Chap 02. Bases de données – SQL

Livre p 309 – Chap 19 Requêtes SQL et mises à jour
Livre p 329 – Chap 20 Systèmes de Gestion de Bases de Données

1. Création d'une base de données

a. Définition des tables

En SQL, toutes les lignes de commande doivent se terminer par un point-virgule « ; ».
Les différentes tables sont créées par la commande :

CREATE TABLE *nom_table_1* (*nom_attribut_1 domaine_1 contrainte_1,*
…, *nom_attribut_n domaine_n contrainte_n*) ;

Remarque : On peut ajouter des espaces et même passer à la ligne pour une meilleure lisibilité.
Attention : Si une table existe déjà, il faut d'abord la supprimer avec la commande :

DROP TABLE *nom_table* ;

Attention : On ne peut pas supprimer une table servant de référence pour des clefs étrangères…

b. Types de données

On trouve de nombreux domaines possibles pour les attributs…

- des données numériques :
  INT : Nombre entier sur 32 bits (signé).
  DECIMAL(*p, s*) : Nombre décimal ayant *p* chiffres significatifs, dont *s* chiffres après la virgule.
  FLOAT(*n*) : Nombre à virgule flottante ayant une mantisse codée sur *n* bits (entre 1 et 53).
  REAL : Nombre à virgule flottante sur 32 bits, équivalent à FLOAT(24).
- des chaînes de caractères :
  CHAR(*n*) : Texte de longueur fixe : *n* caractères (complété éventuellement par des espaces).
  VARCHAR(*n*) : Texte de longueur maximale *n* caractères.
  TEXT : Texte de taille quelconque (maximum 65 535 caractères).
- d'autres types de données :
  DATE : Date au format AAAA-MM-JJ.
  TIME : Heure au format HH-MM:SS.
  DATETIME : Date et heure au format AAAA-MM-JJ HH-MM:SS (équivalent à TIMESTAMP).
  BOOLEAN : TRUE, FALSE ou UNKNOW (Attention : format non standard).

c. Contraintes d'intégrité

Des contraintes d'intégrité peuvent être spécifiées après le domaine, par exemple :

UNIQUE : Interdit à deux entités différentes d'avoir le même contenu pour un même attribut.
NON NULL : Oblige un attribut à être rempli lors de l'enregistrement.

On précise généralement à la fin de la création la clef primaire et les clefs étrangères éventuelles.

PRIMARY KEY : Clef primaire (peut être composée de plusieurs attributs).
FOREIGN KEY : Clef étrangère (doit faire référence à la clef primaire d'une autre table).

Exemple : FOREIGN KEY (Id_client) REFERENCES Client(Id_client)

On peut aussi ajouter des contraintes utilisateur avec la commande CHECK en indiquant des tests logiques à effectuer avec les opérateurs logiques (AND, OR, NOT, IN, =, <, >, <=, >=, != ou <>)

Exemple : CHECK (note >= 0 AND note <= 20) ;

d. Insertion dans une table

Pour ajouter un ou plusieurs enregistrements dans une table on utilise la commande :

INSERT INTO *nom_table* VALUES (*att_1_valeur_1, att_2_valeur_1*, …, *att_n_valeur_1*),
, …, (*att_n_valeur_k, att_n_valeur_k*, …, *att_n_valeur_k*) ;

Remarque : Si on ne précise pas d'ordre, les attributs seront dans le même ordre que lors de leur création.

2. Requêtes SQL (Structured Query Language)

a. Extraction de données

On peut lister facilement tous les enregistrements d'un ou de plusieurs attributs d'une table :

SELECT *nom_attribut_1 ... nom_attribut_n* FROM *nom_table* ;

Remarque : le caractère « * » permet de sélectionner tous les attributs.

Remarque : On peut aussi ajouter des paramètres ou utiliser une fonction d'agrégation…

COUNT(*nom_attribut*) : pour compter le nombre d'enregistrements.
MIN(*nom_attribut*) : pour afficher le minimum ou le 1er dans l'ordre alphabétique.
MAX(*nom_attribut*) : pour afficher le maximum ou le dernier dans l'ordre alphabétique.
DISTINCT *nom_attribut* : pour supprimer les doublons à l'affichage.
ORDER BY *nom_attribut* ASC : pour afficher les données dans l'ordre croissant
ORDER BY *nom_attribut* DESC : pour afficher les données dans l'ordre décroissant.

Exemples : SELECT COUNT(DISTINCT nom) FROM eleves ;
SELECT nom, prenom FROM eleves ORDER BY nom ; (ASC par défaut)

De plus, si les données sont des nombres, on peut effectuer des calculs, par exemple…

SUM(*nom_attribut*) : pour calculer la somme des données.
AVG(*nom_attribut*) : pour calculer la moyenne des données.

b. Utilisation d'une ou plusieurs clauses

On peut préciser des critères avec la commande WHERE suivie de tests logiques à effectuer sur des attributs avec les opérateurs classiques (AND, OR, NOT, IN, =, <, >, <=, >=, != ou <>)

Exemple : SELECT nom, prenom FROM eleves WHERE date_naissance = 2004 ;

De plus, l'opérateur LIKE permet d'effectuer une « recherche floue » en l'associant à « % » ou « _ »

Exemples : SELECT nom, prenom FROM eleves WHERE nom LIKE "De %" ;
N'affichera que les élèves dont le nom commence par "De "
SELECT nom, prenom FROM eleves WHERE prenom LIKE "P_ _ _ _ _" ;
N'affichera que les élèves dont le prénom commence par "P" et possède 6 lettres

c. Modification des données

On peut ajouter un nouvel attribut avec la commande ALTER TABLE … ADD COLUMN …

Exemple : ALTER TABLE eleves ADD COLUMN age INT ;

On peut modifier un ou plusieurs enregistrements avec la commande UPDATE … SET …

Exemple : UPDATE eleves SET prenom = "Jean-Phi." WHERE prenom = "Jean-Philippe" ;

On peut même en effacer avec la commande DELETE FROM ….

Exemple : DELETE FROM eleves WHERE date_naissance < 2004 ;

Attention : On ne peut pas effacer un enregistrement qui est utilisé dans une clef étrangère !

d. Jointure

Lorsque l'on veut récupérer des enregistrements en croisant plusieurs tables, on doit réaliser une jointure. Il existe plusieurs sortes de jointures, nous n'étudierons que la plus simple nommée INNER JOIN, ou plus simplement JOIN, elle permet de réaliser l'intersection entre deux tables en utilisant un critère de liaison qui indique quel attribut de la 1ère table correspond à quel attribut de la 2ème table.

Exemple :
```
SELECT nom, prenom, solde FROM eleves
    JOIN cantine ON eleves.id_eleve = cantine.id_eleve
        WHERE solde < 0 ;
```

Remarque : Pour une meilleure lisibilité, il est conseillé d'indiquer pour chaque attribut la table qui lui correspond, on peut pour cela utiliser un alias (diminutif) avec la commande AS

Exemple :
```
SELECT e.nom, e.prenom, c.solde FROM eleves AS e
    JOIN cantine AS c ON e.id_eleve = c.id_eleve
        WHERE c.solde < 0 ;
```