Le modèle physique des données

M164 — Créer des bases de données et y insérer des données

F. Mauron, J. Frossard

EPAI

13 novembre 2023

Plan

Plan

Introduction

Lors du module 162, « Analyser et modéliser des données », nous avons vu :

- Les différentes étapes pour élaborer un **modèle conceptuel des données (MCD)** sous la forme d'un diagramme entité-association.
- Comment le transformer en un **modèle logique des données (MLD)** sous la forme d'un schéma de base de données relationnelle.
- Comment s'assurer que le schéma est normalisé (1NF) et comment réduire la redondance (2NF et 3NF).

Dans le cadre de ce module 164, nous allons voir comment passer du modèle logique à un **modèle physique des données (MPD)** qui peut être interprété par un système de gestion de base de données (SGBD).

Le modèle physique des données

Le modèle physique des données est une description du modèle logique produite pour un SGBD particulier, dans son langage.

Le passage du modèle logique vers un modèle physique est une étape importante dans le développement d'une base de données.

- Le modèle logique devient une réalité opérationnelle.
- En plus des exigences métier, le modèle physique prend en compte des éléments tels les spécificités du SGBD dans lequel doit être créée la base de données, la sécurité, ou encore l'optimisation des performances.

Chaque SGBD (Oracle, MySQL, SQL Server, etc.) a ses spécificités propres. Le passage d'un système à l'autre nécessite généralement des adaptations plus ou moins importantes du modèle physique.

Cycle de vie d'une base de données (I/III)

Analyse et conception

- Le développement commence par une analyse des besoins métier pour produire un **modèle conceptuel des données (MCD)** , par exemple, sous la forme d'un **diagramme entité-association**.
- Le modèle conceptuel des données est réalisé **en collaboration avec des spécialistes du domaine à modéliser** (spécialistes métier) ; jamais par un informaticien seul. L'informaticien est parfois remplacé par un *businesss analyst*.
- Le **modèle logique des données (MLD)** décrit la structure de données sous une forme qui peut être implémentée dans un système informatique, tout en étant indépendante d'un SGBD spécifique.
- Le modèle logique est réalisé par un informaticien. Il s'agit souvent d'un **schéma de base de données relationnelle**.

Cycle de vie d'une base de données (II/III)

Réalisation

- Le modèle logique est transcrit dans un langage compréhensible par le SGBD dans lequel la base de données doit être créée : **le modèle physique des données (MPD)**
- Dans le cas d'un schéma de base de données relationnel, ce langage est le SQL.
- Le modèle physique comprend notamment la spécification :
  - des **options de la base de données** telles que l'**encodage des caractères** et les règles de classement et de tri (*collation*),
  - de la **structure des tables** et de leurs attributs pour le stockage des données,
  - des **contraintes** (type, clé primaire, clé étrangère) pour assurer l'intégrité des données,
  - des **index** pour améliorer les performances des requêtes.

Cycle de vie d'une base de données (III/III)

Déploiement

- Le modèle physique est utilisé pour créer la base de données, les tables, les contraintes, les index, etc., dans le SGBD de production.
- La base de données est mise à disposition des programmes et des utilisateurs en tenant compte de la sécurité.

Exploitation et maintenance

- Après le déploiement, le modèle physique peut encore nécessiter des ajustements pour répondre à des changements dans les exigences du système d'information.
- La mise au point des index est faite en grande partie à partir de statistiques sur des observations récoltées durant l'exécution des requêtes en exploitation.
- L'exploitation est souvent assurée par un administrateur de base de données (DBA). Son travail requiert de solides compétences en système, réseau, et développement.

Plan

Qu’est-ce que le SQL ?

Le SQL (Structured Query Language) est un langage formel conçu pour la gestion et la manipulation de bases de données relationnelles. On l’utilise, typiquement, pour l’analyse de données, la génération de rapports, des sites web, des applications d’entreprise, et bien plus.

Le SQL est basé sur la théorie du modèle relationnel, ce qui en fait un langage puissant et flexible.

Si le schéma d’une base de données est **normalisé** (1FN), alors le SQL permet de réaliser **n’importe quelle requête** sur les données de cette base de données.

En général, les systèmes de gestion de bases de données SQL sont conçus pour prendre en charge de grandes quantités de données.

Un langage, quatre sous-langages

On peut distinguer quatre sous-langages dans le SQL :

- **Langage de description des données (DDL) :** Il est utilisé pour définir la structure des bases de données. Cela inclut la création, la modification et la suppression des bases des données, des tables et des attributs, des contraintes, des règles, etc.
- **Langage de contrôle des données (DCL) :** Il permet de gérer les droits et les privilèges des utilisateurs, pour contrôler l'accès aux données et aux fonctionnalités du SGBD.
- **Langage de manipulation des données (DML) :** Il est spécifiquement conçu l'interaction avec les bases de données et notamment la création, la mise à jour, la suppression et l'interrogation des données.
- **Langage de contrôle de transaction (TCL) :** Il gère les transactions, qui sont des séquences d'opérations effectuées en tant qu'unité unique de travail. Les transactions assurent l'intégrité des données, même en cas de panne.

Le standard SQL

Le SQL est un langage standardisé par l'ANSI (American National Standards Institute) et par l'ISO (International Organization for Standardization).

Avantages de la standardisation :

- **Apprentissage facilité :** Comme la plupart des SGBDR utilisent le SQL, cela infléchit la courbe d'apprentissage (rend l'apprentissage plus facile) lors du passage d'un système à un autre.
- **Compatibilité :** Comme les SGBDR se conforment généralement assez bien au standard SQL, cela facilite le portage des applications d'un système à l'autre.

Inconvénient de la standardisation :

- **Évolution lente :** Comme le standard évolue lentement, chaque éditeur étend le langage à sa manière, ce qui entraîne **certaines incompatibilités**.

Historique du SQL (années 1970)

L'histoire de SQL, le langage standard de gestion de bases de données relationnelles, est à la fois riche et fascinante. Voici un aperçu historique simplifié :

Années 1970 : Les débuts

- En 1970, Edgar F. Codd, un chercheur d'IBM, publie un article intitulé "A Relational Model of Data for Large Shared Data Banks", jetant les bases théoriques des bases de données relationnelles.
- IBM développe un prototype de système de gestion de base de données relationnelle appelé "System R", qui implémente pour la première fois le langage SQL (initialement appelé SEQUEL).

Historique du SQL (années 1980 et 1990)

Années 1980 : Standardisation et expansion

- Le SQL devient une norme de l'ANSI en 1996, puis de l'ISO en 1987.
- De nombreuses sociétés, dont Oracle, IBM, Microsoft et d'autres, adoptent le SQL pour leurs propres systèmes de gestion de bases de données.

Années 1990 : Évolution et maturité

- L'évolution du SQL se poursuit avec des ajouts importants comme le support des transactions, des sous-requêtes, des triggers, et des vues matérialisées.
- En 1992, la publication d'une importante révision du standard, SQL-92 (ou SQL2), élargit considérablement les capacités du langage.

Historique du SQL (années 2000 et 2010)

Années 2000 : SQL et Internet

- Avec l'expansion d'Internet, le SQL joue un rôle clé dans le développement de sites web dynamiques et d'applications en ligne.
- En 2003, la publication de SQL :1999 (ou SQL3) apporte des fonctionnalités orientées objet et des améliorations pour la gestion des données.

Années 2010 : Nouveaux défis et innovations

- Le SQL s'adapte aux nouveaux défis posés par les Big Data et le cloud computing.
- Des extensions et des variantes de SQL sont développées pour répondre aux besoins spécifiques de systèmes comme les bases de données en mémoire, NoSQL, et les entrepôts de données.

Historique du SQL (aujourd'hui et demain)

Aujourd'hui et demain :

- SQL reste le langage standard pour les bases de données relationnelles, étant largement enseigné, utilisé et continuellement développé pour répondre aux besoins modernes de gestion des données.
- L'évolution continue de SQL est caractérisée par l'intégration de nouvelles fonctionnalités pour le traitement analytique, la gestion de données non structurées, et l'intelligence artificielle.

Plan

Qu'est-ce qu'un système de gestion de base de données ?

Un **système de gestion de bases données** (*database management system*) ou SGBD (*DBMS*) est un ensemble de programmes (un système logiciel) qui permettent de gérer des bases de données.

Les principales fonctionnalités d'un SGBD sont :

- Définir la structure de données
- Manipuler (insérer, modifier, supprimer) et interroger les données.
- Assurer la sécurité et l'intégrité des données.
- Contrôler l'accès aux données

En plus des SGBD pour les bases de données relationnelles (SGBDR) sur lesquels nous allons exclusivement nous concentrer dans ce module, il existe également des SGBD dits NoSQL qui seront abordés dans le module 165.

SGBD client-serveur et SGBD embarqué

On peut distinguer deux types de SGBD :

SQGB client-serveur :

- Gestion des données centralisée
- Processus serveur (SGBD) et processus clients (applications).
- Clients locaux (sur la même machine) ou distants (à travers le réseau)
- Prise en charge sécurisé de multiples utilisateurs et de grandes bases de données

SGBD embarqué :

- Gestion des données intégrée directement dans l'application.
- SGBD et application fonctionnent dans le même processus.
- Empreinte mémoire réduite avec des fonctionnalités adaptées.
- Idéal pour des environnements à ressources limitées, tels que l'IoT.

SGBDR client-serveur sous licence libre (open source)

- **MySQL :** largement utilisé pour les applications web. Supporte une large gamme de langages de programmation. Distribué sous licence libre (GPLv2) et commerciale, par Oracle. La licence commerciale donne accès à des fonctionnalités qui ne sont pas disponibles sous licence libre.
- **MariaDB :** Fork de MySQL sous licence libre (GPLv2), créé par le même développeur (Michael Widenius) après l'acquisition de MySQL par Oracle. MariaDB est compatible avec MySQL, mais dispose de fonctionnalités que n'a pas MySQL sous licence libre.
- **PostgreSQL :** Système de gestion de base de données relationnel et objet, sous licence libre (BSD). Réputé pour sa **conformité aux standards SQL** et son extensibilité.

SGBDR client-serveur sous licence propriétaire

- **Oracle Database :** Système de gestion de base de données relationnelle propriétaire, largement utilisé dans les grandes entreprises. Offre des fonctionnalités avancées pour le traitement des transactions et la gestion de grandes bases de données.
- **Microsoft SQL Server (MSSQL) :** Utilisé pour l'analyse et le traitement des données dans l'environnement Windows. Initialement issu d'une collaboration avec la société Sybase (aujourd'hui disparu) pour le portage de son SGBD sous Windows, mais dont l'évolution a par la suite été indépendante.
- **IBM Db2 :** Souvent utilisé sur des mainframes pour des applications transactionnelles avec de grands volumes de données, de hautes performances et une haute sécurité.
- **SAP ASE (Sybase ASE) :** Produit développé par Sybase (acquis par SAP) et aujourd'hui en fin de vie. Utilisé principalement pour des applications transactionnelles à haute performance, notamment dans le secteur de la gestion d'entreprise.

SGBDR embarqués sous licence libre

- **SQLite :** Bibliothèque en langage C permettant d'embarquer un SGBD SQL léger dans une application. Supporte de nombreux langages de programmation en plus du C (C#, C++, Go, Java, JavaScript, PHP, Python, Ruby, etc.).
- **H2 :** SGBDR en mémoire écrit en Java. Il est utilisé dans de nombreuses applications et plateformes, en particulier pour le développement et les tests, en raison de sa vitesse élevée et de sa simplicité de déploiement. Il fonctionne aussi bien comme SGBD embarqué que comme SGBD client-serveur.
- **Derby :** SGBDR entièrement écrit en Java, également connu sous le nom de Java DB. Souvent utilisé pour les applications Java, y compris les applications web et mobiles, en raison de sa facilité d'intégration avec la plateforme Java.

Plan

Création d'une base de données

Avant de décrire le schéma d'une base de données, il est faut créer une base de données. L'instruction ci-dessous crée une base de données nommée « my_database ».

```
CREATE DATABASE my_database;
```

Il est aussi possible de spécifier d'autres options, notamment l'encodage des caractères. Par exemple, pour encoder les caractères en UTF-8 :

```
CREATE DATABASE my_database
    DEFAULT CHARACTER SET = 'utf8mb4';
```

Remarque : En SQL, les instructions se **terminent** par un point-virgule (;) et, par convention, on écrit généralement les mots-clés en majuscule.

Création de tables

Avant de créer les tables, il faut sélectionner la base de données :

```
USE my_database;
```

On peut ensuite utiliser l'instruction ci-dessous pour créer une table nommée « product » avec trois attributs (« product_id », « name », « description »).

```
CREATE TABLE product (
    product_id INT,
    name VARCHAR(100),
    description VARCHAR(2000)
);
```

Le nom de l'attribut est suivi du nom de son type séparé par une ou plusieurs espaces. Les descriptions des attributs sont **séparées** par des virgules (,). Par convention, le nom du type s'écrit en majuscules.

Types de données du SQL (Nombres)

Type de Donnée	MariaDB / MySQL	PostgreSQL	SQL Server
Entier	TINYINT, SMALLINT, INT, BIGINT,	TINYINT, SMALLINT, INT, BIGINT,	TINYINT, SMALLINT, INT, BIGINT,
Virgule flottante - simple précision - double précision	 FLOAT, DOUBLE	 REAL, DOUBLE PRECISION	 REAL, FLOAT
Décimal	DECIMAL, NUMERIC	DECIMAL, NUMERIC	DECIMAL, NUMERIC

Attention : Avec SQL Server, FLOAT est synonyme de DOUBLE PRECISION.

Types de données du SQL (Date et heure)

Type de Donnée	MariaDB / MySQL	PostgreSQL	SQL Server
Date/Heure	DATE, TIME, TIMESTAMP, DATETIME	DATE, TIME, TIMESTAMP, TIMESTAMPTZ, INTERVAL	DATE, TIME, DATETIME, DATETIME2, DATETIMEOFFSET, SMALLDATETIME,

Types de données du SQL (chaines)

Type de Donnée	MariaDB / MySQL	PostgreSQL	SQL Server
Chaîne de caractères	CHAR, VARCHAR, TEXT, NCHAR, NVARCHAR, NTEXT	CHAR, VARCHAR	CHAR, VARCHAR, TEXT, NCHAR, NVARCHAR, NTEXT
Chaîne d'octets	BINARY, VARBINARY, BLOB	BYTEA	BINARY, VARBINARY, IMAGE

Types de données du SQL (autre)

Type de Donnée	MariaDB / MySQL	PostgreSQL	MSSQL
Booléen	BOOLEAN	BOOLEAN	BIT
UUID	CHAR(36) ou BINARY(16)	UUID	UNIQUEIDENTIFIER

Remarque : Un UUID (universally unique identifier) ou GUID (globaly unique identifier) est un standard qui permet de construire un nombre de 128 bits (16 octets) globalement unique sans nécessiter de coordination centralisée. Il est souvent constitué d'une combinaison d'information sur le système, de l'heure, et de données aléatoire. Par exemple : 03640d5e-86e3-11ee-b9d1-0242ac120002.

Contraintes — NOT NULL

On peut ajouter la contrainte NOT NULL à la suite du type dans la description d'un attribut pour indiquer que cet attribut doit obligatoirement avoir une valeur :

```
CREATE TABLE product (
    product_id INT NOT NULL,
    name VARCHAR(100) NOT NULL,
    description VARCHAR(2000)
);
```

Contraintes — Clé primaire

Pour spécifier la contrainte de clé primaire pour une clé composée d'un ou plusieurs attributs, on utilise la syntaxe suivante :

```sql
CREATE TABLE product (
    product_id INT NOT NULL,
    name VARCHAR(100) NOT NULL,
    description VARCHAR(2000),

    PRIMARY KEY (product_id)
);
```

```sql
CREATE TABLE order_product (
    order_id INT NOT NULL,
    product_id INT NOT NULL,
    quantity INT NOT NULL,
    unit_price DECIMAL(10,2) NOT NULL,

    PRIMARY KEY (order_id, product_id)
);
```

Dans l'instruction de gauche, la clé primaire est « product_id ». Dans celle de droite, la clé primaire est le couple « order_id », « product_id ».

Contraintes — Clé étrangère

Pour spécifier une contrainte de clé étrangère, on utilise la syntaxe suivante :

```
CREATE TABLE order_product (
    order_id INT NOT NULL,
    product_id INT NOT NULL,
    quantity INT NOT NULL,
    unit_price DECIMAL(10,2) NOT NULL,

    PRIMARY KEY (order_id, product_id),

    FOREIGN KEY (order_id) REFERENCES order(order_id),
    FOREIGN KEY (product_id) REFERENCES product(product_id)
);
```

La clé étrangère « order_id » référence la clé primaire « order_id » de la table « order ». La clé étrangère « product_id » référence la clé primaire « product_id » de la table « product ».