Cyril Seguenot

2017

Résumé

Support d’accompagnement d’une formation à la modélisation de bases de données faite en présentiel

Modélisation de bases de données

Support de cours

Toute représentation ou reproduction intégrale ou partielle faite sans le consentement de l’auteur ou de ses ayants droit ou ayants cause est illicite selon le Code de la propriété intellectuelle (Art L 122-4) et constitue une contrefaçon réprimée par le Code pénal.

Seules sont autorisées (Art L122-5) les copies ou reproductions strictement réservées à l’usage privé du copiste et non destinées à une utilisation collective, ainsi que les analyses et courtes citations justifiées par le caractère critique, pédagogique ou d’information de l’œuvre à laquelle elles sont incorporées, sous réserve, toutefois, du respect des dispositions des articles L 122-10 à L 122-12 du même Code, relatives à la reproduction par reprographie.

©Cyril Seguenot 2017

SOMMAIRE

[1 Introduction 3](#_Toc479078066)

[2 La méthode Merise 3](#_Toc479078067)

[2.1 Objectif 3](#_Toc479078068)

[2.2 Niveau conceptuel 4](#_Toc479078069)

[2.3 Niveau organisationnel 4](#_Toc479078070)

[2.4 Niveau logique 5](#_Toc479078071)

[2.5 Niveau Physique 6](#_Toc479078072)

[3 Le modèle entité/relation (E/R) 6](#_Toc479078073)

[3.1 Les entités et les propriétés 6](#_Toc479078074)

[3.2 Les relations 10](#_Toc479078075)

[3.3 Processus global de modélisation 15](#_Toc479078076)

[4 Le modèle relationnel 15](#_Toc479078077)

[4.1 Les concepts manipulés 15](#_Toc479078078)

[4.2 Transformation du MCD en MLD 17](#_Toc479078079)

[4.3 Transformation du MLD en MPD 18](#_Toc479078080)

[4.4 Les contraintes d’intégrité 19](#_Toc479078081)

[4.5 Génération du script SQL 20](#_Toc479078082)

[5 Le SGBD SQL Server 21](#_Toc479078083)

[5.1 Connexion 22](#_Toc479078084)

[5.2 Utilisateurs et rôles 22](#_Toc479078085)

[5.3 Mappage Connexion - utilisateur 22](#_Toc479078086)

[5.4 Schéma 26](#_Toc479078087)

# Introduction

Nous présenterons tout d’abord succinctement la méthode Merise et ses différents niveaux de description d’un système d’informations.

Nous nous intéresserons ensuite plus particulièrement à la description du système du point de vue des données. Nous présenterons pour cela en détails le modèle entité/relation, qui est un modèle conceptuel de données (MCD), puis le modèle relationnel, qui est sa retranscription aux niveaux logique et physique.

# La méthode Merise

## Objectif

Merise est une méthode d'analyse, de conception et de réalisation de systèmes d’informations (SI)

Elle est devenue un projet opérationnel au début des années 1980 à la demande du ministère de l'industrie, et a surtout été utilisée en France, et principalement pour les projets d'envergure, notamment des grandes administrations publiques ou privées.

Merise est basée sur le principe de la séparation des données et des traitements, et décrit le système d'information sous 3 angles complémentaires :

* Les échanges / la communication entre les parties du système pour appréhender son comportement face aux évènements.
* Les traitements : décrivent comment le système prend en charge les événements en entrée, et comment il produit les événements en sortie.
* Les données : constituent la mémoire du système d'information



**DOSSIER D'INSCRIPTION**

Je voudrais m'inscrire au concours



Accueil

**Nom :**

**Diplôme :**

Résultat

Déclencheur

Accusé de réception



Traitement

Mémorisation



Echanges

Modèles de communication

Modèles de traitement

Modèles de données

Merise définit plusieurs modèles qui sont répartis sur plusieurs niveaux :

|  |  |  |  |
| --- | --- | --- | --- |
| **Niveau \ Modèle** | **Interface** | **Statique** | **Dynamique** |
| **Conceptuel** | MCC | MCD | MCT CVO |
| **Organisationnel** | MOC | MOD | MOT CVO |
| **Logique** |  | MLD | MLT |
| **Physique** | Maquette IHM | MPD | Code |

## Niveau conceptuel

L'étude conceptuelle s'attache aux invariants de l'entreprise ou de l'organisme du point de vue du métier : quels sont les activités, les métiers gérés par l'entreprise, quels sont les grands processus traités, de quoi parle-t-on en matière de données, quelles notions manipule-t-on ?... Et ce, indépendamment des choix techniques ou d’organisation, qui ne sont abordés que dans les niveaux suivants.

**MCC** : modèle conceptuel de communication

**MCD** : Modèle Conceptuel de Données

Il décrit la structure des données du SI, en décrivant les concepts et les relations entre eux. Exemple : qu’est ce qui décrit un client, une commande, un produit, et quelles relations existe-t-il entre eux ?

**MOT/CVO** : Modèle Organisationnel des Traitements / Cycle de Vie des Objet.

Il répond à la question « Qui fait quoi ? »

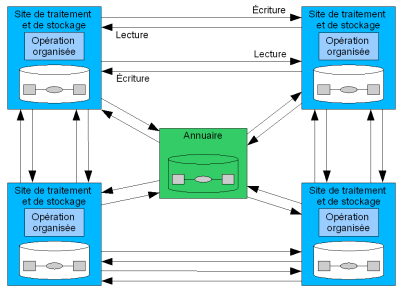
## Niveau organisationnel

Le niveau organisationnel fait intervenir l’aspect géographique, avec la notion générale de **site**, qui peut désigner par exemple un poste de travail dans une autre pièce ou un autre bâtiment, une succursale…

**MOC** : modèle organisationnel de communication.

Le MOC représente les flux de communication entre les sites du système, et entre le système et l’extérieur.

Exemple : le MOC ci-dessous décrit les échanges dans une gestion décentralisée des données, dans laquelle chaque site peut faire à la fois du traitement et du stockage de données :



**MOD** : Modèle Organisationnel de Données.

Détermine quel site manipule telle partie du MCD

Objectifs :

* Maîtriser la redondance d'information entre les différents sites
* Dimensionner les moyens nécessaires à mettre en œuvre au niveau des sites (taille des disques durs, quantité de mémoire vive …)

## Niveau logique

À ce niveau, on précise les modèles conceptuels en faisant des choix d’organisation.

**MLD** : Modèle Logique de Données. Il décrit les données et leur organisation de la façon la plus détaillée possible, mais sans faire référence à leur implémentation physique dans la base.

Types d’organisations possibles :

|  |  |
| --- | --- |
|  | **Le modèle hiérarchique : l**es données sont classées hiérarchiquement, selon une arborescence descendante. Ce modèle utilise des pointeurs entre les différents enregistrements. Il s'agit du premier modèle de SGBD |
|  | **Le modèle réseau** : comme le modèle hiérarchique ce modèle utilise des pointeurs vers des enregistrements. Toutefois la structure n'est plus forcément arborescente dans le sens descendant |
|  | **Le modèle relationnel** : les données sont décrites sous forme de tables comportant des colonnes, dont on précise le type. Les tables sont reliées par des relations, et les données sont manipulées selon la théorie mathématique des relations. |
|  | **Le modèle déductif** : les données sont représentées sous forme de tables, mais leur manipulation se fait par calcul de prédicats |
|  | **Le modèle objet** : les données sont stockées sous forme d'objets, c'est-à-dire de structures appelées classes présentant des données membres. Les champs sont des instances de ces classes |

Dans le cas d’un modèle de type relationnel, le MLD décrit :

* Les entités
* Les propriétés
* Les relations en termes de clés primaires et de clés étrangères

**MLT** : Modèle Logique des Traitements

## Niveau Physique

**MPD** : Modèle Physique de Données

Le MPD est une spécialisation du MLD pour un SGBD donné (SQL Server, Oracle, DB2, MySQL…). Il décrit les tables et propriétés en utilisant les types de données et d’objets propres au SGBD visé.

Au niveau physique, les traitements sont quant à eux réellement implémentés par du code.

# Le modèle entité/relation (E/R)

Le modèle entité/relation, abrégé en E/R ou ERD (entity-relationship diagram) est une façon de représenter un modèle conceptuel de données (MCD). Il représente de façon statique, à l’aide d’entités et de relations, l’ensemble des données gérées par une application ou un système d’information.

L'application gère les valeurs prises par ces données au rythme de survenue des événements.

Les règles de transition d'un état à un autre des données sont fournies par le modèle de traitement.

## Les entités et les propriétés

### Définitions

**Une entité** est la représentation d’un élément abstrait (immatériel) ou concret (matériel), qui regroupe des informations qui lui sont spécifiques.

Numéro de série

Marque

Modèle

Année

Véhicule

Nom de l’entité

Propriété identifiant

Propriétés

Toute entité doit posséder **un** **identifiant,** qui est une propriété permettant de distinguer une occurrence de l’entité parmi les autres.

**Une occurrence** est un exemplaire de l’entité

Exemple : si l’on considère l’entité Véhicule décrite plus haut, en voici des occurrences :

* La Renault Clio V de 2015 dont le N° de série est 598756423598
* La Peugeot 3008 de 2016 dont le N° de série est 745632159875
* La Peugeot 3008 de 2016 dont le N° de série est 957845745698
* …

NB/ On peut faire une analogie avec la programmation orientée objet : entité ≡ classe, et occurrence ≡ objet.

**Une propriété** (on dit aussi **attribut**) est la plus petite partie d’information manipulée par le système.

### Règles de modélisation

#### Atomicité

Une propriété doit être atomique, c’est-à-dire non décomposable.

Cette notion est relative, car elle dépend des besoins. Selon le système décrit, un même concept pourra ainsi être décrit par une seule propriété, ou par une entité avec plusieurs propriétés, voire même par plusieurs entités.

Ex : Pour modéliser l’adresse d’un client, on peut simplement créer une propriété Adresse sur l’entité Client. Mais si on veut pouvoir associer plusieurs adresses au client (ex : pour la facturation et pour la livraison), il faut dans ce cas créer une entité Adresse. Pour éviter la redondance d’information (cf. plus bas), on créera aussi une entité Commune

Id

Nom

Prénom

Client

Id

Numéro

Rue

Adresse

Id

Nom

Prénom

Adresse

Client

Id

Code postal

Nom commune

Commune

La description doit donc être en adéquation avec les besoins, afin que chaque propriété puisse être considérée comme atomique.

NB/ Si cette règle est respectée, le modèle est dit en « première forme normale (1FN) »

#### Dépendance fonctionnelle directe

Les propriétés de l’entité doivent être en dépendance fonctionnelle directe avec son identifiant.

Exemple :

Numéro de série

Marque

Modèle

Année

Nom concessionnaire

Adresse concessionnaire

Véhicule

Numéro de série

Marque

Modèle

Année

Véhicule

Code

Nom

Adresse

Concessionnaire

Le nom et l’adresse du concessionnaire ne sont pas en dépendance fonctionnelle directe avec le numéro de série du véhicule.

Si la dépendance fonctionnelle est correcte, à une valeur de l’identifiant ne doit correspondre qu’une seule valeur de chacune des propriétés.

Exemple : le nom d’un salarié du concessionnaire ne peut pas figurer dans l’entité Concessionnaire.

#### Non-redondance

Une propriété ne doit figurer qu’une seule fois sur un modèle. Exemple :

Numéro de série

Marque

Modèle

Année

Véhicule

Code

Nom

Adresse

Marque

Concessionnaire

Numéro de série

Modèle

Année

Véhicule

Code

Nom concession

Adresse

Concessionnaire

Code marque

Nom marque

Marque

#### Elémentarité

Une propriété ne doit pas pouvoir être déduite des autres.

Exemple :

Numéro de série

Marque

Modèle

Millésime

Date mise circulation

~~Ancienneté~~

~~Description~~

Véhicule

Si l’on considère que la description est du type « Renault Clio V de 2015 », alors cette information peut être déduite des propriétés Marque, modèle et millésime.

De même, l’ancienneté du véhicule peut être déduite de sa date de 1ère mise en circulation

#### Autres règles

**Pas de synonymes** : il ne faut pas créer plusieurs propriétés désignant la même information. Ex : « Numéro de série » et « Numéro d’identification du véhicule »

**Pas de polysémie** (même mot avec plusieurs sens) : des informations différentes doivent être modélisées par des propriétés différentes.

Ex : une personne et un véhicule ont tous deux un nom (nom du modèle pour le véhicule), mais ce nom sera décrit par des propriétés distinctes, dans des entités distinctes.

#### Recommandations

Dans le cadre du modèle relationnel, une application stricte de ces règles peut parfois produire une base de données peu propice à l’écriture de requêtes SQL performantes. Dans ce cas, on fera quelques adaptations, quitte à ne pas respecter toutes les règles de modélisation (on dit qu’on dénormalise le modèle).

Il est important d’être pragmatique et de rechercher le meilleur compromis entre le respect des règles de modélisation et la facilité d’écriture/performance des requêtes

## Les relations

Les entités ont bien entendu des liens de sens entre elles. Souvent cela peut s’exprimer par un verbe. Exemples :

* Un concessionnaire vend des véhicules.
* Un client passe des commandes.
* Une personne suit un stage.
* …

Dans le modèle E/R, les liens sont modélisés par des **relations**, avec des **cardinalités**. Dans le cas particulier de la notation Merise, les liens sont modélisés par des **associations** et peuvent relier plus de 2 entités.

### Les cardinalités

Les cardinalités indiquent les nombres minimum et maximum d’occurrences possibles de chaque côté de la relation.

Une cardinalité est représentée par un symbole précis, éventuellement accompagné d’une paire de nombres, mais le symbole suffit :

|  |  |
| --- | --- |
| Les relations et leurs cardinalités  http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc38058.1600/doc/image/pentri1310760632381_3-rad1232021031084.image | Exemple  **Client**  **Commande** |

Dans l’exemple ci-dessus :

* Une commande est obligatoirement passée par un client et un seul
* Un client peut passer zéro ou plusieurs commandes

Remarques :

* Dans la pratique, c’est cette relation avec les cardinalités 1,1 et 0,n qu’on rencontre le plus souvent.
* La cardinalité 1,n n’est pas utilisée, car même si elle peut avoir un sens, elle induit ensuite dans le MPD la création de contraintes qui empêchent l’insertion de données dans les tables.

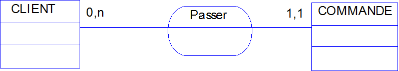
### Les différentes notations

Il existe plusieurs notations pour représenter les relations :

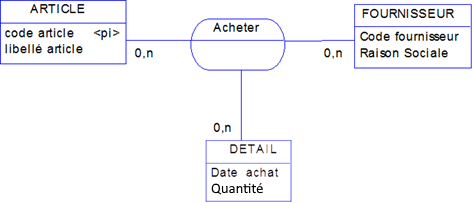
**La notation Entité / Relation** (E/R) est celle que l’on vient de voir.

**La notation Merise** : dans cette notation, on utilise plutôt des **associations**. Cette notion est proche de celle de relation, mais un peu plus large. L’association est représentée par une bulle, dans laquelle on place une expression pour la décrire (souvent un verbe). Les exemples ci-dessous illustrent les différences entre association et relation :

/!\ Les cardinalités d’une association sont inversées par rapport à celles d’une relation !



Une association peut relier 3 entités :



Dans la notation E/R, cette association est modélisée par une entité.

**Autres notations :**

|  |  |
| --- | --- |
| **IDEF1X**  http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc38058.1600/doc/image/pentri1310760632381_3-rad1232021042256.image | **Barker**  http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc38058.1600/doc/image/pentri1310760632381_3-rad1232021046960.image  Le style de ligne définit le caractère obligatoire :  http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc38058.1600/doc/image/pentri1310760632381_3-rad1232021048460.image |

### Lien identifiant et identifiant relatif

Certaines entités ne peuvent pas exister sans d’autres. Par exemple, une concession automobile ne peut pas exister sans une marque.

Pour représenter cette relation forte, l’identifiant le la marque devrait faire partie de celui de la concession. Mais dans ce cas, on ne respecterait plus la règle de non-redondance. La notion de lien identifiant permet de résoudre le problème :

Code concession

Nom concession

Adresse

Concession

Code marque

Nom marque

Marque

Code concession

Code marque

Nom concession

Adresse

Concession

Code marque

Nom marque

Marque

A droite, le lien identifiant est symbolisé de façon particulière. Il permet de dire que la relation fait partie de l’identifiant de la concession, et qu’une concession ne peut pas exister sans marque.

Le code concession devient un identifiant **relatif**, car il n’est plus suffisant à lui seul pour garantir l’unicité.

Un lien identifiant est stable dans le temps. La concession ne peut ainsi pas changer de marque.

Un autre exemple encore plus flagrant est celui d’un hôtel et de ses chambres. Le lien identifiant qui existe entre les deux a été représenté ci-dessous par une association Merise :

|  |  |
| --- | --- |
| id_relatif | Dans la notation Merise, un lien identifiant est symbolisé par des parenthèses autour de la cardinalité |

### Relations multiples entre 2 entités

Deux entités peuvent avoir plusieurs relations entre elles. Dans ce cas, on précise le sens de chaque relation par un libellé.

Exemple :

Personne

Concession

Dirige

Est cliente de

Si on veut modéliser à la fois le dirigeant d’une concession et ses clients, on peut utiliser la même entité Personne. Mais il existe bien dans ce cas 2 liens de natures différentes entre les 2 entités.

### Relation d’une entité sur elle-même

Personne

Est parente de

Certaines personnes peuvent n’avoir de lien de parenté avec aucune autre, c’est pourquoi la cardinalité doit être 0,1 et non 1,1 d’un des 2 côtés de la relation.

### Héritage

Plusieurs entités peuvent avoir une nature et des propriétés communes, tout en ayant chacune des propriétés spécifiques. Dans ce cas, il existe une relation d’héritage entre elles.

Exemple :

Id

Nom

Prénom

**Mail**

Personne

Matricule

Fonction

Service

Poste

Salarié

Adresse

CP

Ville

Client

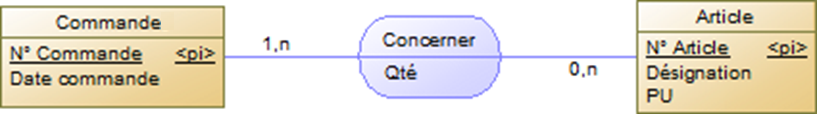
Les salariés et les clients sont aussi des personnes ayant un nom et un prénom, mais les salariés ont des propriétés spécifiques et les clients également.

Il n’y a toujours qu’une entité mère, mais il peut y avoir 1 à n entités dérivées.

NB/ La relation d’héritage dans le modèle E/R est analogue à celle qui peut exister entre des classes dans un diagramme UML de classes.

### Relations avec attributs

Une relation peut elle-même comporter des attributs, comme illustré dans l'exemple ci-dessous :



Ici, la quantité commandée pour chaque article est spécifiée sur la relation entre la commande et l'article. En effet, cette information n'est pas une caractéristique de l'article, ni de la commande.

NB/ Dans la notation Barker, les attributs s relations ne sont pas directement visibles sur le schéma

## Processus global de modélisation

Pour concevoir un modèle conceptuel de données, on procédera de la façon suivante :

* Déterminer les entités
* Déterminer les propriétés et l’identifiant de chacune
* Vérifier les règles de dépendance fonctionnelle, de non-redondance, d’atomicité et d’élémentarité
* Créer les relations entre les entités (éventuellement lien identifiants), avec les bonnes cardinalités.
* Vérifier que le modèle répond bien à toutes les exigences métier formulées par le demandeur, et qu’il permet bien de fournir les résultats attendus.

On procède généralement de façon itérative, en enrichissant progressivement le modèle, ou en supprimant au contraire ce qui n’est pas utile, et en répétant plusieurs fois ce processus.

# Le modèle relationnel

Comme nous l’avons vu plus haut, il existe différents modèles logiques de données (hiérarchique, objet…etc.). Nous allons voir ici en détail le modèle relationnel, qui est l’un des plus utilisés. Il se retranscrit aux niveaux logique et physique à travers différents concepts.

## Les concepts manipulés

Le tableau ci-dessous indique les correspondances entre les concepts manipulés au niveau conceptuel d’une part, et logique/physique d’autre part :

|  |  |
| --- | --- |
| **Conceptuel** | **Logique/physique** |
| Entités | Tables |
| Propriétés | Colonnes (= champs) |
| Identifiants | Clés primaires |
| Relations | Clés étrangères ou tables |

Les clés primaires et étrangères font partie des **contraintes d’intégrités**, que nous aborderons en détail un peu plus loin.

Les tables stockent les valeurs des champs dans des **lignes de table**.

**Les domaines**

Dans un MCD ou un MLD, la notion de domaine se rapproche de celle d’un type de données, mais sans être aussi précise. Nous pouvons par exemple définir des domaines : Id, Code, Libellé court, Libellé long, Taux, Durée, Longueur…etc., afin de typer les propriétés des entités.

Avant la transformation du MLD en MPD, on pourra associer à chaque domaine un type du SGBD visé, de façon à ce que les colonnes des tables soient correctement typées.

Exemple :

|  |  |
| --- | --- |
| **Domaine** | **Type associé du SGBD** |
| Id | Integer |
| ShortLabel | varchar(30) |
| LongLabel | nvarchar(100) |
| … | … |

**Remarques**

Les logiciels de modélisation ne font pas tous une distinction claire entre les différents types de modèles (conceptuel, logique et physique). Par exemple, SAP Power Designer distingue bien les trois, tandis qu'Oracle Data Modeler ne gère que deux types de modèles appelés « logique » et « relationnel ». Le premier correspond en fait plutôt au MCD, et le second au MPD.

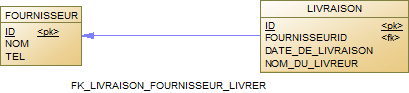
Il est tout à fait possible et même courant de passer directement du MCD au MPD. Il faut dans ce cas décrire suffisamment précisément les propriétés, identifiants, domaines et relations au niveau conceptuel.

L’utilisation de domaines n’est pas obligatoire. Les logiciels de modélisation fournissent des listes de types qu’on peut utiliser dans les MCD et MLD.

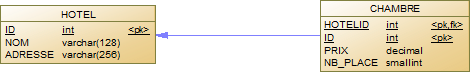
## Transformation du MCD en MLD

On passe du MCD au MLLD en appliquant les règles suivantes (X vaut 0 ou 1) :

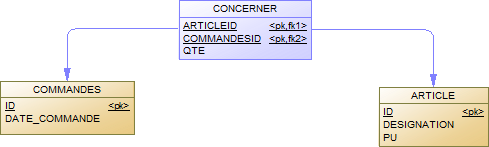
* Les entités sont transformées en tables. L'identifiant de l'entité devient la clé primaire de la table.
* Les relations X,1 - X,N donnent lieu à des clés étrangères. L'identifiant de l'entité mère devient une clé étrangère dans l'entité fille



Remarque : dans le cas d’un lien identifiant, l’identifiant de l’entité mère fera aussi partie de la clé primaire dans la table fille :



* Chaque relation X,N - X,N est transformée en table, dont la clé primaire est constituée des identifiants des entités en relation.



## Transformation du MLD en MPD

Le modèle physique de données (MPD) précise encore le MLD en étant spécifique à un SGBD donné (SQL Server, Oracle, DB2, MySQL…), c’est à dire en utilisant les types de données et d’objets de ce SGBD.

La transformation du MLD en MPD peut être réalisée de façon automatique par le logiciel de conception, mais il n’est pas rare de devoir apporter ensuite soi-même des modifications au MPD généré.

## Les contraintes d’intégrité

Prenons un exemple de MPD simple modélisant des produits, des commandes et des détails de commandes :



Dans chaque table :

* Les colonnes précédées du symbole PK font partie de la clé primaire (Primary Key)
* Les colonnes précédées du symbole FK font partie de la clé étrangère (Foreign Key)
* Les colonnes précédées du symbole pfK font à la fois partie de la clé primaire et d’une clé étrangère
* Les colonnes précédées du symbole \* sont des colonnes obligatoires, c’est-à-dire qu’elles ne peuvent pas contenir la valeur NULL

FK\_OrderDetails\_XXX sont les noms des clés étrangères

Clé primaire, clé étrangère et caractère non nullable sont des **contraintes d’intégrité**.

**La clé primaire** est une contrainte qui porte sur une seule table et une ou plusieurs colonnes. Elle garantit que chaque enregistrement aura une combinaison de valeurs unique pour les colonnes clé.

**La clé étrangère** est une contrainte qui porte généralement sur deux tables, et une ou plusieurs colonnes de chaque table. Elle garantit que les valeurs de colonnes de clé étrangère de la 1ère table correspondent à des valeurs de colonnes de clé primaire de la 2ème table. Elle permet ainsi d’associer les enregistrements de 2 tables.

Une même colonne peut faire partie à la fois d’une clé primaire et d’une clé étrangère. C’est le cas des colonnes ProductId et OrderId de la table OrderDetails

Voici un extrait du contenu des tables pour mieux comprendre ces notions :

|  |  |
| --- | --- |
| **Table Products** | |
| **ProductId** | **ProductName** |
| 11 | Queso Cabrales |
| 14 | Tofu |
| 41 | Jack's New England Clam Chowder |
| 42 | Singaporean Hokkien Fried Mee |
| 51 | Manjimup Dried Apples |
| 65 | Louisiana Fiery Hot Pepper Sauce |
| 72 | Mozzarella di Giovanni |

|  |  |  |  |
| --- | --- | --- | --- |
| **Table Orders** | | | |
| **OrderID** | **RequiredDate** | **Freight** | **ShipName** |
| 10248 | 1996-08-01 | 32,38 | Vins et alcools Chevalier |
| 10249 | 1996-08-16 | 11,61 | Toms Spezialitäten |
| 10250 | 1996-08-05 | 65,83 | Hanari Carnes |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table OrdersDetails** | | | | |
| **ProductId** | **OrderID** | **UnitPrice** | **Quantity** | **Discount** |
| 11 | 10248 | 14,00 | 12 | 0 |
| 42 | 10248 | 9,80 | 10 | 0 |
| 72 | 10248 | 34,80 | 5 | 0 |
| 14 | 10249 | 18,60 | 9 | 0 |
| 51 | 10249 | 42,40 | 40 | 0 |
| 41 | 10250 | 7,70 | 10 | 0 |
| 51 | 10250 | 42,40 | 35 | 0,15 |
| 65 | 10250 | 16,80 | 15 | 0,15 |

Dans la table OrdersDetails, on voit que :

* Tous les enregistrements ont une valeur unique du couple (ProductId, OrderId). Ceci est imposé par la contrainte de clé primaire de cette table.
* Toutes les valeurs de ProductId se retrouvent bien dans la table Products. Ceci est imposé par la contrainte de clé étrangère FK\_OrderDetails\_Products
* Toutes les valeurs de OrdertId se retrouvent bien dans la table Orders. Ceci est imposé par la contrainte de clé étrangère FK\_OrderDetails\_Orders

Si on essayait d’insérer des valeurs qui ne respectent pas les contraintes d’intégrité, la requête SQL renverrait une erreur.

## Génération du script SQL

Le MPD permet de générer à son tour un script SQL pour créer tous les objets de la base (tables et contraintes d’intégrité).

Voici le code SQL qui peut être généré à partir du MPD précédent :

CREATE TABLE Orders (

OrderId int NOT NULL,

RequiredDate date,

Freight money,

ShipName nvarchar(40)

)

GO

CREATE TABLE OrdersDetails (

OrderId int NOT NULL,

ProductId int NOT NULL,

UnitPrice money,

Quantity smallint NOT NULL,

Discount real NOT NULL

)

GO

CREATE TABLE Products (

ProductId int NOT NULL,

ProductName nvarchar(40) NOT NULL

)

GO

ALTER TABLE Orders ADD CONSTRAINT PK\_Orders

PRIMARY KEY (OrderId)

GO

ALTER TABLE OrdersDetails ADD CONSTRAINT PK\_OrdersDetails

PRIMARY KEY (OrderId, ProductId)

GO

ALTER TABLE Products ADD CONSTRAINT PK\_Products

PRIMARY KEY (ProductId)

GO

ALTER TABLE OrdersDetails ADD CONSTRAINT FK\_OrdersDetails\_Orders

FOREIGN KEY (OrderId) REFERENCES Orders (OrderId)

GO

ALTER TABLE OrdersDetails ADD CONSTRAINT FK\_OrdersDetails\_Products

FOREIGN KEY (ProductId) REFERENCES Products (ProductId)

GO

# Le SGBD SQL Server

Ce chapitre décrit les notions et le vocabulaire nécessaires pour comprendre et utiliser un serveur et des bases de données SQL Server.

Quelques rappels :

**Une donnée** est une information quelconque (exemple nom d’une personne, prix d’un article…)

**Une base de données** est une collection cohérente et structurée de données mémorisées sur un support permanent.

**Un système de gestion de base de données (SGBD)** est un logiciel assurant structuration, stockage, maintenance, mise à jour et consultation des données contenues dans une base de données.

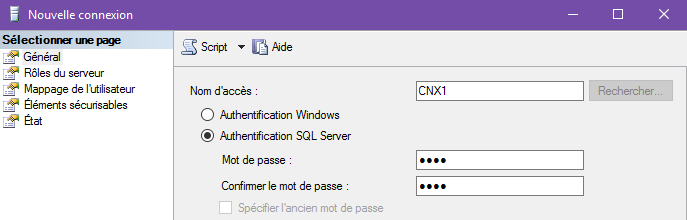
**Une base de données relationnelle** est un système d’information présentant les données sous forme de tables et de relations.

## Connexion

**Une connexion** (objet LOGIN en SQL) est un objet défini au niveau du serveur SQL, qui permet un accès sécurisé au serveur. L’authentification peut être faite :

* Par l’intermédiaire du compte Windows
* Directement auprès du serveur SQL, par la saisie d’un mot de passe

L’image ci-dessous montre la création d’une connexion sur le serveur SQL, avec authentification SQL Server. Il faut donc définir le mot de passe de connexion.



Pour accéder à une base de données, une application doit décrire la connexion qu’elle veut utiliser au moyen d’une chaîne de connexion. En utilisant cette chaîne, elle instancie un objet SqlConnection pour établir la connexion avec le serveur.

La création d’une nouvelle instance de connexion est une opération coûteuse. Pour éviter de la répéter trop souvent, le serveur d’application qui héberge l’application peut gérer un pool de connexions, qui est un ensemble de connexions préétablies, et recyclées.

## Utilisateurs et rôles

**Les utilisateurs et les rôles** (objets USER et ROLE en SQL) sont des entités de sécurité définies au niveau d'une base de données

Un utilisateur Guest et un rôle public sont créés par défaut à la création de la base.

Guest ne peut pas être supprimé, mais on peut lui interdire la consultation de la base.

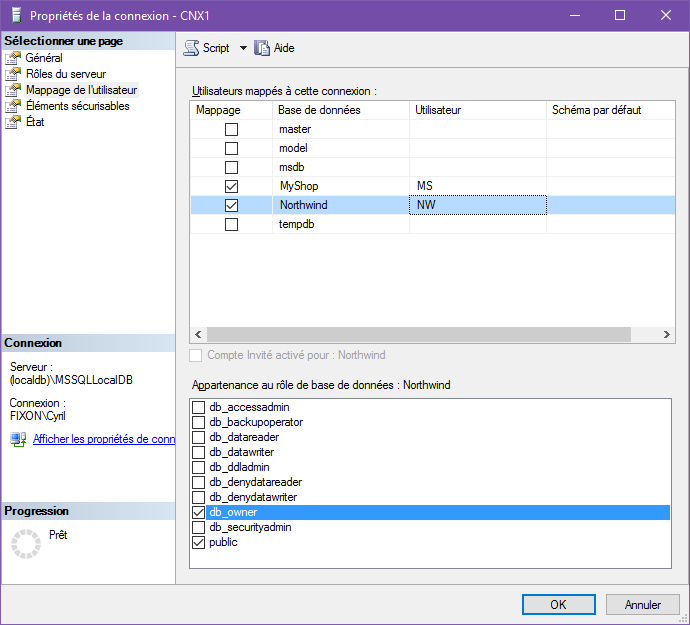
Public est affecté par défaut à tous les utilisateurs de la base de données et permet la consultation des données.

NB/ Les entités INFORMATION\_SCHEMA et sys apparaissent comme des utilisateurs bien qu'elles n'en soient pas vraiment. Ce ne sont pas des entités de sécurité.

## Mappage Connexion - utilisateur

Pour chaque base à laquelle on veut accéder par l'intermédiaire d'une connexion, il faut spécifier un nom d'utilisateur sur lequel la connexion sera mappée.

Ceci peut être réalisé dans la fenêtre de propriétés de la connexion :



Pour chaque mappage, on peut définir les rôles que possédera l'utilisateur sur la base.

L'exemple ci-dessus affiche les propriétés de mappage d’une connexion CNX1. Cette connexion est utilisée pour accéder :

* A la base MyShop par l’intermédiaire de son utilisateur MS
* A la base NorthWind par l’intermédiaire de son utilisateur NW

En outre, on a affecté le rôle db\_owner à l’utilisateur NW pour le désigner comme propriétaire de la base.

Quand on désactive un mappage (en décochant la case à cocher), on supprime automatiquement l'utilisateur mappé.

Pour chaque base, une connexion ne peut être mappée qu'avec un seul utilisateur.

A l'inverse, une base peut avoir plusieurs utilisateurs, mappés sur des connexions différentes.

Si une application accède à une base par l'intermédiaire d'une seule connexion et d'un seul utilisateur, elle peut utiliser plusieurs instances de la même connexion, gérées par un pool de connexions, pour permettre à plusieurs personnes de travailler en même temps sur la base.

Une application peut utiliser plusieurs connexions différentes pour accéder à une même base, selon le type d'opération qu'elle doit réaliser. Par exemple, on peut avoir 2 connexions :

* Une pour l'utilisation courant de l'application
* Une autre mappée avec un utilisateur ayant des droits étendus, pour faire des opérations de maintenance sur la base.

NB/ La notion d'utilisateur au sens SQL (USER) ne doit donc pas être confondue avec celle d'utilisateur d'une application. On peut voir un User comme un moyen d'associer une connexion à une base, en appliquant des droits sur les objets de cette base.



Le schéma ci-dessus illustre les notions présentées précédemment. Il montre :

* Une instance SQL Server gérant 2 bases de données DBX et DBY, et 2 connexions CNXA et CNXB
* Une application nommée MyApp hébergée par un serveur d’application IIS, et utilisant 2 chaînes de connexions pour utiliser les connexions CNXA et CNXB.
  + CNXA est mappée avec les utilisateurs UX1 de la base DBX et UY1 de la base DBY
  + CNXB est mappée avec les utilisateurs UX2 de la base DBX et UY2 de la base DBY

## Schéma

**Un schéma** est équivalent à un espace de noms sur une base de données. Ainsi, deux tables ne peuvent pas avoir le même nom à l'intérieur du même schéma.

Un schéma appartient à un seul utilisateur ou rôle. Cependant, on peut octroyer des droits aux utilisateurs ou rôles sur plusieurs schémas.

Les schémas sys et INFORMATION\_SCHEMA sont réservés pour les objets système. On ne peut pas créer d'objets dans ces schémas et on ne peut pas les supprimer.

Le schéma dbo constitue le schéma par défaut d'une base de données qui vient d'être créée. Il appartient au compte d'utilisateur dbo. Lorsqu’on crée un nouvel utilisateur, son schéma par défaut est dbo si on n’en spécifie aucun.

NB/ Lorsque des objets de base de données sont référencés à l'aide d'un nom sans schéma, SQL Server consulte d'abord le schéma par défaut de l'utilisateur. Si l'objet y est introuvable, SQL Server recherche ensuite dans le schéma dbo. Si l'objet ne s’y trouve pas non plus, une erreur est retournée.