquez sur **OK**.
   Le composant concept type apparaît dans le diagramme.

Vous pouvez également créer un composant de concept type dans un diagramme de structure de concept type. Dans ce cas là, vous devez préciser le concept type cible dans l'assistant de création d'un composant de concept type.

☛ *Pour plus de détails, voir "Le diagramme de structure de concept type", page 70.*

Décrire les variations d'un concept type

Certains concepts type sont des déclinaisons d'autres concepts type : ils sont caractérisés par les mêmes composants de concept type.

Avec **HOPEX Information Architecture**, cette relation est décrite par une **Variation**.

📖 *Une variation décrit comment un concept peut être varié sous une autre forme. La variante est un objet quasi-similaire à l'objet varié mais avec des propriétés ou des relations qui peuvent différer.*

☛ *Pour plus de détails sur les variations et les substitutions, voir le guide **HOPEX Common Features**, chapitre "Manipuler les objets du référentiel", "Les variations d'objets".*

Accéder aux variations d'un concept type

Pour accéder aux variations d'un concept type :

1. Ouvrez la fenêtre de propriétés du concept type.

2. Sélectionnez l'onglet **Sur-type.**
   La liste des variations associées au concept type apparaît.

Créer une variation d'un concept type à partir d'un domaine d'information métier

Pour spécifier qu'un concept type hérite des caractéristiques définies pour un autre concept type à partir d'un diagramme de domaine d'information métier :

1. Dans la barre d'insertion, cliquez sur le bouton **Variation**.
2. Cliquez sur le concept type à varier, et faites glisser la souris jusqu'au nouveau concept type, avant de relâcher votre pression.
3. Renseignez le **Nom** et cliquez sur **Ajouter**.
   Un lien fléché du concept type à varier vers le concept type racine apparaît.

   ☛ *La variation est représentée par un lien mais il s'agit d'un objet **HOPEX**.*

Le diagramme de structure de concept type

Avec **HOPEX Information Architecture**, un diagramme de structure de concept type décrit la structure interne de l'instance de concept type par relations définies vers les autres concepts type qui le caractérisent.

Ce diagramme est composé des *composants de concept type* qui permettent de caractériser le concept type en le reliant à d'autres concepts type.

```
Par exemple, un "Type d'abonnement" est caractérisé par un
"Type de prêt".
```

📖 *Un composant de concept type permet de spécifier la relation entre deux concepts type.*

☛ *Pour plus de détails, voir "Décrire les composants d'un concept type", page 68.*

DÉFINIR DES VUES DE CONCEPT

> 🕮 *Une vue de concept permet de représenter le périmètre sémantique couvert par un objet métier. Une vue de concept est construite à partir d'une sélection de plusieurs concepts reliés dans le contexte spécifique de la vue.*

HOPEX Information Architecture fournit un éditeur qui permet de créer et de visualiser des vues métier et leurs composants.

> ☛ *Selon le même principe, la vue de données peut être utilisée pour naviguer à partir des Classes ou Entités. Pour plus de détails, voir "Les vues de données logiques", page 11.*

Créer une vue de concept

Pour créer une vue de concept avec **HOPEX Web Front-End** :

1. Cliquez sur le menu de navigation puis sur **Information métier**.
2. Dans le volet de navigation, cliquez sur **Vues de concept**.
3. Affichez toutes les vues de concept.

   > ☛ *Pour créer une vue de concept avec **HOPEX Windows Front-End**, sélectionnez l'onglet **Bibliothèque/IA > Eléments de dictionnaire > Vue de concept.***

4. Cliquez sur **Nouveau**.
   L'assistant de création d'une vue de concept apparaît.
5. Saisissez le **Nom local**.
6. Dans la section **Terme**, le tableau **Termes existants** dresse la liste des termes portant le même nom que la vue.

   > 🕮 *Un terme est un mot ou groupe de mots considéré dans sa valeur de désignation, en particulier dans un vocabulaire spécialisé.*

7. Dans le champ **Texte de la définition**, saisissez le texte de la définition de la vue et cliquez sur **Suivant**.
8. Pour spécifier le concept source de la vue de concept, cliquez sur **Nouveau**.
9. Dans la fenêtre qui apparaît indiquez :
   - la **MetaClasse** sur laquelle porte la vue (concept, concept d'état ou concept événement)
   - Le concept de référence de la vue de données
10. Cliquez sur **Ajouter**.
11. Cliquez sur **OK** pour fermer l'assistant de création d'une vue de concept.
    La nouvelle vue de concept apparaît dans la liste.

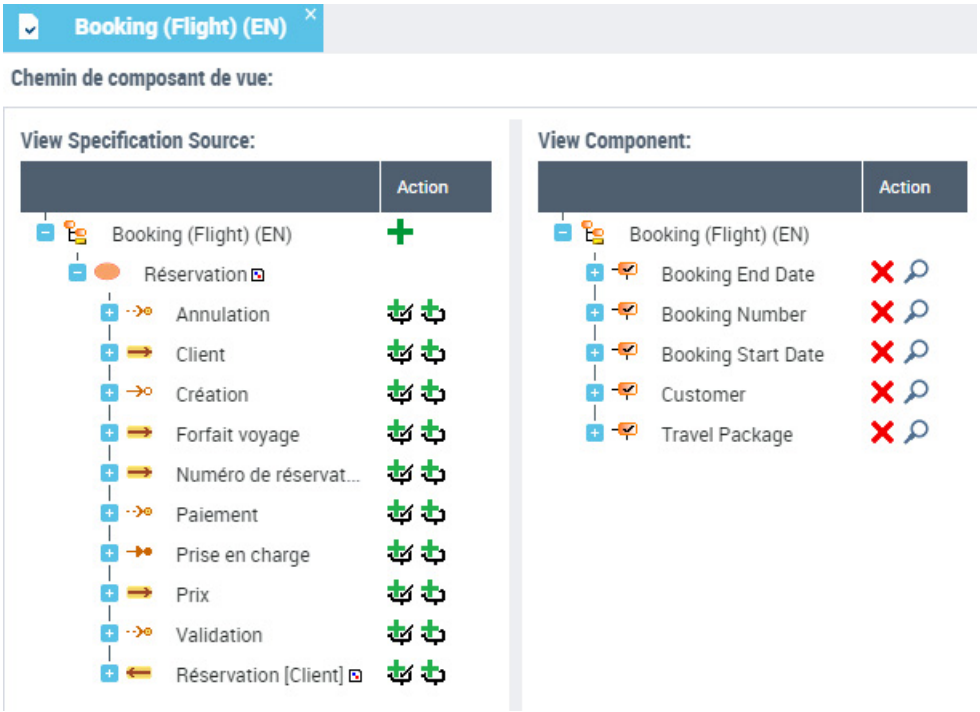
    > ☛ *Le nouvelle vue de concept est également accessible à partir de du menu **Ma vue de concept**.*

Définir le contenu de la vue de concept

Affichage des objets de la vue

L'éditeur de vue se décompose en plusieurs parties :

- la partie gauche présente l'ensemble des composants du concept source de la vue, tels que définis dans le domaine de connaissance
- la partie droite présente les composants du concept qui sont conservés dans la vue de concept créée
- les boutons de la colonne **Action** permettent d'ajouter les composants à la vue de concept.



Ajouter un objet source à la vue de concept

Pour ajouter un objet source à une vue de concept :

1. Ouvrez la vue de concept.
2. Du côté des objets sources, sous la colonne **Action**, cliquez sur le bouton  .
3. Dans la fenêtre qui apparaît, indiquez :
   - la **MetaClasse** sur laquelle porte la vue (concept, concept d'état ou concept événement)
   - Le concept de référence de la vue de données
4. Cliquez sur **Ajouter**.

Une fois le concept source défini, vous pouvez sélectionner les composants de ce concept - ou le concept lui-même - à ajouter à la vue de concept.

Ajouter un composant à la vue de concept

A partir des objets sources de la vue, vous pouvez définir des composants embarqués et des composants référencés.

Un composant embarqué permet de prendre dans la vue toutes les informations qui composent l'objet. Un composant référence ne fait que référencer l'objet dans la vue.

Pour ajouter un composant embarqué à la vue de concept :

1. Dans l'arbre de la partie gauche, sélectionnez le composant que vous souhaitez ajouter à la vue.
2. Cliquez sur le bouton **Ajouter un composant d'inclusion de vue.**
   Le composant ajouté apparaît dans l'arbre de droite.

   ☛ *De la même manière, vous pouvez **Ajouter un composant référencé**.*

Les objets embarqués dans la vue sont précédés d'une coche, contrairement aux objets référencés.

Les vues sont ensuite accessibles dans un rapport. Pour plus de détails voir "Report DataSets", page 75.

Le rapport de vue

Le rapport de vue offre un compte-rendu sur une vue de concept et ses composants.

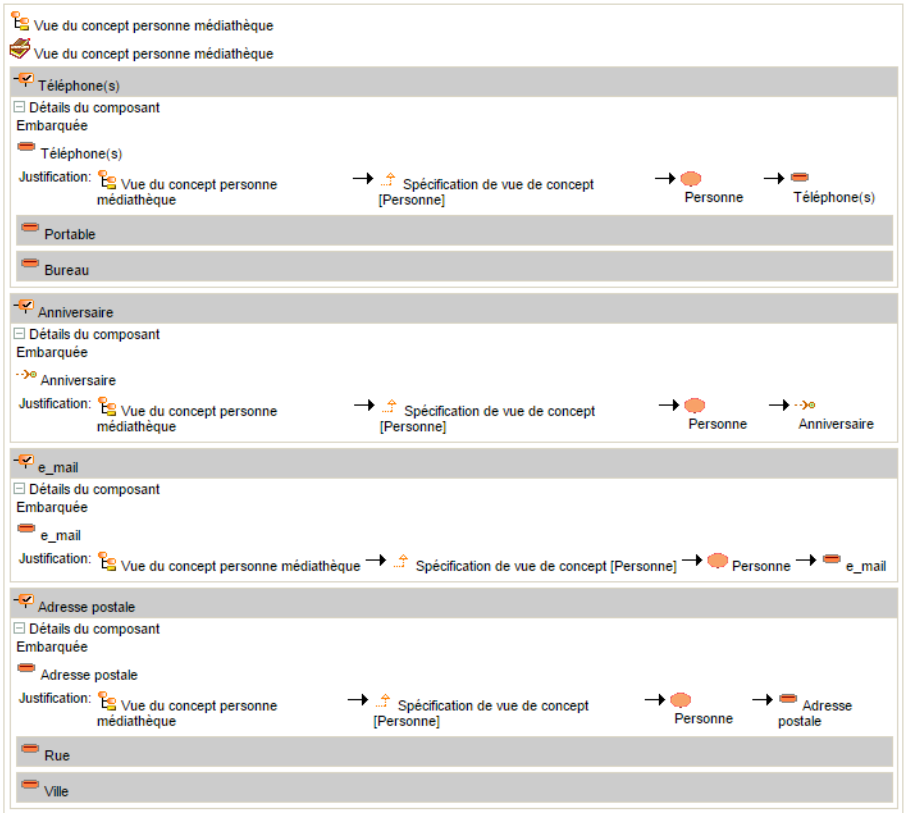
Paramètres du rapport

Il s'agit ici de définir les données en entrée du rapport.

Paramètres	Type du paramètre	Contraintes
Vue	Vue	Obligatoire.
Sous-vue	oui ou non	Oui par défaut
Justification	oui ou non	Oui par défaut
Niveau de profondeur	Entier	

Exemple de rapport

L'exemple ci-dessous permet de visualiser les éléments de la vue basée sur le concept "Personne".



RAPPORTS SUR LES DONNÉES MÉTIER

HOPEX Information Architecture offre différents types de rapport qui visent à analyser les données métier définies dans le référentiel.

☛ *Pour plus de détails sur le fonctionnement des rapports, voir le chapitre "Générer des rapports" dans le guide **HOPEX Common Features**.*

☛ *Les rapports sur les diagrammes disponibles en standard avec **HOPEX** sont aussi accessibles avec **HOPEX Information Architecture**.*

Accéder aux rapports

Pour accéder aux rapports d' **HOPEX Information Architecture** :

1. Cliquez sur le menu de navigation puis sur **Rapports**.

Rapport de vue

Voir "Le rapport de vue", page 77.

Rapport de glossaire

HOPEX Information Architecture propose un rapport de glossaire prêt à l'emploi pour construire automatiquement le glossaire métier de termes provenant d'un ensemble de domaines de connaissance. Pour chaque terme, le glossaire affiche une liste des définitions associées avec leur texte, synonyme et liste de composants.

L'utilisateur peut indiquer s'il souhaite afficher la traduction.

Paramètres du rapport

Il s'agit ici de définir les données en entrée du rapport.

Paramètres	Type du paramètre	Contraintes
Liste des biblio-thèques	Bibliothèque	Critère de sélection des termes affi-chés. Non obligatoire.
Liste des domaines de connaissance	domaine de connaissance	Critère de sélection des termes affi-chés. Un domaine obligatoire si pas de Bibliothèque.
Glossaire des traductions	oui ou non	
Option exemple		

Exemple de rapport

L'exemple ci-dessous permet de visualiser les termes et leurs liens avec les contextes.

2 Médiathèque	
Abandon inscription	Evénement 1.
Adolescent	Etat 1. (Exemples) Adolescent (Etat de) Personne (Through Component) Personne
Adresse postale	Représentation Type 1. (Composant 1) Rue (Type) Rue (Présence) Toujours (Cardinalité) 1 (Composant 2) Ville (Type) Ville (Présence) Toujours (Cardinalité) 1
Adulte	Etat 1. (Exemples) Personne (Etat de) Personne (Through Component) Personne
Anniversaire	Evénement 1. (Exemples) Anniversaire John Smith, Prêt anniversaire (Composant 1) Périodisation d'évènement type (Type) Annuel (Présence) Toujours (Cardinalité) 1

Report DataSets

Un report DataSet est un tableau de données créé à partir d'objets du référentiel, sur lequel peuvent être générés des rapports instantanés.

HOPEX Information Architecture fournit les report DataSets suivants :

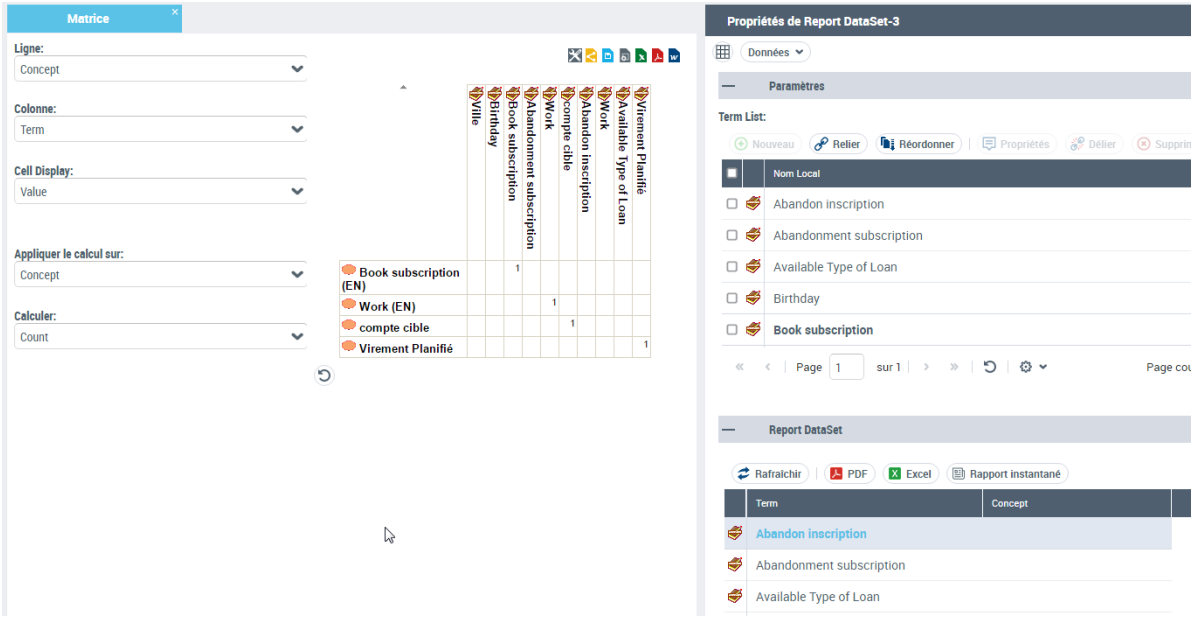
- "Définition des termes", page 76 ;
- "Matrice domaine de connaissance x Concept", page 76.

Définition des termes

Ce rapport permet d'identifier les liens entre les termes et les concepts.

Paramètre	Type du paramètre	Contraintes
Objets racine	Terme	Listes d'objets.

Il permet par exemple de créer une matrice **Terme / Concept** qui présente la liste des concepts qui utilisent un terme.

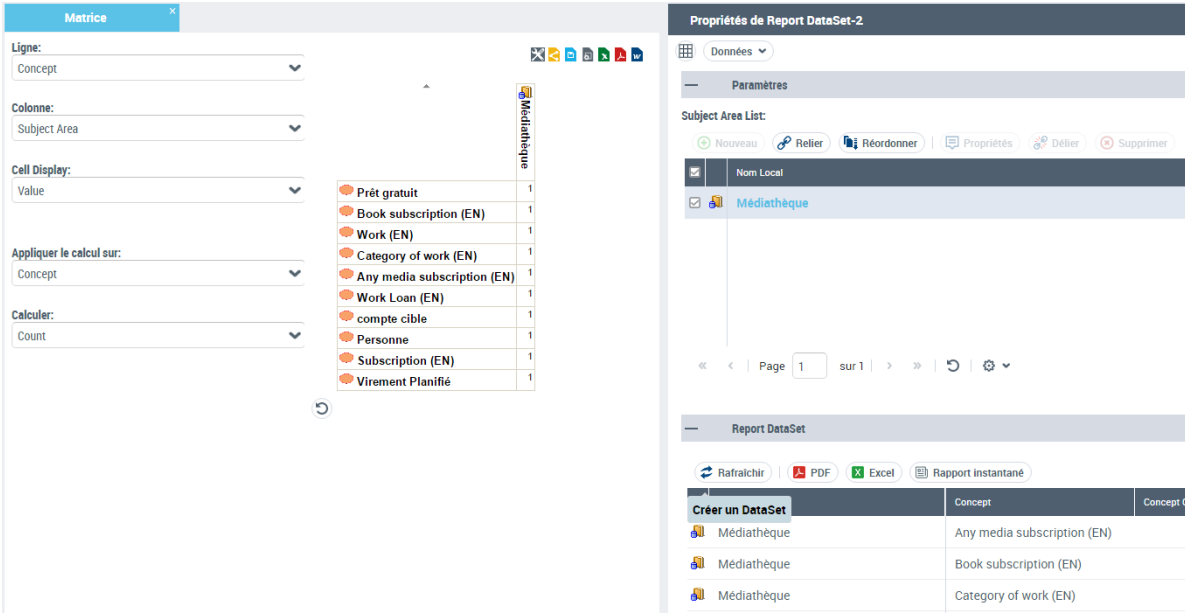


Matrice domaine de connaissance x Concept

Un concept peut être référencé par un ou plusieurs domaines de d'informations métier.

Paramètre	Type du paramètre	Contraintes
Objet racine	Domaine de connais-sance	Listes d'objets.

Cette structure de données permet par exemple de créer une **Matrice Domaine capacité x Concept**. qui permet de dresser la liste des concepts référencés dans un domaine de connaissances.



HOPEX Logical Data

Guide d’utilisation

HOPEX V2R1

HOPEX INFORMATION ARCHITECTURE - COUCHE LOGIQUE

HOPEX Information Architecture permet aux organisateurs et aux architectes d'entreprise de décrire le fonctionnement de l'entreprise par la modélisation des données utilisées lors de la mise en œuvre des processus et des applications. A cette fin, **HOPEX Information Architecture** met à leur disposition plusieurs notations.

A partir des modèles de données logiques, vous pouvez construire les modèles physiques correspondants, autrement dit de créer les tables d'une base de données, avec ses colonnes, index et clés ainsi que les dessins du diagramme relationnel.

Options de modélisation des données

Les formalismes

Vous pouvez modéliser les données logiques à partir de deux formalismes :

- le paquetage de données, pour construire les diagrammes de classes (notation UML)
- le modèle de données, pour les diagrammes de données (notations standard, IDEF1X, I.E, Merise)

Pour afficher un des formalismes :

1. Dans le bureau, cliquez sur le menu **Menu principal** > **Paramètres** > **Options.**
2. Dans l'arbre de navigation, dépliez le dossier **Modélisation des données**.
3. Cliquez sur **Formalisme de données**.
4. Dans la partie droite de la fenêtre cochez le(s) formalisme(s) que vous voulez afficher.
5. Cliquez sur **OK**.
   Les dossiers correspondant aux paquetages et aux modèles de données apparaissent dans le volet de navigation **Données logiques**.

Les notations

Vous disposez d'une notation standard de modèle de données, cochée par défaut.Pour afficher une autre notation (DEF1X, I.E ou Merise) :

1. Dans le bureau, cliquez sur le menu **Menu principal** > **Paramètres** > **Options.**
2. Dans l'arbre de navigation, dépliez le dossier **Modélisation des données**.
3. Cliquez sur **Notation des données**.
4. Dans la partie droite de la fenêtre cochez les notations que vous voulez utiliser.
5. Cliquez sur **OK**.

Accès au référentiel

Pour utiliser les fonctionnalités de **HOPEX Information Architecture**, vous devez avoir un accès au référentiel en mode "Avancé" :

1. Cliquez sur **Menu principal** > **Paramètres** > **Options**.
2. Dans la partie gauche de la fenêtre, cliquez sur le dossier **Référentiel**.
3. Dans la partie droite, vérifiez que l'accès au référentiel est en mode "Avancé".

A PROPOS DE CE GUIDE

Structure du guide

Ce guide traite les points suivants :

Ressources complémentaires

Ce guide est complété par :

- le guide **HOPEX Common Features**, qui décrit l'interface Web et les outils spécifiques aux solutions HOPEX.

  ☛ *Il peut être utile de consulter ce guide pour une présentation générale de l'interface.*

- le guide d'administration **HOPEX Power Supervisor.**

Conventions utilisées dans le guide

Styles et mises en forme

☛ *Remarque sur les points qui précèdent.*

📖 *Définition des termes employés.*

☺ *Astuce qui peut faciliter la vie de l'utilisateur.*

Compatibilité avec les versions précédentes.

Ce qu'il faut éviter de faire.

Remarque très importante à prendre en compte pour ne pas commettre d'erreurs durant une manipulation.

Les commandes sont présentées ainsi : **Fichier > Ouvrir**.

Les noms de produits et de modules techniques sont présentés ainsi : **HOPEX**.

ARCHITECTURE DES DONNÉES LOGIQUES

HOPEX Information Architecture offre un ensemble d'outils nécessaires à la modélisation des données logiques dans le formalisme classes/associations.

Vous pouvez décrire la façon dont les données d'une organisation sont utilisées par les processus et les applications qu'elle utilise. A travers les concepts de Domaine de données et de Vue de données, vous pouvez détailler une structure de données logiques dans un contexte d'utilisation particulier.

Il est également possible d'associer les entités et les classes aux concepts créés lors de phase d'analyse sémantique.

Les points abordés dans ce chapitre sont :

ACCÉDER AUX DONNÉES LOGIQUES

Dans **HOPEX Information Architecture**, l'accès aux données logiques du référentiel est réservé à l'**Architecte de données**.

Afficher le volet de navigation des données logiques

Pour accéder aux données logiques du référentiel :

1. Dans le bureau d'**HOPEX Information Architecture**, cliquez sur le menu de navigation puis sur **Données logiques**.
   Le volet de navigation affiche les types d'objets correspondants :
   - Paquetages de données
   - Modèles de données (si l'option correspondante est cochée)
   - Domaines de données
   - Vues de données

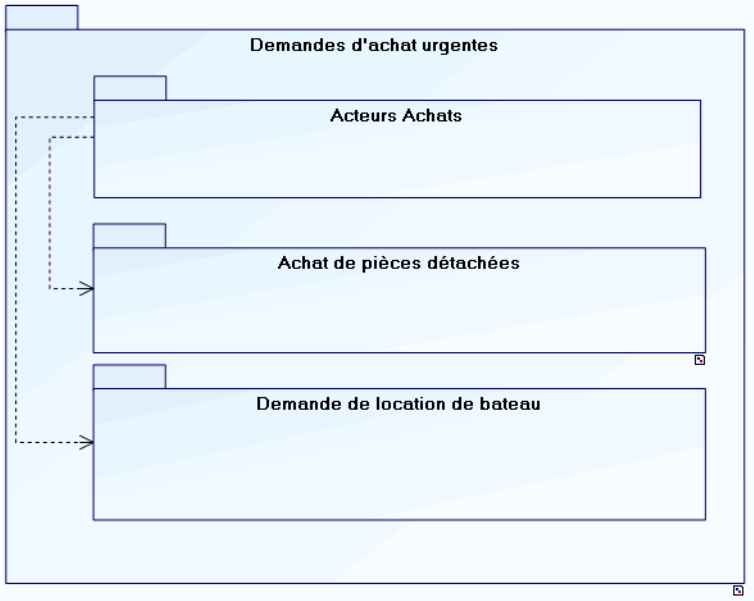
Structure des données logiques - Éléments de base

Paquetage (UML)

Un paquetage permet de représenter la structure statique d'un système, en particulier les types d'objets manipulés dans le système, leur structure interne et les relations qui existent entre eux.

Le paquetage est un élément détenteur, il fournit un espace de nommage pour les éléments qu'il regroupe.

Le paquetage vous permet de classer les éléments référencés dans un projet. Vous pouvez créer des sous-paquetages dans un paquetage afin de classer plus finement les objets, par exemple les acteurs d'un projet.




```
Les demandes d'achat urgentes sont prévues pour traiter les
achats de pièces détachées et les demandes de location de
bateau. Dans ces deux cas, les utilisateurs sont des acteurs
du domaine des achats.
```

☛ *Pour plus de détails sur l'utilisation des paquetages, voir le guide* ***HOPEX UML****.*

La représentation graphique des éléments d'un paquetage de fait dans un **diagramme de classes**.

Voir :

Modèle de données

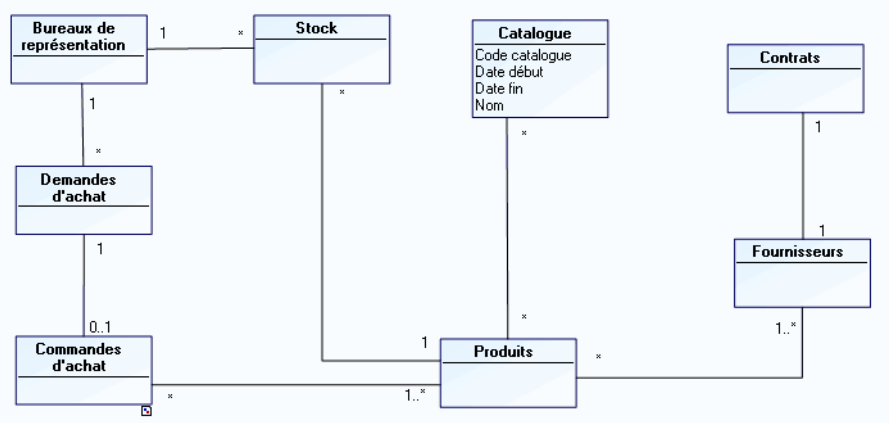
Tout comme le paquetage, le modèle de données permet de représenter la structure statique d'un système, en particulier les types d'objets manipulés dans le système, leur structure interne et les relations qui existent entre eux.

La représentation graphique des éléments d'un modèle de données se fait dans un **diagramme de données**.

Pour plus de détails sur la création et la mise à jour d'un modèle de données voir "Le modèle de données", page 23.

Exemple

Le modèle de données du projet "Automatisation des demandes d'achat" est présenté ci-dessous.



L'application gère les demandes d'achat, les commandes et les stocks de produits dans chacun des bureaux de représentation.

Un catalogue centralisé des produits et des fournisseurs est mis en place.

Les contrats avec les fournisseurs référencés sont également accessibles depuis l'application.

☛ *Pour plus de détails sur la création et la mise à jour d'un modèle de données, voir "Modéliser les données".*

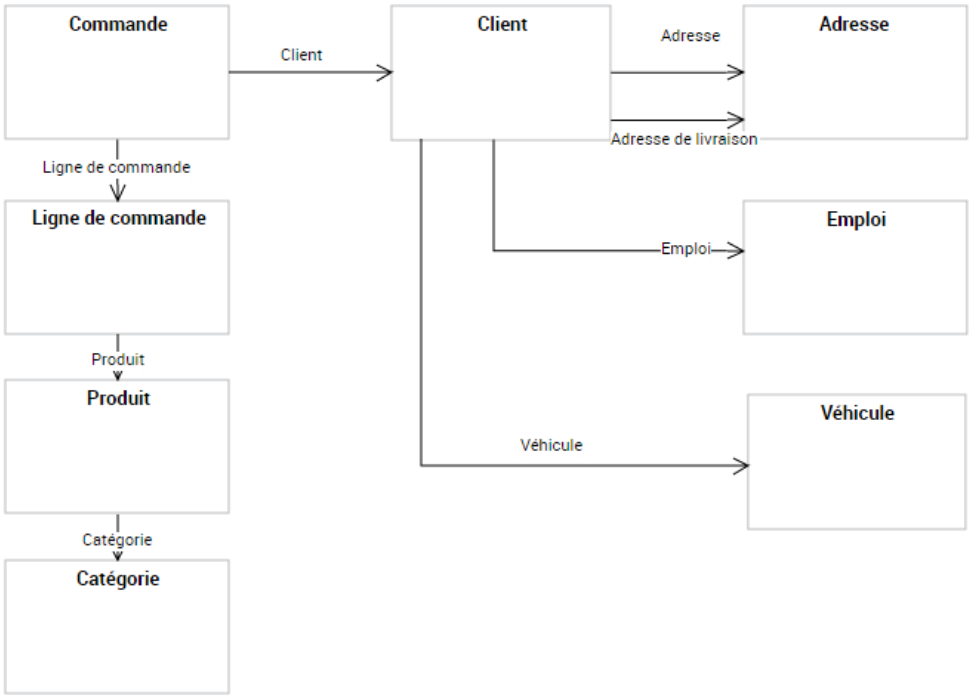
Domaine de données

Un domaine de données représente une structure de données restreinte, dédiée à la description d'un stockage de données logicielles. Il est constitué de classes et/ou de vue de classes et peut être décrit par un diagramme de domaine de données.

Pour plus de détails, voir "Les domaines de données logiques", page 8.

Exemple

Le diagramme de domaine de données suivant représente une structure de données relative aux Commandes ; il décrit sous forme de tout/partie les classes et leurs relations.



Pour répondre à des cas d'utilisation précis, vous pouvez créer des Vues de données dans lesquelles vous pouvez visualiser et modifier le périmètre couvert par des classes.

Vue de données logiques

Une vue de données permet de représenter le périmètre couvert par un élément de modèle de données ou de domaine de données. Une vue de données est construite à partir d'une sélection de plusieurs classes reliées dans le contexte spécifique de la vue. Voir "Les vues de données logiques", page 11.

LES DOMAINES DE DONNÉES LOGIQUES

Un domaine de données logiques permet de définir une structure de données logiques constituée de classes et de vues de classes.

Un domaine de données logiques est détenu par un paquetage, il peut référencer des objets détenus dans d'autres paquetages.

Il est possible de définir le mode d'accès (création, suppression, etc.) aux objets référencés par un domaine de données en les intégrant comme composants du domaine de données.

Une structure physique correspondante peut être définie via un domaine de données physiques. Celui-ci est constitué de tables et de vues de tables.

Créer un domaine de données logiques

Pour créer un domaine de données logiques :

1. Cliquez sur le menu de navigation puis sur **Données logiques**.
2. Dans le volet de navigation, cliquez sur **Tous les domaines de données logiques**.
   La liste des domaines de données logiques apparaît.
3. Cliquez sur le bouton **Nouveau**.
   Une fenêtre de création apparaît.
4. Saisissez le nom du domaine de données.
5. Indiquez éventuellement le paquetage détenteur.
6. Cliquez sur **OK**.
   Le domaine de données apparaît dans la liste.

La carte de données logiques

Une carte de données logiques est un outil d'urbanisation des informations logiques. Elle permet de représenter un ensemble de domaines de données logiques dans un contexte particulier.

Pour créer une carte de données logiques :

1. Cliquez sur le menu de navigation puis sur **Données logiques**.
2. Dans le volet de navigation, cliquez sur **Carte de données logiques**.
3. Affichez toutes les cartes de données logiques.
4. Cliquez sur **Nouveau**.
   La carte créée apparaît.

Pour créer le diagramme de la carte de données logiques :

1. Faites un clic droit sur la carte et sélectionnez **Nouveau** > **Diagramme de carte de données logiques**.
   Le diagramme apparaît dans la zone d'édition.

Les composants d'une carte de données logiques

Dans une carte de données logiques vous pouvez ajouter des composants internes et externes.

Les composants internes sont les domaines de données qui font partie du périmètre de la carte (qu'ils appartiennent ou non au paquetage détenteur).

Les composants externes sont ceux qui sont utilisés dans la carte mais qui ne font pas partie du périmètre étudié.

Le diagramme de domaine de données logiques

Un domaine de données logiques peut être décrit par un diagramme.

Un diagramme d'entités de domaine de données logiques est un diagramme de structure qui définit des classes et leurs relations suivant le formalisme Tout/Partie, en rapport avec le sujet du domaine de données décrit.

Vous pouvez relier plusieurs diagrammes de domaine de données à un domaine de données logiques, suivant ce que vous voulez décrire.

Créer le diagramme d'un domaine de données logiques

Pour créer un diagramme de domaine de données à partir du domaine de données logiques :

- Faites un clic droit sur le domaine de données logiques et sélectionnez **Nouveau** > **Diagramme d'entités du domaine de données logiques**.

Ajouter un objet au diagramme

Dans le diagramme du domaine de données, vous pouvez ajouter un nouvel objet ou relier un objet existant.

Ajouter une classe

Pour ajouter une nouvelle classe dans le diagramme :

1. Dans la barre d'insertion du diagramme, cliquez sur le bouton **Classe** puis cliquez dans le diagramme.
   La fenêtre **Ajout d'une classe** apparaît.
2. Saisissez le nom de la classe et cliquez sur **Ajouter**.

Ajouter une vue de données

Pour ajouter une nouvelle vue de données dans le diagramme :

1. Dans la barre d'insertion du diagramme, cliquez sur le bouton V**ue de données** puis cliquez dans le diagramme.
   La fenêtre **Ajout d'une vue de données** apparaît.
2. Saisissez le nom de la vue de données et cliquez sur **ajouter**.

3. L'éditeur de vue apparaît. Il vous permet de définir les composants de la vue. Voir "Créer une vue de données logiques", page 11.

Ajouter un composant au domaine de données

Il est possible d'attacher des objets au domaine de données à travers des composants. Un composant référence un objet (classe ou vue de classe) et définit le type d'accès à l'objet en question (lecture seule, modification, suppression, etc.).

Le domaine de données est rattaché à un paquetage ; les objets créés directement par le biais de composants sont reliés automatiquement au paquetage du domaine de données.

Vous pouvez créer un composant à partir d'un objet qui figure dans le diagramme du domaine de donné ou à partir des propriétés du domaine de données.

Pour créer un composant à partir d'un objet du diagramme de domaine de données :

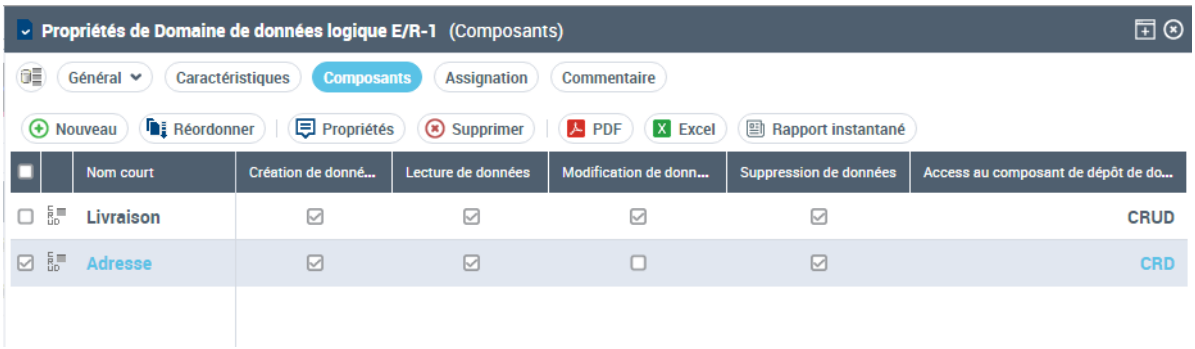
- Dans le diagramme, faites un clic droit sur l'objet en question et sélectionnez **Ajouter à (nom du domaine de données)**.
  Le nom du composant créé apparaît dans les propriétés du domaine de données. Par défaut il porte le nom de l'objet qu'il référence.

Définir le mode d'accès à l'objet référencé

Sur le composant vous pouvez définir le mode d'accès à l'objet référencé (création, lecture, suppression, etc.).

Pour définir le mode d'accès à l'objet dans le domaine de données :

1. Ouvrez les propriétés du domaine de données.
2. Cliquez sur la liste déroulante puis sur **Composants**.
   La liste des composants du domaine de données apparaît.
3. Sélectionnez le composant et cochez les cases qui correspondent aux types d'accès voulus (Création, Lecture, etc.).



LES VUES DE DONNÉES LOGIQUES

Une vue de données permet de représenter le périmètre couvert par un élément de modèle de donnée. Une vue de données est construite à partir d'une sélection de classes reliées dans le contexte spécifique de la vue.

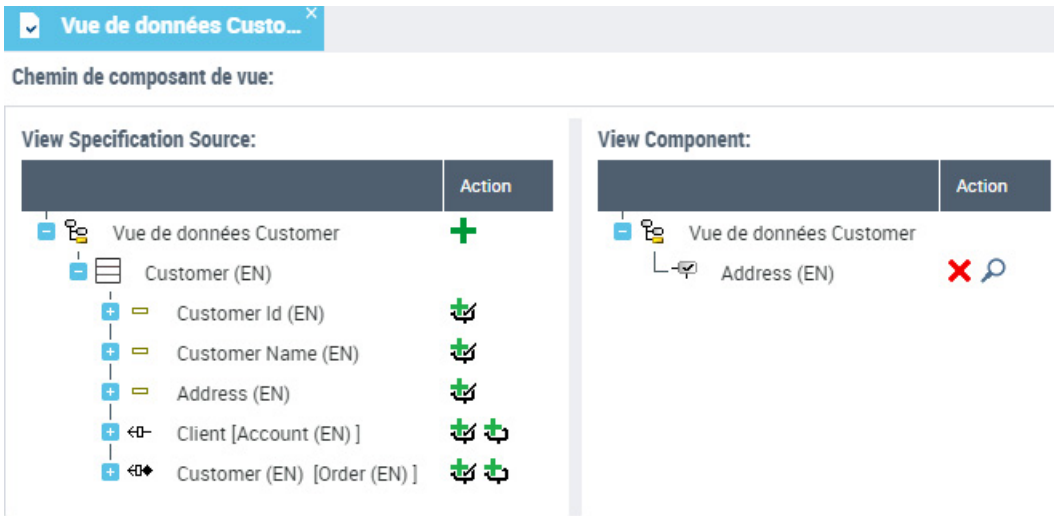
> ☛ *Selon le même principe, la vue de concept est utilisée pour visualiser le périmètre sémantique d'un objet métier. Pour plus de détails, voir "Définir des vues de concept", page 75.*

Créer une vue de données logiques

Créer une vue de données logiques consiste à :

- définir les objets sources, sur lesquels porte la vue (une classe ou une vue de données)
- définir plus précisément les propriétés des objets sources à prendre en compte dans la vue (les attributs, les parties)

```
Par exemple, pour la gestion de commandes, vous avez besoin
de récupérer l'adresse de livraison disponible pour chaque
client. Pour prendre en compte uniquement cette
information, vous allez créer une vue sur la classe Client
qui prend uniquement l'attribut "Adresse", sans tenir
compte des autres attributs que peut contenir la classe
Client.
```



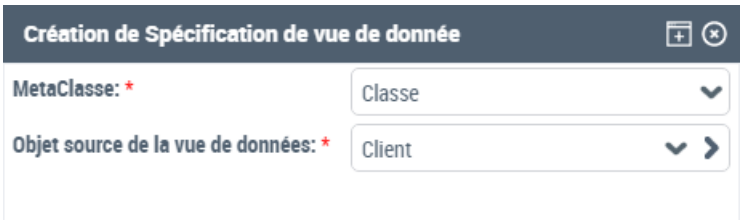
A partir des objets sources (arbre de gauche), vous pouvez définir dans la vue des composants embarqués et des composants référencés.

Un composant embarqué indique que toutes les informations qui composent l'objet source sont à prendre en compte lors de l'exploitation de la vue (par exemple les parties et les attributs associés à une classe). Un composant référencé ne fait que référencer l'objet dans la vue.

Créer une vue de données (à partir de la liste des vues)

Pour créer une vue de données avec **HOPEX Web Front-End** :

1. Cliquez sur le menu de navigation puis sur **Données logiques**.
2. Dans la zone d'édition, cliquez sur **Vues de données**.
3. Affichez toutes les vues de données.
4. Cliquez sur **Nouveau**.
   la fenêtre de création d'une vue de données apparaît.
5. Pour spécifier l'objet source de la vue de données, cliquez sur **Nouveau**.
6. Dans la fenêtre qui apparaît indiquez :
   - la **MetaClasse** sur laquelle porte la vue.
   - L'objet source de la vue de données.



7. Cliquez sur **Ajouter**.
8. Répétez la procédure pour ajouter éventuellement d'autres objets sources.
9. Cliquez sur **OK**.
   La nouvelle vue apparaît dans la liste des vues de données.

Créer une vue de données à partir d'un objet

Vous pouvez définir l'objet source d'une vue en créant la vue directement sur l'objet en question.

☛ *Il est possible par la suite d'ajouter un autre objet à la vue.*

Pour créer une vue de données sur un objet :

1. Faites un clic droit sur l'objet en question et sélectionnez **Nouveau** > **Vue de données**.
   L'assistant de création d'une vue de donnée apparaît.
2. Saisissez le nom de la vue.
3. Indiquez éventuellement le détenteur.
4. Cliquez sur **OK**.
   L'éditeur de vue apparaît.

Ajouter un objet source à la vue de données

Les informations qui composent une vue de données peuvent provenir de différents objets sources.

Vous pouvez définir un objet source lors de la création de la vue de données (voir "Créer une vue de données à partir d'un objet", page 12). Vous pouvez ajouter de nouveaux objets sources par la suite.

Pour ajouter un objet source à une vue de données :

1. Ouvrez la vue de données.
2. Du côté des objets sources, sous la colonne **Action**, cliquez sur le bouton  .
3. Dans la fenêtre qui apparaît, indiquez :
   - la **MetaClasse** sur laquelle porte la vue.
   - L'objet source de la vue de donnée.
4. Cliquez sur **Ajouter**.

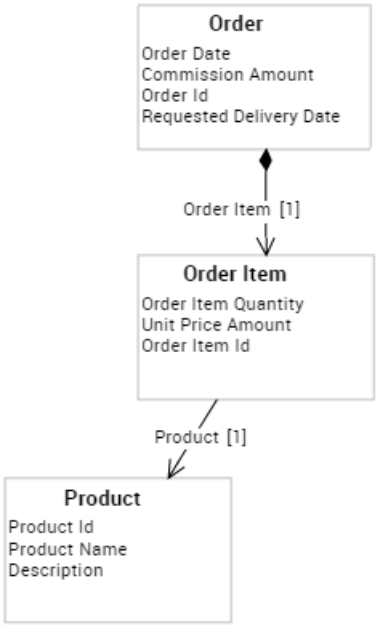
Une fois l'objet source défini, vous pouvez sélectionner les composants de cet objet à ajouter à la vue de données.

Affichage des objets sources dans la vue de données

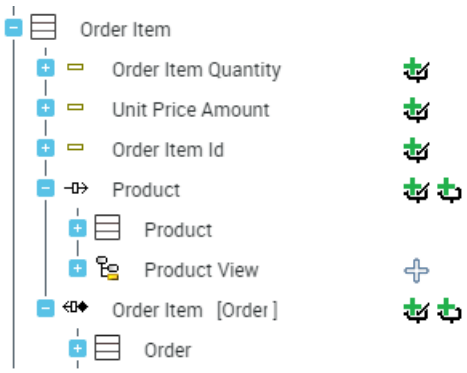
	**Classe**
	Attribut
	Partie
	Partie (classe composante)
	Partie (classe composée)

Exemple

Modèle logique



Vue de données logiques



Définir les composants de la vue de données

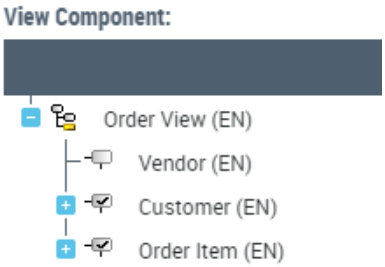
Composant embarqué

Un composant embarqué permet de prendre dans la vue toutes les informations qui composent l'objet (par exemple les parties et les attributs associés à une classe).

Pour ajouter un composant embarqué à la vue :

1. Ouvrez la vue de données.
2. Du côté des objets sources, sélectionnez l'élément à ajouter à la vue de données.

3. Sous la colonne **Action**, cliquez sur le bouton **Ajouter un composant d'inclusion de vue**.
   L'objet apparaît dans la partie droite de l'éditeur de vue.



Composant référencé

Référencer un composant dans la vue permet d'afficher l'objet dans la vue, sans embarquer toutes ses propriétés.

Vous pouvez référencer dans la vue les objets qui contiennent un certain nombre d'informations comme les classes. Pour les attributs, seul le bouton d'inclusion est proposé.

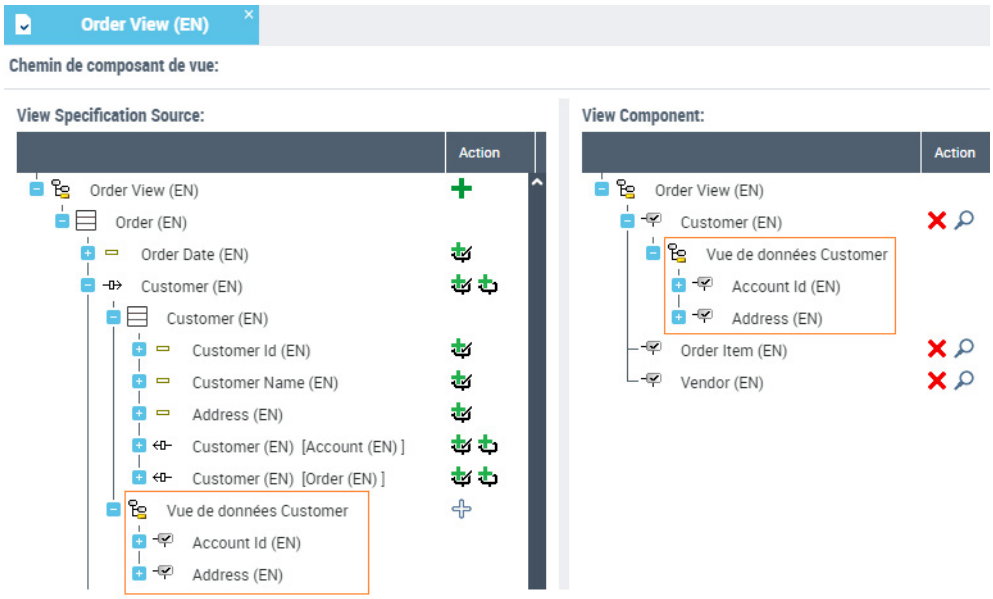
Pour référencer un objet dans la vue :

1. Ouvrez la vue de données.
2. Du côté des objets sources, sélectionnez l'élément à ajouter à la vue de données.
3. Sous la colonne **Action**, cliquez sur le bouton **Ajouter un composant de référencement de vue**.
   L'objet apparaît dans la partie droite de l'éditeur de vue.

Utiliser une vue dans une autre vue

Lorsque vous embarquez une classe dans une vue de données, tous les attributs de la classe sont ajoutés par défaut dans la vue. Vous pouvez limiter la liste des attributs à ceux déjà définis dans une vue.

Ci-dessous, seuls les attributs définis dans la vue "Customer" (Account Id et Address) sont ajoutés à la vue "Order".



Pour ajouter une vue de données (source) à une vue de données (cible) :

1. Ouvrez la vue de données cible.
2. Sous la partie gauche, dépliez la classe sur laquelle porte la vue source à ajouter.

   ☛ *La classe source a été préalablement embarquée dans la vue cible.*

3. Sélectionnez la vue source et sous la colonne **Action**, cliquez sur **Ajouter une vue**.
   La vue associée à la classe apparaît dans la partie droite de l'éditeur de vue, sous le nom de la classe en question.

RELIER LES CONCEPTS MÉTIER AUX DONNÉES LOGIQUES

Les concepts métier définis dans **HOPEX Information Architecture** peuvent être mis en œuvre dans le SI avec le support de la méthode et du formalisme UML.

Le travail de "réalisation des concepts" consiste ainsi à rapprocher les éléments des modèles de données avec les concepts métier afin de :

- définir de manière plus précise les objets manipulés au niveau de l'architecture du SI,
- assurer un meilleur partage du vocabulaire et une meilleure communication globale entre intervenants métier et intervenants SI.

La réalisation de concept

A travers le concept de "réalisation", vous pouvez relier des objets de la vue logique à des éléments de dictionnaire.

Les éléments logiques auxquels peuvent être reliés des éléments de dictionnaire sont :

- les classes
- les attributs de classe

Créer une réalisation de concept

Pour associer une classe à un concept, par exemple :

1. Dans le bureau Information Architecture, cliquez sur le menu de navigation puis sur **Données logiques**.
2. Dans la zone d'édition, cliquez sur **Hiérarchie des paquetages**.
   La liste des paquetages du référentiel apparaît dans la fenêtre d'édition.
3. Dépliez le dossier du paquetage qui vous intéresse.
4. Sélectionnez la classe que vous voulez relier à un concept et ouvrez sa fenêtre de propriétés.
5. Cliquez sur la page **Caractéristiques > Réalisation**.
6. Cliquez sur **Nouveau**.
   La fenêtre **Ajout d'une réalisation détenue** apparaît.
7. Dans le champ **Type d'objet**, sélectionnez "Réalisation d'information métier" et cliquez sur **Suivant**.
8. Dans la champ **MetaClasse** sélectionnez "Concept".
9. Dans le champ **Information métier réalisée**, sélectionnez le concept qui vous intéresse.
10. Cliquez sur **Ajouter**.
    La réalisation de concept apparaît dans la fenêtre de propriétés de la classe avec le nom du concept sélectionné.

Créer une réalisation de composant de concept

De la même façon, vous pouvez relier un attribut d'une classe à un composant de concept. Il faut au préalable avoir créé une réalisation de concept entre la classe qui détient l'attribut et le concept qui détient le composant.

Pour relier un attribut à un composant de concept :

1. Sélectionnez l'attribut et affichez ses propriétés.

   ```
   Par exemple, sélectionnez l'attribut "Code client" de la
   classe "Client".
   ```

2. Dans la fenêtre des propriétés, cliquez sur la liste déroulante puis sur **Caractéristiques** > **Réalisations**.
3. Dans le cadre **Réalisation de composant**, cliquez sur **Nouveau**.
4. Sélectionnez la MetaClass que vous voulez créer, à savoir "Réalisation de composant d'information métier", et cliquez sur **Suivant**.
   Pour sélectionner le composant d'information métier à relier à l'attribut :
5. Dans le champ **Contexte de réalisation**, sélectionnez la réalisation concernée.
6. Cliquez sur la flèche située à droite du champ **Composant réalisé** pour voir la liste des composants associés au concept.
7. Sélectionnez le composant d'information métier et cliquez sur **OK**.

Le rapport de réalisation

Vous pouvez utiliser le rapport de réalisation pour visualiser la couverture de réalisation (ou d'implémentation) des éléments de dictionnaire par un autre élément de l'architecture.

Pour accéder au rapport :

1. Cliquez sur le menu de navigation puis sur **Rapports**.
2. Dans le volet de navigation, cliquez sur **Rapport de réalisation**.

Paramètres du rapport

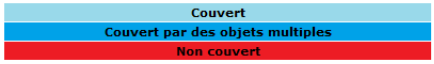
Il s'agit ici de définir les données en entrée du rapport.

Paramètres	Type du paramètre	Contraintes
Liste des objets	Acteur, Application, Bibliothèque, Capacité, Classe, Concept, Concept d'état, Concept événement, Concept type, Contenu, Contrat d'échange, Cycle de vie de concept, Echange, Entité (MD) Fonctionnalité Métier Processus applicatif, Processus fonctionne, Processus métier, Processus organisation-nel, Service applicatif Vue de données Vue de concept	Un objet obligatoire.
Mode d'affichage des réalisateurs	Afficher les réalisateurs Afficher les réalisateurs et les pourcentages Ne pas afficher les réali-sateurs	
Mode d'affichage des indicateurs	Booléen	

Exemple de rapport

L'exemple ci-dessous permet de visualiser le taux de couverture des objets passé en paramètre.

On note que les réalisation des composants structurels des concepts passés en paramètre sont également affichés.



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

LE DIAGRAMME DE CLASSES

HOPEX Information Architecture met à votre disposition deux formalismes pour décrire les données logiques :

- le paquetage de données, pour construire les diagrammes de classes (notation UML)
- le modèle de données, pour les diagrammes de données (notations standard, IDEF1X, I.E, Merise). Voir "Le modèle de données", page 23.

La description des données en notation UML s'effectue dans un diagramme de classes.

Créer un paquetage

Un paquetage partitionne le domaine d'étude et les travaux associés. Il est destiné à contenir les éléments modélisés. La représentation graphique de tout ou partie de ces éléments se fait dans un diagramme de classes.

Une base de données peut être reliée dès sa création à un paquetage. C'est sur cette base de données que pourront se lancer par la suite les différents outils de traitements des données (génération, synchronisation etc.). Le paquetage de la base de données est le détenteur par défaut des classes et associations représentées dans le diagramme de classes.

Créer un paquetage

Pour créer un paquetage dans **HOPEX Information Architecture** :

1. Dans le bureau, cliquez sur le menu de navigation puis sur **Données logiques**.
2. Dans le volet de navigation, cliquez sur **Paquetages**.
   La liste des paquetages apparaît dans la zone d'édition.
3. Cliquez sur le bouton **Nouveau**.
   Le paquetage créé est ajouté dans la liste des paquetages. Vous pouvez modifier son nom.

Relier un paquetage à une base de données

Pour créer un paquetage à partir d'une base de données :

1. Cliquez sur le menu de navigation puis sur **Données logiques**.
2. Dans le volet de navigation, cliquez sur **Bases de données**.
3. Dans la zone d'édition, cliquez sur l'icône de la base de données concernée et sélectionnez **Nouveau** > **Paquetage**.

Pour relier un paquetage existant à une base de données :

- Cliquez sur l'icône de la base de données et sélectionnez **Relier** > **Paquetage**.

Créer un diagramme de classes

Un diagramme de classes est une représentation graphique permettant de représenter la structure statique d'un système, en particulier les types d'objets manipulés dans le système, leur structure interne et les relations qui existent entre eux.

Un diagramme de classes inclut :

- Des classes, qui représentent les concepts de base (client, compte, produit, etc.).
- Des parties, qui définissent les relations entre les différentes classes.
- Des attributs, qui décrivent les caractéristiques des classes.
- Des opérations, qui peuvent être effectuées sur les objets de la classe.

> ☛ *Les opérations ne sont pas prises en compte par les outils de **HOPEX Information Architecture** (synchronisation, génération etc.).*

Pour créer le diagramme de classes d'un paquetage dans **HOPEX Information Architecture** :

1. Dans le bureau, cliquez sur le menu de navigation puis sur **Données logiques**.
2. Dans le volet de navigation, cliquez sur **Paquetages**.
   La liste des paquetages apparaît dans la zone d'édition.
3. Cliquez sur l'icône du paquetage concerné et sélectionnez **Nouveau** > **Diagramme de classes**.

Le nouveau diagramme de classes s'ouvre.

Notez que lorsque vous créez un paquetage à partir d'une base de données, un diagramme de classes est créé automatiquement en même temps.

Plus plus de détails sur la construction d'un diagramme de classes, voir "Le diagramme de classes".

LE MODÈLE DE DONNÉES

Pour vous aider à décrire les données logiques, **HOPEX Information Architecture** propose une notation simple permettant de représenter tous les cas courants et basée sur le modèle de données.

Un modèle de données permet de représenter la structure statique d'un système, en particulier les types d'objets manipulés dans le système, leur structure interne et les relations qui existent entre eux.

Avec la couche physique d'**HOPEX Information Architecture**, les modèles de données peuvent être mis en correspondance avec des modèles physiques.

Les points suivants sont abordés ici :

Des notations spécialisées sont également disponibles :

PRINCIPES DE LA MODÉLISATION DES DONNÉES

Modéliser les données consiste à identifier les entités considérées d'intérêt pour représenter l'activité de l'entreprise, et définir les associations qui existent entre elles. Il faut que les entités et les associations qui constituent le diagramme de données associé à un secteur de l'entreprise suffisent à le décrire complètement sur le plan sémantique. En d'autres termes, on doit pouvoir décrire l'activité de l'entreprise en utilisant seulement ces entités et associations.

Ceci n'implique pas que, pour chaque mot ou verbe utilisé pour cette explication, il y ait un correspondant direct dans le diagramme de données. Il s'agit de pouvoir traduire ce que l'on veut exprimer, au travers des entités et des associations.

La spécification des modèles de données est souvent considérée comme la partie la plus importante dans la modélisation d'un système d'information.

Synthèse des concepts

Modèle de données

Le modèle de données permet de représenter la structure statique d'un système, en particulier les types d'objets manipulés dans le système, leur structure interne et les relations qui existent entre eux.

Un modèle de données regroupe un ensemble d'entités avec leurs attributs, les associations qui existent entre ces entités, des contraintes qui portent sur ces entités et associations, etc.

Diagramme de données

Un diagramme de données est une représentation graphique du modèle ou d'une partie du modèle.

Un diagramme de données est représenté par :

- des *entités*, qui représentent les concepts de base (client, compte, produit, etc.).
- des *associations*, qui définissent les relations entre les différentes entités.
- des *attributs*, qui décrivent les caractéristiques des entités et, dans certains cas, des associations.

L'attribut ou l'ensemble d'attributs qui permet d'identifier de façon unique l'entité est appelé identifiant.

Le diagramme de données est complété avec la définition des multiplicités.

CONSTRUIRE UN MODÈLE DE DONNÉES

Pour la version HOPEX Windows Front-End, voir le guide au format .pdf livré sur le site Support.

Créer le modèle de données

L'utilisation des modèles de données nécessite de cocher une option. Voir "Options de modélisation des données", page 1.

Pour créer un modèle de données :

1. Dans le bureau, cliquez sur le volet de navigation **Données logiques**.
2. Affichez la liste des **Modèles de données**.
   La liste des modèles de données apparaît dans la fenêtre d'édition.
3. Dans la fenêtre d'édition, cliquez sur le bouton **Nouveau**.
   La fenêtre de création d'un modèle de données apparaît.
4. Saisissez le nom du modèle de données, et éventuellement un détenteur.
5. Cliquez sur **OK**.
   Le modèle de données créé apparaît dans la liste des modèles de données.

Créer un diagramme de données

Un diagramme de données est une représentation graphique du modèle ou d'une partie du modèle.

Pour créer un diagramme de données :

- Faites un clic droit sur le modèle de données et sélectionnez **Nouveau > Diagramme de données**.
  Le diagramme de données s'ouvre.

Les type de données

Un type de données permet de mettre en commun des caractéristiques communes à plusieurs attributs.

Lorsque vous créez un modèle de données, le paquetage de types de données "Référence de types de données" lui est automatiquement relié par défaut. Ainsi, la liste des *types de données* qu'il contient est disponible sur chaque attribut des entités du modèle. Mais vous pouvez lui attribuer un autre *paquetage de types de données*.

Le paquetage de données de référence d'un modèle de données est visible dans la fenêtre de propriétés du modèle, sous la page **Caractéristiques**.

Pour plus d'informations, voir "Paquetages de types de données", page 106.

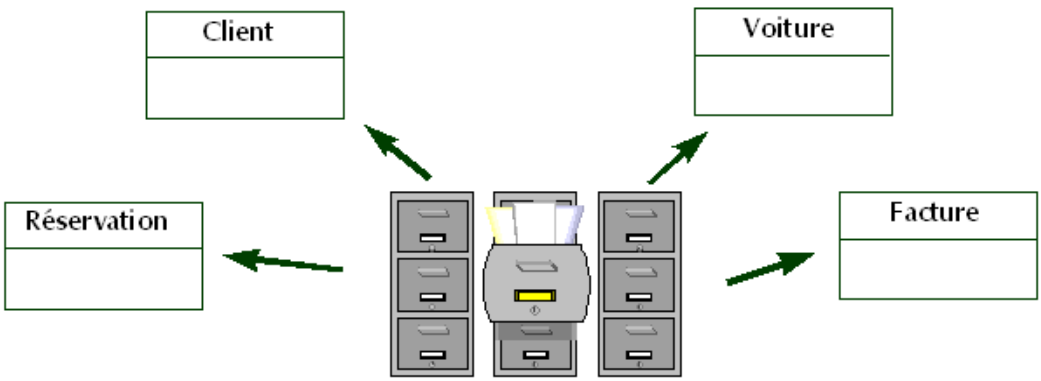
LES ENTITÉS

> 🕮 *Une entité est un regroupement d'objets possédant des caractéristiques communes et un comportement semblable. Les entités sont des éléments de gestion considérés d'intérêt pour représenter l'activité de l'entreprise et sont donc généralement conservés à cet effet. Elles pourront, par exemple, donner lieu à des tables dans une base de données.*

Une *entité* est décrite par une liste d'attributs.

Une entité est reliée à d'autres entités via des associations. L'ensemble des entités et des associations constitue le noyau d'un diagramme de données.

Nous pouvons illustrer la notion d'entité en comparant les entités à des fiches dans des tiroirs.



Les entités peuvent représenter des objets de gestion.

```
Exemples : Client, Commande, Produit, Personne, Société,
etc.
```

Les entités peuvent représenter des objets techniques utilisés dans l'industrie.

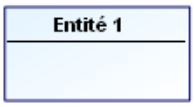
```
Exemples : Alarme, Capteur, Zone
```

Créer une entité

Pour créer une entité :

1. Dans la barre d'insertion du diagramme de données, cliquez sur le bouton **Entité** 
2. Cliquez sur le plan de travail du diagramme.
   La fenêtre **Ajout d'une entité** s'ouvre.
3. Saisissez le **Nom** de l'entité.

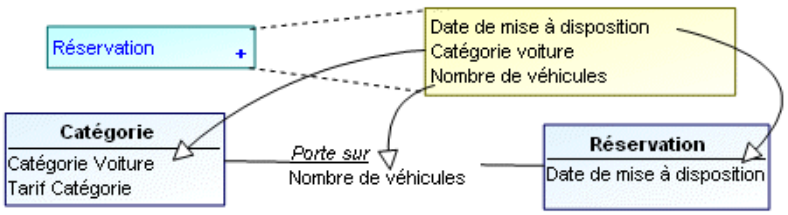
   ☛ *Quand le bouton **OK** ou **Ajouter** est grisé, c'est que la fenêtre où il apparaît n'a pas été complètement renseignée.*
4. Cliquez sur **Ajouter**.
   L'entité apparaît dans le diagramme.



☺ *Vous pouvez créer plusieurs entités à la suite sans revenir à la barre d'outil en double-cliquant sur le bouton **Entité**. Pour revenir ensuite au mode normal, utilisez la touche <Echap>, ou cliquez sur la flèche de la barre d'outils.*

Les attributs

Les entités et les associations peuvent être caractérisées par des *attributs*.



Ces attributs ont pu, par exemple, être révélés par l'étude du contenu des messages qui circulent à l'intérieur de l'entreprise.

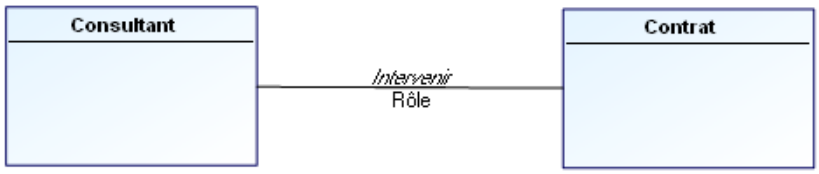
🕮 *Un attribut est la donnée élémentaire mémorisée dans le système d'information de l'entreprise. Un attribut est une propriété quand il décrit une entité ou une association, un identifiant quand il est choisi comme moyen d'identification de chaque exemplaire d'une entité.*

Exemples :

- "Nom du client" (attribut de l'entité client).
- "N° client" (identifiant de l'entité client).
- "Solde du compte" (attribut de l'entité compte).

Un attribut caractérise une association quand l'attribut dépend de l'ensemble des entités participant à l'association.

Dans l'exemple présenté ci-après, le rôle qu'un "Consultant" a joué sur un "Contrat" dépend du consultant et du contrat, donc de l'association "Intervenir".



Créer un attributs

Pour définir un attribut sur une entité :

1. Cliquez avec le bouton droit sur l'entité et sélectionnez **Propriétés**.
   La fenêtre des propriétés de l'entité s'ouvre.
2. Cliquez sur la liste déroulante puis sur **Attributs**.
   La page des attributs apparaît.
3. Cliquez sur le bouton **Nouveau**.
   Un nom vous est automatiquement proposé pour ce nouvel attribut. Vous pouvez le modifier.
4. Cliquez sur **OK**.

Vous pouvez préciser son **Type de données**.

```
Exemple : Numérique.
```

☛ *Voir "Types des attributs", page 103 pour plus de détails sur les types de données qui peuvent être affectés à un attribut.*

Attributs hérités

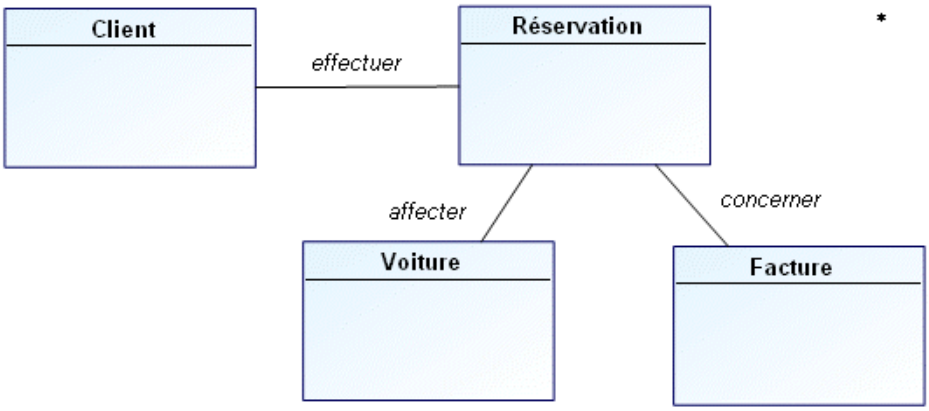
Lorsqu'une généralisation existe entre une entité générale et une entité particulière, l'entité particulière hérite des attributs de l'entité générale.

Voir "Généralisations", page 43.

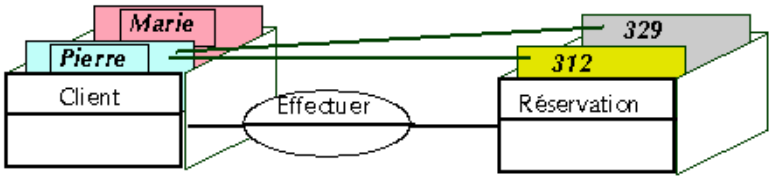
LES ASSOCIATIONS

📖 *Une association est une relation entre deux ou plusieurs entités. Elle peut comporter des attributs qui caractérisent l'association entre ces entités.*

Les *associations* peuvent être comparées à des liens entre des fiches.



Le dessin suivant permet de visualiser "en trois dimensions" les situations qu'un diagramme de données permet de mémoriser.



Pierre et Marie sont des clients. Pierre a effectué les réservations numéros 312 et 329.

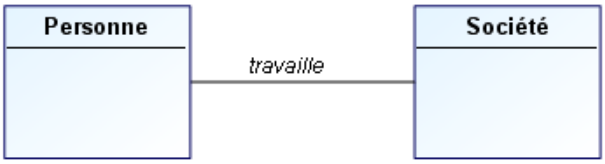
Un diagramme de données doit permettre de mémoriser toutes les situations du contexte de l'entreprise, mais rien que celles-là.

☛ *Le diagramme ne doit pas permettre de représenter des situations irréalistes ou aberrantes.*

Exemples d'association :

- Un client passe une commande.
- Une commande comprend plusieurs produits.

- Une personne travaille pour une société.



- Une alarme est déclenchée par un capteur.
- Un capteur couvre une zone.
- Une fenêtre affiche une chaîne de caractères.

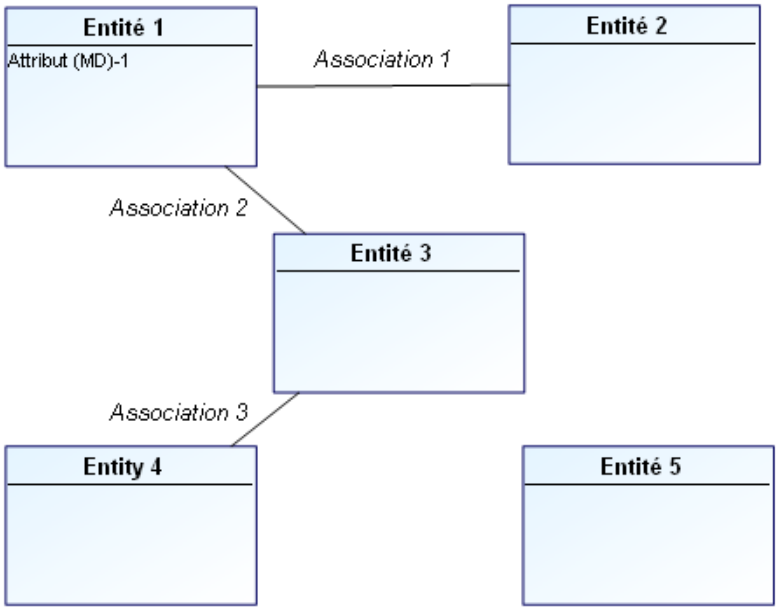
Créer une association

Pour créer une association :

1. Dans la barre d'insertion du diagramme de données, cliquez sur le bouton **Association** 
2. Cliquez sur une des entités concernées et, en gardant le bouton de la souris enfoncé, déplacez le pointeur jusqu'à la deuxième entité, avant de relâcher votre pression.
   L'association apparaît sous forme d'un trait.
3. Pour préciser le nom de l'association, cliquez sur le bouton droit sur l'association et sélectionnez **Propriétés**.

   ☛ *Veillez à bien cliquer sur le trait qui matérialise l'association, et non sur un des rôles situés aux extrémités de l'association.*
4. Dans la page **Caractéristiques**, dans le champ **Nom Local**, saisissez le nom de l'association.
5. Cliquez sur **OK**.

Exemple



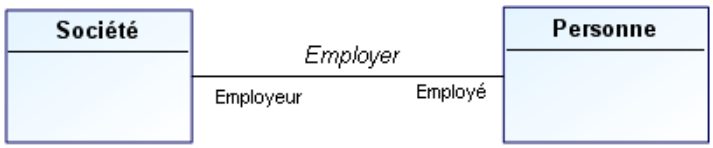
En cas d'erreur, vous pouvez supprimer un élément ou un lien en sélectionnant la commande **Supprimer** du menu contextuel de l'élément ou du lien.

Définir le rôle des associations

📖 *Un rôle permet d'indiquer une des entités concernées par l'association. L'indication des rôles est particulièrement importante dans le cas d'une association entre une entité et elle-même.*

Chaque extrémité d'une association permet de préciser le *rôle* joué par chaque entité dans l'association.

Visuellement, le nom du rôle se distingue du nom d'une association, car il est placé près de son extrémité. De plus, il apparaît en caractères droits, alors que le nom de l'association est en italique.



☺ *La barre d'état (située au bas de la fenêtre) permet aussi de distinguer les différentes zones : lorsque vous déplacez la souris le long de l'association, elle indique si vous vous trouvez sur l'association ou sur un rôle.*

Lorsque deux entités sont reliées par une seule association, le nom des entités suffit souvent à caractériser le rôle ; nommer les rôles prend tout son intérêt lorsque plusieurs associations relient deux entités.

Multiplicités

Chaque rôle d'une association porte une indication de multiplicité qui montre combien d'objets de l'entité considérée peuvent être liés à un objet de l'autre entité. La multiplicité est une information portée par le rôle, sous la forme d'une expression entière bornée. On l'indique en particulier pour chacun des rôles que jouent les entités dans une association.

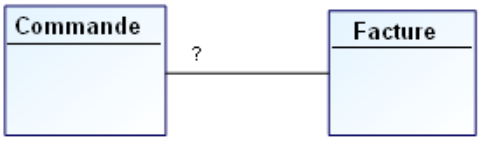
La multiplicité exprime le nombre de participations minimum et maximum d'un objet donné d'une entité à une association.

Les multiplicités usuelles sont "1", "0..1", "*" ou "0..*", "1..*", et "M..N" où "M" et "N" sont des entiers :

- La multiplicité "1" indique que chaque objet de l'entité est relié par cette association une fois et une seule.
- La multiplicité "0..1" indique qu'un objet de l'entité ne peut être relié par cette association qu'une fois au plus.
- La multiplicité "*" ou "0..*" indique qu'un objet de l'entité peut être relié par l'association une ou plusieurs fois ou pas du tout.
- La multiplicité "1..*" indique que chaque objet de l'entité est obligatoirement relié par l'association et qu'il peut l'être plusieurs fois.
- La multiplicité "M..N" indique que chaque objet de l'entité est obligatoirement relié par l'association au moins "M" fois et qu'il peut l'être au maximum "N" fois.

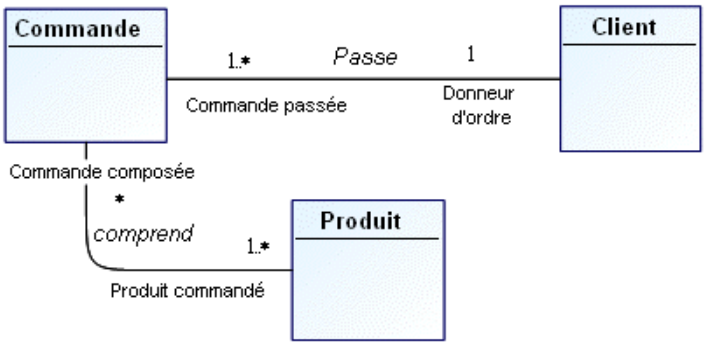
1	Un et un seul
0..1	Zéro ou un
M..N	De M à N (entiers naturels)
*	De zéro à plusieurs
0..*	De zéro à plusieurs
1..*	De un à plusieurs

Exemple :



0..1	A une commande correspond une facture au maximum ou aucune.
*	Aucune restriction n'est imposée sur le nombre de factures correspondant à une commande.
1	A chaque commande correspond une facture et une seule.
1..*	A chaque commande correspond une ou plusieurs factures.

Autres exemples de multiplicité :



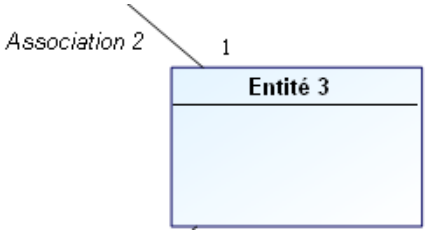
1..*	Un client peut passer une ou plusieurs commandes.
1	Une commande est passée par un et un seul client.
1..*	Une commande comprend un ou plusieurs produits.
*	Un produit peut faire partie de plusieurs commandes, ou d'aucune.
0..1	Une personne travaille pour une société.
1..*	Une alarme est déclenchée par un ou plusieurs capteurs.
1	Un capteur couvre une et une seule zone.
1..*	Une fenêtre affiche une ou plusieurs chaînes de caractères.

Pour préciser la multiplicité d'un rôle :

1. Dans le diagramme de données, cliquez avec le bouton droit sur le trait qui se trouve entre l'association et l'entité, afin d'ouvrir le menu contextuel du rôle.

2. Cliquez sur **Propriétés**.
   La fenêtre de propriétés du rôle s'ouvre.
3. Cliquez sur liste déroulante puis sur **Caractéristiques**.
4. Dans le champ **Multiplicité**, sélectionnez la multiplicité voulue.

La représentation de l'association change en fonction de la nouvelle valeur de ses multiplicités.



☛ *Dans HOPEX Windows Front-End, la multiplicité est également affichée dans le menu contextuel du rôle. Si le menu affiché ne propose pas les multiplicités, vérifiez que vous avez bien cliqué sur le trait qui matérialise le rôle, et non sur l'association.*

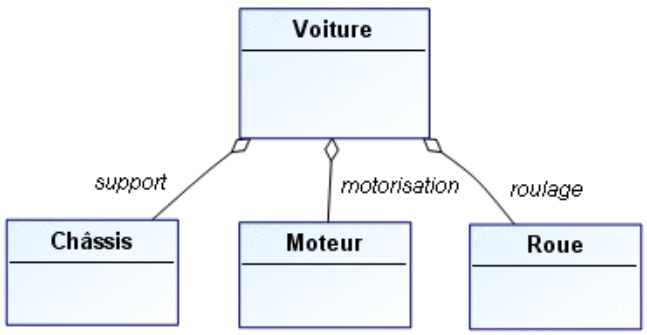
Autres caractéristiques des associations

Agrégation

L'agrégation est une forme particulière d'association qui indique que l'une des entités contient l'autre.

Exemple d'*agrégation* :

```
Une voiture comprend un châssis, un moteur et des roues.
```



Pour définir l'agrégation entre les entités "Voiture" et "Moteur" :

1. Cliquez avec le bouton droit sur le rôle joué par l'entité "Voiture" dans son association avec l'entité "Moteur" et sélectionnez **Propriétés**.
   Les propriétés du rôle apparaissent.
2. Cliquez sur **Caractéristiques**.

3. Dans le champ **Tout/partie,** sélectionnez "Agrégat".
   Un losange représentant l'agrégation apparaît alors sur le rôle.

   ☺ *Dans HOPEX Windows Front-End vous pouvez définir l'agrégation directement dans le menu contextuel du rôle.*

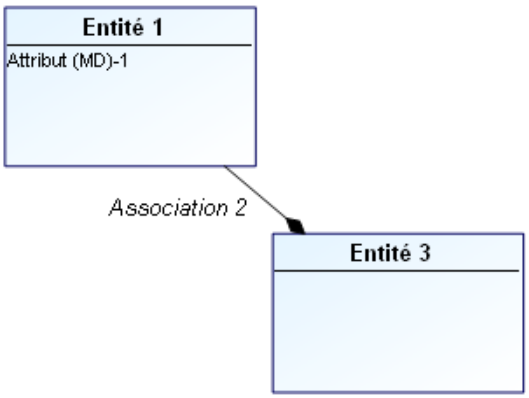
Composition

La composition est une agrégation forte pour laquelle la durée de vie des composants coïncide avec celle du composé. Une composition est une agrégation immuable avec une multiplicité 1.

Exemple de *composition* :

```
Une commande est composée de plusieurs lignes de commande
qui n'existent plus si la commande est supprimée.
```

La composition est matérialisée par un losange noir.



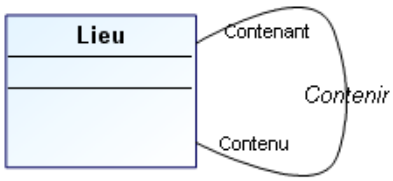
Pour préciser la composition d'un rôle :

1. Cliquez avec le bouton droit sur le rôle et sélectionnez **Propriétés**.
   Les propriétés du rôle apparaissent.
2. Cliquez sur la liste déroulante puis sur **Caractéristiques**.
3. Dans le champ **Tout/partie,** sélectionnez "Composé".

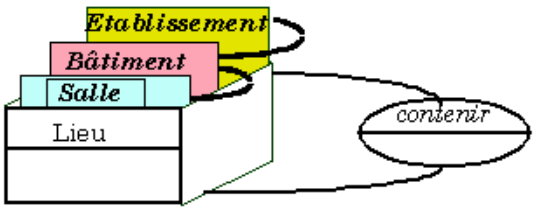
   ☺ *Dans HOPEX Windows Front-End vous pouvez définir la composition directement dans le menu contextuel du rôle.*

Utiliser les associations réflexives

Certaines associations mettent en jeu plusieurs fois la même entité.



Une salle de classe, un bâtiment, un établissement scolaire sont tous des lieux.



Une salle de classe est contenue dans un bâtiment, lui-même contenu dans un établissement scolaire.

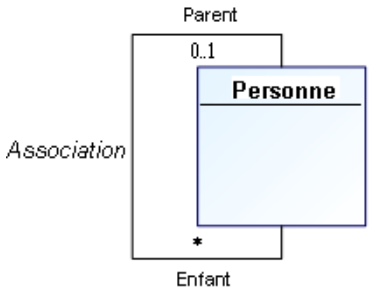
Une association réflexive porte deux fois sur la même entité.

Pour créer une association réflexive :

1. Dans la barre d'insertion du diagramme de données, cliquez sur le bouton **Association** 

2. Cliquez dans l'entité concernée et faites glisser la souris en dehors de cette entité, puis revenez-y ; relâchez ensuite le bouton de la souris. L'association réflexive apparaît sous forme d'un demi-cercle en ligne brisée.

> ☛ *Dans le cas d'une association entre une entité et elle-même, il est indispensable de préciser les rôles afin de distinguer les liens correspondants dans le dessin.*



Ci-dessus "Parent" et "Enfant" sont les deux *rôles* joués par l'entité "Personne" dans l'association.

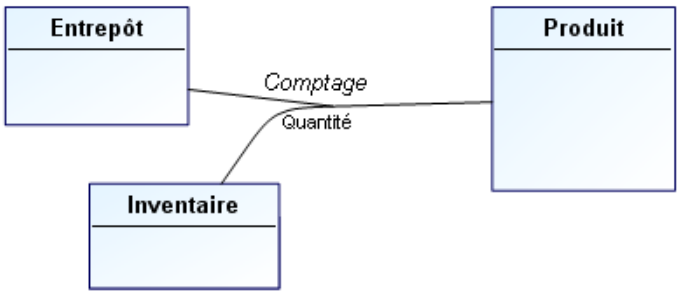
> 🕮 *Un rôle permet d'indiquer une des entités concernées par l'association. L'indication des rôles est particulièrement importante dans le cas d'une association entre une entité et elle-même.*

Vous pouvez segmenter une ligne en y ajoutant des sommets pour modifier le tracé des traits. Vous pouvez en particulier segmenter un rôle, pour lui faire contourner un obstacle par exemple. Pour cela, cliquez sur la ligne en tenant la touche <Ctrl> enfoncée et déplacez cette ligne.

Définir une association "plus que binaire"

Certaines associations associent non pas deux, mais davantage d'entités. Ces associations sont, en principe, rares.

Exemple : Lors d'un inventaire, une certaine quantité de produit a été comptée dans chaque entrepôt.



Pour créer une association ternaire :

1. Dans le diagramme de données, créez l'association entre deux des entités.
2. Cliquez sur le bouton **Rôle de l'association**  et reliez la troisième entité à l'association.

LES CONTRAINTES

📖 *Une contrainte représente un contrôle ou une règle de gestion qui doit être appliquée lors de l'exécution d'un traitement.*

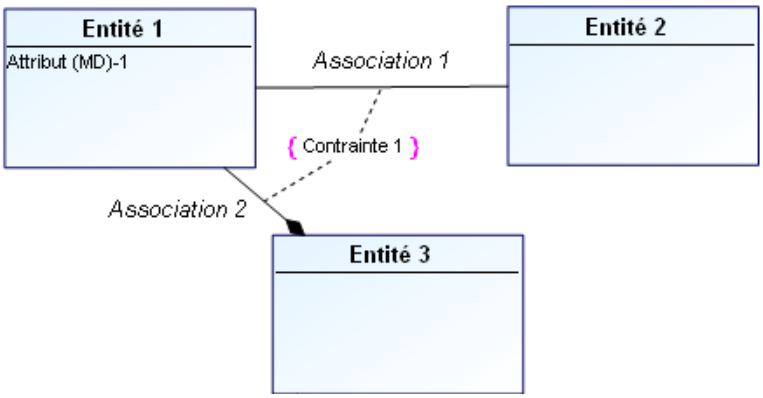
La plupart des *contraintes* impliquent les associations entre les entités.

Exemples de contraintes :

```
La personne responsable d'un service doit appartenir à ce
service.

Toute commande facturée doit avoir été préalablement
livrée.

La date de livraison doit être postérieure à la date de
commande.

Un capteur couvrant une zone ne peut déclencher qu'une
alarme protégeant cette même zone.
```

Pour créer une contrainte :

1. Dans la barre d'insertion du diagramme, cliquez sur le bouton **Contrainte** {}.
2. Cliquez sur une des associations concernées par la contrainte et faites glisser la souris jusqu'à la deuxième association, avant de relâcher votre pression.
   La fenêtre **Ajout d'une contrainte** s'ouvre.
3. Saisissez le nom de la contrainte puis cliquez sur **Ajouter**.
   La contrainte apparaît dans le diagramme.



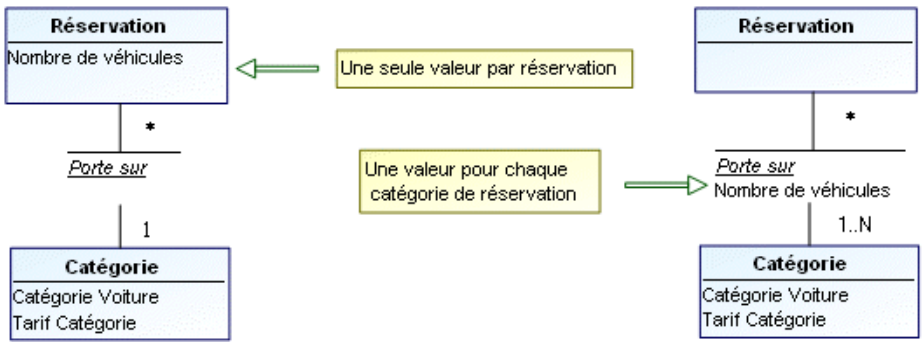
☺ *Enregistrez régulièrement votre dessin à l'aide du bouton* ***Enregistrer***

RÈGLES DE NORMALISATION

Les formes normales sont des règles qui visent à éviter des erreurs de modélisation. A ce jour, il existe de six à sept formes normales. Nous allons voir les trois premières.

Première forme normale

Règle : La valeur d'un attribut est fixée de manière unique dès que l'on connaît le ou les objets concernés.

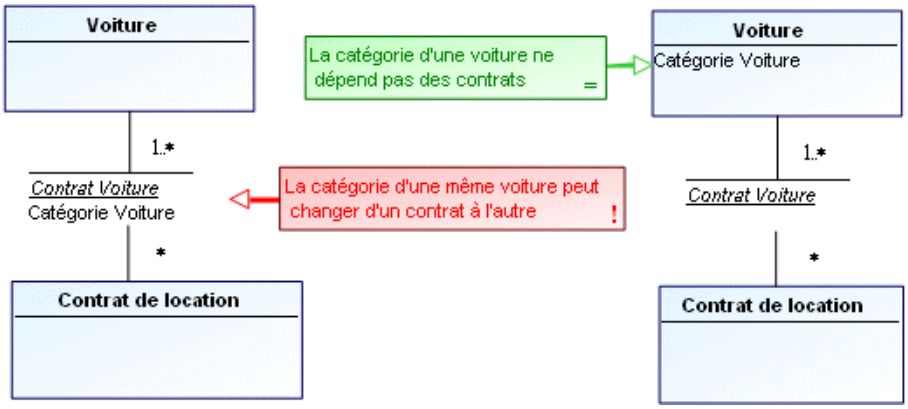


Si le nombre de véhicules est porté par l'entité "Réservation", on ne peut indiquer que le nombre total de véhicules pour une réservation. On doit donc faire une réservation par catégorie de véhicule loué (multiplicité 1).

Si le nombre de véhicules est porté par l'association, on peut préciser le nombre de véhicules réservés pour chaque catégorie sur l'association. On peut donc faire une seule réservation pour plusieurs catégories de véhicules (multiplicité 1..N).

Deuxième forme normale

Règle : La valeur d'un attribut d'association n'est fixée que lorsque l'on connaît toutes les entités concernées.

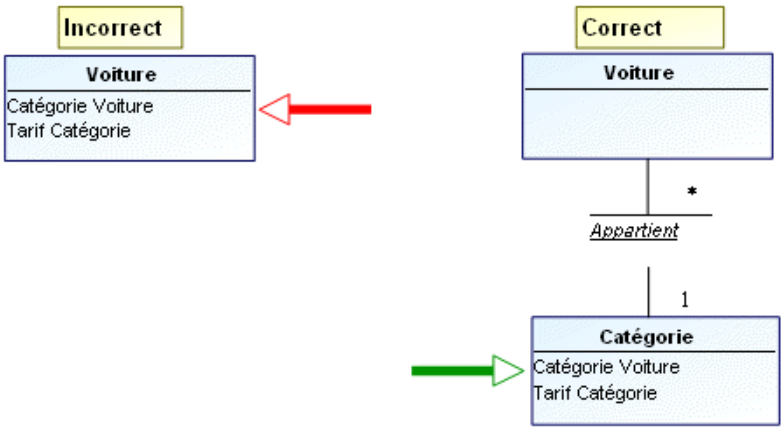


Le fait que la catégorie de voiture porte sur l'association "Catégorie Voiture" suppose que la catégorie de la voiture puisse changer d'un contrat à l'autre, ce qui ne serait pas très honnête.

Pour que la catégorie de la voiture ne dépende pas du contrat, il faut qu'elle soit portée par l'entité "Voiture".

Troisième forme normale

Règle : Un attribut doit dépendre directement et uniquement de l'entité qu'il décrit.



Si le "Tarif Catégorie" est porté par l'entité "Voiture", cela signifie que deux voitures de la même catégorie peuvent avoir un "Tarif Catégorie" différent.

Pour éviter cela, il faut créer une entité "Catégorie" qui portera le tarif.

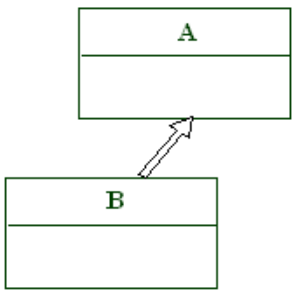
☺ *Cette règle permet de faire émerger des concepts qui n'apparaissent pas dans la première ébauche de diagramme de données.*

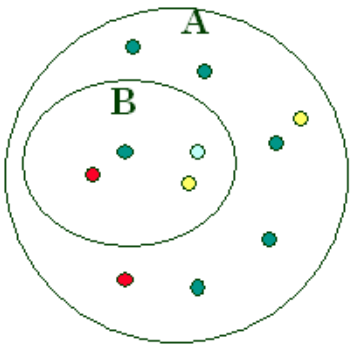
GÉNÉRALISATIONS

Voir :

Qu'est-ce qu'une généralisation ?

📖 *Une généralisation représente une relation d'héritage entre une entité générale et une entité plus spécifique. L'entité spécifique est cohérente avec l'entité plus générale et hérite de ses caractéristiques et de son comportement. Elle peut cependant comporter des attributs ou des associations supplémentaires. Tout objet de l'entité spécifique est aussi un objet de l'entité générale.*





L'entité A est une *généralisation* de l'entité B. Cela suppose que tous les objets de l'entité B sont aussi des objets de l'entité A. Autrement dit, B est un sous-ensemble de A. B est alors la sous-entité, A la super-entité.

```
Exemple :
A : Personne, B : Parisien.
```

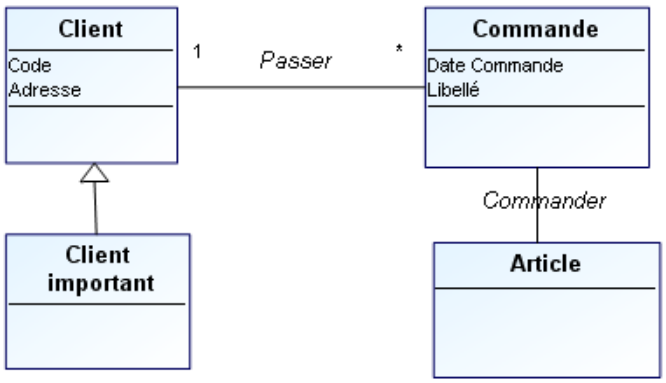
B étant un sous-ensemble de A, les objets de l'entité B "héritent" des caractéristiques de ceux de l'entité A.

Il n'est donc pas nécessaire de décrire de nouveau pour l'entité B :

- Ses attributs
- Ses associations

Exemple :

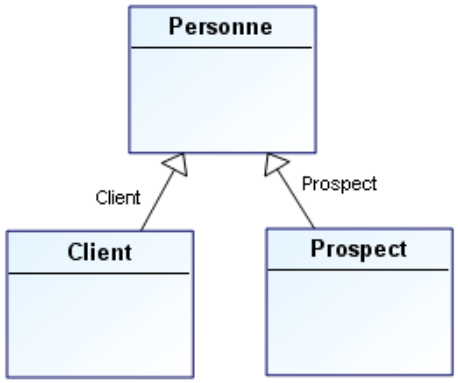
L'entité "Client important" qui représente les clients dont le "C.A. sur les 12 derniers mois" dépasse 1 million d'euros, peut être une spécialisation de l'entité "Client".



Dans l'exemple qui précède, les associations et les attributs spécifiés pour "Client" sont aussi valables pour "Client important".

Autres exemples de généralisations :

"prospect" et "client" sont deux sous-entités de "personne".



"commande export" est une sous-entité de l'entité "commande".

"personne physique" et "personne morale" sont deux sous-entités de l'entité "personne".

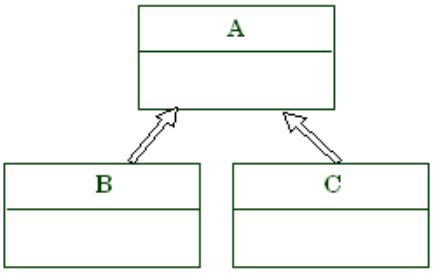
"polygone", "ellipse" et "cercle" sont des sous-entités de l'entité "forme".

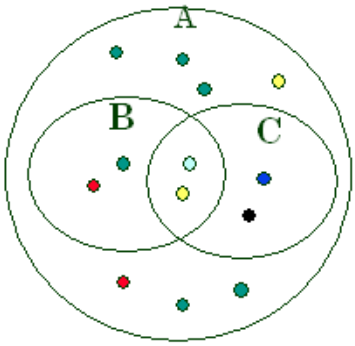
"chêne", "orme", et "bouleau" sont des sous-entités de l'entité "arbre".

"véhicule à moteur", "véhicule tout-terrain" et "véhicule amphibie" sont des sous-entités de l'entité "véhicule".

"camion" est une sous-entité de l'entité "véhicule à moteur".

Cas de plusieurs sous-entités

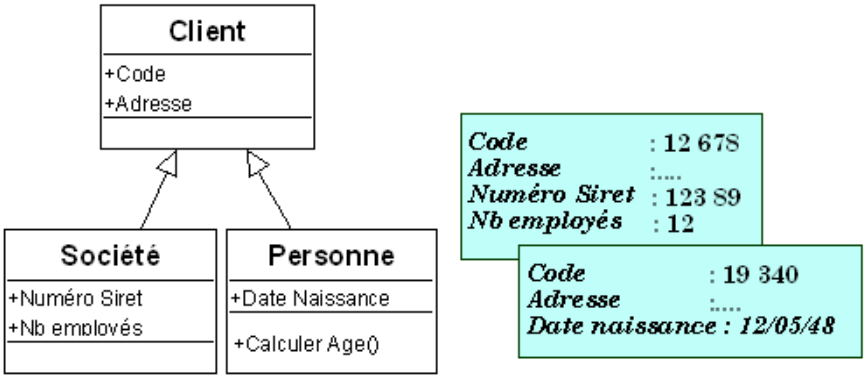




Plusieurs sous-entités d'une même entité :

- Ne sont pas forcément exclusives.
- Ne forment pas nécessairement une partition.

Intérêt des sous-entités

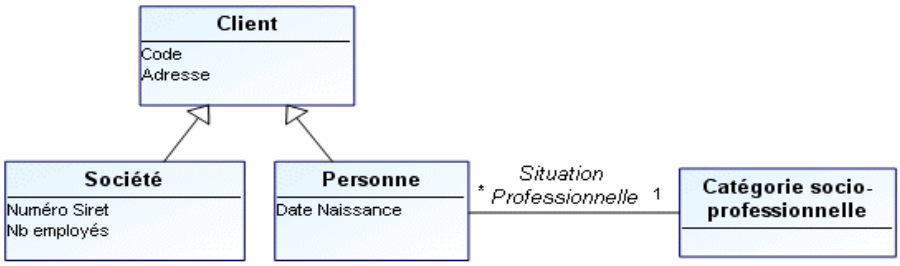


Une sous-entité hérite de tous les attributs et associations de sa super-entité, mais elle peut avoir des attributs ou des associations que ne possède pas sa super-entité.

Une sous-entité peut ainsi avoir des attributs spécifiques. Ceux-ci n'ont de sens que pour une sous-entité particulière. Dans l'exemple ci-dessus :

- Le "numéro de Siret" et le "nombre d'employés" n'ont de sens que pour une "société".
- La "date de naissance" est caractéristique d'une "personne", pas d'une "société".

Une sous-entité peut également avoir des associations spécifiques.



- Une "personne" entre dans une "catégorie socio-professionnelle" : "cadre", "employé", "commerçant", "agriculteur", etc. Cette classification n'a pas de sens pour une "société" (il existe également une classification pour les entreprises, mais ce n'est pas la même que pour les personnes).

Héritage multiple

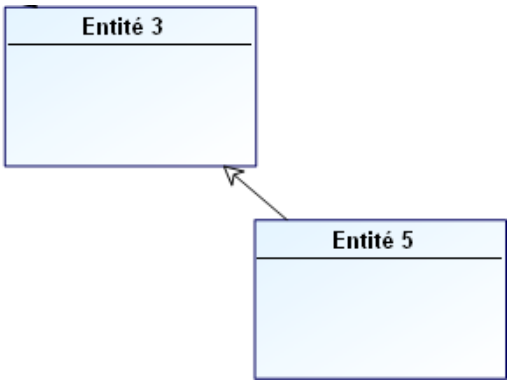
Il est parfois utile de spécifier une entité ayant plusieurs super-entités. La sous-entité hérite alors de toutes les caractéristiques des deux super-entités. Cette possibilité doit être utilisée avec précaution.

Créer une généralisation

Pour créer une *généralisation* :

1. Dans la barre d'insertion du diagramme de données, cliquez sur le bouton **Généralisation** 

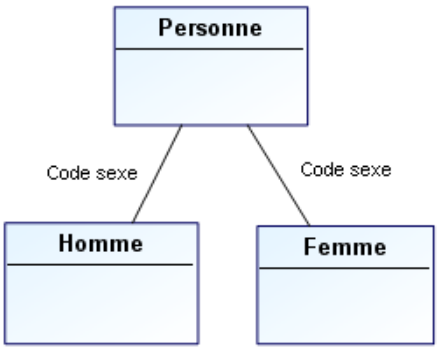
2. Cliquez dans la sous-entité, par exemple "Entité 5", et faites glisser la souris jusqu'à l'entité générale, par exemple "Entité 3", avant de relâcher votre pression.
   La généralisation apparaît dans le diagramme, matérialisée par une flèche.



Discriminant

Le discriminant est l'attribut de l'entité générale dont la valeur permet de répartir les objets entre les sous-entités associées à la généralisation.

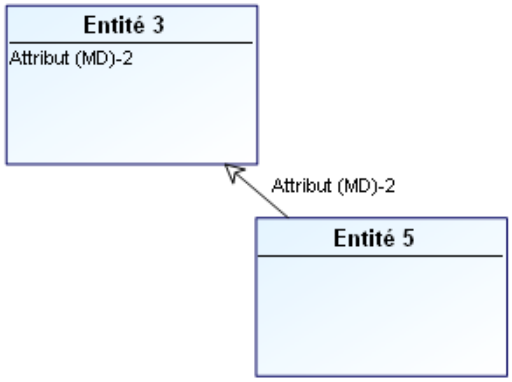
Par exemple, l'attribut code-sexe permet de répartir les objets de l'entité personne entre la sous-entité homme ou femme.



Pour définir un discriminant sur une généralisation :

1. Ouvrez les propriétés de la généralisation.
2. Cliquez la liste déroulante puis sur **Caractéristiques**.
3. Dans le champ **Discriminant**, cliquez sur la flèche puis sur **Relier Attribut (MD)**.

4. Recherchez et sélectionnez le discriminant parmi les attributs de la super-entité.
   Une fois sélectionné, le discriminant s'affiche sur la généralisation.



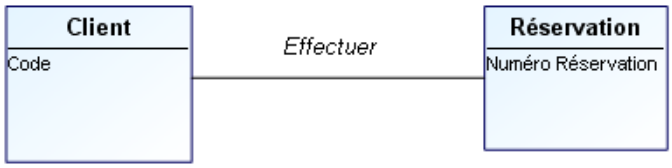
☛ *Il est également possible de préciser si une généralisation est* ***Complète*** *: dans ce cas toutes les instances de la super-entité appartiennent à au moins une des sous-entités de cette généralisation.*

LES IDENTIFIANTS

Définir l'identifiant d'une entité

Tout objet possède une identité qui caractérise son existence propre. L'*identifiant* permet de distinguer tout objet d'une entité de façon non ambiguë. Cela permet, entre autres, de distinguer deux objets dont toutes les valeurs d'attribut sont identiques.

> 🕮 *Un identifiant est constitué d'un ou plusieurs attributs ou rôles obligatoires qui permettent d'identifier de façon unique une entité.*



Le client dont le code est 2718 effectue la réservation numéro 314159.

Chaque entité possède un identifiant unique dont la valeur permet de retrouver sans ambiguïté chacun de ses exemplaires.

Par défaut, l'identifiant est implicite. Dans ce cas, une clé primaire sera générée automatiquement à partir du nom de l'entité.

Identification par un attribut

Il est possible de choisir un des attributs de l'entité comme identifiant. Pour cela :

1. Ouvrez les propriétés de l'entité.
2. Cliquez sur la liste déroulante puis **Attributs**.
   La liste des attributs apparaît.
3. Pour l'attribut choisi, sélectionnez la valeur "Oui" dans la colonne **Identifiant**.

> ☛ *Dans HOPEX Windows Front-End, une clé candidate constituée de cet attribut est créée automatiquement pour cette entité. La clé primaire correspondante sera créée dans la table générée pour l'entité lors de la synchronisation du modèle de données avec le modèle relationnel avec **HOPEX Database Builder**.*

CORRESPONDANCE DES MODÈLES DE DONNÉES

La modélisation de données reflète l'activité d'une entreprise et s'appuie sur l'historique du métier. Les différences constatées entre les modèles sont généralement d'ordre culturel ou liées à des conventions qui varient d'une personne à l'autre et dans le temps. Aussi, dans l'expression d'un besoin métier, le modélisateur doit tenir compte de l'existant et réussir à concilier différentes visions d'une même réalité.

La correspondance des modèles de données facilite l'alignement de ce patrimoine hétérogène sur un socle sémantique commun.

Objectifs fonctionnels

Distinguer les définitions d'entreprise et les données métiers

Pour assurer la cohérence des données métiers, les modélisateurs peuvent se référer à des définitions d'entreprise servant de cadre de référence.

La correspondance de modèles de données établit une frontière entre les définitions de niveau entreprise et les données métiers tout en assurant une traçabilité. L'outil Dictionnaire complète cette approche en permettant de constituer un vocabulaire métier sur la base d'un dictionnaire.

Voir le chapitre "Gérer un dictionnaire dans HOPEX" du guide HOPEX Common Features.

Intégrer les modèles existants

Les modèles existants qui décrivent un patrimoine applicatif doivent être pris en compte lors de l'élaboration de nouveaux modèles ou dans un projet de refonte. Les besoins varient selon les cas d'emplois :

- Démarche de type "As-is to-be" : l'évolution d'un modèle de données est progressive et s'appuie sur un état de référence stable, qui correspond généralement aux données du système en production.
- Mise en place progiciel : chaque progiciel (PGI, CRM, etc.) impose son modèle de données, ce qui accroît la tendance au morcellement et au cloisonnement du SI. D'où le besoin de disposer d'un modèle indépendant, lié aux différents modèles imposés.

La correspondance de modèles de données est un moyen de rapprocher les modèles de données de différentes sources.

Cas d'emploi

Un cas typique de mise en correspondance de données intervient dans le cadre d'échanges entre des applications disposant de leurs propres modèles de données. Lorsque le nombre d'applications devient trop important, vous pouvez mettre en

place un modèle pivot de référence qui va servir d'intermédiaire entre les applications et éviter ainsi la multiplication des correspondances.

Lancer l'éditeur de correspondances

L'éditeur de correspondances est l'outil qui permet d'aligner deux modèles de données ou de mettre en correspondance les vues logiques et physiques d'une base de données. Il se compose d'un arbre de correspondances qui juxtapose les vues des deux modèles.

Vous pouvez lancer l'éditeur de correspondances depuis :

- Le menu **Menu principal** d'HOPEX
- Un modèle de données
- Un paquetage de données
- Une base de données

Pour lancer l'éditeur de correspondance à partir du menu **Menu principal** :

1. Cliquez sur le menu **Menu principal** > **Éditeur de correspondances**.
   Une fenêtre apparaît.
2. Laissez l'option cochée par défaut **Créer un arbre de correspondances**, et cliquez sur **Suivant**.
3. Indiquez le nom du nouvel arbre de correspondance.
4. Dans le champ **Nature**, précisez le type de correspondance effectué.
5. Dans les cadres **Objet gauche** et **Objet droit**, à partir des types d'objet concernés, sélectionnez les modèles que vous souhaitez aligner.
6. Cliquez sur **OK**.
   L'éditeur affiche l'arbre de correspondance qui juxtapose les deux modèles.

Une fois l'arbre de correspondances créé, vous pouvez le retrouver ultérieurement dans l'éditeur de correspondances.

Créer une correspondance

Pour créer une correspondance entre deux objets :

1. Dans l'éditeur de correspondances, sélectionnez successivement les deux objets concernés.
2. Cliquez sur le bouton **Créer un élément de correspondance**.

   ☛ *En Windows Front-End, vous pouvez également créer la correspondance à partir du menu contextuel du dernier objet sélectionné, en utilisant la commande **Établir une correspondance**.*

La correspondance se crée à partir du dernier objet sélectionné.

Supprimer une correspondance

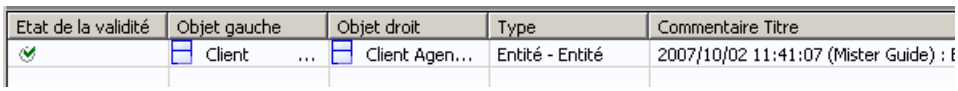
Pour supprimer une correspondance sur un objet :

- Cliquez sur l'objet en question et cliquez sur le bouton **Suppression d'un élément de correspondance**.

> ☛ *En Windows Front-End vous pouvez également supprimer une correspondance en utilisant la commande **Enlever la correspondance** du menu contextuel de l'objet sélectionné.*

Détails des correspondances

Les objets qui ont une correspondance sont cochés en vert. Lorsque vous sélectionnez l'un de ces objets dans l'arbre de navigation, sa correspondance apparaît dans la fenêtre de détails située par défaut en bas de l'éditeur de correspondances. Elle regroupe le nom des objets reliés, le type des objets et éventuellement un commentaire.



Propriétés d'une correspondance

Pour visualiser les propriétés d'une correspondance :

- Dans la fenêtre de détails de l'éditeur, sélectionnez la correspondance et cliquez sur le bouton **Propriétés**.

Etat des objets

Des indicateurs permettent d'indiquer l'état des objets mis en correspondance.

Les objets peuvent être caractérisés par les états suivants :

 Valide

 Invalide (Lorsqu'un objet a conservé une correspondance vers un objet qui n'existe plus)

 Sans correspondance

Source de la correspondance

En sélectionnant un objet dans l'arbre d'un des modèles présentés dans l'éditeur, vous pouvez retrouver sa correspondance dans l'autre modèle.

Pour afficher la correspondance d'un objet :

1. Sélectionnez l'objet en question.
   S'il a une correspondance, celle-ci s'affiche en bas de l'éditeur.
2. Sélectionnez la correspondance et cliquez sur le bouton **Chercher**
   Les deux objets en correspondances apparaissent en gras dans l'éditeur.

Exemple de correspondance entre modèles de données

Différents niveaux de modélisation peuvent couvrir des besoins distincts. Prenons l'exemple de deux modèles de données. Un modèle de données métier "Gestion des commandes (MD)" se situe au niveau conceptuel. Il décrit au niveau métier comment doivent être gérées les commandes.

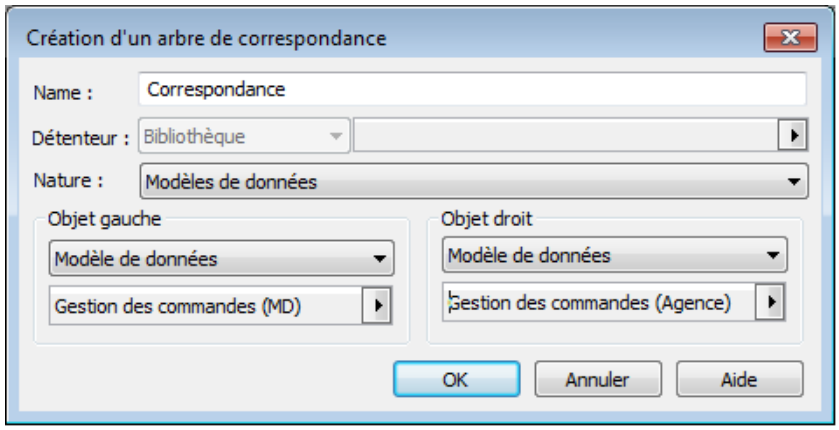
Au niveau logique, le modèle de données "Gestion des commandes (Agence)" présente une vision opérationnelle des données du système d'informations propre à chaque agence.

On retrouve des concepts identiques dans chacun des modèles. Il s'agit cependant d'objets distincts.

Vous pouvez mettre en correspondance ces deux modèles de données afin de favoriser la cohésion entre les besoins métiers et les systèmes qui les supportent.

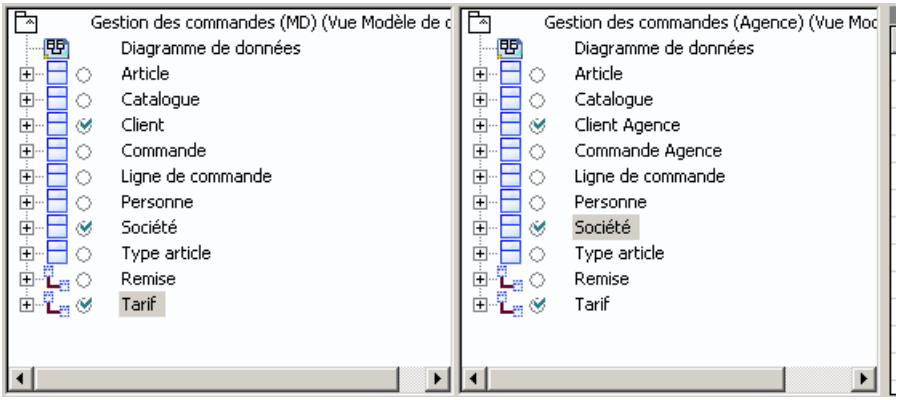
Pour cela :

1. Ouvrez l'**Editeur de correspondance**.
2. Créez un arbre de correspondance.
3. Sélectionnez les deux modèles à aligner.



4. Cliquez sur **OK**.
   L'éditeur affiche l'arbre de correspondance qui juxtapose les deux modèles.

5. Créez les correspondances entre les objets similaires puis enregistrez.



Une fois les modèles mis en correspondance, vous pouvez savoir quel objet logique est rattaché à un objet métier. Vous pouvez de même analyser l'impact des changements effectués au niveau métier sur le niveau opérationnel et inversement.

AUTRES NOTATIONS FOURNIES AVEC IA

Ce chapitre présente les autres notations disponibles avec **HOPEX Information Architecture.**

LA NOTATION IDEF1X

Condition préalable

Pour utiliser la notation IDEF1X, vous devez cocher l'option correspondante :

1. Dans le bureau d'**HOPEX Information Architecture**, cliquez sur le menu **Menu principal** > **Paramètres** > **Options.**
2. Dans l'arbre de navigation, dépliez le dossier **Modélisation des données**.
3. Cliquez sur **Notation des données**.
4. Dans la partie droite de la fenêtre cochez la notation IDEF1X.
5. Cliquez sur **OK**.

A propos de la modélisation des données avec IDEF1X

Modéliser les données consiste à identifier les objets de gestion (entités) et les associations ou relations entre ces objets, considérés d'intérêt pour représenter l'activité de l'entreprise.

IDEF1X est utilisé pour produire un modèle graphique qui représente la sémantique et la structure de l'information manipulée à l'intérieur d'une entreprise ou d'un système. L'utilisation de ce standard permet la construction d'un modèle sémantique qui peut servir de support à la gestion des données en tant que ressource, à l'intégration des systèmes d'information, ainsi qu'à la construction de bases de données informatisées.

Un des principaux objectifs est de servir de support à l'intégration des systèmes. L'approche de l'intégration avec IDEF1X est focalisée sur la saisie, la gestion et l'utilisation d'une seule définition sémantique des données appelée "schéma conceptuel". Le "schéma conceptuel" fournit une définition des données manipulées à l'intérieur de l'entreprise qui est unique, n'est pas biaisée par une seule application de ces données et qui est indépendante de la façon dont les données sont stockées physiquement ou de la façon dont on y accède. Le principal objectif de ce schéma conceptuel est de fournir une définition cohérente de la signification des données et de leurs interrelations qui puisse être utilisée pour intégrer ces données, les partager, et gérer leur intégrité. Un schéma conceptuel doit posséder trois caractéristiques essentielles :

- Il doit être cohérent avec l'infrastructure de l'entreprise et doit être valide dans tous ses domaines d'application.
- Il doit être extensible, de façon à ce que chaque nouvelle donnée puisse être définie sans avoir à modifier les données précédemment définies.
- Il doit être adaptable aussi bien aux différentes visions des utilisateurs qu'à une large variété de structures de stockage et d'accès aux données.

Les éléments de base d'un modèle IDEF1X sont :

- Les objets au sujet desquels les données sont conservées, c-à-d les personnes, lieux, idées, évènements, etc., qui sont représentés par des boîtes ;
- Les relations entre ces objets, représentées par des lignes qui relient ces boîtes ; et
- Les caractéristiques de ces objets représentées par les noms des attributs à l'intérieur des boîtes.

Synthèse des concepts

Dans **HOPEX Information Architecture**, un modèle de données (IDEF1X) est représenté par :

- Des entités, qui représentent les concepts de base (client, compte, produit, etc.).
- Des associations, qui définissent les associations entre les différentes entités.
- Des attributs qui décrivent les caractéristiques des entités.

L'attribut qui permet d'identifier de façon unique l'entité est appelé identifiant.

Le modèle de données est complété par la définition des multiplicités (ou cardinalités).

Créer un modèle de données (IDEF1X)

Pour créer un modèle de données :

1. Dans **HOPEX**, cliquez sur le volet de navigation **Données logiques**.
2. Dans le volet de navigation, cliquez sur **Tous les modèles de données**.
3. Dans la fenêtre d'édition, cliquez sur le bouton **Nouveau**.
   La fenêtre de création d'un modèle de données apparaît.
4. Saisissez le nom du modèle.
5. Cliquez sur **OK**.
   Le modèle de données apparaît dans la liste des modèles de données.

Diagramme de données (IDEF1X)

Un diagramme de données est une représentation graphique du modèle ou d'une partie du modèle.

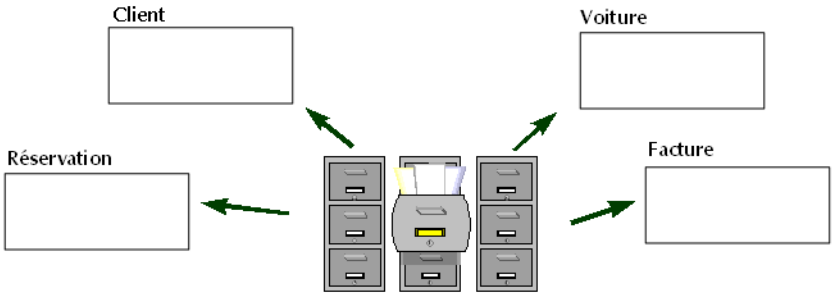
Pour créer un diagramme de données :

- Cliquez avec le bouton droit sur le modèle de données et sélectionnez **Nouveau > Diagramme de données (IDEF1X)**.
  Le diagramme de données s'ouvre.

Entités (IDEF1X)

Une entité représente un ensemble de choses réelles ou abstraites (personnes, objets, lieux, événements, idées, combinaisons d'objets, etc.) qui ont des attributs ou des caractéristiques en commun. Un membre individuel de l'ensemble est appelé une "instance de l'entité".

Nous pouvons illustrer la notion d'*entité* par une comparaison, par exemple, des fiches dans des tiroirs.



Une entité représente une classe particulière d'objets, dont tous les exemplaires peuvent être décrits de la même manière.

Une entité est "indépendante" si chaque instance de l'entité peut être définie de manière unique sans connaître ses relations avec d'autres entités. Une entité est "dépendante" si l' identification de manière unique d'une de ses instances dépend de ses relations avec d'autres entités.

Une entité est représentée par une boîte. Si l'entité est "dépendante", les angles de la boîte sont arrondis.

Créer une entité

Pour créer une entité :

1. Cliquez sur le bouton **Entité** de la barre d'objets du diagramme.
2. Cliquez sur le plan de travail du diagramme.
   La fenêtre **Ajout d'une entité** s'ouvre.
3. Saisissez le nom de l'entité.
4. Cliquez sur **Créer** (Windows Front-End) ou **Ajouter** (Web Front-End).
   L'entité apparaît dans le diagramme.

Attributs

Un attribut représente un type de caractéristique ou une propriété associée à un ensemble d'objets abstraits ou concrets. Une instance d'entité a généralement une valeur spécifique pour chacun des attributs. Une combinaison d'un ou de plusieurs attribut(s) peut être utilisée comme identifiant quand elle permet d'identifier de manière unique chacune des instances d'une entité.

Exemples d'*attribut* :

- Nom du client" (attribut de l'entité client).
- "N° client" (identifiant de l'entité client).
- "Solde du compte" (attribut de l'entité compte).

Définir les attributs

Pour créer un attribut :

1. Cliquez avec le bouton droit sur l'entité et sélectionnez **Propriétés**.
   La fenêtre des propriétés de l'entité s'ouvre.
2. Cliquez sur l'onglet **Attributs**.
3. Pour ajouter un nouvel attribut à l'entité, cliquez sur le bouton .
   Un nom vous est automatiquement proposé pour ce nouvel attribut. Vous pouvez le modifier.

Vous pouvez préciser son **Type de données**.

```
Exemple : Numérique.
```

Un type de données permet de mettre en commun des caractéristiques communes à plusieurs attributs. Les types de données sont implémentés sous forme de classes.

Voir "Types des attributs", page 103 pour plus de détails sur les types de données qui peuvent être affectés à un attribut.

Attributs hérités

Lorsqu'une relation de catégorisation (généralisation) existe entre une entité générale et une entité particulière, l'entité particulière hérite des attributs de l'entité générale.

Voir "Relations de catégorisation (généralisations) - (IDEF1X)", page 69.

Préciser l'identifiant d'une entité

Pour préciser l'identifiant d'une entité :

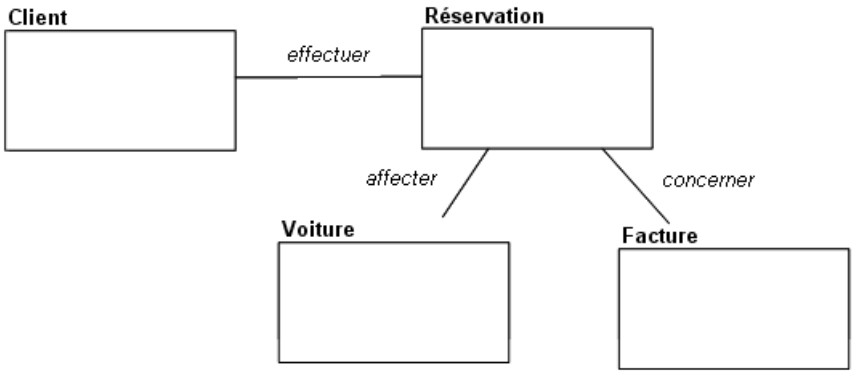
1. Ouvrez la fenêtre de propriétés de l'entité.
2. Cliquez sur l'onglet **Attributs**.
3. Pour l'attribut choisi sélectionnez la valeur "Oui" dans la colonne **Identifiant**.

Pour plus de détails, voir "Définir l'identifiant d'une entité", page 51.

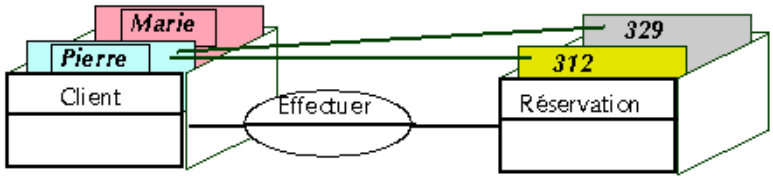
Associations (IDEF1X)

Une association est une relation entre deux ou plusieurs entités. Elle peut comporter des attributs qui caractérisent l'association entre ces entités.

Les *associations* peuvent être comparées à des liens entre des fiches.



Le dessin suivant permet de visualiser "en trois dimensions" les situations qu'un diagramme de données permet de mémoriser.



Pierre et Marie sont des clients. Pierre a effectué les réservations numéros 312 et 329.

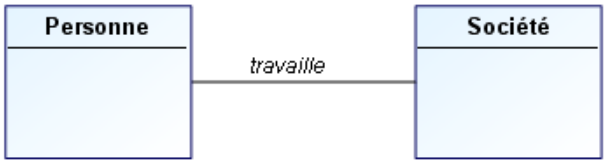
Un diagramme de données doit permettre de mémoriser toutes les situations du contexte de l'entreprise, mais rien que celles-là.

> ☛ *Le diagramme ne doit pas permettre de représenter des situations irréalistes ou aberrantes.*

Exemples d'association :

- Un client passe une commande.
- Une commande comprend plusieurs produits.

- Une personne travaille pour une société.



- Une alarme est déclenchée par un capteur.
- Un capteur couvre une zone.
- Une fenêtre affiche une chaîne de caractères.

Relation identifiante obligatoire

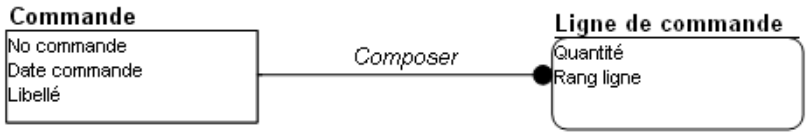
Une relation identifiante obligatoire est une association entre des entités dans laquelle chaque instance de la première entité est associée à zéro, une ou plusieurs instances de la seconde entité et chaque instance de la seconde entité est associée à une instance de la première entité et est identifiée par cette association. La seconde entité est toujours une entité dépendante représentée par une boîte aux angles arrondis. Une relation identifiante est représentée par une ligne pleine avec un point du côté de l'entité dépendante.

Si une instance d'entité est identifiée par son association avec une autre entité, la relation est dite "identifiante", et chaque instance de cette entité doit être associée avec exactement une instance de l'autre entité. Par exemple, si une ou plusieurs tâches sont associées à chaque projet et que les tâches sont identifiées de manière unique à l'intérieur d'un projet, une relation identifiante existe entre les entités "Projet" et "Tâche". C'est-à-dire que le projet associé doit être connu pour pouvoir identifier de manière unique une tâche des autres. L'existence d'un fils dans une relation identifiante est toujours dépendante de celle de son parent, c-à-d qu'une instance de l'entité fils ne peut exister que si elle est reliée à une instance de l'entité parente.

Pour créer une *relation identifiante* :

1. Dans la barre d'objets du diagramme, cliquez sur le bouton **Relation identifiante obligatoire**  .
2. Cliquez sur l'entité parente, et en gardant le bouton de la souris enfoncé, déplacez le pointeur jusqu'à l'entité fille, avant de relâcher votre pression.

L’association apparaît dans le diagramme. Elle est représentée par un trait plein avec un point à l'extrémité du trait du côté de l'entité dépendante. La forme de l'entité dépendante est automatiquement changée en une boîte aux angles arrondis.



Relation obligatoire identifiante

Dans l'exemple ci-dessus, une commande est composée de lignes de commandes, et chaque ligne de commande est identifiée par son association avec la commande. La ligne de commande est une entité dépendante représentée par une boîte aux angles arrondis.

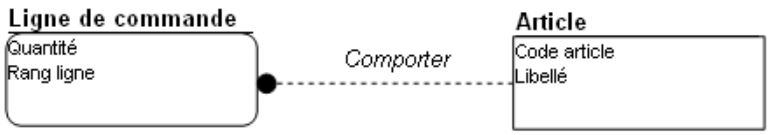
Relation non-identifiante obligatoire

Une relation non-identifiante obligatoire est une association entre des entités dans laquelle chaque instance de la première entité est associée à zéro, une ou plusieurs instances de la seconde entité et chaque instance de la seconde entité est associée à une instance de la première entité mais n’est pas identifiée par cette association. Elle est représentée par une ligne pointillée avec un point du côté de l’entité dépendante.

Si chaque instance d'une entité peut être identifiée de manière unique sans connaître l'instance de l'autre entité associée, la relation est "non-identifiante". Par exemple, bien qu'il puisse exister une relation de dépendance entre les entités "Acheteur" et "Ordre d'achat", les ordres d'achat peuvent être identifiés de manière unique par un numéro d'ordre d'achat sans nécessairement connaître l'acheteur concerné.

Pour créer une *relation non-identifiante* :

1. Dans la barre d’objets du diagramme, cliquez sur le bouton **Relation non-identifiante obligatoire** .
2. Cliquez dans l'entité parente, et en gardant le bouton de la souris enfoncé, déplacez le pointeur jusqu’à l'entité fille, avant de relâcher votre pression.
   L’association apparaît dans le diagramme.



Relation obligatoire non-identifiante

Dans l'exemple ci-dessus, une ligne de commande inclut un article, mais elle n'est pas identifiée par son association avec cet article.

Relation optionnelle non-identifiante

> 🕮 *Une relation optionnelle est une association entre entités dans laquelle chaque instance de la première entité est associée à zéro, une ou plusieurs instances de la seconde entité et chaque instance de la seconde entité est associée à zéro, une ou plusieurs instances de la première entité. Elle est représentée par une ligne pointillée avec un point du côté de la seconde entité et un petit losange à l'autre extrémité.*

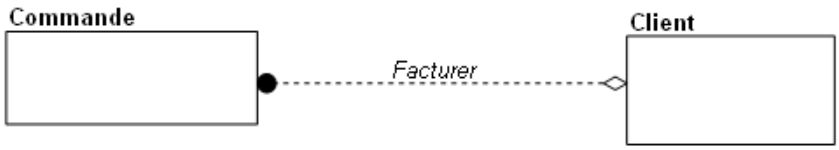
Dans une *relation optionnelle* non-identifiante, chaque instance de l'entité fille est reliée à zéro ou une instance de l'entité parente.

Une relation optionnelle non-identifiante représente une dépendance conditionnelle. Une relation optionnelle non-identifiante est représentée par un trait pointillé entre l'entité parente et l'entité fille, avec un petit losange du côté de l'entité parente.

Une instance de l'entité fille pour laquelle chaque attribut de la clé étrangère qui correspond à la relation possède une valeur, doit être associée à une instance de l'association parente dans laquelle les valeurs des attributs de la clé primaire de l'entité parente doit être égale à la valeur des attributs de la clé étrangère de l'entité fille.

Pour créer une relation optionnelle non-identifiante :

1. Dans la barre d'objets du diagramme, cliquez sur le bouton **Relation optionnelle**  .
2. Cliquez sur l'entité parente, et en gardant le bouton de la souris enfoncé, déplacez le pointeur jusqu'à l'entité fille, avant de relâcher votre pression.
   L'association apparaît dans le diagramme.



Relation optionnelle

Dans l'exemple ci-dessus, une commande devrait être facturée à un client, mais ce n'est pas obligatoire (problèmes de livraison, etc.).

Relation non-spécifique

> 🕮 *Une relation non-spécifique est une association entre entités dans laquelle chaque instance de la première entité est associée à zéro, une ou plusieurs instances de la seconde entité et chaque instance de la seconde entité est associée à zéro, une ou plusieurs instances de la première entité. Elle est représentée par une ligne tirée entre les deux entités avec un point à chaque extrémité.*

Des *relations non-spécifiques* sont utilisées dans des vues Entité-Relation de haut niveau pour représenter des relations plusieurs-à-plusieurs entre des entités.

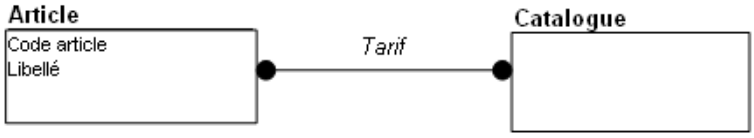
Dans le développement initial d'un modèle, il est souvent pratique d'identifier des "relations non-spécifiques" entre les entités. Ces relations non-spécifiques sont raffinées dans les phases de développement ultérieures du modèle.

Une relation non-spécifique, également appelée "relation plusieurs-à-plusieurs", est une association entre deux entités dans laquelle chaque instance de la première entité est associée avec zéro, une ou plusieurs instances de la deuxième entité et chaque instance de la deuxième entité est associée avec zéro, une ou plusieurs instances de la première entité. Par exemple, si un employé peut être affecté à plusieurs projets et si plusieurs employés peuvent être affectés au même projet, le lien entre les entités "Employé" et "Projet" peut être exprimé par une relation non-spécifique. Cette relation non-spécifique pourra être remplacée par des relations spécifiques plus tard dans le développement du modèle en introduisant une troisième entité, telle que "Affectation Projet", qui est une entité fille commune dans les relations spécifiques avec les entités "Employé" et "Projet". Ces nouvelles relations spécifient que chaque employé est affecté à zéro, un, ou plusieurs projets. Chaque assignement de projet est pour exactement un employé et un projet. Les entités introduites pour résoudre les relations non-spécifiques sont parfois appelées "entités associatives" ou "entités d'intersection".

La définition d'une relation non-spécifique peut ensuite être complétée en spécifiant les cardinalités pour chaque sens de la relation.

Pour créer une relation non-spécifique :

1. Dans la barre d'objets du diagramme, cliquez sur le bouton **Relation non-spécifique**  .
2. Cliquez sur la première entité, et en gardant le bouton de la souris enfoncé, déplacez le pointeur jusqu'à la deuxième entité, avant de relâcher votre pression.
   L'association apparaît dans le diagramme.



Relation non-spécifique

Dans l'exemple ci-dessus, un article peut apparaître dans zéro, un ou plusieurs catalogues et un catalogue peut contenir zéro, un ou plusieurs articles.

Entité associative

📖 *Une entité associative est une entité qui est introduite pour résoudre une relation non-spécifique ou pour afficher des attributs en tant que propriétés d'une association.*

Des relations non-spécifiques sont utilisées dans des vues Entité-Relation de haut niveau pour représenter des relations plusieurs-à-plusieurs entre des entités. Dans une vue basée sur les clés ou définie complètement avec ses attributs, toutes les associations entre les entités doivent être exprimées sous forme de relations spécifiques. Cependant, dans le développement initial d'un modèle, il a été souvent

pratique d'identifier des "relations non-spécifiques" entre les entités. Ces relations non-spécifiques sont raffinées dans les phases de développement ultérieures du modèle.

Les entités introduites pour résoudre les relations non-spécifiques sont parfois appelées "entités associatives" ou "entités d'intersection".

Pour créer une *entité associative* :

1. Dans la barre d'objets du diagramme, cliquez sur le bouton **Entité** .
2. Cliquez sur le plan de travail du diagramme.
   La fenêtre **Ajout d'une entité** s'ouvre.
3. Saisissez le Nom de l'entité associative.
4. Cliquez sur **Créer** (Windows Front-End) ou **Ajouter** (Web Front-End).
   L'entité apparaît dans le diagramme.
5. Cliquez sur le bouton **Relation identifiante obligatoire** .
6. Cliquez sur la première entité, et en gardant le bouton de la souris enfoncé, déplacez le pointeur jusqu'à l'entité associative, avant de relâcher votre pression.
   L'association apparaît dans le diagramme. La forme de l'entité associative est changée en une boîte aux angles arrondis pour indiquer qu'il s'agit d'une entité dépendante.
7. Créez de la même manière la deuxième association en cliquant sur la deuxième entité et, en gardant le bouton de la souris enfoncé, déplacez le pointeur jusqu'à l'entité associative, avant de relâcher votre pression.

☛ *Vous pouvez ajouter des attributs à une entité associative.*

Entité associative

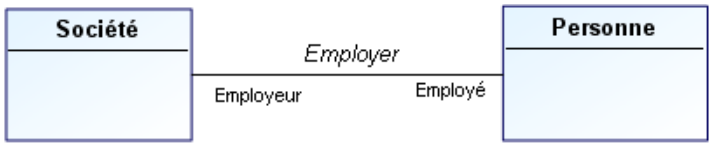
Dans l'exemple ci-dessus, un article peut faire l'objet d'une remise pour zéro, un ou plusieurs clients, et un client peut bénéficier de remises sur zéro, un ou plusieurs articles. Dans chaque cas, le taux de remise est indiqué sur l'entité associative.

Définir les rôles des associations

📖 *Un rôle permet d'indiquer une des entités concernées par l'association. L'indication des rôles est particulièrement importante dans le cas d'une association entre une entité et elle-même.*

Chaque extrémité d'une association permet de préciser le *rôle* joué par chaque entité dans l'association.

Visuellement, le nom du rôle se distingue du nom d'une association, car il est placé près de son extrémité. De plus, il apparaît en caractères droits, alors que le nom de l'association est en italique.



☺ *La barre d'état (située au bas de la fenêtre) permet aussi de distinguer les différentes zones : lorsque vous déplacez la souris le long de l'association, elle indique si vous vous trouvez sur l'association ou sur un rôle.*

Lorsque deux entités sont reliées par une seule association, le nom des entités suffit souvent à caractériser le rôle ; nommer les rôles prend tout son intérêt lorsque plusieurs associations relient deux entités.

Certaines associations peuvent mettre en œuvre plus de deux entités. Ces associations sont généralement rares.

Pour ajouter un rôle à une association :

1. Cliquez sur le bouton **Rôle de l'association**  et reliez l'association à l'entité.

Multiplicités

Chaque rôle d'une association porte une indication de multiplicité qui montre combien d'objets de l'entité considérée peuvent être liés à un objet de l'autre entité. La multiplicité est une information portée par le rôle, sous la forme d'une expression entière bornée. On l'indique en particulier pour chacun des rôles que jouent les entités dans une association.

La multiplicité exprime le nombre de participations minimum et maximum d'un objet donné d'une entité à une association.

Les multiplicités usuelles sont "1", "0..1", "*" ou "0..*", "1..*", et "M..N" où "M" et "N" sont des entiers :

- La multiplicité "1" indique que chaque objet de l'entité est relié par cette association une fois et une seule.
  Elle est représentée par une relation obligatoire avec un point sur ce rôle et pas de point sur le rôle opposé.
- La multiplicité "0..1" indique qu'un objet de l'entité ne peut être relié par cette association qu'une fois au plus.
  Elle est représentée par un "Z" à côté du rôle.
- La multiplicité "*" ou "0..*" indique qu'un objet de l'entité peut être relié par l'association une ou plusieurs fois ou pas du tout.
  C'est la valeur par défaut.
- La multiplicité "1..*" indique que chaque objet de l'entité est obligatoirement relié par l'association et qu'il peut l'être plusieurs fois.
  Elle est représentée par un "P" (pour Positif) à côté du rôle.
- La multiplicité "M..N" indique que chaque objet de l'entité est obligatoirement relié par l'association au moins "M" fois et qu'il peut l'être au maximum "N" fois.

1	Un et un seul
0..1	Zéro ou un (Z)
M..N	De M à N (entiers naturels)
*	De zéro à plusieurs
0..*	De zéro à plusieurs
1..*	De un à plusieurs (P)

Pour préciser la multiplicité d'un rôle :

1. Cliquez avec le bouton droit sur le trait qui se trouve entre l'association et l'entité, afin d'ouvrir le menu contextuel du rôle.
2. Cliquez sur **Propriétés**.
   La page de propriétés du rôle s'ouvre.
3. Cliquez sur l'onglet **Caractéristiques**.
4. Dans le champ **Multiplicité**, sélectionnez la multiplicité voulue.

La représentation de l'association change en fonction de la nouvelle valeur de ses multiplicités.

☛ *Dans HOPEX Windows Front-End, la multiplicité est également affichée dans le menu contextuel du rôle. Si le menu affiché ne propose pas les multiplicités, vérifiez que vous avez bien cliqué sur le trait qui matérialise le rôle, et non sur l'association.*

Relations de catégorisation (généralisations) - (IDEF1X)

📖 *Une généralisation représente une relation d'héritage entre une entité générale et une entité plus spécifique. L'entité spécifique est cohérente avec l'entité plus générale et hérite de ses caractéristiques et*

de son comportement. Elle peut cependant comporter des attributs ou des associations supplémentaires. Tout objet de l'entité spécifique est aussi un objet de l'entité générale.

Qu'est-ce qu'une Catégorisation (Généralisation) ?

Les relations de catégorisation sont utilisées pour représenter des structures dans lesquelles une entité est un "type" (catégorie) d'une autre entité.

Les entités sont utilisées pour représenter des "objets au sujet desquels nous avons besoin d'information". Comme certains objets du monde réel sont des catégories d'autres objets, certaines entités doivent, dans un certain sens, être des catégories d'autres entités. Par exemple, supposons que nous avons besoin d'informations à propos des employés.

Bien que nous disposions d'informations à propos des employés en général, nous pouvons avoir besoin d'informations complémentaires à propos des employés salariés qui soient différentes de celles à propos des employés payés à l'heure. Dans ce cas, les entités "Employé salarié" et "Employé payé à l'heure" sont des catégories de l'entité "Employé". Dans la notation IDEF1X, elles sont reliées par des relations de catégorisation (*généralisation*).

Dans un autre cas, une catégorie d'entité peut être nécessaire pour exprimer une relation qui n'est valide que pour une catégorie particulière, ou pour documenter les différences entre les relations autour des diverses catégories de l'entité. Par exemple, un "Employé à plein temps" peut profiter de la participation aux bénéfices, tandis qu'un "Employé à temps partiel" ne peut pas.

Une "relation de catégorisation" ou "généralisation" est une relation entre une entité, appelée "entité générale" et une autre entité, appelée "catégorie" ou "entité spécialisée". La cardinalité n'est pas spécifiée pour les catégories, car elle est toujours zéro ou un.

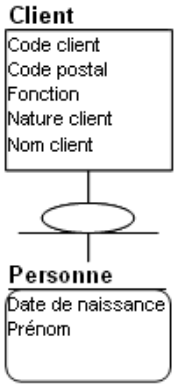
Les catégories sont aussi toujours des entités dépendantes.

Créer une catégorisation

Pour créer une relation de catégorisation :

1. Cliquez sur le bouton **Généralisation**  de la barre d'objets.

2. Cliquez dans l'entité catégorie, et faites glisser la souris jusqu'à l'entité générale, avant de relâcher votre pression.
   La généralisation est représentée dans le diagramme par un cercle souligné, relié par un trait à l'entité générale et par un autre à l'entité spécialisée.



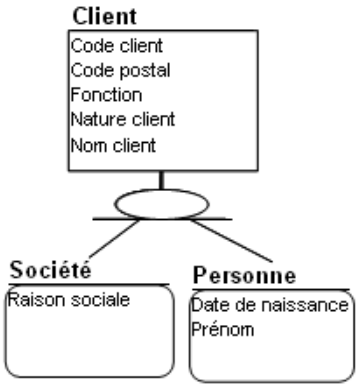
Relation de catégorisation

Dans l'exemple ci-dessus, certains attributs ont un intérêt pour les personnes et n'ont pas de sens pour d'autres catégories de clients. Personne est une entité dépendante représentée par une boîte aux angles arrondis.

Catégories multiples

Un "groupe de catégories" est un ensemble constitué d'une ou de plusieurs relations de catégorisation. Une instance de l'entité générale peut être associée à une instance d'une seule des catégories du groupe, et chaque instance d'une catégorie est associée avec exactement une instance de l'entité générale. Chaque instance de la catégorie représente le même objet du monde réel que l'instance associée de l'entité générale. Dans l'exemple ci-dessus, "Employé" est l'entité générale et "Employé salarié" et "Employé payé à l'heure" sont les catégories. Il y a deux

relations de catégorisation dans ce groupe, une entre "Employé" et "Employé salarié" et une entre "Employé" et "Employé payé à l'heure".



Catégories multiples

Dans l'exemple ci-dessus, les sociétés et les personnes représentent deux catégories de clients.

Multiples groupes de catégories

Comme une instance de l'entité générale ne peut pas être associée à plus d'une instance des catégories du groupe, ces catégories sont mutuellement exclusives. Dans l'exemple précédent, cela implique qu'un employé ne peut pas être à la fois "salarié" et "payé à l'heure". Cependant, une même entité peut être l'entité générale de plus d'un groupe de catégories, et les catégories d'un groupe ne sont pas exclusives des catégories d'autres groupes. Par exemple, "Employé" peut être l'entité générale dans un deuxième groupe de catégories comprenant les catégories "Employé masculin" et "Employé féminin". Une instance d'employé peut être une instance d'"employé salarié" ou d'employé payé à l'heure" et en même temps d'"employé masculin" ou d'"employé féminin".

Catégorisation complète

Dans un groupe de catégorie "complet", chaque instance de l'entité générale est une instance d'une catégorie du groupe, c'est-à-dire que toutes les catégories possibles sont présentes. Par exemple, chaque employé est soit masculin, soit féminin, ainsi le deuxième groupe de catégories est complet. Dans un groupe de catégorie "incomplet", une instance de l'entité générale peut exister sans être associée à une instance d'une des catégories, c'est-à-dire que certaines catégories sont omises. Par exemple, si certains employés sont payés à la commission plutôt que payés à l'heure ou salariés, le premier groupe de catégorie serait incomplet.

Il est possible de spécifier si une relation de catégorisation est complète ou non dans l'onglet **Caractéristiques** de la fenêtre des propriétés d'une généralisation. Si la valeur des la caractéristique **Complète** est "Oui", toutes les instances de l'entité générale appartiennent à au moins une des catégories de la généralisation.

Discriminant

Un attribut de l'entité générale, ou de l'un de ses ancêtres, peut être désigné comme discriminant pour un groupe de catégorie particulier. La valeur prise par le discriminant détermine la catégorie à laquelle appartient l'instance d'entité. Dans l'exemple précédent, le discriminant pour le groupe incluant les catégories "salariés" et "payés à l'heure" pourrait être "Type d'employé". Si un groupe a un discriminant, il doit être différent de tous les autres discriminants.

Pour définir un identifiant sur une généralisation :

1. Ouvrez les propriétés de la généralisation.
2. Cliquez sur **Caractéristiques**.
3. Dans le champ **Discriminant**, sélectionnez le discriminant parmi les attributs de la super-entité.
   Une fois sélectionné, le discriminant s'affiche sur la généralisation.

LA NOTATION I.E.

Condition préalable

Pour utiliser la notation I.E, vous devez cocher l'option correspondante :

1. Dans le bureau d'**HOPEX Information Architecture**, cliquez sur le menu **Menu principal** > **Paramètres** > **Options.**
2. Dans l'arbre de navigation, dépliez le dossier **Modélisation des données**.
3. Cliquez sur **Notation des données**.
4. Dans la partie droite de la fenêtre cochez la notation I.E.
5. Cliquez sur **OK**.

A propos de la modélisation des données avec I.E.

"Information Engineering" fut à l'origine développé par Clive Finkelstein en Australie à la fin des années 1970. Il a ensuite collaboré avec James Martin pour diffuser sa méthode aux Etats-Unis et en Europe.

Information Engineering est un ensemble intégré et évolutif de tâches et de techniques utilisées pour la planification de l'entreprise, la modélisation des données, la modélisation des processus, la conception des systèmes et leur mise en oeuvre. Il permet à une entreprise de maximiser les ressources financières, humaines ou informationnelles disponibles pour supporter la réalisation de ses objectifs métiers.

Tirée par les métiers, Information Engineering est une des méthodes de développement de systèmes dominantes dans le monde, à une époque où les entreprises tentent de se positionner dans la compétition exacerbée qui règne depuis les années 1990.

Elle se focalise sur les données avant les processus, ce qui assure que les entreprises identifient ce qui est requis par leur métier avant d'analyser la façon de le produire. IE fournit un ensemble important de techniques pour l'analyse stratégique que l'on ne retrouve pas dans les méthodologies basées sur les processus.

Information Engineering guide l'entreprise à travers une série d'étapes prédéfinies qui permettent d'identifier l'information importante pour l'entreprise et établit les relations entre les éléments d'information. Ainsi, les besoins en information sont clairement définis à partir des données de gestion et peuvent être traduits directement dans les systèmes qui supportent les plans stratégiques.

La plupart des systèmes d'information développés durant les vingt-cinq dernières années ont été réalisés depuis la vue spécifique de chaque application, ce qui a beaucoup contribué à l'effet tunnel. Le résultat est que beaucoup d'entreprises ont

des systèmes séparés qui sont incapables de partager des données. Dans cette situation, les systèmes ne peuvent pas atteindre leur potentiel initial, et peuvent en fait devenir des fardeaux pour l'entreprise. IE identifie clairement les besoins en partage des données à travers l'entreprise de façon à ce que les systèmes puissent être intégrés en conséquence.

En utilisant IE, les entreprises disposent d'un cadre stable mais flexible sur lequel baser les activités de développement. Cela élimine les redondances et permet la réutilisation de modules de programmation et le partage des données requises par l'entreprise, ce qui aide à alléger le poids de la maintenance.

Modéliser les données consiste à identifier les objets de gestion (entités) et les associations ou relations entre ces objets, considérés d'intérêt pour représenter l'activité de l'entreprise.

I.E. est utilisé pour produire un modèle graphique qui représente la sémantique et la structure de l'information manipulée à l'intérieur d'une entreprise ou d'un système. L'utilisation de ce standard permet la construction d'un modèle sémantique qui peut servir de support à la gestion des données en tant que ressource, à l'intégration des systèmes d'information , ainsi qu'à la construction de bases de données informatisées.

Les éléments de base d'un modèle de données Information Engineering sont :

- Les objets au sujet desquels les données sont conservées, c-à-d les personnes, lieux, idées, évènements, etc., qui sont représentés par des boîtes ;
- Les relations entre ces objets, représentées par des lignes qui relient ces boîtes ; et
- Les caractéristiques de ces objets représentées par les noms des attributs à l'intérieur des boîtes.

Synthèse des concepts

Dans **HOPEX Logical Data**, un modèle de données (I.E.) est représenté par :

- Des entités, qui représentent les concepts de base (client, compte, produit, etc.).
- Des associations, qui définissent les associations entre les différentes entités.
- Des attributs qui décrivent les caractéristiques des entités.

L'attribut qui permet d'identifier de façon unique l'entité est appelé identifiant.

Le modèle de données est complété par la définition des multiplicités (ou cardinalités).

Créer un modèle de données (I.E)

Un modèle de données I.E. présente les entités comme des boîtes rectangulaires (une entité est une personne ou une chose à propos de laquelle des données sont stockées). Les entités sont reliées les unes aux autres, par exemple, une entité "Produit" est achetée par une entité "Client". Des traits reliant les boîtes montrent ces associations. Des indicateurs de cardinalité (multiplicité) sont affichés sur ces traits.

Pour créer un modèle de données :

1. Dans **HOPEX**, cliquez sur le volet de navigation **Données logiques**.
2. Dans le volet de navigation, cliquez sur **Tous les modèles de données**.
3. Dans la fenêtre d'édition, cliquez sur le bouton **Nouveau**.
   La fenêtre de création d'un modèle de données apparaît.
4. Saisissez le nom du modèle.
5. Cliquez sur **OK**.
   Le modèle de données apparaît dans la liste des modèles de données.

Diagramme de données (I.E)

Un diagramme de données est une représentation graphique du modèle ou d'une partie du modèle. La création d'un diagramme varie légèrement suivant que vous êtes en Windows Front-End ou Web Front-End.

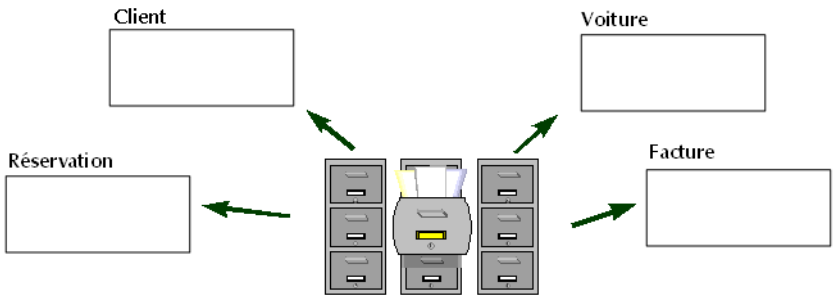
Pour créer un diagramme de données :

- Cliquez avec le bouton droit sur le modèle de données et sélectionnez **Nouveau > Diagramme de données (IE)**.
  Le diagramme de données s'ouvre.

Entités (I.E)

📖 *Une entité représente une personne, un lieu, une chose ou un concept qui possède des caractéristiques d'intérêt pour l'entreprise. Une entité possède divers attributs qui peuvent être stockés dans le système d'information. Ex: Client, Employé, Commande, Facture, etc.*

Nous pouvons illustrer la notion d'*entité* par une comparaison, par exemple, des fiches dans des tiroirs.



Une entité représente une classe particulière d'objets, dont tous les exemplaires peuvent être décrits de la même manière. Une entité est représentée par une boîte rectangulaire.

Créer une entité

Pour créer une entité :

1. Sélectionnez le bouton **Entité** dans la barre d'objets en cliquant dessus avec le bouton gauche de la souris.
2. Cliquez sur le plan de travail du diagramme.
   La fenêtre **Ajout d'une entité** s'ouvre.
3. Saisissez le nom de l'entité.
4. Cliquez sur **Créer** (Windows Front-End) ou **Ajouter** (Web Front-End).
   L'entité apparaît dans le diagramme.

Attributs

Exemples d'*attributs* :

- Nom du client" (attribut de l'entité client).
- "N° client" (identifiant de l'entité client).
- "Solde du compte" (attribut de l'entité compte).

> *Un attribut représente un type de caractéristique ou une propriété associée à un ensemble d'objets abstraits ou concrets. Une instance d'entité a généralement une valeur spécifique pour chacun des attributs. Une combinaison d'un ou de plusieurs attribut(s) peut être utilisée comme identifiant quand elle permet d'identifier de manière unique chacune des instances d'une entité.*

Définir les attributs

Pour créer un attribut :

1. Cliquez avec le bouton droit sur l'entité et sélectionnez **Propriétés**.
   La fenêtre des propriétés de l'entité s'ouvre.
2. Cliquez sur l'onglet **Attributs**.

3. Pour ajouter un nouvel attribut à l'entité, cliquez sur le bouton .
   Un nom vous est automatiquement proposé pour ce nouvel attribut. Vous pouvez le modifier.

Vous pouvez préciser son **Type de données**.

```
Exemple : Numérique.
```

Un type de données permet de mettre en commun des caractéristiques communes à plusieurs attributs. Les types de données sont implémentés sous forme de classes.

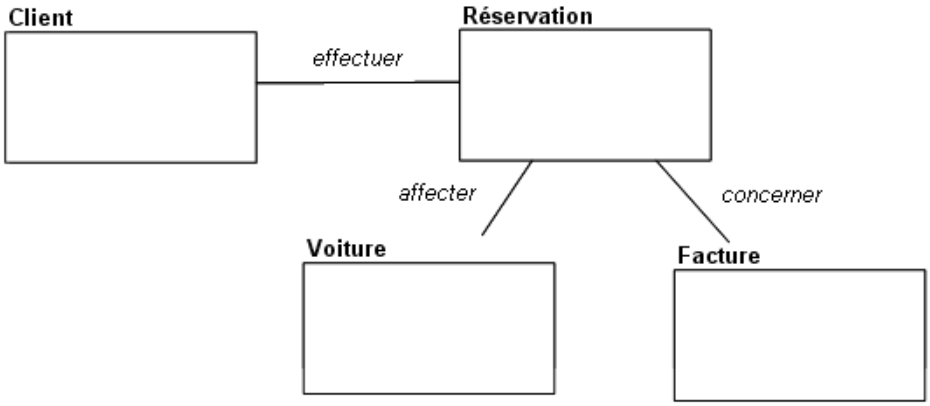
☛ *Voir "Types des attributs", page 103 pour plus de détails sur les types de données qui peuvent être affectés à un attribut.*

Associations (I.E)

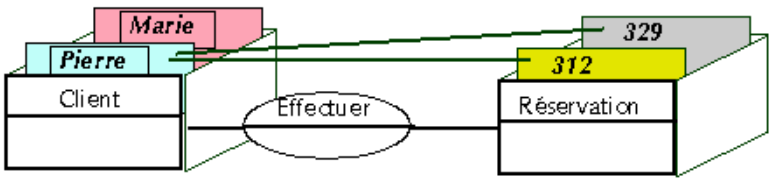
Une association représente un lien significatif entre deux objets. Les associations sont utilisées pour saisir des données au sujet de la relation qui existe entre les deux objets.

Présentation

Les *associations* peuvent être comparées à des liens entre des fiches.



Le dessin suivant permet de visualiser "en trois dimensions" les situations qu'un diagramme de données permet de mémoriser.



Pierre et Marie sont des clients. Pierre a effectué les réservations numéros 312 et 329.

Associations et multiplicités

Chaque rôle d'une association porte une indication de multiplicité qui montre combien d'objets de l'entité considérée peuvent être liés à un objet de l'autre entité. La multiplicité est une information portée par le rôle, sous la forme d'une expression entière bornée. On l'indique en particulier pour chacun des rôles que jouent les entités dans une association.

Pour indiquer qu'un rôle est optionnel, un cercle "O" est placé à l'autre extrémité de du trait, ce qui représente une multiplicité minimum de 0.

Pour indiquer qu'un rôle est obligatoire, un trait "|" est placé à l'autre extrémité de du trait, ce qui représente une multiplicité minimum de 1.

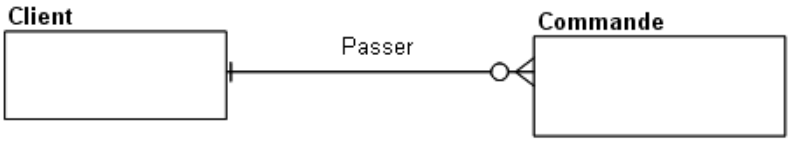
Une patte d'oie est utilisée pour représenter une multiplicité maximum égale à plusieurs.

Combinée avec une multiplicité de 0 ou 1, un trait "|" est souvent utilisé pour représenter une multiplicité maximum de 1.

De cette façon, la combinaison "O|" signifie "au plus un" et la combinaison "| |" ou simplement "|" signifie exactement 1.

Relation obligatoire

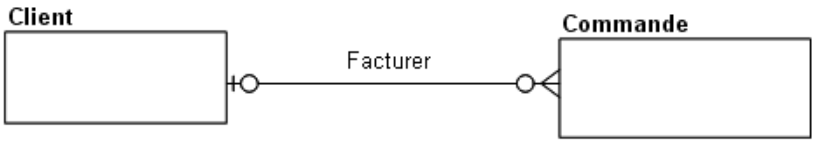
 Une relation obligatoire signifie que chaque instance de la première entité est associée avec exactement une instance de la deuxième entité et que la deuxième entité peut être associée à zéro, une ou plusieurs instances de la première entité.



Dans l'exemple ci-dessus, un client peut passer zéro, une ou plusieurs commandes, mais une commande est toujours passée par un et un seul client.

Relation optionnelle

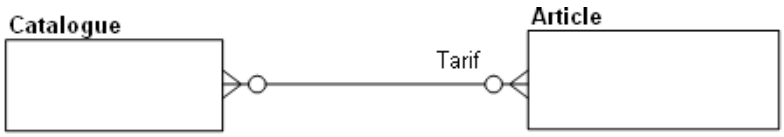
 Une relation optionnelle signifie que chaque instance de la première entité est associée avec zéro ou une instance de la deuxième entité et que la deuxième entité peut être associée à zéro, une ou plusieurs instances de la première entité.



Dans l'exemple ci-dessus, un client peut se voir facturer zéro, une ou plusieurs commandes, et une commande devrait être facturée à un client, mais ce n'est pas obligatoire (problèmes de livraison, etc.).

Relation non-spécifique

 Une relation non spécifique signifie que chaque instance de la première entité est associée avec zéro, une ou plusieurs instances de la deuxième entité et que la deuxième entité peut être associée à zéro, une ou plusieurs instances de la première entité.



Dans l'exemple ci-dessus, un article peut apparaître dans zéro, un ou plusieurs catalogues et un catalogue peut contenir zéro, un ou plusieurs articles.

Créer une association

Pour créer une association :

1. Choisissez le type d'association en cliquant sur le bouton correspondant  ,  ou  dans la barre d'objets.
2. Cliquez dans une des entités concernées, et en gardant le bouton de la souris enfoncé, déplacez le pointeur jusqu'à l'autre entité, avant de relâcher votre pression.
   La fenêtre **Ajout d'une association** s'ouvre.
3. Saisissez le nom de l'association, puis cliquez sur **Créer**.

L'association apparaît dans le diagramme.

Pour modifier la multiplicité d'un rôle :

1. Cliquez avec le bouton droit sur le trait qui se trouve entre l'association et l'entité, afin d'ouvrir le menu contextuel du rôle.
2. Cliquez sur **Propriétés**.
   La page de propriétés du rôle s'ouvre.
3. Cliquez sur l'onglet **Caractéristiques**.
4. Dans le champ **Multiplicité**, sélectionnez la multiplicité voulue.

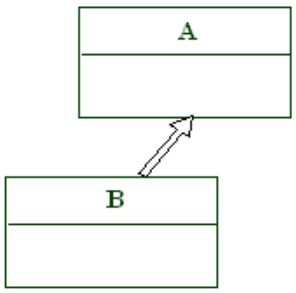
La représentation de l'association change en fonction de la nouvelle valeur de ses multiplicités.

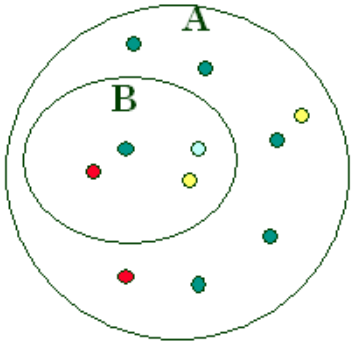
> ☛ *Dans HOPEX Windows Front-End, la multiplicité est également affichée dans le menu contextuel du rôle. Si le menu affiché ne propose pas les multiplicités, vérifiez que vous avez bien cliqué sur le trait qui matérialise le rôle, et non sur l'association.*

Sous-types (I.E)

Une généralisation représente une relation d'héritage entre une entité générale et une entité plus spécifique. L'entité spécifique est cohérente avec l'entité plus générale et hérite de ses caractéristiques et de son comportement. Elle peut cependant comporter des attributs ou des associations supplémentaires. Tout objet de l'entité spécifique est aussi un objet de l'entité générale.

Qu'est-ce qu'un sous-type ?





L'entité B est *sous-type* de l'entité A. Cela suppose que tous les exemplaires de l'entité B sont aussi des exemplaires de l'entité A. Autrement dit, B est un sous-ensemble de A. B est alors la sous-type, A le sur-type.

```
Exemple :

A : Personne, B : Parisien.
```

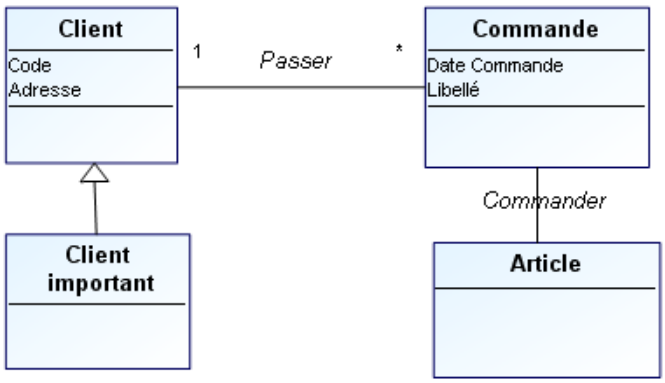
B étant un sous-ensemble de A, les instances de l'entité B "héritent" des caractéristiques de celles de l'entité A.

Il n'est donc pas nécessaire de décrire de nouveau pour l'entité B :

- ses attributs
- ses associations

Exemple :

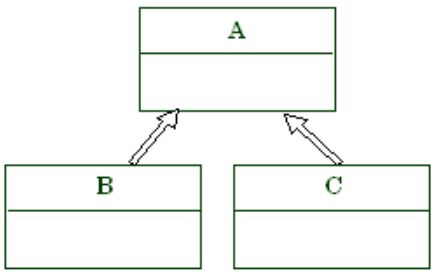
L'entité "Client important" qui représente les clients dont le "C.A. sur les 12 derniers mois" dépasse 1 million d'euros, peut être un sous-type de l'entité client.

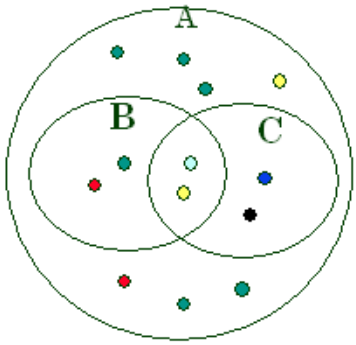


Un sous-type hérite de tous les attributs, associations, rôles et contraintes de son sur-type mais il peut avoir des attributs, associations, rôles ou contraintes que n'a pas son sur-type.

Dans l'exemple ci-dessus, les attributs, les associations, les rôles et les contraintes spécifiés pour "Client" sont aussi valables pour "Client important".

Cas de plusieurs sous-types

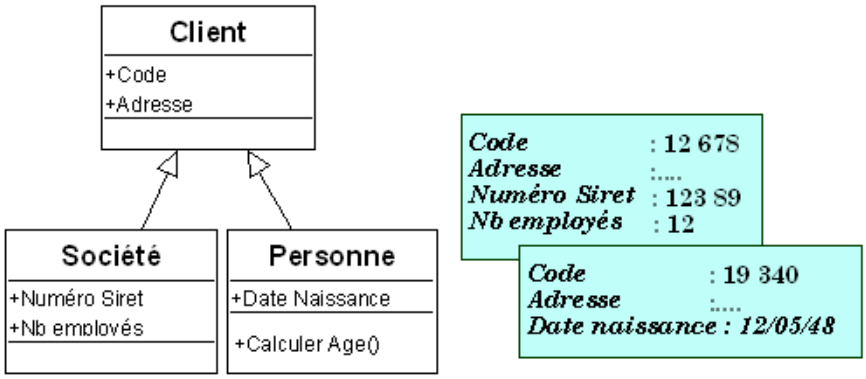




Plusieurs sous-types d'une même entité :

- ne sont pas forcement exclusifs.
- ne forment pas nécessairement une partition du type.

Intérêt des sous-types

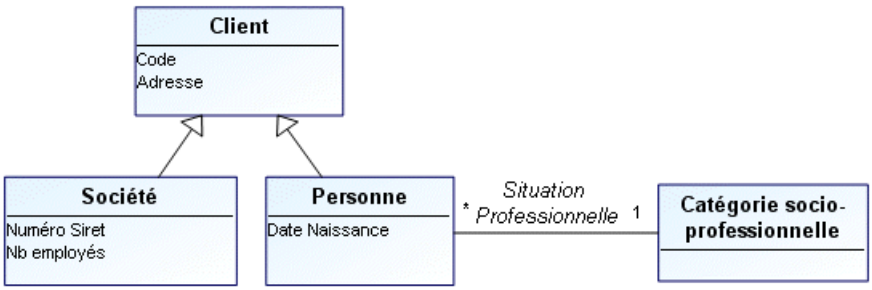


Une entité sous-type hérite de tous les attributs et associations de son sur-type, mais elle peut avoir des attributs ou des associations que ne possède pas son sur-type.

Une entité sous-type peut ainsi avoir des attributs spécifiques. Ceux-ci n'ont de sens que pour une entité sous-type particulière. Dans l'exemple ci-dessus :

- Le numéro de Siret et le nombre d'employés n'ont de sens que pour une société.
- La date de naissance est caractéristique d'une personne, pas d'une société.

Une entité sous-type peut également avoir des associations spécifiques.



- Une personne entre dans une catégorie socio-professionnelle : cadre, employé, commerçant, agriculteur, etc. Cette classification n'a pas de sens pour une entreprise. Il existe également une classification pour les entreprises, mais ce n'est pas la même que pour les personnes.

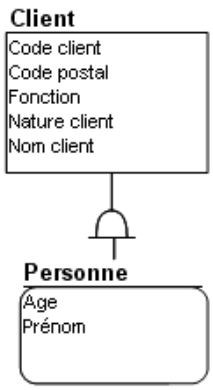
Héritage multiple

Il est parfois utile de spécifier une entité ayant plusieurs entités sur-types. Le sous-type hérite alors de toutes les caractéristiques des deux sur-types. Cette possibilité doit être utilisée avec précaution.

Créer un sous-type

Pour créer un sous-type :

1. Cliquez sur le bouton **Généralisation**  de la barre d'objets.
2. Cliquez sur l'entité sous-type, et faites glisser la souris jusqu'à l'entité sur-type, avant de relâcher votre pression.
   La généralisation est représentée dans le diagramme par un demi-cercle souligné, relié par un trait à l'entité sur-type et par un autre à l'entité sous-type.



Dans l'exemple ci-dessus, certains attributs ont un intérêt pour les personnes et n'ont pas de sens pour d'autres catégories de clients. L'entité sous-type est représentée par une boîte aux angles arrondis.

LA NOTATION MERISE

Condition préalable

Pour utiliser la notation Merise, vous devez cocher l'option correspondante :

1. Dans le bureau d'**HOPEX Information Architecture**, cliquez sur le menu **Menu principal** > **Paramètres** > **Options.**
2. Dans l'arbre de navigation, dépliez le dossier **Modélisation des données**.
3. Cliquez sur **Notation des données**.
4. Dans la partie droite de la fenêtre cochez la notation Merise.
5. Cliquez sur **OK**.

A propos de la modélisation des données

Modéliser les données consiste à identifier les objets de gestion (entités) et les associations ou relations entre ces objets, considérés d'intérêt pour représenter l'activité de l'entreprise.

Il faut que les entités, les associations et les propriétés qui constituent le modèle de données associé à un secteur de l'entreprise suffisent à le décrire complètement sur le plan sémantique.

En d'autres termes, on doit pouvoir décrire l'activité de l'entreprise en utilisant seulement les entités, les associations et les propriétés choisies.

Ceci n'implique pas que, pour chaque mot ou verbe utilisé pour cette explication, il y ait un correspondant direct dans le modèle de données. Il s'agit de pouvoir traduire ce que l'on veut exprimer, au travers des entités, des associations et des propriétés.

Synthèse des concepts

Dans **HOPEX Information Architecture**, un modèle de données (Merise) est représenté par :

- Des entités, qui représentent les concepts de base (client, compte, produit, etc.).
- Des associations, qui définissent les associations entre les différentes entités.
- Des attributs (informations ou propriétés), qui décrivent les caractéristiques des entités et, dans certains cas, des associations.

L'attribut qui permet d'identifier de façon unique l'entité est appelé identifiant.

Le modèle de données est complété par la définition des cardinalités.

Créer un modèle de données (Merise)

Pour créer un modèle de données :

1. Dans **HOPEX Information Architecture**, cliquez sur le volet de navigation **Données logiques**.
2. Dans le volet de navigation, cliquez sur **Tous les modèles de données**.
3. Dans la fenêtre d'édition, cliquez sur le bouton **Nouveau**.
   La fenêtre de création d'un modèle de données apparaît.
4. Saisissez le nom du modèle.
5. Cliquez sur **OK**.
   Le modèle de données apparaît dans la liste des modèles de données.

Diagramme de données (Merise)

Un diagramme de données est une représentation graphique du modèle ou d'une partie du modèle. La création d'un diagramme varie légèrement suivant que vous êtes en Windows Front-End ou Web Front-End.

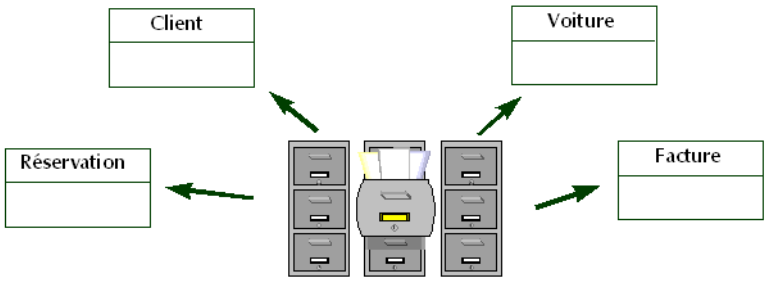
Pour créer un diagramme de données :

- Cliquez avec le bouton droit sur le modèle de données et sélectionnez **Nouveau > Diagramme de données (Merise)**.
  Le diagramme de données s'ouvre.

Les entités (Merise)

Une entité est un objet de gestion considéré d'intérêt pour représenter l'activité de l'entreprise. Une entité est décrite par une liste d'informations (les propriétés), liées à l'entité. Une entité est reliée à d'autres entités via des associations. L'ensemble des entités et des associations constitue le noyau d'un modèle de données.

Nous pouvons illustrer la notion d'*entité* par une comparaison, par exemple, des fiches dans des tiroirs.

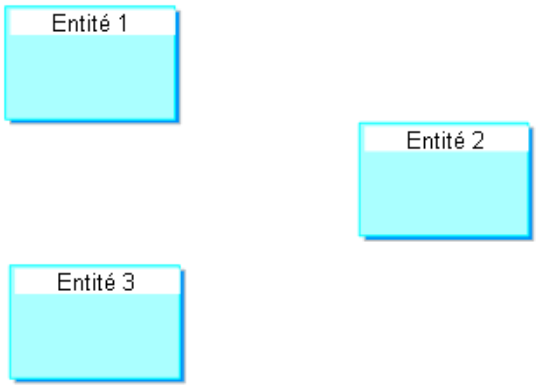


Une entité représente une classe particulière d'objets, dont tous les exemplaires peuvent être décrits de la même manière.

Créer une entité

Pour créer une entité :

1. Sélectionnez le bouton **Entité** dans la barre d'objets.
2. Cliquez sur le plan de travail du diagramme.
   La fenêtre **Ajout d'une entité** s'ouvre.
3. Saisissez le nom de l'entité.
4. Cliquez sur **Créer** (Windows Front-End) ou **Ajouter** (Web Front-End).
   L'entité apparaît dans le diagramme.



☛ *Vous pouvez créer plusieurs entités à la suite sans revenir à la barre d'objets, avec un double-clic sur le bouton . Pour revenir ensuite au mode normal, utilisez la touche <Echap>, ou cliquez sur un autre bouton de la barre d'objets, par exemple sur la flèche .*

☛ *Les objets que vous créez, ainsi que leurs caractéristiques et leurs liens sont enregistrés automatiquement à chaque fois que la pointe de*

la souris prend la forme  . *Le dessin du diagramme n'est enregistré que lorsque vous le demandez explicitement à l'aide du bouton* ***Enregistrer***  .

Préciser l'identifiant d'une entité

Pour préciser l'identifiant d'une entité :

1. Ouvrez la fenêtre de propriétés de l'entité.
2. Cliquez sur l'onglet **Attributs**.
3. Pour l'attribut choisi sélectionnez la valeur "Oui" dans la colonne **Identifiant**.

☛ *Pour plus de détails, voir "Définir l'identifiant d'une entité", page 51.*

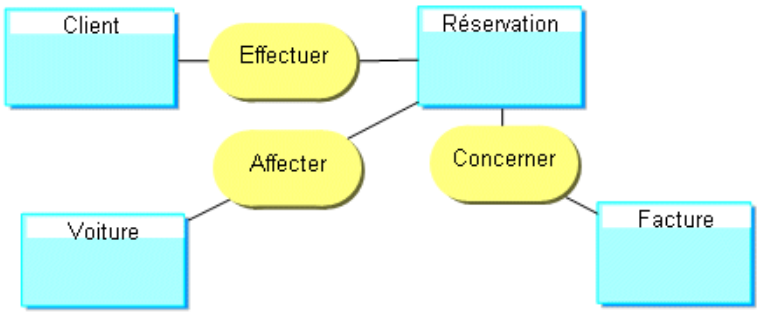
Les associations (Merise)

Une association est une relation entre deux ou plusieurs entités. Une association est dite binaire quand elle relie deux entités, ternaire quand elle en relie trois, etc. Elle peut comporter des propriétés, c'est-à-dire des attributs qui caractérisent l'association des entités.

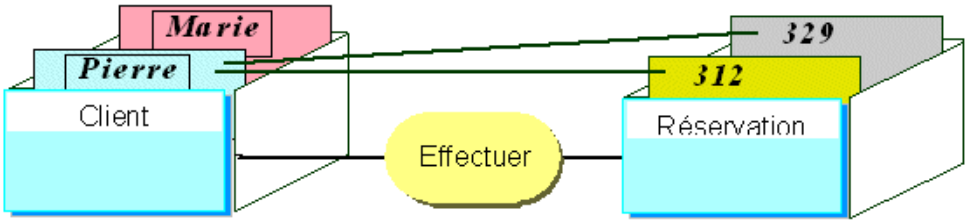
Exemples d'associations

Pour modéliser qu'un "salarié" est responsable d'un "service" et préciser la "date début" de ses fonctions, est créé le modèle de données suivant, où "date début" est une propriété de l'association.

Autre comparaison : des liens entre les fiches.



Le dessin suivant permet de visualiser "en trois dimensions" les situations qu'un modèle de données permet de mémoriser.



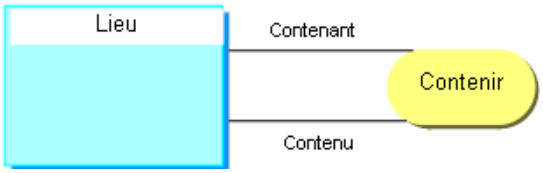
Pierre et Marie sont des clients. Pierre a effectué les réservations numéros 312 et 329.

Un modèle de données doit permettre de mémoriser toutes les situations du contexte de l'entreprise, mais rien que celles-là.

> ☛ *Le modèle ne doit pas permettre de représenter des situations irréalistes ou aberrantes.*

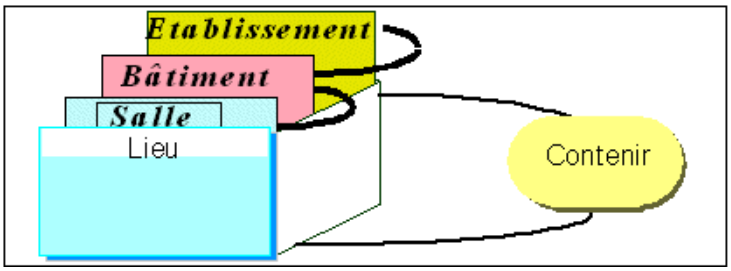
Relation réflexive

Certaines *associations* mettent en jeu la même entité.



Exemple

Une salle de classe, un bâtiment, un établissement scolaire sont tous des lieux.



Une salle de classe est contenue dans un bâtiment, lui-même contenu dans un établissement scolaire.

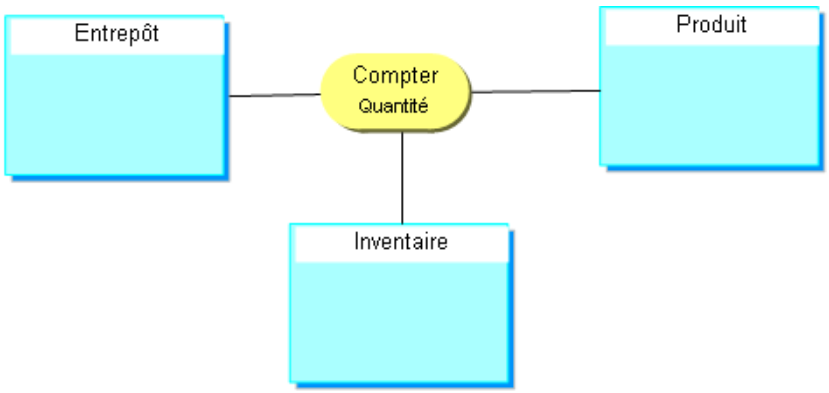
Relation plus que binaire

Certaines associations associent non pas deux, mais davantage d'entités.

Ces associations sont, en principe, rares.

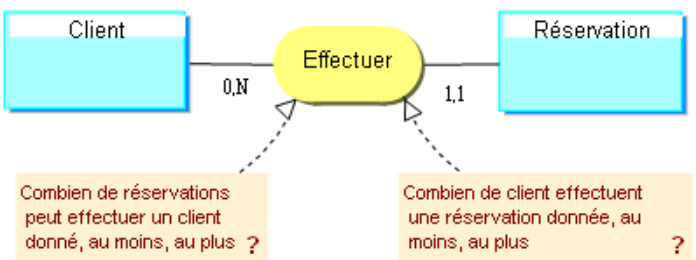
Exemple

Lors d'un inventaire, une certaine quantité de produit a été comptée dans chaque entrepôt.



Les participations ou cardinalités

Cardinalités minimum et maximum expriment le nombre de participations minimum et maximum d'un exemplaire d'une entité à une association.



Les participations ou cardinalités usuelles sont 0,1 1,1 0,N 1,N.

- Participation optionnelle : la cardinalité minimum à 0 indique que l'association n'est pas obligatoirement renseignée.
- Participation obligatoire : La cardinalité minimum à 1 indique que l'association est obligatoirement renseignée.
- Participation unique : La cardinalité maximum à 1 indique que l'entité ne peut être reliée par l'association qu'une fois au plus.
- Participation non unique : La cardinalité maximum à N indique que l'entité peut être reliée par l'association plusieurs fois.

Exemple

L'exemple suivant nous permet d'illustrer la signification de chacune de ces cardinalités ou participations:

0,1 A une commande correspond aucune facture ou une facture au maximum.

0,N Aucune restriction n'est imposée sur le nombre de factures correspondant à une commande. C'est la valeur par défaut.

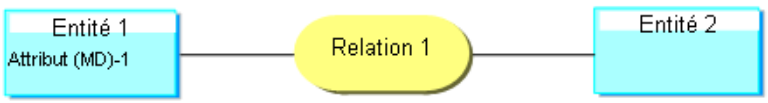
1,1 A chaque commande correspond une facture et une seule.

1,N A chaque commande correspond une ou plusieurs factures.

Créer une association (relation)

Pour créer une association :

1. Cliquez sur le bouton **Association**  de la barre d'objets.
2. Cliquez sur une des entités concernées et faites glisser la souris jusqu'à la deuxième entité, avant de relâcher votre pression.
   La fenêtre **Ajout d'une association** apparaît.
   La flèche placée à l'extrémité du champ **Nom** ouvre un menu qui permet :
   - De **Rechercher** les associations existantes, par l'intermédiaire de la fenêtre **Choix d'une sélection**.
   - De **Lister** les associations de la base.
   - De **Créer** une association.
3. Saisissez le nom de l'association puis cliquez sur **Créer** (Windows Front-End) ou **Ajouter** (Web Front-End)
   L'association apparaît dans le diagramme.



☛ *En cas d'erreur, vous pouvez supprimer un objet en cliquant avec le bouton droit sur cet élément, et en sélectionnant la commande **Supprimer** dans le menu contextuel de l'objet.*

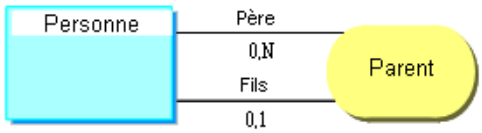
Relation réflexive

Si la demande de création est effectuée sur une entité sans déplacement du pointeur, une association réflexive (aussi appelée "nomenclature") est automatiquement créée sur l'entité.

Dans le cas d'une association entre une entité et elle-même, il est indispensable de préciser les rôles afin de distinguer les liens correspondants dans le dessin.

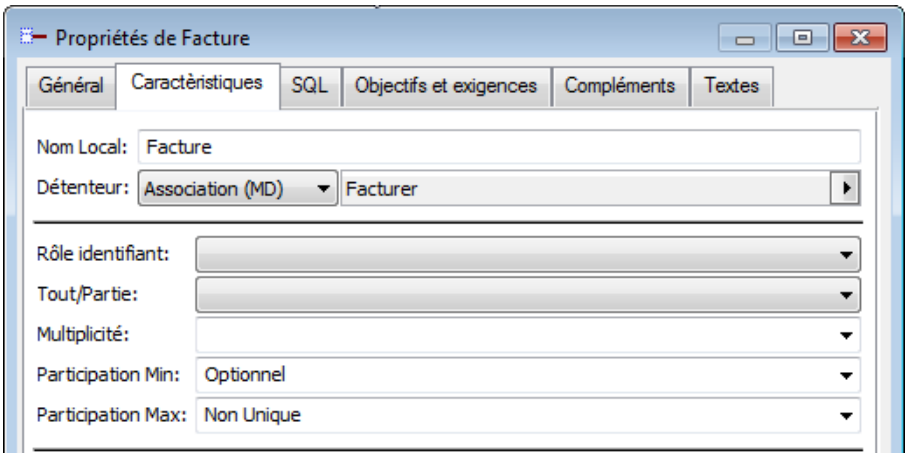
Exemple :

"Père" et "fils" sont les deux rôles joués par l'entité "personne" dans l'association "parent".



Préciser les participations

Dans l'onglet **Caractéristiques** de la fenêtre de propriétés des rôles, vous pouvez indiquer le nombre de participations minimum et maximum de chaque entité à la relation (cardinalités).



Les attributs (informations) - Merise

Les propriétés

Les entités et les associations peuvent être caractérisées par des attributs : leurs propriétés.

Ces attributs ont pu être révélés par l'étude du contenu des messages qui circulent à l'intérieur de l'entreprise.

> ☛ *Un attribut est la donnée élémentaire mémorisée dans le système d'information de l'entreprise. Un attribut est une propriété quand il décrit une entité ou une association, un identifiant quand il est choisi comme moyen d'identification de chaque exemplaire d'une entité.*

Une propriété caractérise une association quand la propriété dépend de l'ensemble des entités participant à l'association.

Dans le diagramme présenté ci-après, le "rôle" qu'un "consultant" a joué sur un "contrat" dépend du consultant et du contrat, donc de l'association "intervenir".

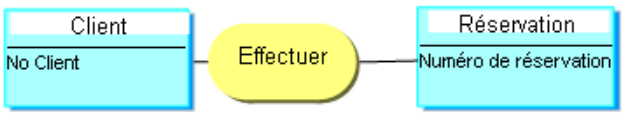
Exemples d'attributs

"Nom du client" (Propriété de l'entité client).

"No client" (identifiant de l'entité client).

"Solde du compte" (Propriété de l'entité compte).

L'identifiant



Le client qui a le numéro 2718 effectue la réservation 314159

Chaque entité possède un *identifiant* unique qui permet de retrouver sans ambiguïté chacun de ses exemplaires.

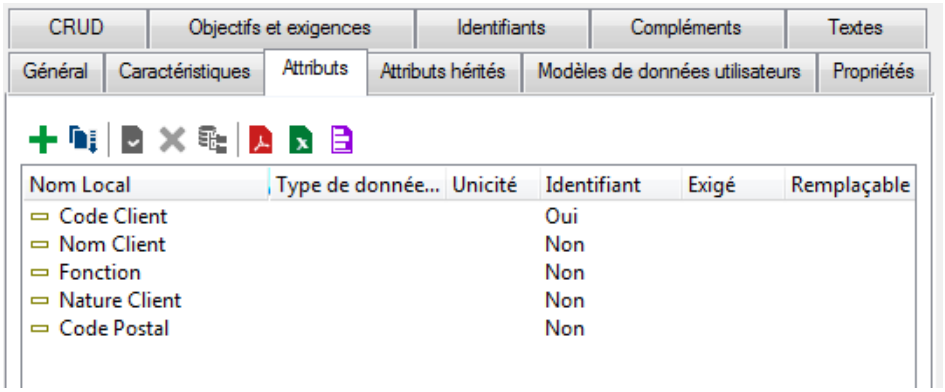
> ☛ *Un identifiant est constitué d'un ou plusieurs attributs ou rôles obligatoires qui permettent d'identifier de façon unique une entité.*

Les associations n'ont pas d'identifiants propres : une association est identifiée par les identifiants des entités reliées.

Créer des attributs

La création des *Attributs* se fait dans la fenêtre de propriétés des associations et des entités.

L'onglet **Attributs** de cette fenêtre présente les attributs déjà reliés à l'entité ou à l'association.



Pour créer un attribut :

- Cliquez sur le bouton ⊕ et saisissez le nom de l'attribut.

Vous pouvez préciser ses caractéristiques (Voir "Description des attributs", page 97 pour plus de détails).

> ☛ *Vous pouvez préciser sa **Longueur**, éventuellement complétée par le nombre de **Décimales** ; il faut noter que le nombre de décimales ne s'ajoute pas à la longueur ; une information de longueur 5 avec deux décimales se présentera sous la forme " 999,99 ".*

Quand vous avez terminé, fermez la fenêtre de propriétés.

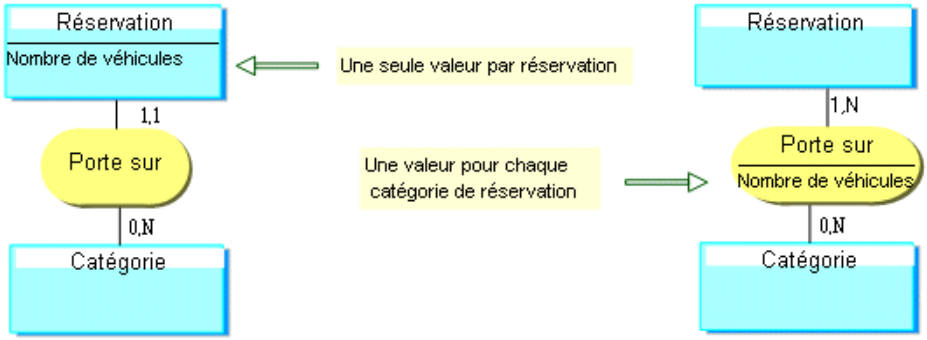
Les règles de normalisation (Merise)

Les formes normales sont des règles qui visent à éviter des erreurs de modélisation.

A ce jour, il existe six ou sept formes normales. Nous allons voir les trois premières.

Première forme normale

La valeur d'une Propriété d'entité (ou d'association) est fixée de manière unique dès que l'on connaît l'entité concernée (les entités concernées).

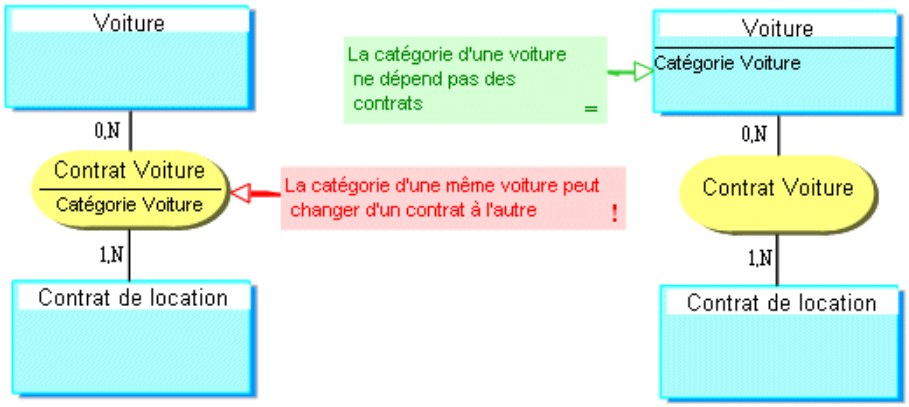


Si le nombre de véhicules est porté par l'entité "Réservation", on ne peut indiquer que le nombre total de véhicules pour une réservation. On doit donc faire une réservation par catégorie de véhicule loué (cardinalités 1,1).

Si le nombre de véhicule est porté par l'association, on peut préciser le nombre de véhicules réservés pour chaque catégorie sur l'association. On peut donc faire une seule réservation pour plusieurs catégories de véhicules (cardinalités 1,N)

Deuxième forme normale

La valeur d'une Propriété d'association n'est fixée que lorsque l'on connaît toutes les entités concernées.

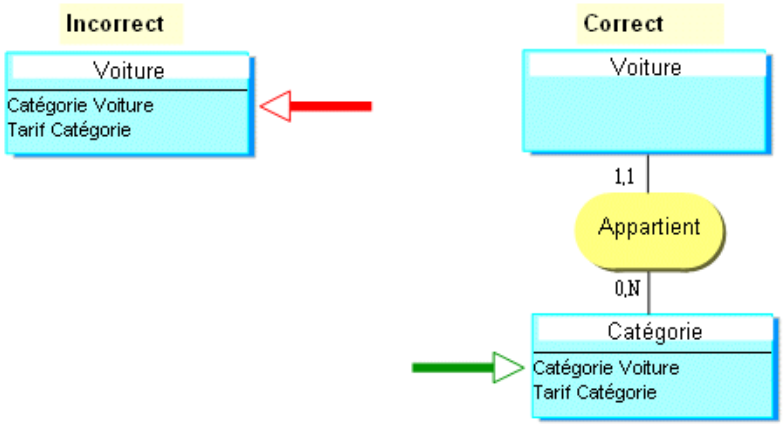


Le fait que la catégorie de voiture porte sur l'association "Catégorie Voiture" suppose que la catégorie de la voiture puisse changer d'un contrat à l'autre, ce qui ne serait pas très honnête.

Pour que la catégorie de la voiture ne dépende pas du contrat, il faut qu'elle soit portée par l'entité "Voiture".

Troisième forme normale

Une Propriété doit dépendre directement et uniquement de l'entité qu'elle décrit.



Si le "Tarif Catégorie" est porté par l'entité "Voiture", cela signifie que deux voitures de la même catégorie peuvent avoir un "Tarif Catégorie" différent. Pour éviter cela, il faut créer une entité "Catégorie" qui portera le tarif.

☛ *Cette règle permet de faire émerger des concepts qui n'apparaissent pas dans la première ébauche de modèle de données.*

Compléter la spécification du modèle de données (Merise)

En cours de spécification, il est souvent nécessaire de compléter le modèle de données.

Les compléments de spécification consistent à :

- Renseigner les caractéristiques Longueur et Décimale et documenter les attributs.

Dans le modèle de données, il est également possible de spécifier :

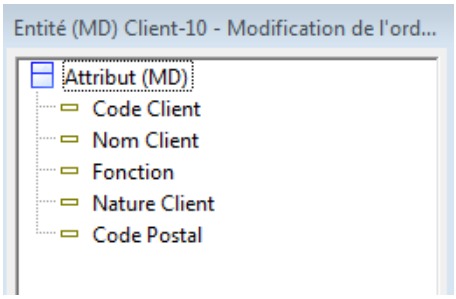
- Les entités sous-types.
- Les contraintes que les données doivent respecter, à titre documentaire. Ces contraintes se traduiront par des contrôles à effectuer lors des traitements de mise à jour des données.

Ordonner les attributs

L'ordre initial des attributs est l'ordre de création (ou du création du lien avec l'entité ou l'association).

Pour modifier cet ordre :

1. Dans l'onglet **Attributs** de la fenêtre de propriétés de l'objet, cliquez sur le bouton **Réordonner**.
   La fenêtre **Modification de l'ordre** est présentée.



Pour réordonner les attributs :

1. Sélectionnez l'attribut à déplacer en cliquant sur son nom.
2. Déplacez le curseur jusqu'à l'emplacement souhaité ; il prend la forme suivante :
   L'attribut est placé à l'endroit indiqué, et l'ordre du lien avec l'entité est modifié.

Cet ordre sera utilisé pour générer l'ordre des colonnes dans les tables. Il sera également exploité dans le document associé au modèle de données.

Description des attributs

La description des attributs peut se faire de deux façons :

- Dans la fenêtre de propriétés de l'entité qui les détient : en saisissant cette description dans les différentes cellules de la liste présentée dans l'onglet **Attributs**.
- Dans la fenêtre de propriétés de chaque attribut. Cette fenêtre est ouverte par la commande **Propriétés** du menu contextuel d'un attribut.

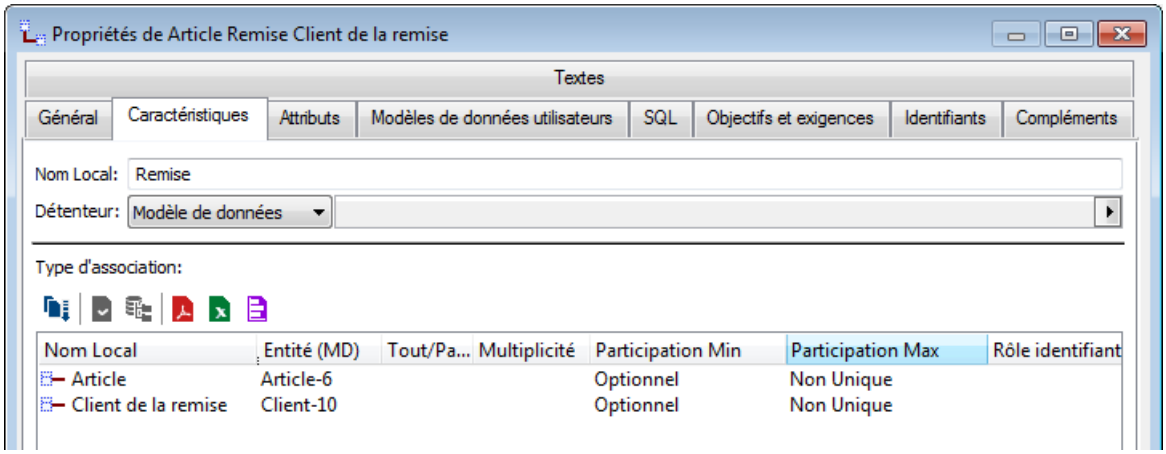
Vous pouvez saisir la valeur des caractéristiques des attributs dans les champs correspondants.

- Le **Type de données** qui est la classe utilisée pour préciser le type de l'attribut.
- Le champ **Identifiant** indique si l'attribut fait partie de l'identifiant de l'entité.
- Le champ **Exigé** qui permet de préciser s'il est obligatoire de saisir une valeur pour cet attribut.
- Le champ **Unicité** qui permet d'indiquer que deux instances de cette entité ne peuvent avoir la même valeur pour cet attribut.
- Le champ **Remplaçable** qui permet d'indiquer que la valeur de cet attribut n'est plus modifiable une fois qu'elle a été saisie.

Participations ou Cardinalités

Pour modifier les participations ou *cardinalités* d'une association :

1. Ouvrez la fenêtre de propriétés de l'association.
2. Cliquez sur l'onglet **Caractéristiques**.
3. Saisissez les valeurs de participations (cardinalités)



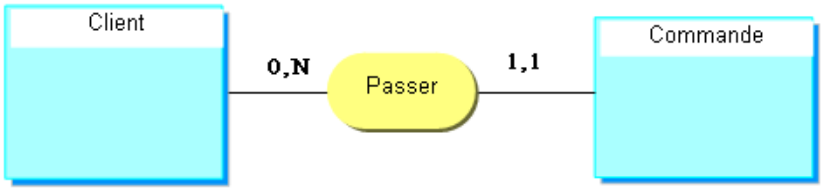
☛ *Une cardinalité est le nombre de fois minimum (respectivement maximum) où une entité "participe" à une association (voir aussi multiplicité).*

Les cardinalités ou participations les plus communément utilisées sont :

- 0 ou 1 pour la cardinalité minimum (participation minimum optionnelle ou obligatoire).
- 1 ou N pour la cardinalité maximum (participation maximale unique ou non unique).
  Des valeurs différentes sont admises.
  Quand plusieurs rôles, c'est-à-dire plusieurs liaisons, existent entre une entité et une association, les cardinalités sont définies pour chaque rôle.
  La cardinalité d'une entité dans une association peut aussi être définie comme suit :
- Pour une association binaire, c'est le nombre minimum (ou maximum) d'exemplaires de l'autre entité intervenant dans l'association, qui peuvent être liés à l'entité de départ.
- Pour une association ternaire, c'est le nombre de couples des deux autres entités intervenant dans l'association, qui peuvent être liés à l'entité de départ.
- Pour une association quaternaire, c'est le nombre de triplets, etc.

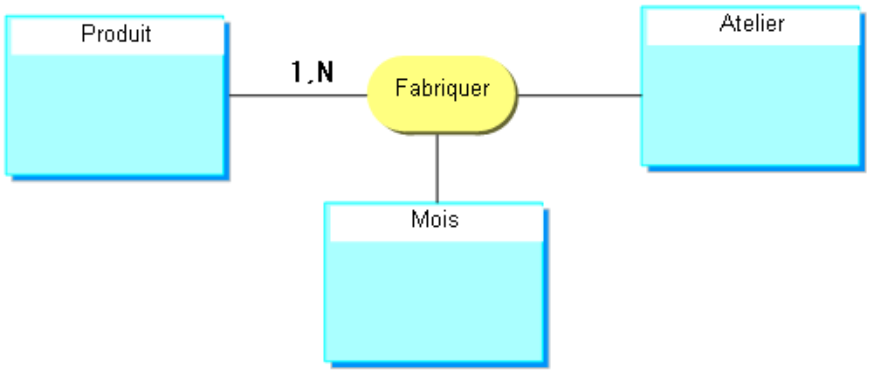
☛ *Si l'expression des cardinalités n'est pas suffisante pour décrire la liaison qui existe entre une entité et une association, par exemple quand une cardinalité dépend d'un contexte d'organisation, il est possible d'utiliser les contraintes de cardinalité, qui permettent une description plus précise.*

Exemples



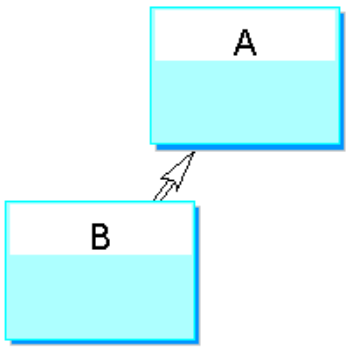
0,N : Le client peut ne pas passer de commande, il peut passer un maximum de N commandes (N indéterminé).

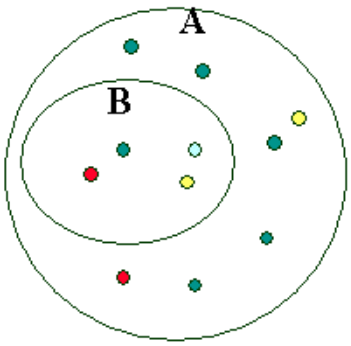
1,1 : La commande doit être passée par un client et un seul.



1,N : Un produit doit être fabriqué au minimum dans 1 atelier pendant un mois. Il peut être fabriqué dans plusieurs ateliers et/ou pendant plusieurs mois (plusieurs couples atelier-mois).

Sous-typage (Merise)





Qu'est-ce qu'un sous-type

L'entité B est sous-type de l'entité A. Cela suppose que tous les exemplaires de l'entité B sont aussi des exemplaires de l'entité A. Autrement dit, B est un sous-ensemble de A.

Exemple A : Personne, B : Parisien.

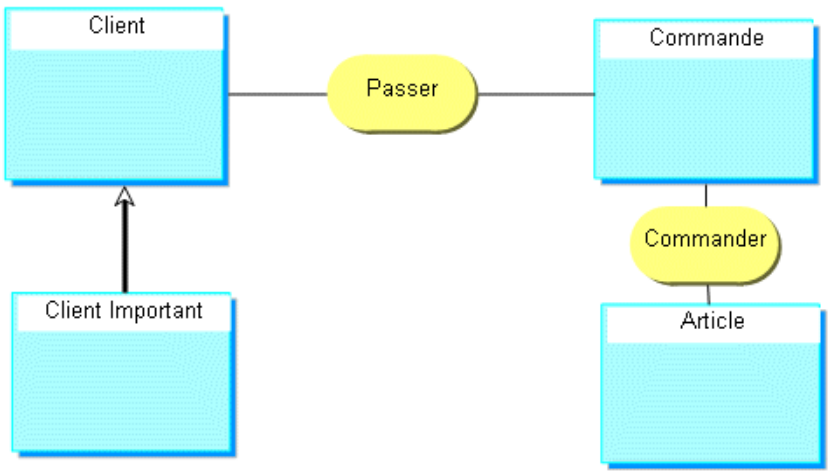
B étant un sous-ensemble de A, les exemplaires de l'entité B " héritent " des caractéristiques de ceux de l'entité A.

Il n'est donc pas nécessaire de décrire de nouveau pour l'entité B :

- ses propriétés,
- ses associations.

Exemple

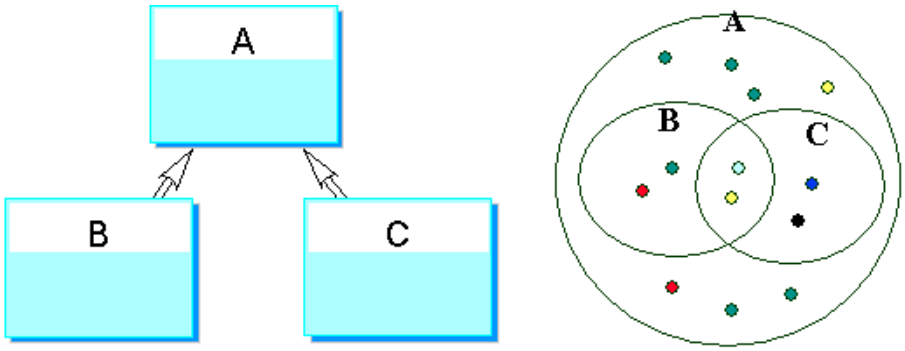
L'entité "Client important" qui représente les clients dont le "C.A. sur les 12 derniers mois" dépasse 1 million d'euros, peut être un sous-type de l'entité client (origine).



Un sous-type hérite de toutes les propriétés, les associations, les rôles et les contraintes de son sur-type mais il peut avoir des propriétés, associations, rôles ou contraintes que n'a pas son sur-type.

Dans l'exemple ci-dessus, les propriétés, les associations, les rôles et les contraintes spécifiés pour "Client" sont aussi valables pour "Client important".

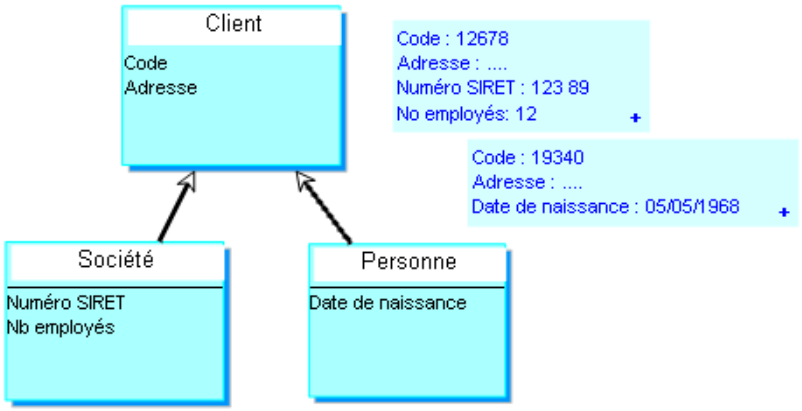
Cas de plusieurs sous-types



Plusieurs sous-types d'une même entité

- ne sont pas forcement exclusifs.
- ne forment pas nécessairement une partition du type.

Intérêt des sous-types

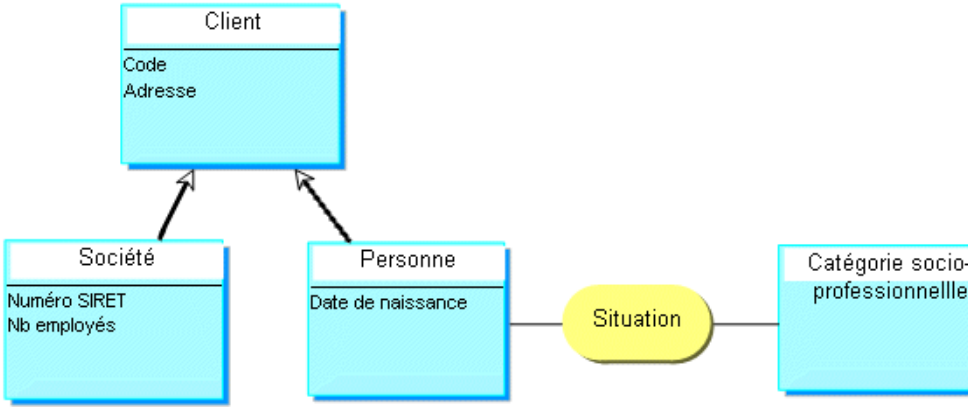


Une entité sous-type peut avoir des propriétés spécifiques. Celles-ci n'ont de sens que pour un sous-type particulier. Dans l'exemple ci-dessus :

- Le numéro de Siret et le nombre d'employés n'ont de sens que pour une société.
- La date de naissance est caractéristique d'une personne, pas d'une société.

> 🕮 *Une entité B est sous-type d'une entité A, si B représente un sous-ensemble de A et que les exemplaires de l'entité B héritent de la description de ceux de l'entité A et si ils ont des éléments descriptifs spécifiques.*
>
> *Le lien Sous-type est représenté graphiquement par une flèche double.*

Une entité sous-type peut également avoir des associations spécifiques.



Une personne entre dans une catégorie socio-professionnelle : cadre, employé, commerçant, agriculteur, etc. Cette classification n'a pas de sens pour une entreprise (il existe également une classification pour les entreprises, mais ce n'est pas la même que pour les personnes.)

TYPES DES ATTRIBUTS

Les points suivants sont abordés ici :

PAQUETAGES DE TYPES DE DONNÉES

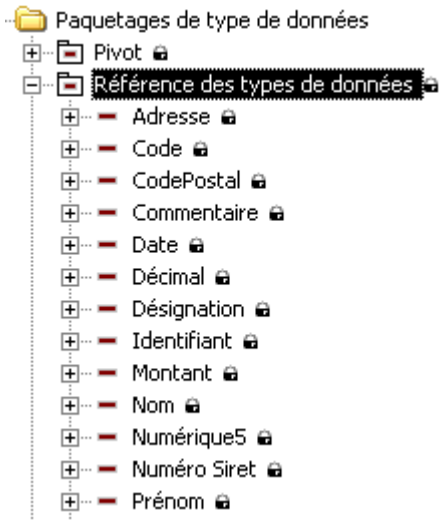
> 🕮 *Un modèle de données permet de représenter la structure statique d'un système, en particulier les types d'objets manipulés dans le système, leur structure interne et les relations qui existent entre eux. Un modèle de données regroupe un ensemble d'entités avec leurs attributs, les associations qui existent entre ces entités, des contraintes qui portent sur ces entités et associations, etc.*
>
> 🕮 *Un type de données permet de mettre en commun des caractéristiques communes à plusieurs attributs. Les types de données sont implémentés sous forme de classes.*
>
> 🕮 *Un paquetage de types de données est un paquetage de référence détenant tout ou partie des types de données utilisés dans l'entreprise. Chacun des autres paquetages sera déclaré client du paquetage de référence des types de données.*

Un type de données définit la nature des valeurs que peut prendre une donnée. Il peut être simple (entier, caractère, text, booléen, date...) ou plus élaboré et composé.

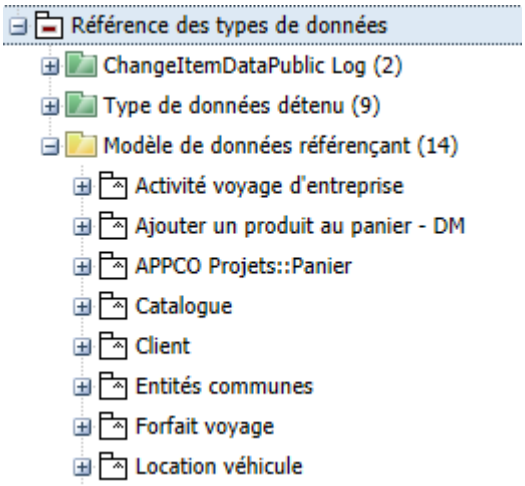
Pour typer les attributs d'une entité, ne sont proposés que les types de données définis pour le modèle de données qui contient cette entité.

Lorsque vous créez un modèle de données, le paquetage de types de données "Référence des types de données" lui est automatiquement associé par défaut.

Ce paquetage "Référence des types de données" détient les types de données "Adresse", "Code", "Date", etc.



En ouvrant l'explorateur sur ce paquetage de type de données, vous pouvez voir qu'il est référencé par plusieurs modèles de données.



Les attributs des entités de ces modèles peuvent donc être typés à l'aide des types de données "Adresse", "Code", "Date", etc.

Créer un nouveau paquetage de types de données

Vous pouvez définir un nouveau paquetage de types de données de référence détenant les types de données utilisés dans l'entreprise.

Pour créer votre propre paquetage de types de données :

1. Dans le bureau d'**HOPEX Information Architecture**, cliquez sur le volet de navigation **Données logiques**.
2. Affichez la liste des **Paquetages de types de données**.
3. Dans la zone d'édition, cliquez sur **Nouveau**.
4. Tapez le nom du paquetage et cliquez sur **OK**.

Vous pouvez ensuite ajouter des types à ce paquetage.

Créer un type de données

Pour créer un type de données :

1. Cliquez avec le bouton droit sur le nom du paquetage et ouvrez ses **Propriétés**.
   Les propriétés du paquetage apparaissent.
2. Cliquez sur la liste déroulante puis sur **Type de données**.
3. Cliquez sur **Nouveau**.
   La fenêtre de création d'un type de données apparaît.
4. Saisissez le nom du type et cliquez sur **OK**.

Type de données composé

Vous pouvez créer des types de données composés en leur ajoutant une liste d'attributs, par exemple un type "Adresse" composé du numéro, de la rue, du code postal, de la ville et du pays.

Valeur littérale

Vous pouvez affecter des valeurs littérales à un type de données qui définissent les valeurs qu'il peut prendre. Les attributs basés sur un tel type de données ne peuvent prendre que les valeurs définies par le type de données.

Une fois le nouveau paquetage de types de données créé, il convient de le référencer sur le modèle de données client.

Référencer un paquetage de types de données

Pour relier un paquetage de types de données à un modèle de données :

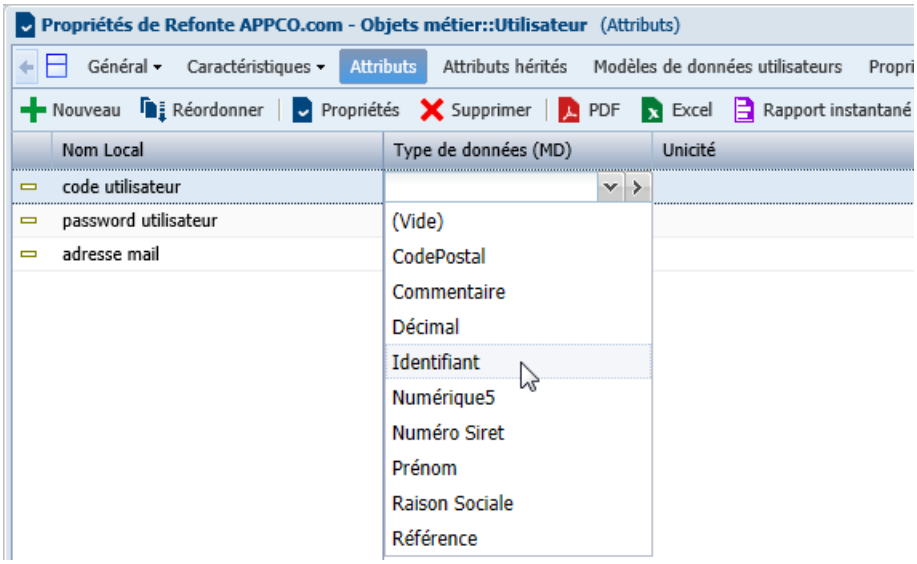
1. Dans le bureau d'**HOPEX Information Architecture**, cliquez sur le menu de navigation puis sur **Données logiques**.
2. Dans le volet de navigation des données logiques, cliquez sur **Tous les modèles de données**.
   La liste des modèles de données apparaît dans la zone d'édition.
3. Faites un clic droit sur le modèle concerné et ouvrez ses **Propriétés**.
4. Cliquez sur la liste déroulante puis sur **Paquetage des types de données utilisés**.
5. Cliquez sur **Relier**.
   La fenêtre de recherche apparaît.
6. Cliquez sur **Chercher**.
   La liste des paquetages de types de données s'affiche.
7. Sélectionnez le paquetage voulu et cliquez sur **Relier**.

AFFECTER DES TYPES AUX ATTRIBUTS

Une fois le paquetage de types de données référencé pour le modèle de données, la liste des types qu'il contient est disponible sur chaque attribut des entités du modèle. Il vous reste à sélectionner celui qui convient.

Pour définir le type d'un attribut :

1. Cliquez avec le bouton droit sur l'entité qui contient l'attribut.
2. Sélectionnez **Propriétés**.
   La fenêtre de propriétés de l'attribut apparait.
3. Cliquez sur la liste déroulante puis sur **Attributs**.
4. Dans le champ **Type de données (MD)** correspondant à l'attribut, sélectionnez le type voulu dans la liste.
5. Cliquez sur **Appliquer**.



HOPEX Database Builder

Guide d’utilisation

HOPEX V2R1

HOPEX INFORMATION ARCHITECTURE - COUCHE PHYSIQUE

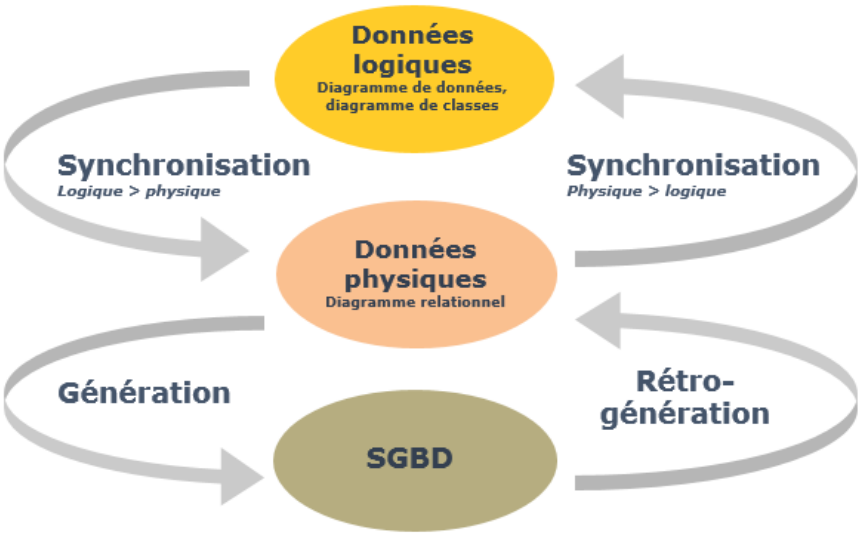
HOPEX Information Architecture permet de concevoir et de modéliser des bases de données.

Une base de données décrit l'organisation d'un système sous forme d'un ensemble de données stockées de façon structurée.

HOPEX Information Architecture intègre par défaut les niveaux de modélisation logique et physique et permet le passage d'un modèle à un autre. Vous pouvez ainsi :

- Construire un diagramme de données ou diagramme de classes.
- Créer, à partir de ce diagramme, les tables d'une base de données, avec ses colonnes, index et clés ainsi que les dessins du diagramme relationnel correspondant.
- Optimiser le diagramme relationnel obtenu et générer les ordres SQL de définition des tables. **HOPEX Information Architecture** permet en particulier de prendre en compte des évolutions du modèle conceptuel sans perdre les optimisations effectuées sur le diagramme relationnel.
- Rétro-générer la définition d'une base de données à l'aide du protocole ODBC, pour créer les tables et colonnes correspondantes dans **HOPEX**

Information Architecture, et obtenir le diagramme de données ou diagramme de classes correspondant.



Options de modélisation des données

Les formalismes

Vous pouvez modéliser les données logiques à partir de deux formalismes :

- le paquetage de données, pour construire les diagrammes de classes (notation UML). Ce formalisme est coché par défaut.
- le modèle de données, pour les diagrammes de données (notations standard, IDEF1X, I.E, Merise)

Pour afficher un des formalismes :

1. Dans le bureau, cliquez sur le menu **Menu principal** > **Paramètres** > **Options.**
2. Dans l'arbre de navigation, dépliez le dossier **Modélisation des données**.
3. Cliquez sur **Formalisme de données**.
4. Dans la partie droite de la fenêtre cochez le(s) formalisme(s) que vous voulez afficher.
5. Cliquez sur **OK**.
   Les dossiers correspondant aux paquetages et aux modèles de données apparaissent dans le volet de navigation **Données logiques**.

Formalisme logique et synchronisation

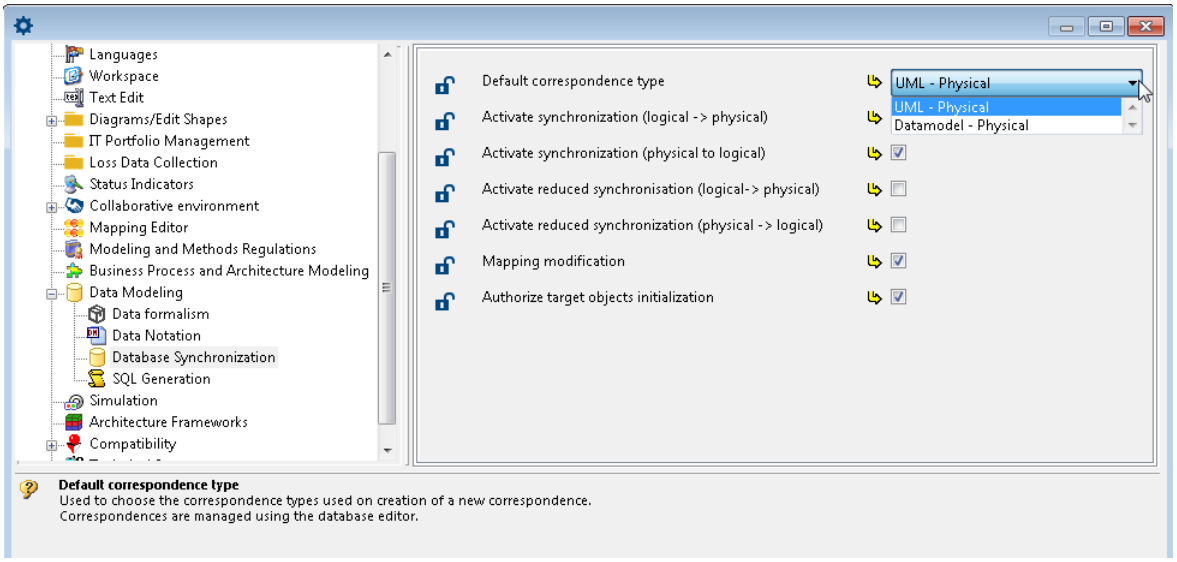
Dans **HOPEX Information Architecture** V2R1, le formalisme logique appliqué par défaut dans la synchronisation est la notation UML avec la prise en compte des

parties. Les associations ne sont plus traitées ; si vous synchronisez un modèle de données en modèle physique, les associations du modèle ne sont pas traitées par la synchronisation.

A titre de compatibilité il est possible de revenir au traitement précédent, à savoir la prise en compte de la notation UML et des modèles de données, avec le traitement des associations et non des parties. Le changement se fait dans l'application **HOPEX Administration**.

Pour accéder à l'option :

1. Ouvrez l'outil d'administration.
2. Ouvrez les options de l'environnement concerné.
3. Dans la fenêtre des options, dans l'arbre de gauche, dépliez le dossier **Data Modeling**.
4. Cliquez sur **Database synchronization**.
5. Dans la partie droite de la fenêtre, dans **Default correspondence type**, sélectionnez la valeur voulue :
   - UML - Physical : option par défaut (prise en compte des parties)
   - Datamodel - Physical : ancienne option (prise en compte des associations)



Les notations

Vous disposez d'une notation standard de modèle de données, cochée par défaut.Pour afficher une autre notation (DEF1X, I.E ou Merise) :

1. Dans le bureau, cliquez sur le menu **Menu principal** > **Paramètres** > **Options.**
2. Dans l'arbre de navigation, dépliez le dossier **Modélisation des données**.
3. Cliquez sur **Notation des données**.
4. Dans la partie droite de la fenêtre cochez les notations que vous voulez utiliser.

5. Cliquez sur **OK**.

Accès au référentiel

Pour utiliser les fonctionnalités de **HOPEX Information Architecture**, vous devez avoir un accès au référentiel en mode "Avancé" :

1. Cliquez sur **Menu principal** > **Paramètres** > **Options**.
2. Dans la partie gauche de la fenêtre, cliquez sur le dossier **Référentiel**.
3. Dans la partie droite, vérifiez que l'accès au référentiel est en mode "Avancé".

A PROPOS DE CE GUIDE

Structure du guide

Ce guide traite les points suivants :

- "Base de données et modélisation physique", page 7 indique comment créer une base de données et son diagramme relationnel.
- "Synchroniser les modèles logiques et physiques", page 1, décrit comment synchroniser automatiquement un modèle conceptuel de données avec une base de données décrite en formalisme relationnel, et vice versa.
- "Correspondance des modèles", page 69, présente l'application qui permet de mettre en correspondance les différentes vues d'une base de données.
- "Dénormaliser les modèles logiques et physiques", page 79, explique comment modifier des modèles en fonction de besoins spécifiques.
- "Types des attributs et des colonnes", page 99, présente les types de données des attributs et leur correspondance avec les datatypes des colonnes.
- "Générer des scripts SQL", page 1, présente la génération des ordres SQL de définition des tables à partir des spécifications effectuées avec **HOPEX Information Architecture**.
- "Options SQL avancées", page 135, décrit l'utilisation des vues, triggers, procédures stockées, ainsi que l'ajout de propriétés physiques sur les objets d'une base de données.
- "Rétro-générer des tables", page 153, présente comment créer les tables et les colonnes dans **HOPEX Information Architecture** à partir des bases de données accessibles par le protocole ODBC, et proposer le diagramme de classes correspondant en formalisme UML.
- "Utilitaire d'extraction ODBC", page 159, décrit l'utilitaire utilisé pour l'accès aux bases de données.
- "Glossaire", page 1, présente les définitions des objets manipulés dans **HOPEX Information Architecture**.
- "Tableaux de correspondances entre types pivots et datatypes", page 171, liste les tableaux de correspondances entre les types pivots et les datatypes de tous les SGBD concernés.

Ressources complémentaires

Ce guide est complété par :

- le guide **HOPEX Common Features**, qui décrit l'interface Web et les outils spécifiques aux solutions MEGA.

  ☛ *Il peut être utile de consulter ce guide pour une présentation générale de l'interface.*

- le guide d'administration **HOPEX Power Supervisor**.

Conventions utilisées dans le guide

Styles et mises en forme

☛ *Remarque sur les points qui précèdent.*

📖 *Définition des termes employés.*

☺ *Astuce qui peut faciliter la vie de l'utilisateur.*

Compatibilité avec les versions précédentes.

Ce qu'il faut éviter de faire.

Remarque très importante à prendre en compte pour ne pas commettre d'erreurs durant une manipulation.

Les commandes sont présentées ainsi : **Fichier > Ouvrir**.

Les noms de produits et de modules techniques sont présentés ainsi : **HOPEX**.

BASE DE DONNÉES ET MODÉLISATION PHYSIQUE

Le modèle de données permet de représenter la structure statique des données au niveau métier. La base de données est l'objet qui permet de stocker et d'organiser ces données en vue de leur utilisation par des programmes correspondant à des applications distinctes, et de manière à faciliter l'évolution indépendante des données et des programmes.

Vous pouvez relier un modèle de données et le diagramme associé à une base de données. Vous avez ensuite la possibilité de créer le diagramme relationnel correspondant.

Les points abordés ici sont :

ARCHITECTURE DES DONNÉES PHYSIQUES

Structure des données physiques - Éléments de base

Base de données

Une base de données permet de spécifier la structure de stockage physique des données.

Voir "La base de données", page 11.

Domaine de données physiques

Un domaine de données représente une structure de données physiques restreinte. Il est constitué de tables et/ou de vue physiques et peut être décrit par un diagramme.

Pour répondre à des cas d'utilisation précis, vous pouvez créer des Vues physiques dans lesquelles vous pouvez visualiser et modifier le périmètre couvert par des tables.

Voir "Les domaines de données physiques", page 9.

Vue physique

Une vue physique permet de représenter le périmètre couvert par un élément de domaine de données. Une vue physique est construite à partir d'une sélection de plusieurs tables reliées dans le contexte spécifique de la vue..

Voir "Définir les vues d'une base de données", page 136

LES DOMAINES DE DONNÉES PHYSIQUES

Un domaine de données physiques permet de définir une structure de données physiques constituée de tables et/ou de vues physiques.

Créer un domaine de données physiques

Pour créer un domaine de données physiques :

1. Cliquez sur le menu de navigation puis sur **Données physiques**.
2. Dans le volet de navigation, cliquez sur **Domaines de données physiques**.
   On distingue trois types de domaine de données physiques :
   - Domaine de données relationnelles : représente un ensemble de données stockées dans un système de gestion de base de données relationnel et utilisé dans l'architecture technique d'une application.
   - Domaine de données non SQL : représente un ensemble de données stockées dans un système de gestion de base de données NOSQL et utilisé dans l'architecture technique d'une application.
   - Structures de fichier
3. Sélectionnez le type de domaine de données et cliquez sur le bouton **Nouveau**.
   Le nouveau domaine de données est créé. Vous pouvez ouvrir ses propriétés pour modifier ou compléter ses caractéristiques.

Domaine de données relationnelles

Un domaine de données relationnelles peut être décrit par deux types de diagramme :

- le diagramme des tables qui permet de visualiser un ensemble de tables et leurs relations (FK).
- le diagramme de structure qui permet de décomposer un domaine en sous-domaines.

Vous pouvez relier plusieurs diagrammes à un domaine de données, suivant ce que vous voulez décrire.

Pour créer un diagramme à partir du domaine de données relationnelles :

- Faites un clic droit sur le domaine de données physiques et sélectionnez **Nouveau**, suivi du type de diagramme (tables ou structure).

La carte de données physiques

Une carte de données physiques est un outil d'urbanisation des informations physiques. Elle permet de représenter un ensemble de domaines de données physiques dans un contexte particulier.

Créer une carte de données physiques

Pour créer une carte de données physiques :

1. Cliquez sur le menu de navigation puis sur **Données physiques**.
2. Dans le volet de navigation, cliquez sur **Cartes de données physiques**.
3. Sélectionnez le type de carte de données physiques.
4. Cliquez sur **Nouveau**.
   La carte créée apparaît.

Pour créer le diagramme de la carte de données physique :

1. Faites un clic droit sur la carte et sélectionnez **Nouveau** > **Diagramme de carte de données**.
   Le diagramme apparaît dans la zone d'édition.

Les composants d'une carte de données physiques

Dans une carte de données physiques vous pouvez ajouter des composants internes et externes.

Les composants internes sont les domaines de données qui font partie du périmètre de la carte (qu'ils appartiennent ou non au même élément détenteur).

Les composants externes sont ceux qui sont utilisés dans la carte mais qui ne font pas partie du périmètre étudié.

LA BASE DE DONNÉES

Sur une base de données, et en fonction du SGBD cible, vont être définis les paramètres de pilotage des différents outils de traitement des données (synchronisation, génération, rétro-génération etc.).

Créer une base de données

Une base de données permet de spécifier la structure de stockage physique des données.

Pour créer une *base de données* dans **HOPEX Information Architecture** :

1. Cliquez sur le menu de navigation puis sur **Données physiques**.
2. Dans le volet de navigation, cliquez sur **Base de données**.
   La liste des bases de données apparaît dans la zone d'édition.
3. Cliquez sur le bouton **Nouveau**.
   La base de données créée apparaît dans la liste des bases. Vous pouvez modifier son nom.

Propriétés d'une base de données

Pour accéder aux propriétés d'une base de données :

1. Cliquez sur le menu de navigation et cliquez sur **Données physiques**.
2. Dans le volet de navigation cliquez sur **Base de données**.
   La liste des bases de données apparaît dans la zone d'édition.
3. Sélectionnez la base de données voulue et cliquez sur le bouton **Propriétés**  .
   La fenêtre de propriétés de la base de données apparaît.
4. Cliquez sur la liste déroulante pour accéder aux différentes pages de propriétés.

Les pages de propriétés permettent :

- D'accéder aux **Composants** de la base de données (tables, vues physiques, groupements de données, etc.).
- De modifier les **Caractéristiques** de la base (nom, SGBD cible, etc.).
- De définir les **Responsabilités**.
- De définir les **Risques** associés, les **Standards** utilisés, les **Objectifs et exigences**.
- De définir les **Options** liées à :
  - la génération des tables. Voir "Paramétrer la génération d'une base de données", page 10.
  - la synchronisation. Voir "Paramétrer la synchronisation", page 37.

Associer un paquetage à une base de données

Vous pouvez créer un paquetage de données à partir de la base de données ou lui relier un paquetage existant. Le paquetage permet de représenter la structure de la base de données, les classes et les parts qu'elles contient.

Le paquetage de la base de données est le détenteur par défaut des objets représentés dans le diagramme de classes. Néanmoins, il est possible d'utiliser des objets détenus dans d'autres paquetages.

> ☛ *De la même façon, vous pouvez relier un modèle de données à une base de données, lorsque vous avez sélectionné le formalisme correspondant. Voir "Les formalismes", page 2.*

Pour créer un paquetage à partir d'une base de données :

- Cliquez sur l'icône de la base de données et sélectionnez **Nouveau** > **Paquetage**. Cette commande crée un nouveau paquetage ainsi que le diagramme de classes qui lui est associé.

Pour relier un paquetage à une base de données :

1. Cliquez sur l'icône de la base de données et sélectionnez **Relier** > **Paquetage de données**.
   La fenêtre de recherche apparaît.
2. Cliquez sur **Chercher**.
3. Sélectionnez le paquetage voulu et cliquez sur **Relier**.

Vous pouvez voir le nom des paquetages associés à une base de données dans les propriétés de la base de données, sous la page **Caractéristiques**.

Importer une version de SGBD

Lors de la création d'un nouveau référentiel, seule la version de SGBD **SQL ANSI/ ISO 9075:1992** est installée. Si vous choisissez un autre SGBD cible pour une base de données, il faut importer le solution pack des datatypes SQL qui est compressé dans le fichier **SGBD SQL Type.exe**.

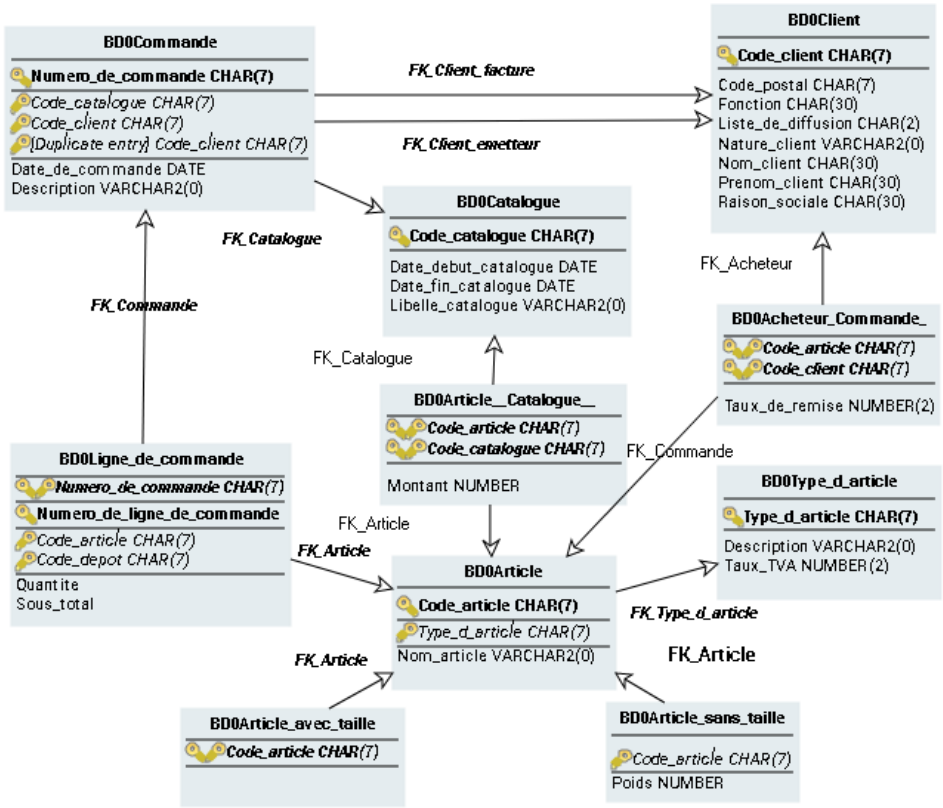
Le fichier **SGBD SQL Type.exe** est disponible dans le dossier Utilities\Solution Pack de votre répertoire d'installation HOPEX.

Une fois le solution pack importé dans **HOPEX Information Architecture**, vous pouvez sélectionner le SGBD en question dans la fenêtre de propriétés de la base de données.

Voir aussi "Paramétrer la synchronisation", page 37.

LE DIAGRAMME RELATIONNEL

Un diagramme relationnel (DR) décrit une base de données : il représente les structures des données physiques exploitées par les programmes.



La description dans **HOPEX Information Architecture** des diagrammes relationnels permet d'envisager l'utilisation d'une interface avec le SGBD retenu, garantissant ainsi la cohérence sémantique des données de conception avec celles qui sont exploitées.

Construire le diagramme relationnel

Généralement, le diagramme relationnel est construit en deux phases :

1. La synchronisation automatisée du ou des diagrammes de données permet d'obtenir le diagramme "brut".
   Voir "Synchroniser les modèles logiques et physiques", page 1.

2. L'optimisation du diagramme, aussi appelée dénormalisation, permet de prendre en compte les besoins d'accès aux données de l'applicatif et d'optimiser les performances de la base de données.
Voir "Dénormaliser les modèles logiques et physiques", page 79.

Le concept central d'un diagramme relationnel est la table, issue d'une entité ou d'une association.

> 🕮 *La table est une structure logique de données, utilisée comme référence pour le passage en réalisation ; c'est l'élément central de la base de données. Une table est accessible par une clé primaire, et éventuellement des clés étrangères ; elle est décrite par une séquence ordonnée de colonnes. Une table est généralement issue d'une entité ou d'une association.*

Une *table* est accessible par une ou plusieurs clés, dont le type indique si elles sont primaires ("primary key") ou étrangères ("foreign key"). Il est possible de définir les index d'une table, en précisant s'ils sont uniques et leur sens de tri (ascendant ou descendant). Les clés et les index sont reliés aux colonnes qui les constituent.

Créer des objets dans le diagramme

Pour créer une clé ou un index dans le diagramme relationnel:

1. Cliquez sur la table concernée.
La liste des commandes associées à la table apparaît.

2. Cliquez sur le bouton **Clé** ou le bouton **Index**

> ☛ *Assurez-vous que les colonnes sur lesquelles doivent porter cette clé ou cet index existent déjà dans la table.*

Vous pouvez également utiliser la page **Composants** de la fenêtre de propriétés de la base de données pour créer ces objets. Voir "Créer manuellement une clé", page 20 et "Créer un index", page 21.

Pour créer une clé étrangère :

1. Cliquez sur le bouton **Clé** , placez le pointeur sur la première table puis, en maintenant le bouton enfoncé, faites glisser le pointeur vers la seconde table.
Une fenêtre de création apparaît.
2. Indiquez le nom de la clé et cliquez sur **Ajouter**.
Une seconde fenêtre demande si vous voulez créer automatiquement les colonnes de clé étrangères à partir de celles de la clé primaire.
3. Cliquez sur **Oui** pour valider ou **Non** pour créer

Paramétrer l'affichage des diagrammes relationnels

De même que pour les diagrammes de données, il est possible de préciser les éléments qui doivent apparaître dans le diagramme :

- Soit par le bouton **Vues et détails** qui indique globalement les types d'objets présentés dans le diagramme.
- Soit par les options d'affichage des éléments, qui permettent de définir quelles caractéristiques des objets doivent être présentées.

Pour paramétrer l'affichage d'un seul objet :

- Faites un clic droit sur l'objet et sélectionnez **Formes et détails**.

  ☛ *Lorsque l'affichage concerne un objet, la fenêtre **Affichage** présente d'abord les formes qu'il est possible d'utiliser pour la présentation de l'objet. La sélection d'un élément dans l'arborescence provoque l'affichage de son contenu.*

DÉFINIR LES COMPOSANTS D'UNE BASE DE DONNÉES

Une base de données est un ensemble de données organisé en vue de son utilisation par des programmes correspondant à des applications distinctes et de manière à faciliter l'évolution indépendante des données et des programmes.

Une base de données est constituée de tables, de colonnes, de clés et d'index :

- La ***table*** est l'unité logique de stockage des colonnes.
- Une ***colonne*** est contenue dans une table.
- De même qu'un identifiant "estampille" une entité, la ***clé primaire*** de la table "estampille" une ligne de table.
- Une ***clé étrangère*** permet l'accès à une autre table, et le contrôle de cohérence des colonnes entre les tables concernées.
- Un ***index*** permet d'accélérer l'accès aux données ; il peut être unique ou non, ascendant ou descendant.

Les tables d'une base de données

Il est possible de consulter et de mettre à jour les tables d'une base de données de deux façons :

- Dans son diagramme relationnel, c'est-à-dire le diagramme des tables de la base de données.
- Dans ses propriétés, sous la page **Composants**.
  La page des composants affichent les tables de la base de données.
  Le nom du **Groupement de données** associé est indiqué, le cas échéant.

Créer une table

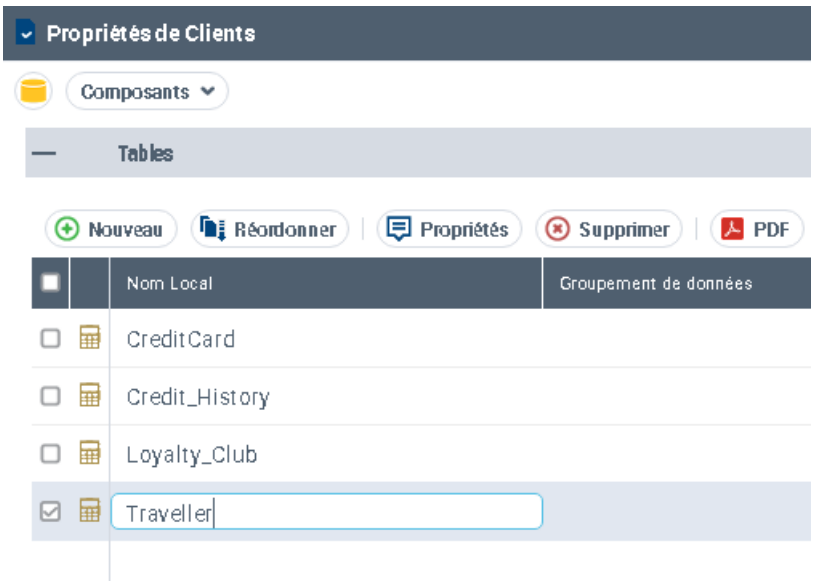
Voir préalablement : "Propriétés d'une base de données", page 11.

Pour créer une table à partir des propriétés de la base de données :

1. Ouvrez la fenêtre de propriétés de la base de données
2. Cliquez sur la liste déroulante puis sur **Composants**.
3. Dans la section **Tables**, cliquez sur le bouton **Nouveau**.
   La nouvelle table apparaît dans la fenêtre. Elle se nomme par défaut "Table1".

Pour la renommer :

- Double-cliquez sur le nom de la table et saisissez le nouveau nom.



Supprimer une table

Pour supprimer une table, avec ses colonnes, clés et index :

- Faites un clic droit sur la table et cliquez sur **Supprimer** 
  Un message demande confirmation de la suppression.

  ☛ *Une table supprimée ne sera pas recréée automatiquement lors d'une nouvelle synchronisation. Pour recréer la table, lors de l'étape de validation des résultats de la synchronisation, validez l'action de création proposée pour cette table (cochez la case correspondante).*
  Voir "Etape 4 : valider les résultats", page 14.

Les colonnes d'une table

Visualiser les colonnes

Voir préalablement : "Propriétés d'une base de données", page 11.

Pour consulter les colonnes d'une table :

1. Ouvrez la fenêtre de propriétés de la base de données.

2. Cliquez sur la liste déroulante puis sur **Composants**.
   La page des composants affiche la section **Colonnes**.
   Pour chaque colonne sont présentés :
   - Son **Nom local**
   - Son type (**Datatype**)

     ☛ *L'administrateur peut compléter la liste des datatypes (voir "Créer de nouveaux datatypes", page 113).*
   - Éventuellement sa longueur (**Lng**) et son nombre de décimales (**Déc**)
   - La valeur de l'attribut **Not null**
   - Sa **valeur par défaut** : lors de la génération de la table, la valeur prise par défaut est celle de l'attribut dont elle découle. Si aucune valeur initiale n'est renseignée au niveau de l'attribut ou si vous voulez modifier la valeur d'une colonne, entrez une valeur dans ce champ.
   - Le fait que la colonne soit reliée ou non à une clé primaire (PK) ou étrangère (FK). Cela est indiqué par **Y** ("Oui") ou **N** ("Non").

Il est possible de modifier le **Nom Local** d'une colonne en cliquant sur son nom et en saisissant le nouveau nom. Ce nom local sera utilisé dans le script généré pour la table.

Il est possible d'indiquer un **Nom SQL** directement dans la page **SQL** de la fenêtre de propriétés d'un attribut dans le diagramme de données. De cette façon, toutes les colonnes créées à partir de cet attribut auront pour base le même nom local. De plus ce nom sera réutilisé lors des synchronisations successives, y compris lors d'une réinitialisation totale ou partielle.

Il est également possible de modifier la valeur des autres caractéristiques de la colonne.

☛ *Ces modifications seront conservées lors des synchronisations ultérieures.*

Il est possible de créer des colonnes qui ne proviennent pas d'attributs du diagramme de données, que ce soit dans une table générée ou créée par l'utilisateur.

Créer une colonne

Voir préalablement : "Propriétés d'une base de données", page 11.

Pour créer une colonne :

1. Ouvrez les propriétés de la table concernée.
2. Sous la page **Composants**, dans la section **Colonnes**, cliquez sur le bouton **Nouveau**.

   ☛ *Lorsque la création d'une colonne n'est pas effectuée à partir des **Propriétés** d'une table, par exemple avec l'explorateur, il est nécessaire de sélectionner au préalable la table qui la contiendra, sinon un message indique que la création est impossible.*

Supprimer une colonne

Pour supprimer une colonne :

➤ Faites un clic droit sur la colonne et sélectionnez **Supprimer**.

☛ *Une colonne supprimée ne sera pas recréée automatiquement lors d'une nouvelle synchronisation. Pour recréer la colonne, lors de l'étape*

de validation des résultats de la synchronisation, validez l'action de création proprosée pour cette colonne (cochez la case correspondante).
Voir "Etape 4 : valider les résultats", page 14.

Le bouton **Réordonner** permet d'accéder à la fenêtre de **Modification de l'ordre**.

Modifier les clés et les index

La synchronisation permet de créer de façon automatique les clés primaires et étrangères, ainsi que les index sur ces clés.

Lorsque ces créations sont demandées :

- Les clés primaires portent sur les colonnes correspondant aux identifiants.
- Les clés étrangères portent sur les colonnes qui migrent dans les tables à cause d'une association contrainte.

Un index est créé sur chaque clé.

Il est possible de compléter, modifier ou supprimer les clés et index proposés lors de la synchronisation. Pour y accéder :

1. Faites un clic droit sur la table concernée et sélectionnez **Propriétés**.
2. Cliquez sur la liste déroulante puis sur **Composants**.

Dans la section **Clés** sont indiqués :

- Le type de la clé (**Type-Clé**) : "Etrangère" ou "Primaire".
- Dans le cas d'une clé étrangère :
  - La table référencée.
  - La gestion de l'intégrité référentielle en mise à jour (**On Update**) et en suppression (**On Delete**) ; référez-vous à la documentation du SGBD cible pour le type d'ordres gérés.

> ☛ *Lorsqu'une colonne dite "migrante" est créée dans une table pour prendre en compte une association contrainte, il est possible de demander que le SGBD contrôle la valeur mise à jour dans cette colonne. Le SGBD vérifie alors que cette valeur existe toujours dans la table d'origine (intégrité référentielle).*

Lors d'une mise à jour (**On Update**) ou d'une suppression (**On Delete**) dans la table d'origine, le SGBD peut :

- Mettre à jour les valeurs dans les tables concernées, avec l'option **Cascade**.
- Ne rien faire, avec l'option **No Action**.
- Interdire la mise à jour ou la suppression, avec l'option **Restrict**.
- Remettre la valeur par défaut dans les tables concernées, avec l'option **Set Default**.
- Remettre la valeur à **Null** dans les tables concernées, avec l'option **Set Null**.

Dans la section **Index** sont indiqués :

- Son **Type** : Bitmap, Standard, Unique, Unique where not null.
- Son **Sens de Tri** (**Ascendant** ou **Descendant**).
- S'il s'agit d'un index groupé **(Clustered)**.

> ☛ *La création d'une colonne à partir d'une clé ou d'un index n'est pas possible. Il faut d'abord créer la colonne dans la table, puis la relier à la clé ou à l'index.*

Créer manuellement une clé

Voir préalablement : "Les tables d'une base de données", page 16.

Pour créer une clé :

1. Faites un clic droit sur la table concernée et sélectionnez **Propriétés**.
2. Cliquez sur la liste déroulante puis sur **Composants**.
3. Sous la section **Clés**, cliquez sur le bouton **Nouveau.**
   La fenêtre de création d'une clé apparaît.
4. Sélectionnez le type de clé à créer ;"étrangère" ou "primaire". La création d'une clé varie selon le type indiqué.

Clé primaire

Lorsque vous sélectionnez le type "Primaire", la clé apparaît dans les propriétés de la table.

Pour définir les propriétés de la clé :

- Faites un clic droit sur la clé et sélectionnez **Propriétés**.
  Dans la page **Colonnes** vous pouvez spécifier les colonnes sur lesquelles porte la clé.

Il est également possible de spécifier la clé primaire d'une table dans la section **Identifiants** de la fenêtre de propriétés de l'entité dont elle est issue. Voir "Définir l'identifiant d'une entité", page 45.

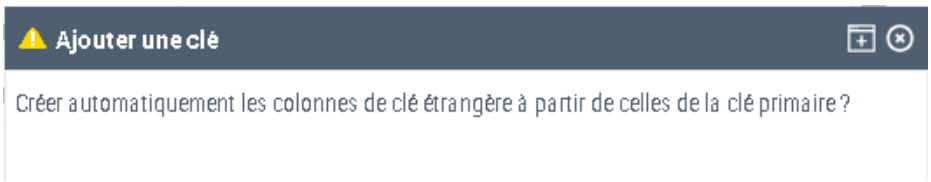
Vous pouvez rendre l'identifiant "explicite" et choisir les attributs de l'entité - voire les attributs d'une autre entité (reliée par une association contrainte) - qui constituent l'ID.

La clé ainsi spécifiée est créée dans la table lors de la synchronisation.

Clé étrangère

Lorsque la clé créée est une clé étrangère, une liste des tables de la base de données est présentée.

1. Sélectionnez la table de référence sur laquelle porte la clé étrangère.
   Si la table que vous choisissez comporte une clé primaire, un message apparaît.



2. Sélectionnez **Oui**.
   la clé apparaît dans les propriétés de la table.

Vous pouvez modifier le **Nom local** de la clé (le nom complet d'une clé est composé du nom de la base de données à laquelle elle appartient, suivi du nom de la table, puis de son nom local : par exemple, "BD Bourse::Concerner::Cle1").

Pour une clé étrangère, comme lors de l'édition d'une clé, il est possible de préciser la gestion de l'intégrité référentielle en mise à jour (**On Update**) et en suppression (**On Delete**).

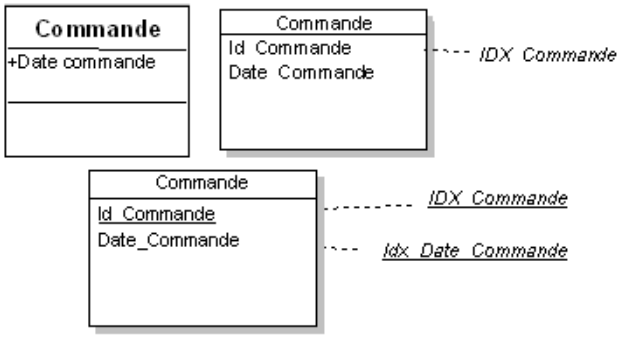
Créer un index

Voir préalablement : "Les tables d'une base de données", page 16.

Des index sont créés automatiquement sur les clés primaires et étrangères. Il est possible d'ajouter, aux index générés, des colonnes utilisées fréquemment comme critères de recherche.

☺ *Il est également possible de spécifier un index dans la page **Identifiants** de la fenêtre de propriétés de l'entité dont la table est issue. L'index ainsi spécifié sera créé dans la table lors de la synchronisation.*

Exemples d'index :



Pour créer un index :

1. Faites un clic droit sur la table concernée et sélectionnez **Propriétés**.
   La fenêtre de propriétés apparaît.
2. Cliquez sur la liste déroulante puis sur **Composants**.
3. Sous la section **Index**, cliquez sur le bouton **Nouveau**.
   La fenêtre **Création d'un index** est présentée.
4. Saisissez le **Nom local** et cliquez sur **OK**.

En fonction des possibilités offertes par le SGBD utilisé, il est possible de préciser le **Type** de l'index, le **Sens de tri** de l'index ("Ascendant" ou "Descendant"), et s'il s'agit d'un index groupé (**Clustered**).

Il est alors possible de sélectionner les colonnes de la clé (de l'index) dans la page **Colonnes** de la fenêtre de propriétés de l'index.

Ajouter une colonne à une clé ou un index

Voir préalablement : "Les tables d'une base de données", page 16.

Pour ajouter une colonne à une clé (ou à un index) :

1. Faites un clic droit sur la table concernée et sélectionnez **Propriétés**.
   La fenêtre de propriétés de la table apparaît.
2. Cliquez sur la liste déroulante puis sur **Composants**.
3. Sous la section **Index**, faites un clic droit sur l'index et sélectionnez **Propriétés**.
   La fenêtre de propriétés de l'index apparaît.
4. Cliquez sur la liste déroulante puis sur **Colonnes**.
5. Cliquez sur le bouton **Relier**.
   La fenêtre de recherche apparaît.
6. Recherchez et sélectionnez la colonne à ajouter à la clé (ou à l'index).

Il est possible d'indiquer le sens de tri de la clé ou de l'index, qui peut être "Ascendant" ou "Descendant".

SPÉCIFIER LES CLÉS PRIMAIRES ET ÉTRANGÈRES

Lorsque les clés d'une base de données ne sont pas complètement spécifiées, il est nécessaire de les **Compléter**.

Spécifier les clés primaires

Pour spécifier les clés primaires d'une base de données :

- Faites un clic droit sur la base de données et sélectionnez **Compléter les clés des tables**.
  La fenêtre **Compléter les clés** apparaît.

  ☛ *Lorsque la base est complètement spécifiée, la fenêtre présente une liste vide : aucun complément de spécification n'est nécessaire.*

La liste **Proposition** permet d'indiquer le critère utilisé pour compléter les clés primaires :

- **A partir des index uniques** : les colonnes qui appartiennent à un index unique sont proposées comme composantes de la clé primaire.
- **A partir des colonnes obligatoires** : ces colonnes sont proposées comme composantes d'une clé.
- **Par rapprochement des noms** : si le même nom de colonne est retrouvé dans plusieurs tables, la colonne est proposée comme clé primaire.

Chaque clé est proposée sous la table à laquelle elle appartient.

Pour valider une clé primaire :

1. Cochez la case de la colonne **Périmètre** qui correspond à la clé.
   Les colonnes associées sont automatiquement cochées par défaut. Vous pouvez éliminer celles qui répondent aux critères de recherche mais ne sont pas des constituants de la clé.
2. Cliquez sur le bouton **Appliquer**.
   Le bouton **Appliquer** retire de la liste des propositions les clés explicitement acceptées ou refusées.

Pour les clés étrangères, deux clés incluant une même colonne sur une même table sont incompatibles : l'acceptation de l'une provoque automatiquement le rejet de l'autre.

Il n'est pas possible de sélectionner plusieurs clés primaires sur une même table : l'acceptation d'une clé provoque le rejet des autres.

☺ *Il vous est possible de compléter la spécification des clés en plusieurs passages. Cela vous permet de consulter le contenu de la base pendant que vous faites vos choix. Pour cela :*

- *Cliquez sur le bouton **Appliquer** pour enregistrer vos modifications.*
- *Cliquez sur le bouton **Annuler** pour quitter cette fenêtre sans lancer de traitement.*

Spécifier les clés étrangères

Pour spécifier les clés étrangères d'une base de données :

- Faites un clic droit sur la base de données et sélectionnez **Compléter les clés des tables**.
  La fenêtre de **Compléter les clés** apparaît.

  ☛ *Lorsque la base est complètement spécifiée, la fenêtre présente une liste vide : aucun complément de spécification n'est nécessaire.*

La liste **Proposition** permet d'indiquer le critère utilisé pour compléter les clés étrangères :

- **A partir des index**
- **Par rapprochement des noms**

Si la proposition est faite à partir des index, la proposition se base sur les index non uniques de la table. La table de référence est indiquée après le nom de la clé.

Pour valider une clé étrangère :

1. Cochez la case de la colonne **Périmètre** qui correspond à la clé.
2. Cliquez sur le bouton **Appliquer**.
   Le bouton **Appliquer** retire de la liste des propositions les clés explicitement acceptées ou refusées.

Lorsqu'aucune table de référence n'est définie, l'assistant en propose automatiquement. Les clés qui n'ont pas de table de référence ne peuvent être acceptées.

Lors de la proposition de clés, on peut trouver plusieurs tables possédant une clé primaire identique. Ce pourrait être le cas par exemple, pour les tables correspondant aux différents sous-types d'une même entité.

RÈGLES DE MODÉLISATION D'UNE BASE DE DONNÉES

HOPEX Information Architecture fournit un ensemble de règles qui permettent de contrôler la modélisation des bases de données. Le règlement physique contient les règles relatives au schéma relationnel d'une base de données. Il est utilisé pour vérifier le schéma relationnel correspondant dans un SGBD.

Le règlement physique contient les règles relatives aux spécifications techniques propres au SGBD de la base de données. Il est utilisé pour vérifier la cohérence des paramètres physiques du schéma relationnel spécifique au SGBD.

Contrôler une base de donnée

Vous pouvez lancer un contrôle sur la base de données ou sur un objet de la base de données.

Pour contrôler une base de données :

1. Faites un clic droit sur le nom de la base de données.
2. Sélectionnez **Administrer** > **Contrôler** > **Règlement avec Propagation**.
   Lorsque plusieurs règlements peuvent s'appliquer à l'objet contrôlé, une fenêtre vous demande de sélectionner le règlement voulu.

Le contrôle s'applique à la base de données ainsi qu'aux objets qu'elle détient.

Les résultats apparaissent dans un rapport HTML.

Pour plus de détails sur les contrôles, voir le guide **HOPEX Common Features**, "Explorer le référentiel", "Les outils de contrôle des objets".

SYNCHRONISER LES MODÈLES LOGIQUES ET PHYSIQUES

La synchronisation est le traitement qui permet de traduire un diagramme de classes exprimé avec le formalisme classes/parties en un modèle physique exprimé avec le formalisme relationnel, et inversement. Elle met ainsi en correspondance les objets des deux modèles.

Une option de compatibilité vous permet également de mettre en correspondance un modèle de données (entités/associations) et un modèle physique. Voir "Formalisme logique et synchronisation", page 2.

Ce processus est à réaliser à certaines périodes. En effet, tout au long du projet de modélisation, ces modèles subissent des changements chacun de leur côté. La synchronisation intervient lorsque vous voulez confronter les deux modèles et rétablir de façon automatique les correspondances canoniques qui les unissent.

La fonctionnalité de synchronisation est disponible avec l'accès au référentiel **HOPEX** en mode "Avancé".

La synchronisation de modèles d'une base de données peut s'effectuer dans un sens ou dans l'autre - soit dans le sens physique > logique soit dans le sens logique > physique - mais pas les deux à la fois. Une fois le sens de la synchronisation déterminé, il convient de ne pas inverser la synchronisation.

Voir aussi "Correspondance des modèles", page 69.

Options d’affichage de la synchronisation

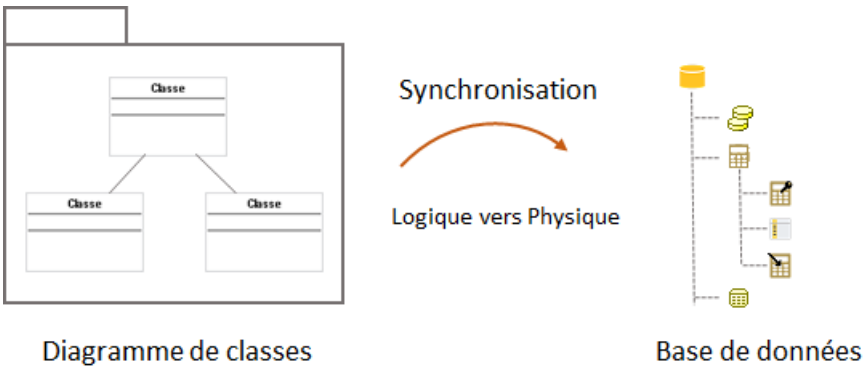
Certaines options de synchronisation sont filtrées par défaut. Pour les afficher :

1. Cliquez sur **Menu principal** > **Paramètres** > **Options**.
2. Dans l'arborescence **Options** du volet de gauche, dépliez le dossier **Modélisation de données**, et cliquez sur le sous-dossier **Synchronisation de base de données**.
3. Dans le volet de droite, activez les options de synchronisation voulues :
   - synchronisation logique > physique
   - synchronisation physique > logique
   - synchronisation réduite logique > physique
   - synchronisation réduite physique > logique

Vous pouvez voir que le formalisme logique appliqué par défaut dans la synchronisation est la notation UML. Voir "Les formalismes", page 2.

RÈGLES DE SYNCHRONISATION "LOGIQUE VERS PHYSIQUE"

Les règles qui suivent sont utilisées pour la transformation des diagrammes de classes et modèles de données en formalisme relationnel.



☛ *Voir également "Paramétrer la génération des noms", page 36 et "Types des attributs et des colonnes", page 99.*

Synchronisation logique > physique : les domaines de données applicatifs et logiques

En mode logique > physique, la synchronisation des domaines de données applicatifs et logiques donnent lieu à des domaines de données relationnelles.

Voir aussi "Les domaines de données physiques", page 9.

Synchronisation logique > physique : les Entités (ou Classes)

En mode logique > physique, les classes et les entités sont traitées de la même façon dans l'outil de synchronisation.

Par défaut la synchronisation applique le formalisme logique du diagramme de classes. Voir "Formalisme logique et synchronisation", page 2.

Règle générale

- Toute entité non abstraite du modèle devient une table.
- L'identifiant de l'entité devient la clé primaire de la table. Si l'identifiant est implicite, une colonne est automatiquement créée. Voir "Paramétrer

la génération des noms", page 36.

- Les attributs de l'entité deviennent des colonnes de la table.
- Des règles de correspondance permettent de déduire les datatypes des colonnes à partir du type de données (MD) de chaque attribut. Les paramétrages proposés dépendent du SGBD.

> ☛ *Pour plus d'informations à ce sujet, voir "Types des attributs et des colonnes", page 99.*

Sous-entité

- La clé étrangère issue de la dépendance entre la sous-entité et sa super-entité est créée.

Entité abstraite

Une entité abstraite ne donne lieu à aucune table lors de la synchronisation.

Si des associations contraintes pointent vers une entité abstraite, les clés étrangères correspondantes ne sont pas créées, mais les colonnes correspondant aux clés étrangères sont créées pour respecter l'intégrité de la table.

Lorsqu'une sous-entité est abstraite, toutes les colonnes et clés étrangères de la table correspondante sont prises en charge par la table correspondant à la super entité.

Inversement, lorsqu'une une super-entité est abstraite, toutes les colonnes et clés étrangères de la table correspondante sont prises en charge par la table correspondant à la sous-entité.

Pour définir une entité abstraite :

1. Ouvrez la fenêtre de propriétés de l'entité.
2. Cliquez sur l'onglet **Caractéristiques**.
3. Dans le champ **Abstraite**, sélectionnez "Oui".

Entité réalisée

On dit d'une entité qu'elle est réalisée si elle donne lieu à la création d'une table lors d'une synchronisation.

Une entité "non réalisée" est traitée comme une entité abstraite.

Contrairement à la propriété "abstraite" qui caractérise l'entité dans tous ses cas d'emploi, le concept "Réalisé" s'applique uniquement dans le cadre de la synchronisation d'une base de données. Voir "Mode Réalisé", page 16.

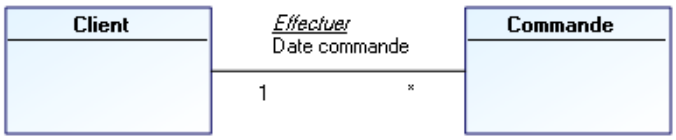
Synchronisation logique > physique : les Associations

La synchronisation des associations est disponible avec l'ancien formalisme UML qui prend en compte les associations et non les parties. Voir "Options de modélisation des données", page 2.

Association contrainte (multiplicités : 0,1 ou 1,1)

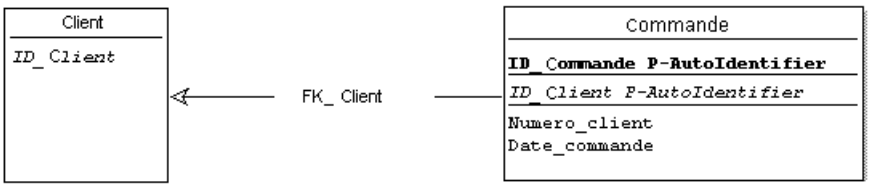
Une association contrainte est une association binaire dont la multiplicité maximum d'un de ses rôles est 1. Dans ce cas, il n'est pas nécessaire de créer une table correspondant à cette association. Une colonne rajoutée dans la table correspondant à l'entité suffit.

Une association contrainte (une de ses multiplicités maximum est à 1) ne donne pas lieu à une table. Dans l'exemple suivant, une commande comporte un client et un seul.



La synchronisation de ce diagramme de données donne l'un des deux résultats suivants :

L'association ne donne pas lieu à une table



- Une colonne correspondant à la clé de l'entité "Client" est créée dans la table "Commande".
- Une colonne est également créée pour chaque attribut de l'association.
- Une clé étrangère "FK_Client" est ajoutée pour assurer le contrôle de la colonne "ID_Client" de la table "Commande". Elle indique que les valeurs possibles pour la colonne "Numéro Client" de la table "Commande" sont celles qui existent dans la colonne "Numéro Client" de la table "Client".

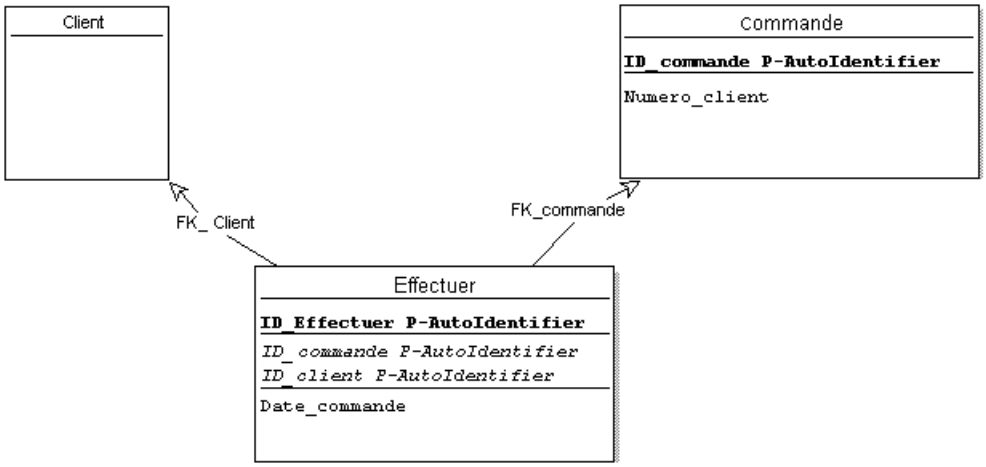
La clé étrangère ("foreign key") est créée à partir de l'identifiant de l'entité.

L'association est transformée en table

Pour que l'association se transforme en table :

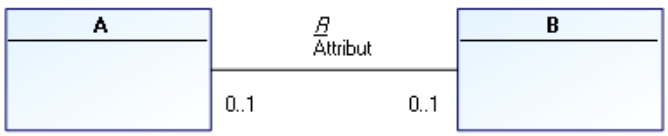
1. Ouvrez la fenêtre de propriétés de l'association.
2. Cliquez sur l'onglet **SQL**.

3. Dans le champ **Correspondance potentielle**, sélectionnez la valeur "Table".



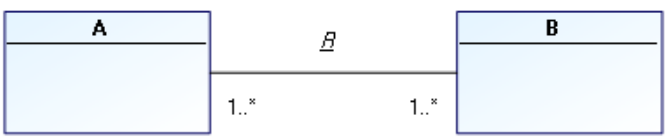
Association contrainte (multiplicités : 0,1 et 0,1)

Dans ce cas particulier, la combinaison des multiplicités est ambiguë. Il n'y a pas d'élément permettant de choisir dans quelle table on va créer la colonne correspondant à l'attribut.



La synchronisation propose une colonne dans chaque table.

Verrous croisés (dead locks)



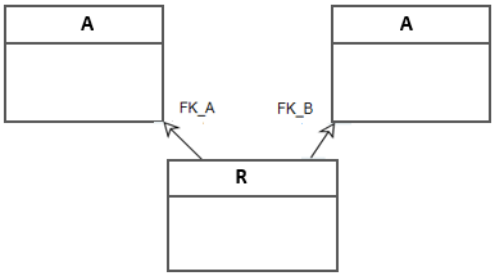
Les multiplicités 1..x, 1..x signifient que chacun des deux objets doit être relié à au moins un objet de l'autre type pour pouvoir exister.

Ceci pose un problème pour créer les premiers objets de chaque type. En effet :

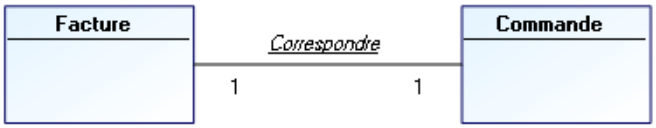
- Un objet de type A doit exister pour pouvoir créer un objet de type B et le lui relier.
- Réciproquement, un objet de type B doit exister pour pouvoir créer un objet de type A et le lui relier.

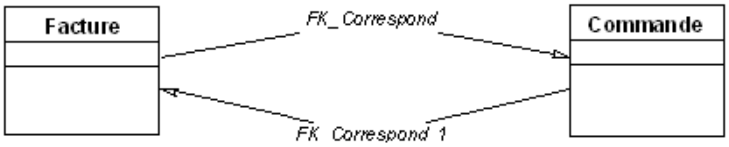
C'est également le cas pour le couple de multiplicités 1, 1..*

Ces cas n'amènent cependant pas de verrou croisé car il est possible de créer les premiers objets de chaque type.



Les multiplicités 1, 1 génèrent plusieurs clés étrangères obligatoires croisées :





On est alors dans une situation de blocage total, car on devrait, pour respecter les contraintes, créer plusieurs tables à la fois.

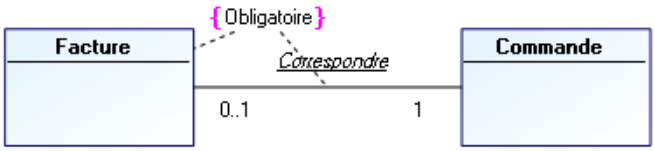
Certains SGBD interdisent complètement la création de tables de ce type.

Pour permettre une synchronisation correcte, il vous faut éviter les situations de ce genre.

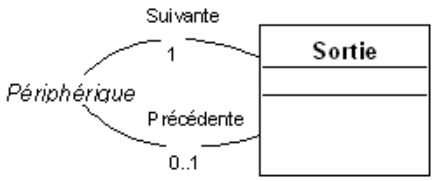
> **Choisissez une des clés étrangères pour laquelle vous mettez la multiplicité 1 et sur l'autre, mettez 0..1. Vous pouvez également retirer la clé étrangère de la table après la synchronisation, mais c'est moins pratique.**

Pour ne pas perdre l'information concernant la multiplicité minimum à 1, vous pouvez ajouter une contrainte comme celle présentée dans le diagramme qui suit. Cette contrainte ne sera pas prise en compte dans la synchronisation.

☛ *Voir "Les contraintes", page 40 pour plus de détails.*

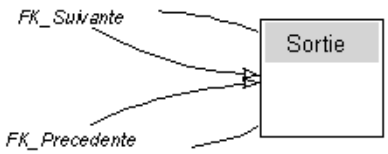


Voici un autre exemple dans le cas d'une nomenclature : Nomenclature 1, 1 ou 0..1, 1.



On veut exprimer que chaque sortie du boulevard périphérique précède une sortie et en suit une autre.

Cette modélisation oblige à créer toutes les sorties à la fois, puisque pour chaque sortie, on doit avoir créé la précédente.



Pour éviter cela, il faut mettre les deux multiplicités à 0..1 et 0..1.

Association non-contrainte

Une association dont les multiplicités maximum sont différentes de 1 donne lieu à une table :

- Une colonne est créée pour chacun des attributs des identifiants des entités reliées.
- La clé primaire de la table porte sur l'ensemble de ces colonnes.
- Une clé étrangère est également constituée pour chaque entité reliée.
- Une colonne supplémentaire est créée pour chaque attribut de l'association.

Avant de lancer la synchronisation, il est souhaitable de contrôler la validité du

diagramme de données, et de contrôler que le paramétrage de la synchronisation est correct. Voir "Règles de modélisation de données", page 51 et "Préparer la synchronisation", page 34.

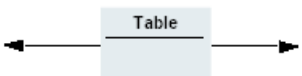
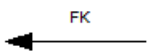
Classe associative

Les associations reliant des classes associatives ne sont pas prises en compte dans la synchronisation.

Synchronisation logique > physique : les Parties (Formalisme UML)

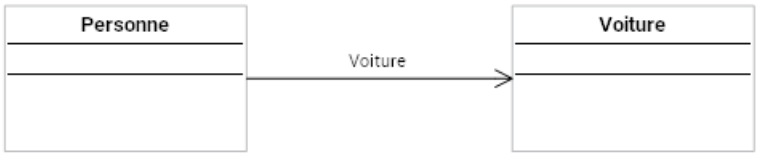
La synchronisation des parties est disponible par défaut avec le nouveau formalisme UML.

Le résultat de la synchronisation est fonction de la combinaison du lien **Tout/partie** (Non renseigné, Agrégation, Composition) et de la **Multiplicité** définis sur la partie.

Multiplicité	**Lien Tout/Partie**	
	Agrégation  **Composition** 	**Non renseigné**
Non renseigné (*) 2..6 1..*	La partie donne lieu à une clé étrangère vers la classe détentrice 	La partie donne lieu à une table entre les deux classes 
1 0..1	La partie donne lieu à une clé étrangère vers la classe référencée  et donne lieu à une clé étrangère vers la classe détentrice 	La partie donne lieu à une clé étrangère vers la classe référencée 

Exemple 1 : Non renseigné / *

Dans l'exemple suivant, la classe "Personne" référence la classe "Voiture", sans contrainte de multiplicité.

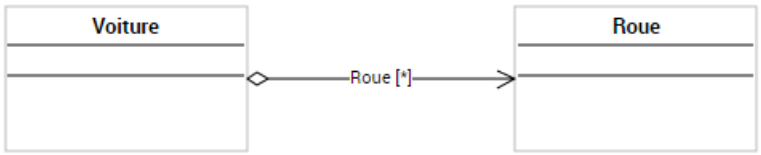


Après synchronisation, la partie "Voiture" donne lieu à une table :

- Une colonne est créée pour chacun des attributs des identifiants des entités reliées.
- La clé primaire de la table porte sur l'ensemble de ces colonnes.
- Une clé étrangère est également constituée pour chaque entité reliée.

Exemple 2 : Agrégation / *

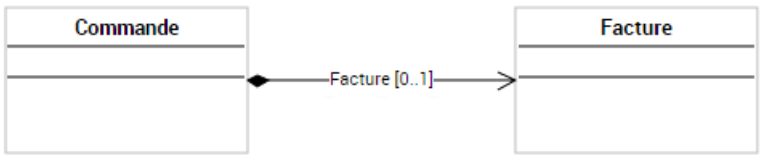
Une voiture peut avoir une ou plusieurs roues.



Après synchronisation, la partie "Roue" donne lieu à une clé étrangère vers la table "Voiture".

Exemple 3 : Composition / 0..1

Une commande détient une facture.



Après synchronisation :

- une clé étrangère référence la table "Facture" dans la table "Commande"
- une clé étrangère référence la table "Commande dans la table "Facture".

DU MODÈLE LOGIQUE AU MODÈLE PHYSIQUE

Cette partie présente comment synchroniser le modèle logique d'une base de données (représenté par un diagramme de données) avec le modèle physique (relationnel) correspondant.

Bien que la synchronisation du modèle relationnel à partir du diagramme de données concerne essentiellement des entités, elle peut également se faire à partir des classes d'un diagramme de classes.

La synchronisation dans l'autre sens, d'un modèle physique vers un modèle logique est également possible mais pas sur une même base de données. Une fois le sens de la synchronisation choisie pour un objet, la synchronisation dans l'autre sens n'est plus possible.

Voir "Du modèle physique au modèle logique", page 28.

Les points qui suivent présentent la synchronisation d'une base de données. Vous pouvez également lancer une synchronisation réduite, autrement dit sur un objet particulier de la base de données. Voir "Synchronisation réduite (mode logique vers physique)", page 19.

Lancer la synchronisation

La synchronisation Logique vers Physique consiste à construire le modèle physique à partir du modèle logique, autrement dit à créer les tables et colonnes correspondant aux entités et attributs du diagramme de données.

L'outil de synchronisation est disponible dans le volet de navigation **Données logiques**. Vous pouvez également ouvrir l'outil de synchronisation directement à partir de la base de données concernée.

Pour lancer une synchronisation Logique vers Physique sur une base de données :

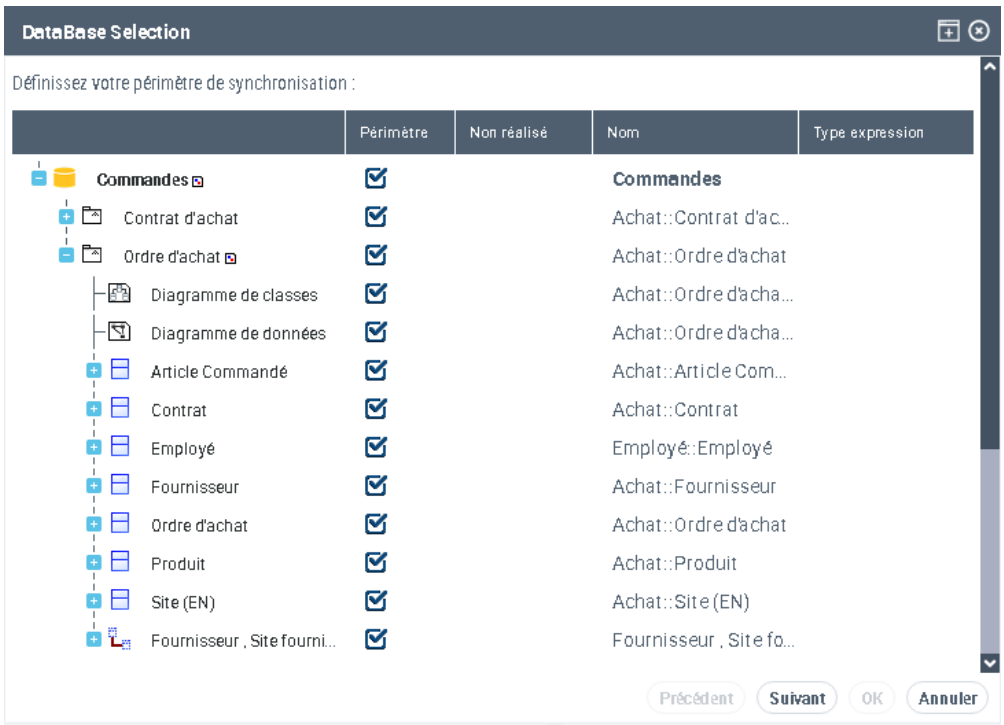
1. Cliquez sur le menu de navigation puis sur **Données logiques.**
2. Dans le volet de navigation cliquez sur **Toutes les bases de données**. La liste des bases de données du référentiel apparaît dans la fenêtre d'édition.
3. Faites un clic droit sur la base de données concernée et sélectionnez **Synchronisation (logique vers physique)**. L'assistant de synchronisation apparaît.

Etape 1 : sélectionner les objets sources à synchroniser

Pour définir le périmètre de la synchronisation :

1. Dans l'arbre de la vue logique, déroulez la liste des objets contenus dans la base de données.

2. Par défaut, tous les objets sont cochés et donc inclus dans la synchronisation. Pour exclure un objet de la synchronisation, décochez-le dans la colonne **Périmètre**. Lorsqu'un objet est exclu, sa correspondance l'est aussi.



3. Par défaut, tous les objets sont "réalisés", autrement dit donnent lieu à la création d'un objet lors de la synchronisation. Pour indiquer qu'un objet est "non réalisé", cochez-le dans la colonne **Non réalisé**. Pour plus d'informations voir "Mode Réalisé", page 16.
4. Une fois la liste des objets définie, cliquez sur le lien **Suivant** de l'assistant.

Etape 2 : options de synchronisation

Parmi les options de synchronisation, vous pouvez :

- Réinitialiser les objets cibles : dans le cas où les objets de la vue logique ont déjà été synchronisés, la synchronisation repart de zéro et supprime les objets cibles existants.

> **Lorsque les modèles ont déjà été synchronisés, pour que les correspondances établies soient prises en compte lors d'une**

nouvelle synchronisation, veillez à ce que l'option "Réinitialisation des objets cibles" soit décochée.

- Recalculer le nom des objets cibles : les noms des objets physiques sont recalculés en fonction de ceux des objets sources. Cela signifie que toute modification manuelle du nom des objets physiques est annulée.
- Prendre en compte les optimisations : toutes les optimisations - dont celles qui ne sont pas cochées dans l'étape de validation (voir étape 4) - sont proposées.
- Prendre en compte les suppressions : les entités, associations et diagrammes qui ont été supprimés font partie du périmètre. En conséquence, la suppression des objets ou des liens cibles correspondants est proposée.
  Voir "Cas d'emploi des options", page 14

Les autres options concernent la mise à jour des propriétés des objets cibles. Par défaut, la synchronisation met à jour l'ensemble des propriétés de chaque objet concerné.

Scheduling

Vous pouvez exécuter la synchronisation :

- Immédiatement
- dès que possible (après publication des mises à jour)
- A une date et heure prédéfinies

❱ Une fois les options définies, cliquez sur **Suivant**.

Etape 3 : protéger des objets

La synchronisation peut avoir un impact sur tous les objets d'une base de données existante.

❱ Pour garder un objet intact, cochez-le dans la colonne **Figé**.
❱ Cliquez sur **Suivant** pour passer à la suite.

Voir "Protéger des objets", page 16.

Etape 4 : valider les résultats

L'assistant affiche les résultats qu'entraînera la validation de la synchronisation.

Les objets devant subir une modification automatique sont signalés d'une coche.

Des icônes précédant le nom des objets indiquent les actions qui vont être effectuées sur ces objets. Il peut s'agir d'une création  , d'une suppression  ou d'une mise à jour 

Une flèche devant un objet indique que la synchronisation a un impact sur les sous-objets de l'objet en question.

❱ Dépliez l'objet pour visualiser les modifications concernées.

Validation des optimisations

Les optimisations sont des personnalisations sur des objets ainsi soustraits au traitement automatique de la synchronisation.

> Exemples d'optimisation :
>
> Une coche désigne les objets qui vont être modifiés. Si vous ne souhaitez pas valider les modifications portant sur certains objets, vous devez décocher les cases correspondantes. Cette optimisation est conservée lors des prochaines synchronisations.

Par ailleurs, **HOPEX** déduit des optimisations suite aux actions que vous avez pu effectuer manuellement. Si vous avez ajouté une table dans la vue physique sans avoir créé d'objet correspondant dans la vue logique, la synchronisation ne coche pas la suppression de cette table : 

Pour que l'objet soit supprimé, vous devez cocher la case correspondante.

- Une fois les actions sur les objets cibles définies, vous pouvez cliquer sur **Suivant**.

Un compte-rendu vous indique les actions effectuées.

Vous pouvez fermer l'assistant et visualiser les résultats dans l'éditeur.

Cas d'emploi des options

La combinaison des options "Prendre en compte les optimisations" et "Prendre en compte les suppressions" varie en fonction du périmètre des objets que vous souhaitez mettre à jour.

Prendre en compte les optimisations

Lorsque cette option est cochée, la synchronisation propose toutes les créations, suppressions et modifications, y compris les optimisations non cochées par défaut à l'étape de validation.

Lorsque l'option est décochée, seules les modifications cochées par défaut sont proposées à l'étape de validation.

Cette option permet de filtrer le résultat d'une synchronisation pour présenter uniquement les modifications qui ont un impact réel sur les données cibles.

Prendre en compte les suppressions

Lorsque cette option est cochée, la synchronisation prend en compte les entités, associations et diagrammes qui ont été supprimés. En conséquence, la suppression des objets ou liens cibles correspondants est proposée.

Lorsque l'option est décochée, les entités, associations et diagrammes qui ont été supprimés ne sont pas pris en compte. En conséquence, les objets cibles correspondants ne sont pas modifiés.

> **Cette option ne s'applique qu'aux entités, associations et diagrammes. Pour les autres types d'objets supprimés (attributs, identifiants, etc.), l'impact sur les objets ou liens cibles est conditionné non pas par cette option mais par l'objet qui les contient.**

Cette option permet de limiter l'impact d'une synchronisation au strict périmètre source défini par l'utilisateur, en excluant tout objet qui n'est pas explicitement déclaré dans le périmètre. Cette option peut être associée au périmètre de synchronisation pour des cas d'emplois types.

Combinaisons des options possibles

1. Option "Prendre en compte les suppressions" cochée et périmètre de synchronisation complet

Il s'agit d'un cas d'emploi qui privilégie une synchronisation complète entre la source et la cible. Dans ce cas, tous les objets ont une correspondance valide à l'issue de chaque lancement de l'assistant de synchronisation. Ce mode est à utiliser lorsque la source et la cible doivent être mis en cohérence en totalité.

2. Option "Prendre en compte les suppressions" cochée et périmètre de synchronisation partiel

Ce cas d'emploi permet de travailler sur le périmètre choisi, tout en incluant l'impact des objets supprimés. Ce mode permet notamment de s'assurer des suppressions d'objets cibles suite à la suppression d'objets sources de type entité, association ou diagramme : ces objets sources ayant été supprimés, il n'est théoriquement pas possible de les inclure dans le périmètre de synchronisation. Cocher cette option rend ce choix possible. Il s'agit du mode à privilégier lorsque le périmètre est large et que le peu d'objets exclus du périmètre le sont à titre provisoire (ce sont par exemple de nouveaux objets dont on veut différer l'impact sur la cible).

3. Option "Prendre en compte les suppressions" cochée et périmètre de synchronisation vide

Il s'agit d'un mode particulier permettant de "nettoyer" les objets cibles dont la correspondance n'est plus valide, sans autre impact.

4. Option "Prendre en compte les suppressions" non cochée et périmètre de synchronisation partiel

Ce cas d'emploi permet de travailler strictement sur le périmètre choisi, en excluant tout impact en dehors de ce périmètre. Ce mode permet d'éviter des suppressions d'objets cibles suite à la suppression d'objets sources de type entité, association ou diagramme : ces objets sources ayant été supprimés, il n'est théoriquement pas possible de les exclure du périmètre de synchronisation. L'option décochée rend ce choix possible. Il s'agit du mode à privilégier lors d'une synchronisation ponctuelle sur un périmètre restreint, qui ne prend pas en compte la cohérence totale de la source et de la cible. C'est par ailleurs le mode le plus rapide.

5. Option "Prendre en compte les suppressions" non cochée et périmètre de synchronisation complet

Cette combinaison présente un cas d'emploi en principe peu fréquent. Elle correspond à un mode de travail dans lequel la suppression d'objets sources est systématiquement sans effet sur la cible; autrement dit, tout objet cible créé n'est plus supprimé tant que ce mode est activé.

6. Option "Prendre en compte les suppressions" non cochée et périmètre de synchronisation vide

Cette combinaison est sans effet.

Protéger des objets

Vous pouvez protéger un objet afin que la synchronisation n'ait pas d'impact sur lui. Cela exclut l'objet de la synchronisation sans le faire disparaître.

Il existe deux modes de protection des objets ; l'un en amont de la synchronisation et l'autre en aval.

Mode Figé

Le mode "Figé" concerne l'objet cible, celui qui résulte de la synchronisation.

Lorsque vous figez une table du modèle relationnel, vous figez également tous les objets fils de cette table : aucun objet fils n'est créé, modifié ou supprimé suite à la synchronisation.

Vous pouvez figer des objets :

- Avant de lancer la synchronisation, dans l'éditeur de base de données.
- Lors du lancement de la synchronisation, dans les options présentées dans l'assistant. Voir "Etape 3 : protéger des objets", page 13.

Mode Réalisé

Le mode "Réalisé" concerne l'objet source de la synchronisation.

Un objet est dit réalisé s'il donne lieu à la création d'un objet lors d'une synchronisation.

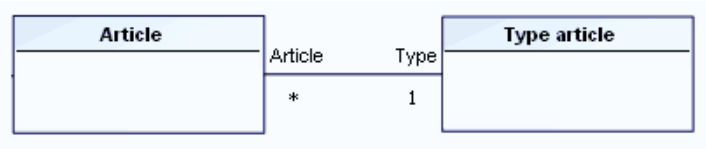
Un objet non réalisé ne donne pas lieu à la création d'un objet lors de la synchronisation mais il est traité comme un objet abstrait. Voir "Entité abstraite", page 4.

Par défaut, tous les objets sont réalisés.

Vous pouvez exclure un objet source de la synchronisation en cochant la colonne "Non réalisé" sur l'objet en question dans l'assistant de synchronisation. Cette action est disponible sur les entités, les attributs et les associations. L'action "Réalisé" ou "Non Réalisé" se propage aux objets fils.

Exemple d'entité non réalisée

L'entité "Article" possède une association vers l'entité "Type article".



L'entité "Type article" est dite "Non réalisée".

Lors de la synchronisation, l'entité "Type article" ne donne lieu à la création d'aucune table. Dans la table "Article", la clé étrangère vers "Type article" n'est pas créée; en revanche, la colonne "Code_Type_Article" est créée.

Résultats de la synchronisation : les correspondances

Quand la synchronisation est terminée, les tables, colonnes, clés et index du diagramme physique ont été mis en phase avec le diagramme de données. Ils peuvent être consultés et modifiés pour prendre en compte les optimisations éventuelles.

Caractéristiques de la correspondance

Pour plus de détails sur la correspondance :

- Dans l'éditeur de correspondances, sélectionnez l'élément de correspondance et cliquez sur **Propriétés**.
- Dans la fenêtre qui apparaît, cliquez sur la liste déroulante puis sur

Caractéristiques.

Périmètre de la synchronisation

Par défaut, tous les objets faisant partie des modèles synchronisés sont inclus dans la synchronisation. Vous avez cependant la possibilité d'exclure un objet de la synchronisation.

Voir "Etape 1 : sélectionner les objets sources à synchroniser", page 11.

Etat de la synchronisation

Vous pouvez protéger un objet afin qu'il ne subisse aucune modification lors de la synchronisation, vous préciserez alors qu'il est "Figé".

Voir aussi "Protéger des objets", page 16.

Sens de la synchronisation

Le sens de la synchronisation sur une correspondance permet de savoir quel objet est mis à jour par rapport à l'autre.

Dans certains cas, la synchronisation est possible dans les deux sens (par exemple, lorsque deux objets pouvant être synchronisés n'ont pas encore de correspondance). Dans d'autres cas, elle n'est possible que dans un sens (par exemple, si l'un des deux objets est déjà synchronisé) ou bien impossible dans les deux (parce que chaque objet a déjà une correspondance ou parce que les types des objets concernés ne peuvent faire l'objet d'une synchronisation). Cette indication est fournie par le champ **Synchronisation**.

Pour résumer :

- **Bidirectionnelle** : la synchronisation se fait dans les deux sens.
- **De gauche à droite :** la synchronisation se fait de gauche à droite (l'objet de droite est synchronisé par rapport à l'objet de gauche).
- **De droite à gauche :** la synchronisation se fait de droite à gauche.
- **Jamais :** pas de correspondance possible entre les deux objets.

Pour plus d'informations sur les correspondances, voir "L'éditeur de base de données", page 70.

SYNCHRONISATION RÉDUITE (MODE LOGIQUE VERS PHYSIQUE)

La fonctionnalité de synchronisation permet de synchroniser un modèle logique et un modèle physique d'une base de données. En phase de conception, il est souvent utile de synchroniser une partie du modèle courant sans avoir besoin de prendre en charge la totalité de la base de données qui peut être très volumineuse. **HOPEX Information Architecture** permet de limiter le périmètre d'une synchronisation à un ensemble d'objets, réduisant ainsi le temps de traitement de la synchronisation.

Les points qui suivent détaillent le mode "Synchronisation réduite" dans le sens Logique > Physique mais il est également disponible dans le sens Physique > Logique.

Objets sources de la synchronisation réduite

La synchronisation réduite est une synchronisation appliquée à un objet autre que la base de données. La synchronisation réduite s'applique uniquement à un objet dont la base de données a déjà été synchronisée.

Les objets sur lesquels vous pouvez lancer une synchronisation réduite sont :

- Classe
- Association
- Paquetage
- Table
- Dossier de Table
- Entité (DM)
- Association (DM)
- Modèle de données

Le périmètre de la synchronisation réduite est déterminé par l'objet sur lequel vous lancez la synchronisation réduite.

Les cas suivants illustrent des cas de synchronisation réduite dans le sens logique vers physique.

Lancement sur un modèle de données

Lorsque vous lancez une synchronisation sur un modèle de données, tous les objets du modèle sont sélectionnés par défaut dans le périmètre de la synchronisation ; ils sont tous cochés dans l'éditeur.

Lancement sur une entité du modèle de données

Lorsque vous lancez une synchronisation à partir d'une entité (ou un autre objet) appartenant à un modèle de données, seuls les objets en lien avec cette entité au sein du même modèle sont sélectionnés par défaut.

Les objets en lien avec l'entité mais qui appartiennent à un autre modèle sont affichés dans l'éditeur (dès lors qu'ils sont reliés à la base de données cible) mais non sélectionnés par défaut. Vous devez cocher les cases associées pour les prendre en compte dans la synchronisation.

Le modèle de données de l'entité synchronisée est pris comme périmètre uniquement lorsque le lien de détention entre le modèle de données et l'entité est clairement identifié : c'est le cas lorsque vous sélectionnez l'entité dans la fenêtre de navigation, ce n'est pas le cas lorsque vous la sélectionnez dans un diagramme.

Lancement sur une entité hors contexte

Lorsque vous lancez la synchronisation sur une entité hors contexte, par exemple dans une fenêtre d'explorateur, tous les objets qui dépendent de l'entité modifiée, compris ou non dans des modèles différents (à condition que ces modèles soient reliés à la base de données cible) sont cochés par défaut dans le périmètre de la synchronisation, car aucun contexte de modèle en particulier n'est identifié.

Les stratégies de synchronisation réduite

Lors de la synchronisation à partir d'un objet, il est possible d'appliquer une des trois stratégies ci-dessous.

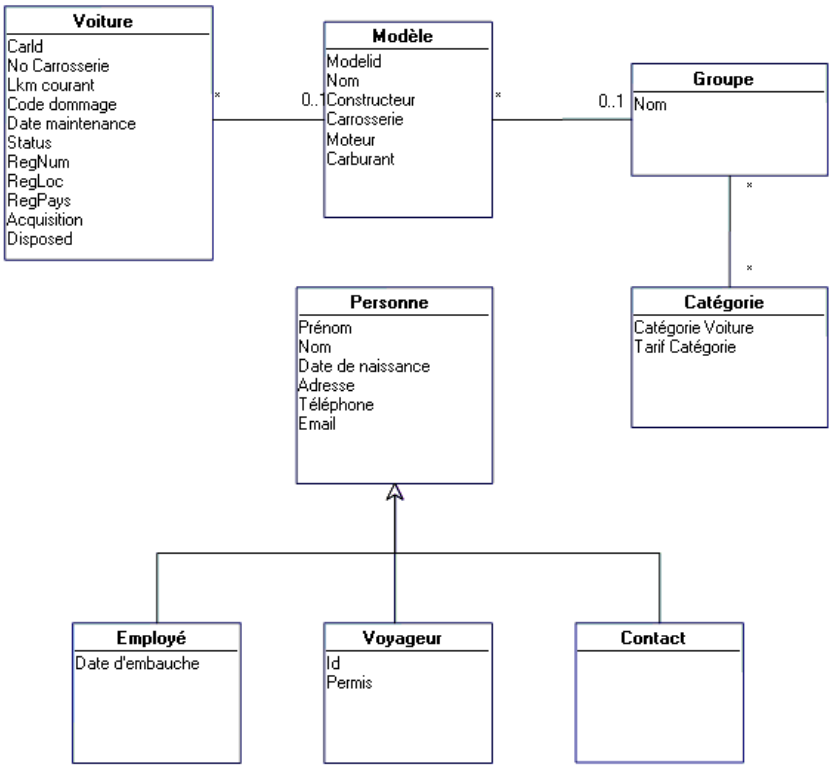
Impact de l'objet synchronisé sur les autres objets

Cette stratégie permet de définir le périmètre de la synchronisation depuis l'objet source et de l'étendre à tous les objets qui dépendent de lui et qui sont susceptibles d'être influencés par sa modification.

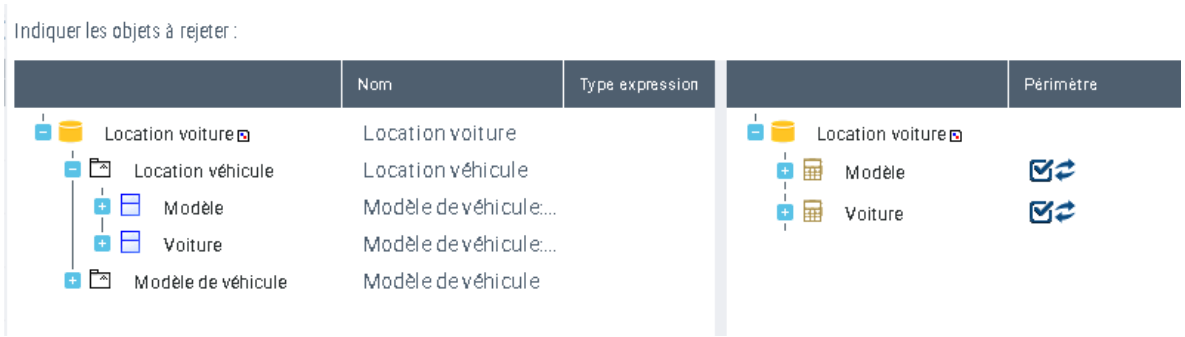
Exemple

Avec cette stratégie, dans le modèle ci-dessous, la synchronisation réduite de l'entité "Modèle" permet d'inclure également l'entité "Voiture" qui a une association contrainte vers l'entité "Modèle".

Modèle logique



Résultats de la synchronisation réduite



Impact des autres objets sur l'objet synchronisé

Cette stratégie permet d'intégrer dans le périmètre de la synchronisation réduite les objets dont dépend directement l'objet source, par exemple, tous les objets

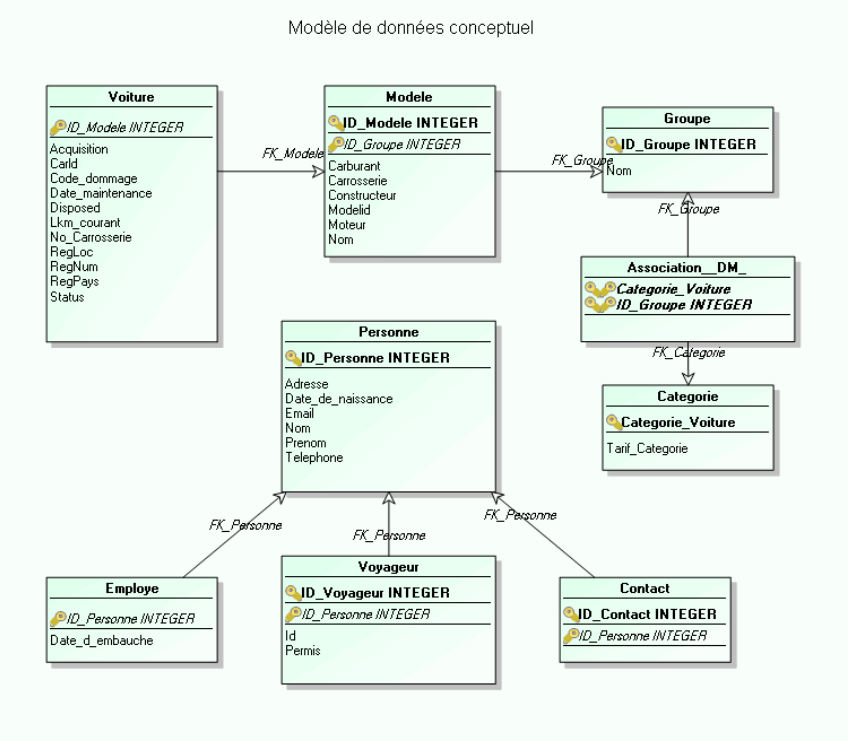
associés à l'entité source et qui sont nécessaires à la mise à jour de la table correspondante.

Exemple

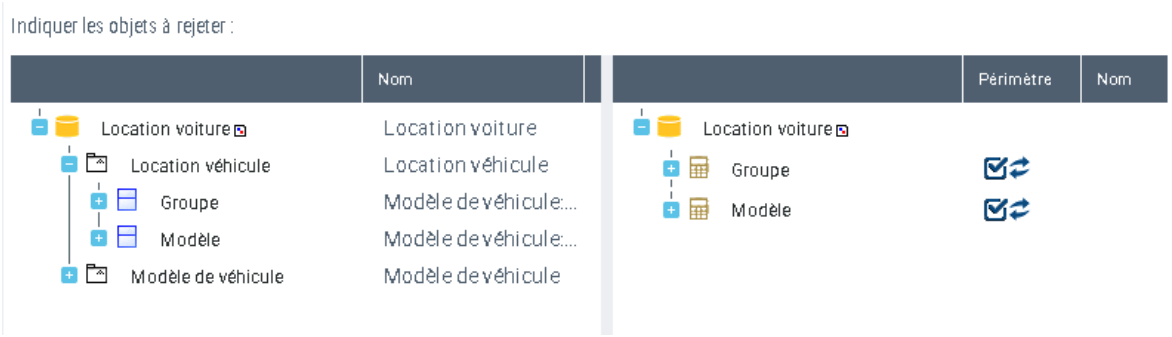
Dans le même exemple que précédemment, en prenant l'entité "Modèle" comme source d'une synchronisation réduite, le périmètre s'étend à l'entité "Groupe" car les tables correspondant à ces deux entités sont liées par une clé étrangère : l'entité "Groupe" peut modifier la table "modèle" associée à l'entité "Modèle" par l'intermédiaire de la clé étrangère "FK_Group" (voir schéma ci-dessous).

L'entité "Voiture" quant à elle n'est pas prise en compte dans le périmètre car elle ne peut pas agir sur la table "Modèle".

Modèle physique



Résultats de la synchronisation réduite

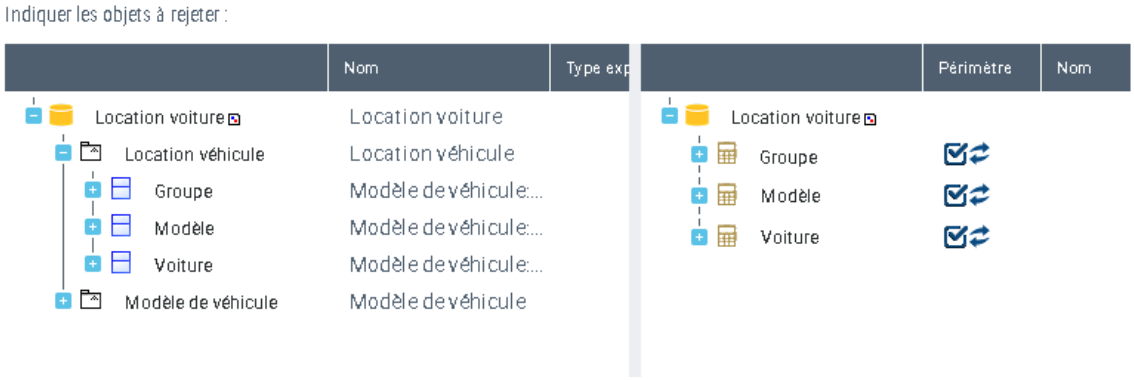


Tous les impacts

Cette stratégie permet de combiner les deux stratégies citées ci-dessus. Ainsi le périmètre de la synchronisation réduite est étendu aux objets dont l'objet source a besoin et à tous les objets susceptibles d'être influencés.

Exemple

Résultat de la synchronisation réduite



Lancer la synchronisation réduite

Avant de commencer, vérifiez que l'option « Synchronisation réduite (logique -> physique) » est active :

1. Dans le bureau, cliquez sur le menu **Menu principal** > **Paramètres** > **Options**.
   La fenêtre des options apparaît.
2. Dans la partie gauche de la fenêtre, dépliez le dossier "Modélisation des données".
3. Cliquez sur "Synchronisation de base de données".

4. Dans la partie droite de la fenêtre, cochez la case "Activer la synchronisation réduite (logique > physique)".

Pour lancer la synchronisation réduite :

1. Faites un clic droit sur l'objet à synchroniser.
2. Sélectionnez **Synchroniser (Logique vers physique)**.
   L'assistant de synchronisation apparaît.

   Une entité peut être utilisée par plusieurs modèles de données, donc par plusieurs bases de données. Quand c'est le cas, vous devez sélectionner la base concernée.
3. Sélectionnez la **Stratégie**.
4. Cliquez sur **Suivant**.
5. Sélectionnez les objets à synchroniser.
   Le périmètre coché par défaut est fonction du contexte dans lequel vous sélectionnez l'objet à synchroniser : si la synchronisation réduite est initialisée à partir d'une entité dans un diagramme, le modèle du diagramme en question est sélectionné. Si l'entité est sélectionnée hors contexte, l'ensemble des modèles dans lesquels elle apparaît s'affiche dans l'éditeur.

   ☛ *Les objets sélectionnés ne sont pas mémorisés ; lors d'une nouvelle synchronisation, le périmètre par défaut s'affiche à nouveau.*
6. Cliquez sur **Suivant**.
7. Définissez les **Options de synchronisation**. Toutes les options de la synchronisation standard sont disponibles à l'exception de l'option "Réinitialiser les objets cibles".
8. Cliquez sur **Suivant**.
   L'option de protection des objets cibles s'affiche ; vous pouvez visualiser les objets figés. Il n'est pas possible de modifier la protection des objets.
9. Validez les résultats en cliquant sur **Suivant**.
   Le rapport de la synchronisation apparaît.
10. Cliquez sur **OK** pour fermer l'assistant de synchronisation.
    L'éditeur de correspondances apparaît (sauf si l'option correspondante a été décochée de l'assistant de synchronisation).

Options de la synchronisation réduite

La synchronisation réduite présente les mêmes options que la synchronisation totale à l'exception de :

- Réinitialisation les objets cibles
- Ordre

En effet, le périmètre réduit de la synchronisation réduite ne permet pas de rendre un résultat valide pour ces deux options.

RELANCER LA SYNCHRONISATION APRÈS MODIFICATIONS

Lorsqu'une base de données a été synchronisée puis modifiée manuellement, les spécifications complémentaires effectuées directement dans la base de données sont conservées, sauf si :

- Une réinitialisation est demandée.
- Des changements intervenus dans les diagrammes de données s'y opposent (ajout ou suppression d'objets ou de liens).

Parmi ces changements peuvent être cités :

- La création d'entités, d'associations, d'attributs dans le diagramme de données
- La suppression d'entités, d'associations, d'attributs dans le diagramme de données
- La modification des caractéristiques d'un attribut
- La modification du nom d'un attribut, d'une entité, ou d'une association
- La modification de la multiplicité maximum d'une association
- La modification des liens d'une association

Les spécifications complémentaires dans le diagramme relationnel peuvent être :

- Des suppressions d'objets créés par la synchronisation
- Des créations d'objets
- Des modifications des caractéristiques d'objets créés par la synchronisation
- Des modifications de l'ordre des colonnes dans les tables, les clés, ou les index.

Synchronisation après modification du diagramme de données

Création d'entités, d'associations et d'attributs dans le diagramme de données

Les éléments correspondants sont créés dans le diagramme relationnel, en suivant les règles utilisées lors de la première synchronisation.

Suppression d'entités, d'associations, d'attributs dans le diagramme de données

Les éléments correspondants sont supprimés dans la base de données. Par exemple, lorsqu'un attribut est supprimé dans le diagramme de données, la ou les colonnes correspondantes sont également supprimées.

Modification des caractéristiques d'un attribut

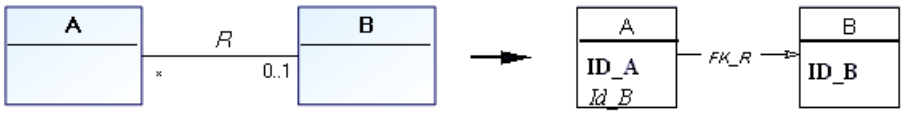
Les modifications des caractéristiques d'un attribut (type, longueur, décimales,...) sont reportées sur la colonne correspondante du diagramme relationnel.

Si la valeur d'une caractéristique de la colonne a été modifiée directement dans le diagramme relationnel, cette valeur est conservée.

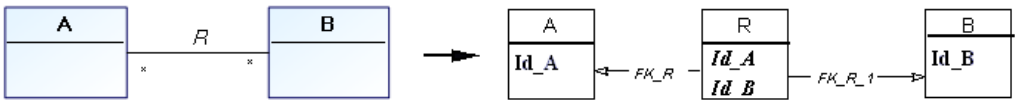
Modification du nom d'un attribut, d'une entité, ou d'une association

La modification du nom d'un attribut, d'une entité, ou d'une association n'est pas reportée sur l'objet correspondant du diagramme relationnel.

Modification de la multiplicité maximum d'une association



Si la multiplicité maximum d'une association qui était à 1, entraînant ainsi la création d'une colonne migrante, passe à N, la colonne migrante est supprimée, et la table justifiée par la multiplicité maximum à N est créée.



Modification des liens d'une association

L'association R ne porte plus sur l'entité B, mais sur l'entité C.

Dans ce cas, la colonne migrante de B dans A n'est plus justifiée.

- Elle est supprimée.
- Une colonne migrante provenant de C est créée.

Synchronisation après des modifications du diagramme physique

Suppression d'une table ou d'une colonne

Les suppressions dans la base de données d'objets qui ont été créés par la synchronisation (table, colonne, clé, index) sont mémorisées et conservées.

Tant que l'entité, l'association ou l'attribut qui justifie la table ou la colonne existe, la table ou la colonne n'est pas recréée.

Pour rétablir une colonne créée par la synchronisation puis supprimée :

1. Lancez la synchronisation de la base de données
2. Lors de l'étape de validation des résultats, confirmez l'action de création (cochez la case correspondante) proposée pour cette colonne.

Créations d'objets

Les créations d'objets (table, colonne, clé, index) effectuées dans le diagramme relationnel sont conservées.

Cependant, la suppression d'objets dont ils dépendent peut entraîner leur suppression.

Par exemple, on crée une colonne dans une table justifiée par une entité. Si on paramètre l'entité comme "non table" ou qu'on délie l'entité du modèle de données, la table correspondante va disparaître, et avec elle la colonne.

☛ *Les objets créés dans le diagramme relationnel peuvent être mis en correspondance manuellement. Les objets créés lors de la synchronisation sont mis en correspondance automatiquement.*

Modifications de caractéristiques d'objets créés par la synchronisation

Les modifications de caractéristiques (nom SQL, longueur, not null, datatypes) d'objets créés par la synchronisation sont conservées.

Modifications d'ordre

Concernant les modifications d'ordre, le traitement dépend des options définies dans l'assistant de synchronisation (voir "Etape 2 : options de synchronisation", page 31).

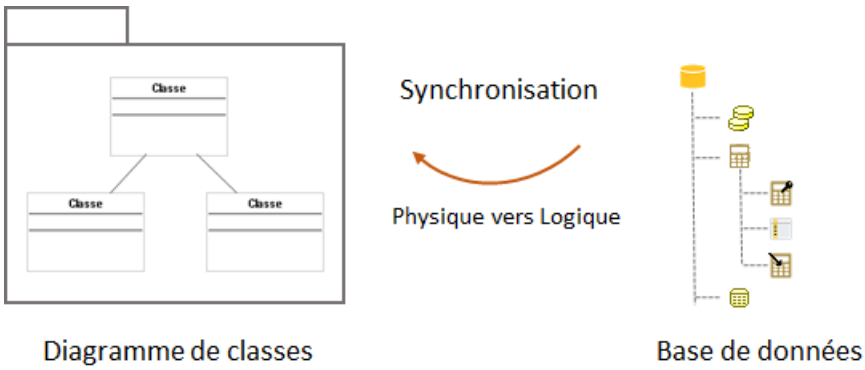
DU MODÈLE PHYSIQUE AU MODÈLE LOGIQUE

Cette partie présente comment synchroniser le modèle physique d'une base de données avec le modèle logique correspondant.

Règles de synchronisation "Physique vers Logique"

Synchroniser le modèle logique à partir du modèle physique permet de créer le diagramme de données d'une base à partir de ses tables.

Les règles utilisées pour cette transformation sont :



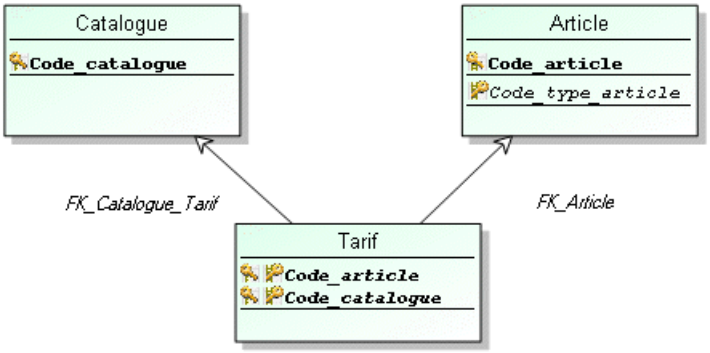
- Une table dont la clé primaire est constituée d'une clé étrangère qui porte sur les mêmes colonnes devient une entité. Une généralisation est créée entre cette entité et l'entité correspondant à la table vers laquelle pointe la clé étrangère.

Exemple niveau physique :

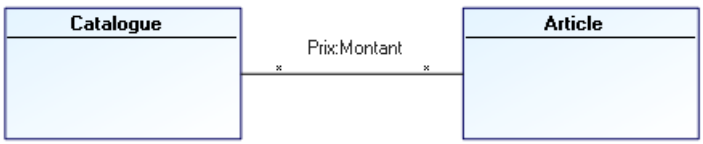
Résultat au niveau logique :

- Une table dont la clé primaire est constituée uniquement de clés étrangères devient une association de multiplicités (*..*). S'il existe des colonnes de la table qui ne font pas partie de la clé primaire, chacune de ces colonnes donne lieu à la création d'un attribut qui est rattaché à l'association.

Exemple niveau physique :



Résultat au niveau logique :



- Une table dont la clé primaire contient des clés étrangères et au moins une colonne qui n'est pas clé étrangère devient une entité. Une association agrégée est créée entre cette entité et l'entité correspondant à la table vers laquelle pointe chacune des clés étrangères.

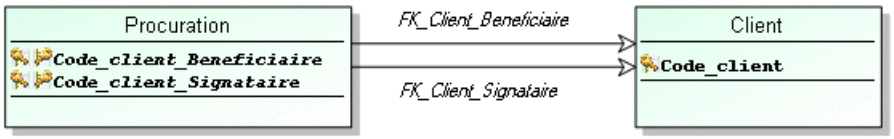
La clé candidate de l'entité est composée des rôles des associations agrégées et des attributs correspondant aux autres colonnes de la clé primaire de la table.

Exemple niveau physique :

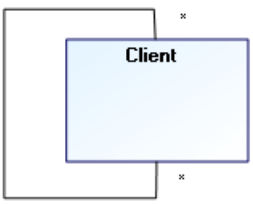
Résultat niveau logique :

- Une table dont la clé primaire est constituée uniquement de deux clés étrangères pointant sur une même table devient une association réflexive de multiplicités (*..*).

Exemple niveau physique :



Résultat niveau logique :



- Dans les autres cas, chaque table devient une entité et ses colonnes les attributs de l'entité.

- Une clé étrangère devient une association (0..1, *). Si toutes les colonnes de la clé sont obligatoires, ses cardinalités deviennent (1, *).

- Les types des attributs sont recalculés à l'aide du tableau de conversion spécifique au SGBD cible (voir "Types des attributs et des colonnes", page 99).

Lancer la synchronisation

Pour lancer la synchronisation :

1. Sélectionnez la base de données concernée (dans la liste des bases de données ou dans un diagramme par exemple).
2. Faites un clic droit sur la base de données et sélectionnez **Synchronisation (Physique vers Logique)**.
   L'assistant de synchronisation apparaît.

Etape 1 : sélectionner les objets à synchroniser

Pour définir le périmètre de la synchronisation :

Déroulez la liste des objets contenus dans la base de données.

1. Par défaut, tous les objets sont cochés et donc inclus dans la synchronisation. Pour exclure un objet de la synchronisation, décochez-le dans la colonne **Périmètre**. Lorsqu'un objet est exclu, sa correspondance l'est aussi.
2. Par défaut, tous les objets sont "réalisés", autrement dit donnent lieu à la création d'un objet lors de la synchronisation. Pour indiquer qu'un objet est "non réalisé", cochez-le dans la colonne **Non réalisé**. Pour plus d'informations voir "Mode Réalisé", page 16.
3. Une fois la liste des objets définie, cliquez sur le lien **Suivant** de l'assistant.

Etape 2 : options de synchronisation

Parmi les options de synchronisation, vous pouvez :

- Réinitialiser les objets cibles : la synchronisation repart de zéro et supprime les objets cibles existants.
- Recalculer le nom des objets cibles : les noms des objets du diagramme de données sont recalculés en fonction de ceux des objets du diagramme relationnel. Cela signifie que toute modification manuelle du diagramme de données est annulée.
- Prendre en compte les optimisations : toutes les optimisations - dont celles qui ne sont pas cochées dans l'étape de validation (voir "Validation des optimisations", page 14) sont proposées.
- Prendre en compte les suppressions : les entités, associations et diagrammes qui ont été supprimés font partie du périmètre. En conséquence, la suppression des objets ou des liens cibles correspondants est proposée.
  Voir "Cas d'emploi des options", page 14

Vous devez également indiquer le modèle de données qui va détenir l'ensemble des objets créés par la synchronisation.

Les autres options concernent la mise à jour des propriétés des objets cibles. Par défaut, la synchronisation met à jour l'ensemble des propriétés de chaque objet concerné.

Scheduling

Vous pouvez exécuter la synchronisation :

- Immédiatement
- dès que possible (après publication des mises à jour)
- A une date et heure prédéfinies

❱ Une fois les options définies, cliquez sur **Suivant**.

Etape 3 : protéger des objets

La synchronisation peut avoir un impact sur tous les objets cibles.

❱ Pour garder un objet intact, cochez-le dans la colonne **Figé**.
❱ Cliquez sur **Suivant** pour passer à la suite.

Voir "Protéger des objets", page 16.

Etape 4 : valider les résultats

L'assistant affiche les résultats qu'entraînera la validation de la synchronisation.

❱ Pour valider ces résultats, cliquez sur **Suivant**.

Un compte-rendu présente la liste des traitements effectués.

Synchronisation réduite

La synchronisation ci-dessus s'applique à une base de données mais vous pouvez également lancer une synchronisation Physique->Logique sur un objet particulier d'une base de données afin de réduire le périmètre et le temps de traitement de la synchronisation. Voir "Synchronisation réduite (mode logique vers physique)", page 19.

Résultat de la synchronisation "Physique vers Logique"

Modèle de données détenteur

Un modèle de données détenteur par défaut des entités est créé lors de la synchronisation Physique vers Logique. Les entités et associations synchronisées lui sont automatiquement rattachées. Vous pouvez par la suite répartir ces entités et associations entre les différents modèles de données de votre étude.

Diagrammes de données

Un diagramme de données est créé pour chacun des diagrammes relationnels de la base de données. Les entités et associations issues des tables du diagramme relationnel correspondant y sont reliées.

Les correspondances

Voir "Résultats de la synchronisation : les correspondances", page 17.

PARAMÉTRER LA SYNCHRONISATION

Cette partie présente les options et les paramètres par défaut pris en compte lors de la synchronisation.

Préparer la synchronisation

Pour préparer la synchronisation :

1. Faites un clic droit sur la base de données et sélectionnez **Propriétés**. La fenêtre de propriétés apparaît.
2. Cliquez sur la liste déroulante puis sur **Options** > **Standard**.
3. Si vous voulez que le nom de la base physique utilisé lors de la génération de scripts SQL soit différent de celui de la base de données, indiquez le nom de la base cible dans le champ **NomSql**.
4. Indiquez éventuellement le **Préfixe** des tables SQL. Ce préfixe sera ajouté devant le nom de chacune des tables générées.

   ☛ *Des paramètres supplémentaires pour le paramétrage des ordres de synchronisation et de génération peuvent être indiqués sous les **Options**. Ces paramètres varient en fonction du SGBD sélectionné.*
5. Cliquez à nouveau sur la liste déroulante de la fenêtre de propriétés puis sur **Caractéristiques**.
6. Sélectionnez le **SGBD cible** ainsi que sa version.

   ☛ *Le type de SGBD détermine :*

   - *Pour la synchronisation, la génération des datatypes des colonnes en fonction du type et de la longueur des attributs (voir "Déduire les datatypes des colonnes à partir des types des attributs", page 104).*
   - *Pour la génération, la syntaxe des ordres SQL générés.*
7. Cliquez sur **OK** pour fermer la fenêtre en prenant en compte les modifications.

Options de création

Sur une base de données

Il est possible de paramétrer la synchronisation pour chaque base de données, afin de modifier :

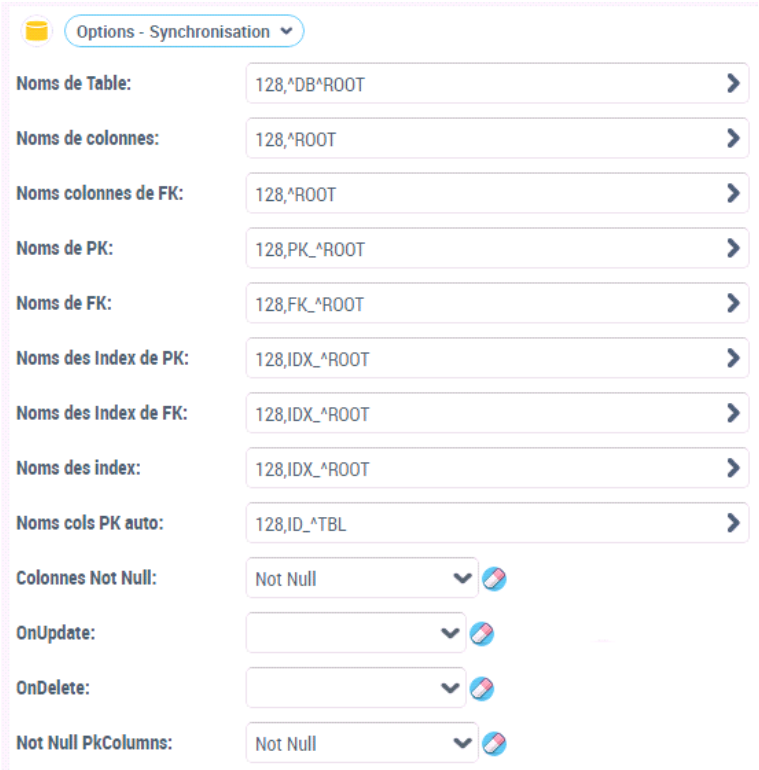
- Ses options de création
- Le traitement de l'intégrité référentielle (clés en suppression et en mise à jour, en fonction des possibilités offertes par le SGBD cible)

Cette configuration concerne également le traitement des colonnes Not Null et la création automatique d'index sur les clés primaires.

Pour configurer les options de création de la base de données :

1. Faites un clic droit sur la base de données et sélectionnez **Propriétés**. La fenêtre de propriétés apparaît.
2. Cliquez sur la liste déroulante puis sur **Options** > **Synchronisation**. Les options correspondantes s'affichent.

Vous pouvez renseigner les paramètres suivants



- **Colonnes Not Null** : active/désactive l'option WITH DEFAULT pour les colonnes Not Null.
- **OnDelete** : stratégie par défaut pour la suppression des clés. Les valeurs possibles sont :
  - **Restrict** : la suppression est refusée.
  - **Cascade** : la suppression d'une colonne est répercutée dans les tables dépendantes.
  - **Set Null** : indique "Null" à la place.
  - **Set Default** : donne la valeur par défaut lorsque celle-ci est renseignée. Si aucune valeur par défaut n'est renseignée, il ne se passe rien ("No action").
  - **No Action** : il ne se passe rien.
- **OnUpdate** : stratégie par défaut pour la mise à jour des clés.
- **Noms cols PK auto** : colonnes issues de l'identifiant implicite. Voir "Règle générale", page 3.

Les paramètres dont le nom commence par "**Noms de**" permettent d'indiquer les règles de génération des noms (voir "Paramétrer la génération des noms", page 36).

Sur le SGBD

Les valeurs par défaut des paramètres de synchronisation et de génération des bases de données sont accessibles dans la fenêtre de propriétés du SGBD utilisé.

Pour afficher les paramétrages de synchronisation d'un SGBD :

1. Dans la barre **HOPEX**, cliquez sur la fenêtre de navigation **Utilitaires**.
2. Faites un clic droit sur le nom du SGBD cible et ouvrez sa fenêtre de **Propriétés**.
3. Cliquez sur la liste déroulante puis sur **Options** > **Synchronisation**.

Par défaut, les paramètres spécifiés au niveau du SGBD sont valables pour toutes les nouvelles bases de données. Lorsque vous modifiez les paramètres de synchronisation au niveau d'une base, cette base ne tient plus compte des paramètres du SGBD.

☛ *Pour plus d'informations sur le paramétrage de la synchronisation, voir également "Lancer la synchronisation", page 11.*

Paramétrer la génération des noms

Règles de construction des noms

Les noms des objets physiques créés lors de la synchronisation "logique vers physique" sont déduits du **Nom Local** des objets logiques dont ils sont issus.

Les noms des objets logiques n'étant soumis à aucune restriction particulière, des règles de transformation s'appliquent par défaut lors de leur synchronisation. Ces règles sont accessibles localement dans la page **Options** > **Standard** des propriétés de la base de données synchronisée, ou globalement dans les propriétés des SGBD cibles :

- **Taille identifiant** : taille maximale d'un identificateur SQL pour ce SGBD cible.
- **1er caractère** : jeu de caractères autorisé pour le 1er caractère d'un identificateur SQL.
- **Caractères autorisés** : jeu de caractères autorisé pour les caractères d'un identificateur SQL.
- **Caractère de remplacement** : caractère de remplacement des caractères non autorisés.
- **Caractères convertis** : jeu de caractères à convertir d'un identificateur SQL.
- **Caractères de conversion** : jeu de caractères correspondant aux caractères à convertir.
- **Conversion majuscule** : conversion en majuscules des identificateurs SQL.

Il est possible d'indiquer un autre nom pour chaque objet synchronisé à l'aide de son **Nom SQL**. Le **Nom SQL** se substitue au **Nom Local** lors de la synchronisation, tout en tenant compte des règles de transformation par défaut.

Le **Nom SQL** des objets logiques est accessible dans la page **Génération** > **SQL** (ou page **SQL**) de leur fenêtre de propriétés.

Vous pouvez donner un nom différent selon la base de données ou le SGBD.

Lorsque les champs **Nom SQL (Base de données)** et **Nom SQL (SGBD)** indiquent des noms différents, c'est le nom défini au niveau de la base de données qui prévaut lors de la synchronisation.

Par défaut, les noms des objets relationnels sont générés selon les masques suivants :

Table	Préfixe de la base de données + nom* de l'entité ou de l'association
Colonne	Nom* de l'attribut
Clé primaire	"PK_" + nom* de l'entité ou de l'association
Clé étrangère	"FK_" + nom* de l'entité cible ou du rôle si renseigné
Index de clé primaire	"IDX_" + nom* de l'entité ou de l'association
Index de clé étrangère	"IDX_" + nom* de l'entité cible ou du rôle si renseigné

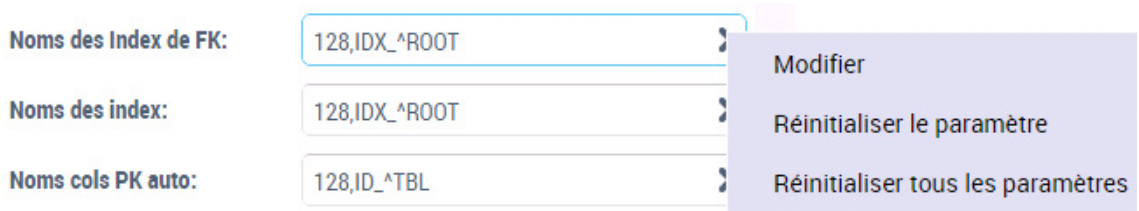
**Nom calculé selon les règles de construction expliquées précédemment.*

Ces masques peuvent être modifiés localement dans chaque base de données, ou globalement pour un SGBD cible donné.

Modifier une règle de construction

Pour modifier le masque d'une règle de construction de nom :

1. Faites un clic droit sur la base de données et sélectionnez **Propriétés**.
2. Cliquez sur la liste déroulante puis sur **Options** > **Synchronisation**.
3. Dans le champ de la règle en question, cliquez sur la flèche.



4. Cliquez sur **Modifier**.
La fenêtre de **Saisie du masque SQL** est présentée.

Saisie du masque SQL

Les masques SQL définissent la règle de construction des noms des objets relationnels lors de la synchronisation.

```
Exemple : dans la base BASE_EMPLOYES qui a pour préfixe EMP,
le masque ^DB_^ROOT génère pour la table issue de l'entité
Client : EMP_CLIENT.
```

Dans la fenêtre de saisie du masque SQL, vous pouvez saisir directement le **Masque**, en utilisant la syntaxe indiquée ci-après, mais vous pouvez également utiliser l'aide à la saisie proposée par le cadre **Composant**.

- Dans la liste du cadre **Composant**, indiquez les éléments qui doivent préfixer les noms. Ces éléments sont les suivants :

^ROOT	• Pour une table : nom* de l'entité ou de l'association dont elle provient. • Pour une colonne : nom* de l'attribut. • Pour une clé primaire : nom* de l'entité ou de l'association. • Pour une clé étrangère : nom* de l'entité cible ou du rôle si renseigné. • Pour une colonne de FK : nom* de l'attribut. • Pour une colonne de PK auto : nom* de l'identifiant de l'entité. • Pour un index sur clé primaire : nom* de l'entité ou de l'association. • Pour un index sur clé étrangère : nom* de l'entité cible cible ou du rôle si renseigné. • Pour un index : nom* de l'attribut.
^DB	Préfixe de la base de données
^TBL	Nom de la table
^TBR	Nom de la table de référence
^KEY	Nom de la clé
^CPT	Composteur

**Nom calculé selon les règles de construction expliquées précédemment.*

La **Taille** indique la longueur limite totale du nom généré. Elle est également disponible pour chacun des éléments utilisés, qui seront tronqués au nombre de caractères indiqué entre parenthèses après l'élément concerné.

La définition d'un **Composteur** ("^CPT") permet de générer automatiquement un numéro d'ordre, et d'en indiquer la longueur (par exemple, ^CPT[^1^] générera "1", "2", "3"; ^CPT[^3^] générera "001", "002", "003").

L'option **Toujours** indique que le compostage commence dès la première occurrence (CLI00, CLI01, .., au lieu de CLI, CLI01,…,).

Vous pouvez également préciser les caractères utilisés comme préfixe et suffixe de ce composteur.

L'option **Unicité du nom** permet de garantir l'unicité du nom d'un objet au niveau de la base de données, du référentiel ou de la table dans lesquels se trouve cet objet. Ainsi, lorsque vous définissez l'unicité de nom au niveau de la table, il n'est

pas possible d'attribuer un même nom à des objets différents au sein d'une même table.

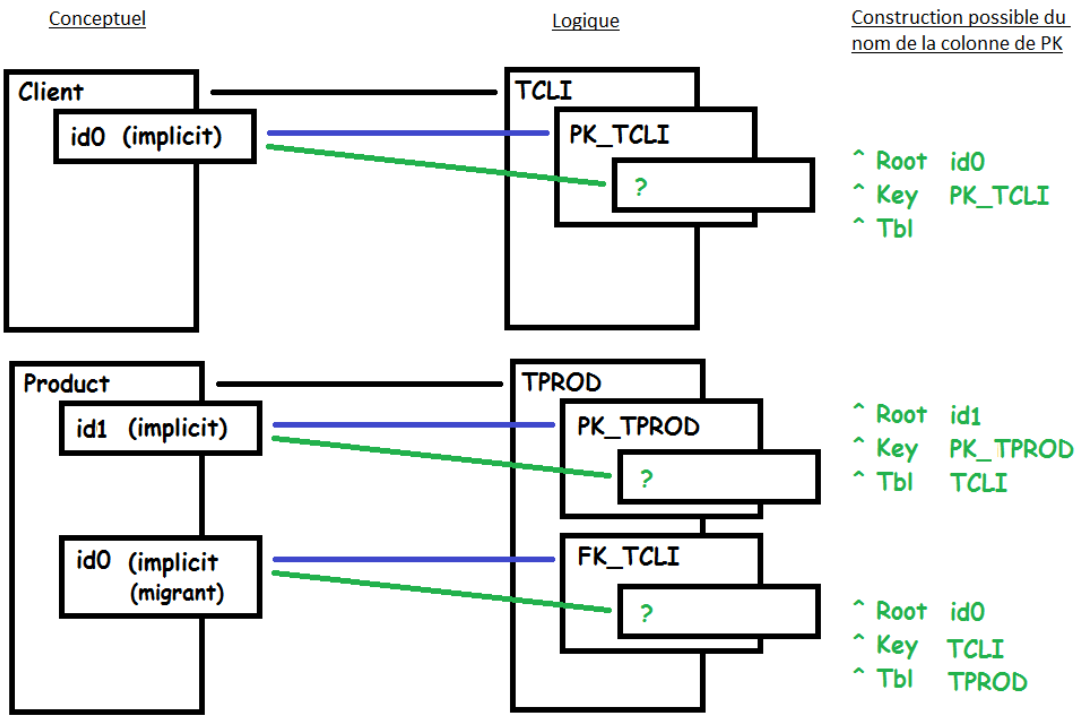
Paramétrer le nom des colonnes de PK (identifiant implicite)

Lors de la synchronisation en mode Logique > Physique, l'identifiant de l'entité devient la clé primaire de la table. Si l'identifiant est implicite, une colonne est automatiquement créée. Pour plus de détails, voir "Règles de synchronisation "logique vers physique"", page 3.

Par défaut le nom d'une colonne issue d'un identifiant implicite est construit à l'aide du mot clé ^TBL qui correspond :

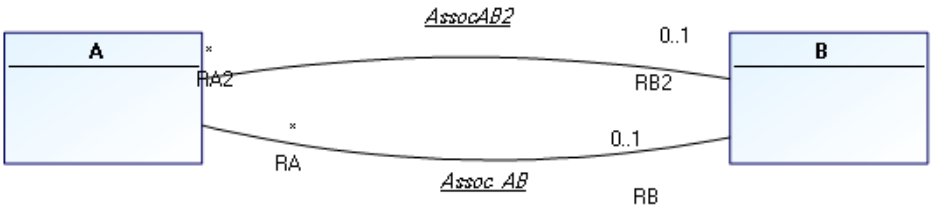
- au nom de la table migrante (autrement dit issue d'une clé étrangère) si l'identifiant est migrant
- au nom de la table si l'identifiant n'est pas migrant

Vous pouvez modifier les règles de construction et construire le nom de ces colonnes avec le mot clé ^KEY qui correspond au nom de la clé étrangère (sans " FK_ ") si l'identifiant est migrant, ainsi qu'avec le mot clé ^ROOT qui correspond au nom de l'identifiant (ID). Voir "Modifier une règle de construction", page 37.

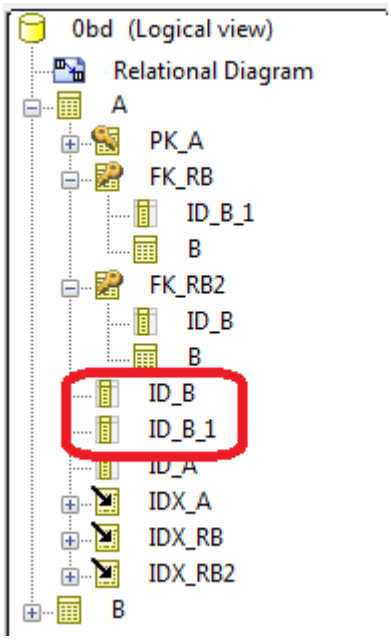


Exemple

Lorsqu'il existe deux associations contraintes entre deux entités comme ci-dessous :



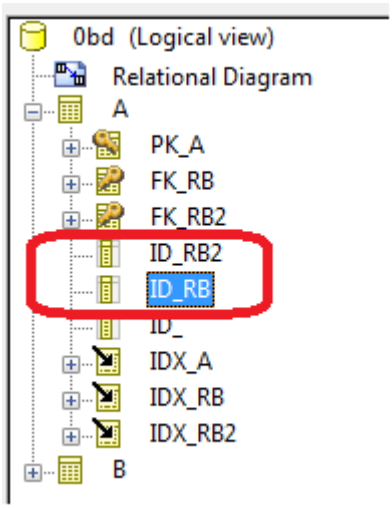
Par défaut, après synchronisation, vous obtenez deux colonnes avec des noms identiques seulement différenciés par le préfixe "1".



Vous pouvez modifier la règle de nommage et construire le nom de ces colonnes avec le mot clé ^KEY qui correspond au nom de la clé étrangère (sans " FK_ ").

Le nom de la clé étrangère étant calculé sur le nom du Rôle lorsqu'il est renseigné, alors le nom obtenu pour les deux colonnes sera différent.

Dans notre exemple, si vous remplacez "ID^TBL" par "ID^KEY" vous obtenez après synchronisation :



SYNCHRONISATION DES DIAGRAMMES

La synchronisation permet de traduire un modèle de données logique en un modèle physique, et inversement. Avant de lancer une synchronisation, les diagrammes sources (diagrammes de données ou diagrammes physiques) doivent être enregistrés et fermés.

Une première synchronisation crée de façon automatique le diagramme du modèle cible. Une nouvelle synchronisation sur un modèle préalablement synchronisé n'entraîne pas une mise à jour automatique du diagramme, autrement dit de la représentation graphique du modèle. Suivant les modifications que vous avez apportées, l'assistant de synchronisation propose ou non de mettre à jour le diagramme cible.

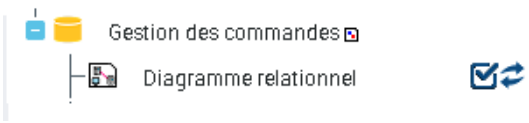
Cas de mise à jour des diagrammes lors d'une synchronisation

L'assistant de synchronisation propose une mise à jour des diagrammes selon les cas suivants.

Après modification du diagramme source

Lorsque vous lancez la synchronisation après modification du diagramme source, l'assistant active par défaut la mise à jour du diagramme cible. La mise à jour est signalée par l'affichage des deux petites flèches bleues.

Ci-dessous, lors d'une synchronisation en mode logique vers physique, la modification du diagramme logique entraîne automatiquement la mise à jour du diagramme physique.



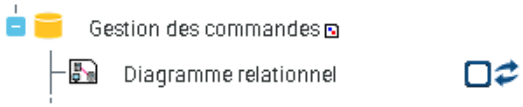
Après modification du diagramme cible

Si vous avez modifié le diagramme cible, par exemple le diagramme physique, par défaut la synchronisation ne propose pas de mettre à jour ce diagramme cible.

Pour afficher la case de mise à jour du diagramme cible, vous devez cocher l'option de synchronisation "Prendre en compte les optimisations". Et pour que la mise à jour soit activée, vous devez cocher la case en question.

Après modification des deux diagrammes

Si les deux diagrammes - logique et physique - ont été modifiés, la mise à jour du diagramme cible est proposée mais non cochée par défaut.



Pas de modification détectée

Si aucun des deux diagrammes n'a été modifié, l'assistant ne propose aucune mise à jour. Il est à noter que toute modification d'un diagramme doit faire l'objet d'une sauvegarde avant fermeture du diagramme pour être prise en compte par l'assistant de synchronisation.

Cas particulier : une entité en correspondance avec deux tables

Lorsqu'une entité du modèle logique est associée à deux tables dans le modèle physique (suite à une fusion des entités par exemple), le diagramme physique mis à jour n'affiche qu'une des deux tables.

CORRESPONDANCE DES MODÈLES

Dans beaucoup de projets de modélisation se pose le problème de la communication entre les équipes d'analystes, d'architectes et les équipes de développement des bases de données.

Il existe dans **HOPEX Information Architecture** deux niveaux de modélisation :

- ✓ Le niveau logique, qui décrit la modélisation des données en termes d'entités et de relations, et qui s'adresse aux analystes et utilisateurs.
- ✓ Le niveau physique (ou relationnel) qui décrit la base de données en termes de tables et qui fait l'interface avec le SGBD. Ce niveau s'adresse au concepteur et à l'administrateur de base de données.

En permettant le passage d'un modèle de données à un autre, l'éditeur de base de données favorise la cohésion entre l'architecture des données et les systèmes qui les supportent.

Les points abordés ci-après sont :

- ✓ "L'éditeur de base de données", page 70
- ✓ "Détails des correspondances", page 73

L'ÉDITEUR DE BASE DE DONNÉES

L'éditeur d'une base de données permet de mettre en correspondance de façon manuelle les différentes vues d'une base de données, qui sont le modèle logique et le modèle physique (ou relationnel).

L'assistant de synchronisation met en correspondance de façon automatique les deux vues. Voir "Synchroniser les modèles logiques et physiques", page 1.

Après synchronisation vous pouvez créer ou modifier manuellement des correspondances dans l'éditeur, mais cette méthode ne garantit plus la cohérence entre les deux modèles. L'assistant de dénormalisation permet de maintenir cette cohérence. Voir "Dénormaliser les modèles logiques et physiques", page 79.

Lancer l'éditeur sur une base de données

Pour ouvrir l'éditeur sur une base de données:

- Cliquez sur l'icône de la base de données et sélectionnez **Éditeur de correspondances**.
  L'éditeur de correspondances juxtapose la vue logique et la vue physique de la base. Lorsqu'un arbre de correspondance existe, il l'affiche automatiquement. Lorsqu'aucun arbre n'a été créé pour la base de données, une fenêtre vous propose de le créer.

Créer un arbre de correspondance logique/physique

Pour créer un arbre de correspondance :

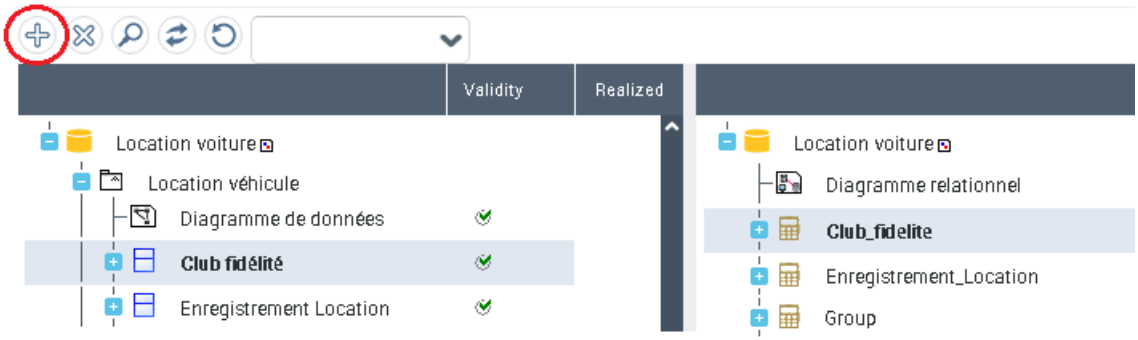
1. Dans la fenêtre de création, indiquez le nom du nouvel arbre de correspondance.
2. Dans le champ **Nature**, précisez le type de correspondance effectué : "Logique/Physique".
3. Dans les cadres **Objet gauche** et **Objet droit**, sélectionnez les modèles logique et physique que vous souhaitez aligner.
4. Cliquez sur **OK**.
   L'éditeur affiche l'arbre de correspondance qui juxtapose les deux modèles.

Créer une correspondance

Pour créer une correspondance entre une entité et une table :

1. Dans l'éditeur de la base de données, sélectionnez l'entité puis la table.

2. Cliquez sur le bouton **Etablir une correspondance**.

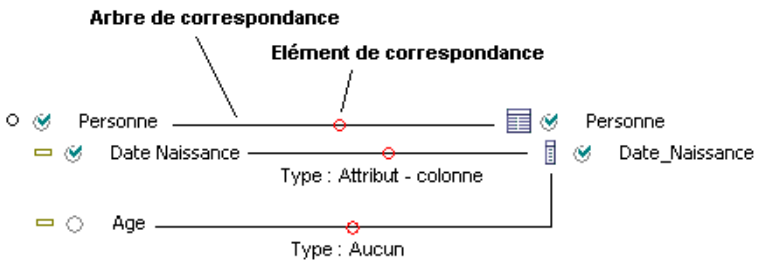


La correspondance se crée à partir du dernier objet sélectionné. Ainsi, pour créer une correspondance du modèle logique vers physique, autrement dit pour définir un objet du modèle physique à partir d'un objet du modèle logique, vous devez sélectionner l'objet du modèle logique puis celui du modèle physique, et créer la correspondance depuis ce dernier. Si une correspondance ne peut être créée, un message d'erreur apparaît (Voir "Sens de la synchronisation", page 19).

Exemple de nouvelle correspondance

Prenons l'entité "Personne" qui contient l'attribut "Date Naissance". Elle a comme correspondance dans le formalisme physique la table "Personne" qui contient la colonne "Date_Naissance".

Supposons que l'on ajoute l'attribut "Age" sur l'entité. Celui-ci peut être calculé depuis la date de naissance. Pour ne pas créer de colonne correspondant à ce nouvel attribut lors de la synchronisation, vous pouvez le relier directement à la colonne "Date_Naissance".



Pour créer une correspondance entre l'attribut "Age" et la colonne "Date_Naissance" :

1. Dans l'éditeur, sélectionnez d'un côté la colonne "Date_Naissance" et de l'autre l'attribut "Age" .
2. Cliquez sur le bouton **Etablir une correspondance**.

Une fois la correspondance créée, une coche apparaît devant l'attribut "Age".

Supprimer une correspondance

Pour supprimer une correspondance sur un objet :

- Sélectionnez l'objet en question et cliquez sur le bouton **Enlever la correspondance**.

DÉTAILS DES CORRESPONDANCES

Les objets qui ont une correspondance sont cochés en vert. Lorsque vous sélectionnez l'un de ces objets, sa correspondance apparaît dans la fenêtre de propriétés de la correspondance située par défaut en bas de l'éditeur de base de données. Elle regroupe le nom des objets reliés dans les deux formalismes, le type des objets et éventuellement un commentaire.

Exemple de correspondance

On sélectionne la table "Client" dans l'arbre de la vue logique :

La correspondance affiche les objets suivants :

Validité	Objet logique	Objet physique	Type
Valide	Client	Client	Entité - Table

Cela signifie que la table "Client" est issue de l'entité du même nom.

Propriétés d'une correspondance

Pour visualiser les propriétés d'une correspondance :

1. Dans l'éditeur de correspondances, sélectionnez la correspondance et cliquez sur **Propriétés**.
2. Dans la fenêtre qui apparaît, cliquez sur la liste déroulante puis sur **Caractéristiques**.

Voir aussi "Caractéristiques de la correspondance", page 19.

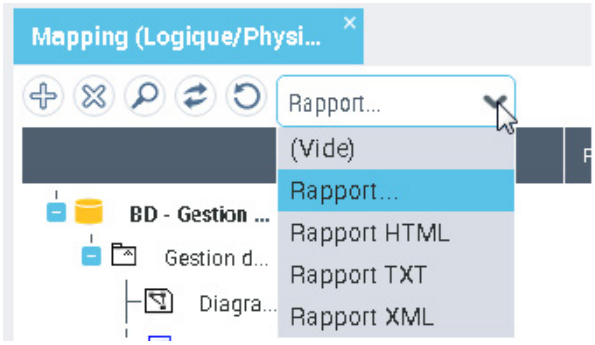
Rapport des correspondances

Vous pouvez générer dans un document le détail des correspondances entre les deux modèles de la base de données. Il peut s'agir d'un fichier HTML, texte ou XML.

Générer un rapport HTML

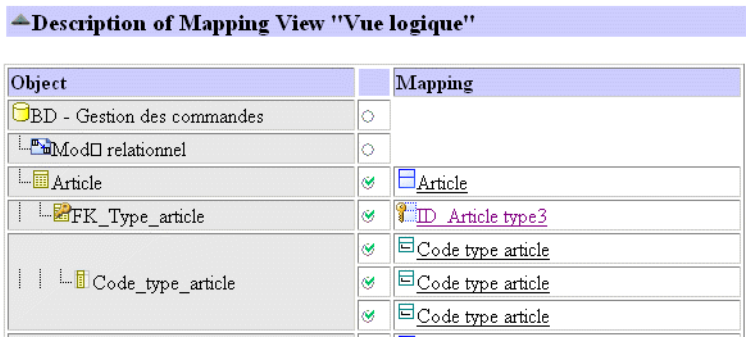
Pour générer le rapport HTML d'un arbre de correspondance :

- Dans la barre d'outils de l'éditeur de la base de données, cliquez sur la liste déroulante puis sur **Rapport HTML**.



Le fichier correspondant s'ouvre. Il présente sous forme de tableau le détail des correspondances de la vue physique d'une part et de la vue logique d'autre part.

Chaque ligne du tableau présente un objet du modèle et son équivalence dans l'autre modèle. Au milieu de chaque ligne apparaît l'état de la correspondance entre les deux objets.



A la fin du document, vous trouvez également la liste des correspondances invalides.

Etat des objets

Des indicateurs permettent d'indiquer l'état des objets mis en correspondance. Une barre de filtrage vous permet d'afficher l'ensemble ou uniquement certains de ces indicateurs. Cette barre est disponible sous le menu **Affichage** > **Barre d'outils** de l'éditeur.

Les objets peuvent être caractérisés par les états suivants :

 Valide

 Invalide (Lorsqu'un objet a conservé une correspondance vers un objet qui n'existe plus)

 Sans correspondance

 Figé (Protégé)

 Standard

Enregistrer l'affichage des indicateurs de l'éditeur

Une option vous permet d'enregistrer l'état des indicateurs dans l'éditeur de correspondance. Elle est propre à l'utilisateur et à l'arbre de correspondance courants. Ainsi, un utilisateur qui a coché l'option d'affichage des indicateurs retrouvera automatiquement l'état des objets dans l'arbre de correspondance qu'il a précédemment créé.

L'option d'affichage des indicateurs de l'éditeur est décochée par défaut. Pour l'activer :

1. Dans le bureau, cliquez sur **Menu principal** > **Paramètres** > **Options**.
2. Dans la partie gauche de la fenêtre des options, cliquez sur **Éditeur de correspondance**.
3. Dans la partie droite, cochez l'option **Enregistrer l'affichage des indicateurs de l'éditeur**.
4. Cliquez sur **OK**.

Source de la correspondance

En sélectionnant un objet dans un des formalismes présentés dans l'éditeur, vous pouvez afficher sa correspondance dans l'autre formalisme.

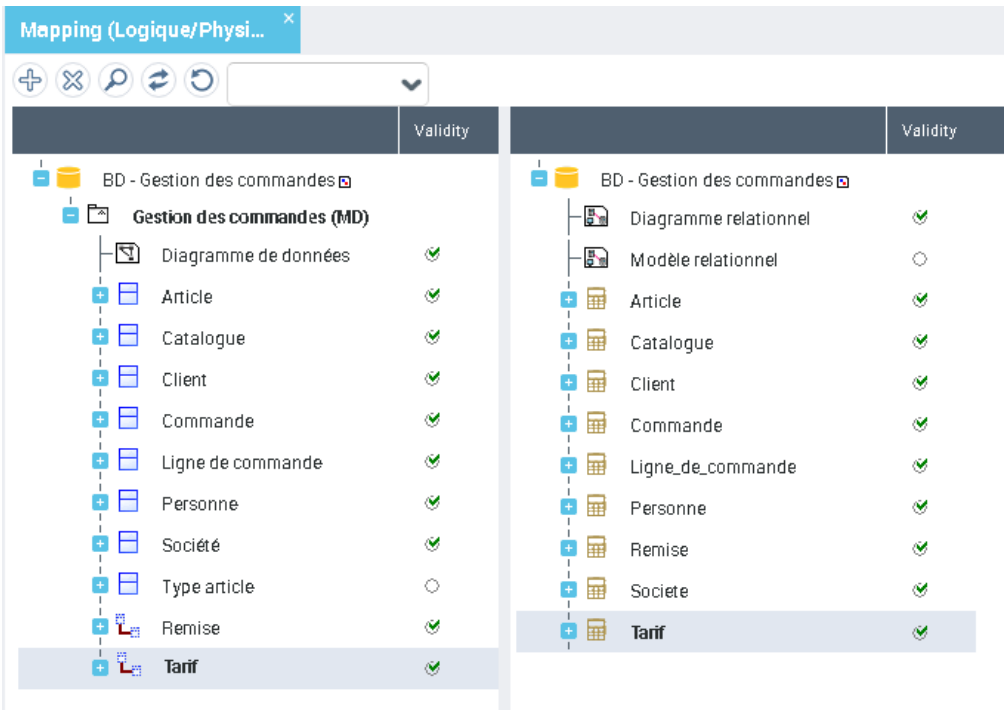
Pour afficher la correspondance d'un objet :

- Dans l'éditeur de correspondances, sélectionnez l'objet en question et cliquez sur le bouton **Chercher**.

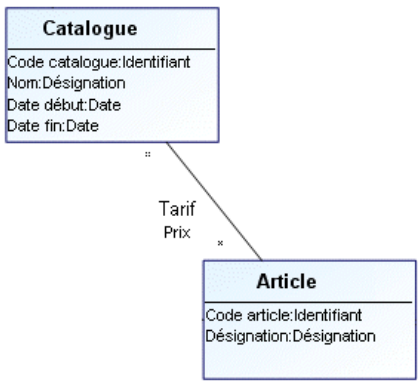
L'éditeur affiche l'objet source.

Exemple de correspondance

Prenons la table "Tarif" créée dans le modèle logique. Dans le menu contextuel de la table, cliquez sur **Chercher**. Vous constatez qu'elle correspond à une association.



En ouvrant le diagramme du modèle logique, vous pouvez voir qu'il s'agit de l'association "Tarif" qui relie les entités "Catalogue" et "Article".



Lors de la synchronisation, cette association non contrainte donne lieu à une table dans laquelle :

- Une colonne est créée pour l'identifiant de chaque entité reliée.
- La clé primaire de la table porte sur l'ensemble de ces colonnes.
- Une clé étrangère est constituée pour chaque entité reliée.

Pour plus d'informations sur la synchronisation des associations, voir "Synchronisation logique > physique : les Associations", page 4.

Dessin de la correspondance

Pour visualiser le dessin d'une correspondance :

- Sélectionnez dans l'arbre de correspondance les objets concernés.

Une fenêtre apparaît en bas de l'éditeur et présente le dessin des objets et les liens de correspondance qui les unissent.

DÉNORMALISER LES MODÈLES LOGIQUES ET PHYSIQUES

Après avoir synchronisé un modèle de données et un modèle physique, vous pouvez les faire évoluer en apportant des modifications directement dans leur diagramme. Mais cette méthode ne garantit pas la cohérence entre les deux modèles.

Les assistants de dénormalisation ont été créés pour maintenir cette cohérence. Il vous permettent de toucher à la définition d'un modèle tout en maintenant les correspondances avec l'autre.

Les assistants vous permettent également d'effectuer rapidement des opérations de type duplication ou fusion d'entités sans report des correspondances et/ou sans suppression des objets sources lorsque vous souhaitez reporter les modifications dans le modèle relationnel.

Dans HOPEX Information Architecture V2R1, la fonctionnalité de dénormalisation est disponible uniquement pour le modèle de données, elle ne s'applique pas aux paquetages (notation UML).

Les points suivants sont abordés ici :

PRINCIPES DE LA DÉNORMALISATION

La dénormalisation permet de transformer ou d'affiner les modèles en fonction de besoins spécifiques : choix de modélisation, optimisation des performances, choix de mise en œuvre physique, redondances etc.

L'outil de dénormalisation est présenté sous forme d'assistants permettant de réaliser ces transformations.

Un assistant s'applique à un objet ou à un groupe d'objets. Selon le type d'optimisation demandé, la dénormalisation crée dans un modèle de nouveaux objets (objets cibles) à partir des objets de départ (objets sources).

Dénormalisation : cohérence des modèles

La dénormalisation effectue un changement sur l'objet d'un modèle. Lorsque ce modèle a été mis en correspondance avec un autre (voir la "Synchroniser les modèles logiques et physiques", page 1), vous devez gérer l'impact du changement sur l'autre modèle.

Lors de la dénormalisation, vous pouvez ainsi :

- Reporter ou non les correspondances avec les objets synchronisés.

- Supprimer ou conserver les objets sources.

Report des correspondances

Le report des correspondances permet d'assurer la stabilité et la cohérence entre deux modèles. Lorsqu'une dénormalisation crée un nouvel objet au niveau logique, la correspondance avec l'objet physique est reportée sur le nouvel objet logique. Lorsque vous décochez cette option, les correspondances vers les nouveaux objets ne sont pas créées, les deux modèles doivent donc être re-synchronisés.

Afin que la synchronisation valide les changements occasionnés par la dénormalisation, veillez à ce que l'option "Réinitialisation des objets cibles" soit décochée.

Supprimer les objets sources

Par défaut, les objets sources sont supprimés. La non suppression des objets sources vous permet de conserver les objets de départ après la dénormalisation.

Synchronisation et dénormalisation

Dans l'activité de modélisation de données, synchronisation et dénormalisation doivent souvent être combinées pour répondre à tel ou tel cas d'emploi.

Exemple

Soit une entité PAIEMENT que vous voulez représenter dans la base de données par trois tables VIREMENT, CHEQUE et AUTRE. Pour réaliser cette modélisation :

1. Créez l'entité PAIEMENT.
2. Lancez la synchronisation (logique vers physique) pour obtenir la table PAIEMENT.
3. Lancez l'assistant physique pour partitionner horizontalement la table PAIEMENT.
4. Renommez les trois duplicatas en VIREMENT, CHEQUE et AUTRE.

Les trois tables ainsi obtenues sont dès lors liées à l'entité PAIEMENT et suivent ses évolutions lors des futures synchronisations.

Combinaison des options de dénormalisation et synchronisation

L'impact d'une dénormalisation sur deux modèles synchronisés varie en fonction des options choisies.

Prenons l'exemple d'une dénormalisation logique. Les combinaisons possibles sont :

- Suppression des objets sources + report des correspondances : les objets sources du modèle logique sont supprimés. Le niveau physique reste inchangé ; la correspondance avec les objets logiques résultant de la dénormalisation est assurée.
- Suppression des objets sources + non report des correspondances : les objets physiques correspondant aux objets logiques résultant de la dénormalisation sont créés. Les objets physiques correspondant aux objets logiques supprimés sont supprimés.
- Non suppression des objets sources + report des correspondances : les objets sources du modèle logique sont conservés. Le niveau physique reste inchangé; la correspondance avec les objets résultant de la dénormalisation est assurée.
- Non suppression des objets sources + non report des correspondances : les objets physiques correspondant aux objets résultant de la dénormalisation sont créés. Les objets physiques existants sont conservés.

> **Cas particuliers : fusions ascendante et descendante :**
> **Dans le cas d'une fusion ascendante, l'entité sur-type a un rôle particulier. La dénormalisation conserve sa correspondance dans tous les cas.**
> **De même, dans le cas d'une fusion descendante, les entités sous-types jouent un role particulier; leur correspondance est conservée dans tous les cas.**

Dénormalisation : cas d'emploi

La combinaison des options de dénormalisation varie en fonction du mode de conception de vos modèles.

1. Préserver la stabilité du niveau physique lorsqu'une modification est appliquée au niveau logique.

Contexte : une synchronisation a déjà été établie entre les niveaux logique et physique. Le niveau physique est en production. Une modification doit être appliquée au niveau logique, sans impact sur le niveau physique.

Options préconisées pour la dénormalisation : report des correspondances, suppression des objets sources.

Ce cas d'emploi correspond à un mode de travail orienté maintenance anticipée : des modifications sont effectuées par anticipation sur le niveau logique, sachant que le niveau physique ne doit pas être modifié jusqu'à nouvel ordre.

Résultat : après la dénormalisation, les correspondances sont rétablies entre les objets logiques cibles et les objets physiques. Lors de la synchronisation, rien ne change au niveau physique.

2. Faire évoluer le niveau physique lorsqu'une dénormalisation est appliquée au niveau logique.

Contexte : une synchronisation a déjà été établie entre les niveaux logique et physique. Le niveau physique n'est pas figé et doit évoluer en fonction du niveau logique.

Options préconisées pour la dénormalisation : non report des correspondances, suppression des objets sources.

Ce cas d'emploi permet de privilégier les évolutions du niveau logique, sans se préoccuper de l'impact au niveau physique.

Résultat : après la dénormalisation, les objets logiques cibles sont sans correspondance; les objets physiques correspondant aux objets logiques sources ne sont plus synchronisés. Après synchronisation, le niveau physique est mis à jour : les objets physiques correspondant aux objets logiques sources (objets existants avant la dénormalisation) sont supprimés ; de nouveaux objets physiques correspondant aux objets logiques cibles (objets créés par la dénormalisation) sont créés.

3. Faciliter le développement du niveau logique, en phase de développement.

Contexte : une synchronisation a déjà été établie entre les niveaux logique et physique, ou bien le niveau physique n'est pas encore implémenté.

Options préconisées pour la dénormalisation : non report des correspondances, non suppression des objets sources.

Ce cas d'emploi correspond à un mode de travail "incrémental" : les objets sources du niveau logique restent inchangés. Le modèle est enrichi des objets cibles issus de la dénormalisation. Ces objets cibles donnent lieu à une partie nouvelle dans le niveau physique, la partie physique existante restant stable.

Résultat : après la dénormalisation, les correspondances restent inchangées. Après synchronisation, de nouveaux objets physiques correspondant aux nouveaux objets logiques sont créés ; les objets physiques correspondant aux objets logiques sources sont inchangés.

4. Favoriser la mise en place de scenarios multiples, en phase de développement.

Contexte : une synchronisation a déjà été établie entre les niveaux logique et physique, plusieurs options de modélisation coexistent temporairement, pour un seul niveau physique.

Options préconisées pour la dénormalisation : report des correspondances, non suppression des objets sources.

Ce cas d'emploi, à utiliser avec précaution, permet de conserver deux options de modélisation au niveau logique, donnant lieu à un résultat commun au niveau physique.

Résultat : après la dénormalisation, les objets physiques restent reliés aux objets logiques sources, et sont également reliés aux objets logiques cibles. Après synchronisation, les objets du niveau physique restent inchangés.

DÉNORMALISATION LOGIQUE

La dénormalisation logique s'applique aux entités (ou classes) et attributs du modèle de données.

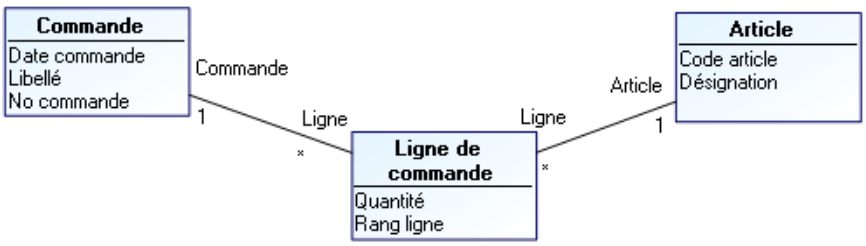
Lancer une dénormalisation logique

Pour dénormaliser le formalisme logique :

1. Faites un clic droit sur la base de données à laquelle est associé le modèle de données et sélectionnez **Dénormalisation logique**.
   Un assistant apparaît.
2. Dans le champ **Sélectionnez le type de dénormalisation**, sélectionnez le type de dénormalisation voulu et suivez les étapes de l'assistant. Voir "Liste des assistants de dénormalisation logique", page 86.

Exemple de dénormalisation logique

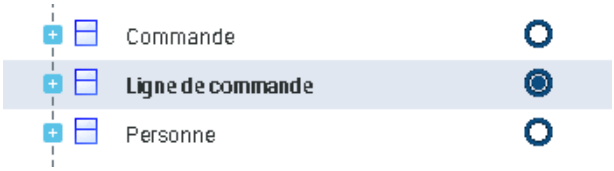
Supposons que vous vouliez transformer l'entité "Ligne de commande" en une association.



Pour transformer cette entité en association :

1. Faites un clic droit sur la base de données "Gestion des commandes" qui contient cette entité et sélectionnez **Dénormalisation logique**.
   Un assistant apparaît.
2. Dans le champ **Sélectionnez le type de dénormalisation**, sélectionnez "Transformer une entité en association".
3. Cliquez sur **Suivant**.

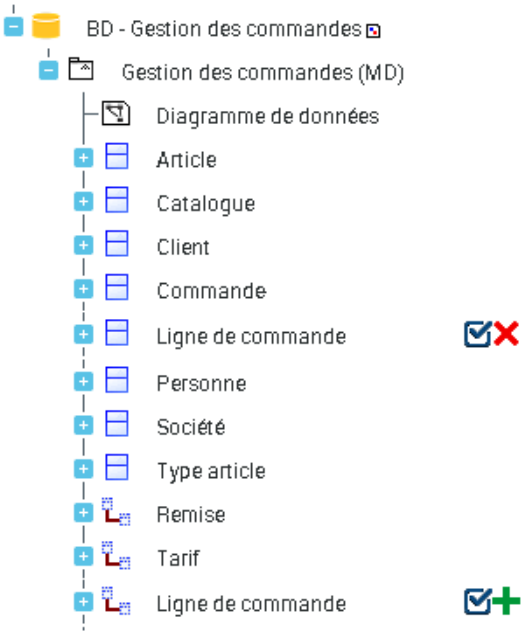
4. Dans l'arbre de l'éditeur, cochez la colonne **Périmètre** face à l'entité "Ligne de commande" afin de la sélectionner.



☛ *Vous avez la possibilité de sélectionner plusieurs entités lors d'une dénormalisation.*

5. Cliquez sur **Suivant**.
Les options de dénormalisation apparaissent. Par défaut, le report des correspondances et la suppression des objets sources sont activés. Cela signifie que l'entité "Ligne de commande" va être supprimée et que le lien de correspondance avec la table "Ligne de commande" va être reporté sur l'association qui la remplace.

6. Cliquez sur **Suivant**.
L'éditeur affiche les changements occasionnés par cette dénormalisation. Vous pouvez voir que l'entité "Ligne de commande" va être supprimée et que l'association "Ligne de commande" va être créée.



☛ *Lorsqu'une entité sélectionnée ne peut être transformée, l'éditeur vous en explique la raison.*

Vous pouvez refuser une modification en décochant la case correspondante.

7. Validez les résultats en cliquant sur **Suivant**.
Cette transformation est définitive et sera prise en compte par la prochaine synchronisation.
A la fin de la dénormalisation, vous pouvez voir que l'association "Ligne de commande" qui remplace l'entité est désormais en correspondance avec la table "Ligne de commande".

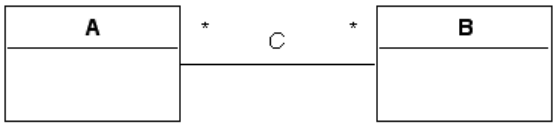
> ☛ *Si un objet est protégé, il n'est pas possible de le sélectionner pendant la dénormalisation.*

Liste des assistants de dénormalisation logique

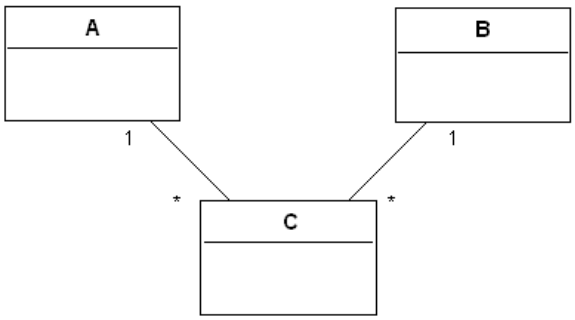
Transformer une association en entité

Cette dénormalisation permet de transformer une association n-aire, quelles que soient ses multiplicités, en entité. Une association de multiplicité '*,1' est créée entre cette entité et les entités de l'association.

Avant



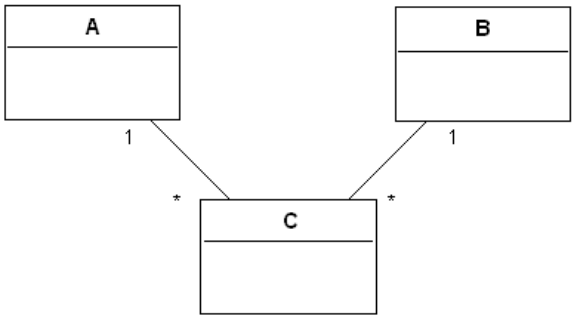
Après



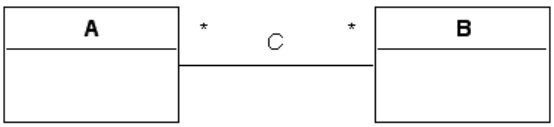
Transformer une entité en association

Cette dénormalisation permet de transformer une entité qui possède n associations binaires dont les rôles opposés sont de multiplicité '1' en association. Une nouvelle association n-aire de multiplicité * est créée entre ces entités.

Avant



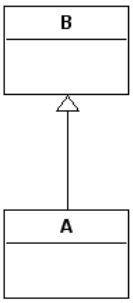
Après



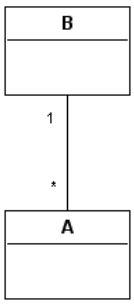
Transformer une généralisation en association

Cette dénormalisation permet de transformer une généralisation entre deux entités en association. Une association de multiplicité *,1 est créée entre les deux entités, et la généralisation est supprimée.

Avant



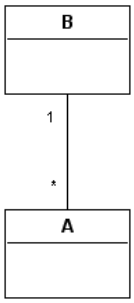
Après



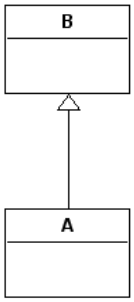
Transformer une association en généralisation

Cette dénormalisation permet de transformer une association 1,* en généralisation. Une généralisation est créée entre les deux entités, et l'association est supprimée.

Avant



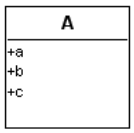
Après



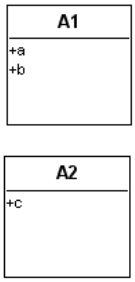
Partition verticale d'une entité

Cette dénormalisation permet de diviser une entité en plusieurs entités. Les entités obtenues se répartissent les attributs, les associations et les généralisations.

Avant



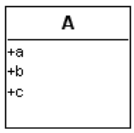
Après



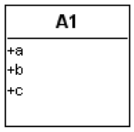
Partition horizontale d'une entité

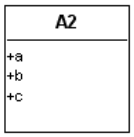
Cette dénormalisation permet de dupliquer une entité.

Avant



Après





Partition horizontale et synchronisation en mode Logique > Physique

Prenons une entité "Catalogue". Après partition horizontale, cette entité donne deux entités "Catalogue-1" et "Catalogue-2".

Après synchronisation du mode Logique > Physique, les deux entités ont pour correspondance une table.

Si vous affichez les propriétés des correspondances, vous remarquez que les deux sont bidirectionnelles, ce qui signifie que entités et table se mettent à jour dans les deux sens de la synchronisation.

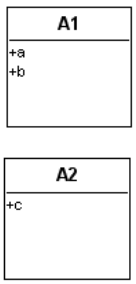
Si vous effectuez des modifications logiques sur ces entités puis relancez la synchronisation, l'éditeur affiche un signal sur la table cible ; en effet il ne sait quelle entité prendre pour effectuer les mises à jour.

Lorsque vous choisissez une entité, celle-ci est conservée comme entité de référence (par exemple Catalogue-1). L'autre entité sera conservée dans un seul sens, autrement dit Catalogue-2 pourra être mise à jour dans le modèle logique mais n'aura pas d'impact sur la table.

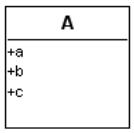
Fusion d’entités

Cette dénormalisation permet de fusionner des entités.

Avant



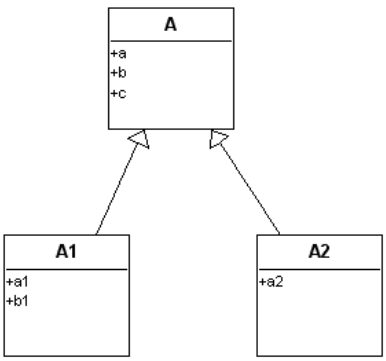
Après



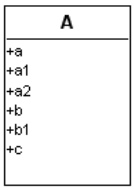
Fusion d’entités ascendantes

Cette dénormalisation permet de fusionner une entité avec son entité mère : tous les attributs et les liens sont reportés sur l'entité parente et l'entité fille est supprimée.

Avant



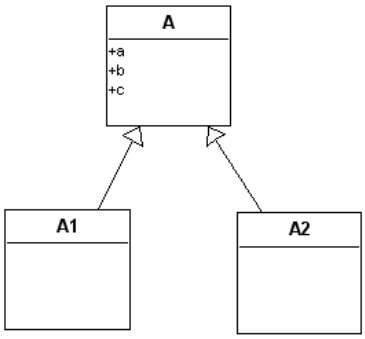
Après



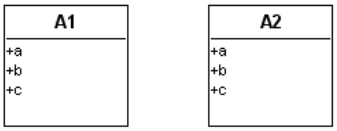
Fusion d'entités descendantes

Cette dénormalisation permet de fusionner une entité avec son entité fille : tous les attributs et les liens sont reportés sur l'entité fille et l'entité parente est supprimée.

Avant



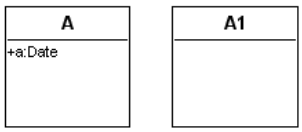
Après



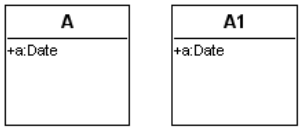
Copier/Coller d'attributs

Cette dénormalisation permet de reporter des attributs d'une entité ou d'une association vers d'autres entités ou d'autres associations.

Avant



Après



DÉNORMALISATION PHYSIQUE

La dénormalisation physique s'applique aux objets de la base de données représentés par les tables, les colonnes, les clés et les index.

Lancer une dénormalisation physique

Pour dénormaliser le formalisme physique :

1. Faites un clic droit sur la base de données concernée et sélectionnez **Dénormalisation physique**.

Un assistant présente l'ensemble des personnalisations possibles sur les objets de la base de données.

Exemple de dénormalisation physique

Il est possible de faire en sorte qu'une table se divise en deux tables distinctes, soit pour séparer les colonnes dans deux tables (partition verticale), soit pour dupliquer les informations d'une table dans deux autres (partition horizontale).

Prenons l'exemple d'une prise de commande. Cette prise de commande est représentée par une entité au niveau logique, elle donne une table au niveau physique. Supposons que cette prise de commande soit amenée à être gérée différemment au niveau technique selon qu'il s'agisse d'une commande par téléphone ou d'une commande par Internet. On peut intégrer ce changement dans le modèle physique sans être obligé de modifier en parallèle le modèle logique. Pour cela, on va faire en sorte que l'entité représentant la prise d'une commande se divise en deux tables; une table pour les commandes par téléphone, une autre pour les commandes Internet. En intégrant cette modification depuis l'assistant, la partition devient automatique lors de chaque synchronisation, et les deux modèles restent cohérents.

Pour créer une partition horizontale telle que celle décrite ci-dessus :

1. Faites un clic droit sur la base de données "Gestion des commandes" qui contient la table "Commande" et sélectionnez **Dénormalisation physique**.
   Un assistant apparaît.
2. Dans le champ **Sélectionnez le type de dénormalisation**, sélectionnez «Partition horizontale d'une table».
3. Cliquez sur **Suivant**.
4. Dans l'arbre de l'éditeur, sélectionnez la table que vous voulez dupliquer, ici "Commande".
5. Cliquez sur **Suivant**.
   Les options de dénormalisation apparaissent. Par défaut, le report des correspondances et la suppression des objets sources sont activés. Cela signifie que la table "Commande" va être supprimée et que le lien de

correspondance avec l'entité "Commande" va être reporté sur les deux tables.

Précisez le nombre de partitions, autrement dit le nombre de tables créées. Par défaut, l'outil en crée deux.

6. Cliquez sur **Suivant**.
L'éditeur affiche les changements occasionnés par cette dénormalisation. Vous pouvez voir que la table "Commande" va être supprimée et que deux nouvelles tables vont être créées.

7. Validez les résultats en cliquant sur **Suivant**.
Cette transformation est définitive et sera prise en compte par la prochaine synchronisation.

A la fin de la dénormalisation, vous pouvez voir que les deux nouvelles tables sont désormais en correspondance avec l'entité "Commande".

> **La dénormalisation s'applique ici au modèle physique. La synchronisation qui prendra en compte cette personnalisation devra donc se faire dans le même sens, autrement dit du modèle logique vers le modèle physique. Elle n'est pas valable dans l'autre sens.**

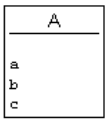
Liste des assistants de dénormalisation physique

Partition verticale d'une table

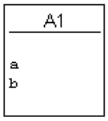
Cette dénormalisation permet de diviser une table en plusieurs tables. Les tables obtenues se répartissent les colonnes.

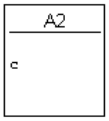
Seules les colonnes ne faisant pas partie d'une clé peuvent être réparties entre les tables.

Avant



Après

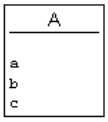




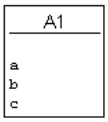
Partition horizontale d'une table

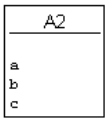
Cette dénormalisation permet de dupliquer une table. Les deux tables obtenues contiennent l'ensemble des colonnes de la table d'origine.

Avant



Après

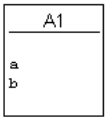


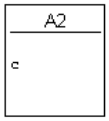


Fusion de tables

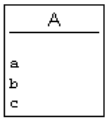
Cette dénormalisation permet de fusionner des tables.

Avant





Après



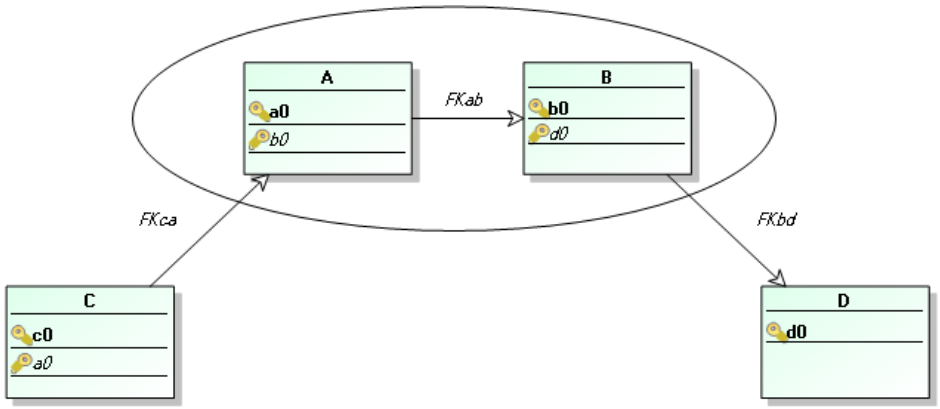
Option des clés primaires

Lorsque vous lancez une fusion de tables, une option vous permet de déterminer la clé primaire de la table fusion : vous pouvez sélectionner une des clés primaires des tables sources ou fusionner l'ensemble des clés primaires des tables sources.

Lorsque vous sélectionnez une clé primaire pour la table fusion, seules les clés étrangères qui font référence à cette clé primaire sont reportées. Les clés étrangères faisant référence aux clés primaires qui ne sont pas reportées ne sont pas prises en compte.

Ainsi, dans l'exemple suivant, si vous fusionnez les tables A et B et conservez la clé primaire de la table A, la clé primaire de B disparaît lors de la fusion. La clé étrangère FKab n'est pas non plus reportée puisqu'elle fait référence à la clé primaire de B.

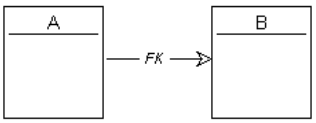
Les autres clés étrangères, FKca et FKbd, sont bien reportées dans le diagramme relationnel.



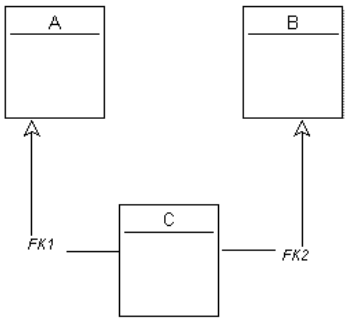
Transformation d'une clé étrangère en table

Cette dénormalisation permet de transformer une clé étrangére en table. Une nouvelle table est créée, ainsi que deux nouvelles clés étrangères. La clé étrangère d'origine est supprimée.

Avant



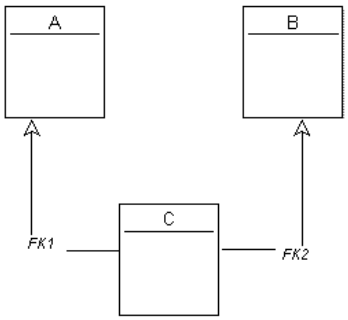
Après



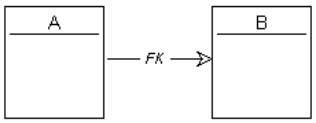
Transformation d'une table en clé étrangère

Cette dénormalisation permet de transformer une table en clé étrangère. La table, ainsi que ses deux clés étrangères sont supprimées, et une nouvelle clé étrangère est créée.

Avant



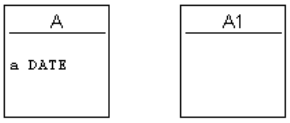
Après



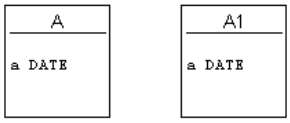
Copier/Coller des colonnes

Cette dénormalisation permet de reporter des colonnes d'une table vers une autre.

Avant



Après



TYPES DES ATTRIBUTS ET DES COLONNES

Les données n'ont pas toutes le même type de valeur. Déterminer le type des données permet d'indiquer leur format et ainsi de favoriser leur manipulation par les différents outils de traitement de données.

HOPEX gère les types de données aux différents niveaux de modélisation, en assurant la correspondance entre les types de données du niveau logique et les types de données pris en charge par les différents SGBD supportés.

Les points suivants sont abordés ici :

TYPES DE DONNÉES DES ATTRIBUTS

Un type de données permet de mettre en commun des caractéristiques communes à plusieurs attributs.

Pour typer les attributs d'une entité, ne sont proposés que les types de données définis pour le *modèle de données* qui contient cette entité.

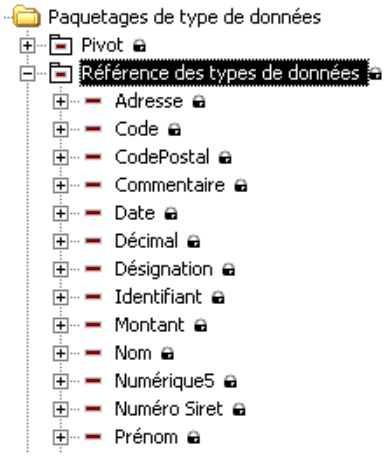
> 🕮 *Un modèle de données permet de représenter la structure statique d'un système, en particulier les types d'objets manipulés dans le système, leur structure interne et les relations qui existent entre eux. Un modèle de données regroupe un ensemble d'entités avec leurs attributs, les associations qui existent entre ces entités, des contraintes qui portent sur ces entités et associations, etc.*

Paquetages de types de données

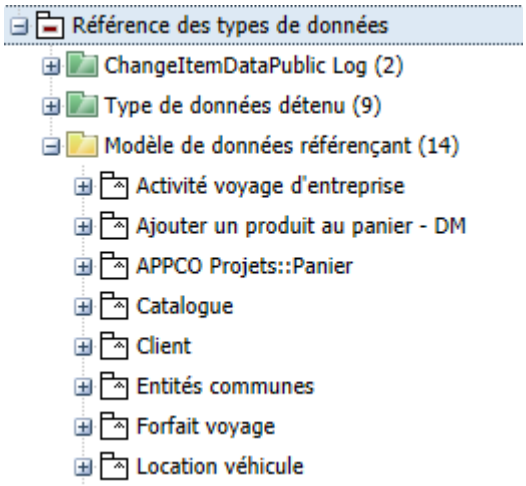
> 🕮 *Un paquetage de types de données est un paquetage de référence détenant tout ou partie des types de données utilisés dans l'entreprise. Chacun des autres paquetages sera déclaré client du paquetage de référence des types de données.*

Lorsque vous créez un modèle de données, le *paquetage de types de données* "Référence des types de données" lui est automatiquement associé par défaut.

Ce paquetage "Référence des types de données" détient les types de données standard "Adresse", "Code", "Date", etc.



En ouvrant l'explorateur sur ce paquetage de type de données, vous pouvez voir qu'il est référencé par plusieurs modèles de données.



Les attributs des entités de ces modèles peuvent donc être typés à l'aide des types de données "Adresse", "Code", "Date", etc.

Créer un nouveau paquetage de types de données

Vous pouvez définir un nouveau paquetage de types de données de référence détenant les types de données utilisés dans l'entreprise.

Pour créer votre propre paquetage de types de données :

1. Dans le bureau, cliquez sur le menu de navigation puis sur **Données logiques**.
2. Dans le volet de navigation, cliquez sur le dossier **Paquetages de types de données**.
   La liste des paquetages de types de données du référentiel apparaît dans la fenêtre d'édition.
3. Cliquez sur **Nouveau**.
   La fenêtre de création d'un paquetage de type de données apparaît.
4. Saisissez son nom et éventuellement un détenteur.
5. Cliquez sur **OK**.

Vous pouvez ensuite ajouter des types à ce paquetage.

Créer un type de données

Pour créer un type de données :

1. Faites un clic droit sur le paquetage de types de données et sélectionnez **Propriétés**.
   La fenêtre de propriétés du paquetage apparaît.
2. Cliquez sur la liste déroulante puis sur **Type de données**.

3. Cliquez sur **Nouveau**.
   La fenêtre de création d'un type de données apparaît.
4. Saisissez le nom du type et cliquez sur **OK**.

Type de données composé

Vous pouvez créer des types de données composés en leur ajoutant une liste d'attributs, par exemple un type "Adresse" composé du numéro, de la rue, du code postal, de la ville et du pays.

Valeur littérale

Vous pouvez affecter des valeurs littérales à un type de données qui définissent les valeurs qu'il peut prendre. Les attributs basés sur un tel type de données ne peuvent prendre que les valeurs définies par le type de données.

Une fois le nouveau paquetage de types de données créé, il convient de le référencer sur le modèle de données client.

Référencer un paquetage de types de données

Lorsque vous définissez un nouveau paquetage de types de données, vous devez le relier au modèle de données concerné.

Pour relier un paquetage de types de données à un modèle de données :

1. Dans le bureau, cliquez sur le menu de navigation puis sur **Données logiques**.
2. Dans le volet de navigation, cliquez sur le dossier **Tous les modèles de données**.
   La liste des modèles de données apparaît dans la zone d'édition.
3. Faites un clic droit sur le modèle de données concerné et sélectionnez **Propriétés**.
   La fenêtre de propriétés du modèle de données apparaît.
4. Cliquez sur la liste déroulante puis sur **Caractéristiques**.
5. Dans le champ **Reference**, cliquez sur la flèche puis sur **Relier Paquetage**.
   La fenêtre de recherche apparaît.
6. Cliquez sur **Chercher**.
   La liste des paquetages de types de données s'affiche.
7. Sélectionnez le paquetage voulu et cliquez sur **OK**.

Affecter des types aux attributs

Une fois le paquetage de types de données référencé pour le modèle de données, la liste des types qu'il contient est disponible sur chaque attribut des entités du modèle. Il vous reste à sélectionner celui qui convient.

Pour définir le type d'un attribut :

1. Faites un clic droit sur l'entité qui contient l'attribut et sélectionnez **Propriétés**.
   La fenêtre de propriétés de l'entité apparaît.
2. Cliquez sur la liste déroulante puis sur **Attributs**.
3. Dans la colonne **Type de données (MD)** qui correspond à l'attribut, sélectionnez le type voulu dans la liste.
4. Cliquez sur **Appliquer**.

DÉDUIRE LES DATATYPES DES COLONNES À PARTIR DES TYPES DES ATTRIBUTS

Les types de données définis au niveau logique ne sont pas toujours compréhensibles pour le SGBD cible. Ils nécessitent dans ce cas d'être convertis en types de données correspondant au SGBD visé.

Cette conversion intervient notamment lors de la synchronisation. Les types de données des attributs définis dans le modèle logique sont traduits en datatypes pour les colonnes générées.

La conversion est assurée par un lien d'équivalence avec des types pivots. Les types pivots constituent un intermédiaire entre les types de données logiques et les datatypes générés.

Types pivots

Les types pivots sont des types de données définis indépendamment du SGBD cible et que vous pouvez utiliser quand vous ne connaissez pas encore le système dans lequel sera hébergée la base de données, ou quand plusieurs systèmes sont susceptibles d'être utilisés.

Les types pivots ont un datatype équivalent dans chaque SGBD supporté. Ils permettent ainsi de définir une seule fois les types des attributs et de les réinterpréter ensuite en fonction du SGBD cible.

Pour disposer des datatypes d'un SGBD, vous devez importer le solution pack correspondant. Voir "Importer une version de SGBD", page 9.

Liste des types pivots

Une fois importés, les types pivots sont disponibles dans l'onglet de navigation **Données logiques**, sous le paquetage de types de données "Pivot".

Types alphanumériques		Compléments
P-String	Chaîne de caractères alphanumériques	
P-Text	Chaîne de caractères alphanumériques	
P-Character	Chaîne de caractères alphanumériques de taille fixe	Longueur
P-Varchar	Chaîne de caractères alphanumériques de taille variable	

Types numériques

P-Decimal	Décimal	
P-Double		
P-Float		
P-Integer	Entier	
P-Long Integer		
P-Long Real		
P-Real		
P-Smallint		
P-Tinyint		
P-Numeric	Numérique	Longueur, Décimale
P-Currency	Montant exprimé en monnaie	Longueur, Décimale

Types dates

P-Date	Date
P-Time	Heure
P-DateTime	Date et heure

Types binaires

P-Binary	Chaîne binaire
P-Byte	Chaîne binaire
P-Timestamp	Identification générée automatiquement à partir de la date et de l'heure exprimée en millièmes de secondes après le 01 Janvier 1970
P-Boolean	Booléen valant 0 ou 1
P-Multimedia	Chaîne binaire
P-Varbinary	Chaîne binaire

Relier un type de données à un type pivot

Les types de données contenus dans le paquetage "Reférence des types de données" et associés par défaut à tout nouveau modèle de données sont reliés à ces types pivots. Aussi, lorsque vous créez de nouveaux types de données, il est nécessaire de les relier aux types pivots correspondants afin qu'ils puissent être exploités par la suite au niveau physique.

Pour relier un type de données à un type pivot :

1. Faites un clic droit sur le type de données et sélectionnez **Propriétés**. La fenêtre de propriétés du type de données apparaît.
2. Cliquez sur la liste déroulante puis sur **Caractéristiques**.
3. Dans le champ **SQL Datatype**, sélectionnez le type pivot.

Prenons le type de données "Code". Ouvrez sa fenêtre de propriétés et cliquez sur la page **Caractéristiques**. Dans le champ **SQL Datatype**, vous pouvez voir qu'il est relié au type pivot "P-Character.

Lors de la synchronisation d'un modèle logique vers un modèle physique, ce type pivot "P-Character" donnera un datatype CHAR, VARCHAR, LONG ou TEXT suivant le SGBD concerné par la synchronisation. Vous pouvez modifier le SGBD cible sans avoir à modifier le type de données, **HOPEX** assure la conversion automatique. Voir "Correspondances entre types pivots et datatypes", page 110.

Relier un type de données à un type pivot dans la notation UML

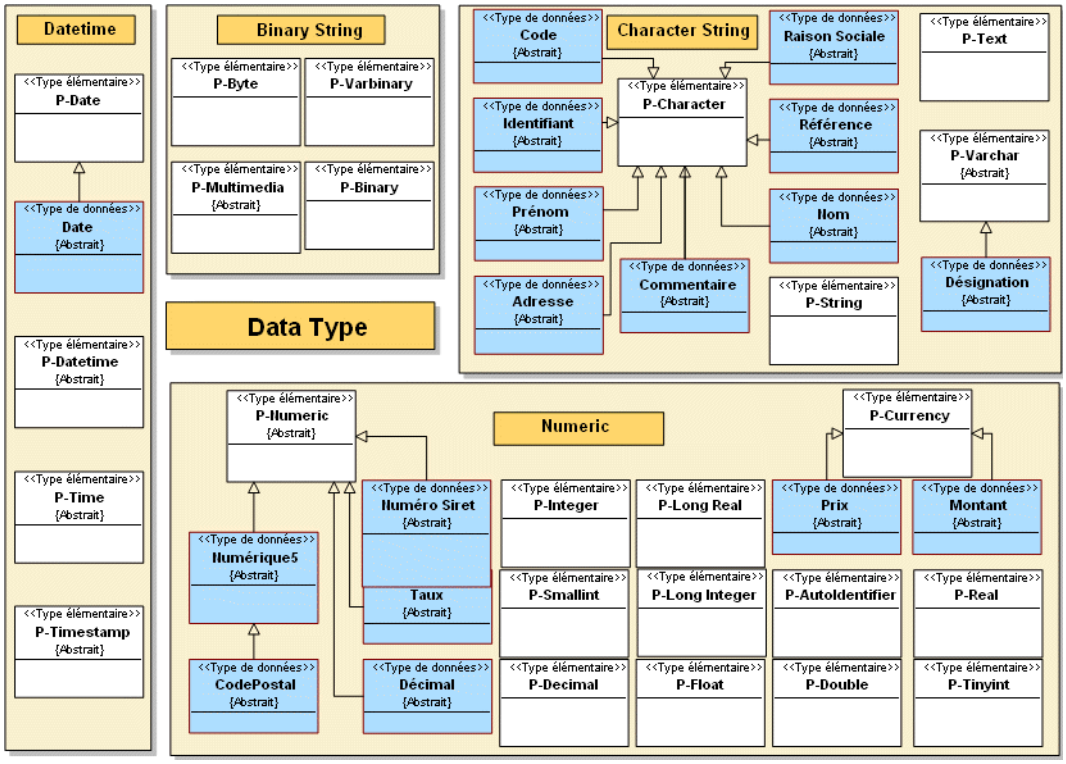
Si vous utilisez la notation UML et les diagrammes de classes pour modéliser vos données - et à titre de compatibilité avec les versions antérieures de **HOPEX Database Builder** - d'autres méthodes de référencement des types pivots sont possibles.

Vous pouvez créer de nouveaux types de données et les relier à des types pivots :

- Par héritage
- Par un lien de correspondance
- Par un lien d'équivalence
- En créant un type de données composé

Par héritage

Vous pouvez définir vos propres types de données en les déclarant sous-classes des types pivots proposés en standard comme dans l'exemple ci-dessous.



Les types de données définis comme sous-classes vont hériter automatiquement des caractéristiques de leur super-classe. En particulier, la règle de transformation en datatype de la super-classe est appliquée à la sous-classe.

Il est possible de préciser sur la sous-classe une longueur et un nombre de décimales. Ceux-ci seront pris en compte pour la génération des datatypes s'ils n'ont pas déjà été définis pour la super-classe.

Par un lien de correspondance

Pour effectuer ce lien :

1. Ouvrez la fenêtre de propriétés de la classe.
2. Cliquez sur la liste déroulante puis sur **Génération** > **SQL**.
3. Indiquez le **Type SQL** associé à la classe.

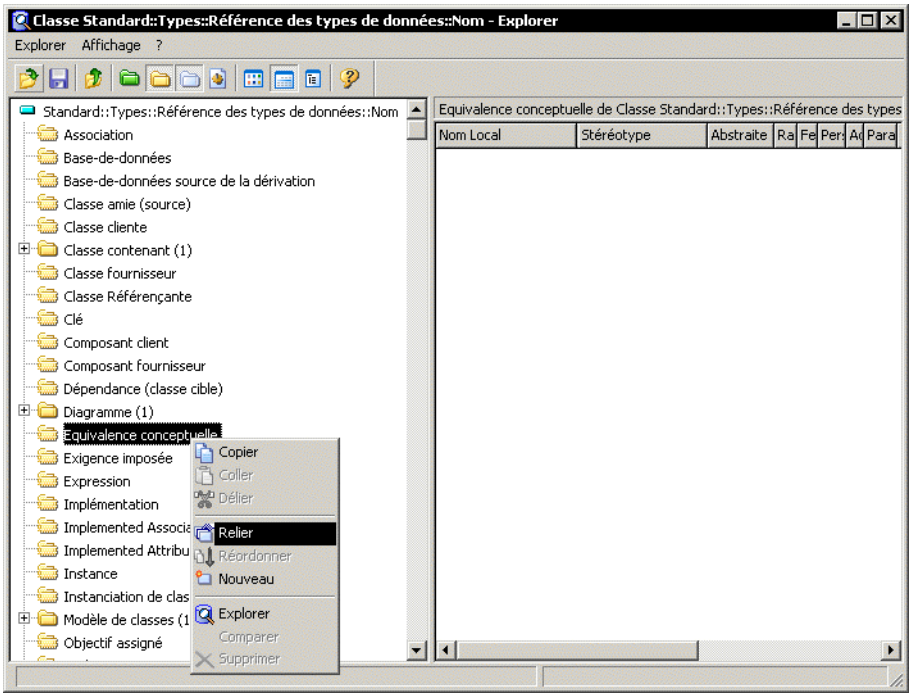
   ☛ *Seuls les types pivots du paquetage Standard::Types::Pivot sont proposés dans la liste.*
4. Précisez si nécessaire la longueur et le nombre de décimales à appliquer.

Par un lien d'équivalence

Il est possible de définir un nouveau type de données en créant un lien d'équivalence avec un type pivot. Pour cela :

1. Créez un nouveau type de données.
2. Faites un clic droit sur le type et sélectionnez **Explorer**.
3. Dans la fenêtre qui apparaît, cliquez sur le bouton **Collections vides**.

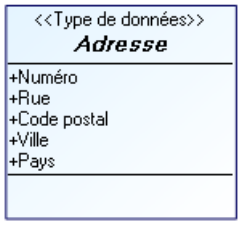
   ☛ *Si besoin, cliquez sur le bouton*  *pour afficher les commandes cachées.*
4. Cliquez avec le bouton droit sur le dossier "Equivalence conceptuelle" et sélectionnez **Relier**.



5. Dans la boîte standard de recherche, sélectionnez le type pivot que vous souhaitez relier à votre type.

En créant un type de données composé

On peut définir un type de données composé en lui précisant une liste d'attributs.

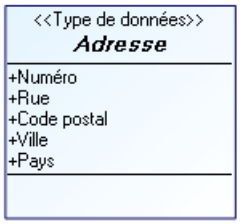


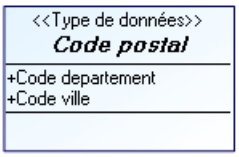
Ici le type "Adresse" est composé du numéro, de la rue, du code postal, de la ville et du pays.

Un attribut de type "Adresse" donnera lieu lors de la dérivation à ces cinq colonnes.

Il est possible de décomposer un type à plusieurs niveaux en affectant un type décomposé à l'un de ses attributs.

Par exemple, on peut décomposer le code postal en code ville et code département.





CORRESPONDANCES ENTRE TYPES PIVOTS ET DATATYPES

Les types pivots établissent une correspondance entre les types de données logiques auxquels ils sont reliés et les datatypes pour lesquels ils ont un équivalent dans chaque SGBD cible.

Les liens d'équivalence comportent des conditions qui permettent de les distinguer les uns des autres.

Pour visualiser les correspondances entre les types pivots et les datatypes des différents SGBD supportés, voir "Tableaux de correspondances entre types pivots et datatypes", page 171.

Exemple de correspondances entre les types pivots et les datatypes pour Oracle 8

Pivot vers Datatype

Pivot	Condition	Datatype
P-AutoIdentifier		NUMBER
P-Binary		RAW(@L)
P-Boolean	L<2 or L ø	RAW(1)
	L>1	RAW(@L)
P-Byte		RAW(1)
P-Character	L<256 or L ø	CHAR(@L)
	L>2000	LONG
	255<L<2001	VARCHAR2(@L)
P-Currency		NUMBER(@L,@D)
P-Date		DATE
P-Datetime		DATE
P-Decimal		NUMBER(@L,@D)
P-Double		NUMBER(@L,@D)
P-Float		NUMBER(@L,@D)
P-Integer		NUMBER(@L)
P-Long Integer		NUMBER(@L)

Pivot	Condition	Datatype
P-Long Real		NUMBER(@L,@D)
P-Multimedia		LONG RAW
P-Numeric	L=0 or L ø	NUMBER
	L>0 et D ø	NUMBER(@L)
	L>0 and D not ø	NUMBER(@L,@D)
P-Real		NUMBER(@L,@D)
P-Smallint		NUMBER(@L)
P-String		LONG
P-Text		VARCHAR2(@L)
P-Time		DATE
P-Timestamp		ROWID
P-Tinyint		NUMBER(@L)
P-Varbinary		LONG RAW
P-Varchar	L>2000 or L=0 or L ø	LONG
	0<L<2001	VARCHAR2(@L)

Datatype vers Pivot

Datatype	Condition	Pivot
CHAR(L)		P-Character
DATE		P-Date
LONG		P-String
LONG RAW		P-Multimedia
NUMBER		P-Numeric
NUMBER(L)		P-Numeric
NUMBER(L,D)		P-Numeric
RAW(1)		P-Boolean
RAW(L)		P-Boolean
ROWID		P-Timestamp
VARCHAR2(L)		P-Varchar

Dans ce tableau, on peut voir que le type "P-Numeric" a trois correspondances pour les classes types, grâce à trois conditions différentes sur les liens d'équivalence.

Exemple : si P_Numeric est affecté à un attribut et que la longueur de cet attribut est de 10, alors la colonne justifiée par cet attribut via la synchronisation donnera Number(10).

La condition est écrite en langage VB Script. Les principaux éléments de la condition sont les suivants :

- **Sub ConditionInvoke (Colonne, ByRef bValid)** : la première ligne constitue la signature de la fonction.
- **Colonne** : la colonne est donnée en paramètre d'entrée.
- **bValid** : c'est le paramètre de retour. Sa valeur est "True" si la condition est vérifiée, "False" en cas contraire.

Exemple:

```
Sub ConditionInvoke (Colonne, ByRef bValid)
   bValid = False
   If (IsNumeric(Colonne.Length)) Then bValid = True
End Sub
```

Dans la condition, il est possible de préciser les éléments suivants :

- Présence d'un nombre
- Présence d'une décimale
- Plage concernée (exemple : compris entre 0 et 150)

CRÉER DE NOUVEAUX DATATYPES

Chaque datatype est implémenté sous forme de classe ; il est propre à une version de SGBD. Il est possible d'utiliser des masques avec les datatypes.

Exemple pour Oracle 10

Objectif

Faire apparaître un datatype Data8 (numérique avec une longueur et une décimale) dans les scripts ORACLE.

Etapes

Les étapes sont les suivantes :

1. Créez un nouveau datatype dans **HOPEX**. Éventuellement, créez un masque.
2. Reliez le datatype à la version du SGBD cible (en l'occurrence Oracle 10).
3. Reliez le datatype au type correspondant dans le paquetage "Pivot".
4. Paramétrez les conditions sur chaque lien, et ce dans les deux sens (du datatype aux types pivots et du type pivot aux datatypes).

> ☛ *Pour plus d'informations sur les liens d'équivalence et les conditions, voir "Déduire les datatypes des colonnes à partir des types des attributs", page 104.*

Conditions préalables

Pour voir apparaître les paquetages contenant les datatypes des SGBD, vous devez importer le solution pack correspondant. Voir "Importer une version de SGBD", page 9.

> ☛ *Vous pouvez relier le datatype à une cible de génération que vous aurez créée. Pour créer une cible de génération, voir "Créer une nouvelle version de SGBD a venir", page 8 et "Dupliquer une version de SGBD", page 8.*
>
> 💣 **Il est recommandé de ne définir un datatype que dans une seule version de SGBD.**

De plus, certaines données sont protégées dans **HOPEX**. Afin de pouvoir modifier les objets contenus dans les paquetages des SGBD :

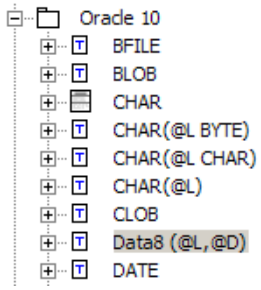
1. Dans le bureau, cliquez sur **Menu principal** > **Paramètres** > **Options**.
2. Dans la partie gauche, cliquez sur **Référentiel**.
   La liste des options liées au référentiel apparaît dans la partie droite de la fenêtre.
3. Dans le champ "Autoriser la modification des données MEGA", sélectionnez "Autoriser".
4. Cliquez sur **OK**.

Créer un nouveau datatype

Pour créer un nouveau datatype :

1. Dans le bureau, cliquez sur le menu de navigation puis sur **Données logiques**.
2. Cliquez sur le dossier **Tous les Paquetage**.
3. Faites un clic droit sur le paquetage "Oracle 10" et sélectionnez **Nouveau > Classe**. La fenêtre de **Création d'une classe** s'ouvre.
4. Nommez votre classe "Data8 (@L,@D)".
5. Ouvrez la fenêtre de propriétés de cette nouvelle classe.
6. Dans la page **Caractéristiques**, sélectionnez dans le champ **Stéréotype** la valeur "Expression", puis cliquez sur **Appliquer**.
7. Le champ **Type expression** apparaît. Dans ce champ, sélectionnez la valeur "Data8 (@L,@D)". Supprimez également les guillemets qui entourent cette valeur.

Vous pouvez constater dans le navigateur qu'une nouvelle classe "Data8" est créée automatiquement.



☛ *Cette nouvelle classe est créée automatiquement pour les besoins de fonctionnement d'UML.*

Relier le datatype au type pivot

Si vous voulez obtenir ce datatype après synchronisation, vous devez lui donner un équivalent au niveau logique :

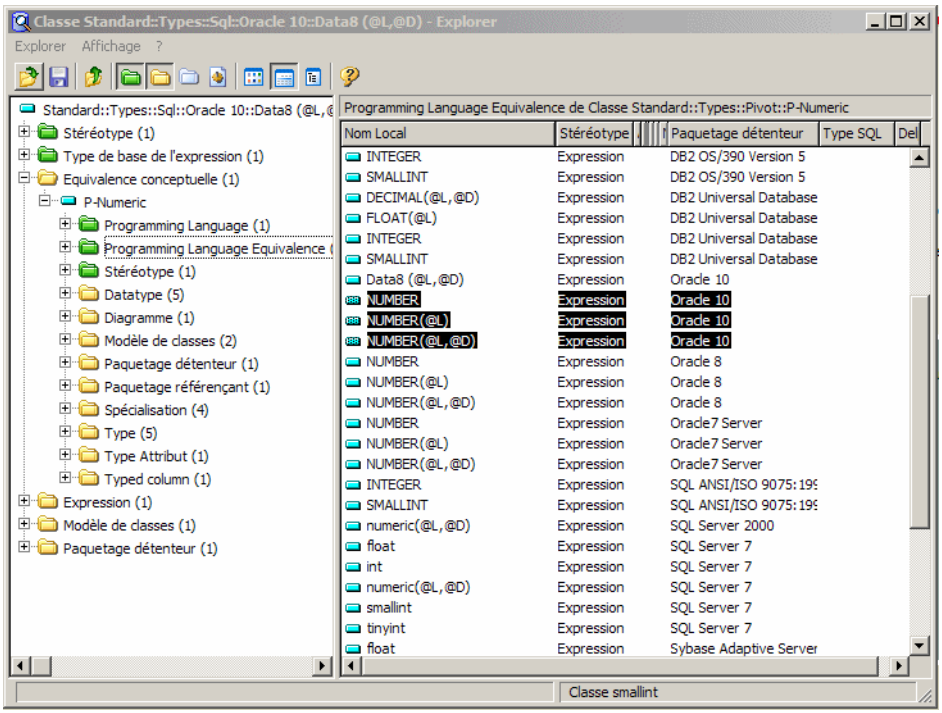
1. Ouvrez la fenêtre de propriétés du datatype "Data8 (@L,@D)".
2. Cliquez sur la liste déroulante puis sur **Compléments**.
3. Faites un clic droit sur le dossier "Equivalence conceptuelle" et sélectionnez **Relier**.
4. Dans la fenêtre de recherche, sélectionnez la classe "P-Numeric".

Paramétrer les conditions sur les liens

Pour paramétrer la condition sur les liens :

1. Faites un clic droit sur la classe "Data8" et sélectionnez **Explorer**.
2. Dépliez le dossier "Equivalence conceptuelle".

3. Sélectionnez le dossier vert "Programming Language Equivalence".
Vous constatez qu'il existe trois autres correspondances pour Oracle 10.



Il s'agit donc de modifier les conditions sur ces correspondances pour qu'elles soient cohérentes avec les conditions posées sur le nouveau datatype.

4. Ouvrez la fenêtre de propriétés du datatype "NUMBER(@L,@D)".
5. Dans la page **Textes**, sélectionnez "Condition d'équivalence de langage", et modifiez le texte pour qu'il corresponde à ceci :

```
Sub ConditionInvoke (Colonne, ByRef bValid)
   bValid = False
   Dim IsNumericLength
   IsNumericLength = IsNumeric(Colonne.Length)
   Dim IsNumericDecimal
   IsNumericDecimal = IsNumeric(Colonne.Decimal)
   If (IsNumericLength and IsNumericDecimal) Then
      If (Colonne.Length <> 8) Then
         bValid = True
      End If
   End If
End Sub
```

6. De la même façon, ajoutez le texte suivant dans la fenêtre de propriétés du nouveau datatype "Data8".

```
Sub ConditionInvoke (Colonne, ByRef bValid)
   bValid = False
   Dim IsNumericLength
   IsNumericLength = IsNumeric(Colonne.Length)
   Dim IsNumericDecimal
   IsNumericDecimal = IsNumeric(Colonne.Decimal)
   If (IsNumericLength and IsNumericDecimal) Then
      If (Colonne.Length = 8) Then
         bValid = True
      End If
   End If
End Sub
```

Vérifier les datatypes

Pour vérifier les datatypes :

1. Faites un clic droit sur une base de données du navigateur de **HOPEX Information Architecture** et sélectionnez **Tables**.
2. Affichez les propriétés d'une colonne concernée par les conditions posées sur le datatype.
3. Vérifiez que le masque affiché dans la colonne **Datatype** est bien "Data8 (%l, %d)" pour cette colonne.

Exemple pour SQL Server 7

Objectif

Relire des colonnes SQL Server 7 comportant un datatype non standard.

Les manipulations sont les mêmes que pour Oracle (voir "Exemple pour Oracle 10", page 113). Cette fois-ci, nous ne créerons pas de masque.

Créer un nouveau datatype

Pour créer un nouveau datatype :

1. Dans le bureau, cliquez sur le menu de navigation puis sur **Données logiques**.
2. Cliquez sur le dossier **Tous les Paquetage**.
3. Faites un clic droit sur le paquetage SQL Server 7 et sélectionnez **Nouveau > Classe**.
   La fenêtre de **Création d'une classe** s'ouvre.
4. Nommez votre classe "TLibelleLong".
5. Ouvrez la fenêtre de propriétés de cette nouvelle classe.
6. Dans la page **Caractéristiques**, sélectionnez dans le champ **Stéréotype** la valeur "Expression", puis cliquez sur **OK**.

Relier le datatype au type pivot

Pour relier le datatype au type pivot :

1. Ouvrez la fenêtre de propriétés du datatype "TLibelleLong".
2. Sélectionnez la page **Compléments**.
3. Faites un clic droit sur le dossier "Equivalence contextuelle" et sélectionnez **Relier**.
4. Dans la fenêtre de recherche, sélectionnez la classe "P-Text".

Paramétrer les conditions sur les liens

Pour paramétrer la condition sur les liens :

1. Faites un clic droit sur la classe "TLibelleLong" et sélectionnez **Explorer**.
2. Sélectionnez le dossier vert "Programming Language Equivalence". Vous constatez qu'il existe une autre correspondance pour SQL Server 7.
3. Ouvrez la fenêtre de propriétés du datatype "text".
4. Dans la page **Textes**, sélectionnez "Condition d'équivalence de langage", et modifiez le texte pour qu'il corresponde à ceci :

```
Sub ConditionInvoke (Colonne, ByRef bValid)
   bValid = False
   Dim IsNumericLength
   IsNumericLength = IsNumeric(Colonne.Length)
   If (IsNumericLength) Then
      If (Colonne.Length > 255) Then
         bValid = True
      End If
   End If
End Sub
```

5. De la même façon, ajoutez le texte suivant dans la fenêtre de propriétés du nouveau datatype "TLibelleLong".

```
Sub ConditionInvoke (Colonne, ByRef bValid)
   bValid = False
   Dim IsNumericLength
   IsNumericLength = IsNumeric(Colonne.Length)
   If (IsNumericLength) Then
      If (Colonne.Length <= 255) Then
         bValid = True
      End If
   End If
End Sub
```

GÉNÉRER DES SCRIPTS SQL

La fonctionnalité de génération SQL produit des fichiers de scripts SQL, qui, à partir des objets logiques de votre référentiel **HOPEX** (BD, table, colonne, etc.), vous permettent de créer, modifier ou de mettre à jour les objets correspondants dans le SGBD cible de votre choix.

La génération prend en compte les paramètres hérités du SGBD cible (spécifié pour la base de données), paramètres que vous pouvez personnaliser à un niveau global (voir "Paramétrer la génération d'une base de données", page 8), ou à un niveau de détail plus fin, sur une colonne ou sur une clé primaire, par exemple.

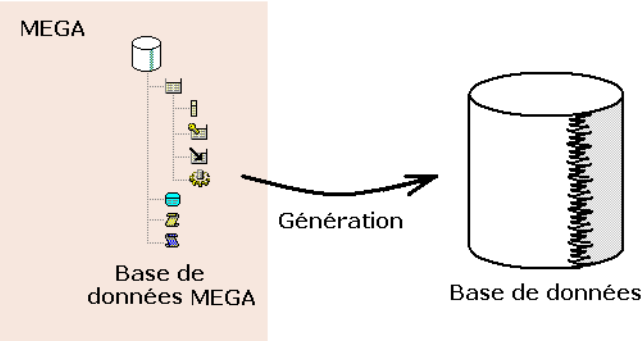
Pour les principaux SGBD cibles du marché, l'éditeur de base de données rend accessible une "vue physique" qui vous permet d'optimiser la grammaire SQL des scripts générés, pour y intégrer des options techniques spécifiques au SGBD choisi, tel le partitionnement. Voir "Ajouter des propriétés physiques aux objets d'une base de données", page 145.

Enfin, le niveau logique peut être complété par la génération d'objets physiques propres à chaque base de données pour un SGBD, comme les vues logiques, les procédures stockées et les triggers. Voir "Options SQL avancées", page 135.

Les différents modes de génération présentés ci-après autorisent la prise en compte de contraintes liées à l'administration des bases de données sous différents systèmes, avec un maximum de souplesse.

Les points abordés ici sont :

LANCER LA GÉNÉRATION SQL



Objets de la génération SQL

Les objets pris en compte par la génération sont :

- Table
- Colonne
- Clé primaire
- Clé étrangère
- Index
- Groupe de données
- Vue logique
- Vue matérialisée
- Trigger
- Procédure stockée
- Fonction
- Synonyme
- Séquence
- Cluster
- Partition

Les scripts générés par **HOPEX** ne gèrent que la structure des objets relationnels, leur contenu n'est pas pris en charge.

Lancer l'assistant de génération

L'affichage de certaines cibles de génération peut être filtré. Avant de lancer la génération, vérifiez que la cible de génération choisie est bien activée :

1. Dans le bureau, cliquez sur **Menu principal** > **Paramètres** > **Options**.

2. Dans la partie gauche de la fenêtre, cliquez sur le dossier **Modélisation des données** puis sur **Génération SQL**.
   Ce dossier contient l'ensemble des générateurs SQL supportés.
3. Dans la partie droite de la fenêtre, cochez ceux que vous voulez afficher dans **HOPEX Information Architecture**.

Pour les cibles de génération, référez-vous aux "SGBD et versions supportées", page 7.

Pour lancer une génération SQL :

1. Dans le bureau, cliquez sur le menu de navigation puis sur **Données physiques**.
2. Dans le volet de navigation, cliquez sur **Outils**.
3. Dans la zone d'édition, cliquez sur **Génération de code SQL**.
   Un assistant apparaît.
4. Définissez le **Périmètre de génération** :
   - base de données pour une génération complète
   - autre objet SQL pour une génération partielle.
5. Sélectionnez la base de données concernée et, le cas échéant, l'objet de la base de données.
6. Cliquez sur **Suivant**.
7. Sélectionnez le **Mode de génération** :
   - "Création" : génère les ordres de création de l'ensemble des objets.
   - "Suppression" : génère uniquement les ordres de suppression d'objets.
   - "Remplacement" : lance la suppression d'objets, puis les re-crée (afin d'éviter la création de doublons, par exemple). Suppose que le SGBD cible supporte ce mode de génération.
   - "Modification" : prend en compte les modifications uniquement. Contrairement aux autres modes qui agissent sans tenir compte d'un éventuel existant, ce mode permet de se connecter au serveur de SGBD et d'obtenir un accès en lecture à la base de données déjà créée. L'assistant compare le fichier des données **HOPEX** et les informations de la base de données. Après analyse des deux structures, l'assistant génère les ordres SQL de modification correspondants. Voir "Génération incrémentale", page 4.

     ☛ *Ce mode n'est disponible que pour les principaux SGBD. Voir "SGBD et versions supportées", page 7.*
8. Cliquez sur **Suivant**.
   Une fenêtre présente les objets générés.
9. Cliquez sur **Suivant**.
   La génération est lancée. Une fenêtre présente les fichiers contenant le résultat.

   Le bouton **Ouvrir** permet de visualiser les fichiers résultats de la génération.

GÉNÉRATION INCRÉMENTALE

Lorsqu'une base de données a déjà été générée, vous pouvez répercuter par la suite uniquement les changements apportés à la base de données grâce au mode "Modification" de la génération SQL.

La génération incrémentale permet, pour une base de données :

- de consulter dans un rapport .html les différences entre la base de données et sa représentation dans **HOPEX**.
- de produire les scripts SQL permettant de mettre à jour la base de données cible à partir de sa description dans **HOPEX**.

Objets de la génération incrémentale

Les objets gérés par la génération incrémentale sont les mêmes que ceux de la génération en mode "Création" : table, colonne, clé primaire, clé étrangère, index, groupe de données, vue logique, vue matérialisée, trigger, procédure stockée, fonction, synonyme, séquence, cluster, partition.

Les scripts générés par **HOPEX** ne gère que la structure des objets relationnels, leur contenu n'est pas pris en charge. Des options de génération incrémentale permettent d'isoler les ordres SQL qui nécessitent des précautions particulières ou des traitements complémentaires.

Lancer la génération incrémentale

Options de la génération

La génération incrémentale se fait dans un fichier global; elle s'effectue à partir de la base de données et non sur un objet modifié en particulier.

Avant de lancer la génération :

1. Faites un clic droit sur la base de données et sélectionnez **Propriétés**. La fenêtre de propriétés de la base de données apparaît.
2. Cliquez sur la liste déroulante puis sur **Options** > **Génération**.
3. Dans le champ **Ventilation Script**, sélectionnez "Un fichier global".
4. Cliquez sur **OK**.

Sous les options, vous devez également indiquer le mode de génération incrémentale, qui autorise ou non la suppression d'objets.

Pour plus de détails sur les options de génération, voir "Paramétrer la génération d'une base de données", page 8.

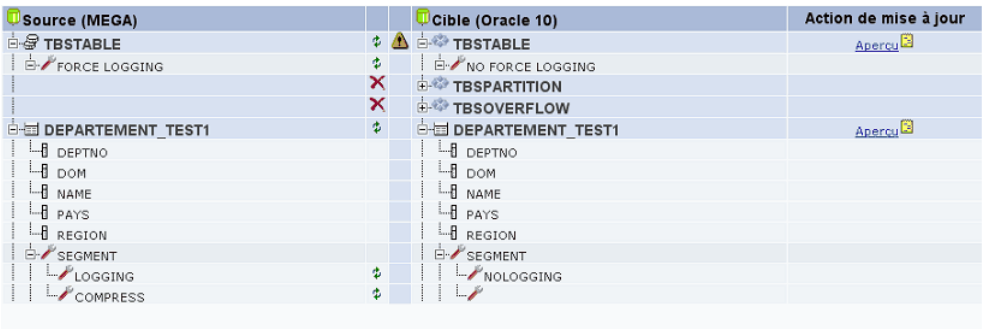
Lancer l'assistant de génération

Pour lancer la génération incrémentale :

1. Faites un clic droit sur la base de données et sélectionnez **Générer le code**.
2. Dans le champ **Mode de génération**, sélectionnez "Modification".

   ☛ *La commande "Modification" n'est disponible que pour les principaux SGBD. Voir "SGBD et versions supportées", page 7.*
3. Sélectionnez la **Source de données**. La génération incrémentale peut s'effectuer :
   - A partir d'une connexion ODBC.
   - A partir d'un fichier d'extraction. Voir "Utilitaire d'extraction ODBC", page 159.
4. Cliquez sur **Suivant**.
5. Une fois connecté au SGBD cible, saisissez le nom du propriétaire. Cela vous permet de filtrer les tables à prendre en compte dans la génération.
6. Cliquez sur **Suivant**.

La fenêtre de résultat présente deux fichiers, le fichier .sql ainsi qu'un fichier "Report.htm". Ce dernier fichier est un compte-rendu de génération. C'est un fichier dynamique qui présente le contenu de la base initiale et les modifications apportées.



Chaque ligne de la liste décrit :

- L'objet **HOPEX**. Celui-ci peut être vide si il a été supprimé du référentiel **HOPEX**.
- Le symbole de mise à jour du SGBD par rapport à **HOPEX**. Les différentes actions possibles sur les objets sont :
  - la création
  - la modification
  - la suppression
  - le remplacement
- Un symbole d'avertissement lorsque la mise à jour d'un objet du SGBD n'est pas complète ou que celle-ci doit être opérée avec prudence.

Lorsque cette icône est présente, un bloc dans le script généré détaille ce qui ne peut être mis à jour.

- L'objet du côté SGBD. Celui-ci peut être vide dans le cadre d'une création du côté **HOPEX**.
- Le lien vers le script de mise à jour de l'objet.

A partir de chaque objet, vous pouvez accéder à l'ensemble de ses sous-objets en dépliant l'arborescence correspondante. Vous pouvez également visualiser les paramètres physiques. Dans le cadre d'une modification d'un objet, seuls les paramètres physiques modifiés sont affichés.

Il existe également un compte-rendu de génération (.txt) dans la fenêtre de propriétés de la base générée.

PARAMÉTRER LA GÉNÉRATION SQL

Paramétrer la version de SGBD

SGBD et versions supportées

La liste suivante est donnée à titre indicatif. Elle ne prétend pas être exhaustive et elle est susceptible d'évoluer en fonction de la sortie de nouvelles versions des SGBD.

Produit	Éditeur	Versions supportées
SQL ANSI	ISO 9075	1992
DB2	IBM	OS 390 V5 / OS 390 V7 / OS 390 V8 UDB V5 / UDB V7 / UDB V8
Ingres II	Computer Associates	2.0
Dynamic Server	Informix	7.3
Oracle	Oracle	8 / 9i / 10 / 11
SQL Server	Microsoft	7 / 2000 / 2005 / 2008
Adaptative Server	Sybase	11 / 12.5
Teradata Database	Teradata	14
PostgreSQL	PostgreSQL Global Development Group	9.3

Modifier les propriétés de la version de SGBD

Le paramétrage de la génération est effectué pour le contrôle, entre autres, de la taille des identificateurs générés, le contrôle des caractères autorisés, etc.

L'accès à un SGBD se fait à partir d'une base de données associée à ce SGBD.

Pour paramétrer les options de génération d'une version de SGBD :

1. Faites un clic droit sur la base de données concernée et sélectionnez **Propriétés**.
   La fenêtre des propriétés de la base de données apparaît.
2. Cliquez sur la liste déroulante puis sur **Caractéristiques**.
3. Dans le champ **SGBD cible**, cliquez sur la flèche située à l'extrémité afin d'accéder aux propriétés du SGBD.
4. Dans la fenêtre des propriétés du SGBD, cliquez sur la liste déroulante puis sur **Options** > **Génération**.

5. Modifiez les paramètres que vous souhaitez changer.

> ☛ *Les paramètres disponibles varient en fonction du SGBD cible. Si le sous-onglet **Génération** n'apparaît pas, cliquez sur le menu **Outils** > **Options** de la barre HOPEX, puis dépliez le dossier "Modélisation des données > Génération SQL" et vérifiez que l'option de génération liée au SGBD en question est bien cochée.*

Paramétrer la génération d'une base de données

Pour paramétrer la génération d'une base de données :

1. Ouvrez la fenêtre de propriétés de la base de données.
2. Cliquez sur la liste déroulante puis sur **Options** > **Génération**.

De même que pour le paramétrage de la cible, les paramètres proposés varient en fonction du SGBD, et un message explicatif indique l'utilisation de chaque paramètre.

Vous pouvez notamment renseigner les paramètres suivants :

- Les paramètres **Nom Trigger** définissent les noms des trois types de trigger.
- **Base numéro d'erreur** : numéro d'erreur utilisateur pour le SGBD courant.
- **Val. par défaut** : active/désactive la génération des ordres DEFAULT pour les colonnes.
- **Quoted Identifier** : active/désactive la génération des guillemets autour des identifiants SQL (Nom SQL).
- **Qualifier** : permet de préfixer le nom des objets. Voir "Préfixer le nom des objets", page 10.
- **Mode génération inc** : ce paramètre s'applique à la génération incrémentale et peut prendre les valeurs "Alter" et "Drop/Create".
  - "Alter" n'autorise pas la suppression d'objets (tables, indexes, etc.) au niveau des scripts générés. Seules les instructions pouvant être réalisées au travers de la commande ALTER sont générées. Pour les paramètres physiques, la suppression reste autorisée (c'est le cas notamment pour les partitions).
  - "Drop/Create" autorise la suppression d'objets. Si une mise à jour ne peut s'effectuer au travers de la commande ALTER, l'objet est supprimé puis recréé.

  Par défaut, le paramètre prend la valeur "Alter".
- **Ventilation script** : indique si le résultat de la génération doit être créé dans un fichier unique, ou réparti dans un fichier distinct pour chaque type d'objet ou pour chaque objet.
- **Script SQL** : nom du fichier généré lorsqu'il s'agit d'un fichier global. Par défaut, il s'appelle MEGASQL.SQL. Vous pouvez le personnaliser au

niveau du SGBD. La flèche située à l'extrémité du champ vous permet de réinitialiser le paramètre.

> **Vous pouvez également réinitialiser tous les paramètres de l'objet en question. Cette action est à utiliser avec précaution.**

- **Répertoire Script** : répertoire de génération relatif.
- Les différents paramètres **Ext**. permettent de préciser les extensions de chaque fichier généré, pour les tables, les groupements de données, les vues etc.
- **Conversion** : format des fichiers générés (ANSI Windows ou ASCII MS-DOS).
- **CREATE CLUSTER** : active/désactive la génération des ordres CREATE CLUSTER.
- **CREATE TABLE** : active/désactive la génération des ordres CREATE TABLE.
- **CREATE TABLESPACE** : active/désactive la génération des ordres CREATE TABLESPACE.
- **PRIMARY KEY** : active/désactive la génération des ordres PRIMARY KEY.
- **FOREIGN KEY** : active/désactive la génération des ordres FOREIGN KEY.
- **CREATE INDEX** : active/désactive la génération des ordres CREATE INDEX.
- **CREATE PROCEDURE** : active/désactive la génération des procédures stockées.
- **CREATE INDEX PK** : active/désactive la génération des ordres CREATE INDEX pour les index de clés primaires.
- **CREATE INDEX[UNIQUE]** : active/désactive la génération des ordres CREATE INDEX pour les index uniques.
- **CREATE VIEW** : active/désactive la génération des vues logiques.
- **CREATE SEQUENCE** : active/désactive la génération des ordres CREATE SEQUENCE.
- **CREATE SYNONYM** : active/désactive la génération des ordres CREATE SYNONYM.
- **CREATE TRIGGER** : active/désactive la génération des triggers.
- **Commentaires** : active/désactive la génération des commentaires **HOPEX** dans le script SQL.
- **UNIQUE** : active/désactive la génération des ordres UNIQUE.
- **UNIQUE[PK]** : active/désactive la génération des ordres UNIQUE portant sur les clés primaires.
- **Syntaxe PRIMARY KEY** : les ordres PRIMARY KEY sont générés dans l'ordre CREATE TABLE ou dans un ordre ALTER TABLE.
- **Position FOREIGN KEY** : génération des ordres FOREIGN KEY après chaque CREATE TABLE ou groupés en fin de script.
- **COMMENT ON TABLE** : commentaires sur les tables (0 : pas de commentaire, 1 : une ligne, Total : texte complet).
- **COMMENT ON COLUMN** : commentaires sur les colonnes (0 : pas de commentaire, 1 : une ligne, Total : texte complet).

☛ *La génération des commentaires n'est possible que pour les systèmes cibles qui les acceptent (Oracle, DB2,...).*

- Les différents paramètres **Complément** activent/désactivent la génération des compléments sur les tables, les groupements de données etc.
- **Tbspace des tables** : par défaut, les tables sont générées dans le tablespace SYSTEM.
- **Tbspace des index** : par défaut, les index sont générés dans le tablespace SYSTEM.

Préfixer le nom des objets

Pour la plupart des SGBD, il existe une notion de schéma, qui permet de définir un regroupement logique pour les objets.

Ainsi, à la création d'une table, par exemple, celle-ci peut être automatiquement rangée dans un schéma, et si celui-ci n'est pas renseigné, un schéma par défaut peut lui être automatiquement affecté.

Dans **HOPEX** il n'existe pas de notion de schéma mais un concept qui lui est propre - le Qualifier - qui permet de préfixer les noms des objets d'une base de données lors de la génération. Si vous souhaitez par exemple que les objets d'une base aient leur nom préfixé par "MEGA", vous devez saisir cette valeur dans le champ Qualifier des objets en question.

Héritage

Le Qualifier peut être défini au niveau de la base de données et également sur tous les autres types d'objet. Il existe un système d'héritage : si le Qualifier n'est pas spécifié au niveau d'une table, par défaut c'est la valeur saisie sur la base de données qui est prise en compte.

Pour préfixer le nom d'un objet lors de la génération :

1. Ouvrez la fenêtre de propriétés de l'objet en question.
2. Cliquez sur la liste déroulante puis sur **Options** > **Génération**.
3. Dans le champ **Qualifier**, saisissez la valeur qui viendra préfixer le nom de l'objet.

SGBD concernés

Le Qualifier est disponible pour les SGBD suivants :

- Oracle
- SQL Server
- DB2
- MySQL

SYNTAXE SUPPORTÉE

Instruction CREATE TABLE

L'instruction CREATE TABLE définit une table. La définition inclut :

- Le nom de la table
- Les noms et attributs de ses colonnes
- Les attributs de la table tels que ses clés primaire et étrangères

La syntaxe est la suivante :

```
CREATE TABLE nom-table (nom-col1 type-col1 [NOT NULL]
...
nom-coln type-coln [NOT NULL])
```

Pour DB2, la syntaxe est la suivante :

```
CREATE TABLE nom-table (nom-col1 type-col1 [NOT NULL]
...
nom-coln type-coln [NOT NULL])
[in Tablespace <Nom>]
```

Pour Oracle, la syntaxe est la suivante :

```
CREATE TABLE nom-table (nom-col1 type-col1 [NOT NULL]
...
nom-coln type-coln [NOT NULL])
[Tablespace <Nom>]
```

- ***nom-table*** : Valeur " SQL " de la table, ou, à défaut, nom de la table ; les caractères non reconnus sont remplacés par "_"
- ***nom-col*** : Valeur de l'attribut SQL Name de la colonne ou, à défaut, nom de la colonne ; les caractères non reconnus sont remplacés par "_"
- ***type-col***
- ***NOT NULL*** : Voir "Gestion de NOT NULL", page 11
- ***Tablespace*** : DB2 et Oracle : Nom du tablespace cible pour les tables

La clause PRIMARY KEY est précisée dans l'ordre "CREATE TABLE" (voir "Clause PRIMARY KEY", page 12).

Gestion de NOT NULL

Les clauses NULL, NOT NULL et NOT NULL WITH DEFAULT sont générées automatiquement sur les colonnes de clés primaires et sur les colonnes issues d'attributs obligatoires lors de la synchronisation.

Ces valeurs peuvent être initialisées à "Null", "Not Null" ou "Not Null with Default" en fonction du paramétrage défini dans la fenêtre de propriétés de la base de

données pour **Colonnes Not Null**, dans le sous-onglet **Synchronisation** de l'onglet **Options**.

Les valeurs ainsi proposées peuvent ensuite être modifiées sur chaque colonne.

Clause PRIMARY KEY

Définition d'une clé primaire

Une ou plusieurs colonnes d'une table permettent d'identifier chaque ligne de cette table. Les valeurs dans ces colonnes doivent être renseignées. Elles constituent la clé primaire ("primary key") de la table.

Une table doit avoir une seule clé primaire ou aucune.

Chaque nom de colonne doit identifier une colonne de la table et la colonne ne doit pas être identifiée plus d'une fois.

Traitements et génération des ordres SQL

Après la déclaration des noms de colonnes de la table, si l'option **PRIMARY KEY** est active, le ou les noms de colonnes de la clé primaire sont déclarés de la manière suivante :

```
PRIMARY KEY (liste des colonnes de la clé primaire)
```

La clause PRIMARY KEY est générée dans l'ordre **CREATE TABLE**.

Exemple 1	Exemple 2
La clé primaire "CP" n'a qu'une colonne, "Col-cp". CREATE TABLE nom-table (Col-cp CHAR(9) NOT NULL, info1 CHAR(7), info2 CHAR(7), PRIMARY KEY (Col-cp)	La clé primaire "CP1" a pour colonnes "CP11" et "CP12". CREATE TABLE nom-table (CP11 CHAR(9) NOT NULL, CP12 CHAR(9) NOT NULL, info1 CHAR(7), info2 CHAR(7), PRIMARY KEY (CP11, CP12)

Pour Oracle, la clause PRIMARY KEY complète est la suivante :

```
CONSTRAINT <nom de la clé> (liste des colonnes de la clé primaire)
```

Clause FOREIGN KEY

L'intégrité référentielle peut être exprimée, selon le système cible, soit par des clauses FOREIGN KEY, soit par la génération de triggers. Pour Oracle, elle est exprimée soit par des triggers, soit par des clauses FOREIGN KEY, en fonction du paramétrage de la base de données.

Une ou plusieurs colonnes d'une table peuvent faire référence à une clé primaire dans cette table ou dans une autre. Ces colonnes constituent une clé étrangère. Ces colonnes peuvent ne pas avoir de valeur sur chaque ligne.

La table qui contient la clé primaire référencée est une table parente. La table qui contient la clé étrangère est une table dépendante.

Chaque nom de colonne doit identifier une seule colonne de la table et une même colonne ne doit pas être identifiée plus d'une fois. Si la même liste de noms de colonne est spécifiée dans plus d'une clause FOREIGN KEY, ces clauses ne doivent pas identifier la même table.

Le nom de la table spécifié dans la clause FOREIGN KEY doit identifier une table parente. Une clé étrangère de table dépendante doit avoir le même nombre de colonnes que la clé primaire de la table parente.

Le nombre de clés étrangères n'est pas limité.

Traitement et génération des ordres SQL

Après la déclaration des clés primaires (PRIMARY KEY), le(s) nom(s) de colonnes de la (des) clé(s) étrangère(s) de la table sont déclarés avec l'indication FOREIGN KEY :

```
FOREIGN KEY (liste des colonnes de la clé étrangère)
REFERENCES <Nom table parente> [ON DELETE <Action>] [ON
UPDATE <Action>]
```

ou :

```
ALTER TABLE nomtable [ADD] FOREIGN KEY (liste des colonnes
de la clé étrangère) REFERENCES <Nom table parente> [ON
DELETE <Action>] [ON UPDATE <Action>]
```

Pour Oracle, la syntaxe est la suivante :

```
CONSTRAINT <nom de la clé étrangère> (liste des colonnes de
la clé étrangère)  REFERENCES <Nom table parente> [ON DELETE
<Action>] [ON UPDATE <Action>]
```

ou :

```
ALTER TABLE...
```

Exemples

La table "nom-table1" a deux clés étrangères. Ces clés n'ont pas de composante.

```
CREATE TABLE nom-table1
(cp1 CHAR(9) NOT NULL,
cp2-rel12 CHAR(7) NOT NULL,
cp3-rel13 CHAR(7) NOT NULL,
info1 CHAR(7),
info2 CHAR(7),
PRIMARY KEY (cp1))
ALTER TABLE nom-table1 ADD FOREIGN KEY(cp2-rel12)
REFERENCES nom-table2
ALTER TABLE nom-table2 ADD FOREIGN KEY(cp3-rel13)
REFERENCES nom-table3
```

La table "nom-table1" possède une clé étrangère "ce2" qui a deux composantes, "ce21" et "ce22". La clé étrangère "ce2" n'a pas de référence (elle est donc composante de la clé primaire d'une autre table).

```
CREATE TABLE nom-table1
(cp1 CHAR(9) NOT NULL,
ce21 CHAR(7) NOT NULL,
ce22 CHAR(7) NOT NULL,
info1 CHAR(7),
PRIMARY KEY (cp1))
ALTER TABLE nom-table 1 ADD FOREIGN KEY (ce21, ce22)
REFERENCES nom-table 2
```

La table "nom-table1" a une clé étrangère, "ce2". La clé étrangère "ce2" est équivalente à la clé primaire "cp2" qui a deux composantes, "cp21" et "cp22". Les colonnes identifiées par "cp21" et "cp22" sont "NOT NULL".

```
CREATE TABLE nom-table1
(cp1 CHAR(9) NOT NULL,
cp21 CHAR(7) NOT NULL,
cp22 CHAR(7) NOT NULL,
info1 CHAR(7),
info2 (CHAR7),
PRIMARY KEY (cp1))
ALTER TABLE nom-table1 ADD FOREIGN KEY (cp21, cp22)
REFERENCES nom-table2
```

Clause UNIQUE

Une clause UNIQUE est générée pour chaque index unique de la table, sauf si cet index correspond à la clé primaire.

Traitement et génération des ordres SQL

Pour chaque index unique, la clause suivante est générée :

```
UNIQUE (col1,...,coln)
```

(col1,...n,coln) représente les colonnes de l'index.

Instruction CREATE INDEX (Oracle, Sybase, SQL Server)

Définition d'un index

Un index est un ensemble de colonnes d'une table sur lesquelles un accès direct est défini.

Pour Sybase et SQL Server, la valeur de l'attribut type-index de l'index détermine le type de l'index généré : UNIQUE, CLUSTERED ou UNIQUE CLUSTERED.

Traitement et génération des ordres SQL

Pour chaque index, une clause est générée, en fonction du SGBD cible.

Pour Oracle :

```
CREATE INDEX (colonne1,..., colonneN) [TABLESPACE
(NomTbSpace)
```

(colonne1,..., colonneN) représente les colonnes de l'index ; NomTbSpace est le nom du tablespace pour les index (voir "Paramétrer la génération SQL", page 7).

Pour Sybase et SQL Server :

```
CREATE [UNIQUE] [CLUSTERED] INDEX (NomIndex) (NomTable)
(colonne1,..., colonneN)
```

(colonne1,..., colonneN) représente les colonnes de l'index ; NomTbSpace est le nom du tablespace pour les index (voir "Paramétrer la génération SQL", page 7).

Clause CREATE VIEW

Une vue est définie sur une base de données. Elle peut utiliser une ou plusieurs tables.

```
CREATE VIEW nom-vue
AS
SELECT
(nom-colonne, nom-colonne,...)
```

Il est possible de compléter cette définition dans la spécification de la vue (voir "Définir les vues d'une base de données", page 136).

OPTIONS SQL AVANCÉES

Vous pouvez compléter la définition des objets relationnels par des définitions d'objets techniques tels que les vues, triggers, procédures stockées, dont la spécification est propre à chaque SGBD cible et qui peuvent être générés séparément ou dans le fichier SQL de la base de données.

Vous pouvez ainsi :

- ✓ "Définir les vues d'une base de données", page 136
- ✓ "Définir les triggers pour une base de données", page 140
- ✓ "Utiliser des procédures stockées", page 143
- ✓ "Ajouter des propriétés physiques aux objets d'une base de données", page 145

Les outils présentés dans les pages qui suivent sont disponibles avec un accès au référentiel en mode "Avancé".

DÉFINIR LES VUES D'UNE BASE DE DONNÉES

Une vue physique est une table virtuelle dont la structure et le contenu sont déduits d'une ou plusieurs autres tables par une requête SQL.

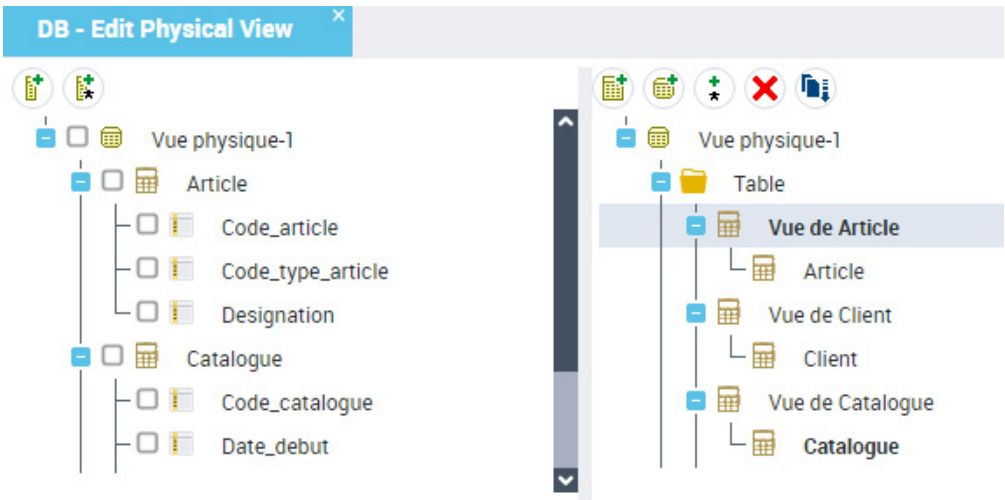
La création des *vues* d'une base de données se fait par l'intermédiaire d'un arbre, qui permet de générer automatiquement une partie de la définition de la vue, et peut être complétée par l'utilisateur.

Créer les vues d'une base de données

Pour créer une vue physique d'une base de données :

1. Dans le bureau, cliquez sur le menu de navigation puis sur **Données physiques**.
2. Dans le volet de navigation, cliquez sur **Vues physiques**.
   La liste des vues physiques apparaît dans la zone d'édition.
3. Cliquez sur **Nouveau**.
   L'assistant de création des vues apparaît.
4. Dans le champ **Détenteur**, sélectionnez la base de données concernée.
5. Sous le champ **Table**, cliquez sur **Nouveau**.
6. Sélectionnez les tables sur lesquelles porte la vue.
7. Cliquez sur **OK**.
   L'éditeur de la vue physique apparaît.

Le navigateur de gauche affiche les tables de la base de données sur lesquelles porte la vue physique, avec leurs colonnes. L'arbre de droite affiche les tables et les colonnes de vue, qui constituent la vue. Par défaut, celles-ci portent le nom des tables et colonnes sources ; vous pouvez les renommer.



Ajouter une table ou une colonne de vue

Pour ajouter une table à une vue :

1. Dans l'arbre droit de l'éditeur, faites un clic droit sur le dossier **Table** et sélectionnez **Table de vue**.

Sélectionnez la table voulue et cliquez sur **OK.**

Pour ajouter une colonne à une vue :

1. Dans la partie droite de l'éditeur, faites un clic droit sur le dossier **Colonne** et sélectionnez **Colonne de vue**.
2. Sélectionnez la colonne voulue et cliquez sur **OK**.

Définition SQL

La fenêtre d'édition des vues présente, dans le cadre **Définition SQL**, le code de définition de la vue tel qu'il sera généré. Le code est, au départ, calculé en fonction de la définition indiquée dans l'arborescence.

Vous pouvez modifier ce code, notamment à l'aide des jointures. Vous pouvez également saisir directement les modifications dans le cadre SQL.

Jointures de vue

Par défaut, la fenêtre d'édition des vues propose les clés étrangères des tables sélectionnées, lorsqu'elles existent.

Il est ainsi possible de compléter la spécification d'une vue en lui associant des clés étrangères, sources potentielles de jointures.

Pour associer une clé étrangère à la vue :

- Sélectionnez dans arborescence la clé étrangère et glissez-la dans le champ de définition SQL.

Mode utilisateur

Vous pouvez modifier le code de la vue en tapant directement dans le cadre de définition SQL :

- Cliquez sur le bouton **Enregistrer** pour que la **Définition SQL** soit conservée dans le référentiel en tant que telle.

Après modification, il est possible de revenir à la définition telle qu'elle est déduite de l'arborescence :

- Cliquez sur le bouton **Initialiser la définition SQL** .
  Un message vous avertit que la définition préalablement sauvegardée va être réinitialisée. Autrement dit, tous les compléments éventuellement apportés à la définition de la vue sont perdus.
- Cliquez sur **OK** pour confirmer.

Champs

Les catégories de champ correspondent aux types d'objets utilisés dans l'arbre déclaratif : table, vue, colonne et colonne de clé étrangère. Les champs affichés dans la définition SQL correspondent aux éléments déclarés dans l'arbre.

Le type clé étrangère ne donne pas lieu à une catégorie de champ : les champs utilisables proviennent des colonnes de clé et non pas des clés elles-mêmes.

Le bouton **Propriétés du champ**  vous permet d'afficher les propriétés de l'objet correspondant au champ sélectionné.

Si un objet est ajouté dans l'arbre, un champ correspondant devient disponible pour insertion.

Si un objet est renommé dans l'arbre ou dans le référentiel, ses références restent valides et les champs sont affichés avec le nouveau nom dans le texte.

Si un objet est supprimé dans l'arbre ou dans le référentiel, ses références deviennent invalides et sont signalées comme telles dans les champs.

Définir un groupement de données

Un groupement de données - ou tablespace - est un regroupement dans une même page ou des pages physiques voisines de la base de données des lignes de plusieurs tables à des fins d'optimisation, en particulier pour les jointures. Ex : Tablespace dans DB2, Cluster dans Oracle etc.

Pour définir des *groupements de données* dans la base de données :

1. Ouvrez la fenêtre de propriétés de la base de données.
2. Cliquez sur la liste déroulante puis sur **Composants**.
3. La section **Groupements de données** affiche la liste des groupements de données
4. Cliquez sur le bouton **Nouveau**.
5. Dans la fenêtre qui apparaît, indiquez le **Nom** du groupement de données.
6. Cliquez sur **OK**.

Vous allez ensuite ouvrir la fenêtre de propriétés du groupement de données pour définir les tables et les index qu'il inclut.

Pour spécifier les tables et les index inclus dans le groupement de données :

1. Sélectionnez le groupement de données et cliquez sur le bouton **Propriétés**.
   La fenêtre des propriétés du groupement de données apparaît.
2. Cliquez sur la liste déroulante puis sur **Tables**.
3. Cliquez sur le bouton **Relier**.
4. Dans la liste des tables de la base de données qui est présentée, sélectionnez les tables de la base de données qu'il devra inclure.

   ☛ *Le bouton **Délier** permet d'enlever une table de la liste en cas d'erreur.*
5. Effectuez les mêmes opérations pour les index du groupement de données.

6. Cliquez sur **OK**.

DÉFINIR LES TRIGGERS POUR UNE BASE DE DONNÉES

Un trigger est un traitement enregistré dans une base de données et qui est déclenché automatiquement lors de la mise à jour d'une table.

Créer un trigger

Les triggers se définissent au niveau des tables d'une base de données.

> ☛ *Il faut noter que les triggers sont définis en fonction du SGBD cible; c'est pourquoi il est important de contrôler, avant de créer les triggers, que le SGBD cible est correct.*
>
> *Si le SGBD cible est modifié a posteriori, les triggers créés pour ce SGBD ne sont pas supprimés, mais ils sont inactivés.*

Pour créer un trigger :

1. Dans le bureau, cliquez sur le menu de navigation puis sur **Données physiques**.
2. Dans le volet de navigation, sous le dossier **Bases de données**, cliquez sur **Hiérarchie des bases de données**.
   La liste des bases de données apparaît dans la zone d'édition.
3. Dépliez le dossier de la base de données puis de la table concernées.
4. Faites un clic droit sur le dossier **Trigger** et sélectionnez **Nouveau** > **Trigger**.
   La fenêtre de création d'un trigger apparaît.
5. Indiquez le nom du trigger et les actions déclenchées. Voir "Déclenchement du trigger", page 140.

Déclenchement du trigger

Le trigger peut se déclencher suite à l'une des trois actions suivantes :

- Lors de la **Création** d'une ligne dans la table
- Lors de la **Suppression** d'une ligne
- Lors d'une **Modification** de la table ou d'une colonne en particulier

De plus, vous pouvez choisir de le lancer **Avant** ou **Après** ces actions, sur l'ensemble de la table ou sur chaque ligne concernée.

Références

Les champs «Référence des anciens tuples» et «Référence des nouveaux tuples» créent dans le code du trigger des références aux lignes insérées, supprimées ou mises à jour.

Le nom indiqué dans le champ "Référence des anciens tuples" correspond à la ligne qui existait avant la mise à jour.

Le nom du champ «Référence des nouveaux tuples» fait référence à la ligne après la mise à jour.

En cas d'insertion, seule la nouvelle ligne est valide.

En cas de suppression, seule l'ancienne est valide.

Définition SQL

L'option **Définition SQL**, dans les propriétés du trigger**,** présente le code du trigger.

Pour accéder à cette option, vous devez avoir accès au référentiel en mode "Expert".

Pour afficher le mode expert :

1. Dans le bureau, cliquez sur **Menu principal** > **Paramètres** > **Options** de **HOPEX**.
2. Dans la partie gauche de la fenêtre cliquez sur le dossier **Référentiel**.
3. Dans la partie droite de la fenêtre, dans le champ **Accès au référentiel**, sélectionnez le mode "Expert".

Pour afficher le code du trigger :

1. Faites un clic droit sur le trigger et sélectionnez **Propriétés**.
   La fenêtre des propriétés du trigger apparaît.
2. Cliquez sur la liste déroulante puis sur **Textes**.
3. Cliquez sur l'onglet **Définition SQL**.

Intégrité référentielle

L'intégrité référentielle est gérée par la création des clés étrangères sur une base de données.

Elle regroupe l'ensemble des contraintes permettant de contrôler l'incidence d'une modification d'une table dans les tables qui lui sont liées.

Il se peut que l'existence de clés dans certains SGBD n'implique pas un contrôle systématique. Il se peut également que vous vouliez personnaliser les contraintes à appliquer sur une table en particulier.

C'est pourquoi vous pouvez générer dans des triggers le code qui correspond à la gestion de l'intégrité référentielle.

Pour générer l'intégrité référentielle d'une table :

1. Faites un clic droit sur la base de données concernée et sélectionnez **Générer les triggers**.
   La fenêtre de génération des triggers apparaît.
2. Sélectionnez une des options de génération :
   - Générer la cible par type
   - Générer la cible par intégrité du référentiel
3. Sélectionnez les tables de la base de données.

4. Cliquez sur **Suivant**.
   Les triggers sont automatiquement créés pour les tables sélectionnées.
   Une fois la génération terminée, les triggers apparaissent sous le dossier **Trigger** disponible sous chaque table. Il existe trois type de trigger :
   - Un trigger de mise à jour (U_ suivi du nom de la table), qui permet de spécifier l'action à réaliser en cas de modification d'une ligne de la table faisant partie de la clé étrangère.
   - Un trigger de suppression (D_), qui spécifie l'action à réaliser en cas de suppression.
   - Un trigger d'insertion (I_), qui spécifie l'action à réaliser en cas d'insertion.

Ces triggers ne sont valables que pour un SGBD cible. Lorsque vous changez de SGBD, vous devez regénérér les triggers.

UTILISER DES PROCÉDURES STOCKÉES

HOPEX Information Architecture vous permet de créer des *procédures stockées*.

Une procédure stockée combine un langage procédural et des requêtes SQL au sein d'un programme. Elle permet d'exécuter une tâche particulière sur une base de données. Elle est enregistrée dans une base de données et peut être appelée depuis un programme extérieur à la base de données ou depuis un trigger.

Une procédure stockée peut être implémentée de deux façons; sous la forme d'une procédure ou d'une fonction.

> 🕮 *Une procédure est un ensemble d'instructions exécutant un sous-programme.*
>
> 🕮 *Une fonction est une procédure retournant une valeur à la fin de l'exécution.*

Pour créer une procédure stockée sur une base de données :

1. Ouvrez la fenêtre de propriétés de la base de données.
2. Cliquez sur la liste déroulante puis sur **Composants**.
   La section **Procédures stockées** affiche la liste des procédures stockées.
3. Cliquez sur le bouton **Nouveau**.
4. Dans la fenêtre qui apparaît, indiquez le nom de la procédure et sa nature (Procédure ou Fonction).
5. Cliquez sur **OK**.
   La procédure stockée apparaît. Ouvrez ses propriétés pour définir son code.

Exemple de procédure stockée pour Oracle

Voici l'exemple d'une procédure stockée mettant à jour le prix unitaire d'une pièce en fonction de l'identifiant de la pièce :

```
CREATE PROCEDURE update_part_unitprice (part_id IN INTEGER,
new_price IN NUMBER)
IS
 Invalid_part EXCEPTION;
BEGIN
-- HERE'S AN UPDATE STATEMENT TO UPDATE A DATABASE RECORD
 UPDATE sales.parts
   SET unit_price = new_price
   WHERE id = part-id;
-- HERE'S AN ERROR-CHECKING STATEMENT
 If SQL%NOTFOUND THEN
   RAISE invalid_part;
 END IF;
EXCEPTION
-- HERE'S AN ERROR-HANDLING ROUTINE
 WHEN invalid_part THEN
```

```
    raise_application_error(-20000, 'Invalid Part ID');
END update_part_unitprice;
```

AJOUTER DES PROPRIÉTÉS PHYSIQUES AUX OBJETS D'UNE BASE DE DONNÉES

Une fois votre base de données définie dans un diagramme relationnel, vous pouvez générer les scripts SQL correspondants à destination de différents SGBD.

Le volet de navigation des données physiques vous permet de compléter la modélisation physique d'une base de données en spécifiant les paramètres propres à chaque SGBD et de produire ainsi des scripts SQL complets.

Vous pouvez également importer dans **HOPEX** les paramètres physiques définis sur des objets rétro-générés. Voir la "Rétro-génération des propriétés physiques", page 156.

Vous pouvez adapter un même modèle logique à plusieurs SGBD. Il n'est pas nécessaire de dupliquer les objets.

SGBD cibles

Pour définir un SGBD cible sur une base de données :

1. Ouvrez la fenêtre de propriétés de la base de données concernée.
2. Cliquez sur la page **Caractéristiques**.
3. Indiquez le champ **SGBD cible** dans le champ correspondant.

Voir aussi "Importer une version de SGBD", page 12.

Créer des propriétés physiques

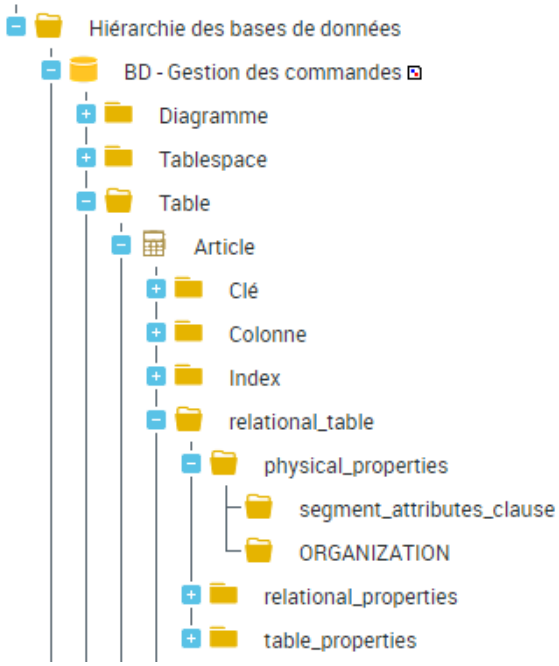
Pour créer des propriétés physiques sur les objets d'une base de données :

1. Dans le bureau, cliquez sur le menu de navigation puis sur **Données physiques**.
2. Dépliez le dossier **Bases de données** et cliquez sur **Hiérarchie des bases de données**.
   La liste des bases de données du référentiel apparaît dans la zone d'édition.
3. Dépliez le dossier et les sous-dossiers de la base de données concernée.

Les paramètres sont présentés sous forme arborescente, conformément à la grammaire SQL du SGBD considéré (référez-vous à la documentation SQL du SGBD).

Deux types de dossier sont présentés dans l'arbre :

- Les dossiers de navigation.
- Les groupes de paramètres que vous devez instancier.



Chaque groupe de paramètres, représenté par un objet "clause SQL", dispose d'une page de propriétés permettant de définir leur valeur.

Les clauses SQL ainsi définies sont accessibles comme des objets standard du référentiel. Il est par exemple possible de rechercher des objets SQL ayant une valeur donnée pour un paramètre.

Par défaut, les clauses ne sont pas réutilisables d'un objet à l'autre. Il est néanmoins possible de définir une clause pour un objet et de la relier à d'autres objets. Dans ce cas, toute modification de la clause affecte les objets qui l'utilisent.

Objets contenant des paramètres physiques

Tous les objets dans **HOPEX** ne supportent pas les paramètres physiques. Ceux-ci concernent uniquement :

- Les groupes de données
- Les tables
- Les index
- Les clusters

Créer une nouvelle clause

Pour définir les paramètres d'un objet :

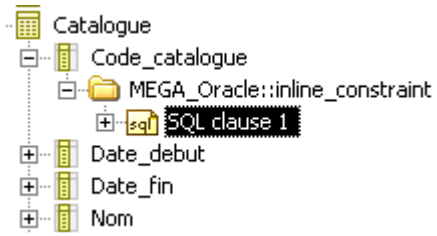
1. Cliquez avec le bouton droit sur le groupe de paramètres correspondant et sélectionnez **Nouveau** > **SQL clause**.
2. Ouvrez la fenêtre de propriétés de la clause et saisissez la valeur du paramètre à définir.

Relier une clause

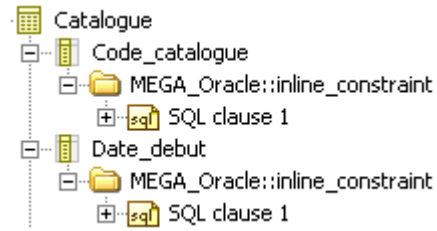
Vous pouvez affecter une même clause à plusieurs objets, à condition de relier le bon type de clause.

Prenons la base de données "Gestion des commandes" ayant pour SGBD Oracle 9i.

Sur la colonne "Code_catalogue", créez la "Clause 1" de type "inline_constraint".



Vous pouvez relier la "Clause 1" à une autre colonne. S'agissant du même type de clause, celle-ci est copiée sans problème sur la nouvelle colonne.



En revanche, si vous reliez la "Clause 1" à un objet dont le type est différent de celui défini initialement sur la "Clause 1" - par exemple "Storage_clause " - alors la "Clause 1" change de type pour prendre celui du dernier élément relié. Autrement dit, la "Clause 1" qui était de type "inline_constraint" prend le type "storage_clause". Ce changement se répercute sur les colonnes de départ auxquelles était reliée la "Clause 1".

Nommage des clauses

Cas standard

Par défaut, le nom de la clause prend le nom de la clause type à laquelle elle est rattachée. Lorsque vous rattachez une clause à un autre type, le nom s'adapte automatiquement.

Nommage spécifique

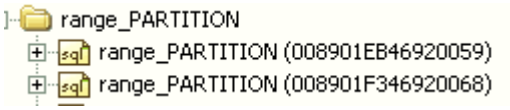
Vous avez la possibilité de donner un nom spécifique à une clause. Dans ce cas, le nom de la clause devient statique; le changement de clause type ne modifie pas le nom de la clause.

☛ *Vous pouvez revenir en mode dynamique en forçant le nom à vide.*

Un nommage spécifique est indispensable lorsqu'une clause est utilisée dans différents contextes (clause générique).

Clauses multiples

Pour un niveau donné, plusieurs clauses peuvent être rattachées à une même clause type. Afin de distinguer les différentes clauses, le nom de la clause est constitué du nom de la clause type puis de son hexaIdAbs.



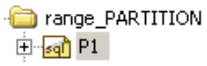
Nommage à partir d'une propriété

Il est possible de modifier le comportement standard en définissant une règle de nommage automatique pour un type de clauses SQL. Ce paramétrage s'effectue au niveau de la propriété _settings de la clause type. Dans l'exemple ci-dessous, le paramétrage effectué sur la clause type "range_PARTITION" pour Oracle 9i indique que le nom des clauses SQL de ce type se construit à partir de la valeur de la propriété PARTITION.

```
[NameIdentification]
AttrForNameIdentification=A3F2DE8C417E06C9
```

Une fois ce paramètre effectué, les noms des clauses SQL est automatiquement calculé à partir des valeurs de la propriété PARTITION.

Property	Value
Key compression	
PARTITION	P1
Value list	

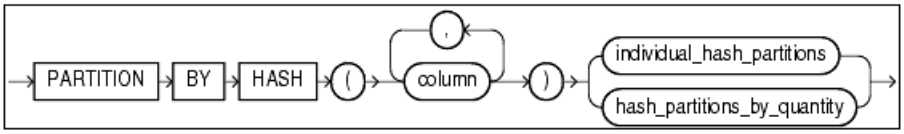


Le nom des clauses SQL n'est pas pris en compte dans la génération SQL. Dans l'exemple fourni, c'est bien la valeur de la propriété PARTITION qui alimente les scripts SQL générés.

Exemple de personnalisation d'un modèle physique

Vous pouvez partitionner une table afin de faciliter l'accès aux données ou de gérer différemment des blocs d'informations.

Supposons que vous vouliez partionner la table "Ligne de commande" de la base de données "Gestion des commandes" selon la méthode *by hash* d'Oracle. Cette méthode permet de calculer dynamiquement la partition d'une table.



Syntaxe de l'instruction Hash partitioning

Vérifiez que la base de données a pour SGBD cible Oracle 9i.

- Ouvrez sa fenêtre de propriétés et cliquez sur la page **Caractéristiques**. Le nom du SGBD est indiqué dans le champ **SGBD cible**.

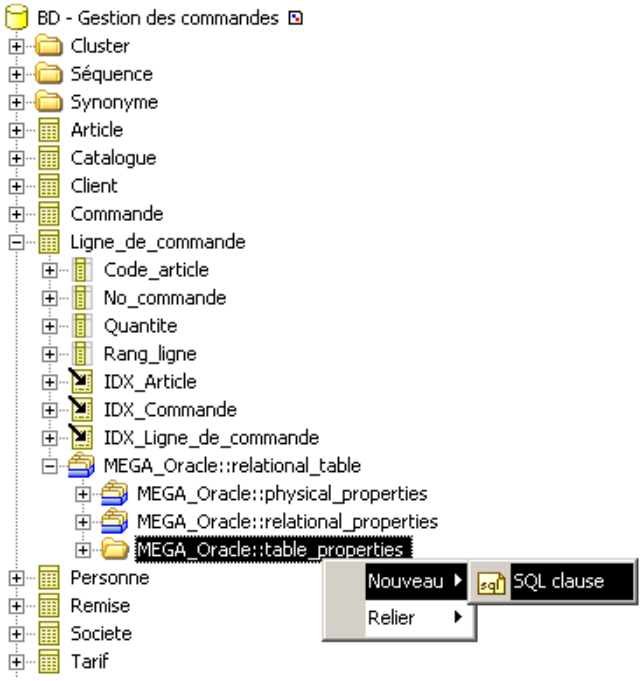
Affichez les propriétés physiques de la base de données "Gestion des commandes" :

1. Dans le bureau, cliquez sur le menu de navigation puis sur **Données physiques**.
2. Dépliez le dossier **Bases de données** et cliquez sur **Hiérarchie des bases de données**.
   La liste des bases de données du référentiel apparaît dans la zone d'édition.
3. Dépliez le dossier et les sous-dossiers de la base de données "Gestion des commandes" pour afficher les paramètres liés à la grammaire Oracle.

Pour partitionner la table "Ligne de commande" :

1. Dépliez la table "Ligne_de_commande".
2. Dépliez le groupe de paramètres "MEGA_Oracle::relational_table".

3. Cliquez avec le bouton droit sur le type de clause "MEGA_Oracle::table_properties" et sélectionnez **Nouveau** > **SQL clause**.



4. Nommez la clause "Ligne_de_commande/Table_properties". Elle apparaît dans l'arbre de navigation.
5. Sous cette clause, dépliez le groupe de paramètres "MEGA_Oracle::table_partitioning_clauses". Il contient les différents types de partionnement qu'il est possible de réaliser dans Oracle.
6. Sur le dossier "MEGA_Oracle::hash_partitioning", créez la clause "Ligne_de_commande/hash_partitioning".
7. Ouvrez sa fenêtre de propriétés.
8. Dans la page **PARTITION BY HASH**, indiquez les colonnes sur lesquelles s'applique le découpage. Pour cela, reliez les colonnes concernées par la partition.
9. Fermez la fenêtre de propriétés.
10. Sous la clause "Ligne_de_commande/hash_partitioning" sont disponibles deux types de clauses :
    - **individual_hash_partitions** : permet de nommer chaque partition.
    - **hash_partitions_by_quantity** : permet de définir le nombre de partitions que vous voulez créer.
11. Créez la clause "Ligne_de_commande/Hash_partition_by_quantity".
12. Ouvrez sa fenêtre de propriétés.
13. Cliquez sur la page **PARTITIONS**.
14. Dans le champ **Hash partition quantity**, indiquez le nombre de partitions. Ces partitions sont représentées par des groupements de données.
15. Sous le champ **STORE IN**, reliez les groupements de données.

Pour obtenir le script correspondant à ce partitionnement, cliquez avec le bouton droit sur la table "Ligne de commande" et sélectionnez **Générer le code**.

```
SPOOL \_MEGASQL.LST;
PROMPT ------------------------------------------------ ;
PROMPT  Compte-Rendu de génération ;
PROMPT ------------------------------------------------ ;
/* ---------------------------------------------------------------------------------------------------------------------
/* Begin of generation Oracle 9i for database BD___Gestion_des_commandes the 5 Juin 2006 at 17:38:54 */
/* ---------------------------------------------------------------------------------------------------------------------
/* ---------------------------------------------------------------------------------------------------------------------
/* Table Ligne_de_commande */
/* ---------------------------------------------------------------------------------------------------------------------
CREATE TABLE "Ligne_de_commande"
(
  "Code_article" CHAR(6),
  "No_commande" CHAR(5),
  "Quantite" NUMBER(7),
  "Rang_ligne" NUMBER(7),
  CONSTRAINT "PK_Ligne_de_commande" PRIMARY KEY
  (
    "No_commande",
    "Rang_ligne"
  )
)
PARTITION BY HASH ("No_commande","Rang_ligne")
PARTITIONS 5 STORE IN ("Tbs1","Tbs2","Tbs3","Tbs4") TABLESPACE "SYSTEM";
/* ---------------------------------------------------------------------------------------------------------------------
/* Foreign Key FK_Commande */
/* ---------------------------------------------------------------------------------------------------------------------
ALTER TABLE "Ligne_de_commande"
ADD CONSTRAINT "FK_Commande" FOREIGN KEY
(
  "No_commande"
) REFERENCES "Commande";
/* ---------------------------------------------------------------------------------------------------------------------
/* Foreign Key FK_Article */
/* ---------------------------------------------------------------------------------------------------------------------
ALTER TABLE "Ligne_de_commande"
ADD CONSTRAINT "FK_Article" FOREIGN KEY
(
  "Code_article"
) REFERENCES "Article";
SPOOL OFF;
DEFINE _EDITOR = "notepad.exe";
EDIT \_MEGASQL.LST
EXIT;
/* ---------------------------------------------------------------------------------------------------------------------
/* End of generation Oracle 9i for database BD___Gestion_des_commandes the 5 Juin 2006 at 17:38:55 */
/* ---------------------------------------------------------------------------------------------------------------------
```

Générer le fichier SQL

Lorsque vous avez personnalisé un objet, vous pouvez générer le fichier script correspondant afin d'observer les résultats, sans pour autant re-générer toute la base de données.

Par exemple, pour générer le fichier SQL d'un index :

- Faites un clic droit sur l'index et sélectionnez **Générer le code**.

Voir aussi "Générer des scripts SQL", page 1.

RÉTRO-GÉNÉRER DES TABLES

La rétro-génération permet, à partir de bases de données existantes, de créer dans le référentiel **HOPEX**, les tables et colonnes correspondantes.

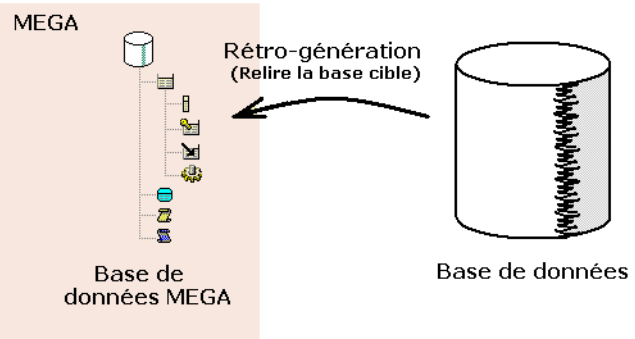
Cette rétro-génération est possible par extraction préalable. Voir "Utilitaire d'extraction ODBC", page 159.

Les tables et colonnes sont intégrées dans une base de données où elles peuvent être facilement maintenues et documentées.

Les points suivants sont abordés ici :

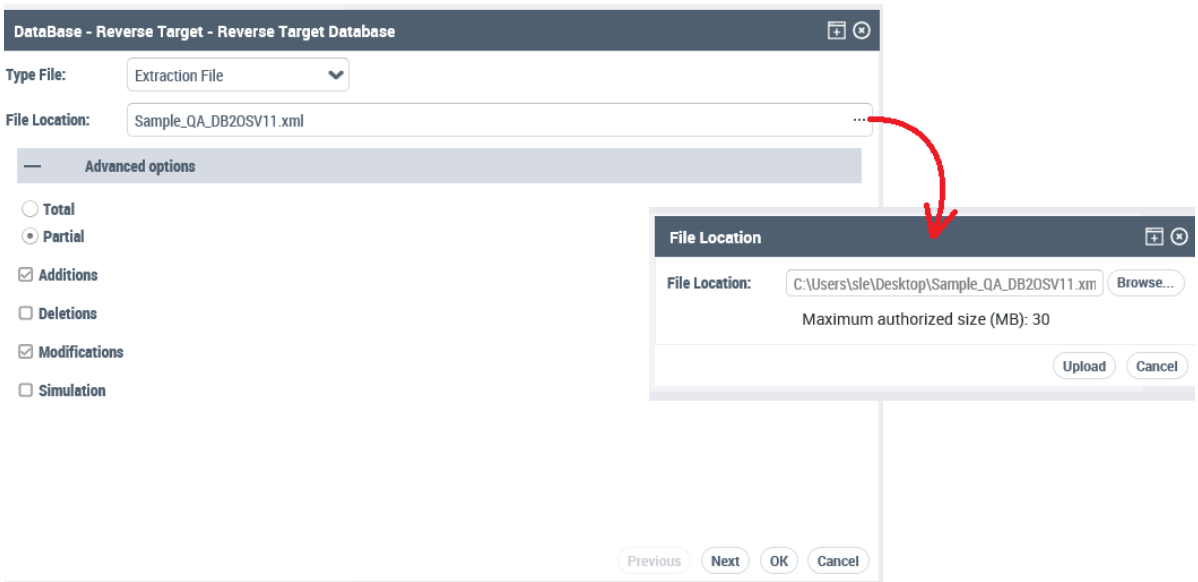
Les outils présentés dans les pages qui suivent sont disponibles avec un accès au référentiel en mode "Avancé".

LANCER LA RÉTRO-GÉNÉRATION



Pour lancer la rétro-génération :

1. Faites un clic droit sur la base de données et sélectionnez **Relire la base cible**.
   La fenêtre **Relire une base cible** apparaît.
2. Sélectionnez le **Type** de données à prendre en compte : "Fichier d'extraction" obtenu avec l'utilitaire d'extraction de données (voir "Utilitaire d'extraction ODBC", page 159).
3. Indiquez l'emplacement du fichier.



4. Précisez les **Options** :
   Si la base a déjà été rétro-générée, vous pouvez spécifier l'étendue de la **Réinitialisation**, qui peut être :
   - **Totale** : toutes les tables existantes sont supprimées.
   - **Partielle** et concerner uniquement les **Ajouts**, les **Suppressions** ou les **Modifications**.

   L'option **Simulation** permet de simuler l'opération, et de générer un compte-rendu indiquant les impacts sur le référentiel.
5. Cliquez sur le bouton **Suivant** pour lancer la rétro-génération.
   Des messages rendent compte du déroulement de l'exécution.
6. A la fin du traitement, un rapport affiche le détail de l'exécution.
   A la fin de la rétro-génération, les tables et les colonnes de la base sont créées ; en revanche, le diagramme relationnel doit être créé pour pouvoir consulter la base de données sur un modèle graphique.

> ☺ *Si certains datatypes non standard ont été créés dans le SGBD, il faut les rajouter dans le paramétrage pour qu'ils soient reconnus par* ***HOPEX.***

RÉTRO-GÉNÉRATION DES PROPRIÉTÉS PHYSIQUES

La rétro-génération permet également de créer dans **HOPEX** les propriétés physiques sur les objets d'une base de données.

Les propriétés physiques sont des paramètres permettant d'exprimer, pour un objet relationnel (table, index, etc.), la manière dont les informations vont être stockées au sein d'une base de données. Ces paramètres sont spécifiques à chaque SGBD et peuvent évoluer selon les versions d'un même SGBD.

Voir aussi "Ajouter des propriétés physiques aux objets d'une base de données", page 145.

Valeurs par défaut

Certaines propriétés du SGBD sont automatiquement rétro-générées dans **HOPEX** même lorsqu'elles n'ont pas été explicitement spécifiées.

Afin de ne pas récupérer ces valeurs par défaut, **HOPEX** fournit pour chaque SGBD et pour chaque version de SGBD une clause générique qui contient la liste de ces valeurs par défaut et les traite de façon spécifique.

Lors d'une rétro-génération, les propriétés des SGBD dont les valeurs sont égales aux valeurs définies dans la clause générique ne sont pas importées.

Vous pouvez activer la clause générique en important dans **HOPEX** le fichier .mol associé à chaque SGBD dans le dossier Mega_Std de votre installation.

Élimination des valeurs redondantes et transverses

Lors de la rétro-génération d'objets dans **HOPEX**, certaines propriétés physiques qui n'ont pas été clairement déterminées sont automatiquement restituées par le SGBD au travers d'un mécanisme d'héritage.

Avec Oracle par exemple, la valeur de la propriété PCTFREE dans une table, si elle n'a pas été spécifiée, est directement héritée de celle du tablespace qui lui est rattaché. Une telle valeur est dite transverse car elle est issue d'un héritage entre deux types d'objets distincts. Une valeur est dite redondante si l'héritage est issu d'objets de même type.

Lors de la rétro-génération, **HOPEX** ne récupère pas les valeurs transverses et redondantes.

Seule la gestion des valeurs redondantes peut être personnalisée .

Cas spécifiques

Propriétés physiques des tablespaces

Dans certains cas, la rétro-génération de propriétés physiques des objets requiert une connexion ODBC bénéficiant des droits Administrateur du SGBD. Par exemple, avec Oracle, la rétro-génération des propriétés physiques des Tablespaces nécessite que vous utilisiez un compte "System".

Rétro-génération des clusters

La rétro-génération d'un cluster dans Oracle s'effectue correctement si l'utilisateur de connexion vérifie l'une des deux conditions suivantes :

- L'utilisateur est le propriétaire du cluster
- L'utilisateur à un profil administrateur

Lorsque le cluster n'est pas accessible, il n'est pas rétro-généré.

Lorsque l'utilisateur voit le cluster mais n'est ni propriétaire, ni administrateur, le cluster est rétro-généré mais la liaison entre les colonnes du cluster et les colonnes des tables qui lui sont rattachées n'est pas rétro-générée dans **HOPEX**.

> ☛ *D'un point de vue technique, pour un utilisateur non administrateur, la rétro-génération s'appuie sur la vue oracle sys.all_clu_columns (relation entre les colonnes de cluster et les colonnes de tables). Cette vue permet de lire uniquement les informations relatives aux objets pour lesquels l'utilisateur est propriétaire.*

UTILITAIRE D'EXTRACTION ODBC

L'utilitaire d'extraction ODBC permet d'extraire la description d'une base de données accessible par le protocole ODBC. Cette description, obtenue en format structuré, peut être ensuite utilisée pour la rétro-génération dans **HOPEX** ou pour la génération en mode modification.

Les points suivants sont abordés ici :

EXTRAIRE LA DESCRIPTION D'UNE BASE DE DONNÉES

L'utilitaire d'extraction est destiné à être lancé à partir d'un poste qui ne dispose pas nécessairement de **HOPEX**, le résultat obtenu étant ensuite transféré sur un poste **HOPEX** pour être traité par la rétro-génération. Il peut également être utilisé pour la génération en mode modification.

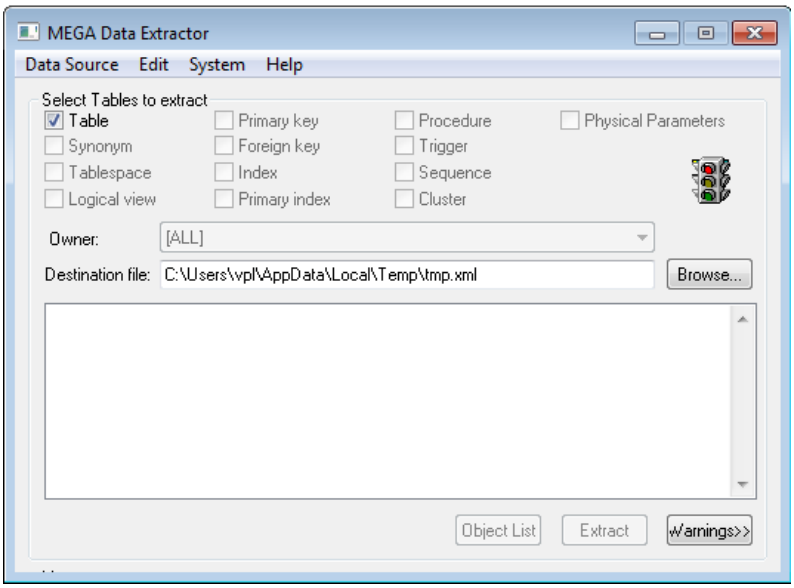
Pour l'extraction de données, le poste de travail doit être connecté à la base de données avec le protocole ODBC (voir la documentation de ce produit pour son installation). Le pilote de la base de données doit avoir un niveau de conformité ("conformance level") supérieur ou égal à 1.

Pour installer cet utilitaire, il faut copier dans un dossier les fichiers mgwdbx32.exe et mg_dbex.dll. Le cas échéant, le fichier odwdbex.ini doit être créé.

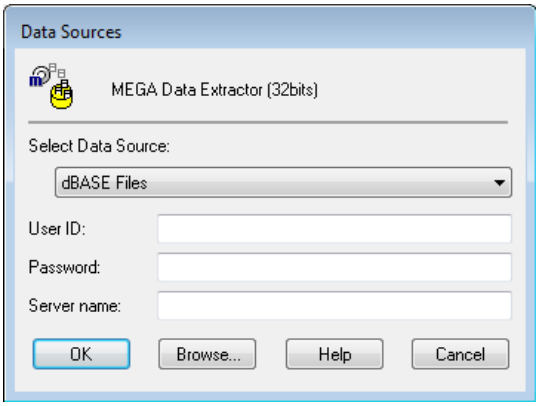
L'utilitaire d'extraction peut être installé indépendamment de **HOPEX Information Architecture**.

Pour extraire la description d'une base de données :

1. Dans le dossier d'installation d'**HOPEX**, sous le dossier "Utilities\MEGA Data Extractor", lancez l'utilitaire d'extraction mgwdbx32.exe.



2. Connectez-vous à une source de données en cliquant sur le menu **Data Source > Connect**.
   La fenêtre **Data Sources** s'affiche.



   La liste déroulante présente les connexions ODBC.

   ☛ *Cette liste est vide si ces connexions ne sont pas définies, ou n'ont pu être établies.*

3. Le bouton **Browse** permet d'accéder au gestionnaire ODBC pour définir une nouvelle source de données, ou installer un nouveau pilote.
4. En fonction de la connexion, il peut être nécessaire de préciser un code utilisateur (**User ID**), un mot de passe (**Password**) et un nom de serveur (**Server name**). Si d'autres paramètres sont requis par le pilote ODBC, ils seront demandés au début de la connexion.
5. Validez les informations saisies en cliquant sur **OK**.
   Le feu vert indique que la connexion est établie. Un message apparaît si la connexion ne s'est pas correctement établie. Il faut dans ce cas vérifier la définition de la source de données.

   Une fois la connexion établie, vous pouvez choisir les options d'extraction.

   ☛ *Si certaines des options restent grisées, c'est parce que le pilote ne les supporte pas.*

   ☺ *Pour obtenir des informations sur le protocole ODBC utilisé, activez la commande **ODBC informations** du menu **System**.*

6. Sélectionnez les éléments à extraire, en plus des tables et des colonnes. Par défaut, ces éléments sont tous sélectionnés.
   Toutes les tables auxquelles l'utilisateur a accès, qu'elles lui appartiennent ou non, sont présentées. Les tables *synonymes* peuvent également apparaître si la case correspondante a été cochée.

   🕮 *Un synonyme est un nom alternatif donné à un objet (table, vue, procédure stockée, synonyme et séquence). On peut définir un synonyme pour désigner un objet d'une autre base de données.*

   Il est possible de filtrer les tables par propriétaire (**Owner**), en le sélectionnant dans la zone correspondante. L'affichage de la liste des propriétaires, ainsi que celle de leurs tables peut prendre plusieurs secondes. L'extraction des tables prend plusieurs minutes.

Les éléments suivants sont inclus dans l'extraction :

- Clés primaires (**Primary keys**).
- Clés étrangères (**Foreign keys**).
- Index (**Index**) : il s'agit des index qui ne portent pas sur des clés primaires.
- Index primaires (**Primary index**) : il s'agit des index qui portent sur des clés primaires.

> **Les primitives ODBC permettant d'extraire ces éléments ne sont pas supportés par tous les pilotes des sources de données ; un message en rend compte dans le fichier compte-rendu. De plus, certains SGBD ne gèrent pas les concepts correspondants, qui sont alors ignorés.**

Le champ **Destination file** permet de préciser le chemin et le nom du fichier d'extraction ; le bouton **Browse** permet de parcourir les dossiers.

7. Après sélection des options d'extraction, appuyez sur **Extract** pour démarrer le traitement.
   Un message rend compte du nombre de tables extraites. L'activation du bouton **Warnings** permet de consulter simultanément le compte-rendu.

   > *Durant le traitement, le bouton **Extract** devient **Cancel** et permet d'interrompre l'extraction.*

   Il est possible de visualiser la liste des tables accessibles en activant le bouton **List Tables**, et d'extraire uniquement les tables sélectionnées par l'utilisateur dans la liste alors obtenue (toutes les tables sont sélectionnées par défaut).
8. A la fin de l'extraction, la commande **Edit > Report file** permet de consulter le compte-rendu ; la commande **Edit > Extraction file** permet de consulter le résultat.

Le fichier résultat pourra être ensuite transféré sur un poste qui dispose de **HOPEX Database Builder**, pour être exploité par la rétro-génération (voir "Rétro-générer des tables", page 153). Il contient la description de la base sous forme d'objets **HOPEX**.

Une fois l'opération d'extraction terminée :

- Activez la commande **Disconnect** du menu **Data Source** pour vous déconnecter de la source de données.
- Reconnectez-vous à une autre source de données ou quittez la fenêtre en activant la commande **Exit** du menu **Data Source**.

Fichier de compte-rendu de l'extraction

Le fichier de compte-rendu de l'extraction des tables par l'utilitaire d'extraction ODBC s'appelle <FIC>_CRD.TXT où <FIC> représente les trois premiers caractères du nom du fichier résultat.

Il contient la liste des tables relues.

Exemple :

```
==================================================
 Data Source Extracting : DATASOURCE
==================================================
Table    : OWNER.NOMTABLE1
Table    : OWNER.NOMTABLE2
 (suite)
==================================================
End of extraction
==================================================
```

Fichier résultat de l'extraction

Le fichier résultat de l'extraction contient la description des tables et des colonnes, résultat de la relecture. Ce fichier porte l'extension ".xml".

Exemple de fichier d'extraction :

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <XMDB Name="Xmdb" Version="2.0">
  - <DATABASE Name="Database">
    - <TABLE Name="TABLE.AURORA036IIOP036SYSTEM036PROPERTIES" DBBName="AURORA$IIOP$SYSTEM$PROPERTIES">
      - <TBLCOL Name="TBLCOL.AURORA036IIOP036SYSTEM036PROPERTIES.KEY" Order="1" DBBName="KEY" Decimale="0"
          Length="200" NotNull="N">
          <COLDATATYPE Name="COLDATATYPE.AURORA036IIOP036SYSTEM036PROPERTIES.KEY"
            DataType="SQL_VARCHAR" TypeTech="SQL_Oracle8VARCHAR2" />
        </TBLCOL>
      - <TBLCOL Name="TBLCOL.AURORA036IIOP036SYSTEM036PROPERTIES.VALUE" Order="2" DBBName="VALUE"
          Decimale="0" Length="1024" NotNull="N">
          <COLDATATYPE Name="COLDATATYPE.AURORA036IIOP036SYSTEM036PROPERTIES.VALUE"
            DataType="SQL_VARCHAR" TypeTech="SQL_Oracle8VARCHAR2" />
        </TBLCOL>
      - <INDEX Name="INDEX.AURORA036IIOP036SYSTEM036PROPERTIES.KEY095C" DBBName="KEY_C" Type="U">
          <IDXCOL Name="IDXCOL.AURORA036IIOP036SYSTEM036PROPERTIES.KEY095C.KEY"
            Reference="TBLCOL.AURORA036IIOP036SYSTEM036PROPERTIES.KEY" />
        </INDEX>
      </TABLE>
    - <TABLE Name="TABLE.JAVA036HTTP036DEPLOYMENT036DIGEST036"
        DBBName="JAVA$HTTP$DEPLOYMENT$DIGEST$">
      - <TBLCOL Name="TBLCOL.JAVA036HTTP036DEPLOYMENT036DIGEST036.USERNAME" Order="1"
          DBBName="USERNAME" Decimale="0" Length="30" NotNull="M">
          <COLDATATYPE Name="COLDATATYPE.JAVA036HTTP036DEPLOYMENT036DIGEST036.USERNAME"
            DataType="SQL_VARCHAR" TypeTech="SQL_Oracle8VARCHAR2" />
        </TBLCOL>
      - <TBLCOL Name="TBLCOL.JAVA036HTTP036DEPLOYMENT036DIGEST036.IS095DOC095ROOT" Order="4"
          DBBName="IS_DOC_ROOT" Decimale="0" Length="1" NotNull="N">
          <COLDATATYPE Name="COLDATATYPE.JAVA036HTTP036DEPLOYMENT036DIGEST036.IS095DOC095ROOT"
            DataType="SQL_DECIMAL" TypeTech="SQL_Oracle8DECIMAL" />
        </TBLCOL>
      </TABLE>
    </DATABASE>
  </XMDB>
```

PERSONNALISER L'EXTRACTION ODBC

Lorsque l'extraction est incomplète ou ne correspond pas à vos besoins, vous pouvez personnaliser l'extraction avec le fichier Odwdbex.ini. Ce paramétrage dépend du driver ODBC que vous utilisez.

Vous pouvez personnaliser l'extraction de différentes façons, en utilisant :

- les API standard ODBC, disponibles pour les concepts principaux (Table, Colonne, Clé, Index, etc.)
- des requêtes **HOPEX**, fournies en remplacement des API standard ODBC
- des requêtes personnalisées.

Par défaut, on utilise les API standard ODBC pour les concepts principaux et des requêtes **HOPEX** pour les autres concepts. Pour certains drivers ODBC, on utilise des requêtes **HOPEX** pour les concepts principaux, le résultat obtenu par les API standard ODBC étant incomplet.

Utilisation du fichier Odwdbex.ini et des requêtes personnalisées

Pour personnaliser l'extraction :

1. Renseignez-vous auprès de votre administrateur de base de données pour obtenir les requêtes personnalisées correspondant à votre driver ODBC qui permettent de sélectionner les objets (Ex : clés primaires, clés étrangères, séquences, etc).
2. Dans le dossier "All users" de Windows, créez un fichier nommé "Odwdbex.ini" (exemple: C:\Documents and Settings\All Users\ApplicationData\Mega\Odwdbex.ini).
3. Éditez le fichier et ajoutez les requêtes pour les concepts dont vous voulez gérer le comportement. Les concepts qui ne sont pas cités ici restent inchangés.

   [<DBMS Name>]

   PRIMARY KEYS="Requête personnalisée"

   FOREIGN KEYS="Requête personnalisée"

   TBLCOLUMNS="Requête personnalisée"

   ...

La valeur de <DBMS Name> dépend de l'utilitaire ODBC. Pour avoir la valeur appropriée :

1. Lancez l'outil d'extraction **HOPEX** (mgwdbx32.exe).
2. Dans le menu **Data source**, sélectionnez la source de données.
3. Cliquez ensuite sur **System** > **ODBC Informations**.
4. Lisez le "DBMS Name".

Vous pouvez éditer le fichier Odwdbex.ini en cliquant sur le menu **System** > **Edit Odwdbex.ini** dans l'outil d'extraction **HOPEX**. Assurez-vous que ce fichier est archivé.

Pour plus d'informations sur le format des requêtes, voir "Format des clauses SELECT", page 166

Utilisation des API standard ODBC

Pour forcer l'utilisation des API ODBC :

1. Éditez le fichier Odwdbex.ini.
2. Au niveau de chaque concept concerné, modifiez la stratégie d'extraction en utilisant le mot clé : USE_DRIVER_ODBC.

```
Exemple dans le fichier ODWDBEX.INI :

[<Dbms Name>]

INDEXES=USE_DRIVER_ODBC
```

FORMAT DES CLAUSES SELECT

Il est important de respecter la syntaxe indiquée ; en particulier, les "1" ne doivent pas être omis. De plus, il faut noter que les clauses doivent être écrites sur une seule ligne dans le fichier ODWDBEX.INI.

Clés primaires

```
SELECT
1,
TABLE_OWNER,
TABLE_NAME,
COLUMN_NAME,
KEY_SEQUENCE,
PK_NAME
FROM ...
WHERE ...
```

- TABLE_OWNER : propriétaire de la table de la clé primaire
- TABLE_NAME : nom de la table de la clé primaire
- COLUMN_NAME : nom de la colonne de la clé primaire
- KEY_SEQUENCE : numéro de la colonne dans la clé (à partir de 1)
- PK_NAME : nom de la clé primaire ; "1" si ce nom n'est pas supporté par le SGBD.

Clés étrangères

```
SELECT
1,
PKTABLE_OWNER,
PKTABLE_NAME,
1,
1,
FKTABLE_OWNER,
FKTABLE_NAME,
FKCOLUMN_NAME,
KEY_SEQ,
UPDATE_RULE,
DELETE_RULE,
FK_NAME
FROM ...
WHERE...
```

- PKTABLE_OWNER : nom du propriétaire de la table de la clé primaire (table de référence)
- PKTABLE_NAME : nom de la table de la clé primaire
- FKTABLE_OWNER : nom du propriétaire de la table de la clé étrangère
- FKTABLE_NAME : nom de la table de la clé étrangère
- FKCOLUMN_NAME : nom de la colonne de la clé étrangère
- KEY_SEQ : numéro de la colonne dans la clé (à partir de 1)
- UPDATE_RULE : "R" : Restrict, "C" : Cascade
- DELETE_RULE : "R" : Restrict, "C" : Cascade
- FK_NAME : nom de la clé étrangère ; "1" si ce nom n'est pas supporté par le SGBD.

Index

```
SELECT
1,
TABLE_OWNER,
TABLE_NAME,
NON_UNIQUE,
1,
INDEX_NAME,
TYPE,
SEQ_IN_INDEX,
COLUMN_NAME,
COLLATION
FROM ...
WHERE...
```

- TABLE_OWNER : nom du propriétaire de la table de la concernée par la statistique ou par l'index
- TABLE_NAME : nom de la table de l'index
- NON_UNIQUE : les index doivent avoir une valeur unique
- INDEX_NAME : nom de l'index
- TYPE : Type de l'index
- SEQ_IN_INDEX : numéro de la colonne dans la clé (à partir de 1)
- COLUMN_NAME : nom de la colonne
- COLLATION : tri de la colonne ; "A" ascendant, "D" descendant

Colonnes

```
SELECT
1,
COLUMN_OWNER,
TABLE_NAME,
COLUMN_NAME,
DataType ODBC,
DataType Name,
Precision,
Lentgh,
Scale,
1,
NULLABLE,
COMMENT,
DEFAULT_VALUE,
1,
1,
1,
Order
WHERE [Jointure sur <MEGA:OWNER><MEGA:OBJECT_NAME>]
```

- <MEGA:OWNER> est remplacé par l'utilisateur, le Schéma ou "".
- <MEGA:OBJECT_NAME>] est remplacé par le nom de la table.
- COLUMN_OWNER : nom de la colonne, chaîne de 128 caractères.
- TABLE_NAME : nom de la table, chaîne de 128 caractères.
- DataType ODBC : type de données sous la forme d'un nombre entier. Cette valeur est la valeur des types de données ODBC donc voici un rappel :

```
# -1 (SQL_LONGVARCHAR)
# -2 (SQL_BINARY
# -3 (SQL_VARBINARY)
# -4 (SQL_LONGVARBINARY)
# -5 (SQL_BIGINT)
# -6 (SQL_TINYINT)
# -7 (SQL_BIT)
# 0 (SQL_UNKNOWN_TYPE)
# 1 (SQL_CHAR)
# 2 (SQL_NUMERIC)
# 3 (SQL_DECIMAL)
# 4 (SQL_INTEGER)
```

```
# 5 (SQL_SMALLINT)
# 6 (SQL_FLOAT)
# 7 (SQL_REAL)
# 8 (SQL_DOUBLE)
# 9 (SQL_DATE)
# 10 (SQL_TIME)
# 11 (SQL_TIMESTAMP)
# 12 (SQL_VARCHAR)
```

- DataType Name : nom du type de données, chaîne de 128 caractères. Il est construit comme ceci : "SQL_<DbmsName><String>"
- Precision : longueur dans MEGA si "Length" est vide.
- Lentgh : longueur dans MEGA si plus grand que 0.
- Scale : nombre entier
- NULLABLE : nombre entier spécifiant si la colonne peut être NULL . Valeurs ODBC possible : 0 (SQL_NO_NULLS), 1 (SQL_NULLABLE) ou 3 (SQL_NULL_WITH_DEFAULT).
- COMMENT : commentaire de la colonne, chaîne de 1257 caractères.
- DEFAULT_VALUE : valeur par défaut de la colonne, chaîne de 1257 caractères.

TABLEAUX DE CORRESPONDANCES ENTRE TYPES PIVOTS ET DATATYPES

Les tableaux suivants donnent les correspondances entre les types pivots et les datatypes des différents SGBD supportés et leurs versions.

DB2 OS/390 VERSION 5

Pivot --> Datatype (DB2 OS/390 Version 5)

Pivot	Condition	Datatype
P-AutoIdentifier		INTEGER
P-Binary		CHAR(@L) FOR BIT DATA
P-Boolean		CHAR(@L) FOR BIT DATA
P-Byte		CHAR (1) FOR BIT DATA
P-Character	Not Unicode and (L=0 or L ø)	CHAR
	Not Unicode and 0<L<256	CHAR(@L)
	Unicode and (L<256 or L ø)	GRAPHIC(@L)
	Not Unicode and L>255	VARCHAR(@L)
	Unicode and L>255	VARGRAPHIC(@L)
P-Currency		DECIMAL(@L,@D)
P-Date		DATE
P-Datetime		TIMESTAMP
P-Decimal	L=0 or L ø	DECIMAL
	L>0 and D ø	DECIMAL(@L)
	L>0 and D not ø	DECIMAL(@L,@D)
P-Double		DOUBLE
P-Float	L=0 or L ø	FLOAT
	L<>0	FLOAT(@L)
P-Integer		INTEGER
P-Long Integer		INTEGER
P-Long Real		REAL

Pivot --> Datatype (DB2 OS/390 Version 5)

Pivot	Condition	Datatype
P-Multimedia		LONG VARCHAR FOR BIT DATA
P-Numeric	L not ø and D not ø	DECIMAL(@L,@D)
	L>9 and D ø	FLOAT(@L)
	4<L<10 and D ø	INTEGER
	L<5 or L ø	SMALLINT
P-Real		REAL
P-Smallint		SMALLINT
P-String		LONG VARCHAR
P-Text		LONG VARCHAR
P-Time		TIME
P-Timestamp		TIMESTAMP
P-Tinyint		SMALLINT
P-Varbinary		VARCHAR(@L) FOR BIT DATA
P-Varchar	Not Unicode and (L=0 or L ø)	VARCHAR
	Not Unicode and L<>0	VARCHAR(@L)
	Unicode	VARGRAPHIC(@L)

Datatype --> Pivot (DB2 OS/390 Version 5)

Datatype	Condition	Pivot
CHAR		P-Character
CHAR (1) FOR BIT DATA		P-Byte
CHAR(L)		P-Character
CHAR(L) FOR BIT DATA		P-Boolean
DATE		P-Date
DECIMAL		P-Decimal

Datatype --> Pivot (DB2 OS/390 Version 5)

Datatype	Condition	Pivot
DECIMAL(L)		P-Decimal
DECIMAL(L,D)		P-Decimal
DOUBLE		P-Double
FLOAT		P-Float
FLOAT(L)		P-Float
INTEGER		P-Integer
LONG VARCHAR		P-Text
LONG VARCHAR FOR BIT DATA		P-Multimedia
REAL		P-Real
SMALLINT		P-Smallint
TIME		P-Time
TIMESTAMP		P-Timestamp
VARCHAR		P-Varchar
VARCHAR(L)		P-Varchar
VARCHAR(L) FOR BIT DATA		P-Varbinary

DB2 OS/390 VERSION 7

Pivot --> Datatype (DB2 OS/390 Version 7)

Pivot	Condition	Datatype
P-AutoIdentifier		INTEGER
P-Binary		CHAR(@L) FOR BIT DATA
P-Boolean		CHAR(@L) FOR BIT DATA
P-Byte		CHAR (1) FOR BIT DATA
P-Character	Not Unicode and (L=0 or L ø)	CHAR
	Not Unicode and 0<L<256	CHAR(@L)
	Unicode and (L<256 or L ø)	GRAPHIC(@L)
	Not Unicode and L>255	VARCHAR(@L)
	Unicode and L>255	VARGRAPHIC(@L)
P-Currency		DECIMAL(@L, @D)
P-Date		DATE
P-Datetime		TIMESTAMP
P-Decimal	L=0 or L ø	DECIMAL
	L>0 and D ø	DECIMAL(@L)
	L>0 and D not ø	DECIMAL(@L, @D)
P-Double		DOUBLE
P-Float	L=0 or L ø	FLOAT
	L<>0	FLOAT(@L)
P-Integer		INTEGER
P-Long Integer		INTEGER
P-Long Real		REAL

Pivot --> Datatype (DB2 OS/390 Version 7)

Pivot	Condition	Datatype
P-Multimedia		LONG VARCHAR FOR BIT DATA
P-Numeric	L not ø and D not ø	DECIMAL(@L, @D)
	L>9 and D ø	FLOAT(@L)
	4<L<10 and D ø	INTEGER
	(L<5 and D ø) or L ø	SMALLINT
P-Real		REAL
P-Smallint		SMALLINT
P-String		LONG VARCHAR
P-Text		LONG VARCHAR
P-Time		TIME
P-Timestamp		TIMESTAMP
P-Tinyint		SMALLINT
P-Varbinary		VARCHAR(@L) FOR BIT DATA
P-Varchar	Not Unicode and (L=0 or L ø)	VARCHAR
	Not Unicode and L<>0	VARCHAR(@L)
	Unicode	VARGRAPHIC(@L)

Datatype --> Pivot (DB2 OS/390 Version 7)

Datatype	Condition	Pivot
CHAR		P-Character
CHAR (1) FOR BIT DATA		P-Byte
CHAR(L)		P-Character
CHAR(L) FOR BIT DATA		P-Boolean
DATE		P-Date
DECIMAL		P-Decimal

Datatype --> Pivot (DB2 OS/390 Version 7)

Datatype	Condition	Pivot
DECIMAL(L)		P-Decimal
DECIMAL(L, D)		P-Decimal
DOUBLE		P-Double
FLOAT		P-Float
FLOAT(L)		P-Float
INTEGER		P-Integer
LONG VARCHAR		P-Text
LONG VARCHAR FOR BIT DATA		P-Multimedia
REAL		P-Real
SMALLINT		P-Smallint
TIME		P-Time
TIMESTAMP		P-Datetime
VARCHAR(L)		P-Varchar
VARCHAR(L) FOR BIT DATA		P-Varbinary

DB2 FOR z/OS VERSION 8

Pivot --> Datatype (DB2 for z/OS Version 8)

Pivot	Condition	Datatype
P-AutoIdentifier		INTEGER
P-Binary		CHAR(@L) FOR BIT DATA
P-Boolean		CHAR(@L) FOR BIT DATA
P-Byte		CHAR (1) FOR BIT DATA
P-Character	Not Unicode and (L=0 or L ø)	CHAR
	Not Unicode and 0<L<256	CHAR(@L)
	Unicode and (L<256 or L ø)	GRAPHIC(@L)
	Not Unicode and L>255	VARCHAR(@L)
	Unicode and L>255	VARGRAPHIC(@L)
P-Currency		DECIMAL(@L, @D)
P-Date		DATE
P-Datetime		TIMESTAMP
P-Decimal	L=0 or L ø	DECIMAL
	L>0 and D ø	DECIMAL(@L)
	L>0 and D not ø	DECIMAL(@L, @D)
P-Double		DOUBLE
P-Float	L=0 or L ø	FLOAT
	L<>0	FLOAT(@L)
P-Integer		INTEGER
P-Long Integer		INTEGER
P-Long Real		REAL

Pivot --> Datatype (DB2 for z/OS Version 8)

Pivot	Condition	Datatype
P-Multimedia		LONG VARCHAR FOR BIT DATA
P-Numeric	L not ø and D not ø	DECIMAL(@L, @D)
	L>9 and D ø	FLOAT(@L)
	4<L<10 and D ø	INTEGER
	(L<5 and D ø) or L ø	SMALLINT
P-Real		REAL
P-Smallint		SMALLINT
P-String		LONG VARCHAR
P-Text		LONG VARCHAR
P-Time		TIME
P-Timestamp		TIMESTAMP
P-Tinyint		SMALLINT
P-Varbinary		VARCHAR(@L) FOR BIT DATA
P-Varchar	Not Unicode and (L=0 or L ø)	VARCHAR
	Not Unicode and L<>0	VARCHAR(@L)
	Unicode	VARGRAPHIC(@L)

Datatype --> Pivot (DB2 for z/OS Version 8)

Datatype	Condition	Pivot
CHAR		P-Character
CHAR (1) FOR BIT DATA		P-Byte
CHAR(L)		P-Character
CHAR(L) FOR BIT DATA		P-Boolean
DATE		P-Date
DECIMAL		P-Decimal

Datatype --> Pivot (DB2 for z/OS Version 8)

Datatype	Condition	Pivot
DECIMAL(L)		P-Decimal
DECIMAL(L, D)		P-Decimal
DOUBLE		P-Double
FLOAT		P-Float
FLOAT(L)		P-Float
INTEGER		P-Integer
LONG VARCHAR		P-Text
LONG VARCHAR FOR BIT DATA		P-Multimedia
REAL		P-Real
SMALLINT		P-Smallint
TIME		P-Time
TIMESTAMP		P-Datetime
VARCHAR(L)		P-Varchar
VARCHAR(L) FOR BIT DATA		P-Varbinary

DB2 UNIVERSAL DATABASE VERSION 5

Pivot --> Datatype (DB2 Universal Database Version 5)

Pivot	Condition	Datatype
P-AutoIdentifier		INTEGER
P-Binary		CHAR(@L) FOR BIT DATA
P-Boolean		CHAR(@L) FOR BIT DATA
P-Byte		CHAR (1) FOR BIT DATA
P-Character	Not Unicode and (L=0 or L ø)	CHAR
	Not Unicode and 0<L<255	CHAR(@L)
	Unicode and (L<256 or L ø)	GRAPHIC(@L)
	Not Unicode and L>254	VARCHAR(@L)
	Unicode and L>255	VARGRAPHIC(@L)
P-Currency		DECIMAL(@L,@D)
P-Date		DATE
P-Datetime		TIMESTAMP
P-Decimal	L=0 or L ø	DECIMAL
	L>0 and D ø	DECIMAL(@L)
	L>0 and D not ø	DECIMAL(@L,@D)
P-Double		DOUBLE
P-Float	L=0 or L ø	FLOAT
	L<>0	FLOAT(@L)
P-Integer		INTEGER
P-Long Integer		BIGINT
P-Long Real		REAL
P-Multimedia		BLOB(@L)

Pivot --> Datatype (DB2 Universal Database Version 5)

Pivot	Condition	Datatype
P-Numeric	L not ø and D not ø	DECIMAL(@L,@D)
	L>9 and D ø	FLOAT(@L)
	4<L<10 and D ø	INTEGER
	(L<5 and D not ø) or L ø	SMALLINT
P-Real		REAL
P-Smallint		SMALLINT
P-String		LONG VARCHAR
P-Text		LONG VARCHAR
P-Time		TIME
P-Timestamp		TIMESTAMP
P-Tinyint		SMALLINT
P-Varbinary		VARCHAR(@L) FOR BIT DATA
P-Varchar	Not Unicode and (L=0 or L ø)	VARCHAR
	Not Unicode and L<>0	VARCHAR(@L)
	Unicode	VARGRAPHIC(@L)

Datatype --> Pivot (DB2 Universal Database Version 5)

Datatype	Condition	Pivot
BIGINT		P-Long Integer
BLOB(L)		P-Multimedia
CHAR		P-Character
CHAR (1) FOR BIT DATA		P-Byte
CHAR(L)		P-Character
CHAR(L) FOR BIT DATA		P-Boolean
DATE		P-Date
DECIMAL		P-Decimal

Datatype --> Pivot (DB2 Universal Database Version 5)

Datatype	Condition	Pivot
DECIMAL(L)		P-Decimal
DECIMAL(L,D)		P-Decimal
DOUBLE		P-Double
FLOAT		P-Float
FLOAT(L)		P-Float
INTEGER		P-Integer
LONG VARCHAR		P-Text
REAL		P-Real
SMALLINT		P-Smallint
TIME		P-Time
TIMESTAMP		P-Timestamp
VARCHAR		P-Varchar
VARCHAR(L)		P-Varchar
VARCHAR(L) FOR BIT DATA		P-Varbinary

DB2 UNIVERSAL DATABASE VERSION 7

Pivot --> Datatype (DB2 Universal Database Version 7)

Pivot	Condition	Datatype
P-AutoIdentifier		INTEGER
P-Binary		CHAR(@L) FOR BIT DATA
P-Boolean		CHAR(@L) FOR BIT DATA
P-Byte		CHAR (1) FOR BIT DATA
P-Character	Not Unicode and (L=0 or L ø)	CHAR
	Not Unicode and 0<L<255	CHAR(@L)
	Unicode and (L<256 or L ø)	GRAPHIC(@L)
	Not Unicode and L>254	VARCHAR(@L)
	Unicode and L>255	VARGRAPHIC(@L)
P-Currency		DECIMAL(@L,@D)
P-Date		DATE
P-Datetime		TIMESTAMP
P-Decimal	L=0 or L ø	DECIMAL
	L>0 and D ø	DECIMAL(@L)
	L>0 and L not ø	DECIMAL(@L,@D)
P-Double		DOUBLE
P-Float	L=0 or L ø	FLOAT
	L<>0	FLOAT(@L)
P-Integer		INTEGER
P-Long Integer		BIGINT
P-Long Real		REAL
P-Multimedia		BLOB(@L)

Pivot --> Datatype (DB2 Universal Database Version 7)

Pivot	Condition	Datatype
P-Numeric	L not ø and D not ø	DECIMAL(@L,@D)
	L>9 and D ø	FLOAT(@L)
	4<L<10 and D ø	INTEGER
	L<5 or L ø	SMALLINT
P-Real		REAL
P-Smallint		SMALLINT
P-String		LONG VARCHAR
P-Text		LONG VARCHAR
P-Time		TIME
P-Timestamp		TIMESTAMP
P-Tinyint		SMALLINT
P-Varbinary		VARCHAR(@L) FOR BIT DATA
P-Varchar	Not Unicode and (L=0 or L ø)	VARCHAR
	Not Unicode and L<>0	VARCHAR(@L)
	Unicode	VARGRAPHIC(@L)

Datatype --> Pivot (DB2 Universal Database Version 7)

Datatype	Condition	Pivot
BIGINT		P-Long Integer
BLOB(L)		P-Multimedia
CHAR		P-Character
CHAR (1) FOR BIT DATA		P-Byte
CHAR(L)		P-Character
CHAR(L) FOR BIT DATA		P-Boolean
DATE		P-Date
DECIMAL		P-Decimal

Datatype --> Pivot (DB2 Universal Database Version 7)

Datatype	Condition	Pivot
DECIMAL(L)		P-Decimal
DECIMAL(L,D)		P-Decimal
DOUBLE		P-Double
FLOAT		P-Float
FLOAT(L)		P-Float
INTEGER		P-Integer
LONG VARCHAR		P-Text
REAL		P-Real
SMALLINT		P-Smallint
TIME		P-Time
TIMESTAMP		P-Timestamp
VARCHAR		P-Varchar
VARCHAR(L)		P-Varchar
VARCHAR(L) FOR BIT DATA		P-Varbinary

DB2 UNIVERSAL DATABASE VERSION 8

Pivot --> Datatype (DB2 Universal Database Version 8)

Pivot	Condition	Datatype
P-AutoIdentifier		INTEGER
P-Byte		CHAR (1) FOR BIT DATA
P-Character	Not Unicode and (L=0 or L ø)	CHAR
	Not Unicode and 0<L<255	CHAR(@L)
	Unicode and (L<256 or L ø)	GRAPHIC(@L)
	Not Unicode and L>254	VARCHAR(@L)
	Unicode and L>255	VARGRAPHIC(@L)
P-Currency		DECIMAL(@L,@D)
P-Date		DATE
P-Datetime		TIMESTAMP
P-Decimal	L=0 or L ø	DECIMAL
	L>0 and D ø	DECIMAL(@L)
	L>0 and L not ø	DECIMAL(@L,@D)
P-Double		DOUBLE
P-Float	L=0 or L ø	FLOAT
	L<>0	FLOAT(@L)
P-Integer		INTEGER
P-Long Integer		BIGINT
P-Long Real		REAL
P-Numeric	L not ø and D not ø	DECIMAL(@L,@D)
	L>9 and D ø	FLOAT(@L)
	4<L<10 and D ø	INTEGER
	L<5 or L ø	SMALLINT

Pivot --> Datatype (DB2 Universal Database Version 8)

Pivot	Condition	Datatype
P-Real		REAL
P-Smallint		SMALLINT
P-String		LONG VARCHAR
P-Text		LONG VARCHAR
P-Time		TIME
P-Timestamp		TIMESTAMP
P-Tinyint		SMALLINT
P-Varbinary		VARCHAR(@L) FOR BIT DATA
P-Varchar	Not Unicode and (L=0 or L ø)	VARCHAR
	Not Unicode and L<>0	VARCHAR(@L)
	Unicode	VARGRAPHIC(@L)

Datatype --> Pivot (DB2 Universal Database Version 8)

Datatype	Condition	Pivot
BIGINT		P-Long Integer
CHAR		P-Character
CHAR (1) FOR BIT DATA		P-Byte
CHAR(L)		P-Character
DATE		P-Date
DECIMAL		P-Decimal
DECIMAL(L)		P-Decimal
DECIMAL(L,D)		P-Decimal
DOUBLE		P-Double
FLOAT		P-Float
FLOAT(L)		P-Float
INTEGER		P-Integer

Datatype --> Pivot (DB2 Universal Database Version 8)

Datatype	Condition	Pivot
LONG VARCHAR		P-Text
REAL		P-Real
SMALLINT		P-Smallint
SMALLINT		P-Tinyint
TIME		P-Time
TIMESTAMP		P-Timestamp
VARCHAR		P-Varchar
VARCHAR(L)		P-Varchar
VARCHAR(L) FOR BIT DATA		P-Varbinary

DB2 UNIVERSAL DATABASE VERSION 9

Pivot --> Datatype (DB2 Universal Database Version 9)

Pivot	Condition	Datatype
P-AutoIdentifier		INTEGER
P-Binary		CHAR(@L) FOR BIT DATA
P-Boolean		CHAR(@L) FOR BIT DATA
P-Byte		CHAR (1) FOR BIT DATA
P-Character	Not Unicode and (L=0 or L ø)	CHAR
	Not Unicode and 0<L<255	CHAR(@L)
	Unicode and (L<256 or L ø)	GRAPHIC(@L)
	Not Unicode and L>254	VARCHAR(@L)
	Unicode and L>255	VARGRAPHIC(@L)
P-Currency		DECIMAL(@L,@D)
P-Date		DATE
P-Datetime		TIMESTAMP
P-Decimal	L=0 or L ø	DECIMAL
	L>0 and D ø	DECIMAL(@L)
	L>0 and L not ø	DECIMAL(@L,@D)
P-Double		DOUBLE
P-Float	L=0 or L ø	FLOAT
	L<>0	FLOAT(@L)
P-Integer		INTEGER
P-Long Integer		BIGINT
P-Long Real		REAL
P-Multimedia		BLOB(@L)

Pivot --> Datatype (DB2 Universal Database Version 9)

Pivot	Condition	Datatype
P-Numeric	L not ø and D not ø	DECIMAL(@L,@D)
	L>9 and D ø	FLOAT(@L)
	4<L<10 and D ø	INTEGER
	L<5 or L ø	SMALLINT
P-Real		REAL
P-Smallint		SMALLINT
P-String		LONG VARCHAR
P-Text		LONG VARCHAR
P-Time		TIME
P-Timestamp		TIMESTAMP
P-Tinyint		SMALLINT
P-Varbinary		VARCHAR(@L) FOR BIT DATA
P-Varchar	Not Unicode and (L=0 or L ø)	VARCHAR
	Not Unicode and L<>0	VARCHAR(@L)
	Unicode	VARGRAPHIC(@L)

Datatype --> Pivot (DB2 Universal Database Version 9)

Datatype	Condition	Pivot
BIGINT		P-Long Integer
BLOB(L)		P-Multimedia
CHAR		P-Character
CHAR (1) FOR BIT DATA		P-Byte
CHAR(L)		P-Character
CHAR(L) FOR BIT DATA		P-Boolean
CLOB(L)		CLOB
DATE		P-Date

Datatype --> Pivot (DB2 Universal Database Version 9)

Datatype	Condition	Pivot
DECIMAL		P-Decimal
DECIMAL(L)		P-Decimal
DECIMAL(L,D)		P-Decimal
DOUBLE		P-Double
FLOAT		P-Float
FLOAT(L)		P-Float
INTEGER		P-Integer
LONG VARCHAR		P-Text
REAL		P-Real
SMALLINT		P-Smallint
SMALLINT		P-Tinyint
TIME		P-Time
TIMESTAMP		P-Timestamp
VARCHAR		P-Varchar
VARCHAR(L)		P-Varchar
VARCHAR(L) FOR BIT DATA		P-Varbinary

DB2 VERSION 9 FOR OS

Pivot --> Datatype (DB2 Version 9 For OS)

Pivot	Condition	Datatype
P-AutoIdentifier		INTEGER
P-Binary		CHAR(@L) FOR BIT DATA
P-Boolean		CHAR(@L) FOR BIT DATA
P-Byte		CHAR (1) FOR BIT DATA
P-Character	Not Unicode and (L=0 or L ø)	CHAR
	Not Unicode and 0<L<256	CHAR(@L)
	Unicode and (L<256 or L ø)	GRAPHIC(@L)
	Not Unicode and L>255	VARCHAR(@L)
	Unicode and L>255	VARGRAPHIC(@L)
P-Currency		DECIMAL(@L, @D)
P-Date		DATE
P-Datetime		TIMESTAMP
P-Decimal	L=0 or L ø	DECIMAL
	L>0 and D ø	DECIMAL(@L)
	L>0 and D not ø	DECIMAL(@L, @D)
P-Double		DOUBLE
P-Float	L=0 or L ø	FLOAT
	L<>0	FLOAT(@L)
P-Integer		INTEGER
P-Long Integer		INTEGER
P-Long Real		REAL

Pivot --> Datatype (DB2 Version 9 For OS)

Pivot	Condition	Datatype
P-Multimedia		BLOB
		BLOB(@L)
		CLOB
		CLOB FOR MIXED DATA
		CLOB(@L)
		CLOB(@L) FOR MIXED DATA
		LONG VARCHAR FOR BIT DATA
P-Numeric	L not ø and D not ø	DECIMAL(@L, @D)
	L>9 and D ø	FLOAT(@L)
	4<L<10 and D ø	INTEGER
	(L<5 and D ø) or L ø	SMALLINT
P-Real		REAL
P-Smallint		SMALLINT
P-String		LONG VARCHAR
P-Text		LONG VARCHAR
P-Time		TIME
P-Timestamp		TIMESTAMP
P-Tinyint		SMALLINT
P-Varbinary	L <=1024	VARCHAR(@L) FOR BIT DATA
	L>1024	XML
P-Varchar	Not Unicode and (L=0 or L ø)	VARCHAR
	Not Unicode and L<>0	VARCHAR(@L)
	Unicode	VARGRAPHIC(@L)

Datatype --> Pivot (DB2 Version 9 For OS)

Datatype	Condition	Pivot
BLOB		P-Multimedia
BLOB(L)		P-Multimedia
CHAR		P-Character
CHAR (1) FOR BIT DATA		P-Byte
CHAR(L)		P-Character
CHAR(L) FOR BIT DATA		P-Boolean
CLOB		P-Multimedia
CLOB FOR MIXED DATA		P-Multimedia
CLOB(L)		P-Multimedia
CLOB(L) FOR MIXED DATA		P-Multimedia
DATE		P-Date
DECIMAL		P-Decimal
DECIMAL(L)		P-Decimal
DECIMAL(L, D)		P-Decimal
DOUBLE		P-Double
FLOAT		P-Float
FLOAT(L)		P-Float
INTEGER		P-Integer
LONG VARCHAR		P-Text
LONG VARCHAR FOR BIT DATA		P-Multimedia
REAL		P-Real
SMALLINT		P-Smallint
TIME		P-Time
TIMESTAMP		P-Datetime
VARCHAR(L)		P-Varchar

Datatype --> Pivot (DB2 Version 9 For OS)

Datatype	Condition	Pivot
VARCHAR(L) FOR BIT DATA		P-Varbinary
XML		P-Varbinary

INFORMIX DYNAMIC SERVER 7.3

Pivot --> Datatype (Informix Dynamic Server 7.3)

Pivot	Condition	Datatype
P-AutoIdentifier		SERIAL
P-Binary		BYTE
P-Boolean		SMALLINT
P-Byte		BYTE
P-Character		CHAR(@L)
P-Currency		MONEY(@L,@D)
P-Date		DATE
P-Datetime		DATETIME
P-Decimal	D ø	DECIMAL(@L)
	D not ø	DECIMAL(@L,@D)
P-Double		FLOAT
P-Float	L=0 or L ø	FLOAT
	L<>0	SMALLFLOAT(@L)
P-Integer		INTEGER
P-Long Integer		INTEGER
P-Long Real		FLOAT
P-Multimedia		BYTE
P-Numeric	L>9	DECIMAL(@L)
	4<L<10	INTEGER
	L<5 or L ø	SMALLINT
P-Real		FLOAT
P-Smallint		SMALLINT
P-String		VARCHAR(@L)
P-Text		VARCHAR(@L)

Pivot --> Datatype (Informix Dynamic Server 7.3)

Pivot	Condition	Datatype
P-Time		DATETIME
P-Timestamp		SERIAL
P-Tinyint		SMALLINT
P-Varchar	D ø	VARCHAR(@L)
	D not ø	VARCHAR(@L,@D)

Datatype --> Pivot (Informix Dynamic Server 7.3)

Datatype	Condition	Pivot
BYTE		P-Multimedia
CHAR(L)		P-Character
DATE		P-Date
DATETIME		P-Datetime
DECIMAL(L)		P-Decimal
DECIMAL(L,D)		P-Decimal
FLOAT		P-Float
INTEGER		P-Integer
MONEY(L,D)		P-Currency
SERIAL		P-Timestamp
SMALLFLOAT(L)		P-Float
SMALLINT		P-Boolean
VARCHAR(L)		P-Varchar
VARCHAR(L,D)		P-Varchar

INGRES II 2.0

Pivot --> Datatype (Ingres II 2.0)

Pivot	Condition	Datatype
P-AutoIdentifier		integer
P-Binary		long byte(@L)
P-Boolean		integer1
P-Byte		byte(@L)
P-Character		char(@L)
P-Currency		money(@L,@D)
P-Date		date
P-Datetime		date
P-Decimal		decimal(@L,@D)
P-Double		integer1
P-Float		float4
P-Integer		integer
P-Long Integer		integer
P-Long Real		float
P-Multimedia	L<2001 or L ø	byte(@L)
	L>2000	long byte(@L)
P-Numeric	L>9	float
	4<L<10	integer
	L<3 or L ø	integer1
	2<L<5	smallint
P-Real		float
P-Smallint		smallint
P-String		long varchar(@L)
P-Text		text(@L)

Pivot --> Datatype (Ingres II 2.0)

Pivot	Condition	Datatype
P-Time		date
P-Timestamp		date
P-Tinyint		integer1
P-Varbinary		byte varying(@L)
P-Varchar		varchar(@L)

Datatype --> Pivot (Ingres II 2.0)

Datatype	Condition	Pivot
byte varying(L)		P-Varbinary
byte(L)		P-Multimedia
char(L)		P-Character
date		P-Datetime
decimal(L,D)		P-Decimal
float		P-Real
float4		P-Float
integer		P-Integer
integer1		P-Tinyint
long byte(L)		P-Binary
long varchar(L)		P-String
money(L,D)		P-Currency
smallint		P-Smallint
text(L)		P-Text
varchar(L)		P-Varchar

MySQL 4.1

Pivot --> Datatype (MySQL 4.1)

Pivot	Condition	Datatype
	L>0 and D not ø	REAL (@L,@D) UNSIGNED
	L=0 or L ø	REAL UNSIGNED
	L>0 and D not ø	REAL (@L,@D) UNSIGNED ZEROFILL
	L=0 or L ø	REAL UNSIGNED ZEROFILL
P-AutoIdentifier		INTEGER
P-Binary	L ø or L<=0	BINARY
	L is numeric and L<>0	BINARY (@L)
P-Boolean		BOOLEAN
P-Byte	L ø or L<=0	BIT
	L is numeric and L<>0	BIT (@L)
P-Character	Not Unicode and L=0	CHAR
	Not Unicode and L<>0	CHAR (@L)
	Unicode and L<>0	CHAR (@L) UNICODE
	Unicode and L=0	CHAR UNICODE
P-Character Ascii	L ø or L<=0	CHAR ASCII
	L is numeric and L<>0	CHAR(@L) ASCII
P-Character Binary	Not Unicode and L<>0	CHAR (@L) BINARY
	Unicode and L<>0	CHAR (@L) UNICODE BINARY
	Not Unicode and L=0	CHAR BINARY
	Unicode and L=0	CHAR UNICODE BINARY

Pivot --> Datatype (MySQL 4.1)

Pivot	Condition	Datatype
P-Character Unicode	L is numeric and L<>0	CHAR (@L) UNICODE
	L ø or L<=0	CHAR UNICODE
P-Character Unicode Binary	L is numeric and L<>0	CHAR (@L) UNICODE BINARY
	L ø or L<=0	CHAR UNICODE BINARY
P-Currency		DECIMAL(@L,@D)
P-Date		DATE
P-Datetime		DATETIME
P-Decimal	L=0 or L ø	DECIMAL
	L>0 and D ø	DECIMAL (@L)
	L>0 and D not ø	DECIMAL(@L,@D)
P-Decimal Unsigned	L>0 and D ø	DECIMAL (@L) UNSIGNED
	L>0 and D not ø	DECIMAL (@L,@D) UNSIGNED
	L=0 or L ø	DECIMAL UNSIGNED
P-Decimal Unsigned Zerofill	L>0 and D ø	DECIMAL (@L) UNSIGNED ZEROFILL
	L>0 and D not ø	DECIMAL (@L,@D) UNSIGNED ZEROFILL
	L=0 or L ø	DECIMAL UNSIGNED ZEROFILL
P-Double	L ø or L<=0	DOUBLE PRECISION
	L<>0 or D<>0	DOUBLE PRECISION (@L,@D)
P-Double Unsigned	L>0 and D not ø	DOUBLE PRECISION (@L,@D) UNSIGNED
	L=0 or L ø	DOUBLE PRECISION UNSIGNED

Pivot --> Datatype (MySQL 4.1)

Pivot	Condition	Datatype
P-Double Unsigned Zero-fill	L>0 and D not ø	DOUBLE PRECISION (@L,@D) UNSIGNED ZEROFILL
	L=0 or L ø	DOUBLE PRECISION UNSIGNED ZEROFILL
P-Float	L=0 or L ø	FLOAT
	L>0 and D ø	FLOAT (@L)
	L>0 and D not ø	FLOAT (@L,@D)
P-Float Unsigned	L>0 and D ø	FLOAT (@L) UNSIGNED
	L>0 and D not ø	FLOAT (@L,@D) UN-SIGNED
	L=0 or L ø	FLOAT UNSIGNED
P-Float Unsigned Zerofill	L>0 and D ø	FLOAT (@L) UNSIGNED ZEROFILL
	L>0 and D not ø	FLOAT (@L,@D) UN-SIGNED ZEROFILL
	L=0 or L ø	FLOAT UNSIGNED ZE-ROFILL
P-Integer	L ø or L<=0	INTEGER
	L is numeric and L<>0	INTEGER (@L)
P-Integer Unsigned	L is numeric and L<>0	INTEGER (@L) UN-SIGNED
	L ø or L<=0	INTEGER UNSIGNED
P-Integer Unsigned Zerofill	L is numeric and L<>0	INTEGER (@L) UN-SIGNED ZEROFILL
	L ø or L<=0	INTEGER UNSIGNED ZEROFILL
P-Long Integer	L ø or L<=0	BIGINT
	L is numeric and L<>0	BIGINT (@L)

Pivot --> Datatype (MySQL 4.1)

Pivot	Condition	Datatype
P-Long Integer Unsigned	L is numeric and L<>0	BIGINT (@L) UNSIGNED
	L ø or L<=0	BIGINT UNSIGNED
P-Long Integer Unsigned Zerofill	L is numeric and L<>0	BIGINT (@L) UNSIGNED ZEROFILL
	L ø or L<=0	BIGINT UNSIGNED ZEROFILL
P-Longblob		LONGBLOB
P-Longtext		LONGTEXT
P-Mediumblob		MEDIUMBLOB
P-Mediumint	L ø or L<=0	MEDIUMINT
	L is numeric and L<>0	MEDIUMINT (@L)
P-Mediumint Unsigned	L is numeric and L<>0	MEDIUMINT (@L) UNSIGNED
	L ø or L<=0	MEDIUMINT UNSIGNED
P-Mediumint Unsigned Zerofill	L is numeric and L<>0	MEDIUMINT (@L) UNSIGNED ZEROFILL
	L ø or L<=0	MEDIUMINT UNSIGNED ZEROFILL
P-Mediumtext		MEDIUMTEXT
P-Multimedia		BLOB
P-National Varchar	L ø or L<0	NATIONAL VARCHAR
	L is numeric and L>=0	NATIONAL VARCHAR (@L)
P-National Varchar Binary	L is numeric and L>=0	NATIONAL VARCHAR (@L) BINARY
	L ø or L<0	NATIONAL VARCHAR BINARY

Pivot --> Datatype (MySQL 4.1)

Pivot	Condition	Datatype
P-Numeric	L=0 or L ø	NUMERIC
	L>0 and D ø	NUMERIC (@L)
	L>0 and D not ø	NUMERIC (@L,@D)
P-Real	L ø or L<=0	REAL
	L>0 and D not ø	REAL (@L,@D)
P-Smallint		SMALLINT
	L is numeric and L<>0	SMALLINT (@L)
P-Smallint Unsigned	L is numeric and L<>0	SMALLINT (@L) UNSIGNED
	L ø or L<=0	SMALLINT UNSIGNED
P-Smallint Unsigned Zerofill	L is numeric and L<>0	SMALLINT (@L) UNSIGNED ZEROFILL
	L ø or L<=0	SMALLINT UNSIGNED ZEROFILL
P-String		VARCHAR(@L)
P-Text		TEXT
P-Time		TIME
P-Timestamp		TIMESTAMP
P-Tinyblob		TINYBLOB
P-Tinyint	L ø or L<=0	TINYINT
	L is numeric and L<>0	TINYINT (@L)
P-Tinyint Unsigned	L is numeric and L<>0	TINYINT (@L) UNSIGNED
	L ø or L<=0	TINYINT UNSIGNED
P-Tinyint Unsigned Zerofill	L is numeric and L<>0	TINYINT (@L) UNSIGNED ZEROFILL
	L ø or L<=0	TINYINT UNSIGNED ZEROFILL
P-Tinytext		TINYTEXT

Pivot --> Datatype (MySQL 4.1)

Pivot	Condition	Datatype
P-Varbinary	L ø or L<0	VARBINARY
	L is numeric and L>=0	VARBINARY (@L)
P-Varchar	L ø or L<0	VARCHAR
	L is numeric and L>=0	VARCHAR(@L)
P-Varchar Binary	L is numeric and L>=0	VARCHAR (@L) BINARY
	L ø or L<0	VARCHAR BINARY
P-Wide Character	L ø or L<=0	NATIONAL CHAR
	L is numeric and L<>0	NATIONAL CHAR (@L)
P-Wide Character Binary	L is numeric and L<>0	NATIONAL CHAR (@L) BINARY
	L ø or L<=0	NATIONAL CHAR BINARY
P-Year	L ø or L<=0	YEAR
	L is numeric and L<>0	YEAR (@L)

Datatype --> Pivot (MySQL 4.1)

Datatype	Condition	Pivot
BIGINT		P-Long Integer
BIGINT (L)		P-Long Integer
BIGINT (L) UNSIGNED		P-Long Integer Unsigned
BIGINT (L) UNSIGNED ZEROFILL		P-Long Integer Unsigned Zerofill
BIGINT UNSIGNED		P-Long Integer Unsigned
BIGINT UNSIGNED ZEROFILL		P-Long Integer Unsigned Zerofill
BINARY		P-Binary
BINARY (L)		P-Binary
BLOB		P-Multimedia

Datatype --> Pivot (MySQL 4.1)

Datatype	Condition	Pivot
BOOLEAN		P-Boolean
CHAR (L)		P-Character
CHAR (L) BINARY		P-Character Binary
CHAR (L) UNICODE		P-Character Unicode
CHAR (L) UNICODE BINARY		P-Character Unicode Binary
CHAR ASCII		P-Character Ascii
CHAR BINARY		P-Character Binary
CHAR UNICODE		P-Character Unicode
CHAR UNICODE BINARY		P-Character Unicode Binary
CHAR(L) ASCII		P-Character Ascii
DATE		P-Date
DATETIME		P-Datetime
DECIMAL (L)		P-Decimal
DECIMAL (L) UNSIGNED		P-Decimal Unsigned
DECIMAL (L) UNSIGNED ZEROFILL		P-Decimal Unsigned Zerofill
DECIMAL (L,D) UNSIGNED		P-Decimal Unsigned
DECIMAL (L,D) UNSIGNED ZEROFILL		P-Decimal Unsigned Zerofill
DECIMAL UNSIGNED		P-Decimal Unsigned
DECIMAL UNSIGNED ZEROFILL		P-Decimal Unsigned Zerofill
DOUBLE PRECISION		P-Double
DOUBLE PRECISION (L,D)		P-Double

Datatype --> Pivot (MySQL 4.1)

Datatype	Condition	Pivot
DOUBLE PRECISION (L,D) UNSIGNED		P-Double Unsigned
DOUBLE PRECISION (L,D) UNSIGNED ZERO-FILL		P-Double Unsigned Zero-fill
DOUBLE PRECISION UNSIGNED		P-Double Unsigned
DOUBLE PRECISION UNSIGNED ZEROFILL		P-Double Unsigned Zero-fill
FLOAT		P-Float
FLOAT (L)		P-Float
FLOAT (L) UNSIGNED		P-Float Unsigned
FLOAT (L) UNSIGNED ZEROFILL		P-Float Unsigned Zerofill
FLOAT (L,D)		P-Float
FLOAT (L,D) UNSIGNED		P-Float Unsigned
FLOAT (L,D) UNSIGNED ZEROFILL		P-Float Unsigned Zerofill
FLOAT UNSIGNED		P-Float Unsigned
FLOAT UNSIGNED ZE-ROFILL		P-Float Unsigned Zerofill
INTEGER		P-Integer
INTEGER (L)		P-Integer
INTEGER (L) UN-SIGNED		P-Integer Unsigned
INTEGER (L) UN-SIGNED ZEROFILL		P-Integer Unsigned Zerofill
INTEGER UNSIGNED		P-Integer Unsigned
INTEGER UNSIGNED ZEROFILL		P-Integer Unsigned Zerofill
LONGBLOB		P-Longblob

Datatype --> Pivot (MySQL 4.1)

Datatype	Condition	Pivot
LONGTEXT		P-Longtext
MEDIUMBLOB		P-Mediumblob
MEDIUMINT		P-Mediumint
MEDIUMINT (L)		P-Mediumint
MEDIUMINT (L) UNSIGNED		P-Mediumint Unsigned
MEDIUMINT (L) UNSIGNED ZEROFILL		P-Mediumint Unsigned Zerofill
MEDIUMINT UNSIGNED		P-Mediumint Unsigned
MEDIUMINT UNSIGNED ZEROFILL		P-Mediumint Unsigned Zerofill
MEDIUMTEXT		P-Mediumtext
NATIONAL CHAR		P-Wide Character
NATIONAL CHAR (L)		P-Wide Character
NATIONAL CHAR (L) BINARY		P-Wide Character Binary
NATIONAL CHAR BINARY		P-Wide Character Binary
NATIONAL VARCHAR		P-National Varchar
NATIONAL VARCHAR (L)		P-National Varchar
NATIONAL VARCHAR (L) BINARY		P-National Varchar Binary
NATIONAL VARCHAR BINARY		P-National Varchar Binary
NUMERIC		P-Numeric
NUMERIC (L)		P-Numeric
NUMERIC (L,D)		P-Numeric
REAL		P-Real

Datatype --> Pivot (MySQL 4.1)

Datatype	Condition	Pivot
REAL (L,D)		P-Real
REAL (L,D) UNSIGNED		
REAL (L,D) UNSIGNED ZEROFILL		
REAL UNSIGNED		
REAL UNSIGNED ZEROFILL		
SMALLINT		P-Smallint
SMALLINT (L)		P-Smallint
SMALLINT (L) UNSIGNED		P-Smallint Unsigned
SMALLINT (L) UNSIGNED ZEROFILL		P-Smallint Unsigned Zerofill
SMALLINT UNSIGNED		P-Smallint Unsigned
SMALLINT UNSIGNED ZEROFILL		P-Smallint Unsigned Zerofill
TEXT		P-Text
TIME		P-Time
TIMESTAMP		P-Timestamp
TINYBLOB		P-Tinyblob
TINYINT		P-Tinyint
TINYINT (L)		P-Tinyint
TINYINT (L) UNSIGNED		P-Tinyint Unsigned
TINYINT (L) UNSIGNED ZEROFILL		P-Tinyint Unsigned Zerofill
TINYINT UNSIGNED		P-Tinyint Unsigned
TINYINT UNSIGNED ZEROFILL		P-Tinyint Unsigned Zerofill
TINYTEXT		P-Tinytext
VARBINARY		P-Varbinary

Datatype --> Pivot (MySQL 4.1)

Datatype	Condition	Pivot
VARBINARY (L)		P-Varbinary
VARCHAR		P-Varchar
VARCHAR (L) BINARY		P-Varchar Binary
VARCHAR BINARY		P-Varchar Binary
VARCHAR(L)		P-Varchar
YEAR		P-Year
YEAR (L)		P-Year

MySQL 5.0

Pivot --> Datatype (MySQL 5.0)

Pivot	Condition	Datatype
	L>0 and D not ø	REAL (@L,@D) UNSIGNED
	L=0 or L ø	REAL UNSIGNED
	L>0 and D not ø	REAL (@L,@D) UNSIGNED ZEROFILL
	L=0 or L ø	REAL UNSIGNED ZEROFILL
P-AutoIdentifier		INTEGER
P-Binary	L ø or L<=0	BINARY
	L is numeric and L<>0	BINARY (@L)
P-Boolean		BOOLEAN
P-Byte	L ø or L<=0	BIT
	L is numeric and L<>0	BIT (@L)
P-Character	Not Unicode and L=0	CHAR
	Unicode and L<>0	CHAR (@L) UNICODE
	Unicode and L=0	CHAR UNICODE
	Not Unicode and L<>0	CHAR(@L)
P-Character Ascii	L ø or L<=0	CHAR ASCII
	L is numeric and L<>0	CHAR(@L) ASCII
P-Character Binary	Not Unicode and L<>0	CHAR (@L) BINARY
	Unicode and L<>0	CHAR (@L) UNICODE BINARY
	Not Unicode and L=0	CHAR BINARY
	Unicode and L=0	CHAR UNICODE BINARY

Pivot --> Datatype (MySQL 5.0)

Pivot	Condition	Datatype
P-Character Unicode	L is numeric and L<>0	CHAR (@L) UNICODE
	L ø or L<=0	CHAR UNICODE
P-Character Unicode Binary	L is numeric and L<>0	CHAR (@L) UNICODE BINARY
	L ø or L<=0	CHAR UNICODE BINARY
P-Currency		DECIMAL(@L,@D)
P-Date		DATE
P-Datetime		DATETIME
P-Decimal	L=0 or L ø	DECIMAL
	L>0 and D ø	DECIMAL (@L)
	L>0 and D not ø	DECIMAL(@L,@D)
P-Decimal Unsigned	L>0 and D ø	DECIMAL (@L) UNSIGNED
	L>0 and D not ø	DECIMAL (@L,@D) UNSIGNED
	L=0 or L ø	DECIMAL UNSIGNED
P-Decimal Unsigned Zerofill	L>0 and D ø	DECIMAL (@L) UNSIGNED ZEROFILL
	L>0 and D not ø	DECIMAL (@L,@D) UNSIGNED ZEROFILL
	L=0 or L ø	DECIMAL UNSIGNED ZEROFILL
P-Double	L=0 or L ø	DOUBLE PRECISION
	L>0 and D not ø	DOUBLE PRECISION (@L,@D)
P-Double Unsigned	L>0 and D not ø	DOUBLE PRECISION (@L,@D) UNSIGNED
	L=0 or L ø	DOUBLE PRECISION UNSIGNED

Pivot --> Datatype (MySQL 5.0)

Pivot	Condition	Datatype
P-Double Unsigned Zero-fill	L>0 and D not ø	DOUBLE PRECISION (@L,@D) UNSIGNED ZEROFILL
	L=0 or L ø	DOUBLE PRECISION UNSIGNED ZEROFILL
P-Float	L=0 or L ø	FLOAT
	L>0 and D ø	FLOAT (@L)
	L>0 and D not ø	FLOAT (@L,@D)
P-Float Unsigned	L>0 and D ø	FLOAT (@L) UNSIGNED
	L>0 and D not ø	FLOAT (@L,@D) UN-SIGNED
	L=0 or L ø	FLOAT UNSIGNED
P-Float Unsigned Zerofill	L>0 and D ø	FLOAT (@L) UNSIGNED ZEROFILL
	L>0 and D not ø	FLOAT (@L,@D) UN-SIGNED ZEROFILL
	L=0 or L ø	FLOAT UNSIGNED ZE-ROFILL
P-Integer	L ø or L<=0	INTEGER
	L is numeric and L<>0	INTEGER (@L)
P-Integer Unsigned	L is numeric and L<>0	INTEGER (@L) UN-SIGNED
	L ø or L<=0	INTEGER UNSIGNED
P-Integer Unsigned Zerofill	L is numeric and L<>0	INTEGER (@L) UN-SIGNED ZEROFILL
	L ø or L<=0	INTEGER UNSIGNED ZEROFILL
P-Long Integer	L ø or L<=0	BIGINT
	L is numeric and L<>0	BIGINT (@L)

Pivot --> Datatype (MySQL 5.0)

Pivot	Condition	Datatype
P-Long Integer Unsigned	L is numeric and L<>0	BIGINT (@L) UNSIGNED
	L ø or L<=0	BIGINT UNSIGNED
P-Long Integer Unsigned Zerofill	L is numeric and L<>0	BIGINT (@L) UNSIGNED ZEROFILL
	L ø or L<=0	BIGINT UNSIGNED ZEROFILL
P-Longblob		LONGBLOB
P-Longtext		LONGTEXT
P-Mediumblob		MEDIUMBLOB
P-Mediumint	L ø or L<=0	MEDIUMINT
	L is numeric and L<>0	MEDIUMINT (@L)
P-Mediumint Unsigned	L is numeric and L<>0	MEDIUMINT (@L) UNSIGNED
	L ø or L<=0	MEDIUMINT UNSIGNED
P-Mediumint Unsigned Zerofill	L is numeric and L<>0	MEDIUMINT (@L) UNSIGNED ZEROFILL
	L ø or L<=0	MEDIUMINT UNSIGNED ZEROFILL
P-Mediumtext		MEDIUMTEXT
P-Multimedia		BLOB
P-National Varchar	L ø or L<0	NATIONAL VARCHAR
	L is numeric and L<>0	NATIONAL VARCHAR (@L)
P-National Varchar Binary	L is numeric and L<>0	NATIONAL VARCHAR (@L) BINARY
	L ø or L<0	NATIONAL VARCHAR BINARY

Pivot --> Datatype (MySQL 5.0)

Pivot	Condition	Datatype
P-Numeric	L=0 or L ø	NUMERIC
	L>0 and D ø	NUMERIC (@L)
	L>0 and D not ø	NUMERIC (@L,@D)
P-Real	L=0 or L ø	REAL
	L>0 and D not ø	REAL (@L,@D)
P-Smallint	L ø or L<=0	SMALLINT
	L is numeric and L<>0	SMALLINT (@L)
P-Smallint Unsigned	L is numeric and L<>0	SMALLINT (@L) UNSIGNED
	L ø or L<=0	SMALLINT UNSIGNED
P-Smallint Unsigned Zerofill	L is numeric and L<>0	SMALLINT (@L) UNSIGNED ZEROFILL
	L ø or L<=0	SMALLINT UNSIGNED ZEROFILL
P-String		VARCHAR(@L)
P-Text		TEXT
P-Time		TIME
P-Timestamp		TIMESTAMP
P-Tinyblob		TINYBLOB
P-Tinyint	L ø or L<=0	TINYINT
	L is numeric and L<>0	TINYINT (@L)
P-Tinyint Unsigned	L is numeric and L<>0	TINYINT (@L) UNSIGNED
	L ø or L<=0	TINYINT UNSIGNED
P-Tinyint Unsigned Zerofill	L is numeric and L<>0	TINYINT (@L) UNSIGNED ZEROFILL
	L ø or L<=0	TINYINT UNSIGNED ZEROFILL
P-Tinytext		TINYTEXT

Pivot --> Datatype (MySQL 5.0)

Pivot	Condition	Datatype
P-Varbinary	L ø or L<0	VARBINARY
	L is numeric and L>=0	VARBINARY (@L)
P-Varchar	L ø or L<=0	VARCHAR
	L is numeric and L=0	VARCHAR(@L)
P-Varchar Binary	L is numeric and L<>0	VARCHAR (@L) BINARY
	L ø or L<0	VARCHAR BINARY
P-Wide Character	L ø or L<=0	NATIONAL CHAR
	L is numeric and L<>0	NATIONAL CHAR (@L)
P-Wide Character Binary	L is numeric and L<>0	NATIONAL CHAR (@L) BINARY
	L ø or L<=0	NATIONAL CHAR BINARY
P-Year	L ø or L<=0	YEAR
	L is numeric and L<>0	YEAR(@L)

Datatype --> Pivot (MySQL 5.0)

Datatype	Condition	Pivot
BIGINT		P-Long Integer
BIGINT (L)		P-Long Integer
BIGINT (L) UNSIGNED		P-Long Integer Unsigned
BIGINT (L) UNSIGNED ZEROFILL		P-Long Integer Unsigned Zerofill
BIGINT UNSIGNED		P-Long Integer Unsigned
BIGINT UNSIGNED ZEROFILL		P-Long Integer Unsigned Zerofill
BINARY		P-Binary
BINARY (L)		P-Binary
BIT		P-Byte

Datatype --> Pivot (MySQL 5.0)

Datatype	Condition	Pivot
BIT (L)		P-Byte
BLOB		P-Multimedia
BOOLEAN		P-Boolean
CHAR		P-Character
CHAR (L) BINARY		P-Character Binary
CHAR (L) UNICODE		P-Character Unicode
CHAR (L) UNICODE BINARY		P-Character Unicode Binary
CHAR ASCII		P-Character Ascii
CHAR BINARY		P-Character Binary
CHAR UNICODE		P-Character Unicode
CHAR UNICODE BINARY		P-Character Unicode Binary
CHAR(L)		P-Character
CHAR(L) ASCII		P-Character Ascii
DATE		P-Date
DATETIME		P-Datetime
DECIMAL		P-Decimal
DECIMAL (L)		P-Decimal
DECIMAL (L) UNSIGNED		P-Decimal Unsigned
DECIMAL (L) UNSIGNED ZEROFILL		P-Decimal Unsigned Zerofill
DECIMAL (L,D) UNSIGNED		P-Decimal Unsigned
DECIMAL (L,D) UNSIGNED ZEROFILL		P-Decimal Unsigned Zerofill
DECIMAL UNSIGNED		P-Decimal Unsigned
DECIMAL UNSIGNED ZEROFILL		P-Decimal Unsigned Zerofill

Datatype --> Pivot (MySQL 5.0)

Datatype	Condition	Pivot
DECIMAL(L,D)		P-Decimal
DOUBLE PRECISION		P-Double
DOUBLE PRECISION (L,D)		P-Double
DOUBLE PRECISION (L,D) UNSIGNED		P-Double Unsigned
DOUBLE PRECISION (L,D) UNSIGNED ZERO-FILL		P-Double Unsigned Zero-fill
DOUBLE PRECISION UNSIGNED		P-Double Unsigned
DOUBLE PRECISION UNSIGNED ZEROFILL		P-Double Unsigned Zero-fill
FLOAT		P-Float
FLOAT (L)		P-Float
FLOAT (L) UNSIGNED		P-Float Unsigned
FLOAT (L) UNSIGNED ZEROFILL		P-Float Unsigned Zerofill
FLOAT (L,D)		P-Float
FLOAT (L,D) UNSIGNED		P-Float Unsigned
FLOAT (L,D) UNSIGNED ZEROFILL		P-Float Unsigned Zerofill
FLOAT UNSIGNED		P-Float Unsigned
FLOAT UNSIGNED ZE-ROFILL		P-Float Unsigned Zerofill
INTEGER		P-Integer
INTEGER (L)		P-Integer
INTEGER (L) UN-SIGNED		P-Integer Unsigned
INTEGER (L) UN-SIGNED ZEROFILL		P-Integer Unsigned Zerofill

Datatype --> Pivot (MySQL 5.0)

Datatype	Condition	Pivot
INTEGER UNSIGNED		P-Integer Unsigned
INTEGER UNSIGNED ZEROFILL		P-Integer Unsigned Zerofill
LONGBLOB		P-Longblob
LONGTEXT		P-Longtext
MEDIUMBLOB		P-Mediumblob
MEDIUMINT		P-Mediumint
MEDIUMINT (L)		P-Mediumint
MEDIUMINT (L) UNSIGNED		P-Mediumint Unsigned
MEDIUMINT (L) UNSIGNED ZEROFILL		P-Mediumint Unsigned Zerofill
MEDIUMINT UNSIGNED		P-Mediumint Unsigned
MEDIUMINT UNSIGNED ZEROFILL		P-Mediumint Unsigned Zerofill
MEDIUMTEXT		P-Mediumtext
NATIONAL CHAR		P-Wide Character
NATIONAL CHAR (L)		P-Wide Character
NATIONAL CHAR (L) BINARY		P-Wide Character Binary
NATIONAL CHAR BINARY		P-Wide Character Binary
NATIONAL VARCHAR		P-National Varchar
NATIONAL VARCHAR (L)		P-National Varchar
NATIONAL VARCHAR (L) BINARY		P-National Varchar Binary
NATIONAL VARCHAR BINARY		P-National Varchar Binary
NUMERIC		P-Numeric

Datatype --> Pivot (MySQL 5.0)

Datatype	Condition	Pivot
NUMERIC (L)		P-Numeric
NUMERIC (L,D)		P-Numeric
REAL		P-Real
REAL (L,D)		P-Real
REAL (L,D) UNSIGNED		
REAL (L,D) UNSIGNED ZEROFILL		
REAL UNSIGNED		
REAL UNSIGNED ZEROFILL		
SMALLINT		P-Smallint
SMALLINT (L)		P-Smallint
SMALLINT (L) UNSIGNED		P-Smallint Unsigned
SMALLINT (L) UNSIGNED ZEROFILL		P-Smallint Unsigned Zerofill
SMALLINT UNSIGNED		P-Smallint Unsigned
SMALLINT UNSIGNED ZEROFILL		P-Smallint Unsigned Zerofill
TEXT		P-Text
TIME		P-Time
TIMESTAMP		P-Timestamp
TINYBLOB		P-Tinyblob
TINYINT		P-Tinyint
TINYINT (L)		P-Tinyint
TINYINT (L) UNSIGNED		P-Tinyint Unsigned
TINYINT (L) UNSIGNED ZEROFILL		P-Tinyint Unsigned Zerofill
TINYINT UNSIGNED		P-Tinyint Unsigned

Datatype --> Pivot (MySQL 5.0)

Datatype	Condition	Pivot
TINYINT UNSIGNED ZEROFILL		P-Tinyint Unsigned Zero-fill
TINYTEXT		P-Tinytext
VARBINARY		P-Varbinary
VARBINARY (L)		P-Varbinary
VARCHAR		P-Varchar
VARCHAR (L) BINARY		P-Varchar Binary
VARCHAR BINARY		P-Varchar Binary
VARCHAR(L)		P-Varchar
YEAR		P-Year
YEAR(L)		P-Year

ORACLE 10

Pivot --> Datatype (Oracle 10)

Pivot	Condition	Datatype
P-AutoIdentifier		NUMBER
P-Binary		RAW(@L)
P-Boolean	L<2 or L ø	RAW(1)
	L>1	RAW(@L)
P-Byte		RAW(1)
P-Character	Not Unicode and (L<2001 or L ø)	CHAR(@L)
	L>4000	LONG
	Unicode and (L<2001 or L ø)	NCHAR(@L)
	Unicode and 2000<L<4001	NVARCHAR2(@L)
	Not Unicode and 2000<L<4001	VARCHAR2(@L)
P-Currency		NUMBER(@L,@D)
P-Date		DATE
P-Datetime		DATE
P-Decimal		NUMBER(@L,@D)
P-Double		NUMBER(@L,@D)
P-Float	L=0 or L>126 or L ø	FLOAT
	0<L<127	FLOAT(@L)
P-Integer		NUMBER(@L)
P-Long Integer		NUMBER(@L)
P-Long Real		NUMBER(@L,@D)
P-Multimedia		LONG RAW

Pivot --> Datatype (Oracle 10)

Pivot	Condition	Datatype
P-Numeric	L=0 or L ø	NUMBER
	L>0 and D ø	NUMBER(@L)
	L>0 and D not ø	NUMBER(@L,@D)
P-Real		NUMBER(@L,@D)
P-Smallint		NUMBER(@L)
P-String		LONG
P-Text	Unicode	NVARCHAR2(@L)
	Not Unicode	VARCHAR2(@L)
P-Time		DATE
P-Timestamp	L>9 or L ø	TIMESTAMP
	L<10	TIMESTAMP(@L)
P-Tinyint		NUMBER(@L)
P-Varbinary		LONG RAW
P-Varchar	L>4000 or L=0 or L ø	LONG
	Unicode and 0<L<4001	NVARCHAR2(@L)
	Not Unicode and 0<L<4001	VARCHAR2(@L)

Datatype --> Pivot (Oracle 10)

Datatype	Condition	Pivot
CHAR(L)		P-Character
DATE		P-Date
FLOAT		P-Float
FLOAT(L)		P-Float
LONG		P-String
LONG RAW		P-Multimedia
NUMBER		P-Numeric

Datatype --> Pivot (Oracle 10)

Datatype	Condition	Pivot
NUMBER(L)		P-Numeric
NUMBER(L,D)		P-Numeric
RAW(1)		P-Boolean
RAW(L)		P-Boolean
TIMESTAMP		P-Timestamp
TIMESTAMP(L)		P-Timestamp
VARCHAR2(L)		P-Varchar

ORACLE 11

Pivot --> Datatype (Oracle 11)

Pivot	Condition	Datatype
P-AutoIdentifier		NUMBER
P-Binary		RAW(@L)
P-Boolean	L<2 or L ø	RAW(1)
	L>1	RAW(@L)
P-Byte		RAW(1)
P-Character	Not Unicode and (L<2001 or L ø)	CHAR(@L)
	L>4000	LONG
	Unicode and (L<2001 or L ø)	NCHAR(@L)
	Unicode and 2000<L<4001	NVARCHAR2(@L)
	Not Unicode and 2000<L<4001	VARCHAR2(@L)
P-Currency		NUMBER(@L,@D)
P-Date		DATE
P-Datetime		DATE
P-Decimal		NUMBER(@L,@D)
P-Double		NUMBER(@L,@D)
P-Float	0<L<127	FLOAT(@L)
	L=0 or L>126 or L ø	FLOAT
P-Integer		NUMBER(@L)
P-Long Integer		NUMBER(@L)
P-Long Real		NUMBER(@L,@D)

Pivot --> Datatype (Oracle 11)

Pivot	Condition	Datatype
P-Numeric	L=0 or L ø	NUMBER
	L>0 and D ø	NUMBER(@L)
	L>0 and D not ø	NUMBER(@L,@D)
P-Real		NUMBER(@L,@D)
P-Smallint		NUMBER(@L)
P-String		LONG
P-Text	Unicode	NVARCHAR2(@L)
	Not Unicode	VARCHAR2(@L)
P-Time		DATE
P-Timestamp	L<10	TIMESTAMP(@L)
	L>9 or L ø	TIMESTAMP
P-Tinyint		NUMBER(@L)
P-Varchar	L>4000 or L=0 or L ø	LONG
	Unicode and 0<L<4001	NVARCHAR2(@L)
	Not Unicode and 0<L<4001	VARCHAR2(@L)

Datatype --> Pivot (Oracle 11)

Datatype	Condition	Pivot
CHAR(L)		P-Character
DATE		P-Date
FLOAT		P-Float
FLOAT(L)		P-Float
LONG		P-String
NUMBER		P-Numeric
NUMBER(L)		P-Numeric
NUMBER(L,D)		P-Numeric

Datatype --> Pivot (Oracle 11)

Datatype	Condition	Pivot
RAW(1)		P-Boolean
RAW(L)		P-Boolean
TIMESTAMP		P-Timestamp
TIMESTAMP(L)		P-Timestamp
VARCHAR2(L)		P-Varchar

ORACLE 8

Pivot --> Datatype (Oracle 8)

Pivot	Condition	Datatype
P-AutoIdentifier		NUMBER
P-Binary		RAW(@L)
P-Boolean	L<2 or L ø	RAW(1)
	L>1	RAW(@L)
P-Byte		RAW(1)
P-Character	L<2001 or L ø	CHAR(@L)
	L>4000	LONG
	2000<L<4001	VARCHAR2(@L)
P-Currency		NUMBER(@L,@D)
P-Date		DATE
P-Datetime		DATE
P-Decimal		NUMBER(@L,@D)
P-Double		NUMBER(@L,@D)
P-Float		NUMBER(@L,@D)
P-Integer		NUMBER(@L)
P-Long Integer		NUMBER(@L)
P-Long Real		NUMBER(@L,@D)
P-Multimedia		LONG RAW
P-Numeric	L=0 or L ø	NUMBER
	L>0 et D ø	NUMBER(@L)
	L>0 and D not ø	NUMBER(@L,@D)
P-Real		NUMBER(@L,@D)
P-Smallint		NUMBER(@L)
P-String		LONG

Pivot --> Datatype (Oracle 8)

Pivot	Condition	Datatype
P-Text		VARCHAR2(@L)
P-Time		DATE
P-Timestamp		ROWID
P-Tinyint		NUMBER(@L)
P-Varbinary		LONG RAW
P-Varchar	L>4000 or L=0 or L ø	LONG
	0<L<4001	VARCHAR2(@L)

Datatype --> Pivot (Oracle 8)

Datatype	Condition	Pivot
CHAR(L)		P-Character
DATE		P-Date
LONG		P-String
LONG RAW		P-Multimedia
NUMBER		P-Numeric
NUMBER(L)		P-Numeric
NUMBER(L,D)		P-Numeric
RAW(1)		P-Boolean
RAW(L)		P-Boolean
ROWID		P-Timestamp
VARCHAR2(L)		P-Varchar

ORACLE 9I

Pivot --> Datatype (Oracle 9i)

Pivot	Condition	Datatype
P-AutoIdentifier		NUMBER
P-Binary		RAW(@L)
P-Boolean	L<2 or L ø	RAW(1)
	L>1	RAW(@L)
P-Byte		RAW(1)
P-Character	Not Unicode and (L<2001 or L ø)	CHAR(@L)
	L>4000	LONG
	Unicode and (L<2001 or L ø)	NCHAR(@L)
	Unicode and 2000<L<4001	NVARCHAR2(@L)
	Not Unicode and 2000<L<4001	VARCHAR2(@L)
P-Currency		NUMBER(@L,@D)
P-Date		DATE
P-Datetime		DATE
P-Decimal		NUMBER(@L,@D)
P-Double		NUMBER(@L,@D)
P-Float	L=0 or L>126 or L ø	FLOAT
	0<L<127	FLOAT(@L)
P-Integer		NUMBER(@L)
P-Long Integer		NUMBER(@L)
P-Long Real		NUMBER(@L,@D)
P-Multimedia		LONG RAW
P-Numeric	L=0 or L ø	NUMBER
	L>0 and D ø	NUMBER(@L)
	L>0 and D not ø	NUMBER(@L,@D)

Pivot --> Datatype (Oracle 9i)

Pivot	Condition	Datatype
P-Real		NUMBER(@L,@D)
P-Smallint		NUMBER(@L)
P-String		LONG
P-Text	Unicode	NVARCHAR2(@L)
	Not Unicode	VARCHAR2(@L)
P-Time		DATE
P-Timestamp	L>9 or L ø	TIMESTAMP
	L<10	TIMESTAMP(@L)
P-Tinyint		NUMBER(@L)
P-Varbinary		LONG RAW
P-Varchar	L>4000 or L=0 or L ø	LONG
	Unicode and 0<L<4001	NVARCHAR2(@L)
	Not Unicode and 0<L<4001	VARCHAR2(@L)

Datatype --> Pivot (Oracle 9i)

Datatype	Condition	Pivot
CHAR(L)		P-Character
DATE		P-Date
FLOAT		P-Float
FLOAT(L)		P-Float
LONG		P-String
LONG RAW		P-Multimedia
NUMBER		P-Numeric
NUMBER(L)		P-Numeric
NUMBER(L,D)		P-Numeric
RAW(1)		P-Boolean
RAW(L)		P-Boolean

Datatype --> Pivot (Oracle 9i)

Datatype	Condition	Pivot
TIMESTAMP		P-Timestamp
VARCHAR2(L)		P-Varchar

POSTGRESQL9.3

Pivot --> Datatype (PostgreSQL9.3)

Pivot	Condition	Datatype
P-Boolean		boolean
P-Byte	L=0 or L ø	bit
	Valid	bit(@L)
P-Character	L=0 or L ø	char
	Valid	char(@L)
P-Currency		money
P-Date		date
P-Decimal	L=0 or L ø	decimal
	L>=1 and D ø	decimal(@L)
	L>=1	decimal(@L,@D)
P-Double		double precision
P-Integer		integer
P-Long Integer		bigint
P-Numeric	L=0 or L ø	numeric
	L>=1 and D ø	numeric(@L)
	L>=1	numeric(@L,@D)
P-Real		real
P-Smallint		smallint
P-Text		text
P-Time	L=0 or L ø	time
	Valid	time(@L)
P-Timestamp	L=0 or L ø	timestamp
	L <> 0	timestamp(@L)

Pivot --> Datatype (PostgreSQL9.3)

Pivot	Condition	Datatype
P-Varchar	L=0 or L ø	varchar
	Valid	varchar(@L)

Datatype --> Pivot (PostgreSQL9.3)

Datatype	Condition	Pivot
bigint		P-Long Integer
bit		P-Byte
bit(L)		P-Byte
boolean		P-Boolean
char		P-Character
char(L)		P-Character
date		P-Date
decimal		P-Decimal
decimal(L)		P-Decimal
decimal(L,D)		P-Decimal
double precision		P-Double
integer		P-Integer
money		P-Currency
numeric		P-Numeric
numeric(L)		P-Numeric
numeric(L,D)		P-Numeric
real		P-Real
smallint		P-Smallint
text		P-Text
time		P-Time
time(L)		P-Time
timestamp		P-Timestamp

Datatype --> Pivot (PostgreSQL9.3)

Datatype	Condition	Pivot
timestamp(L)		P-Timestamp
varchar		P-Varchar
varchar(L)		P-Varchar

SQL ANSI/ISO 9075:1992

Pivot --> Datatype (SQL ANSI/ISO 9075:1992)

Pivot	Condition	Datatype
P-AutoIdentifier		INTEGER
P-Binary		BIT VARYING(@L)
P-Boolean		BIT(@L)
P-Byte		BIT(@L)
P-Character		CHAR(@L)
P-Currency		DECIMAL(@L,@D)
P-Date		DATE
P-Datetime		DATETIME
P-Decimal		DECIMAL(@L,@D)
P-Double		DOUBLE PRECISION
P-Float		FLOAT
P-Integer		INTEGER
P-Long Integer		INTEGER
P-Long Real		REAL
P-Multimedia		BIT VARYING(@L)
P-Numeric	L>4	INTEGER
	L<5 or L ø	SMALLINT
P-Real		REAL
P-Smallint		SMALLINT
P-String		VARCHAR(@L)
P-Text		VARCHAR(@L)
P-Time		TIME
P-Timestamp		DATETIME
P-Tinyint		SMALLINT

Pivot --> Datatype (SQL ANSI/ISO 9075:1992)

Pivot	Condition	Datatype
P-Varbinary		BIT VARYING(@L)
P-Varchar		VARCHAR(@L)

Datatype --> Pivot (SQL ANSI/ISO 9075:1992)

Datatype	Condition	Pivot
BIT VARYING(L)		P-Multimedia
BIT(L)		P-Boolean
CHAR(L)		P-Character
DATE		P-Date
DATETIME		P-Datetime
DECIMAL(L,D)		P-Currency
DOUBLE PRECISION		P-Double
FLOAT		P-Float
INTEGER		P-Integer
REAL		P-Real
SMALLINT		P-Smallint
TIME		P-Time
VARCHAR(L)		P-Varchar

SQL SERVER 2000

Pivot --> Datatype (SQL Server 2000)

Pivot	Condition	Datatype
P-AutoIdentifier		uniqueidentifier
P-Binary		binary(@L)
P-Boolean		bit
P-Byte		bit
P-Character	Not Unicode and (L<8001 or L ø)	char(@L)
	Unicode and (L<8001 or L ø)	nchar(@L)
	Unicode and L>8000	ntext
	Not Unicode and L>8000	text
P-Currency		money
P-Date		smalldatetime
P-Datetime		datetime
P-Decimal		decimal(@L,@D)
P-Double		numeric(@L,@D)
P-Float		float
P-Integer		int
P-Long Integer		bigint
P-Long Real		real
P-Multimedia		image
P-Numeric		numeric(@L,@D)
P-Real		real
P-Smallint		smallint

Pivot --> Datatype (SQL Server 2000)

Pivot	Condition	Datatype
P-String	Unicode	ntext
	Not Unicode	text
P-Text	Unicode	ntext
	Not Unicode	text
P-Time		datetime
P-Timestamp		timestamp
P-Tinyint		tinyint
P-Varbinary		varbinary(@L)
P-Varchar	Unicode and L>8000	ntext
	Unicode and (L<8001 or L ø)	nvarchar(@L)
	Not Unicode and L>8000	text
	Not Unicode and (L<8001 or L ø)	varchar(@L)
P-Wide Character		nchar(@L)
P-Wide String		nvarchar(@L)

Datatype --> Pivot (SQL Server 2000)

Datatype	Condition	Pivot
bigint		P-Long Integer
binary(L)		P-Binary
bit		P-Boolean
char(L)		P-Character
datetime		P-Datetime
decimal(L,D)		P-Decimal
float		P-Float
image		P-Multimedia
int		P-Integer

Datatype --> Pivot (SQL Server 2000)

Datatype	Condition	Pivot
money		P-Currency
nchar(L)		P-Wide Character
numeric(L,D)		P-Numeric
nvarchar(L)		P-Wide String
real		P-Real
smalldatetime		P-Date
smallint		P-Smallint
text		P-Text
timestamp		P-Timestamp
tinyint		P-Tinyint
uniqueidentifier		P-AutoIdentifier
varbinary(L)		P-Varbinary
varchar(L)		P-Varchar

SQL SERVER 2005

Pivot --> Datatype (SQL Server 2005)

Pivot	Condition	Datatype
P-AutoIdentifier		uniqueidentifier
P-Binary		binary(@L)
P-Boolean		bit
P-Byte		bit
P-Character	Not Unicode and (L<8001 or L ø)	char(@L)
	Unicode and (L<8001 or L ø)	nchar(@L)
	Unicode and L>8000	ntext
	Not Unicode and L>8000	text
P-Currency		money
		smallmoney
P-Date		smalldatetime
P-Datetime		datetime
P-Decimal		decimal(@L,@D)
P-Double		numeric(@L,@D)
P-Float		float
P-Integer		int
P-Long Integer		bigint
P-Long Real		real
P-Multimedia		image
P-Numeric		numeric(@L,@D)
P-Real		real
P-Smallint		smallint

Pivot --> Datatype (SQL Server 2005)

Pivot	Condition	Datatype
P-String	Unicode	ntext
	Not Unicode	text
P-Text	Unicode	ntext
	Not Unicode	text
P-Time		datetime
P-Timestamp		timestamp
P-Tinyint		tinyint
P-Varbinary		varbinary(@L)
P-Varchar	Unicode and L>8000	ntext
	Unicode and (L<8001 or L ø)	nvarchar(@L)
	Not Unicode and L>8000	text
	Not Unicode and (L<8001 or L ø)	varchar(@L)
P-Wide Character		nchar(@L)
P-Wide String		nvarchar(@L)

Datatype --> Pivot (SQL Server 2005)

Datatype	Condition	Pivot
bigint		P-Long Integer
binary(L)		P-Binary
bit		P-Boolean
char(L)		P-Character
datetime		P-Datetime
decimal(L,D)		P-Decimal
float		P-Float
image		P-Multimedia
int		P-Integer

Datatype --> Pivot (SQL Server 2005)

Datatype	Condition	Pivot
money		P-Currency
nchar(L)		P-Wide Character
numeric(L,D)		P-Numeric
nvarchar(L)		P-Wide String
real		P-Real
smalldatetime		P-Date
smallint		P-Smallint
smallmoney		P-Currency
text		P-Text
timestamp		P-Timestamp
tinyint		P-Tinyint
uniqueidentifier		P-AutoIdentifier
varbinary(L)		P-Varbinary
varchar(L)		P-Varchar

SQL SERVER 2008

Pivot --> Datatype (SQL Server 2008)

Pivot	Condition	Datatype
P-AutoIdentifier		uniqueidentifier
P-Binary		binary(@L)
P-Boolean		bit
P-Byte		bit
P-Character	Not Unicode and (L<8001 or L ø)	char(@L)
	Unicode and (L<8001 or L ø)	nchar(@L)
	Unicode and L>8000	ntext
	Not Unicode and L>8000	text
P-Currency	L not empty or L > 10	money
	L empty or L < 11	smallmoney
P-Date		smalldatetime
P-Datetime		datetime
P-Decimal		decimal(@L,@D)
P-Double		numeric(@L,@D)
P-Float	L empty	float
	L not empty	float(@L)
P-Integer		int
P-Long Integer		bigint
P-Long Real		real
P-Multimedia		image
P-Numeric		numeric(@L,@D)
P-Real		real

Pivot --> Datatype (SQL Server 2008)

Pivot	Condition	Datatype
P-Smallint		smallint
P-String	Unicode	ntext
	Not Unicode	text
P-Text	Unicode	ntext
	Not Unicode	text
P-Time		time
P-Timestamp		timestamp
P-Tinyint		tinyint
P-Varbinary		varbinary(@L)
P-Varchar	Unicode and L>8000	ntext
	Unicode and (L<8001 or L ø)	nvarchar(@L)
	Not Unicode and L>8000	text
	Not Unicode and (L<8001 or L ø)	varchar(@L)
P-Wide Character		nchar(@L)
P-Wide String		nvarchar(@L)

Datatype --> Pivot (SQL Server 2008)

Datatype	Condition	Pivot
bigint		P-Long Integer
binary(L)		P-Binary
bit		P-Boolean
char(L)		P-Character
datetime		P-Datetime
decimal(L,D)		P-Decimal
float		P-Float
float(L)		P-Float

Datatype --> Pivot (SQL Server 2008)

Datatype	Condition	Pivot
image		P-Multimedia
int		P-Integer
money		P-Currency
nchar(L)		P-Wide Character
numeric(L,D)		P-Numeric
nvarchar(L)		P-Wide String
real		P-Real
smalldatetime		P-Date
smallint		P-Smallint
smallmoney		P-Currency
text		P-Text
time		P-Time
timestamp		P-Timestamp
tinyint		P-Tinyint
uniqueidentifier		P-AutoIdentifier
varbinary(L)		P-Varbinary
varchar(L)		P-Varchar

SQL SERVER 7

Pivot --> Datatype (SQL Server 7)

Pivot	Condition	Datatype
P-AutoIdentifier		timestamp
P-Binary		binary(@L)
P-Boolean		bit
P-Byte		bit
P-Character	0<L<251	char(@L)
	L>250 or L=0 or L ø	text
P-Currency		money
P-Date		smalldatetime
P-Datetime		datetime
P-Decimal		decimal(@L,@D)
P-Double		numeric(@L,@D)
P-Float		float
P-Integer		int
P-Long Integer		int
P-Long Real		real
P-Multimedia		image
P-Numeric	L>9 and D ø	float
	4<L<10 and D ø	int
	L not ø and D not ø	numeric(@L,@D)
	2<L<5 and D ø	smallint
	(L<3 and D ø) or L ø	tinyint
P-Real		real
P-Smallint		smallint
P-String		char(@L)

Pivot --> Datatype (SQL Server 7)

Pivot	Condition	Datatype
P-Text		text
P-Time		datetime
P-Timestamp		timestamp
P-Tinyint		tinyint
P-Varbinary		varbinary(@L)
P-Varchar		varchar(@L)

Datatype --> Pivot (SQL Server 7)

Datatype	Condition	Pivot
binary(L)		P-Binary
bit		P-Boolean
char(L)		P-String
datetime		P-Datetime
decimal(L,D)		P-Decimal
float		P-Float
image		P-Multimedia
int		P-Integer
money		P-Currency
numeric(L,D)		P-Double
real		P-Real
smalldatetime		P-Date
smallint		P-Smallint
text		P-Character
timestamp		P-Timestamp
tinyint		P-Tinyint
varbinary(L)		P-Varbinary
varchar(L)		P-Varchar

SYBASE ADAPTIVE SERVER 11

Pivot --> Datatype (Sybase Adaptive Server 11)

Pivot	Condition	Datatype
P-AutoIdentifier		timestamp
P-Binary		binary(@L)
P-Boolean		bit
P-Byte		bit
P-Character	Not Unicode and L<256	char(@L)
	Unicode and L<256	nChar(@L)
	L>255 or L ø	text
P-Currency		money
P-Date		smalldatetime
P-Datetime		datetime
P-Decimal	0<L<39 and D not ø	decimal(@L,@D)
	(L>9 and D ø) or (1<L<38 and D not ø)	float
	4<L<10 and D ø	int
	2<L<5 and D ø	smallint
	(L<3 and D ø) or L ø	tinyint
P-Double		double precision
P-Float		float
P-Integer		int
P-Long Integer		int
P-Long Real		real
P-Multimedia		image

Pivot --> Datatype (Sybase Adaptive Server 11)

Pivot	Condition	Datatype
P-Numeric	(L>9 and D ø) or (1<L<38 and D not ø)	float
	4<L<10 and D ø	int
	0<L<39 and D not ø	numeric(@L,@D)
	2<L<5 and D ø	smallint
	(L<3 and D ø) or L ø	tinyint
P-Real		real
P-Smallint		smallint
P-String	Not Unicode	char(@L)
	Unicode	nChar(@L)
P-Text		text
P-Time		datetime
P-Timestamp		timestamp
P-Tinyint		tinyint
P-Varbinary		varbinary(@L)
P-Varchar	Unicode	nVarChar(@L)
	Not Unicode	varchar(@L)

Datatype --> Pivot (Sybase Adaptive Server 11)

Datatype	Condition	Pivot
binary(L)		P-Binary
bit		P-Boolean
char(L)		P-String
datetime		P-Datetime
decimal(L,D)		P-Decimal
double precision		P-Double
float		P-Float

Datatype --> Pivot (Sybase Adaptive Server 11)

Datatype	Condition	Pivot
image		P-Multimedia
int		P-Integer
money		P-Currency
numeric(L,D)		P-Numeric
real		P-Real
smalldatetime		P-Date
smallint		P-Smallint
text		P-Character
timestamp		P-Timestamp
tinyint		P-Tinyint
varbinary(L)		P-Varbinary
varchar(L)		P-Varchar

SYBASE ADAPTIVE SERVER 12.5

Pivot --> Datatype (Sybase Adaptive Server 12.5)

Pivot	Condition	Datatype
P-AutoIdentifier		timestamp
P-Binary	L=0 or L ø	binary
	L > 0	binary(@L)
P-Boolean		bit
P-Byte		bit
P-Character	Not Unicode and (L=0 or L ø)	char
	Not Unicode and 0<L<256	char(@L)
	L>255	text
	Unicode and (L<256 or L ø)	unichar(@L)
P-Currency		money
P-Date		smalldatetime
P-Datetime		datetime
P-Decimal	0<L<39 and D not ø	decimal(@L,@D)
	(L>9 and D ø) or ((L<1 or L>38) and D not ø)	float
	4<L<10 and D ø	int
	2<L<5 and D ø	smallint
	L ø or (L<3 and D ø)	tinyint
P-Double		double precision
P-Float		float
P-Integer		int
P-Long Integer		int
P-Long Real		real

Pivot --> Datatype (Sybase Adaptive Server 12.5)

Pivot	Condition	Datatype
P-Multimedia		image
P-Numeric	(L>9 and D ø) or ((L<1 or L>38) and D not ø)	float
	4<L<10 and D ø	int
	0<L<39 and D not ø	numeric(@L,@D)
	2<L<5 and D ø	smallint
	L ø or (L<3 and D ø)	tinyint
P-Real		real
P-Smallint		smallint
P-String	Not Unicode and (L=0 or L ø)	char
	Not Unicode and (0<L<256)	char(@L)
	L>255	text
	Unicode and (L<256 or L ø)	unichar(@L)
P-Text		text
P-Time		datetime
P-Timestamp		timestamp
P-Tinyint		tinyint
P-Varbinary	L=0 or L ø	varbinary
	L > 0	varbinary(@L)
P-Varchar	L>255	text
	Unicode and (L<256 or L ø)	univarchar(@L)
	Not Unicode and (L=0 or L ø)	varchar
	Not Unicode and 0<L<256	varchar(@L)

Datatype --> Pivot (Sybase Adaptive Server 12.5)

Datatype	Condition	Pivot
binary		P-Binary
binary(L)		P-Binary
bit		P-Boolean
char		P-Character
char(L)		P-Character
datetime		P-Datetime
decimal(L,D)		P-Decimal
double precision		P-Double
float		P-Float
image		P-Multimedia
int		P-Integer
money		P-Currency
numeric(L,D)		P-Numeric
real		P-Real
smalldatetime		P-Date
smallint		P-Smallint
text		P-Text
timestamp		P-Timestamp
tinyint		P-Tinyint
varbinary		P-Varbinary
varbinary(L)		P-Varbinary
varchar		P-Varchar
varchar(L)		P-Varchar

TERADATA DATABASE 14

Pivot --> Datatype (Teradata Database 14)

Pivot	Condition	Datatype
P-AutoIdentifier		NUMBER
P-Boolean		BYTEINT
P-Byte	L=0 or L ø	BYTE
	L>0	BYTE(@L)
P-Character	L=0 or L ø	CHAR
	L>0	CHAR(@L)
P-Date		DATE
P-Datetime		DATE
P-Decimal	L=0 or L ø and D=0 Or D ø	DECIMAL
	L>0 and D=0 Or D ø	DECIMAL(@L)
P-Double		NUMBER(@L,@D)
P-Float		FLOAT
P-Integer		INTEGER
P-Long Integer		BIGINT
P-Long Real		FLOAT
P-Multimedia	L=0 or L ø	BLOB
	L>0	BLOB(@L)
P-Numeric	L=0 or L ø and D=0 Or D ø	NUMBER
	L ø and D > 0	NUMBER(*,@D)
	L>0 and D=0 Or D ø	NUMBER(@L)
	L>0 and D>0	NUMBER(@L,@D)
P-Real		FLOAT
P-Smallint		SMALLINT
P-String		VARCHAR(@L)

Pivot --> Datatype (Teradata Database 14)

Pivot	Condition	Datatype
P-Text		VARCHAR(@L)
P-Time		TIME
	D > 0	TIME(@D)
P-Timestamp		TIMESTAMP
	D > 0	TIMESTAMP(@D)
P-Tinyint		SMALLINT
P-Varbinary		VARBYTE(@L)
P-Varchar		VARCHAR(@L)

Datatype --> Pivot (Teradata Database 14)

Datatype	Condition	Pivot
BIGINT		P-Long Integer
BLOB		P-Multimedia
BLOB(L)		P-Multimedia
BYTE		P-Byte
BYTE(L)		P-Byte
BYTEINT		P-Boolean
CHAR		P-Character
CHAR(L)		P-Character
DATE		P-Date
DECIMAL		P-Decimal
DECIMAL(L)		P-Decimal
FLOAT		P-Real
INTEGER		P-Integer

Datatype --> Pivot (Teradata Database 14)

Datatype	Condition	Pivot
NUMBER		P-Numeric
NUMBER(*,D)		P-Numeric
NUMBER(L)		P-Numeric
NUMBER(L,D)		P-Numeric
SMALLINT		P-Smallint
TIME		P-Time
TIME(D)		P-Time
TIMESTAMP		P-Timestamp
TIMESTAMP(D)		P-Timestamp
VARBYTE(L)		P-Varbinary
VARCHAR(L)		P-Varchar

DB2 Version 10.5: Syntax supported by HOPEX

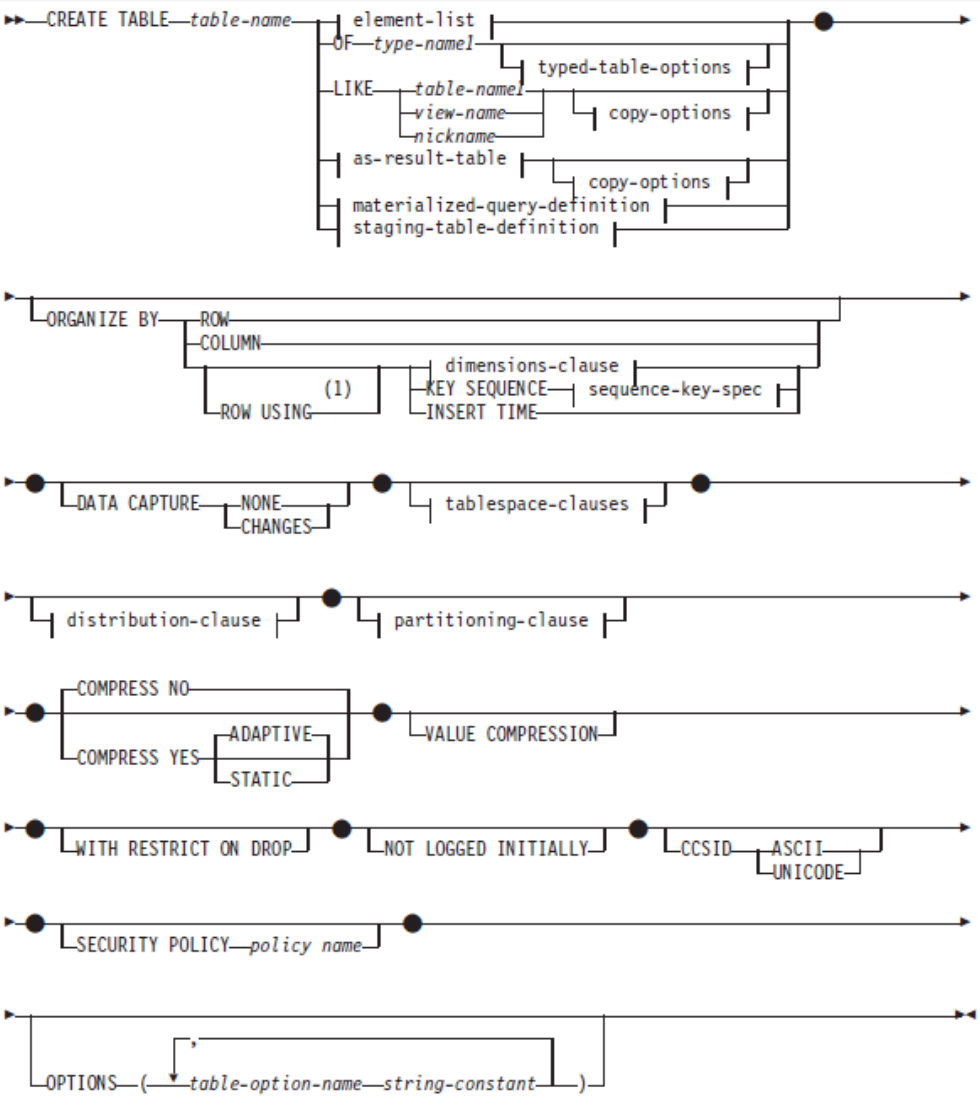
This document details concepts of DB2 V10.5 recognized by MEGA.

Detail of Concepts Managed by DB2 V10.5

All SQL commands are available at the following address:

http://public.dhe.ibm.com/ps/products/db2/info/vr105/pdf/en_US/DB2SQLRefVol1-db2s1e1051.pdf

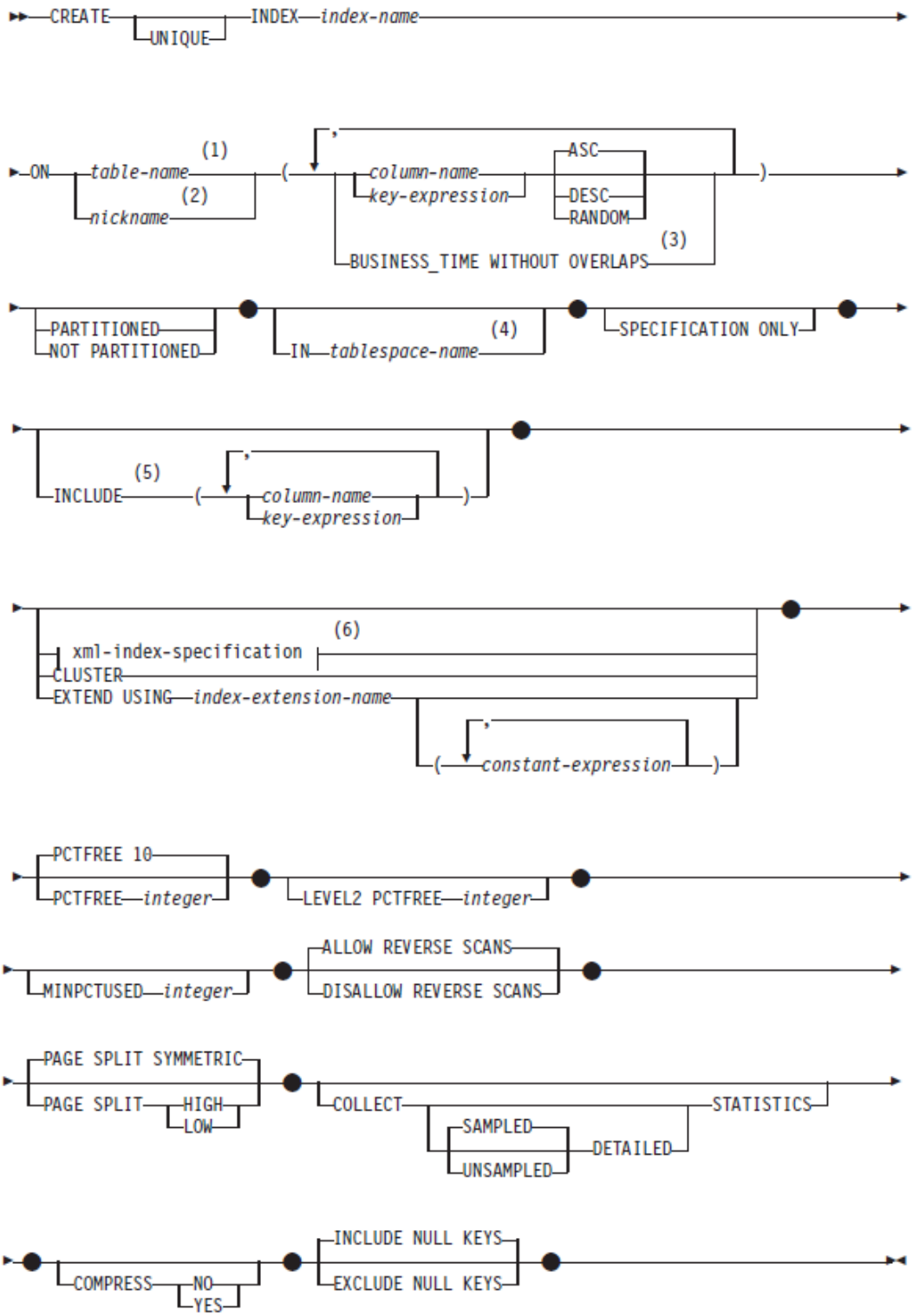
Tables



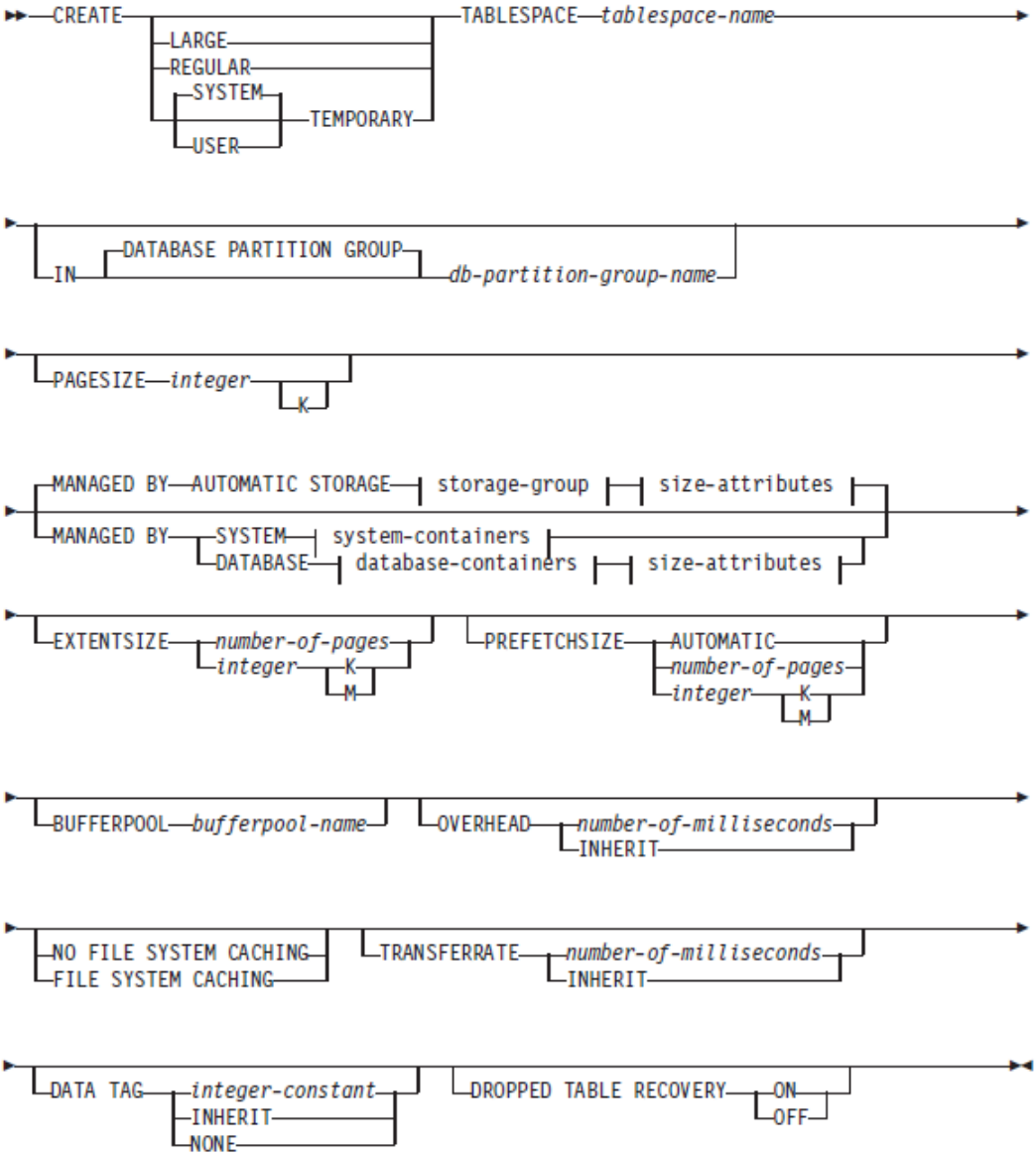
CREATE GLOBAL TEMPORARY TABLE

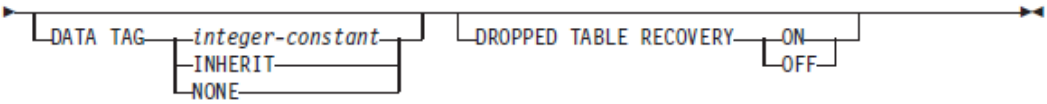
This new table type of DB2V10.5 is not supported by MEGA.

Indexes

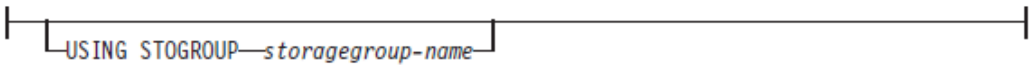

CREATE
UNIQUE
INDEX
index-name
ON
table-name
(1)
nickname
(2)
(
,
column-name
key-expression
ASC
DESC
RANDOM
BUSINESS_TIME WITHOUT OVERLAPS
(3)
)
PARTITIONED
NOT PARTITIONED
IN
tablespace-name
(4)
SPECIFICATION ONLY
INCLUDE
(5)
column-name
key-expression
xml-index-specification
(6)
CLUSTER
EXTEND USING
index-extension-name
constant-expression
PCTFREE 10
PCTFREE
integer
LEVEL2 PCTFREE
ALLOW REVERSE SCANS
DISALLOW REVERSE SCANS
MINPCTUSED
PAGE SPLIT SYMMETRIC
PAGE SPLIT
HIGH
LOW
COLLECT
STATISTICS
SAMPLED
UNSAMPLED
DETAILED
COMPRESS
NO
YES
INCLUDE NULL KEYS
EXCLUDE NULL KEYS

Tablespaces

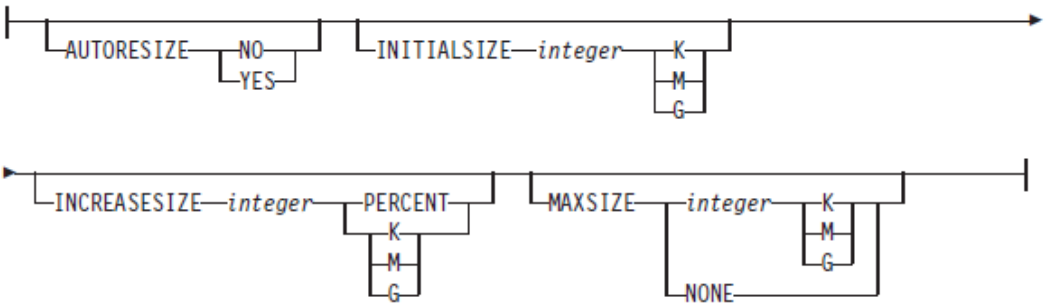




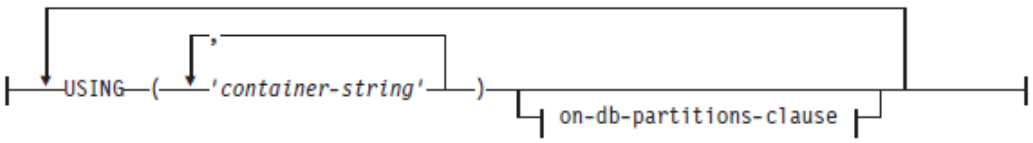
storage-group:



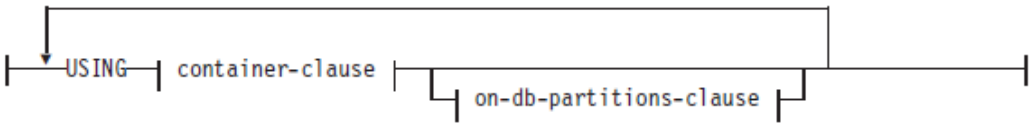
size-attributes:



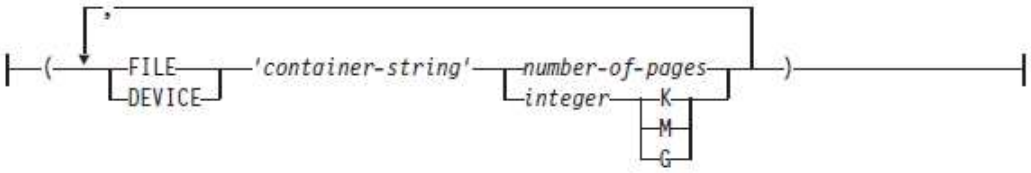
system-containers:



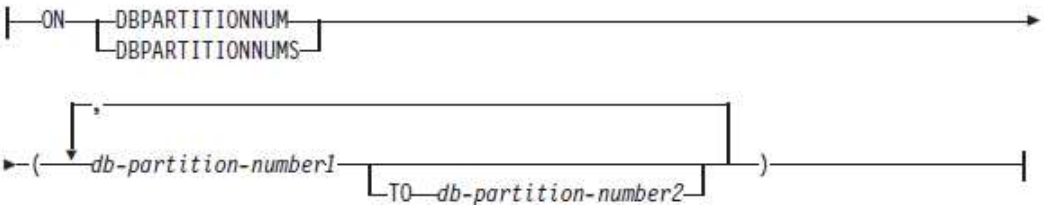
database-containers:



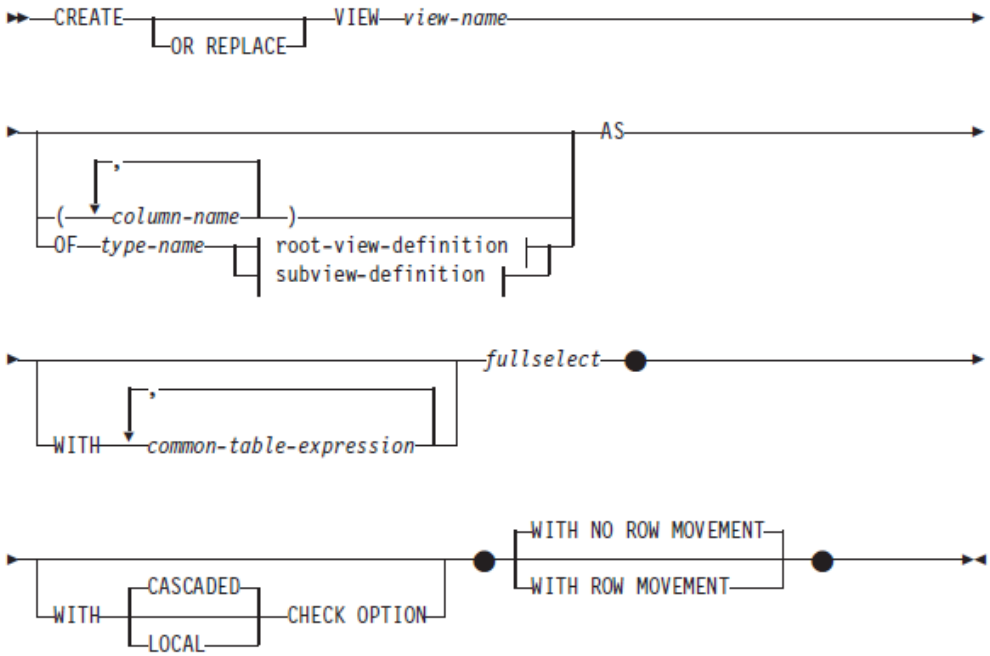
container-clause:



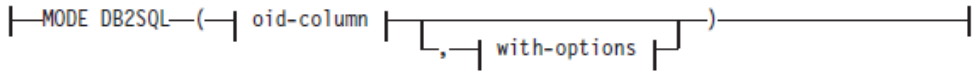
on-db-partitions-clause:



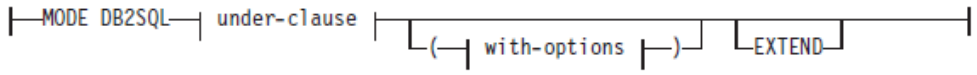
Views



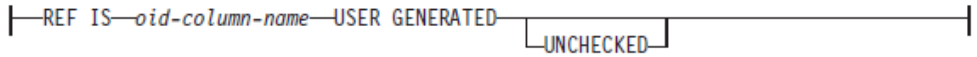
root-view-definition:



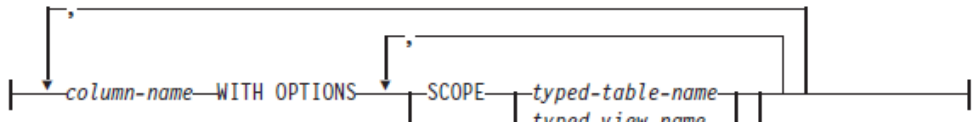
subview-definition:



oid-column:



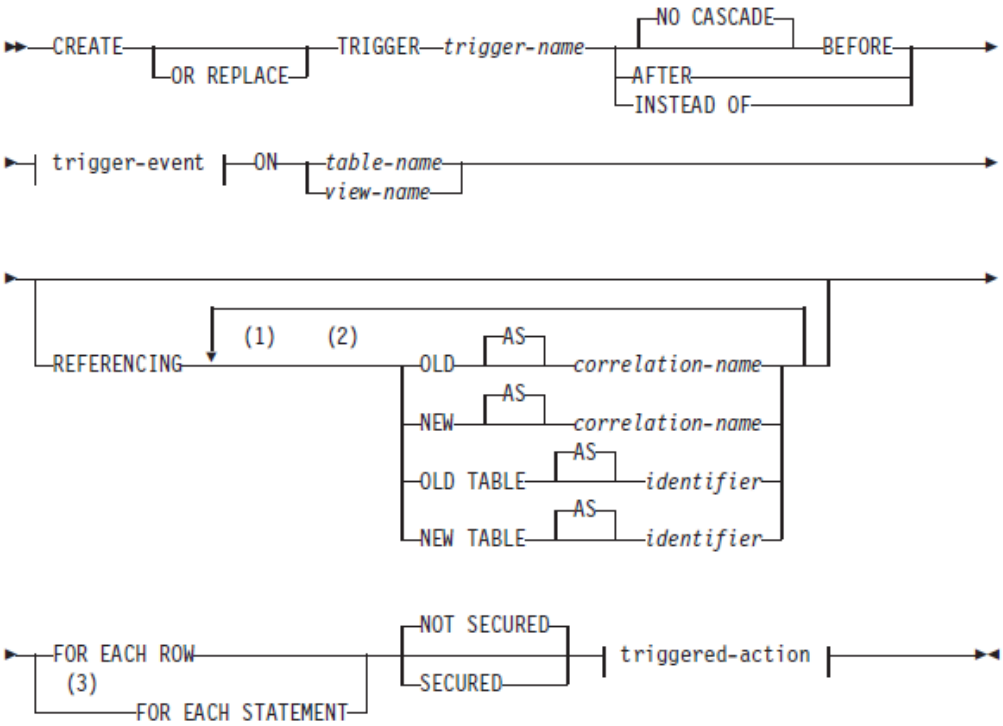
with-options:



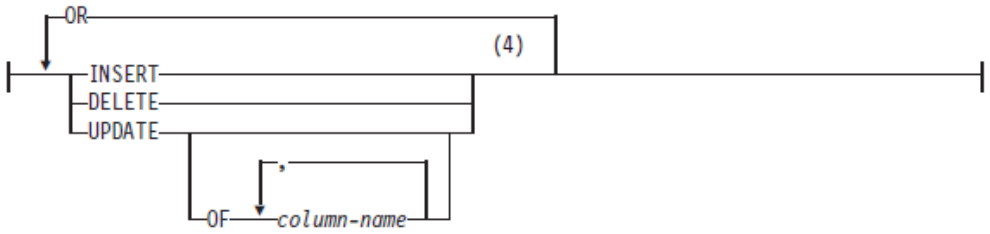
Example:

```
CREATE VIEW MA_PROJ
AS SELECTPROJNO, PROJNAME, RESPEMP
FROM PROJECT WHERE SUBSTR(PROJNO, 1, 2) = ' MA'
```

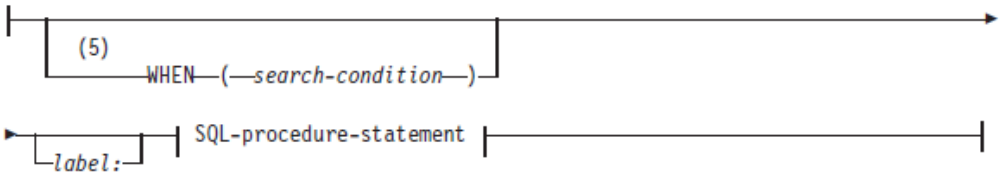
Triggers



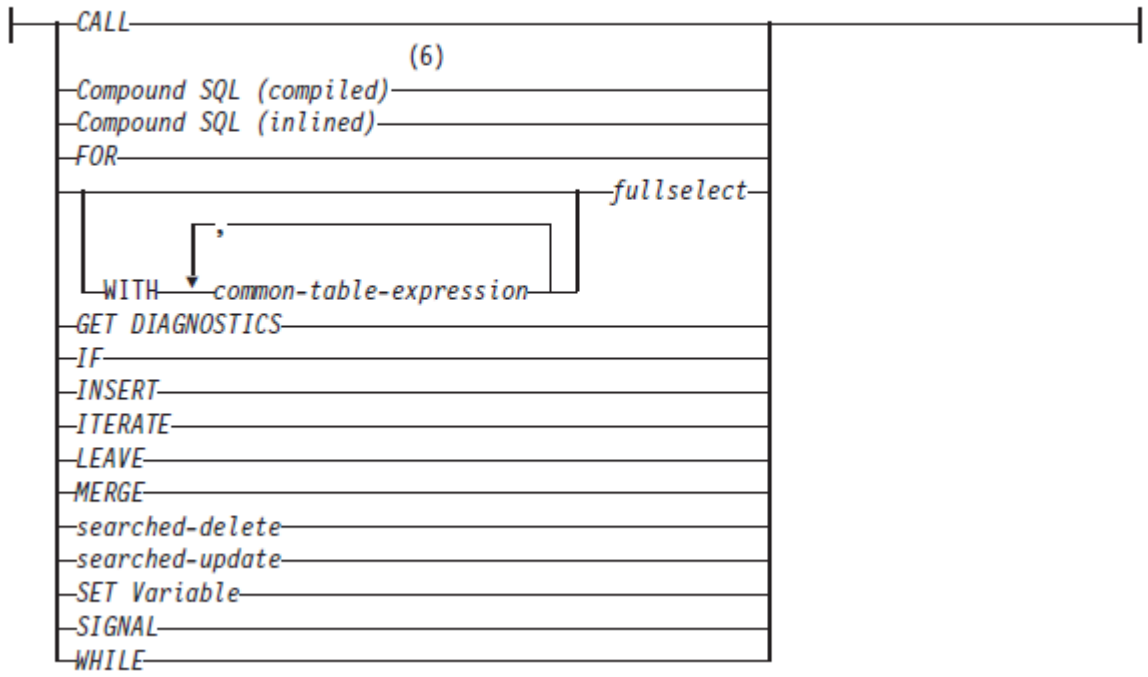
trigger-event:



triggered-action:



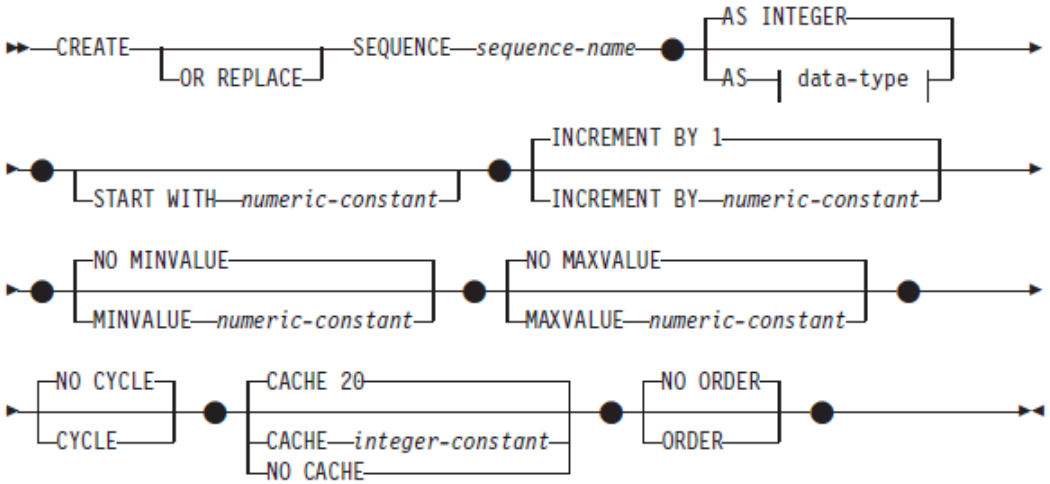
SQL-procedure-statement:



Example:

```
CREATE TRIGGER REORDER
AFTER UPDATE OF ON_HAND, MAX_STOCKED ON PARTS
REFERENCING NEW TABLE AS NTABLE
FOR EACH STATEMENT
BEGIN ATOMIC
SELECT ISSUE_SHIP_REQUEST(MAX_STOCKED - ON_HAND, PARTNO)
FROM NTABLE
WHERE (ON_HAND < 0.10 * MAX_STOCKED);END
```

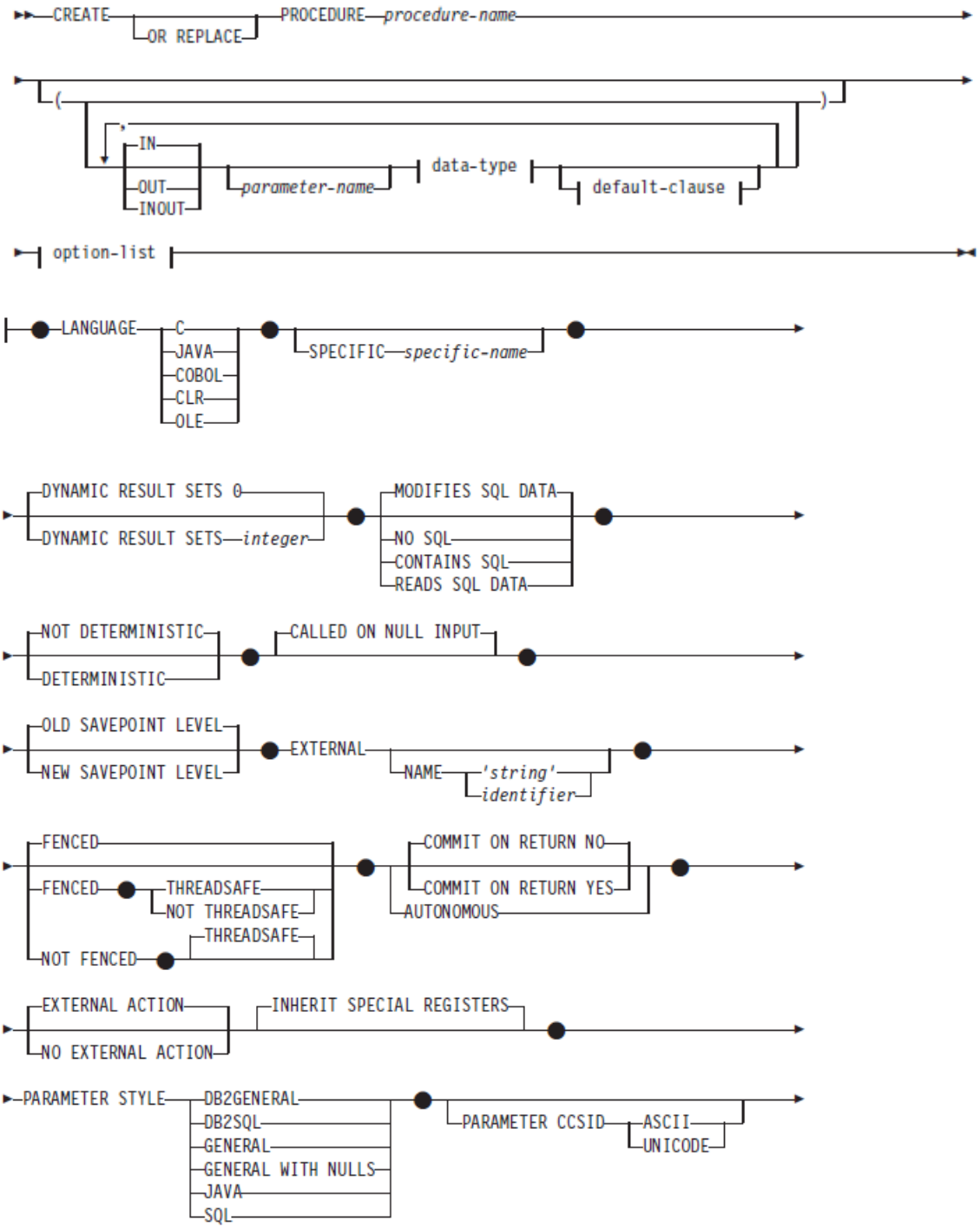
Sequences



Example:

```
CREATE SEQUENCE ORG_SEQ
START WITH 1
INCREMENT BY 1
NO MAXVALUE
NO CYCLE
    CACHE 24
```

Procedures



Example:

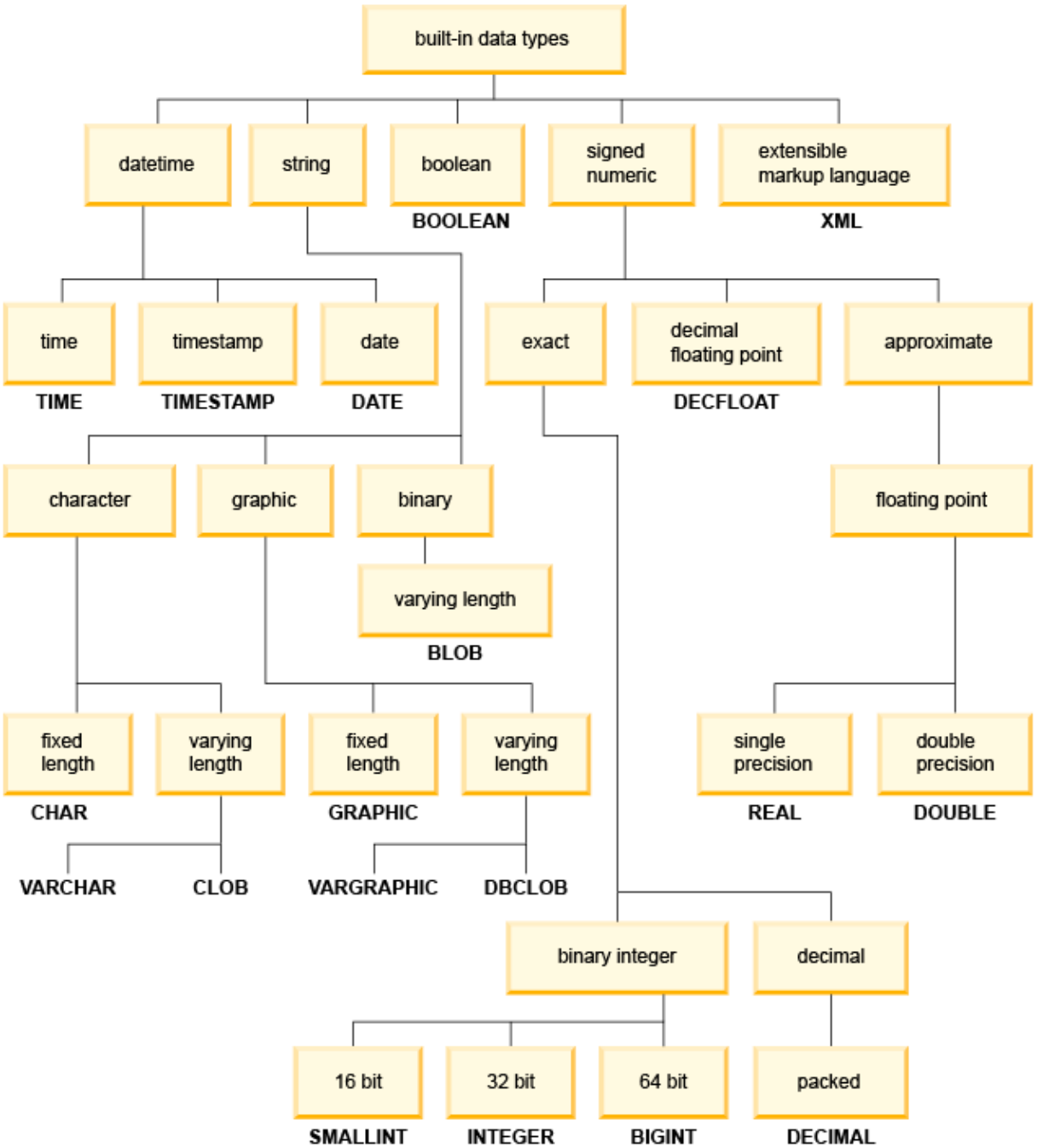
CREATE PROCEDURE PARTS_ON_HAND (**IN** PARTNUM **INTEGER,**
OUT COST **DECIMAL(7,2),**
OUT QUANTITY **INTEGER**)
EXTERNAL NAME ' parts.onhand'
LANGUAGE JAVA PARAMETER STYLE JAVA

Creating Datatype Packages

DB2V10.5 Datatypes

A complete description of datatypes is accessible from the following link:

http://www-01.ibm.com/support/knowledgecenter/SSEPGG_10.5.0/com.ibm.db2.luw.sql.ref.doc/doc/r0008483.html?cp=SSEPGG_10.5.0%2F2-12-2-3&lang=fr



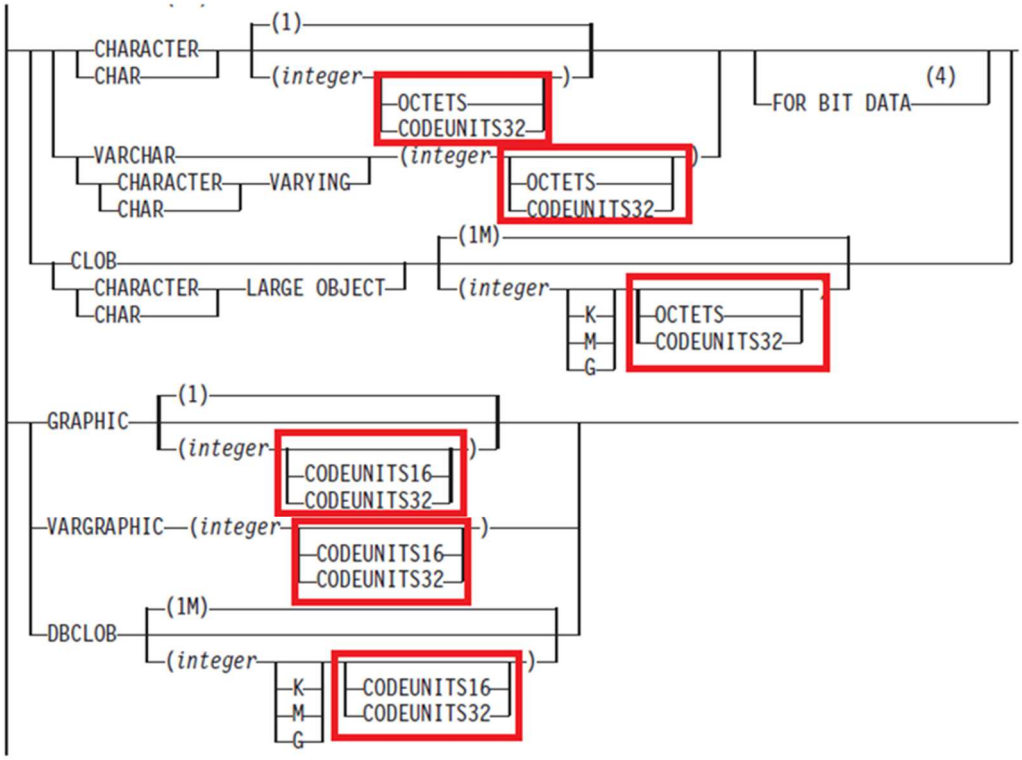
NCHAR, NVARCHAR, NCLOB

The NCHAR, NVARCHAR, NCLOB datatypes are not supported by MEGA.

OCTETS, CODEUNITS32, CODEUNITS16

OCTETS, CODEUNITS32 and CODEUNITS16 are units of length for the character string datatype.

They are not supported by MEGA.



LONG VARCHAR, LONG VARGHAPHIC

The LONG VARCHAR and LONG VARGRAPHIC datatypes are deprecated in version 10.5 but remain accepted in DB2 10.5 syntax.

Physical Parameters

Colors used:

Gray: parameters where reverse engineering is non-directional.

Red: parameters not processed.

Green: parameters reverse engineered.

Tables

- Clause type : table_check
  - CHECK NAME (String)
  - ENFORCED(Enumeration: ENFORCED,NOT ENFORCED,NOT ENFORCED TRUSTED, NOT ENFORCED NOT TRUSTED
- Clause type : table_block
  - DATA CAPTURE (Enumeration: CHANGES,NONE)
  - NOT LOGGED INITIALLY(Boolean)
  - WITH RESTRICT ON DROP
  - COMPRESSION MODE (Enumeration: COMPRESSION NO,COMPRESSION YES ADAPTIVE,COMPRESSION YES STATIC)
  - VALUE COMPRESSION (Boolean)
  - CCSID (UNICODE,ASCII)
  - SECURITY POLICY(String)
  - OPTIONS (VarChar)
  - AS ROW (Enumeration : BEGIN,END)
- Clause type : organize_clause
  - ORGANIZE BY (Enumeration :ROW,COLUMN,DIMENSIONS,KEY SEQUENCE,INSERT TIME)
  - ROW USING (Boolean)
  - ALLOW OVERFLOW(Boolean)
  - PCTFREE (Integer)
  - Not processed:
  -
- Clause type : organize_sequence_key_clause
  - COLUMN (Link)
  - STARTING_CONSTANT
  - ENDING_CONSTANT
- Clause type : distribution_clause
  - DISTRIBUTE BY (Enumeration :HASH,REPLICATION)
  - DISTRIBUTION COLUMNS(Link)
- Clause type : partitioning_clause
  - RANGE(Boolean)
- Clause type : partitioning_range_clause
  - PARTITION NAME(String)

- EVERY(String)
- Used Data Group(Link)

- Clause type: boundary_clause
  - BOUNDARY INCLUSIVE(INCLUSIVE,EXCLUSIVE)
  - BOUNDARY TYPE(STARTING,ENDING)
  - BOUNDARY VALUES(String)
- Clause type: tablespace_option
  - CYCLE(Boolean)
  - Used Data Group (link)
  - Used Data Group 2 (link)
  - Used Data Group 3 (link)

- Clause type : partition_expression
  - COLUMN(Link)
  - NULLS ORDER (Enumeration : FIRST,LAST)

- Clause type : table_period_defintion
  - PERIOD MODE (Enumeration : BUSINESS_TIME,SYSTEM_TIME)
  - BEGIN COLUMN(Link)
  - END COLUMN(Link)

Columns

- Clause type : column_block
  - COMPACT (Boolean)
  - COMPRESS (Boolean)
  - INLINE LENGTH (Boolean)
  - LOGGED (String)
  - IMPLICITLY HIDDEN (Enumeration : IMPLICITLY HIDDEN, NOT HIDDEN)
- Clause type : generated_column_spec
  - GENERATED (Enumeration : GENERATED ALWAYS, GENERATED DEFAULT)
  - GENERATION TYPE (Enumeration : AS GENERATED EXPRESSION,AS ROW CHANGE TIMESTAMP, AS ROW TRANSACTION TIMESTAMP, AS ROW TRANSACTION STARTD ID)
  - START WITH (String)
  - INCREMENT (String)
  - MAX VALUE (OPENED ENUMERATION : NO MAXVALUE)
  - MIN VALUE (OPENED ENUMERATION : NO MINVALUE)

- CACHE (OPENED ENUMERATION : NO CACHE)
- CYCLE (Boolean)
- ORDER (Boolean)
- GENERATE EXPRESSION (VarChar)

Indexes

- Clause type : index_block
  - BUSINESS_TIME WITHOUT OVERLAPS (Boolean)
  - PARTIONED (Enumeration : PARTITIONED ,NOT PARTITIONED)
  - Used Data Group (Link)
  - EXTEND USING (String)
  - VALUE LIST (VarChar)
  - PCTFREE (Percent)
  - LEVEL2 PCTFREE (Percent)
  - MINPCTUSED (String)
  - REVERSE SCANS (ALLOW,DISALLOW)
  - PAGE SPLIT (HIGHT,LOW,SYMMETRIC)
  - COLLECT(STATISTICS,DETAILED STATISTICS,SAMPLED DETAILED STATISTICS,UNSAMPLED DETAILED STATISTICS)
  - COMPRESS(YES,NO)
  - NULL KEYS (INCLUDE NULL KEYS, EXCLUDE NULL KEYS)
  - INCLUDE(Link)

Tablespaces

- Clause type : tablespace_block
  - STORE(LARGE,LOB,LONG,REGULAR,SYSTEM TEMPORARY,USER TEMPORARY)
  - NODEGROUP(Boolean)
  - NODEGROUP NAME(String)
  - PAGESIZE(String)
  - MANAGED(MANAGE BY AUTOMATIC STORAGE,MANAGED BY DATABASE,MANAGED BY SYSTEM)
  - EXTENTSIZE(String)
  - PREFETCHSIZE(String)
  - BUFFERPOOL(String)
  - OVERHEAD(String)
  - TRANSFERRATE(String)

- DROPPED TABLE RECOVERY(ON,OFF)
- AUTORESIZE(Boolean)
- INCREASESIZE(String)
- INITIALSIZE(String)
- MAXSIZE(String)
- FILE SYSTEM CACHING (FILE SYSTEM CACHING
- NO FILE SYSTEM CACHING (NO FILE SYSTEM CACHING,NO NO FILE SYSTEM CACHING)
- DATA TAG (String)

- Clause type : containers
  - Container-type(FILE,DEVICE)
  - Container-strings(VarChar)
  - Container –size (String)
  - container-numbers(String)

Unmanaged physical parameters

- AS

  Specifies that the columns of the table are based on the attributes of the structured type identified by type-name1. If type-name1 is specified without a schema name, the type name is resolved by searching the schemas on the SQLpath (defined by the FUNCPATH preprocessing option for static SQL and by the CURRENT PATH register for dynamic SQL). The type name must be the name of an existing user-defined type (SQLSTATE 42704) and it must be an instantiable structured type (SQLSTATE 428DP) with at least one attribute (SQLSTATE 42997).

- LIKE

  Specifies that the columns of the table have exactly the same name and description as the columns of the identified table (table-name1), view (view-name) or nickname (nickname). The name specified after LIKE must identify a table, view or nickname that exists in the catalog, or a declared temporary table. A typed table or typed view cannot be specified (SQLSTATE428EC).

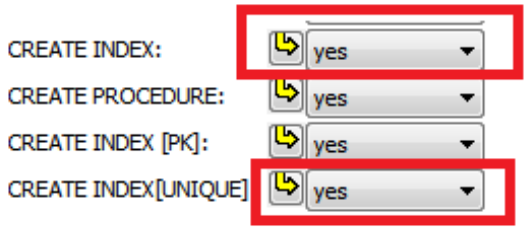
- TYPE_NAME

  Creates a typed table, which takes its structure from the specified composite type (name optionally schema-qualified). A typed table is tied to its type; for example the table will be dropped if the type is dropped (with DROP TYPE ... CASCADE). When a typed table is created, then the data types of the columns are determined by the underlying composite type and are not specified by the CREATE TABLE command. But the CREATE TABLE command can add defaults and constraints to the table and can specify storage parameters.

UNIQUE Constraint

Creating a UNIQUE constraint automatically creates a unique index of the same name. It is therefore recommended not to generate unique indexes.

In the generation options of a database, select one of the two options “Create Index” and “Create Index[Unique]”.

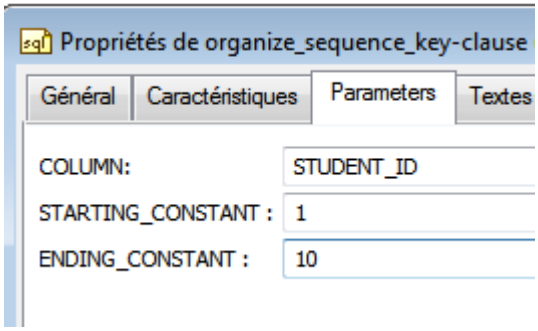


The same rule applies on PK constraints.

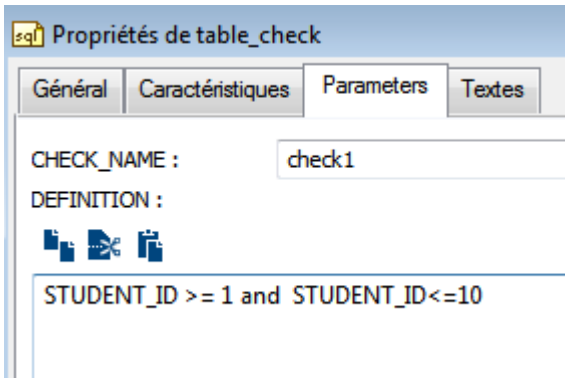
Organize_sequence_key-clause Clause

The “organize_sequence_key-clause” clause is reverse engineered into a check constraint.

For example:



Became:



A Unique index is also created on columns used in “organize_sequence_keyclause” clauses.

In DB2V10.5 reverse engineering of this clause is not supported.

‘EVERY \ DB2’ Properties

The “EVERY” property has no physical value in DB2 system tables. If this property is declared on a partition, it means that the partition is composed of several partitions.

For example:

```
CREATE TABLE PARTITION_range_4
(
ID INTEGER,
CONTENTS CLOB
)
PARTITION BY RANGE (ID) (STARTING 1 ENDING 100, STARTING 101 ENDING 400 EVERY 100)
```

Is equal to:

```
CREATE TABLE PARTITION_range_4
(
ID INTEGER,
CONTENTS CLOB
)
PARTITION BY RANGE (ID) (STARTING 1 ENDING 100, STARTING 101 ENDING 200, STARTING 201 ENDING 300, STARTING 301 ENDING 400)
```

PostgreSql 9.3: Syntax supported by HOPEX

This document details concepts of PostgreSQL 9.3 recognized by MEGA.

Detail of Concepts Managed by PostgreSQL

All SQL commands are available at the following address:

http://www.postgresql.org/docs/9.3/static/sql-commands.html

Tables

```
CREATE [ [ GLOBAL | LOCAL ] { TEMPORARY | TEMP } ] TABLE table_name ( [
  { column_name data_type [ DEFAULT default_expr ] [ column_constraint [ ... ] ]
    | table_constraint
    | LIKE parent_table [ like_option ... ] }
    [, ... ]
] )
[ INHERITS ( parent_table [, ... ] ) ]
[ WITH ( storage_parameter [= value] [, ... ] ) | WITH OIDS | WITHOUT OIDS ]
[ ON COMMIT { PRESERVE ROWS | DELETE ROWS | DROP } ]
[ TABLESPACE tablespace ]

CREATE [ [ GLOBAL | LOCAL ] { TEMPORARY | TEMP } ] TABLE table_name
    OF type_name [ (
  { column_name WITH OPTIONS [ DEFAULT default_expr ] [ column_constraint [ ... ] ]
    | table_constraint }
    [, ... ]
) ]
[ WITH ( storage_parameter [= value] [, ... ] ) | WITH OIDS | WITHOUT OIDS ]
[ ON COMMIT { PRESERVE ROWS | DELETE ROWS | DROP } ]
[ TABLESPACE tablespace ]
```

Indexes

```
CREATE [ UNIQUE ] INDEX [ CONCURRENTLY ] [ name ] ON table_name [ USING method ]
    ( { column_name | ( expression ) } [ COLLATE collation ] [ opclass ] [ ASC | DESC ] [ NULLS { FIRST | LAST } ] [, ...] )
    [ WITH ( storage_parameter = value [, ... ] ) ]
    [ TABLESPACE tablespace_name ]
    [ WHERE predicate ]
```

Tablespaces

```
CREATE TABLESPACE tablespace_name [ OWNER user_name ] LOCATION 'directory'
```

Views

```
CREATE [ OR REPLACE ] [ TEMP | TEMPORARY ] [ RECURSIVE ] VIEW name [ ( column_name [, ...] ) ]
    [ WITH ( view_option_name [= view_option_value] [, ... ] ) ]
    AS query
```

```
CREATE VIEW v_example_character_type AS
 SELECT col_char1, col_char2
  FROM example_character_type
 WHERE col_char1 = 'A';
```

Triggers

```
CREATE [ CONSTRAINT ] TRIGGER name { BEFORE | AFTER | INSTEAD OF } { event [ OR ... ] }
    ON table_name
    [ FROM referenced_table_name ]
    { NOT DEFERRABLE | [ DEFERRABLE ] { INITIALLY IMMEDIATE | INITIALLY DEFERRED } }
    [ FOR [ EACH ] { ROW | STATEMENT } ]
    [ WHEN ( condition ) ]
    EXECUTE PROCEDURE function_name ( arguments )

where event can be one of:

    INSERT
    UPDATE [ OF column_name [, ... ] ]
    DELETE
    TRUNCATE
```

Example:

```
CREATE OR REPLACE FUNCTION test() RETURNS trigger AS $$
      BEGIN
           RETURN 0;
      END IF;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_update
   BEFORE UPDATE ON example_character_type
   FOR EACH ROW
   EXECUTE PROCEDURE test();
```

Sequences

```
CREATE [ TEMPORARY | TEMP ] SEQUENCE name [ INCREMENT [ BY ] increment ]
    [ MINVALUE minvalue | NO MINVALUE ] [ MAXVALUE maxvalue | NO MAXVALUE ]
    [ START [ WITH ] start ] [ CACHE cache ] [ [ NO ] CYCLE ]
    [ OWNED BY { table_name.column_name | NONE } ]
```

Example:

```
CREATE SEQUENCE example_sequence
 INCREMENT 1
 MINVALUE 1
 START 101 ;
```

Functions

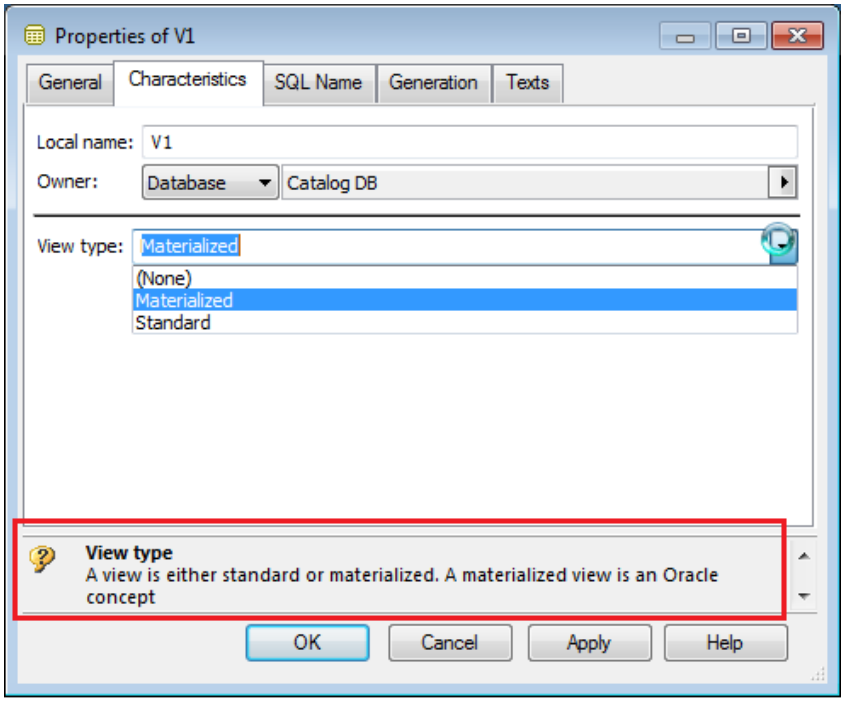
```
CREATE [ OR REPLACE ] FUNCTION
    name ( [ [ argmode ] [ argname ] argtype [ { DEFAULT | = } default_expr ] [, ...] ] )
    [ RETURNS rettype
      | RETURNS TABLE ( column_name column_type [, ...] ) ]
  { LANGUAGE lang_name
    | WINDOW
    | IMMUTABLE | STABLE | VOLATILE | [ NOT ] LEAKPROOF
    | CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT
    | [ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER
    | COST execution_cost
    | ROWS result_rows
    | SET configuration_parameter { TO value | = value | FROM CURRENT }
    | AS 'definition'
    | AS 'obj_file', 'link_symbol'
  } ...
    [ WITH ( attribute [, ...] ) ]
```

Example:

```
CREATE OR REPLACE FUNCTION test() RETURNS trigger AS $$
      BEGIN
            RETURN i + 1;
      END IF;
$$ LANGUAGE plpgsql;
```

Materialized views

This concept is managed in MEGA by specifying the appropriate type to a view.



However, MEGA presents two restrictions:

- It is not possible to specify a tablespace for storage of the view.
- More generally, it is not possible to specify physical parameters.

Domains and types

These are two concepts that enable management of structures of elementary datatypes. These concepts exist for other DBMSs.

These concepts are not initially managed. A detailed study is necessary.

"Trigger events"

This concept enables extension of the trigger concept to a database.

MEGA does not currently offer suitable management of this concept.

Creating Datatype Packages

PostgreSQL datatypes

A complete description of datatypes is accessible from the following link:

http://www.postgresql.org/docs/9.3/static/sql-commands.html

- Complete list

The following sections provide additional information.

Name	Aliases	Description
`bigint`	`int8`	signed eight-byte integer
`bigserial`	`serial8`	autoincrementing eight-byte integer
`bit [ (n) ]`		fixed-length bit string
`bit varying [ (n) ]`	`varbit`	variable-length bit string
`boolean`	`bool`	logical Boolean (true/false)
`box`		rectangular box on a plane
`bytea`		binary data ("byte array")
`character [ (n) ]`	`char [ (n) ]`	fixed-length character string
`character varying [ (n) ]`	`varchar [ (n) ]`	variable-length character string
`cidr`		IPv4 or IPv6 network address

Name	Aliases	Description
`circle`		circle on a plane
`date`		calendar date (year, month, day)
`double precision`	`float8`	double precision floating-point number (8 bytes)
`inet`		IPv4 or IPv6 host address
`integer`	`int`, `int4`	signed four-byte integer
`interval [ fields ] [ (p) ]`		time span
`json`		JSON data
`line`		infinite line on a plane
`lseg`		line segment on a plane
`macaddr`		MAC (Media Access Control) address
`money`		currency amount
`numeric [ (p, s) ]`	`decimal [ (p, s) ]`	exact numeric of selectable precision
`path`		geometric path on a plane
`point`		geometric point on a plane

Name	Aliases	Description
`polygon`		closed geometric path on a plane
`real`	`float4`	single precision floating-point number (4 bytes)
`smallint`	`int2`	signed two-byte integer
`smallserial`	`serial2`	autoincrementing two-byte integer
`serial`	`serial4`	autoincrementing four-byte integer
`text`		variable-length character string
`time [ (p) ] [ without time zone ]`		time of day (no time zone)
`time [ (p) ] with time zone`	`timetz`	time of day, including time zone
`timestamp [ (p) ] [ without time zone ]`		date and time (no time zone)
`timestamp [ (p) ] with time zone`	`timestamptz`	date and time, including time zone
`tsquery`		text search query
`tsvector`		text search document
`txid_snapshot`		user-level transaction ID snapshot

Name	Aliases	Description
uuid		universally unique identifier
xml		XML data

- Numeric types

Name	Storage Size	Description	Range
smallint	2 bytes	small-range integer	-32768 to +32767
integer	4 bytes	typical choice for integer	-2147483648 to +2147483647
bigint	8 bytes	large-range integer	-9223372036854775808 to +9223372036854775807
decimal	variable	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
numeric	variable	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
real	4 bytes	variable-precision, inexact	6 decimal digits precision
double precision	8 bytes	variable-precision, inexact	15 decimal digits precision
smallserial	2 bytes	small autoincrementing integer	1 to 32767
serial	4 bytes	autoincrementing integer	1 to 2147483647
bigserial	8 bytes	large autoincrementing integer	1 to 9223372036854775807

```
CREATE TABLE example_numeric_type
(
  col_bigint                bigint,
  col_decimal1              decimal,
  col_decimal2              decimal(8),
  col_decimal3              decimal(8,2),
  col_double_precision      double precision,
  col_integer               integer,
  col_numeric1              numeric,
  col_numeric2              numeric(8),
  col_numeric3              numeric(8,2),
  col_real                  real,
  col_serial                serial,
  col_bigserial             bigserial,
  col_smallserial           smallserial,
  col_smallint              smallint
);
```

- Monetary types

Name	Storage Size	Description	Range
money	8 bytes	currency amount	-92233720368547758.08 to +92233720368547758.07

```
CREATE TABLE example_monetary_type
(
  col_money      money
)
```

- Character strings

Name	Description
`character varying(n), varchar(n)`	variable-length with limit
`character(n), char(n)`	fixed-length, blank padded
`text`	variable unlimited length

```
CREATE TABLE example_character_type
(
  col_char1 char,
  col_char2 char(12),
  col_varchar1 varchar,
  col_varchar2 varchar(15),
  col_text text
)
```

- Binary types

Name	Storage Size	Description
`bytea`	1 or 4 bytes plus the actual binary string	variable-length binary string

```
CREATE TABLE example_binary_type
(
  col_bytea bytea
)
```

- Time types

Name	Storage Size	Description	Low Value	High Value	Resolution
`timestamp [ (p) ] [ without time zone ]`	8 bytes	both date and time (no time zone)	4713 BC	294276 AD	1 microsecond / 14 digits
`timestamp [ (p) ] with time zone`	8 bytes	both date and time, with time zone	4713 BC	294276 AD	1 microsecond / 14 digits
`date`	4 bytes	date (no time of day)	4713 BC	5874897 AD	1 day
`time [ (p) ] [ without time zone ]`	8 bytes	time of day (no date)	00:00:00	24:00:00	1 microsecond / 14 digits
`time [ (p) ] with time zone`	12 bytes	times of day only, with time zone	00:00:00+1459	24:00:00-1459	1 microsecond / 14 digits
`interval [ fields ] [ (p) ]`	12 bytes	time interval	-178000000 years	178000000 years	1 microsecond / 14 digits

```
CREATE TABLE example_datetime_type
(
  col_date              date,
  col_time1             time,
  col_time2             time(6),
```

```
  col_time3          time with time zone,
  col_time4          time(6) with time zone,
  col_timestamp1     timestamp,
  col_timestamp2     timestamp(6),
  col_timestamp3     timestamp with time zone,
  col_timestamp4     timestamp(6) with time zone,
  col_interval1      interval,
  col_interval2      interval(6),
  col_interval3      interval day,
  col_interval4      interval day(6)
)
```

- Boolean types

Name	Storage Size	Description
boolean	1 byte	state of true or false

```
CREATE TABLE example_boolean_type
(
  col_boolean boolean
)
```

- Geometric types

Name	Storage Size	Representation	Description
point	16 bytes	Point on a plane	(x,y)
line	32 bytes	Infinite line (not fully implemented)	((x1,y1),(x2,y2))
lseg	32 bytes	Finite line segment	((x1,y1),(x2,y2))
box	32 bytes	Rectangular box	((x1,y1),(x2,y2))
path	16+16n bytes	Closed path (similar to polygon)	((x1,y1),...)
path	16+16n bytes	Open path	[(x1,y1),...]
polygon	40+16n bytes	Polygon (similar to closed path)	((x1,y1),...)
circle	24 bytes	Circle	<(x,y),r> (center point and radius)

```
CREATE TABLE example_geometric_type
(
  col_point point,
  col_line line,
  col_lseg lseg,
  col_box box,
  col_path path,
  col_polygon polygon,
  col_circle circle
)
```

- Network types

Name	Storage Size	Description
cidr	7 or 19 bytes	IPv4 and IPv6 networks
inet	7 or 19 bytes	IPv4 and IPv6 hosts and networks
macaddr	6 bytes	MAC addresses

```
CREATE TABLE example_network_type
(
  col_cidr cidr,
  col_inet inet,
  col_macaddr macaddr
)
```

- Type : Bit String

Bit strings are strings of 1's and 0's. They can be used to store or visualize bit masks. There are two SQL bit types: `bit(n)` and `bit varying(n)`, where *n* is a positive integer. `bit` type data must match the length *n* exactly; it is an error to attempt to store shorter or longer bit strings. `bit varying` data is of variable length up to the maximum length *n*; longer strings will be rejected. Writing `bit` without a length is equivalent to `bit(1)`, while `bit varying` without a length specification means unlimited length.

```
CREATE TABLE example_bit_type
(
  col_bit1 bit,
  col_bit2 bit(3),
  col_bit_varying1 bit varying,
  col_bit_varying2 bit varying(5)
)
```

- Type : Other

```
CREATE TABLE example_other_type
(
  col_json json,
  col_uuid uuid,
  col_xml xml
)
```

Physical Parameters

Colors used:

Gray: parameters where reverse engineering is non-directional.

Red: parameters not processed.

Green: parameters reverse engineered.

Tables

- Clause type: table properties
  - POSTGRESQL_TBL_TEMPORARY (boolean)
  - *POSTGRESQL_ TBL_LOG (boolean)*
  - POSTGRESQL_ TBL_NOT_EXISTS (boolean)
  - POSTGRESQL_ TBL_COMMIT (PRESERVE ROWS, DELETE ROWS, DROP)
  - POSTGRESQL_ TBL_OIDS (yes, no)
  - Link to a data group
  - Link to a clause type storage parameter

Example:

CREATE TABLE example_with_storage_parameter (code char(5)) WITH (fillfactor=60, autovacuum_enabled);

- Clause type: table constraint
  - POSTGRESQL_CONSTRAINT_NAME (String)
  - POSTGRESQL_CONSTRAINT_TYPE (UNIQUE, PRIMARY KEY, EXCLUDE, FOREIGN KEY)
  - POSTGRESQL _EXPRESSION (String)
  - POSTGRESQL_DEFERRABLE (yes, no)
  - POSTGRESQL_DEFERRED (yes, no)
  - POSTGRESQL_MATCH_TYPE (Full, partial, simple)
  - POSTGRESQL_UPDATE_ACTION (String)
  - POSTGRESQL_DELETE_ACTION (String)
  - POSTGRESQL_CONDITION (String)
  - Link to a clause type storage parameter
  - Link to a clause type column option (multiple clause)
  - Link to a data group
  - Link to a table
  - Link to a column

```
CREATE TABLE table_with_constraint (
   did     integer CHECK (did > 100),
   name    varchar(40)
);
```

Columns

- Clause type: column properties

- POSTGRESQL_COLLATE (String)

Example:

CREATE TABLE table_with_collation (c1 serial, c2 text COLLATE "C");

- Clause type: column constraint
  - POSTGRESQL_CONSTRAINT_NAME (String)
  - POSTGRESQL_CONSTRAINT_TYPE (UNIQUE, PRIMARY KEY, FOREIGN KEY)
  - POSTGRESQL_CONSTRAINT_EXPRESSION (String)
  - POSTGRESQL_DEFERRABLE (yes, no)
  - POSTGRESQL_DEFERRED (yes, no)
  - POSTGRESQL_MATCH_TYPE (Full, partial, simple)
  - POSTGRESQL_ACTION (String)
  - Link to a clause type storage parameter
  - Link to a data group
  - Link to a table
  - Link to a column

Indexes

- Clause type: index properties
  - POSTGRESQL_IDX_CONCURRENTLY (boolean)
  - POSTGRESQL_IDX_METHOD (btree,hash,gist,spgist,gin)
  - POSTGRESQL_PREDICATE (String)
  - POSTGRESQL_EXPRESSION (String)
  - Link to a clause type storage parameter
  - Link to a data group
- Clause type: column option

Examples:

CREATE INDEX example_character_idx ON example_character_type USING btree (col_char2) WITH (fillfactor=60)

CREATE INDEX index_with_tbs ON example_character_type (col_varchar1) TABLESPACE dbspace;

Tablespaces

- Clause type: tablespace properties
  - POSTGRESQL_TBS_LOCATION (String)

```
CREATE TABLESPACE dbspace LOCATION 'C:\Travail\2014\Database\PostGreSql';

CREATE TABLE table_with_tablespace (
        id serial,
        name text,
        location text
) TABLESPACE dbspace;
```

Generic clause types

- Clause type: storage parameter
  - POSTGRESQL_ FILLFACTOR (integer)
  - POSTGRESQL_VACUUM (boolean)
  - POSTGRESQL_THRESHOLD (integer)
  - POSTGRESQL_SCALE_FACTOR (float)
  - POSTGRESQL_ANALYSE_THRESHOLD (integer)
  - POSTGRESQL_ANALYSE_SCALE_FACTOR (double)
  - POSTGRESQL_COST_DELAY (integer)
  - POSTGRESQL_COST_LIMIT (integer)
  - POSTGRESQL_FREEZE_MIN_AGE (integer)
  - POSTGRESQL_FREEZE_MAX_AGE (integer)
  - POSTGRESQL_FREEZE_TABLE_AGE (integer)
  - POSTGRESQL_MULTIXACT_FREEZE_MIN_AGE (integer)
  - POSTGRESQL_ MULTIXACT_FREEZE_MAX_AGE (integer)

- Clause type: column option
  - Link to a column
  - POSTGRESQL_EXPRESSION (String)
  - POSTGRESQL_COLLATE (String)
  - POSTGRESQL_CLASS (String)
  - POSTGRESQL_SORT (ASK, DESK)
  - POSTGRESQL_NULL (NULLS, NULLS FIRST, NULLS LAST)

Unmanaged physical parameters

- INHERITS

  The optional `INHERITS` clause specifies a list of tables from which the new table automatically inherits all columns. Use of `INHERITS` creates a persistent relationship between the new child table and its parent table(s). Schema modifications to the parent(s) normally propagate to children as well, and by default the data of the child table is included in scans of the parent(s).

- LIKE

  The `LIKE` clause specifies a table from which the new table automatically copies all column names, their data types, and their not-null constraints. Unlike `INHERITS`, the new table and original table are completely decoupled after creation is complete. Changes to the original table will not be applied to the new table, and it is not possible to include data of the new table in scans of the original table. Default expressions for the copied column definitions will only be copied if `INCLUDING DEFAULTS` is specified. The default behavior is to exclude default expressions, resulting in the copied columns in the new table having null defaults.

- TYPE_NAME

  Creates a *typed table*, which takes its structure from the specified composite type (name optionally schema-qualified). A typed table is tied to its type; for example the table will be dropped if the type is dropped (with `DROP TYPE ... CASCADE`). When a typed table is created, then the data types of the columns are determined by the underlying composite type and are not specified by the `CREATE TABLE` command. But the `CREATE TABLE` command can add defaults and constraints to the table and can specify storage parameters.

Properties pages

An Sql Clause contains the definition of all physical parameters of all DBMSs managed by MEGA. To display the correct properties page according to the DBMS and the clause type, a specific properties page must be generated for each DBMS version.

- Create a MetaAttributGroup "PostgreSQL_93_Parameters"

  This should be connected to the SQL Clause MetaClass

- Create a MetaPropertyPage "PostgreSQL_93_ Parameters"

  Setup as follows:

  [Page]

  Order=10

  [Filter]

  Condition = (ItemCount(~ISl(gRH(HTB8[IsClauseForPostgreSQL93])>0)

  [Template]

  ExtraParam=IncludeProfile(~ESl(CMH(HHX7[GeneratePropertyPageForPostgreSQL93]),Origin(Macro)

  Where IsClauseForPostgreSQL93 is a query and GeneratePropertyPageForPostgreSQL93 is a macro. We can use as models the query IsClauseForPostgre93 and the macro GeneratePropertyPageForPostgreSQL93.

- Clause type: table constraint