Introduction aux bases de données relationnelles

Nicolas Casajus

31 mars 2018

Table des matières

1 Base de données relationnelle

Une base de données est un ensemble structuré d'informations réunies sous une même thématique. C'est donc un conteneur permettant de stocker une grande quantité de données. Elle est la pièce centrale d'un système d'information ou d'un système de base de données qui régit la collecte, le stockage, le retraitement et l'utilisation de données. Ce dispositif comporte souvent un logiciel moteur (le système de gestion de bases de données, *SGBD*), des logiciels applicatifs, et un ensemble de règles relatives à l'accès et l'utilisation des informations.

Une base de données présente les caractéristiques suivantes :

— assurer le stockage informatique des données,
— garantir la pérennité des données (même en cas d'une panne technique),
— stocker les données selon leur schéma de structuration,
— garantir la cohérence et l'intégrité des données,
— assurer l'accès simultané aux données,
— contrôler la confidentialité et l'accès aux données.

Le modèle de données (également appelé *schéma de données*) est la description de l'organisation des données. Il se trouve à l'intérieur de la base de données, et renseigne sur les caractéristiques de chaque type de donnée et les relations entre les différentes données qui se trouvent dans la base de données. Il en existe plusieurs types, mais le *modèle relationnel* est très souvent privilégié. Ce modèle est fortement normalisé et s'appuie sur la théorie des ensembles en faisant appel notamment à l'algèbre relationnel.

Selon le modèle relationnel, la base de données est composée d'un ensemble de tables à deux dimensions appelées *relations* dans lesquelles sont placées les données. Les lignes sont appelées des *n-uplets* (ou enregistrements) et sont décrites par des *attributs* (ou champs) placés en colonnes. Dans une base de données relationnelle, chaque enregistrement d'une table contient un groupe d'informations non redondantes relatives à un sujet. Chaque enregistrement sera identifié au moyen d'une *clé primaire*.

Clé primaire : contrainte d'unicité qui permet d'identifier de manière unique un enregistrement dans une table. Une clé primaire peut être composée d'un ou de plusieurs champs de la table. Deux lignes distinctes de la table ne peuvent pas avoir les mêmes valeurs pour les champs définis au niveau de la clé primaire. Il est possible de définir pour une même table plusieurs contraintes d'unicité, mais au plus une seule clé primaire.

Ainsi, n'importe quelle donnée contenue dans la base de données peut être retrouvée à l'aide du nom de la relation, du nom de l'attribut et de la valeur de la clé primaire de cette relation.

2 L'intégrité référentielle

La force du modèle relationnel repose sur le fait que les relations sont reliées entre elles par des liens (associations). Une *clé étrangère* est un attribut d'une relation qui contient une référence à un (ou plusieurs) attribut(s) d'une autre relation. Typiquement, une clé étrangère fait référence à la clé primaire d'une autre relation. La cardinalité d'une association décrit la nature de cette association. Les deux associations les plus couramment rencontrées sont les associations *un-à-un* et *un-à-plusieurs*.

Par exemple, dans la base de données Renards, l'attribut **Numéro de tanière** dans la relation **Description de tanières** est une clé étrangère faisant référence à la clé primaire **Numéro de tanière** de la table **Identité des tanières** selon une association *un-à-plusieurs*. En effet, une tanière peut avoir été décrite plusieurs fois, mais une description ne s'applique qu'à une seule tanière qui doit obligatoirement être présente dans la relation **Identité des tanières**.

L'intégrité référentielle est une situation dans laquelle pour chaque information d'une table A qui fait référence à une information d'une table B, l'information référencée existe dans la table B. L'intégrité référentielle est un gage de cohérence du contenu de la base de données.

Les systèmes de gestion de base de données relationnels (SGBDR) permettent de déclarer des règles de maintien de l'intégrité référentielle sous forme de contraintes. Une fois la contrainte déclarée, le SGBDR refusera toute modification du contenu (insertion d'enregistrement, mise à jour, etc.) de la base de données qui violerait la règle en question et casserait l'intégrité référentielle.

Ainsi, il sera impossible d'insérer la description d'une tanière X dans la relation du même nom si cette tanière est absente de la relation **Identité des tanières** (étant donné que les deux relations sont liées entre elles selon une cardinalité *un-à-plusieurs*).

3 La cohérence des données

Outre l'intégrité référentielle, la cohérence des données intervient également à d'autres niveaux, mais toujours sous forme de contraintes déclarées. Une contrainte d'intégrité est une clause [1] permettant de contraindre la modification de tables, faite par l'intermédiaire de requêtes d'utilisateurs, afin que les données saisies dans la base soient conformes aux données attendues.

Contrainte de domaine : chaque attribut dans une table est défini dans un domaine particulier (on parle aussi de type de données) : parmi les plus fréquents, on rencontre les données de type Nombres entiers, Nombres décimaux, Caractères, Date, Time, Texte, etc. De même, il est possible de restreindre ce domaine de définition en rajoutant des conditions. Par exemple, on pourrait imposer qu'un nombre entier ne soit jamais négatif (longueur, poids, etc.).

Contrainte d'unicité : la clause SQL `UNIQUE` permet de vérifier que la valeur saisie pour un champ n'existe pas déjà dans la table. Cela permet de garantir que toutes les valeurs d'une colonne d'une table sont différentes.

1. En SQL, une clause est un mot clé réservé dont l'appel entrainera une action bien spécifique. Par ex., la clause `INSERT INTO` permet d'insérer un enregistrement dans une table et la clause `SELECT` permet d'extraire certains enregistrements.

Contrainte d'intégrité d'entité : cette contrainte permet de s'assurer que chaque relation contienne bien une clé primaire, permettant ainsi d'identifier de manière unique chaque enregistrement de cette relation.

La clause *NOT NULL* : celle-ci permet de s'assurer qu'une colonne ne contient pas de valeurs nulles (absentes). L'enregistrement ne pourra pas être inséré si aucune valeur n'est spécifiée pour cet attribut.

4 Le langage SQL

Le langage SQL (acronyme de *Structured Query Language*) est un langage informatique normalisé servant à exploiter des bases de données relationnelles. Il fut créé en 1974, puis normalisé à partir de 1986. C'est un langage déclaratif, c'est-à-dire qu'il permet de décrire le résultat escompté sans décrire la manière de l'obtenir et ses instructions s'écrivent d'une manière qui ressemble à celle de phrases ordinaires en anglais.

Les instructions SQL peuvent être classées en quatre grands domaines d'application :

Langage de définition de données (LDD) : c'est un langage orienté au niveau de la structure de la base de données, c'est-à-dire que les commandes manipulent les structures de données et non les données elles-mêmes. C'est ce langage qui permet de définir la structure des tables et des attributs, leurs domaines de définition et les associations entre relations.

Langage de manipulation de données (LMD) : ce langage regroupe des commandes logiques opérant la manipulation des données (lectures et écritures). Les opérations permettant l'insertion, l'extraction, la suppression d'enregistrements, et les mises à jour de données appartiennent à ce langage.

Langage de contrôle de données (LCD) : ce sous-ensemble du SQL permet de contrôler l'accès aux données d'une base de données en définissant par exemple le rôle qu'aura un utilisateur particulier (lecture seule, modification et insertion de données, etc.).

Langage de contrôle des transactions (LCT) : il assure le contrôle transactionnel dans une base de données, c'est-à-dire les caractéristiques des transactions, la validation et l'annulation des modifications [2].

Le langage SQL peut s'utiliser de trois manières différentes :

— à partir de l'interface de programmation du SGBDR ou en connectant des interfaces graphiques tierces (si le SGBDR en est dépourvues);
— grâce à la technique dite du *embedded SQL* dont le principe est d'avoir des instructions SQL incorporées dans le code source d'un programme écrit dans un autre langage (par ex. les langages R et PHP);
— via des procédures SQL stockées dans la base de données qui seront exécutées par le SGBDR. Cette technique est utilisée par les *triggers*, procédures déclenchées automatiquement sur modification du contenu de la base de données.

5 Le système PostgreSQL

PostgreSQL est un système de gestion de base de données relationnelle et objet (SGBDRO). C'est un outil multiplateforme et libre disponible selon les termes de la licence BSD, qui n'est pas contrôlé par une seule entreprise, mais est fondé sur une communauté mondiale de développeurs et d'entreprises. PostgreSQL est plus avancé que ses concurrents dans la conformité aux standards SQL et utilise des types de données modernes, dit enrichis. Il est largement reconnu pour son comportement stable et offre des possibilités de programmation étendues, directement dans le moteur de la base de données, via son langage procédural (PL/pgSQL). Enfin, il comprend un module spatial (PostGIS) qui permet de manipuler des bases de données géographiques.

2. Ce domaine d'utilisation du SQL n'est pas utilisé pour la base de données Renards

Contrairement à Microsoft Access et autres systèmes à base de fichiers, PostgreSQL fonctionne selon un mode client/serveur (noté mode C/S). Alors que dans un système à base de fichiers chaque poste de travail traite localement les données (chacun à une copie du fichier), dans un SGBDR en mode C/S, tous les traitements sont, en principe, effectués sur le serveur par le biais de requêtes utilisateurs. Ainsi, l'ensemble des techniques du C/S est dédié à assurer une plus grande intégrité lors du traitement des données. Notons que le serveur peut être à distance et centralisé ou hébergé localement sur un poste de travail [3]. Pour interroger le serveur, l'utilisateur aura le choix entre plusieurs interfaces d'utilisation :

- **psql:** l'interface en ligne de commande de PostgreSQL,
- **pgAdmin:** outil d'administration graphique basique pour PostgreSQL,
- **phpPgAdmin:** interface web d'administration recommandée pour les utilisateurs qui développent des sites Web avec base de données,
- **Logiciel R:** grâce au package `RPostgreSQL`, il est très facile de se connecter à un serveur PostgreSQL et d'envoyer des requêtes selon la technique du *embedded SQL* (on enveloppe du code SQL dans du code R),
- **Microsoft Access:** idéal pour développer des formulaires de saisie (uniquement pour Windows),
- **LibreOffice Base:** idéal pour développer des formulaires de saisie (multiplateforme).

Seule l'interface en ligne de commande **psql** accompagne l'installation de PostgreSQL, les autres outils doivent être installés et configurés séparément.

Travailler avec R présente l'avantage de réaliser l'ensemble du flux de travail sous un seul logiciel (R), logiciel qui est un langage de scripts (on garde une trace écrite de toutes les étapes réalisées). Libre à vous de choisir l'outil le plus adapté à vos besoins.

3. Dans le cas d'une base de données sollicitée par différents utilisateurs, il est vivement recommandé de la centraliser sur un serveur indépendant des postes de travail individuels.