SQL - Cours 3
Le langage SQL (Structured Query Language)

Ikbel GUIDARA

ikbel.guidara@univ-lyon1.fr
7/11/2017

SQL

- Langage de manipulation de données (LMD)
  - Interrogation/consultation de données
  - Mise à jour des données (insertion, modification et suppression des données)
- Langage de définition des données (LDD)
  - Création des schémas de tables
- Langage d'administration des données
  - Création des utilisateurs, gestion des droits d'accès, etc.
- Implémenté dans plusieurs SGBD

Objectifs de SQL

- Définition des données des bases de données
  - Création de la base de données
  - Modification de la base de données
  - Suppression de la base de données
- Manipulation des données
  - Insertion des données
  - Mise à jour
  - Suppression des données
  - Interrogation
- Contrôle des accès aux données

Objectifs de SQL

- Définition des données
  - CREATE, DROP, ALTER
- Mise à jour des données
  - INSERT, UPDATE, DELETE
- Interrogation
  - SELECT

Interrogation

- SQL définit six clauses pour l'interrogation
  - SELECT (obligatoire)
  - FROM (obligatoire)
  - WHERE
  - ORDER BY
  - GROUP BY
  - HAVING

Projection, Sélection

SELECT (Projection)

- Sélection simple SELECT Colonne FROM Table

Affichage de toute la table

```
SELECT * FROM Table
```

```
SELECT * FROM Personnel
```

Projection sur quelques colonnes

```
SELECT Colonne1, Colonne2 FROM Table
```

```
SELECT nom, prénom, grade FROM Personnel
```

SELECT (Projection)

Eviter les doublons: supprimer les lignes identique

```
SELECT DISTINCT Colonne1 FROM Table
```

```
SELECT DISTINCT NE FROM Obtenu
```

Renommer les colonnes

```
SELECT Colonne1 as col1, Colonne2 as col2 FROM Table
```

```
SELECT NE as numetudiant, nomC FROM Obtenu
```

Projection étendue: implication des calculs dans les projections

```
SELECT Colonne1, Colonne2 FROM Table
```

```
SELECT NE, nomC, note*1.2, annee FROM Obtenu
```

SELECT, WHERE (Sélection)

- Sélection simple avec condition

```
SELECT Colonnes FROM Table WHERE Conditions
```

```
SELECT nom, prénom FROM Personnel WHERE
grade='MCF'
```

- Les conditions peuvent être:
  - Elémentaires: comparaison, liste, intervalle, null, chaine de caractères
  - Complexes: conditions élémentaires reliées par des opérateurs logiques **and**, **or**,..

SELECT, WHERE (Sélection)

- Les conditions élémentaires

Comparaison

Colonne operateur valeur

grade = ‘MCF’
adr != ‘paris’

Liste

Colonne IN (val1, val2, val3)

ville IN (‘paris’, ‘londres’, ‘lyon’)
marque IN ('BMW' , ‘Renault’)

Intervalle

Colonne between|not between val1 and val2

age between 15 and 25
age not between 15 and 20

Null

Colonne is NULL|is NOT NULL

telephone IS NULL
note IS NOT NULL

Récupérer les colonnes qui n’ont pas de valeurs

Récupérer les colonnes qui ont des valeurs

SELECT, WHERE (Sélection)

- Les conditions élémentaires

Chaine de caractères

Colonne like|not like valeur

2 caractères spéciaux :

% désigne une chaine de caractères quelconque y compris la chaine vide

_ désigne un et un seul caractère y compris le caractère blanc

nomC **like** '%S' ==> les noms de cours se terminant par S

nomC **like** 'a__o' ==> les noms de cours à 4 caractères commençant par a et se terminant par o

nomC **not like** 'a%' ==> les noms de cours ne commençant pas par a

```
SELECT * FROM cours WHERE nomC like 'a%'
```

SELECT, WHERE (Sélection)

- Les conditions complexes

Chaine de caractères

SELECT Colonnes FROM Table WHERE Condition1 operateur Condition2…

Opérateurs logiques:

And, Or, Xor

Age> 25 **and** ville in (‘paris’, ‘lyon’, ‘londres’)

Age> 25 **or** ville in (‘paris’, ‘lyon’, ‘londres’)

SELECT * **FROM** cours **WHERE** cycle = 2 **and** nomC='BD';

Les fonctions d'agrégation

Les fonctions d'agrégation

- Ce sont des fonctions qui opèrent sur une colonne pour en calculer la somme, la max, le min …

- Il existe 5 fonctions
  - SUM(col) : renvoie la somme des valeurs de col
  - Count(col) : renvoie le nombre de valeurs de col
  - AVG(col) : renvoie la moyenne des valeurs de col
  - MAX(col) : renvoie la maximum des valeurs de col
  - MIN(col) : renvoie le minimum des valeurs de col

- Ces fonctions ignorent les NULL

Les fonctions d'agrégation

- Nombre d'étudiants
  - SELECT count(NP) FROM etudiant
  - SELECT count(*) FROM etudiant
  - SELECT count(NP) as nombre_total FROM etudiant
- Nombre d'étudiants ayant obtenu des modules
  - SELECT count(distinct NE) as nombreEtudiants FROM obtenu
- La meilleure note en algo
  - SELECT max(note) as maxalgo FROM obtenu WHERE nomC='algo'
- La meilleure et la mauvaise note d'algo
  - SELECT max(note) as maxalgo, min(note) as minalgo FROM obtenu WHERE nomC='algo'

ORDER BY
GROUP BY
HAVING

ORDER BY (Tri)

- Permet d'ordonner le résultat en utilisant un ou plusieurs attributs

- Tout résultat peut être trié (ordonné) sur une ou plusieurs colonnes par ordre croissant ou décroissant

```
SELECT Colonnes FROM Table ORDER BY col1 asc|desc, col2 asc|desc
```

```
SELECT * FROM personne ORDER BY adr asc
SELECT * FROM personne ORDER BY adr asc, nom desc
```

GROUP BY (Groupement)

- Permet de subdiviser la table en groupes ayant une valeur commune
- Les lignes qui partagent les mêmes valeurs seront dans le même groupe
- Exemple: le nombre d'étudiants par age, le nombre d'enseignants par département, la moyenne des notes par étudiant, les étudiants qui suivent le même module, les personnes qui habitent la même ville…

```
SELECT Colonnes FROM Table GROUP BY Colonne1
```

```
SELECT  age, count(*) as nbre_etudiants FROM Etudiant GROUP BY age
```

HAVING (Groupement conditionnel)

- Permet d'identifier quels groupes (avec GROUP BY) doivent être sélectionnés (filtrés)
- Il s'agit de pouvoir générer des groupes et de n'en garder que certains, ceux qui vérifient une condition
- La condition du groupe peut utiliser une fonction d'agrégation et/ou les colonnes du group by
- Exemple: les clients dont le total de leurs différentes réservations dépasse 1000€

```
SELECT Colonnes FROM Table GROUP BY col1, col2 HAVING
condition_du_groupe
```

```
SELECT idClient, SUM(tarif) FROM Reservation GROUP BY
idClient HAVING SUM(tarif)>1000
```

Intersection, Union, Différence
Produit cartésien
Jointure

INTERSECT, UNION, MINUS

- Même nombre de colonnes et meme types

Intersection

```
SELECT Colonnes FROM Table WHERE Conditions
      INTERSECT
SELECT Colonnes FROM Table WHERE Conditions
```

Union

```
SELECT Colonnes FROM Table WHERE Conditions
      UNION
SELECT Colonnes FROM Table WHERE Conditions
```

Différence

```
SELECT Colonnes FROM Table WHERE Conditions
      MINUS
SELECT Colonnes FROM Table WHERE Conditions
```

Produit cartésien, Jointure

- Requêtes multi-tables: interroger plusieurs tables pour répondre à une requête

Produit cartésien

```
SELECT * FROM Table1, Table2,
Table3
```

```
SELECT * FROM personne, enseignant
```

Jointure

```
SELECT * FROM Table1, Table2,
Table3 WHERE Conditions
```

```
SELECT * FROM personne, enseignant
WHERE personne.NP = enseignant.NP
```

Produit cartésien, Jointure

Jointure + Projection

```
SELECT Colonne1, Colonne2 FROM Table1, Table2, Table3 WHERE Conditions
```

```
SELECT Personne.NP, nom FROM personne, enseignant WHERE personne.NP = enseignant.NP
```

Jointure + Projection + Sélection

```
SELECT Colonne1, Colonne2 FROM Table1, Table2, Table3 WHERE Condition1 and Condition2
```

```
SELECT Personne.NP, nom FROM personne, enseignant WHERE personne.NP = enseignant.NP and enseignant.NP=2
```

Produit cartésien, Jointure

- Jointure de deux tables avec des attributs portant le même nom

Alias de tables:
Renommer les tables

```
SELECT Colonnes FROM Table1 T1, Table2 T2 WHERE
Conditions
```

```
SELECT P.NP, nom FROM personne P, enseignant  E
WHERE P.NP = E.NP and E.NP=3
```

Auto-jointure : jointure d'une table avec sa copie

```
SELECT Colonnes FROM Table T1, Table T2 WHERE
Conditions
```

```
SELECT DISTINCT  R1.NE FROM obtenu R1, obtenu R2
WHERE R1.NE = R2.NE and R1.nomc != R2.nomC
```

Numéros d'étudiants ayant obtenu au moins 2 cours

Les sous-requêtes

Les sous-requêtes

- Il est possible d'inclure une requête dans une autre. Le nombre d'imbrications est quelconque. Une sous-requête peut être utilisé dans (FROM, WHERE)

- Vérifier si la valeur de colonne fait partie du résultat de la sous-requête

colonne IN | NOT IN (sous-requête)

- Comparer la valeur de colonne à tous les éléments du résultat

colonne Operateur ALL (sous-requête)

- Comparer la valeur de colonne à au moins une valeur du résultat de la sous-requête

colonne Operateur ANY (sous-requête)

- Comparer la valeur de colonne à la valeur du résultat de la sous-requête. Elle ne peut être utilisée que si la sous-requête renvoie 1 et 1 seul élément

colonne Operateur (sous-requête)

Les sous-requêtes conditionnelles

```
SELECT  * FROM obtenu, (SELECT * FROM cours WHERE nomc!='système') C
WHERE obtenu.nomC=C.nomC;

SELECT NE FROM etudiant WHERE NP IN (SELECT NP FROM Personne WHERE
adr='dijon')

SELECT NE FROM obtenu WHERE note >= ALL (SELECT note FROM obtenu)

SELECT NE FROM obtenu WHERE note >= (SELECT DISTINCT max(note)
FROM obtenu)
```