



LANGAGE SQL

Djamal Benslimane

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.

Connexion avec **oracle sql developer**

- Adresse du serveur ORACLE (nom d'hôte) : **iutdoua-oracle.univ-lyon1.fr**
- Nom de La base de donnée (SID) : **orcl**
- port utilisé : **1521**
- Login (nom d'utilisateur) : **pxxxxxxx**
- mot de passe : **code BIP**

1. Projection

- Affichage de toute la table :
 - **Select** * **from** une_table;
 - **Select** liste_des_attributs **from** une_table;
 - Exemple : **select** * **from** cours;
Select nomC, cycle, Nens **from** cours;
- Projection simple sur quelques colonnes
 - **Select** colonne1, colonne 2 **from** une_table;
 - Exemple : **select** nom, adr **from** personne;
- Projection étendue : implication de calculs dans les projections
 - **Select** expression1, expression2 **from** une_table
 - Exemple : **select** NE, nomC, note*1.2, annee **from** Obtenu
- Renommage des colonnes
 - **Select** colonne1 **as** col, expression **as** col2 **from** une_table
 - Exemple : **select** NE **as** numetudiant, nomC, note*2 **as** noteFinale, annee **from** Obtenu

2. Tri des données et occurrences uniques

- Tri : tout résultat peut être trié (ordonné) sur une ou plusieurs colonnes par ordre croissant ou décroissant
 - Select ... From ... **order by** col1 **asc|desc**, col2 **asc|desc**
 - Exemple :
 - **select** * **from** personne **order by** adr **asc**;
 - **Select** * **from** personne **order by** adr **asc**, nom **desc**;
- Occurrences uniques : pour supprimer des lignes identiques
 - **Select distinct** ... from ... ;
 - Exemple : **select distinct** NE **from** obtenu;

3. Sélections

- **Select From ... where** conditions order by ...
 - L'order by n'est pas obligatoire
 - Conditions : conditions élémentaires reliées par des opérateurs logiques and, or, not.
- **Condition élémentaire :**
 - **colonne operateur valeur**
 - Exemples : age > 20 adr!= 'paris'
 - **Colonne in (val1, val2, val3)**
 - Exemple : ville in ('paris', 'londres', 'lyon')
 - **Colonne between val1 and val2**
 - Exemple : age between 15 and 25
 - **Colonne not between val 1 and val2**
 - exemple : age not between 15 and 20
 - **Colonne is NULL** pour récupérer les colonnes qui n'ont pas de valeurs
 - **Colonne is NOT null** pour récupérer les colonnes qui ont des valeurs

3. Sélections

- Conditions complexes :
 - 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';
- Utilisation des caractères spéciaux pour les conditions sur des chaînes de caractères
 - 2 caractères spéciaux :
 - %** désigne une chaîne de caractères quelconque y compris la chaîne vide
 - _** désigne un et un seul caractère y compris le caractère blanc
 - Exemples :
 - nomC **like** '%S' pour tous les noms de cours se terminant par S.
 - NomC like 'a__o' pour tous les noms de cours à 4 caractères, commençant par a et se terminant par o
 - nomC **not like** 'a%' pour tous les noms de cours ne commençant pas par a
 - Requetes :
 - select * from cours where cycle = 1 and nomC like 'a%'
 - Select * from cours where cycle = 1 and nomC not like 'a%':

4. Produit cartésien et Jointure

- Produit cartésien
 - Select ... From table1, table2, table3
 - Exemple : **select** * **from** personne, enseignant
- Jointure : le where permet de faire les jointures de tables
 - **Select** ... **from** table1, table2, table3 **where** conditions
 - Exemple : **select** * **from** personne, enseignant **where** personne.NP = enseignant.NP;
- Jointure + projection
 - **Select** Personne.NP, nom **from** personne, enseignant **where** personne.NP = enseignant.NP;
- Jointure + projection + selection
 - **Select** Personne.NP, nom **from** personne, enseignant **where** personne.NP = enseignant.NP **and** enseignant.NP=3333;

4. Produit cartésien et Jointure

- Alias de tables : il est possible de renommer des tables dans la requête
 - **Select** P.NP, nom **from** personne P, enseignant E
where P.NP = E.NP **and** E.NP=3333;
- Auto-jointure : jointure d'une table avec sa copie
 - Numéros d'étudiants ayant obtenu au moins 2 cours
 - **Select distinct** R1.NE **from** obtenu R1, obtenu R2
where R1.NE = R2.NE **and** R1.nomc != R2.nomC

5. Intersection, Union, Différence

- Intersection
 - Select col1, col2, col3 from ... where ...
 - Intersect**
 - Select att1, att2, att3 from ... where ..
 - Même nombre de colonnes et meme types
- Union
 - Select col1, col2, col3 from ... where ...
 - union**
 - Select att1, att2, att3 from ... where ..
- Différence
 - Select col1, col2, col3 from ... where ...
 - minus**
 - Select att1, att2, att3 from ... where ..

- select * from cours;
- Select nomC, cycle, Nens from cours;
- select nom, adr from personne;
- select NE, nomC, note*1.2, annee from Obtenu;
- select NE as numetudiant, nomC, note*2 as noteFinale, annee from Obtenu;
- select * from personne order by adr asc;
- Select * from personne order by adr asc, nom desc;
- select DISTINCT NE from obtenu;
- Select * from cours where cycle = 2 and nomC='BD';
- select * from cours where cycle = 1 and nomC like 'a%';
- Select * from cours where cycle = 1 and nomC not like 'a%';
- select * from personne, enseignant;
- select * from personne, enseignant where personne.NP = enseignant.NP;
- Select Personne.NP, nom from personne, enseignant where personne.NP = enseignant.NP;
- Select Personne.NP, nom from personne, enseignant where personne.NP = enseignant.NP and enseignant.NP=3333;
- Select distinct R1.NE from obtenu R1, obtenu R2 where R1.NE = R2.NE and R1.nomc != R2.nomC;
- select NP from personne intersect select NP from etudiant;

6. Les fonctions d'agrégation

- Ce sont des fonctions qui opèrent sur une colonne pour en calculer la somme, la max, le min ...
 - 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
- Ces fonctions s'utilisent partout dans une requête sauf dans la clause WHERE

6. Les fonctions d'agrégation

– Exemples

- 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 (ne pas compter ceux qui n'ont pas encore obtenu des modules)
 - **select count(distinct NE)as** nombreEtudiants **from** obtenu;
- La plus grosse note en algo
 - **Select max(note) as** maxalgo **from** obtenu **where** nomC='algo';
- La plus grosse et la plus faible note d'algo
 - **Select max(note) as** maxalgo, **min(note) as** minalgo **from** obtenu **where** nomC='algo';
- L'étudiant qui a eu la plus grosse note
 - C'est compliqué ... la requête suivante ne le fait pas
 - **Select max(note) as** maxalgo, NE **from** obtenu **where** nomC='algo';

7. Groupement de lignes

- Un groupe de lignes est un ensemble de lignes reliées logiquement par les valeurs de certains attributs.
- Exemple de groupes : les étudiants qui suivent le même module, les personnes qui habitent la même ville, etc.
- **Select ... from ... where ... group by** colonne1, colonne2;
- Les lignes qui partagent les mêmes valeurs de colonne1 et colonne2 seront dans le même groupe.
- De chaque groupe on n'extracte que :
 - Les colonnes du group by
 - Le résultat d'une fonction d'agrégation
- Exemple : **select** NE, count(*) **as** nbre_cours_obtenus **from** obtenu **group by** NE;

7. Groupement de lignes conditionnel

- Il s'agit de pouvoir générer des groupes et de n'en garder que certains, ceux qui vérifient une condition.
- **Select ... from ... where ... group by ... having** condition_de_groupe;
- Condition de groupe peut utiliser une fonction d'agrégation et/ou les colonnes du group by;
- **Select** adr **from** personne **group by** adr **having** count(*)>1;
- Les villes qui ont au moins deux personnes.

8. Les sous-requêtes

- Il est possible d'inclure une requête dans une autre. Le nombre d'imbrications est quelconque.
- Les étudiants habitant Dijon
 - **Select** NE **from** etudiant **where** NP in (**select** NP **from** Personne **where** adr='dijon')
- Le numéro d'étudiant qui a la meilleure note en algo
 - **Select** NE **from** obtenu **where** note in (**select** max(note) **from** obtenu)

8. d'autres exemples de sous-requêtes

Select *

from obtenu, (select * from cours where
nomc!='système') C

Where obtenu.nomC=C.nomC;

Une sous requête peut utilisée dans le from

8. Condition sur une sous-requete

- **colonne IN (sous-requete)** : on vérifie si la valeur de colonne fait partie du résultat de la sous-requête
- **colonne >= (sous-requête)** : ne peut marcher que si la sous-requete renvoie 1 et 1 seul élément
- **colonne >= any (sous-requête)** : on vérifie si la valeur de colonne est supérieure à au moins une valeur du résultat de la sous-requete
- **colonne >= all (sous-requête)** : on vérifie si la valeur de colonne est supérieure à tous les éléments du résultat.

8. Exemples avec Condition sur une sous-requete

- **Select** NE **from** obtenu **where** note \geq all
(**select** note **from** obtenu);

C'est équivalent à :

Select NE **from** obtenu **where** note IN (**select**
distinct max(note) **from** obtenu);

- Select count(NP) from etudiant;
- select count(*) from etudiant;
- select count(distinct NE) as nombreEtudiants from obtenu;
- Select count(NE) from Obtenu:
- Select count(NP) as nombre_total from etudiant;
- Select max(note) as maxalgo, min(note) as minalgo from obtenu where nomC='algo';
- Select max(note) as maxalgo, NE from obtenu where nomC='algo';
- select NE, count(*) as nbre_cours_obtenus from obtenu group by NE;
- select NE, count(*) as nbre_cours_obtenus from obtenu where nomC!='algo' group by NE;
- Select adr from personne group by adr having count(*)>1;
- Select * from obtenu, (select * from cours where nomc!='système') C Where obtenu.nomC=C.nomC;
- Select NE from obtenu where note >= all (select note from obtenu);
- Select NE from obtenu where note IN (select distinct max(note) from obtenu);