# SQL : Module 1 : Introduction et premières requêtes


Site: LMS UCA 2020/2021
Cours: EIIN512B - ECUE Bases de donnees relationnelles
Livre: SQL : Module 1 : Introduction et premières requêtes

Imprimé par: niget Tom
Date: samedi 28 août 2021, 15:00

# Table des matières

# 1. Introduction

Le langage SQL (Structured Query Language) a fait l'objet de plusieurs normalisations successives

•Normes SQL-92 (SQL2)
•Normes SQL-99 (SQL3)
•Normes SQL-2011

Mais il n'est pas interdit d'avoir des ajouts, voir des variantes

Il s'en suit l'existence de differents dialectes (oracle database, postgresql, mysql, mariaDB,sqlite…)

L'ensemble des possibilités offertes par SQL est usuellement décomposé en 4 parties :

- Langage de définition de données : CREATE, ALTER, DROP, RENAME
- Langage de manipulation de données:INSERT, UPDATE, DELETE, SELECT
- Langage de contrôle de données :GRANT, REVOKE
- Langage de contrôle des transactions: SET TRANSACTION, COMMIT, ROLLBACK

SQL est fondé sur l'algèbre relationnelle: toute requête que l'on peut écrire en algèbre relationnelle peut être écrite en SQL.

SQL manipule des **tables** et non des **relations.**  Les colonnes des tables correspondent aux attributs, les lignes aux tuples

Mais la table n'est pas un ensemble: il peut y avoir plusieurs lignes identiques dans une table

# 2. Manipulation de données

SELECT est la commande SQL qui permet de consulter les données des tables d'une base de données relationnelle.
Le résultat d’une commande SELECT est zéro ou plusieurs lignes, avec éventuellement des répétitions.
**Une commande SELECT décrit un jeu de résultat voulus, et non la manière de les obtenir.**
Le système de gestion de base de données transforme la requête en un plan d’exécution de requête, qui peut dépendre du serveur utilisé.

Le SELECT permet l'execution de plusieurs opérateurs de l'algèbre relationnelle:

- sélection
- jointure
- Renommer des attributs
- Projection
- Mais aussi
  - ordonner
  - Renommer temporairement
  - Partitionner ( on oublie pour aujourd'hui)

# 2.1. SELECT affichage

Dans sa version la plus simple SELECT permet d'afficher le contenu d'une table

Exemple

**select * from marque;**

affiche toutes les lignes de la table marque

Dans une telle requête ni l'ordre d'affichage des colonnes, ni l'ordre d'affichage des lignes n'est spécifié.  Deux versions différentes de SQL peuvent ordonner différemment

Pour spécifier l'ordre d'affichage des attributs, il faut les énumérer à la place du *. Seuls les attributs faisant partie de l'enumeration, seront affichés. Le remplacement de * par une énumeration d'attribut est donc une projection

Pour spécifier l'ordre d'affichage des lignes il faut compléter la requête par un order by en spécifiant une liste d'attributs sur lesquels trier les lignes

Exemple la relation marque a été définie par

```
marque (
id INT NOT NULL PRIMARY KEY ,
nom VARCHAR (30) ,
classe INT  ,
pays CHAR (2)  ,
prop INT
);
```

**SELECT  nom, classe from marque order by id;**

affichera les deux colonnes nom et classe , par ordre de IdMcroissant  (on peut trier sur des attributs non affichés)

Enfin l'affichage comporte un entête avec les noms des attributs. Il est possible de les renommer

**SELECT nom as nomMarque, classe as classeMarque from marque order by id;**

**Remarque :** le order by n'a pas d'équivalent en algèbre relationnelle , mais la requête **SELECT nom as nomMarque, classe as classeMarque from marque** est équivalente à $\delta_{nom \leftarrow nomMarque,\ classe \leftarrow classeMarque}(\Pi_{nom,\ classe}(marque))$

Et maintenant cliquez ici pour un peu de pratique

# 2.2. SELECT avec plusieurs tables

SELECT permet aussi de combiner des tuples issus de plusieurs tables.

Exemple

```
marque (
id INT  PRIMARY KEY ,
nom VARCHAR (30)  ,
classe INT  ,
pays CHAR (2)  ,
prop INT
);

enr (
marque INT  ,
num INT L ,
pays CHAR (2)  ,
deposant INT ,
date_enr DATE  ,
CONSTRAINT cle_enr PRIMARY KEY (num , pays )
);
```

Pour faire le **produit cartesien** de ces deux tables il suffit d'écrire

**select * from marque, enr;**

Remarque : A  l'affichage deux colonnes seront intitulées pays. Le produit cartesien SQL contrairement au produit cartesien de l'algèbre relationnel n'impose pas des noms différents pour les attributs.  En fait les deux colonnes peuvent être distinguées, l'une d'entre elle est marque.pays, l'autre enr.pays

Chacune des tables ayant 5 attributs, le résultat a 10 attributs

Pour faire la jointure naturelle de ces tables il suffit d'écrire

**select * from marque natural join enr;**

cette requete est equivalente à marque ⋈ enr

Ici un tuple de marque est join a un tuple de enr si ils ont la même valeur sur des attributs de même nom, pays ici, cette jointure naturelle produit donc des lignes avec toutes les informations sur une marque et un enregistrement pourvu que la marque et l'enregistrement aient le même pays. Les lignes du résultat ont 9 colonnes

SQL permet une **jointure** plus génerale où l'on peut dire sur quels attributs on souhaite que la jointure se fasse

select * from marque **join** enr **on** (marque.id=enr.marque);

joint un tuple de marque et tuple de enr sur l'unique condition marque.id=enr.marque. Les tuples résultants ont 10 attributs

Il est aussi possible d'utiler le mot clé JOIN conjointement au mot clé USING si l'on veut joindre sur des attributs de même nom

select marque **join** enr **using** (pays) est equivalent à select marque natural join enr, la nuance n'existe que si les deux tables ont plusieurs attributs de même nom, cette syntaxe permet de specifier un sous ensemble des attributs de même nom

Allez répondre à la premiere question de ce test pour verifier si vous avez compris

# 2.3. Filtrage des lignes

**SELECT** permet aussi le filtrage des lignes ( en algèbre relationnelle c'est la selection) en ajoutant la clause **WHERE**

exemple

SELECT id from marque where nom="Sprite"

est l'equivalent $\Pi\ i_d\ (\sigma_{nom="Sprite"}(marque))$

Il est tout à fait possible (mais pas recommandé) de se passer du mot clé JOIN

SELECT listeattributs from r JOIN S on (r.A1=s.A2)  peut aussi s'ecrire

SELECT listeattributs from r, s WHERE r.A1=s.A2

De même qu'en algèbre relationnelle il est possible de se passer de la jointure naturelle en la remplaçant par un produit cartesien suivi d'un filtrage

Sans le WHERE, nn peut y utiliser :

- Tous les attributs de la table construite dans le FROM, qu'ils soient affichés ou non
- Les opérateurs logiques: AND, OR, NOT
- Tous les opérateurs compatibles avec le type des attributs, par exemple : Comparateurs: =, <>, <, >, <=, >= Ø ,Opérateurs arithmétiques: +, *, … Concaténations de chaînes: 'foo'||'bar' a pour valeur 'foobar'
- et c'est pas tout….mais c'est tout pour aujourd'hui

Il est temps de passer aux questions 2 et 3 du test en cours : si vous n'avez pas fermé la fenêtre vous pouvez y retourner directement sans cliquer sur le lien du début de cette phrase.