BDD et optimisation SQL

Dans le langage SQL 3 niveaux :

**LMD** : Langage de modification de données (supprimer/ajouter etc)

**LDD** : Langage de définition des données (construction la structure BDD)

LCD : Langage de contrôle des données (administration système)

1. Langage de définition:

Différents types de données :

**Numériques** : integer, float, double,long,tinyint etc

**Chaînes de caractères** :

CHAR(taille), VARCHAR(taille) -> notion de taille comprise entre 1 et 255 (taille max chaîne). Différence entre les deux : VARCHAR en terme de gestion de mémoire plus optimisé. On l’utilise quand on sait que la chaîne va varier. L’autre quand on est sûr que la chaîne reste de taille constante.

TEXT est infini en taille de la chaîne. Seules limites : moi ou disque dur. Sur tout ce qui est traitement de chaîne de caractère il n’est absolument pas optimisé pour la recherche de donnée-> temps de traitement 4 à 5 fois plus long.

Sur Oracle plus le type TEXT.

Exemple : voir feuille

D’autres types : gestion des dates/ fichiers etc

Pour la création d’une table : **CREATE TABLE nomTable () ;**

L’exécution d’une requête SQL est binaire : ok ou pas ok

Convention d’écriture :

Nom de table en majuscule

Nom de colonne en minuscule

Mot clef SQL en majuscule

Création de colonne :

**CREATE TABLE nomTable (**

**NomColonne type de données PRIMARY KEY,**

**NomColonne type de données,**

**NomColonne type de données**

**) ;**

Ou

**CREATE TABLE nomTable (**

**NomColonne type de données,**

**NomColonne type de données,**

**NomColonne type de données,**

**PRIMARY KEY(nom,prenom)**

**) ;**

Pour supprimer une table : **DROP TABLE nomTable ;**

Supprimer le contenu et la structure. Faire attention avec les tables référencées autre part.

Pour le drop c’est une table à la fois.

Pour ajouter une colonne :

**ALTER TABLE nomTable ADD nomColonne typeColonne ;**

Pour supprimer colonne :

**ALTER TABLE nomTable DROP nomColonne;**

Pour la modification du type de donnée d’une colonne :

**ALTER TABLE nomTable MODIFY nomColonne newTypeDonnées;**

Pour renommer une colonne :

**ALTER TABLE nomTable CHANGE ancienNomColonne newName typeColonne;**

1. Langage de Manipulation :

Insérer des données :

**INSERT INTO table(colonne) VALUES(valeurs) ;**

Exemple :

INSERT INTO ETUDIANT (nom,prenom,age) VALUES (‘RENOULT’,’NATALIA’,24) ;

==

INSERT INTO ETUDIANT VALUES (‘RENOULT’,’NATALIA’,24) ;

Les valeurs doivent être dans le même ordre entre les colonnes ET valeurs.

Supprimer des données :

**DELETE FROM nomTable WHERE condition;**

Conditions : >, >=,<,<=,=,!=,and,or

Example :

DELETE FROM ETUDIANT ; -> par défaut supprime les données de la table (vide la table)

Pour cibler des lignes :

DELETE FROM ETUDIANT  WHERE age = 17;

Auto-incrémentation :

**TRUNCATE TABLE nomTable;** -> permet de vider la table et remet l’auto incrémentation à 0.

Modification des données :

**UPDATE nomTable SET modification WHERE condition;**

Example :

UPDATE ETUDIANT SET age = 25 WHERE nom=”RENOULT”;-> modifie que les valeurs.

Extraire de la data :

**SELECT colonne FROM table WHERE condition ;**

Example :

SELECT nom,prenom,age FROM ETUDIANT;

SELECT \* FROM ETUDIANT; -> faire attention au fuite de donnée. \* récupère tout donc le volume de donnée est plus lourd. Le temps de traitement est plus lourd car \* lui rajoute des traitements.

Si on doit afficher toute la table vaut mieux utiliser des conditions même si \* récupère tout.

PARTIE TP :

Généralités et ttt :

1. SELECT COUNT(\*) FROM personne ;
2. SELECT COUNT(DISTINCT(ville)) FROM `personne`;
3. SELECT ville, COUNT(ville) FROM `personne` GROUP BY ville HAVING COUNT(ville)>1
4. SELECT nom, COUNT(nom) FROM `personne` GROUP BY nom order by COUNT (\*) desc limit 1

Manipulation chaînes de caractères:

1. SELECT nom, prenom FROM personne WHERE length(nom)>=5
2. SELECT nom, prenom, length(nom)+length(prenom) FROM personne WHERE MOD(length(nom)+length(prenom),7) GROUP BY nom
3. SELECT prenom FROM `personne` WHERE prenom LIKE '%ll%'
4. SELECT nom, prenom, length(nom) FROM `personne` ORDER BY length(nom) desc LIMIT 1
5. SELECT COUNT(ville) FROM `personne` WHERE ville LIKE '%-%' OR ' % %'
6. SELECT SUBSTR(mail,(LOCATE('@', mail)+1)) FROM personne
7. SELECT nom,prenom,adresse FROM `personne` WHERE adresse LIKE '%Rd%'
8. UPDATE `personne` SET adresse = REPLACE(adresse,'Rd','Road')
9. Pour H : UPDATE personne SET sexe='H' WHERE sexe=' '

Pour F : UPDATE personne SET sexe='F' WHERE prenom LIKE ‘%a’ (faire pour e, i, o, u,y)

1. UPDATE personne SET initial = CONCAT(SUBSTR(nom,1,1), '.', SUBSTR(prenom,1,1))
2. UPDATE personne SET mail\_esgi CONCAT(LOWER(nom),'.',LOWER(prenom),'@esgi.fr')

PARTIE GLOSSAIRE :

**GROUP BY** : S’utilise après la commande WHERE et avant la commande HAVING. Permet d’éviter d’afficher plusieurs fois la même ligne lorsqu’il y a des doublons.

**COUNT()** : Permet de compter le nombre de lignes concernées.

**HAVING()** : Est semblable à WHERE mais permet de filtrer en utilisant des fonctions telles que : SUM()/MIN()/MAX().

**ORDER BY** : Trier par ordre DESC ou ASC (ordre par défaut)

**MOD()** : Modulo

**LIKE** : Permet de faire une recherche sur un modèle en particulier.

* **LIKE ‘%a’** : le caractère “%” est un caractère joker qui remplace tous les autres caractères. Ainsi, ce modèle permet de rechercher toutes les chaines de caractère qui se termine par un “a”.
* **LIKE ‘a%’** : ce modèle permet de rechercher toutes les lignes de “colonne” qui commence par un “a”.
* **LIKE ‘%a%’** : ce modèle est utilisé pour rechercher tous les enregistrement qui utilisent le caractère “a”.
* **LIKE ‘pa%on’** : ce modèle permet de rechercher les chaines qui commence par “pa” et qui se terminent par “on”, comme “pantalon” ou “pardon”.
* **LIKE ‘a\_c’** : peu utilisé, le caractère “\_” (underscore) peut être remplacé par n’importe quel caractère, mais un seul caractère uniquement (alors que le symbole pourcentage “%” peut être remplacé par un nombre incalculable de caractères . Ainsi, ce modèle permet de retourner les lignes “aac”, “abc” ou même “azc”.

**LOCATE()** : Permet de rechercher la position d’une chaîne dans un enregistrement.

* LOCATE(chaine\_recherché, chaine\_source)

**SUBSTR()** : Segmente une chaîne de caractère

* SUBSTR(chaîne\_a\_modifier, caractere\_a\_partir\_duquel\_la\_chaine\_est\_coupee)

**REPLACE()** : Permet de remplacer des caractères dans un chaîne de caractère.

* REPLACE(nom\_colonne, 'ancien texte', 'texte de remplacement')

**CONCAT()** : Permet de concaténer les valeurs de plusieurs colonnes pour ne former qu’une chaîne de caractère.

* CONCAT(colonne1,affichage\_conca,colonne2)

On peut afficher la chaine comme l’on souhaite -> peut avoir plusieurs affichages