BASE DE DONNEES

Pratique du langage SQL.

Speaker : L. Noël KOUKA M.

Table des Matières

CHAPITRE I : Rappels

1. Définition d'une Base des données

Une **base de données** est un ensemble de données (informations) qui ont été **stockées** de façon structurée et organisée sur un **support informatique** afin que leur contenu soit facilement **consultable** et **modifiable**.

Une base de données seule ne suffit donc pas, il est nécessaire d'avoir également :

- **Un système permettant de gérer cette base.**
- **Un langage pour transmettre des instructions à la base de données (par l'intermédiaire du système de gestion).**

2. Notion de systèmes de gestions

a. SGBD

Un système de gestion de base de données (SGBD) est un logiciel (ou un ensemble de logiciels) permettant de manipuler les données (informations) d'une base de données ; manipuler signifie sélectionner et afficher les informations tirées de cette base.

b. SGBD-R

Le R de SGBD signifie **Relationnel.** Un système de gestion de base de données relationnel est un SGBD qui implémente la théorie relationnelle. Par exemple **MySQL, PostgreSQL, Oracle Database …** implémentent cette théorie relationnelle ; c'est donc des SGBD-R.

N.B : Sachiez que dans un SGBD-R, les données sont contenues dans ce qu'on appelle des **relations**, qui sont représentées sous forme de **tables.** Une relation est composée de deux parties : l'**en-tête** et le **corps**. L'en-tête est composé de plusieurs attributs.

Ex : Numéro Nom Prénom Email
Quant au Corps, il s'agit d'un ensemble de **lignes** (ou **n-uplets**) composées d'autant d'éléments qu'il y a d'attributs.

c. Catégorisation des SGBD

Ces systèmes peuvent être catégorisés selon leur fonctionnement :

- **Système propriétaire :** *Oracle Database, Microsoft SQL Serveur, BD2, MaxDB, 4D, dBase, Informix, Sybase ;*
- ***Système libre* :** *MySQL, PostgreSQL, MariaDB, Firebird, Ingres, HSQLDB, Derby, Apache Derby ;*
- **Orienté – Objet :** *ZODB, db4o ;*
- **Embarqué :** *SQLite, Berkeley DB ;*
- **NoSQL :** *Cassandra, Redis, MongoDB, SimpleDB, Big Table, CouchDB, Couchbase, HBase, LevelDB, RethinkDB, Memcached ;*
- **Autre système :** *Access, OpenOffice.org Base, FileMaker, HyperFileSQL, Paradox, Neo4j, Riak, Voldemort.*

CHAPITRE II : Le langage SQL

1. Définition

Le **SQL** (Structured Query Language) est un **langage informatique** qui permet d'interagir avec des **bases de données relationnelles.** C'est le langage pour base de données le plus répandu et c'est bien-sûr celui utilisé par **MySQL**.

Il a été créé dans les années 1970 et c'est devenu standard en 1986 (pour la norme ANSI – 1987 en ce qui concerne la norme ISO).

2. Types de données

Il a été dit en amant que dans une base de données, les informations (données) sont enregistrées (stockées) dans des tables qui, sont elles-mêmes organisées en colonnes.

En SQL et même dans la majorité des langages informatiques, les données sont séparées en plusieurs types : texte, nombre entier, date… Lorsque l'on définit une colonne dans une table de la base, il est donc nécessaire de lui attribuer un type, et tous les enregistrements faits devront correspondre au type de la colonne.

a. Types Numériques

On peut subdiviser le type numérique en deux sous-catégories :

a-1 Nombres Entiers

Le type de données qui acceptent des nombres entiers comme valeur sont désignés par le mot-clé ***INT***, et ses déclinaisons : ***TINYINT, SMALLINT, MEDIUMINT*** et ***BIGINT***. La différence entre ces types est le nombre d'octets (donc la place en mémoire) réservés à la valeur du champ. Le tableau ci-dessous illustre cette différence ainsi que l'intervalle dans lequel la valeur peut-être comprise pour chaque type :

Type	Nombre d'octets	Minimum	Maximum
Tinyint	1	-128	127
Smallint	2	-32768	32767
Mediumint	3	-8388608	8388607
Int	4	-2147483648	2147483647
Bigint	5	-9223372036854775808	9223372036854775807

N.B : L'attribut ***UNSIGNED*** permet donc de masquer ou de ne pas préciser le signe d'une valeur, qu'elle soit positive ou négative mais dans les cas, nous aurons toujours une valeur positive. Alors, la longueur de l'intervalle reste la même et les valeurs restent positives, le minimum valant est 0. Pour le *Tinyint*, la valeur peut aller de 0 à 255.

- ## Limiter la taille d'affichage et l'attribut ZEROFILL

Il est possible de préciser le nombre de chiffre minimum à l'affichage de type INT ou de ses dérivés, il suffit donc de préciser ce nombre entre parenthèses : *INT(x) ;* mais cela ne change pas la capacité de stockage dans la colonne. Par exemple, on peut déclarer un **int (2)** tout en pouvant stocker **96875**.

Si vous ajoutez un nombre avec un nombre de chiffres inférieur au nombre défini, le caractère par défaut sera ajouté à gauche du chiffre afin qu'il prenne la bonne taille. Sans précision, le caractère par défaut est l'espace mais, en combinaison avec l'attribut *ZEROFILL*, lors de l'affichage, cette change donc le caractère par défaut par « 0 ».

Exemple : Déclaration : **INT (4) ZEROFILL**

Affichage :

Nombre stocké	Nombre affiché
45	0045
4156	4156
785164	785164

a-2 Nombres Décimaux

Cinq mots-clés permettent de stocker des nombres décimaux dans une colonne : ***Decimal, Numeric, Float, Real*** et ***Double.***

Numeric et **Decimal** sont équivalents et acceptent deux paramètres : la **précision** (nombre de chiffres significatif stocké : partie entière) et l'**échelle** (le nombre de chiffre après la virgule).

Exemple : Decimal (5. 3) on peut donc stocker des nombre de 5 chiffres significatifs maximum dont 3 chiffres après la virgule (12.354, -54.258, 89.2 ou -56).

b. Chaîne de caractère

- **Char (longueur) :** Chaîne de caractères de taille fixe longueur.

  Ex : Char (x) ; ici nous ne pouvons que stocker x valeurs.

- **Varchar (longueur) :** Chaîne de caractères de taille variable longueur.

  Ex : Char(x) ; ici nous pouvons stocker jusqu'à x valeurs c'est-à-dire dans l'intervalle de la longueur définie.

  - **Text :** Il est subdivisé en deux déclinaisons à savoir : Le **Mediumtext** et le **Longtext.**

c. Type Date

Dans ce type, on a : **Datetime, Time**, **Year** et **Timestamp.**

N.B : Une colonne de type **ENUM** est une colonne pour laquelle on définit un certain nombre de valeurs autorisées de type chaîne de caractères et si vous essayer d'introduire une chaîne de caractère non-autorisée, **Mysql** stockera une chaîne vide (‘ ‘) dans le champ. Le contraire se fait avec le SET.

Ex : Produit **ENUM** (‘lait’, ‘pomme’, ‘fruit’) ;

Produit **SET** (‘lait’, ‘pomme’, ‘fruit’, ‘boisson’, ‘céréale’). Ce sont des types propres à **Mysql**, à manipuler (utiliser) avec précaution (grande prudence).

3. Contraintes d'intégrité

- ***NOT NULL :*** Consiste à désigner l'obligation d'une valeur.

- ***NULL :*** Consiste à dire que le champ n'est pas à titre obligatoire lors des enregistrements (insertions). Par défaut, la valeur Null est acceptée.

- ***CONSTRAINT nom_i :*** Permet de nommer une contrainte, le nom est optionnel.

  Ex : Constraint nom_i constrainte_1…

- ***CONSTRAINT nom_c UNIQUE (att_i, att_j, …) :*** Impose que chaque n-uplet ait une combinaison de valeurs différente pour les attributs att_i, att_j, … Si une des valeurs pour ces attributs est NULL, la contrainte ne s'applique pas sur le n-uplet concerné.

3. Contraintes d'intégrité

- ***CONSTRAINT nom_c PRIMARY KEY (att_i, att_j, …) :*** Indique que l'ensemble d'attribut (att_i, att_j, …) sert d'identifiant principal (également appelé **clé primaire**) pour les n-uplets de la table. Il implique aussi NOT NULL et UNIQUE sur chacun des attributs (att_i, att_j, …) Il y a au maximum une contrainte PRIMARY KEY par table.

- ***CONSTRAINT nom_c FOREIGN KEY (att_1, … , att_k) REFERENCES table_cible (att_1, … , att_k) :*** Il s'agit ici d'une contrainte définissant une clé étrangère. Une clé étrangère est une référence vers la clé primaire d'une table.

- ***CONSTRAINT nom_c CHECK (condition) :*** La condition doit-être vérifiée par chaque n-uplet stocké dans la table. La forme (att IN ('val_1', 'val_2', …)) utilisée pour les types énumérés est un cas particulier de cette contrainte.

  Exemple : CHECK (réponse IN (V, F)) ;

CHAPITRE III : Les sous-ensembles du langage SQL et ses commandes

A. Sous-ensembles du SQL

Le langage SQL est un langage concret qui interagit avec la base de données. Il est subdivisé en trois (3) sous-ensemble à savoir :

- **Le langage de description des données (LDD) :** Permet de définir ou d'établir le schéma ou le squelette d'un environnement de stockage (database) des données informatiques.
- **Le langage de manipulation des données (LMD) :** Permet de manipuler ou consulter aisément les informations stockées dans la base des données.
- **Le langage de contrôle des données (LCD) :** Permet d'administrer la base des données et gérer les contrôles d'accès.

NB : Dans d'autre cas, on peut voir apparaitre un quatrième langage (**LID : Langage d'interrogation des données**) mais qui ne fait partie en réalité que du **LMD** ; donc inutile de les dissocier.

B. Commandes SQL de chaque sous-ensemble

1. Le langage de description des données (LDD)

Il comporte quatre (4) commandes à savoir :

CREATE : Permet de créer une base de donnée, une table, une vue…

- Pour la base des données

Syntaxe : *CREATE Database* *nom_db* ***;***

- **Pour la table**

Syntaxe : *CREATE Table* *nom_table (*

Att_1 type_1, Att_2 type_2, …) ***;***

- **Pour une vue**

Syntaxe : *CREATE View* *nom_vue* ***AS***

SELECT …

- **ALTER :** Permet d'apporter des modifications dans la structure d'une table encore appelée par ***Entité***. Il s'utilise avec : ADD, MODIFY, CHANGED, RENAME, DROP.

Syntaxe : *Alter Table* *nom_table* ***ADD*** *att type* ***[CONSTRAINT] ;***

Alter Table *nom_table* ***MODIFY*** *att nouveau_type*

[CONSTRAINT];

B. Commandes SQL de chaque sous-ensemble

1. Le langage de description des données (LDD)

Alter Table *nom_table* ***RENAME COLUMN*** *att* ***TO*** *nouvel_att* ***;***

Alter Table *nom_table* ***DROP COLUMN*** *att* ***;***

- **DROP :** Permet de faire la suppression d'une structure (base de données, table, vue, …).

Syntaxe : ***Drop Structure*** *nom_structure* ***;***

Ex : Drop Table nom_table ;

- **RENAME :** Permet de renommer une structure.

Syntaxe : ***Rename*** *ancien_nom* ***To*** *nouveau_nom* ***;***

Ex : **Rename** Vendeur **To** Commercant ;

2. Le langage de manipulation des données (LMD)

Il est composé de quatre (4) commandes à savoir :

- **INSERT :** Permet de faire des insertions (des enregistrements) dans les tables. Il est important de savoir que les insertions se font de plusieurs façons (insertion multiple).

Syntaxe : ***Insert into*** *nom_table (att_1, att_2,…)* ***Values*** *('val_1', 'val_2',…);*

Ex : **Insert into** Etudiant (mat_Et, nom_Et) **Values** ('eces-654', 'MABIALA') ;

- **UPDATE :** Permet de faire les mises à jour dans la base de données.

Syntaxe : ***Update*** *nom_table*

Set *att_1 = e1*

att_2 = e2

…

Where *condition* ***;***

Ex : **Update** Etudiant

Set frais_eco = frais_eco * 1.4

Where niveau = 'licence 3' ;

- **DELETE :** Permet de supprimer une colonne dans une relation (table).

Syntaxe : ***Delete From*** *nom_table*

Where condition ;

2. Le langage de manipulation des données (LMD)

Ex : Delete From Etudiant

Where mat_Et = ‘eces-1025’ ;

SELECT : Permet d’afficher les informations contenues dans la base des données. Cette requête ou commande SQL prend plusieurs forme selon les besoins demandés ; on peut l’utiliser avec les clauses : ***Where, Group By, Having*** et ***Order By***

Syntaxe : ***Select*** *att_1, att_2,…*

From *table_1, table_2,…*

Where *condition*

Group By *att_k, att_i,…*

Having *condition_groupe*

Order By *att_n, att_j,… ;*

3. Le langage de contrôle des données (LCD)

Il est composé de deux commandes à savoir :

- **GRANT:** Permet d'attribuer ou d'accorder un ou des privilèges aux utilisateurs.

Syntaxe : ***Grant*** *commande*

On *nom_table*

To *nom_user ;*

Ex : Grant Select

On Etudiant

To Malanda ;

- **REVOKE :** Permet de retirer ou reprendre tous les privilèges (tous les droits) accordés aux utilisateurs.

Syntaxe : ***Revoke*** *commande*

On *nom_table*

From *nom_user ;*

Ex : **Revoke** Select

On Etudiant

From Malanda ;

4. Complémentarité sur le SQL

Il existe aussi le mot-clé **DISTINCT** qui permet d'éliminer les doublons dans le résultat. Par exemple de façon plus simple, on a :

Select Distinct attribut
From nom_table ;

- **Expressions simples :**

  Comparaison (**=**, **!= ou < >**, **<**, **<=**, **>**, **>=**)

  Combinaison d'expression via :

  - le 'et', **^** : **AND**
  - le 'ou', **v** : **OR**

- **Autres conditions :**
- L'opérateur **IN** permet de spécifier un ensemble de valeur possible

Ex : Select nom, prenom
From Etudiant
Where niveau **IN** ('licence', 'master') ;

- L'opérateur ***BETWEEN … AND…*** permet de spécifier un intervalle de valeurs

Ex : Select matri_Et, nom
From Etudiant
Where date_nais **BETWEEN** '1995-01-01'
AND '2015-01-01' ;

Remarque : Attention à ne pas confondre le **AND** du **BETWEEN** avec celui qui correspond au **^**.

4. Complémentarité sur le SQL

- **Valeurs non définies :**

Il est possible d'avoir des valeurs non définies, elles sont représentées par le mot clé NULL. On peut tester si une valeur n'est pas définie grâce à la condition **IS NULL** (ou au contraire **IS NOT NULL**).

Ex : Select *

From Etudiant

Where statut **IS NULL** ;

- **Tri du résultat d'une requête :**

Le résultat d'une requête est trié par la clause ORDER BY. Dans la clause Order By, il est possible de faire suivre le nom d'un attribut par **ASC** ou **DESC** pour indiquer un ordre **Croissan**t ou **Décroissant.**

Ex : Select nom

From Etudiant

Where code_clas = 'GI1'

Order By matri_Et **DESC**, nom ;

- **Sous-requête renvoyant plusieurs lignes :**

Les opérateurs permettant d'utiliser de telles sous-requêtes : **IN**, **ANY** , **ALL** , **EXISTS**.

- **Opérateurs ensemblistes : UNION**, **INTERSECT**, **MINUS.**

Quelques fonctions numériques :

- **ABS(e)** : Valeur absolue de e (l'élément) ;
- **COS(e)** : Cosinus de e avec e en radians ;
- **SQRT(e)** : Racine carrée de e ;
- **MOD(m, n)** : Reste de la division entière de **m** par **n**, (vaut 0 si n=0) ;
- **ROUND(e, n)** :valeur arrondie de e à n chiffres après la virgule , n optionnel et vaut 0 par défaut ;
- **TRUNC(e, n)** : Valeur tronquée de e à n chiffres après la virgule, n optionnel et vaut 0 par défaut.

NB : Pour **ROUND** et **TRUNC** , si **n** est **négatif** cela indique des chiffres avant la virgule.

Fonctions sur les chaîne de caractères

- ***CONCAT(e1, e2)*** : Concaténation de e1 et e2, parfois représenté par l'opérateur binaire || ;
- **REPLACE(e, old, new)** : Renvoie e dans laquelle les occurrences de *old* ont été remplacées par *new* ;
- **UPPER(e)** : Convertit e en majuscules.
- ***LENGTH(e)*** : Longueur de e ;
- **INSTR(e, s)** : Donne la position de la première occurrence *s* dans e.
- **SUBSTR(e, n, l)** ou **SUBSTRING(e, n, l)** : Renvoie la sous-chaîne de e commençant au caractère *n* et de longueur l. Si l n'est précisé, on prend la sous-chaîne du caractère **n** jusqu'à la fin de **e**.

Fonctions de conversion :

- **ASCII(e)** : Renvoie le code ASCII de premier caractère de e.
- **CHR(e)** : Renvoie le caractère dont le code ASCII est e.
- ***TO_NUMBER(e)*** : Convertit la chaîne e en nombre.
- ***TO_CHAR(e, format)*** : Convertit e en chaîne de caractères, e peut-être un nombre ou une date et *format* indique la forme que doit avoir le résultat.
- **TO_DATE(e, format)** : Convertit une chaîne de caractères en date.

  Ex : TO_DATE ('12122003', 'ddmmyyy') **donne la date '2003-12-12'**
- ***CAST(e AS type)*** ou ***CONVERT(e, type)***: Convertit e en type (le type peut-être **BINARY**, **CHAR**, **DATE**, **TIME**, **DATETIME**, **SIGNED**, **UNSIGNED**).

Fonctions sur les dates :

- **ADD_MONTHS(d, n)** : Ajoute *n* mois à *d ;*
- **d1 – d2** : Nombre de jours entre *d1* et *d2 ;*
- **SYSTDATE** : Date courante ;
- **ADDDATE(d, INTERVAL n DAY)** : Ajoute *n* jours à *d (DAY* peut-être remplacé par **SECOND**, **MINUTE**, **HOUR**, **MONTH** ou **YEAR**.
- **SUBDATE(d, INTERVAL n DAY)** : Il est similaire à **ADDDATE** mais effectue une soustraction.
- **DATEDIFF(d1, d2)** : Nombre de jour entre *d1* et *d2 ;*
- **SYSDATE( )** : Date courante.

  Ex : Select matri_Et, **DATEDIFF**(**SYSDATE()**, paiement)

  From Inscription ;

Fonctions d'agrégation :

- **AVG(e) :** Donne la moyenne de l'évaluation de e sur le groupe ;
- **COUNT(e) :** Le nombre d'occurrences de e dans le groupe, (*) peut remplacer e, compte alors le nombre de n-uplets du groupe ;
- **MAX(e) :** La valeur maximale de e pour le groupe ;
- **MIN(e) :** La valeur minimale de e pour le groupe ;
- **SUM(e) :** La somme des valeurs de e pour le groupe ;
- **STDDEV(e) :** L'écart-type de e pour le groupe ;
- **VARIANCE(e) :** La variance de e pour la groupe ;

NB : e peut-être précédé du mot-clé **DISTINCT** : dans ce cas, on élimine les doublons, important pour **COUNT**, **SUM**, **AVG**, **STDDEV** et **VARIANCE**. k

Approuvé par:

Le Chef de département informatique à l'ECES :

SueCaroline **MABOUDI NAOUME**

Merci