Cours de Base de Données
Cours n.4
Le langage SQL (partie I)

Ce cours reprend beaucoup de transparents du cours de Philippe LAHIRE

Elisabetta De Maria - edemaria@i3s.unice.fr

UFR Sciences et Laboratoire I3S, CNRS

2017-2018

Université de Nice Sophia-Antipolis

Introduction

Objectifs de SQL

- Créer la structure de la base de données et de ses table
- Exécuter les tâches de base de la gestion des données, telle que l'insertion, la modification et la suppression de données des tables
- Effectuer des requêtes simples ou complexes

Langage orienté transformation

Origine

Structured **Q**uery **L**anguage

- caractéristiques des langages déclaratifs
- origine : IBM, System R, milieu des années 70
- implémenté dans de nombreux SGBD

Plusieurs versions :

- SQL1 initial : ANSI* —1986
- SQL1 avec intégrité référentielle, ANSI —1989
- SQL2 ANSI —1992
- SQL3 ANSI — 1999 incorpore la notion d'objet

** ANSI = American National Standard Institute*

Caractéristiques de SQL

- Fonctionnalités :
  - Définition des objets de la base de données (LDD)
  - Manipulation de données (LMD)
  - Contrôle des accès aux données
  - Gestion de transactions

- Utilisé par : DBA, développeurs, quelques utilisateurs

Parallèle avec mySQL

Principales Instructions

- Définitions (LDD)

  CREATE, DROP, ALTER

- Mises à jour (LMD)

  INSERT, UPDATE , DELETE

- Interrogations (LMD)

  SELECT

- Contrôle d'accés aux données

  GRANT, REVOKE

- Gestion de transactions

  COMMIT, ROLLBACK

Consultation des données

- Hypothèse:
  - un schéma de base de données est créé
  - Une base de données a été remplie
- La création a été faite par une interface QBE (mySQL)
- On expérimente la consultation avec SQL
- On expérimentera la création du schéma et le remplissage de la base avec SQL plus tard

⇨ **Langage algébrique en SQL**

⇨ **Requêtes mono et multi table(s)**

Exemple pour les requêtes

CLIENT

numéro	nom	adresse	numéro_téléphone
101	Durand	NICE	0493456743
108	Fabre	PARIS	NULL
110	Prosper	PARIS	NULL
125	Antonin	MARSEILLE	NULL

PRODUIT

référence	marque	Prix HT
153	BMW	1000
589	PEUGEOT	1800
158	TOYOTA	1500

VENTE

numéro	référence_produit	numéro_client	date
00102	153	101	12/10/04
00809	589	108	20/01/05
11005	158	108	15/03/05
12005	589	125	30/03/05

Format des requêtes

- SELECT
- FROM
- WHERE
- GROUP BY
- HAVING
- ORDER BY

- FROM spécifie la table ou les tables à utiliser
- WHERE filtre les lignes selon une condition donnée
- GROUP BY forme des groupes de lignes de même valeur de colonne
- HAVING filtre les groupes sujets à une certaine condition
- SELECT spécifie les colonnes qui doivent apparaître dans les résultats
- ORDER BY spécifie l'ordre d'apparition des données dans le résultat

Requêtes simples (SELECT-FROM)

- Afficher le nom et l'adresse des clients

```
SELECT nom , adresse
FROM  Client ;
```

- Afficher toutes les informations des clients

```
SELECT *
FROM  Client ;
```

Elimination des doublons

- Afficher l'adresse de tous les clients

SELECT *adresse*
FROM *Client* ;

SELECT ALL *adresse*
FROM *Client*;

- Afficher toutes les adresses existantes (sans doublons)

SELECT DISTINCT *adresse*
FROM *Client* ;

Par défaut

Projection

Relationnel : On doit pouvoir distinguer chaque tuple !

SQL : le résultat peut ne pas être une relation

Sélection de colonne (clause WHERE)

Les conditions fondamentales de recherche

- *comparaison*
  (salaire>10000, ville='Paris')
- *étendue* ou *intervalle*
  (salaire BETWEEN 20000 and 30000)
- *appartenance à un ensemble*
  (couleur IN ('red', 'vert'))
- *correspondence à un masque*
  (adresse LIKE '%Montréal%')
- *nul*
  (adresse IS NULL)

Opérateur Sélection

- Quels sont les clients dont l'adresse est *Paris*

```
SELECT *
FROM Client
WHERE adresse = 'Paris';
```

- Quels sont les produits dont le prix TTC est supérieur à *1000*

```
SELECT *
FROM Produit
WHERE prix_HT + prix_HT * 0.195 > 1000;
```

Opérations possibles (mySQL)

- Booléennes

  and, or, xor, =, !=, <, >, <=, >=

- Arithmétiques

  +, -, *, /

  +, - *opérateurs unaires*

- Fonctions numériques

  abs, log, cos, sin, mod, power ...

- Arithmétiques sur date

  +, -

- Fonctions sur chaînes

  length, concat, ...

Un grand nombre

Dans Select: attribut calculé (résultat)

Dans Where: participer à la sélection

Précédence des opérateurs

- :=
- ||, OR, XOR
- &&, AND
- BETWEEN, CASE, WHEN, THEN, ELSE
- =, <=>, >=, >, <=, <, <>, !=, IS, LIKE, REGEXP, IN
- |
- &
- <<, >>
- -, +
- *, /, DIV, %, MOD
- ^
- - (unary minus), ~ (unary bit inversion)
- !, NOT
- BINARY, COLLATE

+ faible

+ forte

```
SELECT 1+2*3;
7
```

Utilisation des opérateurs

- **SELECT** ABS (-32); 32
- **SELECT** FLOOR (1.23); 1
- **SELECT** MOD (234, 10); 4
- **SELECT 253** % 7; 1
- **SELECT** ROUND (1.298, 1); 1,3
- **SELECT** ROUND (1.298, 0); 1
- **SELECT** SIGN (234) SIGN (-32) SIGN (0); 1 / -1 / 0
- **SELECT** 3 / 5; 0,60

Bien sur on peut utiliser des attributs

Peut se trouver dans Where

WHERE 3 / 5 < 1;

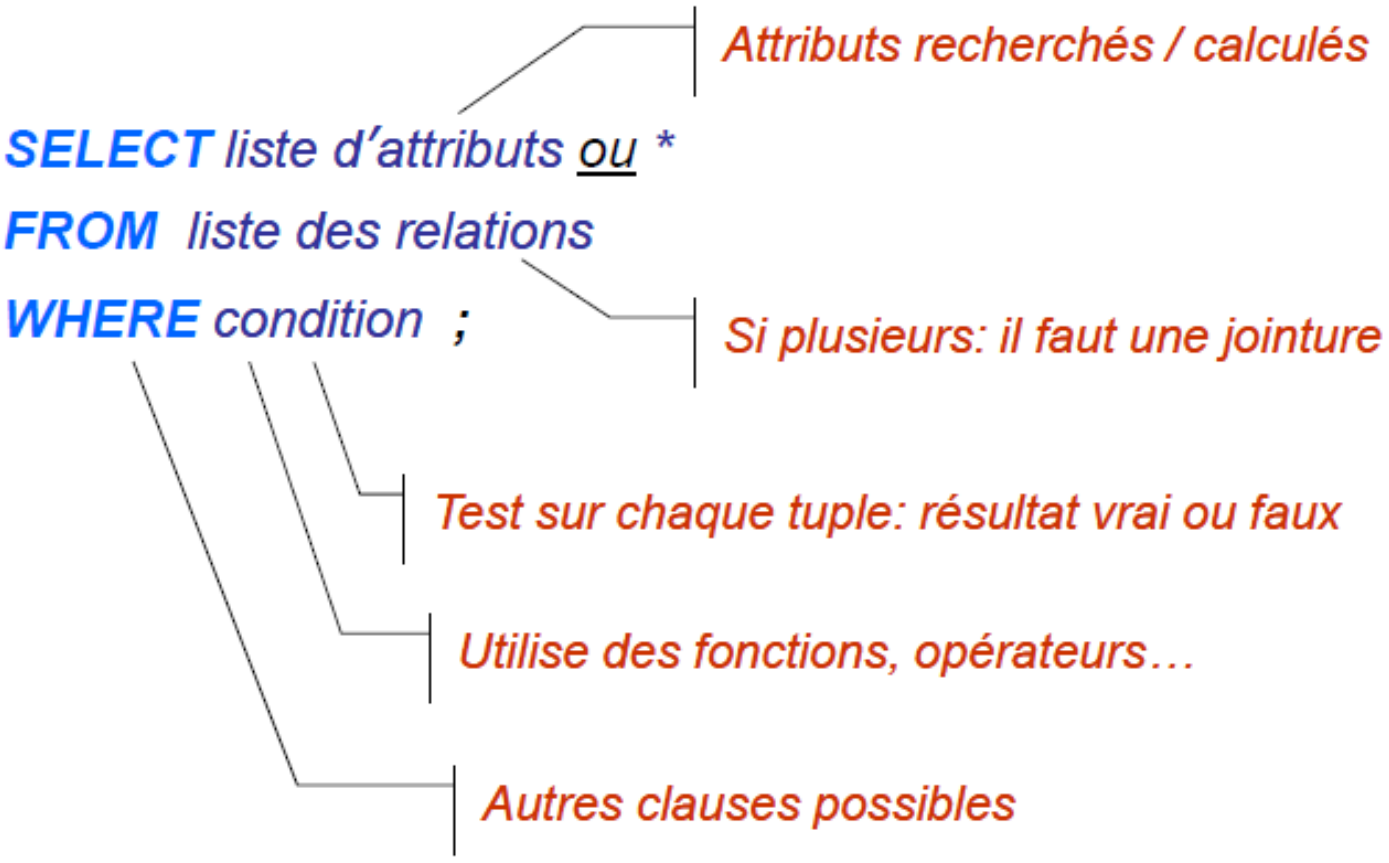
Utilisation des opérateurs (Chaînes)

- **SELECT** CONCAT ('*My*', '*S*', '*QL*'); ‘MySQL’
- **SELECT** CHAR_LENGTH (‘*MySQL*’); 5
- **SELECT** LOCATE ('bar', 'foobarbar'); 4
- **SELECT** LOCATE ('bar', 'foobarbar', 5); 7
- **SELECT** INSERT ('Quadratic', 3, 4, 'What'); 'QuWhattic'
- **SELECT** LOWER (‘*MySQL*’); ‘mysql’
- **SELECT** SUBSTRING ('Quadratically',5,6); ‘ratica'
- **SELECT** 'David!' LIKE ‘David_'; 1
- **SELECT** 'David!' LIKE ‘%D%v%'; 1
- **SELECT** STRCMP (S1, S2); -1,0,1

_ et %

0 (false), **1** (true)

Combinaison *Sélection + projection*



Requêtes simples (1)

- *Y a-t-il des produits dont le nom est « XBOX »*

```
SELECT *
FROM  Produit
WHERE nom = 'XBOX' ;
```

- *Quels sont les ventes réalisés il y a plus de 30 jours?*

```
SELECT *
FROM  Vente
WHERE CURRENT_DATE () > 30 + date ;
```

- *Quels sont les ventes faites après le 1er janvier 2007?*

```
SELECT *
FROM  Vente
WHERE date > DATE ('2007-01-01') ;
```

Requêtes simples (2)

- *Quels sont les ventes dont le montant HT est entre 1000 et 3000 euros et dont le client n'est pas le numéro 101?*

```
SELECT *
FROM  Vente
WHERE  (prix_ht between 1000 and 3000) and
       (numero != 101);
```

- *Quels sont les clients dont le nom est soit Prosper, soit Durand, soit Anthonin ?*

```
SELECT *
FROM  Client
WHERE  nom in ('Prosper', 'Durand', 'Anthonin') ;
```

Requêtes simples (3)

- *Quels sont les clients dont le nom commence par 'P'*

```
SELECT *
FROM  Employe
WHERE  nom LIKE 'P%' ;
```

- *Quels sont les clients dont le nom commence par 'P' et a un 'S' comme 4ème lettre*

```
SELECT *
FROM  Client
WHERE  nom LIKE  'P__S%' ;
```

Requêtes et valeurs nulles

• Quels sont les ventes dont la date de réalisation est inconnue?

```
SELECT *
FROM  Vente
WHERE  date is null ;
```

Ou : IS NOT NULL

Attention :

WHERE date = NULL ⇨ *Whatever* comparé avec NULL
→ Ni vrai ni faux

COUNT, MIN, SUM ⇨ *Ignore les valeurs* NULL
→ sauf COUNT (*)

Toute opération appliquée à NULL donne pour résultat NULL

Les autres clauses (tri)

```
SELECT attribut1 ..
FROM
WHERE expression
ORDER BY attribut1 [ASC] [DESC]…
```

Affichage!

- Donner le numero, le prix HT et la marque des produits selon l'ordre décroissant des marques et l'ordre croissant des prix HT

```
SELECT marque, prix_ht, numero
FROM Produit
ORDER BY marque DESC, prix_ht ASC;
```

Ordre d'affichage!

Mais aussi:

```
ORDER BY 1 DESC, 2 ASC;
```

Requêtes multi- tables (Opérateur *Jointure*)

Deux points de vue:

- exécuter des **boucles imbriquées** (une table par boucle)
  appliquer la clause WHERE dans les boucles
- calculer le **produit cartésien** (une nouvelle table)
  Appliquer la clause WHERE sur chaque ligne

Donner le nom d'un produit et le montant de la vente

```
SELECT Produit.nom, Vente.prix_HT
FROM  Produit , Vente
WHERE Vente.reference_produit = Produit.numero ;
```

Critères de jointure

Il faut joindre les tables (jointure)

Définition de la jointure

SELECT *Client.nom, Vente.prix_ht*

FROM *Client, Vente ;*

→ tous les tuples (*nom*, *prix_ht*)

→ nom est un nom de client et prix_ht est un prix de vente

Pas de critère de jointure → Produit cartésien

Intérêt?

Mots-clés pour exprimer le critère de sélection

Jointure (compléments)

- *Donner la marque des produits dont le prix HT est supérieur à celui d'une BMW*

Renommage

```
SELECT Produit.marque
FROM Produit  Produit_reference, Produit
WHERE Produit_reference.marque = 'BMW' AND
        Produit.prix_HT > Produit_reference.prix_HT ;
```

Ou *de même prix HT*:

```
SELECT Produit.marque
FROM  Produit  Produit_reference, Produit
WHERE Produit_reference.marque = 'BMW' AND
        Produit.prix_ht = Produit_reference.prix_ht AND
        Produit.marque != Produit_reference.marque ;
```

Quelques jointures (compléments)

Que deviennent les tuples non sélectionnés de R1 ou R2?

- dans la jointure (interne ***INNER*** **JOIN**) les tuples qui ne peuvent pas être joints sont éliminés du résultat
- dans la jointure externe (**OUTER JOIN**) les tuples qui ne peuvent pas être joints sont conservés dans le résultat

Pour les tuples de la relation de gauche (R1) et / ou de droite (R2)

Défaut

R1 **FULL OUTER JOIN** R2 : *Remplit R1.* et R2.**

R1 **LEFT OUTER JOIN** R2 : *Remplit R1.**

R1 **RIGHT OUTER JOIN** R2 : *Remplit R2.**

avec NULL si nécessaire.

Exercice (1)

Relations :

- Journal (code-j, titre, prix, type, périodicité)
- Dépôt (no-dépôt, nom-dépôt, adresse)
- Livraison (no-dépôt, code-j, date-liv, quantité-livrée)

Requêtes : donner…

• le prix des journaux livrés le 15/01/07 ?
• tous le nom des hebdomadaires reçus par le dépôt de Paris..
• les titre des journaux livrés à Nice.
• le nom des dépôts qui reçoivent des hebdomadaires dont la quantité livrée excède 100.

Exercice (2)

```
SELECT *
FROM LIVRAISON RIGHT JOIN (DEPOT, JOURNAL)
ON (LIVRAISON.no-depot = DEPOT.no-depot AND
    LIVRAISON.code-j = JOURNAL.code-j)
```

Toutes les lignes de DEPOT et JOURNAL seront présentes
Avec éventuellement rien pour la partie livraison

```
SELECT *
FROM LIVRAISON, DEPOT, JOURNAL
WHERE LIVRAISON.no-depot = DEPOT.no-depot AND
      LIVRAISON.code-j = JOURNAL.code-j)
```

Seulement les lignes de DEPOT et JOURNAL qui correspondent à une livraison

Opérateur *Union*

Afficher la liste des numéros d'employé des responsables de département et des directeurs

Département

numéro-département	...	responsable

Employé

numéro-employé	...	fonction

```
SELECT responsable
FROM Département
UNION
SELECT numéro-employé
FROM Employé
WHERE fonction = 'Directeur';
```

Opérateurs *Intersection, Différence*

Afficher les numéros d'employé des responsables de département qui sont aussi des directeurs

```
SELECT responsable
FROM Département
INTERSECT
SELECT numéro-employé
FROM Employé
WHERE fonction = 'Directeur';
```

```
SELECT responsable
FROM Département
EXCEPT
SELECT numéro-employé
FROM Employé
WHERE fonction = 'Directeur';
```

Afficher les numéros d'employé des responsables de département Sauf ceux qui sont aussi des directeurs

Fonctions d'agrégat

- SUM (*nom d'attribut*)
- COUNT(*)
- COUNT(DISTINCT *nom d'attribut*)
- MAX (*nom d'attribut*)
- MIN (*nom d'attribut*)
- AVG (*nom d'attribut*)
- AVG (DISTINCT *nom d'attribut*)

Dans les clauses :
- SELECT
- HAVING TO

Clause Group by

SELECT *attributs recherchés*
FROM *liste des relations*
WHERE *condition*
GROUP BY *attributs de regroupement*
[**HAVING** *condition sur le groupe*];

```
SELECT COUNT (*)
FROM  Produit
GROUP BY marque
```

```
SELECT AVG (prix_ht), nom
FROM  Produit
GROUP BY marque
```

```
SELECT COUNT (*)
FROM  Produit
WHERE marque <> 'BMW'
GROUP BY marque
HAVING AVG (prix_ht) > 10.5
```

Utilisation des fonctions statistiques

- *Donner la moyenne des prix HT et les prix min. et max.*

SELECT AVG *(prix_ht)*, MIN *(prix_ht)*, MAX *(prix_ht)*
FROM *Produit*

- *Même chose mais par marque*

SELECT AVG *(prix_ht)*, MIN *(prix_ht)*, MAX *(prix_ht)*
FROM *Produit*
GROUP BY *marque*

- Même chose mais par marque si la moyenne > 10,5

SELECT AVG *(prix_ht)*, MIN *(prix_ht)*, MAX *(prix_ht)*
FROM *Produit*
GROUP BY *marque* **HAVING** AVG *(prix_ht) > 10.5*