Introduction aux bases de données relationnelles

ACADYS

Objectifs pédagogiques

À la fin de ce module, vous serez capable de :

- Expliquer le concept de base de données relationnelle
- Identifier les composants clés d'une base de données
- Comprendre la terminologie SQL fondamentale
- Décrire la structure générale d'une requête SQL

1. Qu'est-ce qu'une base de données relationnelle ?

1.1 Définition

Une base de données relationnelle est un type de base de données qui organise les données en tables (relations) reliées entre elles par des relations logiques. Cette organisation permet de stocker, d'organiser et de manipuler efficacement les données.

Les principales caractéristiques d'une base de données relationnelle sont :

- Organisation des données en tables (lignes et colonnes)
- Relations définies entre les tables
- Utilisation de clés pour établir ces relations
- Langage SQL standardisé pour l'interrogation et la manipulation

1.2 Avantages des bases de données relationnelles

- **Intégrité des données** : Des règles peuvent être définies pour garantir la cohérence des données.
- **Flexibilité** : Les données peuvent être interrogées de manière flexible selon différents critères.
- **Évolutivité** : La structure peut évoluer sans nécessairement impacter les applications existantes.
- **Sécurité** : Contrôle d'accès granulaire aux données.
- **Standardisation** : Le langage SQL est largement standardisé.

2. Terminologie essentielle

2.1 Structure d'une base de données relationnelle

| Terme | Définition | |-------|------------| | **Base de données** | Ensemble structuré de données organisées en tables | | **Table** | Collection de données organisées en lignes et colonnes (également appelée "relation") | | **Colonne** | Attribut spécifique d'une table ayant un type de données défini (également appelé "champ") | | **Ligne** | Ensemble de valeurs correspondant à un enregistrement unique dans une table (également appelé "tuple" ou "enregistrement") | | **Clé primaire** | Colonne ou ensemble de colonnes qui identifie de manière unique chaque ligne dans une table | | **Clé étrangère** | Colonne qui établit une relation avec la clé primaire d'une autre table | | **Index** | Structure qui améliore la vitesse de récupération des données |

2.2 Exemple visuel

Voici un exemple simple de deux tables reliées dans une base de données relationnelle :

Table : Clients

| client_id | nom | prenom | email | |-----------|----------|----------|----------------------| | 1 | Dupont | Jean | jean.dupont@email.com | | 2 | Martin | Sophie | sophie.martin@email.com| | 3 | Bernard | Thomas | thomas.b@email.com |

Table : Commandes

| commande*id* / *client*id | date_commande | montant | |-------------|-----------|--------------|---------| | 101 | 1 | 2023-01-15 | 120.50 | | 102 | 3 | 2023-01-16 | 75.00 | | 103 | 1 | 2023-01-20 | 45.80 |

Dans cet exemple, `client_id` est la clé primaire de la table Clients, et elle est utilisée comme clé étrangère dans la table Commandes pour établir une relation entre les deux tables.

3. Introduction au langage SQL

3.1 Qu'est-ce que SQL ?

SQL (Structured Query Language) est un langage standardisé pour communiquer avec les bases de données relationnelles. Il permet de :

- Interroger des données (requêtes)
- Insérer, mettre à jour et supprimer des données
- Créer et modifier la structure des tables
- Définir des contraintes d'intégrité
- Gérer les accès aux données

3.2 Catégories d'instructions SQL

- **DDL (Data Definition Language)** : CREATE, ALTER, DROP, TRUNCATE
- **DML (Data Manipulation Language)** : SELECT, INSERT, UPDATE, DELETE
- **DCL (Data Control Language)** : GRANT, REVOKE
- **TCL (Transaction Control Language)** : COMMIT, ROLLBACK, SAVEPOINT

Dans ce cours de niveau 1, nous nous concentrerons principalement sur le DML, et tout spécialement sur les requêtes SELECT pour extraire des données. Les instructions plus avancées comme GROUP BY et HAVING seront abordées au niveau 2.

4. Structure de base d'une requête SQL

4.1 La requête SELECT simple

La structure de base d'une requête SQL pour récupérer des données est :

```
SELECT colonne1, colonne2, ...
FROM nom_table
WHERE condition;
```

Où :

- `SELECT` indique les colonnes à récupérer
- `FROM` spécifie la table source des données
- `WHERE` (optionnel) définit les conditions de filtrage

4.2 Exemple simple

```
-- Récupérer le nom et le prénom de tous les clients
SELECT nom, prenom
FROM Clients;

-- Récupérer toutes les informations du client dont l'id est 1
SELECT *
FROM Clients
WHERE client_id = 1;
```

4.3 Conventions d'écriture

Bien que SQL ne soit pas sensible à la casse pour les mots-clés, il est de bonne pratique de :

- Écrire les mots-clés SQL en MAJUSCULES
- Mettre chaque clause principale sur une nouvelle ligne
- Utiliser l'indentation pour les clauses secondaires
- Terminer les requêtes par un point-virgule (;)

Exercices pratiques

Exercice 1

En utilisant la table Clients présentée dans ce module, écrivez une requête SQL pour récupérer l'email de tous les clients.

Exercice 2

En utilisant la table Commandes, écrivez une requête SQL pour récupérer toutes les commandes dont le montant est supérieur à 100.

Exercice 3

Identifiez dans les tables suivantes les clés primaires et les clés étrangères :

Table : Produits | produit*id* / *nom*produit | prix | categorie_id | |------------|-------------|------|--------------| | 1 | Ordinateur | 899 | 10 | | 2 | Souris | 25 | 10 | | 3 | Cahier | 5 | 20 |

Table : Categories | categorie*id* / *nom*categorie | |--------------|--------------| | 10 | Informatique | | 20 | Papeterie | | 30 | Mobilier |

Lecture complémentaire

- "Introduction aux bases de données" (chapitre 1-3)
- "Fondamentaux SQL" (documentation en ligne du SGBD utilisé)

Prochaine étape

Dans le prochain module, nous approfondirons l'utilisation des instructions SELECT et FROM pour effectuer des requêtes simples sur une seule table.