Séléction dans une base de données

M1 Statistique

E. Claeys

ICUBE/IRMA
Université de Strasbourg

Base de données, 2018

Programme du cours

1. `SELECT`
2. `WHERE`
3. Sous-reqûetes
4. Quizz
5. TD

Rappel

- Commande pour créer une base de donnée ?
- Commande pour créer une table ?
- Afficher tous les tuples d'une table ?

Rappel

Création d'une base

```
CREATE DATABASE nom_base;
CREATE DATABASE nom_base CARACTER SET 'utf8';
```

Création d'une table

```
CREATE TABLE nom_base;
```

Rappel

Suppression d'une base

```
DROP DATABASE nom_base;
```

Suppression d'une table

```
DROP TABLE nom_table;
```

Rappel

Afficher tous les tuples d'une table

```
SELECT * FROM nom_table
```

SELECT

SELECT est la commande la plus riche du langage SQL. Le résultat d'un ordre SELECT est une relation. Par exemple :

```
SELECT * FROM client;
WHERE etp_id = 12
AND clt_localite = 'PARIS'
```

Si l'on souhaite un nombre restreint d'attributs :

```
SELECT clt_nom, clt_localite, etp_nom FROM client;
WHERE etp_id = 12
AND clt_localite = 'PARIS'
```

SELECT

La commande `SELECT` comporte au moins deux clauses :

- `SELECT` : La liste des attributs que l'on souhaite extraire de la base de données.
- `FROM` : Le nom de la ou des tables que l'on désire interroger. La commande `SELECT` peut être complémentée par la clause facultative `WHERE` qui permet de définir les conditions à remplir pour la sélection des données dans la base

WHERE

La clause `WHERE` suivie d'une ou plusieurs conditions est appliquée à chaque ligne de la table et retourne un résultat VRAI ou FAUX. Seules les lignes pour lesquelles le résultat est VRAI sont sélectionnées. Les conditions peuvent être combinées à l'aide des opérateurs logiques `AND` et `OR`. `NOT` peut être utilisé avant `AND` ou `OR` pour exprimer la négation.

WHERE

Les conditions de la clause `WHERE` peuvent être regroupées en :

- conditions de comparaisons
- conditions de jointures
- conditions de sous-requêtes

Exemple :

```
SELECT * FROM produit
WHERE (prix > 200 AND qte <= 20)
OR (date_com BETWEEN '20-MAR-96' AND '21-DEC-96');
```

WHERE

L'expression logique qui suit la clause `WHERE` porte le nom de prédicat. Les expressions qui servent à construire ce prédicat (l'/les opérande(s)) peuvent être :

- un nom de colonne, une constante numérique ou alphabétique,
- une pseudo-colonne, une valeur nulle ou une combinaison de ces éléments.

Les opérateurs arithmétiques sont : + - * /
On peut également faire appel aux fonctions de date, de conversion de données, etc. ...

Conditions de comparaison

Elles permettent de comparer une colonne ou une expression à une autre colonne ou expression. Les opérateurs de comparaison sont :

- $=$ égal
- $!=, <>$ différent de
- $>=$ supérieur ou égal à
- $>$ supérieur à
- $<$ inférieur à
- $<=$ inférieur ou égal à

Autres opérateurs de comparaison :

- `[NOT] BETWEEN` expr1 `BETWEEN` expr2 `AND` expr3
  → Vrai si expr1 est compris entre expr2 et expr3
- `[NOT] IN` expr1 `IN` (expr2, expr3, ...)
  → Vrai si expr1 est égale à l'une des expressions qui suivent
- `[NOT] LIKE` expr1 `LIKE` chaîne_de_caractères
  → Vrai si expr1 est contenu dans une chaîne de caractères
- `IS [NOT] NULL` expr1 `IS NULL`
  → Vrai si expr1 est nulle

Sélections complexes

Lorsque vous utilisez plusieurs critères et que vous devez donc combiner plusieurs opérateurs logiques, il est extrêmement important de bien structurer la requête. En particulier, il faut placer des parenthèses au bon endroit.
Petit exemple simple :

```
SELECT * FROM Produit
WHERE color='rouge' AND color='vert' OR color='blue'
```

Qu'accepte-t-on ?

- Ce qui est rouge et vert, et ce qui est bleu ?
- Ce qui est rouge, et soit vert, soit bleu ?

Sélections complexes

```
SELECT * FROM Produit
WHERE color='rouge' AND color='vert' OR color='blue'
```

Qu'accepte-t-on ?

- Ce qui est rouge et vert, et ce qui est bleu ?
- Ce qui est rouge, et soit vert, soit bleu ?

`AND` est appliqué avant `OR`.

Pour éviter ce genre de soucis, il faut utiliser des parenthèses pour préciser les opérations à réaliser en priorités.

Sélections complexes

Exemple

Je dispose de la table suivante :

```
CREATE TABLE Employee (
    Id int NOT NULL AUTO_INCREMENT,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age TINYINT UNSIGNED,
    Sexe char(1),
    City varchar(255),
    Commission int UNSIGNED,
    Salary int UNSIGNED,
    PRIMARY KEY (ID)
);
```

Je voudrais les employé qui sont soit de Strasbourg, soit des employés au salaire supérieur à 2000 euros, mais si employés au salaire supérieur à 3000 euros, ils doivent être nées après juin 1985.

Sélections complexes

Je dispose de la table suivante :

```
CREATE TABLE Employee (
    Id int NOT NULL AUTO_INCREMENT,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age TINYINT UNSIGNED,
    Sexe char(1),
    City varchar(255),
    Commission int UNSIGNED,
    Salary int UNSIGNED,
    PRIMARY KEY (ID)
);
```

Je cherche les noms et prénoms des employés qui :

- soit de Strasbourg ;
- 2000 $<$ salaire $<$3000 et salaire $>$ 3000 si nés après juin 1985.

Je voudrais les noms et prénoms des employés qui sont, soit de Strasbourg, soit des employés au salaire supérieur à 2000 euros, mais si employés au salaire supérieur à 3000 euros, ils doivent être nés après juin 1985.

Sélections complexes

Je dispose de la table suivante :

```
CREATE TABLE Employee (
    Id int NOT NULL AUTO_INCREMENT,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age TINYINT UNSIGNED,
    Sexe char(1),
    City varchar(255),
    Commission int UNSIGNED,
    Salary int UNSIGNED,
    PRIMARY KEY (ID)
);
```

Je cherche les noms et prénoms des employés qui :

- soit de Strasbourg ;
- salaire
  - 2000 $<$ salaire $<$3000
  - salaire $>$ 3000 ET nés après juin 1985.

Je voudrais les noms et prénoms des employés qui sont, soit de Strasbourg, soit des employés au salaire supérieur à 2000 euros, mais si employés au salaire supérieur à 3000 euros, ils doivent être nés après juin 1985.

Sélections complexes

Je dispose de la table suivante :

```
CREATE TABLE Employee (
    Id int NOT NULL AUTO_INCREMENT,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age TINYINT UNSIGNED,
    Sexe char(1),
    City varchar(255),
    Commission int UNSIGNED,
    Salary int UNSIGNED,
    PRIMARY KEY (ID)
);
```

Je cherche les noms et prénoms des employés qui :

- soit de Strasbourg ;
- salaire
  - $2000 < \text{salaire} \leq 3000$
  - salaire > 3000 ET nés après juin 1985.

```
SELECT LastName, FirstName FROM Employee
WHERE
  city = 'Strasbourg'
  OR
  (
    (salary BETWEEN 2000 AND 3000)
    OR
    (
       salary > 3000
       AND
       date_naissance < '1985-06-01'
    )
   );
```

Sélections complexes

Je dispose de la table suivante :

```
CREATE TABLE Employee (
    Id int NOT NULL AUTO_INCREMENT,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age TINYINT UNSIGNED,
    Sexe char(1),
    City varchar(255),
    Commission int UNSIGNED,
    Salary int UNSIGNED,
    PRIMARY KEY (ID)
);
```

Je voudrais les noms et prénoms des employés du sexe féminin. Dans le cas des femmes de Strasbourg, je souhaite avoir uniquement celles dont la commission est supérieur à leurs salaire.

Sélections complexes

Je dispose de la table suivante :

```
CREATE TABLE Employee (
    Id int NOT NULL AUTO_INCREMENT,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age TINYINT UNSIGNED,
    Sexe char(1),
    City varchar(255),
    Commission int UNSIGNED,
    Salary int UNSIGNED,
    PRIMARY KEY (ID)
);
```

Je cherche les noms et prénoms des employés qui :

- sont des femmes ;
- si femmes de Strasbourg, la commission est supérieur à leurs salaire.

Je voudrais les noms et prénoms des employés du sexe féminin. Dans le cas des femmes de Strasbourg, je souhaite avoir uniquement celles dont la commission est supérieur à leurs salaire.

Sélections complexes

Je dispose de la table suivante :

```
CREATE TABLE Employee (
    Id int NOT NULL AUTO_INCREMENT,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age TINYINT UNSIGNED,
    Sexe char(1),
    City varchar(255),
    Commission int UNSIGNED,
    Salary int UNSIGNED,
    PRIMARY KEY (ID)
);
```

Je cherche les noms et prénoms des employés qui :

- sont des femmes hors de Strasbourg ;
- sont des femmes de Strasbourg ET dont la commission est supérieur à leurs salaire.

Je voudrais les noms et prénoms des employés du sexe féminin. Dans le cas des femmes de Strasbourg, je souhaite avoir uniquement celles dont la commission est supérieur à leurs salaire.

Sélections complexes

Je dispose de la table suivante :

```
CREATE TABLE Employee (
    Id int NOT NULL AUTO_INCREMENT,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age TINYINT UNSIGNED,
    Sexe char(1),
    City varchar(255),
    Commission int UNSIGNED,
    Salary int UNSIGNED,
    PRIMARY KEY (ID)
);
```

Je cherche les noms et prénoms des employés qui :

- sont des femmes hors de Strasbourg ;
- sont des femmes de Strasbourg ET dont la commission est supérieur à leurs salaire.

```
SELECT LastName, FirstName FROM Employee
WHERE
  ( Sexe = 'F' AND City <> 'Strasbourg')
  OR
  ( Sexe = 'F'
    AND
    City = 'Strasbourg'
    AND
    Commission > Salary
   );
```

Quizz

Quizz

`WHERE` est considéré comme (plusieurs réponses possibles) :

- Un prédicat.
- Suivi d'un ou plusieurs prédicats.
- Suivi de deux opérandes.
- Suivi d'une opérande.

Quizz

`WHERE` est considéré comme (plusieurs réponses possibles) :

- Un prédicat.
- Suivi d'un ou plusieurs prédicats.
- Suivi de deux opérandes.
- Suivi d'une opérande.

Quizz

`AND` est :

- Appliqué avant `OR`
- Appliqué après `OR`
- Appliqué avant `OR` s'il est énonce avant
- Appliqué après `OR` s'il est énonce avant

Quizz

`AND` est :

- Appliqué avant `OR`
- Appliqué après `OR`
- Appliqué avant `OR` s'il est énonce avant
- Appliqué après `OR` s'il est énonce avant

Quizz

La clause `WHERE` :

- Retourne VRAI ou FAUX
- Retourne VRAI ou FAUX pour chaque tuple
- Retourne une table
- Retourne des tuples vérifiant une condition

Quizz

La clause `WHERE` :

- Retourne VRAI ou FAUX
- Retourne VRAI ou FAUX pour chaque tuple
- Retourne une table
- Retourne des tuples vérifiant une condition

Quizz

Les parenthèses permettent (plusieurs réponses possibles)

- de faire joli
- de lister les attributs à renvoyer
- de fixer les requêtes prioritaires
- de faciliter l'écriture d'une requête

Quizz

Les parenthèses permettent (plusieurs réponses possibles)

- de faire joli
- de lister les attributs à renvoyer
- de fixer les requêtes prioritaires
- de faciliter l'écriture d'une requête

Quizz

`SELECT * renvoi`

- Tous les tuples d'une base sans conditions
- Les tuples avec un caractère * dans au moins un attribut
- Tous les tuples d'une table sans conditions
- Rien

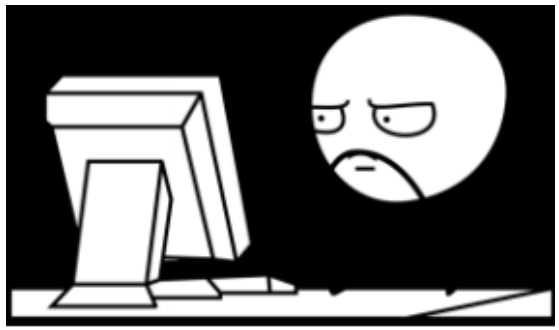
Quizz

`SELECT * renvoi`

- Tous les tuples d'une base sans conditions
- Les tuples avec un caractère * dans au moins un attribut
- Tous les tuples d'une table sans conditions
- Rien

TD n° 0

TD n° 4 : Sélection de données avec MySQL (30/50 minutes)



Pensez à sauvgarder vos commandes !