Bases De Données

Table of contents

Chapitre 2 :Langages de Manipulation Relationnels: SQL

1. Objectif

- A l'issu de ce cours l'apprenant sera capable de *construire* une base de données en formulant les requêtes adéquates .

2. Introduction

Les instructions SQL sont regroupées en catégories .

Langage de définition de données (LDD/DDL)

Création de relations : `CREATE TABLE`

Modification de relations: `ALTER TABLE`

Suppression de relations: `DROP TABLE`

Vues, index .... : `CREATE VIEW` ...

Langage de requêtes (LMD/DML)

`SELECT ..... FROM ..... WHERE .....`

Langage de manipulation de données (LMD /DML)

Insertion de tuples: `INSERT`

Mise à jour des tuples: `UPDATE`

Suppression de tuples: `DELETE`

Langages de protections d'accès : (ou Data Control Language) qui s'occupe de gérer les droits

d'accès aux table:. GRANT

LDD	LMD
Création de relations : CREATE TABLE Modification de relations: ALTER TABLE Suppression de relations: DROP TABLE Vues, index .... : CREATE VIEW ...	Insertion de tuples: INSERT Mise à jour des tuples: UPDATE Suppression de tuples: DELETE

tableau comparatif

3. Langages de définition de données LDD

3.1. Création de table CREATE TABLE

```
1 CREATE TABLE nom-table
2 { ( nom-col type-col [DEFAULT valeur]
3 [ [CONSTRAINT] contrainte-col] )*
4 [ [CONSTRAINT] contrainte-table ]*
5 | AS requête-SQL };
6
```

Légende :

{a | b} : a ou b

[option]

* : applicable autant de fois que souhaité

mot en capitale : mot clé

3.2. LDD (création)

INTEGER : Ce type permet de stocker des entiers signés codés sur 4 octets.

BIGINT : Ce type permet de stocker des entiers signés codés sur 8 octets.

REAL : réels comportant 6 chiffres significatifs codés sur 4 octets.

DOUBLE PRECISION : Ce type permet de stocker des réels comportant 15 chiffres significatifs codés sur 8 octets.

NUMERIC[(précision, [longueur])]

CHAR(longueur) : chaînes de caractères de longueur fixe

VARCHAR(longueur)

DATE : date.

TIMESTAMP : date et d'une heure.

BOOLEAN : valeurs Booléenne.

MONEY : valeurs monétaires.

TEXT : chaînes de caractères de longueur variable.

contrainte-col: contrainte sur une colonne

NOT NULL

PRIMARY KEY

UNIQUE

REFERENCES nom-table [(nom-col)] [ON DELETE CASCADE]

CHECK (condition)

contrainte-table : contraintes sur une table

PRIMARY KEY (nom-col)*

UNIQUE (nom-col)*

FOREIGN KEY (nom-col) REFERENCES nom-table [(nom-col*)] [ON DELETE CASCADE | SET NULL]*

CHECK (condition)

Example:Création de tables

```
CREATE TABLE Doctorant
( nom VARCHAR(20),
prénom VARCHAR(15),
année_insc DECIMAL(4) DEFAULT 2003 ) ;
CREATE TABLE Doctorant
AS SELECT nom, prénom, année_inscr
FROM Etudiant WHERE statut='Doctorant' ;
```

Example:Primary key

```
CREATE TABLE Pays
( nom VARCHAR(20) PRIMARY KEY ,
```

capitale VARCHAR(20) ...)

CREATE TABLE Employe

(nom VARCHAR(30) ,

prenom VARCHAR(30) ,

age NUMBER CHECK (âge BETWEEN 16 AND 70)

adresse VARCHAR(60) , ...

CONSTRAINT Pk_emp PRIMARY KEY (nom,

prénom))

Example:FOREIGN KEY

CREATE TABLE Etudiant (N°E ...)

CREATE TABLE Cours (NomCours ...)

CREATE TABLE Suit

(N°Etud CHAR(9) ,

NomC VARCHAR(25) ,

PRIMARY KEY (N°Etud , NomC) ,

FOREIGN KEY (N°Etud) REFERENCES Etudiant ,

FOREIGN KEY (NomC) REFERENCES Cours)

3.3. Suppression d'une table

```
1 DROP TABLE nom_table [CASCADE CONSTRAINTS]
2
3 CASCADE CONSTRAINTS
4 Supprime toutes les contraintes de clé externe référençant cette table
5 Si on cherche à détruire une table dont certains attributs sont référencés sans
  spécifier CASCADE CONSTRAINT refus
6
```

3.4. Modification d'une table

```
1 ALTER TABLE nom-table
2 { RENAME TO nouveau-nom-table |
3 ADD ( [ (nom-col type-col [DEFAULT valeur]
4 [contrainte-col])* ] |
5 MODIFY (nom-col [type-col] [DEFAULT valeur]
6 [contrainte-col])* |
7 DROP COLUMN nom-col [CASCADE CONSTRAINTS] |
8 RENAME COLUMN old-name TO new-name
9 }
10
```

Ajout ou modification de colonnes

ALTER TABLE nom_table {ADD/MODIFY} ([nom_colonne type [contrainte], ...])

Ajout d'une contrainte de table

ALTER TABLE nom_table ADD [CONSTRAINT nom_contrainte] contrainte

NB: Si des données sont déjà présentes dans la table au moment où la contrainte d'intégrité est ajoutée, toutes les lignes doivent vérifier la contrainte. Dans le cas contraire, la contrainte n'est pas posée sur la table.

Renommer une colonne

ALTER TABLE nom_table RENAME COLUMN ancien_nom TO nouveau_nom

Renommer une table

ALTER TABLE nom_table RENAME TO nouveau_nom

$$a^2+b^2=c^2$$

4. Langage de manipulation de données (LMD)

4.1. Insertion de n-uplets : INSERT INTO

Cas 1: insérer des nouvelles données en spécifiant les valeurs

INSERT INTO nom_table(nom_col_1, nom_col_2, ...) VALUES (val_1, val_2, ...)

Cas 2: Données provenant d'une autre table. La syntaxe est la suivante :

INSERT INTO nom_table (nom_col1, nom_col2, ...) SELECT ...

Note

Le *SELECT* peut contenir n'importe quelle clause sauf un *ORDER BY* et dont le résultat a le même schéma que nom-table

Exp: INSERT INTO daira(lieu, region) SELECT lieu, region FROM wilaya

4.2. Modification de n-uplets

UPDATE nom_table

SET nom_col_1 = {expression_1 | (SELECT ...) },

nom_col_2 = {expression_2 | (SELECT ...) },

...

nom_col_n = {expression_n | (SELECT ...) }

WHERE predicat

4.3. Suppression de n-uplets

DELETE FROM nom_table WHERE predicat

Note

L'absence de clause WHERE, toutes les lignes de la table sont supprimées

5. Langage de requêtes: Interroger une base

Syntaxe générale de la commande SELECT 8

5.1. Syntaxe générale de la commande SELECT

SELECT [ALL | DISTINCT] { | liste_express}*

[FROM liste_table]

[WHERE condition]

[GROUP BY liste_expression]

[HAVING condition]

[ORDER BY liste expression]

Fundamental

DISTINCT permet de ne retenir qu'une occurrence de n-uplet dans le cas où une requête produit plusieurs n-uplets identiques.

Produit cartésien

SELECT * FROM relation_1, relation_2

équi-jointure

SELECT * FROM relation_1, relation_2 WHERE relation_1.A_1 = relation_2.A_2

$R \cup S$

SELECT *

FROM R

UNION

SELECT *

FROM S

$R \cap S$

```
SELECT *
FROM R
INTERSECT
SELECT *
FROM S
```