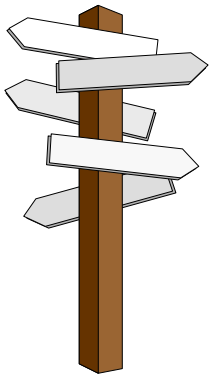
Cours n°4
Le langage de manipulation des données (LMD)

Chantal Reynaud

Université Paris X - Nanterre UFR SEGMI - Licence MIAGE



Plan

I. Introduction

II. Recherche de base (SELECT dans une seule table)

Partie I. Introduction

I. Le langage SQL : historique et fonctions

II. SQL et ORACLE

II.1. Architecture du SGBDR Oracle
II.2. SQL*PLUS ou l ’utilisation interactive de SQL sous Oracle
II.3. Conventions générales

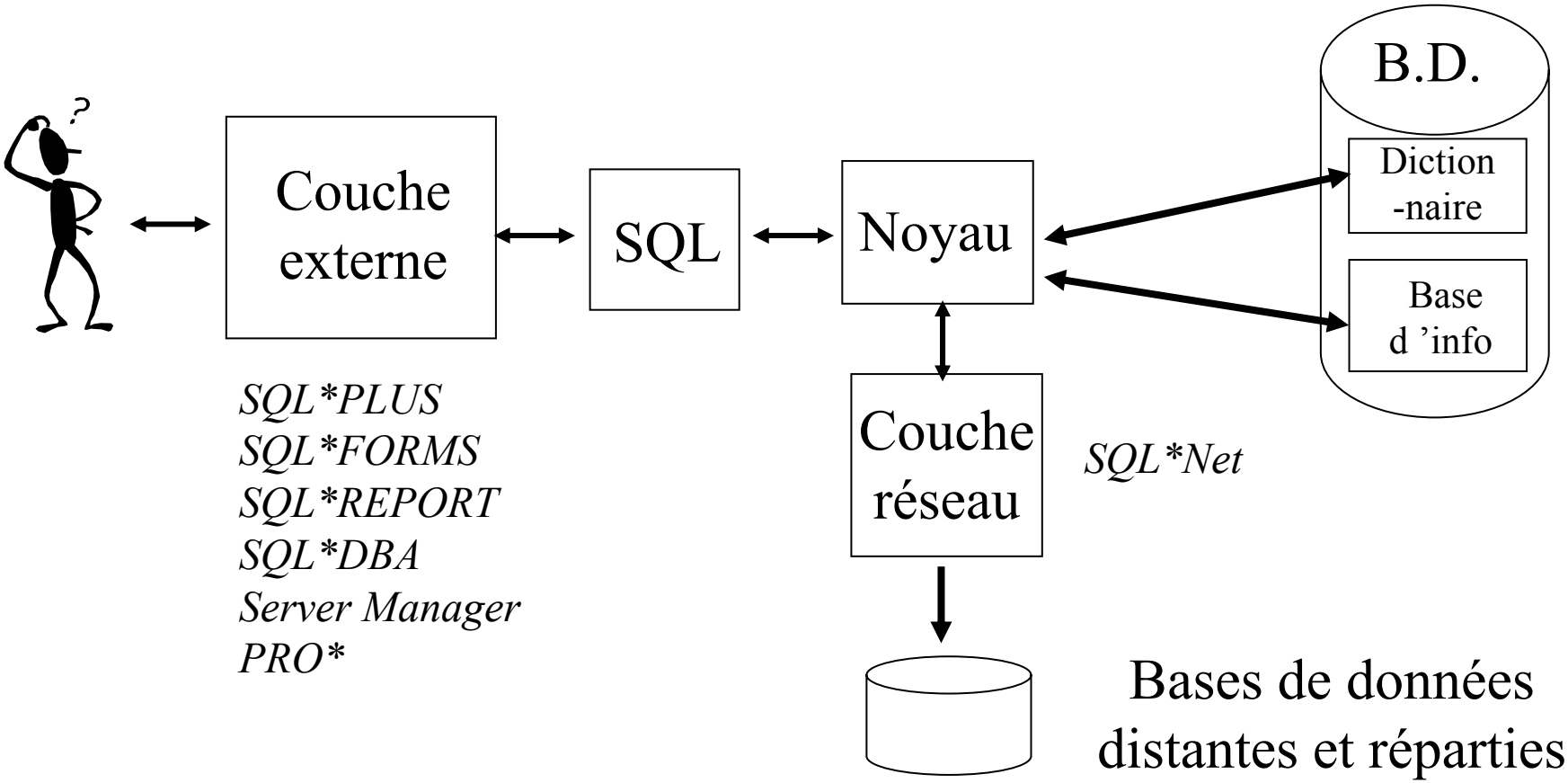
I. Le langage SQL

- SQL signifie « Structured Query Language » ou Langage d ’interrogation structuré.
- SQL est un langage complet de gestion de bases de données relationnelles. Il a été conçu par IBM dans les années 70 et est devenu le langage standard des SGBDR. SQL a été normalisé dès 1986 mais les premières normes, très incomplètes, ont été ignorées des éditeurs de SGBD. La norme actuelle SQL-2 (appelée aussi SQL-92) date de 1992. SQL-3 est en cours d ’homologation.
- C ’est à la fois :

* un langage d ’interrogation de la base
* un langage de manipulation des données
* un langage de définition des données
* un langage de contrôle de l ’accès aux données.

- SQL est utilisé par les principaux SGBDR : DB2, Oracle, Informix, Ingres, SQLServer, ACCESS. Chaque SGBD a cependant sa propre variante du langage. Ce cours présente un noyau de commandes disponibles sur l ’ensemble de ces SGBDR et leur implantation dans Oracle version 8.i.

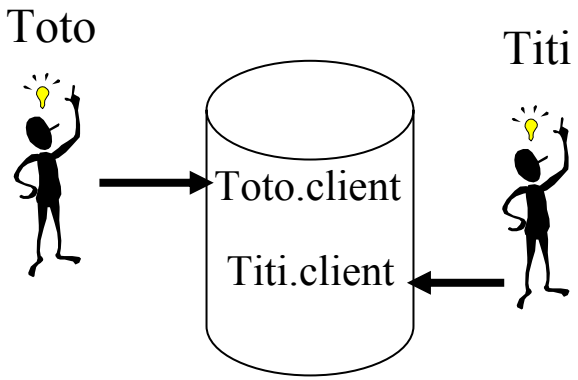
II. SQL et Oracle
Architecture du SGBDR Oracle



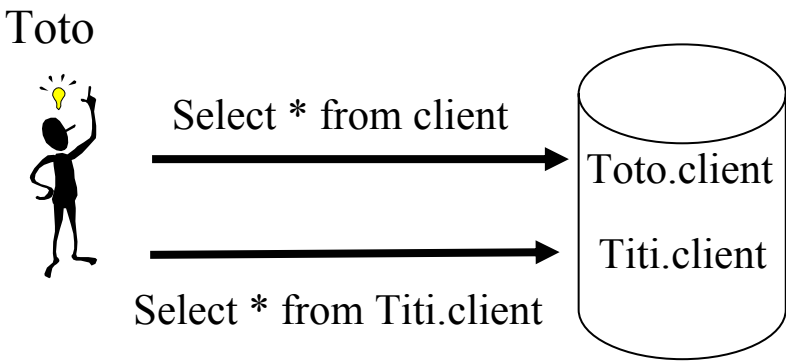
II. SQL - *Conventions générales*

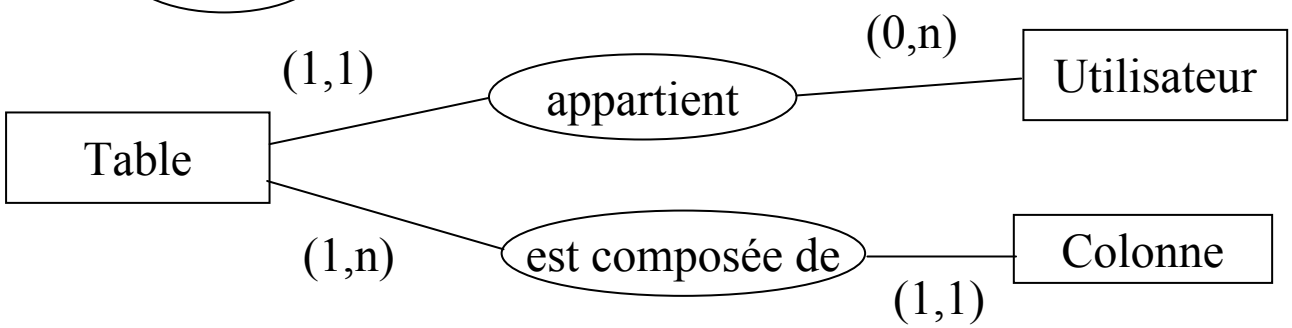
- Une base de données est un ensemble de tables gérées par un SGBD qui peuvent avoir été créées par différents utilisateurs.
- Lorsqu ’un utilisateur nomme une table, il s ’agit d ’une table lui appartenant.
- Pour accéder à une table appartenant à un autre utilisateur, il faut préfixer le nom de la table par le nom de l ’utilisateur et en avoir le droit.

Création d ’une table « client » par différents utilisateurs



Interrogation de la table client





Partie II. Recherche de base (SELECT dans une seule table)

I. Les opérations de projection d ’une table

II. Les opérations de sélection dans une table

III. Le tri des tuples

IV. Fonctions et expressions

V. Requêtes sur les groupes

La recherche de base ou interrogation

SELECT …
FROM …
WHERE …
GROUP BY …
HAVING …
ORDER BY ...

} Obligatoires

- Ordre des clauses à respecter.

II.1. Les opérations de projection d ’une table - *Choix des attributs*

SELECT nom(s) de colonne **FROM** nom_table**;**

a) SELECT * FROM STAGIAIRE;
b) SELECT Nom-St FROM STAGIAIRE;
c) SELECT Num-St**,** Nom-St FROM STAGIAIRE;
d) SELECT Nom-St AS nom FROM STAGIAIRE;
e) SELECT Nom-St nom FROM STAGIAIRE;
f) SELECT Nom-St AS 'nom de stagiaire' FROM STAGIAIRE;

- * signifie que toutes les colonnes de la table sont sélectionnées.
- Le nom complet d ’une colonne d ’une table est le nom de la table suivi d ’un point et du nom de la colonne. Ex : STAGIAIRE.Nom-St. Le nom de la table peut être omis quand il n ’y a pas d ’ambiguïté.
- L ’ordre des attributs dans la table résultat sera l ’ordre d ’apparition dans le SELECT ou l ’ordre dans la table si *
- Par défaut, les attributs de la table résultat portent le même nom que les attributs de la table d ’origine sauf si ce nom est redéfini :

SELECT nom_colonne [AS] nom_redéfini FROM nom_table;

II.1. Les opérations de projection d 'une table

Empêcher les répétitions de lignes

SELECT DISTINCT nom(s) de colonne **FROM** nom_table**;**

SELECT Type-Stage FROM STAGE;

Type-Stage
TP
TP
TP
TP
TP
Cours
TP
Démonstration
Démonstration

SELECT **DISTINCT** Type-Stage 'nature du stage' FROM STAGE;

nature du stage
TP
Cours
Démonstration

Remarque : Si on indique plusieurs noms de tables derrière le FROM, on obtient un produit cartésien.

II.1. Les opérations de sélection dans une table

Les éléments de la clause WHERE

● La clause WHERE permet de spécifier quelles sont les lignes à sélectionner dans une table ou dans le produit cartésien de plusieurs tables. Elle est suivie d ’un prédicat, c ’est-à-dire une expression logique prenant la valeur vrai ou faux et qui sera évalué pour chaque ligne. Les lignes pour lesquelles le prédicat est évalué à vrai sont sélectionnées.

● Les expressions logiques peuvent être composées d ’une suite de conditions combinées entre elles par les opérateurs AND, OR, NOT. L ’opérateur AND est prioritaire par rapport à l ’opérateur OR. Des parenthèses peuvent être utilisées pour imposer une priorité dans l ’évaluation du prédicat, ou simplement pour rendre plus claire l ’expression logique. L ’opérateur NOT placé devant un prédicat en inverse le sens.

● Les différentes formes d ’expressions logiques : comparaison à une valeur, comparaison à une fourchette de valeurs, comparaison à une liste de valeurs, comparaison à un filtre, test sur l ’indétermination d ’une valeur, test ‘ tous ’ ou ‘ au moins un ’, test existentiel, test d ’unicité

II.1. Les opérations de sélection dans une table

La comparaison à une valeur

```
WHERE exp1 = exp2
WHERE exp1 != exp2
WHERE exp1 < exp2
WHERE exp1 > exp2
WHERE exp1 <= exp2
WHERE exp1 >= exp2
WHERE exp1 is NULL
WHERE exp1 is not NULL
```

- Exp2 peut être une constante. Les chaînes de caractères doivent apparaître sous forme d ’une séquence de caractères entre apostrophes.
- Exp2 peut être une expression obtenue par application d ’opérateurs (+, -, x, /) sur des noms d ’attributs ou des constantes
- exp1 is NULL est vraie si exp1 a la valeur NULL. Le prédicat exp1 = NULL est toujours faux et ne permet pas de tester si l ’expression a une valeur indéterminée.

a) SELECT Num-St, Nom-St FROM STAGIAIRE WHERE Adr-St = 'Paris';
b) SELECT Num-Stage FROM STAGE WHERE Nb-jours > 4;
c) SELECT Num-Stage FROM STAGE WHERE (Prix-jour * nb-jours) > 5000;
d) SELECT Num-Stage, Libellé-Stage, Type-Stage, Num-Cat FROM STAGE
WHERE Type-Stage = ’TP ’ AND Num-Cat = 1;
e) SELECT Num-Stage, Libellé-Stage, Type-Stage, Num-Cat FROM STAGE
WHERE NOT(Type-Stage = ’TP ’ OR Num-Cat = 1);

II.1. Les opérations de sélection dans une table

La comparaison à une fourchette ou une liste de valeurs

SELECT … **FROM** … **WHERE** Cond1 [NOT] **BETWEEN** val1 **AND** val2;

SELECT Num-Stage, Libellé-Stage, Num-Cat FROM STAGE
WHERE Num-Cat BETWEEN 1 AND 3;

SELECT … **FROM** … **WHERE** Nom_colonne [NOT] **IN** (liste_valeurs);

SELECT Num-Stage, Libellé-Stage, Num-Cat FROM STAGE
WHERE Num-Cat IN (1,3);

- NOT IN sert à sélectionner les lignes dont l ’attribut dans la clause WHERE contient une valeur autre que celle de la liste
- MATCH est équivalent à IN
- MATCH UNIQUE rend la valeur vraie que si la valeur ne se trouve qu ’une seule fois dans la liste.

II.1. Les opérations de sélection dans une table
La comparaison à un filtre

SELECT … **FROM** … **WHERE** nom_colonne [NOT] LIKE ’Modèle de chaîne';

● Teste l ’égalité de deux chaînes en tenant compte des caractères jokers dans la 2ème chaîne. « _ » remplace 1 caractère exactement, « % » remplace une chaîne de caractères de longueur quelconque.
● ’%objet%’ sélectionne sur la présence de la constante objet à n ’importe quelle place dans la chaîne
● ’%a_’ sélectionne sur la présence d ’un a minuscule à l ’avant-dernière place
● ’%@%%’ sélection sur la présence de % n ’importe où dans la chaîne.

a) SELECT Num-St, Nom-St, Prénom-St FROM STAGIAIRE
WHERE Nom-St LIKE ’Var%’;
b) SELECT Num-St, Nom-St, Prénom-St FROM STAGIAIRE
WHERE Nom-St LIKE ’PE_____N’;
c) SELECT Num-St, Nom-St, Prénom-St FROM STAGIAIRE
WHERE Nom-St LIKE ’_AR%’;

III. Le tri des tuples

- L ’ordre dans lequel les lignes d ’une table résultat apparaissent est indéterminé.
- Pour trier les lignes dans un ordre déterminé, il faut utiliser la clause ORDER BY.

SELECT … **FROM** … **ORDER BY num_colonne1 [DESC], num_colonne2 [DESC], ...** ;

- On peut trier selon la valeur d ’un ou de plusieurs attributs.
- Les attributs selon lesquels le tri est demandé doivent obligatoirement faire partie de la clause SELECT.
- L ’option facultative DESC donne un tri par ordre décroissant. Par défaut, l ’ordre est croissant.
- Le tri se fait d ’abord selon le premier attribut, puis les lignes ayant la même valeur pour ce premier attribut sont triées selon le 2ème attribut, etc.
- Les valeurs nulles sont toujours en tête.

a) SELECT Num-Stage, Libellé-Stage FROM STAGE ORDER BY Num-Stage;
b) SELECT Num-Stage, Libellé-Stage FROM STAGE ORDER BY 1;
c) SELECT Num-Stage, Libellé-Stage, Nom-Cat FROM STAGE
WHERE NOT(Type-Stage = ’Cours’) ORDER BY Num-Cat, Num-Stage DESC;

IV. Fonctions et expressions
Les différentes fonctions disponibles sous Oracle

- Fonctions arithmétiques
- Fonctions chaînes de caractères
- Fonctions Date
- Fonctions de conversion
- Fonctions diverses
- Fonctions de groupe

IV. Fonctions et expressions
Les fonctions de groupe

● Un groupe est un ensemble de tuples d 'une table pour lesquels la valeur d 'un attribut est constante.
● Les fonctions de groupe effectuent un calcul sur l 'ensemble des valeurs d 'un attribut pour un groupe de tuples.
● Il existe 5 fonctions de groupe (7 sous Oracle) : COUNT, SUM, AVG, MIN, MAX (+ VAR, STDEV sous Oracle)
● Une fonction peut apparaître dans une clause SELECT, elle sera affichée comme un attribut ou une expression MAIS, en l 'absence de clause GROUP BY, on ne peut mettre ensemble des fonctions de groupe et des attributs dans un même SELECT.

a) SELECT AVG(Nb-jours) Durée_moyenne FROM STAGE;
b) SELECT AVG(Poids-Art) pmoyen, MAX(PV-Art - PA-Art) Margemax FROM ARTICLE

● COUNT(*) compte les lignes de la table résultat (toutes, même celles dont la valeur est NULL)
● COUNT(distinct attribut) compte le nombre de valeurs différentes prise par l 'attribut

V. Requêtes sur les groupes
La clause Group by

● Elle réarrange la table résultat en un nombre minimum de groupes, tels que, à l 'intérieur de chaque groupe, l 'attribut spécifié possède la même valeur pour chaque tuple. Cette clause n 'affecte pas l 'organisation physique de la table.
● Lorsqu 'on spécifie une clause GROUP BY, les fonctions de groupe sont calculées pour chaque groupe.
● Cette clause permet l 'affichage de l 'attribut commun aux tuples d 'un groupe, suivi de l 'affichage de la valeur des fonctions appliquées au groupe.
● On peut ajouter une clause WHERE qui sélectionne les tuples

SELECT **Nom-Cat**, AVG(Nb-jours) FROM STAGE GROUP BY **Nom-Cat**;

SQL fait des groupes d 'après la valeur de Nom-Cat, puis, pour chaque valeur de Nom-Cat, calcule la moyenne de l 'attribut Nb-jours. Il affiche les valeurs de Nom-Cat correspondant à chaque groupe et la moyenne associée.

V. Requêtes sur les groupes
La clause Having

- Cette clause est l 'équivalent du WHERE appliquée aux groupes.
- Elle ne peut exister qu 'avec une clause GROUP BY.
- Pratiquement, le critère spécifié dans le HAVING porte sur la valeur d 'une fonction calculée dans un groupe.

a) SELECT Nom-Cat, AVG(Nb-jours) FROM STAGE
GROUP BY Nom-Cat
HAVING AVG(Nb-jours) > 4;
b) SELECT Nom-Cat, AVG(Nb-jours) FROM STAGE
WHERE Type-Stage = 'TP'
GROUP BY Nom-Cat
HAVING AVG(Nb-jours) > 4;

Attention :
1) En présence d 'un GROUP BY, tout attribut dans la clause SELECT doit figurer dans la clause GROUP BY.
2) Une fonction de groupe ne se trouver que dans les clauses SELECT ou HAVING.