Cours Bases de données
3ème Année Systèmes d'Information

Chapitre 04
Structured Query Language

Fouad DAHAK
Enseignant-Chercheur
Chargé de cours Bases de données
Ecole Nationale Supérieure d'Informatique (ESI)
(f_dahak@esi.dz – http://dahak.esi.dz)

Table des matières

1. Introduction

Ce polycopié présente un résumé succinct des composantes principales du langage SQL92. Dans certain cas nous donneront des exemples avec MYSQL.

Les différentes versions de SQL :

- SQL1 86: la base
- SQL1 89: l'intégrité
- SQL2 92: la nouvelle norme
- SQL3 99: les évolutions objets

SQL est dérivé de l'algèbre relationnelle et de SEQUEL, Il a été intégré à SQL/DS, DB2, puis ORACLE, INGRES, … La plupart des systèmes supportent SQL1 complet.

2. Composantes du langage SQL

Le SQL est composé de cinq grandes parties:

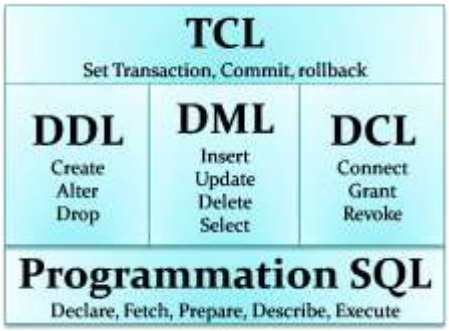
La définition des éléments d'une base de données;

La manipulation des données,

La gestion des droits d'accès,

La gestion des transactions,

La programmation dynamique.



3. Data Definition Language (Langage de Définition des Données) :

Partie de SQL qui permet de créer des bases de données, des tables, des index, des contraintes, etc. Elle traite de la création des schémas de bases de données.

3.1. Create Database

La requête create database est totalement dépendante du SGBD utilisé, car ça concerne la manière dont les données sont enregistrées physiquement sur le disque. C'est pourquoi, on ne retrouve pas beaucoup de détail dans la norme SQL 92. Syntaxe sous MYSQL :

```
CREATE DATABASE [IF NOT EXISTS] db_name
[create_specification [, create_specification] ...]
```

```
create_specification:
[DEFAULT] CHARACTER SET charset_name
| [DEFAULT] COLLATE collation_name
```

Exemple

```
CREATE DATABASE IF NOT EXISTS mabase
```

3.2. Create Table

Create table crée une table de nom NOM_TABLE dans la base de données courante. La syntaxe de cette instruction est respectée par tous les SGBD, la différence qu'on peut trouver est au niveau des types de colonnes utilisés.

```
CREATE TABLE nomtable(
{nomcolonne type [contrainte_colonne […]]
| contrainte _table} [,…]
)
```

Contrainte colonne peut être:

- NOT NULL
- UNIQUE
- PRIMARY KEY
- DEFAULT value

Contrainte_table peut être:

FOREIGN KEY(référence_colonne) REFERENCES référence_table(reference_colonne)

Exemple

```
CREATE TABLE avion(
num_avion  smallint primary key,
Type  varchar(10) not null,
constructeur  varchar(20) not null,
Capacite  smallint  check(capacite>0),
compagnie  varchar(30)  not null);
```

3.3. DROP Database | Table

Drop est une instruction qui permet de supprimer une base de données ou une table.

```
DROP DATABSE  mabase
DROP TABLE  matable
```

3.4. ALTER TABLE

Alter permet de modifier la structure d'une table. On peut ajouter, supprimer ou modifier des colonnes ou les index de la table.

```
ALTER TABLE nomtable {
        ADD CONSTRAINT contrainte
        DROP CONSTRAINT contrainte
        ADD COLUMN definitioncol
        DROP COLUMN colonne
        ALTER COLUMN{
                        SET DEFAULT valeur
                        DROP DEFAULT}
        }
```

Exemple

```
Alter  table  client  add column age int not null,
                drop column nom_clt,
                alter column prenom drop default;
```

3.5. Les contraintes d'intégrité

Les contraintes permettent d'exprimer des conditions devant être respectées par tous les tuples d'une table.

Les contraintes expriment les valeurs que peuvent prendre un attribut :

NOT NULL : l'attribut doit posséder une valeur

DEFAULT : valeur par défaut de l'attribut (quand il n'est pas défini)

UNIQUE : deux tuples ne peuvent pas avoir la même valeur pour cet attribut

CHECK : spécifie une condition devant être satisfaite par tous les tuples de la table.

PRIMARY KEY : Clé primaire.

FOREIGN KEY : Clé étrangère.

Exemples

```
create table avion(
        num_avion smallint primary key,
        type varchar(10) not null,
        constructeur varchar(20) not null,
        capacite smallint check(capacite>0),
        compagnie varchar(30) not null,
        id_tav int references type_av(id_tp)
        );
create table avion(
        num_avion smallint,
        type varchar(10) not null,
        constructeur varchar(20) not null,
        capacite smallint,
```

```
        compagnie varchar(30) not null,
        id_tav int,
        Primary key(num_avion),
         check(capacite>0),
        Foreign key (id_tav) references type_av(id_tp)
        );
create table avion(
        num_avion smallint,
        type varchar(10) not null,
        constructeur varchar(20) not null,
        capacite smallint,
        compagnie varchar(30) not null,
        id_tav int,
        CONSTRAINT `01_001`Primary key
        (num_avion),
        check(capacite>0),
        CONSTRAINT `01_002`Foreign key (id_tav)
        references type_av(id_tp)
        );
```

3.6. Les index

Un index permet d'accélérer la recherche sur un champ ou un ensemble de champs.

```
CREATE TABLE test (blob_col BLOB, INDEX(blob_col(10)));
CREATE INDEX part_of_name ON customer (name(10));
```

4. Data Manipulation Language (Langage de Manipulation des Données)

Partie de SQL qui traite les données (extraction, ajout, suppression, modification) dans les tables.

4.1. INSERT

INSERT permet d'insérer des lignes dans la table nom_table. On peut ne pas spécifier la définition de la table à condition que les valeurs correspondent aux champs de la table et dans l'ordre.

```
INSERT INTO nom_table(Définition de la table) VALUES (valeurs des colonnes)
Uniquement MySQL :
INSERT nom_table SET nom_col=valeur,…
```

Exemples:

```
INSERT INTO client(id_clt,nom_clt,prenom_clt) VALUES('21','MADJID','AMAR')
INSERT INTO client VALUES('21',AHMED','AMAR')
```

INSERT client **SET**(id_clt='21', nom_clt='AHMED', prenom_clt='AMAR' /* MySQL

//Insérer plusieurs lignes

INSERT INTO client(id_clt,nom_clt,prenom_clt) **VALUES** ('21',AHMED','AMAR'), ('21',MALIK',DJAMILA'), ('21',ILHEM',SAIDA'), ('21',ROUMI',MALIKA')

4.2. DELETE

DELETE permet de supprimer des lignes de la table nom_table. La suppression peut concerner toutes les lignes de la table comme on peut définir des critères de suppression dans la clause WHERE.

DELETE FROM *nom_table* **WHERE** *condition*

La clause **WHERE** est optionnelle, si on ne la spécifie pas on supprime toutes les liges de la table. Si non ce seront uniquement les lignes qui satisfont la clause WHERE qui seront supprimées.

Exemples :

Supprimer toutes les lignes

DELETE FROM client

Supprimer uniquement les clients dont l'âge est 39

DELETE FROM client WHERE age_clt>39

Supprimer les clients dont le nom est égal au prénom

DELETE FROM client WHERE nom_clt=prenom_clt

4.3. UPDATE

UPDATE permet de modifier des données d'une table. La modification peut concerner toutes les lignes de la table comme on peut définir des critères de modification dans la clause WHERE.

UPDATE *nom_table* **SET** nom_col=valeur,… **WHERE** *condition*

La clause **WHERE** est optionnelle, si on ne la spécifie pas on modifie toutes les liges de la table. Si non ce seront uniquement les lignes qui satisfont la clause WHERE qui seront modifiées.

Exemples :

Ajouter un point à tous les étudiants

UPDATE etudiant **SET** note=note+1

Ajouter un point uniquement aux étudiants dont la note est inférieure à 10

UPDATE etudiant **SET** note=note+1 **WHERE** note<10

Modifier la note telle que note est la moyenne entre TP et Note quand TP est supérieur à Note

UPDATE etudiant **SET** note=(note+tp)/2 **WHERE** note<tp

4.4. Select

La commande **SELECT** permet d'interroger la base de données, elle existe sous plusieurs formes et pour bien illustrer son fonctionnement nous allons utiliser dans toute la suite de ce cours la base de données suivante représentée avec son script MYSQL:

```
DROP DATABASE IF EXISTS `dbexemple`;
CREATE DATABASE `dbexemple`;
USE `dbexemple`;

DROP TABLE IF EXISTS `personne`;
CREATE TABLE `personne` (
  `id` varchar(10) NOT NULL,
  `nom` varchar(30) NOT NULL,
  `pnom` varchar(30) NOT NULL,
  `daten` datetime default NULL,
  `sexe` varchar(10) default NULL,
  `id_p` varchar(10) default NULL,
  `id_m` varchar(10) default NULL,
  `nbcnj` int(11) default '0',
  PRIMARY KEY  (`id`),
  FOREIGN KEY (`id_p`) REFERENCES `personne` (`id`),
  FOREIGN KEY (`id_m`) REFERENCES `personne` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `wilaya`;
CREATE TABLE `wilaya` (
  `id` varchar(2) NOT NULL,
  `lib` varchar(30) default NULL,
  PRIMARY KEY  (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `mariage`;
CREATE TABLE `mariage` (
  `id_epx` varchar(10) NOT NULL default '',
  `id_eps` varchar(10) NOT NULL default '',
  `datem` datetime default NULL,
  `wilm` varchar(2) default NULL,
  PRIMARY KEY  (`id_epx`,`id_eps`),
  KEY `id_eps` (`id_eps`),
  FOREIGN KEY (`id_epx`) REFERENCES `personne` (`id`),
  FOREIGN KEY (`id_eps`) REFERENCES `personne` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `resider`;
```

```
CREATE TABLE `resider` (
  `id_epx` varchar(10) NOT NULL default '',
  `id_eps` varchar(10) NOT NULL default '',
  `dater` datetime NOT NULL default '0000-00-00 00:00:00',
  `wilr` varchar(2) NOT NULL default '',
  `dated` datetime default NULL,
  PRIMARY KEY  (`id_epx`,`id_eps`,`wilr`,`dater`),
  KEY `id_eps` (`id_eps`),
  KEY `wilr` (`wilr`),
  FOREIGN KEY (`id_epx`,`id_eps`) REFERENCES `mariage` (`id_epx`,`id_eps`),
  FOREIGN KEY (`wilr`) REFERENCES `wilaya` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

4.4.1. Notations :

Tout ce qui est entre des crochets est facultatifs […] ;

La barre verticale | représente un choix 'ou' ;

Les trois points … représentent une suite d'éléments ;

4.4.2. Recherche de base

La recherche de base permet d'effectuer des opérations de projection et de restriction et la combinaison des deux.

```
Select [ALL | DISTINCT] champ1 [AS new_name], champs2 …
From nom_table [AS Alias]
[Where condition]
```

Dans la clause select on indique les colonnes sur lesquelles se fait la projection. Chaque colonne peut être renommée avec l'opérateur ***AS.*** Les colonnes sont séparées par des virgules. Dans le cas où on veut sélectionner toutes les colonnes des tables mentionnées dans la clause From on utilise le caractère astérisque (*). Pour différencier les colonnes des tables on mentionne le nom de la colonne précédée par le nom (ou alias) de la table comme suit : nom_table.nom_colonne. Par défaut ALL est utilisé mais si on souhaite ne pas afficher les doublons on utilise DISTINCT.

La clause **From** permet de spécifier les tables dans lesquelles la recherche sera effectuée, des alias peuvent être déclarés avec l'opérateur ***AS.***

La condition de la clause Where permet de spécifier les critères de sélection. Ces critères sont extrêmement riches dans SQL mais nous nous contenterons de présenter ici uniquement les plus utilisés.

Exemples de requêtes simples avec leurs résultats sur la base de données exemple :

```
mysql> Select *
    -> From Resider;
+--------+--------+---------------------+------+-------+
| id_epx | id_eps | dater               | wilr | dated |
+--------+--------+---------------------+------+-------+
| 1445   | 1289   | 2003-01-11 00:00:00 | 16   | NULL  |
| 28     | 3474   | 2004-01-11 00:00:00 | 10   | NULL  |
| 8834   | 3435   | 2004-01-11 00:00:00 | 16   | NULL  |
+--------+--------+---------------------+------+-------+
```

Au niveau de la clause Where, les conditions suivantes sont possibles :

[NOT] condition de base

Condition between

Condition in

Condition like

Condition null

Condition **AND | OR** Condition

Condition de base :

Colonne = | <> | < | <= | > | >= Constante

```
mysql> Select id_epx,id_eps,dater
    -> From Resider
    -> Where (wilr='16')And(dater>'2003-01-01');
+--------+--------+---------------------+
| id_epx | id_eps | dater               |
+--------+--------+---------------------+
| 1445   | 1289   | 2003-01-11 00:00:00 |
| 8834   | 3435   | 2004-01-11 00:00:00 |
+--------+--------+---------------------+
```

Condition Between :

L'opérateur Between permet de tester l'appartenance à une plage de valeurs.

Colonne Between x and y

```
mysql> Select id_epx,id_eps,dater
    -> From Resider
    -> Where dater Between '2003-01-01' And '2004-01-01';
+--------+--------+---------------------+
| id_epx | id_eps | dater               |
+--------+--------+---------------------+
| 1445   | 1289   | 2003-01-11 00:00:00 |
+--------+--------+---------------------+
```

Condition IN:

L'opérateur IN permet de tester l'appartenance à un ensemble de valeurs. L'ensemble peut être spécifié explicitement comme il peut être le résultat d'une autre requête imbriquée.

Colonne IN (v1,v2,…,vn)

```
mysql> Select id_epx,id_eps,dater
    -> From Resider
    -> Where wilr in ('01','10','06');
+--------+--------+---------------------+
| id_epx | id_eps | dater               |
+--------+--------+---------------------+
| 28     | 3474   | 2004-01-11 00:00:00 |
+--------+--------+---------------------+
```

Condition Like :

L'opérateur like permet de comparer des chaînes de caractères.

Colonne [not] like 'modèle de chaîne'

Modèle de chaîne peut contenir n'importe quel caractère plus deux caractères spéciaux :

'-' (au niveau de mysql c'est le caractère '_' qui est utilisé à la place du '-') remplace un caractère et '%' remplace plusieurs caractères.

```
mysql> select id,nom,pnom
    -> From personne
    -> Where nom like 'a%';
+------+------+---------------+
| id   | nom  | pnom          |
+------+------+---------------+
| 3435 | ABBA |               |
| 8720 | ABBA | MOHAMED RABIE |
+------+------+---------------+
mysql> select id,nom,pnom
    -> From personne
    -> Where nom like 'A__A';
+------+------+---------------+
| id   | nom  | pnom          |
+------+------+---------------+
| 3435 | ABBA |               |
| 8720 | ABBA | MOHAMED RABIE |
+------+------+---------------+
```

Condition Null :

Permet de vérifier si une colonne est null ou non.

Colonne IS [NOT] NULL

```
mysql> select *
    -> From Resider
    -> Where dated is NULL;
+--------+--------+---------------------+------+-------+
| id_epx | id_eps | dater               | wilr | dated |
+--------+--------+---------------------+------+-------+
| 1445   | 1289   | 2003-01-11 00:00:00 | 16   | NULL  |
| 28     | 3474   | 2004-01-11 00:00:00 | 10   | NULL  |
| 8834   | 3435   | 2004-01-11 00:00:00 | 16   | NULL  |
+--------+--------+---------------------+------+-------+
```

4.4.3. Recherche avec jointure

Il existe plusieurs manières d'exprimer des jointures entre plusieurs tables avec SQL. Les jointures vues dans l'algèbre relationnelles sont toutes implémentées dans pratiquement tous les SGBD avec quelques différences de syntaxe pour certains.

La syntaxe globale de la jointure est la suivante :

reference_table, reference_table

```
reference_table [CROSS] JOIN reference_table
reference_table [INNER] JOIN reference_table ON condition_jointure
reference_table NATURAL JOIN reference_table
reference_table LEFT | RIGHT | FULL [OUTER] JOIN reference_table ON condition_jointure
reference_table NATURAL [LEFT | RIGHT | FULL [OUTER]] JOIN reference_table
```

Produit cartésien :

On peut exprimer le produit cartésien avec une simple requête entre deux tables sans la clause where (ligne 1 de la syntaxe) et avec la clause CROSS JOIN (Ligne 2).

```
mysql> select Mariage.*
    -> From Mariage,Resider;
+--------+--------+---------------------+------+
| id_epx | id_eps | datem               | wilm |
+--------+--------+---------------------+------+
| 1445   | 1289   | 2002-12-11 00:00:00 | 38   |
| 1445   | 1289   | 2002-12-11 00:00:00 | 38   |
| 1445   | 1289   | 2002-12-11 00:00:00 | 38   |
| 213    | 5774   | 1973-12-22 00:00:00 | 40   |
| 213    | 5774   | 1973-12-22 00:00:00 | 40   |
| 213    | 5774   | 1973-12-22 00:00:00 | 40   |
| 213    | 896    | 1979-04-30 00:00:00 | 35   |
| 213    | 896    | 1979-04-30 00:00:00 | 35   |
| 213    | 896    | 1979-04-30 00:00:00 | 35   |
| 28     | 3474   | 2009-04-07 00:00:00 | 37   |
| 28     | 3474   | 2009-04-07 00:00:00 | 37   |
```

```
mysql> select Mariage.*
    -> From Mariage Cross Join Resider;
+--------+--------+---------------------+------+
| id_epx | id_eps | datem               | wilm |
+--------+--------+---------------------+------+
| 1445   | 1289   | 2002-12-11 00:00:00 | 38   |
| 1445   | 1289   | 2002-12-11 00:00:00 | 38   |
| 1445   | 1289   | 2002-12-11 00:00:00 | 38   |
| 213    | 5774   | 1973-12-22 00:00:00 | 40   |
| 213    | 5774   | 1973-12-22 00:00:00 | 40   |
| 213    | 5774   | 1973-12-22 00:00:00 | 40   |
| 213    | 896    | 1979-04-30 00:00:00 | 35   |
| 213    | 896    | 1979-04-30 00:00:00 | 35   |
| 213    | 896    | 1979-04-30 00:00:00 | 35   |
| 28     | 3474   | 2009-04-07 00:00:00 | 37   |
| 28     | 3474   | 2009-04-07 00:00:00 | 37   |
```

Thêta jointure :

La thêta-jointure peut être exprimée avec une requête simple avec une condition sur les colonnes communes des deux tables dans la clause Where, ou bien en utilisant la clause INNER JOIN (ligne 3).

```
mysql> Select P.nom,P.pnom
    -> From Personne P, Resider R
    -> Where P.id=R.id_epx;
+-----------+-----------+
| nom       | pnom      |
+-----------+-----------+
| ZERIMECHE | MOAHMED   |
| REKAB     | MED YAZID |
| TEBBAKH   | EL HAIN   |
+-----------+-----------+
```

```
mysql> Select P.nom,P.pnom
    -> From Personne P Inner Join Resider R On P.id=R.id_epx;
+-----------+-----------+
| nom       | pnom      |
+-----------+-----------+
| ZERIMECHE | MOAHMED   |
| REKAB     | MED YAZID |
| TEBBAKH   | EL HAIN   |
+-----------+-----------+
```

On peut utiliser USING(liste des colonnes communes)à la place de ON dans le cas d'une équijointure.

Jointure naturelle :

La jointure naturelle peut être exprimée avec une thêta-jointure en mentionnant dans la clause USING toutes les colonnes en commun et dans la clause SELECT toutes les colonnes une seule fois (donc pas de *). Ou bien en utilisant la clause NATURAL JOIN (ligne 4)

```
mysql> Select Mariage.*
    -> From Mariage Natural Join Resider;
+--------+--------+---------------------+------+
| id_epx | id_eps | datem               | wilm |
+--------+--------+---------------------+------+
| 28     | 3474   | 2009-04-07 00:00:00 | 37   |
| 1445   | 1289   | 2002-12-11 00:00:00 | 38   |
| 8834   | 3435   | 1993-10-09 00:00:00 | 42   |
+--------+--------+---------------------+------+
```

Jointure Externe :

On distingue trois types de jointures externes : la jointure externe à gauche, la jointure externe à droite et la jointure externe globale. Elles sont obtenues avec la clause left | right | full outer join …ON condition (ligne 5). Dans la plupart des SGBD le mot clé OUTER est facultatif comme pour le mot clé INNER.

```
mysql> Select Mariage.*
    -> From Mariage Left Outer Join Resider Using(id_eps,id_epx);
+--------+--------+---------------------+------+
| id_epx | id_eps | datem               | wilm |
+--------+--------+---------------------+------+
| 1445   | 1289   | 2002-12-11 00:00:00 | 38   |
| 213    | 5774   | 1973-12-22 00:00:00 | 40   |
| 213    | 896    | 1979-04-30 00:00:00 | 35   |
| 28     | 3474   | 2009-04-07 00:00:00 | 37   |
| 3236   | 7172   | 1997-04-13 00:00:00 | 22   |
| 5022   | 5911   | 2007-05-23 00:00:00 | 47   |
| 5044   | 4373   | 2008-10-20 00:00:00 | 42   |
| 6041   | 5953   | 1994-11-13 00:00:00 | 31   |
| 7793   | 7712   | 1988-11-16 00:00:00 | 30   |
| 8638   | 9090   | 2000-02-16 00:00:00 | 23   |
| 8720   | 6777   | 1985-05-29 00:00:00 | 36   |
| 8834   | 3435   | 1993-10-09 00:00:00 | 42   |
| 8834   | 5280   | 1974-05-10 00:00:00 | 07   |
| 8834   | 7425   | 1976-06-23 00:00:00 | 28   |
| 8834   | 7639   | 2008-03-14 00:00:00 | 18   |
+--------+--------+---------------------+------+
```

4.4.4. Recherche avec Tri du résultat

Pour trier le résultat d'une recherche on utilise la clause ORDER BY juste après la clause Where.

```
Order By colonne [desc]
```

Desc est spécifié dans le cas d'un tri décroissant.

```
mysql> Select id,nom,pnom
    -> From Personne
    -> Where id>20
    -> Order By nom desc;
+------+------------+----------------+
| id   | nom        | pnom           |
+------+------------+----------------+
| 7712 | ZOUFOUL    | NAAMA          |
| 28   | ZERIMECHE  | MOAHMED        |
| 8834 | TEBBAKH    | EL HAIN        |
| 1289 | RIMOUCHE   | HOUSNA         |
| 1445 | REKAB      | MED YAZID      |
| 6041 | NAOUS      | ASMANE         |
| 8638 | MELKIA     | RAMTANE        |
| 896  | LYFD       | AYADA          |
```

4.4.5. Les expressions SQL

Les expressions SQL permettent d'enrichir les clauses de sélection de la clause SELECT et les conditions de la clause WHERE. Nous ne donnerons pas tous les types d'expressions existants, nous présenterons uniquement quelques exemples d'utilisation de ces expressions sur notre base de données exemple.

```
mysql> Select id,nom,pnom,(datediff(now(),daten) DIV 365) As Age
    -> From Personne;
+------+------------+----------------+------+
| id   | nom        | pnom           | Age  |
+------+------------+----------------+------+
| 110  | BOUFENARA  | REBAI          |   20 |
| 1289 | RIMOUCHE   | HOUSNA         |    1 |
| 1445 | REKAB      | MED YAZID      |   37 |
| 213  | DAHRI      | KHEIRDDINE     |   24 |
| 28   | ZERIMECHE  | MOAHMED        |    5 |
| 3236 | BELHADJ    | EL KHEIR       |   36 |
| 3435 | ABBA       |                |   27 |
| 3474 | BEN NADJEM | FAWZIA         |   34 |
| 4373 | BOUCHETOB  | CHOUKRIA       |   25 |
```

4.4.6. Groupement de lignes

Les expressions utilisées dans la clause Select et Where font des comparaisons entre les valeurs des colonnes d'une ligne donnée. Mais des fois on a besoin d'effectuer de calculs sur des groupes de lignes comme par exemple l'âge moyen des personnes mariées par sexe…etc. Dans SQL la clause GROUP BY nous permet d'effectuer ces calculs. La clause Group by doit être utilisée quand on utilise des fonctions de calcul au niveau de la clause Select, sinon le résultat n'aurait aucune signification. La clause HAVING nous permet d'exprimer des conditions sur le groupe de lignes généré.

```
Select colonnes_de_groupement, fonction(colonne)…
From nom_table
Where condition
Groupe by colonne_de_groupement
Having condition_sur_fonction(colonne)
```

Exemples :

```
mysql> Select Count(*) From Personne;
+----------+
| Count(*) |
+----------+
|       28 |
+----------+
```

```
mysql> Select sexe,count(id) From Personne Group By Sexe;
+----------+-----------+
| sexe     | count(id) |
+----------+-----------+
| FEMININ  |        17 |
| MASCULIN |        11 |
+----------+-----------+
mysql> Select Sexe, Avg(datediff(now(),daten) DIV 365) As MoyenneAge
    -> From Personne
    -> Group By Sexe
    -> Having MoyenneAge>20;
+----------+------------+
| Sexe     | MoyenneAge |
+----------+------------+
| MASCULIN |    22.0000 |
+----------+------------+
```

4.4.7. Les requêtes imbriquées

Un des concepts de SQL qui contribue grandement à sa puissance et sa souplesse est la sélection imbriquée : une sélection imbriquée n'est rien d'autre qu'un bloc de qualification SFW encapsulé à l'intérieur d'un autre bloc de qualification.

On peut utiliser l'opérateur **IN**.

Exemple

```
mysql> Select *
    -> From Wilaya
    -> Where id in (Select wilr from resider);
+----+--------+
| id | lib    |
+----+--------+
| 10 | BOUIRA |
| 16 | ALGER  |
+----+--------+
```

Dans ce cas le SGBD commence d'abord par traiter la requête interne. Le résultat renvoyé est ensuite remplacé dans la requête externe.

Le résultat de l'évaluation de la condition IN est vrai si et seulement si la valeur de l'expression à gauche du IN est égale à au moins une des valeurs du résultat de la requête à droite de IN.

Le résultat est faux si la valeur de l'expression à gauche de IN est différente de toutes les valeurs du résultat de la droite.

Condition ALL|ANY|SOME

Dans la clause Where on utilise des comparaisons entres des colonnes de plusieurs tables et des constantes ou bien d'autres colonnes. Dans un cas général, on peut utiliser des sous requêtes retournant une seule valeur au niveau des conditions. Mais quand les sous requêtes retournent plus d'une valeur il faut utiliser les opérateurs ALL, ANY ou SOME.

ALL renvoie vrai uniquement si la comparaison est vraie pour toutes les valeurs retournées par la sous requête ou bien le résultat de la sous requête est vide.

ANY renvoie vrai quand la condition indiquée est vraie pour au moins une des valeurs retournées par la sous requête.

Exemple

```
mysql> Select Count(id)
    -> From Personne
    -> Where (datediff(now(),daten) DIV 365)>
    -> (Select Avg(datediff(now(),daten) DIV 365) From Personne);
+-----------+
| Count(id) |
+-----------+
|        15 |
+-----------+
```

```
mysql> Select *
    -> From Resider
    -> Where wilr>ANY(Select id From Wilaya Where lib like 'b%');
+--------+--------+---------------------+------+-------+
| id_epx | id_eps | dater               | wilr | dated |
+--------+--------+---------------------+------+-------+
| 1445   | 1289   | 2003-01-11 00:00:00 | 16   | NULL  |
| 28     | 3474   | 2004-01-11 00:00:00 | 10   | NULL  |
| 8834   | 3435   | 2004-01-11 00:00:00 | 16   | NULL  |
+--------+--------+---------------------+------+-------+
```

La condition EXISTS

La condition EXISTS renvoi vrai quand l'évaluation de la sous requête n'est pas vide.

```
mysql> Select Count(*)
    -> From Personne
    -> Where Exists (
    ->                   Select id_epx
    ->                   From Mariage
    ->                   Where Mariage.id_epx=Personne.id
    ->               );
+----------+
| Count(*) |
+----------+
|       11 |
+----------+
```

4.5. Les vues

Présentation :

Une vue est une table virtuelle dont le contenu est défini par une requête. Une vue ressemble à une table réelle, avec un ensemble de colonnes nommées et de lignes de données. Toutefois, une vue n'existe pas en tant qu'ensemble de valeurs de données stocké dans une base de données. Les lignes et les colonnes de données proviennent de tables référencées dans la requête qui définit la vue et sont produites dynamiquement lorsque la vue est référencée.

Une vue fait office de filtre sur les tables sous-jacentes qui y sont référencées. La requête qui définit la vue peut émaner d'une ou de plusieurs tables ou d'autres vues de la base de données en cours ou d'une autre base de données locale ou distante. Les requêtes distribuées peuvent également être employées pour définir des vues qui utilisent des données issues de plusieurs sources hétérogènes. Cela est particulièrement utile si vous souhaitez combiner des données de structure similaire issues de différents serveurs, dont chacun héberge des données relatives à une région différente de votre organisation.

L'interrogation au moyen de vues ne fait l'objet d'aucune restriction, et la modification de données par le biais de vues, de quelques-unes seulement.

Création d'une Vue

```
mysql> Create View HommeMaries
    -> As
    -> Select *
    -> From Personne
    -> Where sexe='Masculin' And
    ->  id IN (
    ->          Select id_epx
    ->          From Mariage);

mysql> Select Id,Nom,Pnom
    -> From HommeMaries;
+------+------------+---------------+
| id   | nom        | pnom          |
+------+------------+---------------+
| 1445 | REKAB      | MED YAZID     |
| 213  | DAHRI      | KHEIRDDINE    |
| 28   | ZERIMECHE  | MOAHMED       |
| 3236 | BELHADJ    | EL KHEIR      |
| 5022 | CHAOU      | BEN AZZOUZ    |
| 5044 | CHIKH DAHO | MOHAMED ARAB  |
| 6041 | NAOUS      | ASMANE        |
| 8638 | MELKIA     | RAMTANE       |
| 8720 | ABBA       | MOHAMED RABIE |
| 8834 | TEBBAKH    | EL HAIN       |
+------+------------+---------------+
```

Modification d'une vue

On ne peut modifier une vue que si :

1. Elle ne référence qu'une seule table,
2. Tous les champs requis dans la table source sont mentionnés dans le Select de la vue.
3. Les valeurs mentionnées respectent les conditions du where de la vue dans le cas d'utilisation de l'option ***With Check Option***.

Suppression d'une vue

Drop view nom_vue ;

5. Data Control Language (Langage de contrôle des Données)

Création d'un nouvel utilisateur :

```
mysql> Create user etudiant identified by 'etudiant';
```

Supprimer un utilisateur :

```
mysql> drop user etudiant;
```

Ajouter des droits niveau table :

```
mysql> Grant Select, Delete, Update On dbexemple.personne to etudiant;
```

Ajouter des droits niveau colonne :

```
mysql> Grant Update(id_epx,id_eps) On dbexemple.mariage to etudiant;
```

Enlever des droits à un utilisateur

```
mysql> Revoke All On dbexemple.personne from etudiant;
```

Annexe 1. Mots réservés du SQL standard 92

ABSOLUTE
ACTION
ADD
ALL
ALLOCATE
ALTER
AND
ANY
ARE
AS
ASC
ASSERTION
AT
AUTHORIZATION
AVG
BEGIN
BETWEEN
BIT
BIT_LENGTH
BOTH
BY
CASCADE
CASCADED
CASE
CAST
CATALOG
CHAR
CHARACTER
CHAR_LENGTH
CHARACTER_LENGTH
CHECK
CLOSE
COALESCE
COLLATE
COLLATION
COLUMN
COMMIT
CONNECT
CONNECTION
CONSTRAINT
CONSTRAINTS
CONTINUE
CONVERT
CORRESPONDING
COUNT
CREATE
CROSS
CURRENT
CURRENT_DATE
CURRENT_TIME
CURRENT_TIMESTAMP
CURRENT_USER
CURSOR
DATE
DAY
DEALLOCATE
DEC
DECIMAL
DECLARE
DEFAULT
DEFERRABLE
DEFERRED
DELETE
DESC
DESCRIBE
DESCRIPTOR
DIAGNOSTICS
DISCONNECT
DISTINCT
DOMAIN
DOUBLE
DROP
ELSE
END

END-EXEC
ESCAPE
EXCEPT
EXCEPTION
EXEC
EXECUTE
EXISTS
EXTERNAL
EXTRACT
FALSE
FETCH
FIRST
FLOAT
FOR
FOREIGN
FOUND
FROM
FULL
GET
GLOBAL
GO
GOTO
GRANT
GROUP
HAVING
HOUR
IDENTITY
IMMEDIATE
IN
INDICATOR
INITIALLY
INNER
INPUT
INSENSITIVE
INSERT
INT
INTEGER
INTERSECT
INTERVAL
INTO
IS
ISOLATION
JOIN
KEY
LANGUAGE
LAST
LEADING
LEFT
LEVEL
LIKE
LOCAL
LOWER
MATCH
MAX
MIN
MINUTE
MODULE
MONTH
NAMES
NATIONAL
NATURAL
NCHAR
NEXT
NO
NOT
NULL
NULLIF
NUMERIC
OCTET_LENGTH
OF
ON
ONLY
OPEN
OPTION
OR
ORDER
OUTER
OUTPUT

OVERLAPS
PAD
PARTIAL
POSITION
PRECISION
PREPARE
PRESERVE
PRIMARY
PRIOR
PRIVILEGES
PROCEDURE
PUBLIC
READ
REAL
REFERENCES
RELATIVE
RESTRICT
REVOKE
RIGHT
ROLLBACK
ROWS
SCHEMA
SCROLL SECOND
SECTION
SELECT
SESSION
SESSION_USER
SET
SIZE
SMALLINT
SOME
SPACE
SQL
SQLCODE
SQLERROR
SQLSTATE
SUBSTRING
SUM
SYSTEM_USER
TABLE
TEMPORARY
THEN
TIME
TIMESTAMP
TIMEZONE_HOUR
TIMEZONE_MINUTE
TO
TRAILING
TRANSACTION
TRANSLATE
TRANSLATION
TRIM
TRUE
UNION
UNIQUE
UNKNOWN
UPDATE
UPPER
USAGE
USER
USING
VALUE
VALUES
VARCHAR
VARYING
VIEW
WHEN
WHENEVER
WHERE
WITH
WORK
WRITE
YEAR
ZONE

Annexe 2 : Les fonctions dans SQL

Agrégation statistique

Fonction	Description	Norme SQL	Mysql	SQL Server
AVG	Moyenne	O	O	O
COUNT	Nombre	O	O	O
MAX	Maximum	O	O	O
MIN	Minimum	O	O	O
SUM	Total	O	O	O

Fonction "système"

Fonction	Description	Norme SQL	MySQL	SQL Server
CURRENT_DATE	Date courante	O	O	N
CURRENT_TIME	Heure courante	O	O	N
CURRENT_TIMESTAMP	Date et heure courante	O	O	O
CURRENT_USER	Utilisateur courant	O	N	O
SESSION_USER	Utilisateur autorisé	O	X	O
SYSTEM_USER	Utilisateur système	O	X	O
CURDATE	Date du jour	N	O	N
CURTIME	Heure courante	N	O	N
DATABASE	Nom de la base de donnée courante	N	O	O
GETDATE	Heure et date courante	N	N	O
NOW	Heure et date courante	N	O	O
SYSDATE	Date et/ou heure courante	N	O	N
TODAY	Date du jour	N	N	N
USER	Utilisateur courant	N	O	O
VERSION	Version du SGBDR	N	O	N

Fonctions générales

Fonction	Description	Norme SQL	MySQL	SQL Server
CAST	Transtypage	O	O	O
COALESCE	Valeur non NULL	O	O	O
NULLIF	Valeur NULL	O	O	O
OCTET_LENGTH	Longueur en octet	O	O	N
DATALENGTH	Longueur	N	N	O
DECODE	Fonction conditionnelle	N	N	N
GREATEST	Plus grande valeur	N	O	N
IFNULL	Valeur non NULL	N	O	O
LEAST	Plus petite valeur	N	N	N
LENGTH	Longueur	N	O	O
NVL	Valeur non NULL	N	N	N
TO_CHAR	Conversion de données en chaîne	N	N	N
TO_DATE	Conversion en date	N	N	N
TO_NUMBER	Conversion en nombre	N	N	N

Fonctions de chaînes de caractères

Fonction	Description	Norme SQL	MySQL	SQL Server
||	Concaténation	O	X	N
CHAR_LENGTH	Longueur d'une chaîne	O	X	N
CHARACTER_LENGTH	Longueur d'une chaîne	O	O	O
COLLATE	Substitution à une séquence de caractères	O	N	N
CONCATENATE	Concaténation	O	N	O
CONVERT	Conversion de format de caractères	O	N	!
LIKE (prédicat)	Comparaison partielle	O	O	O
LOWER	Mise en minuscule	O	O	O
POSITION	Position d'une	O	O	N

	chaîne dans une sous chaîne			
SUBSTRING	Extraction d'une sous chaîne	O	O	N
TRANSLATE	Conversion de jeu de caractères	O	N	N
TO_CHAR	Conversion de données en chaîne	N	N	N
TRIM	Suppression des caractères inutiles	O	O	N
UPPER	Mise en majuscule	O	O	O
CHAR	Conversion de code en caractère ASCII	N	O	O
CHAR_OCTET_LENGTH	Longueur d'une chaîne en octets	N	N	O
CHARACTER_MAXIMUM_LENGTH	Longueur maximum d'une chaîne	N	N	O
CHARACTER_OCTET_LENGTH	Longueur d'une chaîne en octets	N	N	O
CONCAT	Concaténation	N	O	O
ILIKE	LIKE insensible à la casse	N	N	N
INITCAP	Initiales en majuscule	N	N	N
INSTR	Position d'une chaîne dans une autre	N	O	N
LCASE	Mise en minuscule	N	O	O
LOCATE	Position d'une chaîne dans une autre	N	O	O
LPAD	Remplissage à gauche	N	O	N
LTRIM	TRIM à gauche	N	O	O
NCHAR	Conversion de code en caractère UNICODE	N	N	O
PATINDEX	Position d'un motif dans une	N	N	O

	chaîne			
REPLACE	Remplacement de caractères	N	O	O
REVERSE	Renversement	N	O	O
RPAD	Remplissage à droite	N	O	N
RTRIM	TRIM à droite	N	O	O
SPACE	Génération d'espaces	N	O	O
SUBSTR	Extraction d'une sous chaîne	N	N	N
UCASE	Mise en majuscule	N	O	O

Fonctions de chaînes de bits

Fonction	**Description**	**Norme SQL**	**MySQL**	**SQL Server**
BIT_LENGTH	Longueur en bit	O	N	N
&	"et" pour bit logique	N	?	O
|	"ou" pour bit logique	N	?	O
^	"ou" exclusif pour bit logique	N	?	O

Fonctions numériques

Fonction	**Description**	**Norme SQL**	**MySQL**	**SQL Server**
%	Modulo	N	O	O
+ - * / ( )	Opérateurs et parenthésage	O	O	O
ABS	Valeur absolue	N	O	O
ASCII	Conversion de caractère en code ASCII	N	O	O
ASIN	Angle de sinus	N	O	O
ATAN	Angle de tangente	N	O	O
CEILING	Valeur approchée haute	N	O	O
COS	Cosinus	N	O	O
COT	Cotangente	N	O	O
EXP	Exponentielle	N	O	O
FLOOR	Valeur approchée basse	N	O	O
LN	Logarithme népérien	N	N	N

LOG	Logarithme népérien	N	O	O
LOG(n,m)	Logarithme en base n de m	N	N	N
LOG10	Logarithme décimal	N	O	O
MOD	Modulo	N	O	O
PI	Pi	N	O	O
POWER	Elévation à la puissance	N	O	O
RAND	Valeur aléatoire	N	O	O
ROUND	Arrondi	N	O	O
SIGN	Signe	N	O	O
SIN	Sinus	N	O	O
SQRT	Racine carrée	N	O	O
TAN	Tangente	N	O	O
TRUNC	Troncature	N	N	N
TRUNCATE	Troncature	N	O	O
UNICODE	Conversion de caractère en code UNICODE	N	N	O

Fonctions temporelles

Fonction	**Description**	**Norme SQL**	**MySQL**	**SQL Server**
EXTRACT	Partie de date	O	O	N
INTERVAL (opérations sur)	Durée	O	N	N
OVERLAPS (prédicat)	Recouvrement de période	O	N	N
ADDDATE	Ajout d'intervalle à une date	N	O	N
AGE	Age	N	N	N
DATE_ADD	Ajout d'intervalle à une date	N	O	N
DATE_FORMAT	Formatage de date	N	O	N
DATE_PART	Partie de date	N	N	N
DATE_SUB	Retrait d'intervalle à une date	N	O	N
DATEADD	Ajout de date	N	N	O

DATEDIFF	Retrait de date	N	N	O
DATENAME	Nom d'une partie de date	N	N	O
DATEPART	Partie de date	N	N	O
DAY	Jour d'une date	N	N	O
DAYNAME	Nom du jour	N	O	O
DAYOFMONTH	Jour du mois	N	O	N
DAYOFWEEK	Jour de la semaine	N	O	N
DAYOFYEAR	Jour dans l'année	N	O	N
HOUR	Extraction de l'heure	N	O	O
LAST_DAY	Dernier jour du mois	N	N	N
MINUTE		N	O	O
MONTH	Mois d'une date	N	O	O
MONTH_BETWEEN	MONTH_BETWEEN	N		
MONTHNAME	Nom du mois	N	O	O
NEXT_DAY	Prochain premier jour de la semaine	N	N	N
SECOND	Extrait les secondes	N	O	O
SUBDATE	Retrait d'intervalle à une date	N	O	N
WEEK	Numéro de la semaine	N	O	O
YEAR	Année d'une date	N	O	O

Prédicat, opérateurs et structures diverses

Fonction	**Description**	**Norme SQL**	**MySQL**	**SQL Server**
CASE	Structure conditionnelle	O	O	O
IS [NOT] TRUE	Vrai	O	N	N
IS [NOT] FALSE	Faux	O	N	N
IS [NOT] UNKNOWN	Inconnu	O	N	N
IS [NOT] NULL	NULL	O	O	O
INNER JOIN	Jointure interne	O	O	O

LEFT, RIGHT, FULL OUTER JOIN	Jointure externe	O	O	O
NATURAL JOIN	Jointure naturelle	O	O	N
UNION JOIN	Jointure d'union	O	N	N
LEFT, RIGHT, FULL OUTER NATURAL JOIN	Jointure naturelle externe	O	X	N
INTERSECT	Intersection (ensemble)	O	N	N
UNION	Union (ensemble)	O	N	O
EXCEPT	Différence (ensemble)	O	N	N
[NOT] IN	Liste	O	X	O
[NOT] BETWEEN	Fourchette		O	O
[NOT] EXISTS	Existence	O	N	O
ALL	Comparaison à toutes les valeurs d'un ensemble	O	N	O
ANY / SOME	Comparaison à au moins une valeur de l'ensemble	O	N	O
UNIQUE	Existance sans doublons	O	N	N
MATCH UNIQUE	Correspondance	O	N	N
row value construteur	Construteur de ligne valuées	O	N	N
MINUS	Différence (ensemble)	N	N	N
LIMITE	nombre de ligne retournée	N	LIMIT	TOP
identifiant de ligne		N	_rowid	N