Proyecto FAO COPEMED
Universidad de Alicante
Ramón y Cajal, 4
03001 - Alicante, España

GCP/REM/057/SPA

Web : www.fao.org/fi/copemed *Tel : +34 96 514 59 79* *Fax : +34 96 514 59 78* *Email : copemed@ua.es*

Formation à l'utilisation des Systèmes de Gestion de Bases de Données Relationnelles

organisée avec la collaboration du

Centre Royal de Télédétection Spatiale (*)

LANGAGE SQL (STRUCTURED QUERY LANGUAGE)

Support de cours (**)

Rabat (Maroc), 28 février – 3 mars 2000

(*) CRTS 16 bis, Avenue de France Agdal, Rabat Maroc Tel. : +212 (7) 776305 Fax : +212 (7) 776300 E-mail : layachi@crts.gov.ma

(**) Société AT TIME 28, Rue Michlifen, Appt. 2 Agdal – Rabat (Maroc) Tel./Fax : +212 (7) 672030 E-mail : attime@mtds.com

LANGAGE SQL (STRUCTURED QUERY LANGUAGE)

Langage relationnel commercial mettant en jeu, pour la consultation, une combinaison de l'algèbre relationnelle et du calcul relationnel

La norme SQL :

1970 : article de E : F : CODD « A Relational model for large Data Banks » ACM vol 13, No 6, octobre 1970 présentant la théorie des bases de données relationnelle

IBM ➔ system R, SEQUEL

Barkley ➔ INGRESS, QUEL

1980 : SEQUEL enrichi et amélioré, a donné lieu à SQL : langage utilisé par les SGBD SQL/DS, DB2

1986 : norme SQL 86 préparée par le comité (X3H2) de l'ANSI, adoptée également par ISO et par X/open en 1987.

1989 : ANSI publie une extension de la norme sous le nom SQL 89.

L'instruction SELECT :

Elle est utilisée pour extraire des enregistrements de la base de données sous la forme d'un jeu d'enregistrements. l'instruction SELECT ne modifie pas les données contenues dans la base de données ; elle ne fait que les extraire. La forme globale de l'instruction SELECT est :

```
SELECT Listechamps
FROM tablenames IN databasename
WHERE conditions
GROUP BY Liste_des_champs
HAVING Groupe_de_Critères
ORDER BY Liste_des_champs
WITH OWNER ACCESS OPTION
```

Par exemple cette instruction renvoie toutes les colonnes de toutes les enregistrements contenus de la table Dépôt :

```
SELECT * FROM employés ;
```

L'astérisque indique que toutes les colonnes de la ou les tables souhaitées doivent être extraites. Vous pouvez spécifier uniquement certaines colonnes. Une fois affichées, les données de chaque colonne apparaissent dans l'ordre dans lequel elles figurent dans la liste, ce qui vous permet de les réordonner pour plus de lisibilité :

```
SELECT [nom] , [adresse] FROM employés ;
```

Si un nom de champs figure dans plusieurs des tables spécifiées dans la clause FROM, il doit être précédé du nom de la table et de l'opérateur **.** (point). Dans l'exemple suivant, le champ département se trouve à la fois dans la table « employés » et « superviseurs ». L'instruction SQL sélectionne « département » de la table « employés » et supnom de la table « superviseurs »

```
SELECT employés.département, superviseurs.supnom
FROM  employés,superviseurs
WHERE employés.département =  superviseurrs.département ;
```

La Clause WHERE :

Cette clause indique quels enregistrements des tables spécifiées dans la clause FROM doivent être pris en compte dans les résultats de l'instruction SELECT. La clause WHERE est facultative.
Par exemple : vous pouvez sélectionner tous les employés du service ventes de la façon suivante : WHERE departement = 'Sales'

Pour trouver des enregistrements datés du 10 mai 1994 dans une base de données française, vous devez utiliser l'instruction SQL suivante :

```
SELECT *  FROM  Commande  _
WHERE [date commande] = #5/10/98#
```

Le prédicat :

Une ou plusieurs conditions reliées par des opérateurs logiques AND et OR, NOT peut être utilisé pour la négation,

```
SELECT Clientèle.client, ville
FROM Crédit, Clientèle
WHERE Crédit.client = Clientèle.client AND agence = "My Youssef"
```

Conditions de comparaison :

EXPR opérateur de comparaison EXPR

- EXPR : constante, nom de colonne ou une combinaison des deux reliés par des opérateurs arithmétiques : { +, -, *, / }
- Opérateur de comparaison : { =, <>, >, <, >=, <= }

```
SELECT client FROM Crédit
WHERE agence = "My Youssef" AND montant >=700
```

EXPR [NOT] BETWEEN EXPR AND EXPR

```
SELECT client FROM Crédit
WHERE montant BETWEEN 500 AND 700
```

EXPR [NOT] IN (liste de valeurs)

SELECT client FROM Clientèle

WHERE ville IN ("RABAT", "CASA")

Nom colonne [NOT] LIKE Chaîne

Chaîne peut contenir des caractères génériques : Exemple : %, _

SELECT client FROM Clientèle

WHRE ville LIKE "M%"

Nom colonne IS [NOT] NULL

SELECT * FROM Clientèle

WHERE adresse IS NULL

Conditions sous-requêtes :

Permet de comparer une expression ou une colonne au résultat d'une autre requête SELECT (requêtes imbriquées)

EXPR [NOT] IN (requête SELECT)

Appartenance de tuples à une relation (restriction de tuples)

```
SELECT client FROM Crédit
WHERE agence = "My Youssef" AND client IN
(SELECT client FROM Dépôt WHERE agence = "My Youssef")

SELECT client FROM Dépôt
WHERE agence = "My Youssef" AND client NOT IN
(SELECT client FROM Crédit WHERE agence = "My Youssef")
```

Appartenance testée sur plusieurs attributs :

```
SELECT client FROM Crédit
WHERE agence = "My Youssef" AND <agence, client> IN
(SELECT agence, client FROM Dépôt)
```

notion de variable de tuples :

variable associée à une relation particulière, définie dans FROM.

```
SELECT T.client, ville
FROM Crédit S, Cientèle T WHERE S.client = T.client
```

Utile pour comparer deux tuples d'une même relation :

```
SELECT T.client, S.client FROM Dépôt S, Dépôt T
WHERE S.client = "Radi" AND S.agence = T.agence
                    =
SELECT client FROM Dépôt WHERE agence IN
(SELECT agence FROM Dépôt WHERE client = "Radi")
```

EXPR Opérateur de comparaison {ANY |[ALL]} (requête SELECT)

- >ANY, >=ANY, <ANY, <=ANY, <>ANY, (=ANY identique à IN) : comparaison à "quelconque".

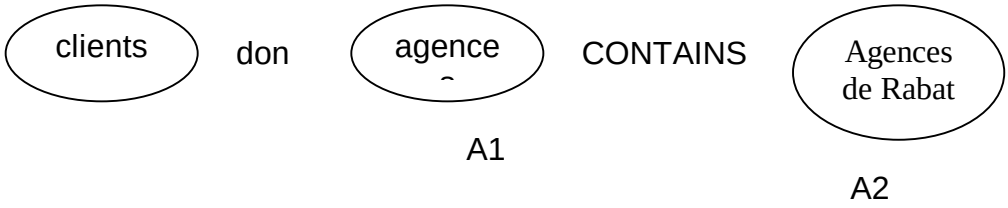
SELECT agence FROM Banque WHERE avoir > ANY
(SELECT avoir FROM Banque WHERE agence = "Bd Mohamed V")

- >ALL, >=ALL, <ALL, <=ALL, <>ALL, =ALL : comparaison à "Tout".

SELECT agence FROM Banque WHERE avoir >ALL
(SELECT avoir FROM Banque WHERE ville = "CASA"

(requête SELECT) [NOT] CONTAINS (requête SELECT)

Tester si un ensemble de tuples est inclus (n'est pas inclus) dans un autre.



SELECT client FROM Dépôt WHERE A1 CONTAINS A2

SELECT client FROM Dépôt S WHERE
(SELECT agence FROM Dépôt T WHERE S.client = T.client)
CONTAINS (SELECT agence FROM Banque WHERE ville = "RABAT")

Fonctions d'agrégat :

Des fonctions s'appliquant sur des groupes de données :

- COUNT(*)
- SUM(nom colonne)
- AVG(nom colonne)
- MIN(nom colonne)
- MAX(nom colonne)

Exemples :

- SELECT MIN(position), MAX(position), AVG(position) FROM Dépôt WHERE agence = "My Youssef"
- SELECT COUNT(*) FROM Clientèle
- SELECT client FROM Dépôt WHERE agence = "Bd, Mohamed V" AND 0 = (SELECT COUNT(*) FROM Clientèle WHERE Dépôt.client = Clientèle.client)
- SELECT client FROM Dépôt WHERE agence = "Bd, Mohammed V" AND NOT EXISTS (SELECT * FROM Clientèle WHERE Dépôt.client = Clientèle.client)

La clause GROUP BY :

Elle combine des enregistrements dont les valeurs sont identiques dans la liste de champs spécifiée en un seul enregistrement. Pour effectuer un regroupement, tous les champs de la liste de champs SELECT doivent être spécifiés soit dans la clause GROUP BY, soit en tant qu'arguments d'une fonction de regroupement SQL, telle que **Sum** (la somme d'un champ) ou **Count** (Compter le nombre d'enregistrements) dans l'instruction SELECT.

GRUOP BY Liste de groupes

L'exemple suivant regroupe les positions de chaque agence et calcule la moyenne de leurs positions :

SELECT agence, AVG(position) FROM Dépôt GROUP BY agence

Condition HAVING :

La clause HAVING affiche tous les enregistrements regroupés par la clause GROUP BY qui satisfont aux conditions de la clause HAVING.

GRUOP BY Liste de groupes HAVING condition

```
SELECT agence, AVG(position) FROM Dépôt
GROUP BY agence HAVING AVG(position) >1200
```

Utilisez la clause WHERE pour exclure les lignes qui ne doivent pas être regroupées et la clause HAVING pour filtrer des enregistrements après leur regroupement.

Classement des données :

ORDER BY Liste de colonnes

Elle détermine l'ordre de tri des enregistrements extraits par l'instruction SELECT.

Dans l'exemple suivant, les enregistrements contenus dans la table Crédit, seront triés dans l'ordre ascendant sur la base du client :

SELECT * FROM Crédit ORDER BY client ASC

Le mot **ASC** est facultatif, il indique l'ordre de tri, par défaut c'est **DESC** (descendant).

Mise à jour des données :

Insertion :

INSERT INTO nom_table [(liste_colonnes)] VALUES (liste_valeurs)

- INSERT INTO Dépôt VALUES ("Kays", 900532, "Radi", 20000)
- INSERT INTO Dépôt SELECT agence, prêt, client, 20000 FROM crédit WHERE agence="gare centrale"

Le nombre et les domaines des colonnes doivent être identiques à ceux de la table d'insertion.

Suppression :

DELETE FROM nom [WHERE condition]

WHERE condition est omise => nom est entièrement supprimée.

- DELETE FROM Crédit
- DELETE FROM Dépôt WHERE agence ="gare centrale"
- DELETE FROM Crédit WHERE montant BETWEEN 0 AND 1000
- DELETE FROM Dépôt WHERE agence IN (SELECT agence FROM Banque WHERE ville ="RABAT")

Modification :

Modifier certaines valeurs de tuples définis par *condition*.

UPDATE nom SET nom_colonne = expression [WHERE condition]

Exemples :

- UPDATE Dépôt SET position = position *1,06
- UPDATE Dépôt SET position = position * 1,6 WHERE position > 10000