

Module 5

SQL : Structured Query Language

Introduction

- Le langage SQL est à la fois :
 - un langage d'interrogation de données (LID) : SELECT ;
 - un langage de manipulation de données (LMD) : UPDATE, INSERT, DELETE
 - un langage de définition des données (LDD) : ALTER, CREATE, DROP
 - un langage de contrôle des données et des utilisateurs (LCD) : GRANT, REVOKE

Langage d'interrogation des données

LID

SQL Interrogation d'une BD

- La commande **SELECT** permet :
 - Les projections
 - Les restrictions
 - Les produits cartésiens
 - Les jointures
 - Les opérations union, intersection et différence

SQL Interrogation d'une BD

- La structure de la commande SELECT :

```
SELECT [ALL | DISTINCT] <Colonne> +
FROM < table>+
WHERE < critères > +
```

SQL La Projection

- SQL n'élimine pas les doubles, à moins que cela soit explicitement demandé par le mot réservé DISTINCT.
- La projection en SQL s'exprime de la manière suivante :

```
SELECT [ALL| DISTINCT] <Colonne> +  
FROM <table>
```

SQL La Projection (2)

Fournisseur(NumFr, NomFr, Adr)

Produit (Ref, designation, cond, prix, #NumFr)

Commande(NumC, Dte, Qte, #Refprd)

Q1 : Donner toutes les commandes

```
SELECT *
FROM Commande ;
```

Q2 : Donner le nom et l' adresse de chaque fournisseur

```
SELECT NomFr , Adr
FROM Fournisseur ;
```

Q3: Quelles sont tous les prix différents.

```
SELECT DISTINCT prix
FROM Produit ;
```

SQL La Restriction

- En SQL la sélection est une combinaison d'une restriction suivie d'une projection.
- La sélection s'exprime comme une projection avec en plus une condition de recherche selon la syntaxe suivante :

```
SELECT [ALL | DISTINCT] <Colonne> +  
FROM <table>  
WHERE <Condition de recherché>
```

SQL La Restriction (2)

■ Exemple

Q1 : Les produits dont le prix dépasse 100 Dh.

```
SELECT *  
FROM Produit  
WHERE prix >100 ;
```

Q2 : Donner les designations et les prix des produits vendus par le fournisseur numéro 125

```
SELECT designation, prix  
FROM Produit  
WHERE NumFr=125;
```

SQL Conditions de recherche

- On utilise les opérateurs :

- de comparaison : < , <= , > , >= , =, <>
- Logiques : NOT , AND , OR
- <expr1> BETWEEN <expr2> AND <expr3>
- <expr1> IN (<exprk>+)
- <exp> LIKE <modèle >
 - où modèle est une chaîne de caractères pouvant contenir l'un des caractères jokers :
 - _ remplace exactement 1 caractère
 - % remplace une chaîne de caractères de longueur quelconque.

SQL Nom de colonne de sortie

- Les colonnes constituant le résultat d'un SELECT peuvent être renommées dans le SELECT
- il suffit de faire suivre l'expression définissant la colonne d'un nom, selon les règles suivantes :
 - le nom (30 caractères maximum) est inséré derrière l'expression définissant la colonne, séparé de cette dernière par un espace.
 - si le nom contient des séparateurs (espace, caractère spécial), ou s'il est identique à un mot clé de SQL (ex : DATE), il doit être mis entre guillemets "".

```
SELECT Ref , designation , prix ,  
       prix*1.05 " Nouveau prix".  
FROM Produit  
WHERE NumFr=125;
```

SQL Le produit cartésien

- Le produit cartésien en SQL est un cas particulier de jointure (c'est une jointure sans condition).
- L'expression d'un produit cartésien en SQL est la suivante :

```
SELECT [ALL | DISTINCT] <Colonne> +  
      FROM <table> +
```

- Dans cette commande SQL on combine le produit cartésien et la projection.
- Le produit peut mettre en œuvre plus de deux tables

SQL La jointure

- On peut joindre jusqu'à 256 tables.
- Une jointure est un produit cartésien suivie d'une restriction .
- Elle se formule simplement en spécifiant plusieurs tables derrière la clause FROM de la façon suivante :

```
SELECT [ALL| DISTINCT] <Colonne> +  
FROM <table>+  
WHERE <Condition de Jointure>
```

- La jointure en SQL combine la jointure et la projection.
Il n'existe pas d'associations implicites entre les tables.

SQL Equi-jointure

Exemple

- Q1 : Donner la liste des produits comportant le nom et l'adresse du fournisseur

```
SELECT Ref , designation , cond, prix, NomFr , Adr  
FROM Fournisseur , Produit  
WHERE Fournisseur.NumFr=Produit.NumFr ;
```

- Q2 : Donner les lignes de commande avec les montants

```
SELECT NumC, Refprd , Dte , Qte , Qte*prix  
FROM Commande , Produit  
WHERE Refprd=Ref ;
```

SQL Auto-jointure

- On peut joindre une table à elle même
- Dans ce cas, il faut renommer au moins l'une des deux occurrences de la table en lui donnant un alias, afin de pouvoir

```
SELECT Employe.Mat , Employe.Nom,  
       Employe.Prn, Superieur.Nom  
  FROM Employe , Employe Superieur  
 WHERE Employe.MatSup=Superieur.Mat ;
```

Q : Donner pour chaque employé le nom de son supérieur hiérarchique.

SQL Autres-jointures

- on peut utiliser d'autres types de comparaisons comme critères de jointures.
- Exemple

Q1 Quels sont les employés gagnant plus que l'employé dont le matricule est A25621?

```
SELECT Employe.Mat , Employe.Nom, Employe.Prn,  
        Employe2.Nom  
  
FROM Employe , Employe Employe2  
  
WHERE Employe.Sal > Employe2.Sal  
      AND Employe2.Mat="A25621"
```

SQL Union , Intersection et Différence

- En SQL les opérateurs ensemblistes se construisent à l'aide de deux SELECT combinés par l'un des opérateurs ensemblistes suivants :
 - l'union : UNION
 - l'intersection : INTERSECT
 - la différence relationnelle : MINUS

```
SELECT ...  
{UNION | INTERSECT | MINUS }  
SELECT ...
```

SQL Classer les sorties d'une requête

- Il est possible dans une requête SELECT de demander la

```
SELECT *  
FROM Employe  
ORDER BY Sal DESC , Nom , Prn;
```

- SELECT Nom , Prn , Sal

- FROM Employe
WHERE Sal >3000
ORDER BY Sal DESC , Nom , Prn;

SQL Fonctions de groupement

Q1 :Le nombre d'employés

```
SELECT COUNT (*)
FROM Employe;
```

Q2 :Le nombre d'employés ayant un salaire >2000

```
SELECT COUNT (*)
FROM Employe
WHERE Sal >2000;
```

Q3 :Le nom, prénom et salaire des employés ayant le salaire le plus élevé

```
SELECT Nom , Prn , Sal
FROM Employe
WHERE Sal = SELECT MAX (Sal)
      FROM Employe;
```

uls de

SQL les agrégats

Q1 : Les moyennes des prix des produits par fournisseur ?

```
SELECT NumFr , AVG(prix)  
FROM Produit  
GROUP BY NumFr
```

- Les fonctions de groupement doivent être indiquées dans les expressions qui suivent SELECT

SQL les agrégats (2)

Q1 : Les sommes par produit des Quantités commandées entre 25/10/02 et 25/11/02 ?

```
SELECT Refprd, SUM(Qte)
FROM Commande
WHERE Dte BETWEEN 25/10/02 ANDt 25/11/02
GROUP BY Refprd;
```

Q2 : liste des salaire moyens par fonction pour les groupes ayant plus de deux employé

```
SELECT Fonct,COUNT(*) Effectif , AVG(Sal) "Salaire moyen"
FROM Employe
GROUP BY Fonct
HAVING COUNT(*)>2 ;
```

Les sous requêtes

- Une sous requête est une requête dont le résultat est utilisé comme argument dans une autre requête
- Les conditions de restriction, clause WHERE, peuvent utiliser le résultat d'une requête
- On distingue 3 cas :
 - La sous requête reproduit une valeur
 - La sous requête reproduit une ligne
 - La sous requête reproduit plusieurs lignes

Les sous requêtes (2) : une valeur

- Lorsque le résultat de la sous requête est une valeur, on utilise la syntaxe :

WHERE <Expression> <Opt> SELECT ...

```
SELECT *
```

```
FROM Produit
```

```
WHERE prix =
```

```
    SELECT prix
```

```
    FROM Produit
```

```
    WHERE Ref ='125';
```

```
SELECT Nom, Prn , Sal
```

```
FROM Employe,
```

```
WHERE Sal = SELECT MAX(Sal)
```

```
        FROM Employe;
```

Les sous requêtes (3) : une ligne

- Lorsque le résultat de la sous requête est une ligne de plusieurs colonnes :

WHERE <liste colonne> <Opt> SELECT ...

Q : Quels sont les employés ayant même fonction et même supérieur que l'employé matriculé B261458 ?

```
SELECT Nom, Fonct, MatSup  
FROM Employe  
WHERE (Fonct, MatSup) =  
      ( SELECT Fonct, Matsup  
        FROM Employe  
        WHERE Mat = 'B261458');
```

Les sous requêtes (3) :Groupe de lignes

- Lorsque le résultat de la sous requête comporte plusieurs lignes on utilisera des quantificateurs pour réaliser des

```
    SELECT Nom, Sal
```

```
        FROM Employe
```

```
        WHERE Sal > ALL
```

```
            SELECT Sal
```

```
                FROM Employe
```

```
                WHERE Fonct = "Ingénieur" ;
```

OI

La clause JOIN

- A partir de SQL2 on peut exprimer les jointures en ajoutant la clause JOIN dans FROM

- Jointure normale

SELECT <colonne>⁺

FROM <Table1> JOIN <Table2>

ON <Condition de jointure>

- Jointure naturelle

SELECT <colonne>⁺

FROM <Table1> NATURAL JOIN <Table2>

[USING (<colonne_de jointure>⁺)]

Langage de définition des données des données

LLD

- Un LDL permet de définir :

- Des schémas de relations
- Des vues
- Des contraintes d'intégrité
- Des index
- Des séquences

Création d'une table

- La commande CREATE TABLE permet de définir le schéma d'une table

CREATE TABLE <nom de table> (<élément de table> +)

- Les éléments d'une table sont :
 - les descriptions des colonnes
 - les descriptions des contraintes de table
- Les colonnes sont décrites par : Nom ; type de données, valeur par défaut et les contraintes de colonne

Création d'une table : domaines de colonne

- Les types standard en SQL sont :
 - INTEGER

```
CREATE TABLE ETUDIANT
```

```
(      Mat   CHAR(15),  
          Nom   CHAR(20),  
          Prn   CHAR(20),  
          Adr   VARCHAR2(80) ,  
          Dtn   DATE ,  
          Sexe  CHAR  
);
```

Création d'une table : Contraintes de colonne

- Les contraintes de colonne portent sur une seule colonne :

[CONSTRAINT<Nom>] < Description constraint>

```
CREATE TABLE ETUDIANT
(
    Mat   CHAR(15) CONSTRAINT Clef_ETUD
          PRIMARY KEY ,
    Nom   CHAR(20) NOT NULL,
    Prn   CHAR(20) NOT NULL,
    Adr   CHAR(80) DEFAULT NULL,
    Dtn   DATE CHECK ( Dtn >1900/01/01 ),
    Sexe CHAR DEFAULT ‘0’
          CHECK( Sexe IN (“0”, “1”))
);
```

Création d'une table :

Contraintes de table

CREATE TABLE ETUDIANT

```
(  Mat  CHAR(15),
    Nom  CHAR(20) NOT NULL,
    Prn  CHAR(20) NOT NULL,
    Adr  CHAR(80) DEFAULT NULL,
    Dtn  DATE,
    Sexe  CHAR DEFAULT '0' ,
    CONSTRAINT Clef_ ETUD          PRIMARY KEY
        (Mat),
        CHECK (
            ( Dtn >1900/01/01 ) And ( Sexe IN ('0', '1')) )
);
```

Création d'une table avec insertion

- On peut créer une table et la remplir à partir des sorties d'une requête :

```
CREATE TABLE <nom de table>  
[(<élément de table> +) ]  
AS <Clause SELECT >
```

Modification de schémas

- On peut modifier le schéma d'une table en ajoutant , modifiant ou supprimant des éléments de table :

ALTER TABLE <nom de table>

ADD | DROP | ALTER [COLUMN |CONSTRAINT]

(élément de table +)

```
ALTER TABLE ETUDIANT ADD
( Sect CHAR(4) DEFAULT NULL
    CHECK Sect IN ( "PC1", "PC2", 'BG1', "BG2",
                    "MP1", "MP2" ),
  Bac  CHAR(20)
);
```

Renommer / Supprimer une table

- La commande DROP TABLE supprime une relation existante :
DROP TABLE<Nom de table>
- La commande RENAME change le nom d'une table :
RENAME <Ancien_Nom> TO <Nouveau_Nom>

Les Domaines

- Un domaine au sens de la norme SQL est la définition d'un type associé à un certain nombre de règles de validité.
- La définition d'un domaine s'opère avec la syntaxe suivante :

CREATE DOMAIN <nom de domaine>
[AS] < Type de donnée >
[DEFAULT valeur]
{ [CONSTRAINT <nom>] CHECK <condition> }⁺

Exemple de définition de domaine ...

- CREATE DOMAIN T_val_plus
AS INTEGER
CONSTRAINT ctr_plus CHECK (VALUE > 0) ;

- CREATE DOMAIN OUI_NON
AS CHAR(3)
DEFAULT 'NON'
CONSTRAINT ctr_Oui_Nom CHECK (UPPER(VALUE)
IN ('OUI', 'NON')) ;

.. Exemple de définition de domaine ...

- CREATE DOMAIN T_maj_8
AS CHAR(8)
CHECK (VALUE = UPPER (VALUE));

- CREATE DOMAIN T_Code
AS CHAR(3)
CHECK (SUBSTR(VALUE,1,1) BETWEEN "A" AND "Z") ,
CHECK(SUBSTR(VALUE,1,1) NOT IN ("I","O")),
CHECK (SUBSTR(VALUE, 2, 1) BETWEEN "0" AND "9")
CHECK (SUBSTR(VALUE, 2, 1) BETWEEN "0" AND "9") ;

Modification de Domaines

- La norme SQL 2 autorise la modification d'un domaine en permettant de rajouter ou retirer une clause défaut ou une contrainte :

```
ALTER DOMAIN nom_domaine  
{ SET DEFAULT valeur_defaut  
| DROP DEFAULT  
| ADD contrainte_de_domaine  
| DROP CONSTRAINT nom_contrainte }
```

Il n'est pas possible de changer le type de données de la définition d'un domaine.

Modification de Domaines : exemple

- ALTER DOMAIN T_Code
ADD CHECK(SUBSTR(VALUE,1,1) NOT IN ("I" , "O")) ;
- ALTER DOMAINE T_val_plus
SET DEFAULT 1 ;

La suppression d'un domaine s'effectue avec un ordre SQL DROP
:

DROP DOMAIN <Nom_domaine> ;

- sont des éléments indispensables à une exploitation performante de base de données.
 - Souvent, lors de la création d'une contrainte :
 - de clef primaire,
 - de clef étrangère ,
 - d'unicité,
- le SGBDR implante automatiquement un index pour assurer la mécanisme de contrainte avec des performances correctes.

Les index (2) : choix des index

- :Indexer en priorité
 - 😊- les clés primaires
 - 😊- les colonnes servant de critère de jointure
 - 😊- les colonnes servant de critère de recherche
- Ne pas indexer :
 - 😢 - les colonnes contenant peu de valeurs distinctes
(index alors peu efficace)
 - 😢 - les colonnes fréquemment modifiées

Index (3) : Création / Suppression

- Commande de création :

***CREATE [UNIQUE] INDEX <nom_index>
ON nom_table (<colonne> +);***

- Commande de suppression :

DROP INDEX <nom_index>;

Exemple :

CREATE INDEX idx_nom_prn ON ETUDIANT (Nom , Prn)

Les vues (view)

- Une vue est une table virtuelle dont le schéma et le contenu sont dérivés d'une ou plusieurs tables réelles par une requête.
- On peut dire qu'une vue est une instance d'une requête.
- Grâce aux vues, chaque utilisateur pourra avoir sa vision propre des données.

Les vues : création / suppression

- Commande de création :

CREATE VIEW <nom_vue>
[<colonne> +]
AS < requête de sélection>
[WITH CHECK OPTION];

- Commande de suppression :

DROP VIEW <nom_vue>

Les vues : exemples

- ***CREATE VIEW Etudiant_PC2***
AS SELECT Mat , Nom , Prn
From ETUDIANT
WHERE Sect = "PC2" ;
- ***CREATE VIEW Ouvrage***
AS SELECT ISBN, Titre, NomEdt, NomAut
From Livre L , Auteurs A , Editeur D, Ecriture E
WHERE L.NumEdt=D.NumEdt
And L.NumL=E.NumL
And E.NumAut=A.NumAut ;

Utilisation des vues

- Une vue peut être utilisée comme une table dans toute requête de type SELECT.
- à la différence des tables, une vue ne peut être mise à jour (INSERT, UPDATE, DELETE) que si elle obéit à la condition suivante :

Comporter suffisamment d'attributs pour permettre un report des mises à jour dans la base sans ambiguïté.

- En pratique, la plupart des systèmes restreignent les possibilités de mise à jour à des vues mono table :
 - contenir la clef primaire de la table
 - ne pas transformer les données (pas de concaténation, addition de colonne, calcul d'agrégat...)
 - ne pas contenir de clause GROUP BY ou HAVING
 - ne pas contenir de sous requête
- La clause WITH CHECK OPTION implique que si la vue peut être mise à jour, alors les valeurs modifiées insérées ou supprimées doivent vérifier les conditions de la clause WHERE .

Création/suppression de base de données

- En SQL une BD est appelée schéma, la commande de création est :

```
CREATE {SCHEMA | DATABASE} 'spécification'  
[USER 'Nom_user' [PASSWORD 'Mot_de_passe']]  
[DEFAULT CHARACTER SET <jeu_de_carac>]
```

- La suppression d'un schéma :

```
DROP {SCHEMA | DATABASE} 'spécification' [CASCADE]
```

Langage de manipulation de données

LMD

Ajout de lignes

- Commande :

***INSERT INTO <table | vue>
[(<Colonne>+)]
{ VALUES (<valeur>+) | <requête Select>***

Le SELECT peut contenir n'importe quelle clause sauf un ORDER BY .

***INSERT INTO Etudiant (Mat , Nom , Prn)
VALUES ('128', 'Bilal' , 'omar' , '235' , 'Dalil' , 'Aicha') ;***

Modification / Suppression de lignes

- Commande de modification :

UPDATE <table | vue>

SET { <Colonne> = <Expression de valeur>}⁺

[WHERE <condition de recherche>];

- Commande de suppression :

DELETE FROM <table | vue>

WHERE <condition de recherche> ;

Exemples de Modifications

```
UPDATE Produit  
SET   prix = 55 ;
```

```
UPDATE ETUDIANT  
SET   Nom = UPPER(Nom),  
      Prn = UPPER(Prenom) ,  
      Adr = UPPER(Adr);
```

```
UPDATE Commande  
SET   Qte =Qte/2  
WHERE Refprd IN (Select Ref FROM Produit WHERE  
Cond=2) ;
```

```
UPDATE Produit  
SET   prix = prix*1.05 ;
```

```
UPDATE ETUDIANT  
SET   Sect="PC2"  
WHERE Sect="PC1" ;
```

Exemples de suppression de lignes

```
DELETE Produit ;
```

```
DELETE ETUDIANT  
WHERE Sect="PC1" ;
```

```
DELETE Commande  
WHERE Refprd IN (Select Ref FROM Produit WHERE  
Cond=2) ;
```



SEJEN

POWERING DECISION MAKING