

TP Bases de données L2-ACAD Section B

TP3 -Création des tables et des indexes

Le Langage SQL " Structured Query Language" est à la fois un langage de :

- **LDD (Langage de Définition de Données)**: Ensemble d'instructions permettant de créer des tables, des indexes, des vues et d'associer à une définition des contraintes.
- **LMD (Langage de Manipulation de Données)** : Ajout, Suppression, Modification et Interrogation des données
- **LCD (Langage de Contrôle de Données)** : Gestion des protections d'accès.

Partie Rappel LDD (Langage de Définition de Données)

LDD est le Langage de Définition de Données (**LDD**, ou en anglais DDL, *Data Definition Language*) pour créer et supprimer des objets dans la base de données (tables, contraintes d'intégrité, vues, etc.).

Les commandes du LDD sont: **CREATE**, **DROP** et **ALTER**

1. La commande CREATE INDEX

SYNTAXE:

```
CREATE INDEX < Nom_index>  
ON <nom_table> (nom_attribut [ASC/DESC] , ... ) ;  
Ou bien sous SQLPLUS avec le mot clé UNIQUE :
```

SYNTAXE:

```
CREATE [UNIQUE] INDEX < Nom_index>  
ON <nom_table> (nom_attribut [ASC/DESC] , ... ) ;
```

Remarque: :

- Le symbole « | » indique le choix.
Exemple : Soit la table Employés (NumEmployé, Nom, Salaire, Fonction, Date_Recrutement, Code_Service, Code_Departement)
- Créer un index pour la table **Employé** sur l'attribut **NumEmployé** dans l'ordre croissant.

```
CREATE INDEX      index_001      ON      Employés (NumEmployé ASC) ;
```

Commande SQL qui permet la création de l'index

le nom de l'index

Commande SQL qui permet de désigner la table avec l'attribut que vous voulez indexer

L'index nommé index_001 sera créé sur l'attribut NumEmployé de la table Employés

2. La commande CREATE TABLE

1. CREATE TABLE : qui n'est plus qu'une réécriture d'un schéma relationnel en ajoutant une déclaration locale du domaine de chaque colonne et en ajoutant les contraintes,

➤ Première forme :

SYNTAXE:

CREATE TABLE < Nom_table> (définition_des_colonnes) ;

Où « définition_des_colonnes » est présenté comme suit:

liste de : <nom_col> <type des données> [taille] [NULL/ NOT NULL]

Type des données :

- **NUMBER(n)** : Entier à n chiffres
- **NUMBER(n, m)** : Réel à n chiffres au total (virgule comprise), m après la virgule
- **INTEGER**
- **VARCHAR2(n)** : Chaîne de n caractères (entre ' ')
- **DATE** : Date au format „JJ-MM-AAAA“

Exemple : Soit la table Employés (**NumEmployé**, Nom, Salaire, Fonction, Date_Recrutement, Code_Service, Code_Departement)

Commande SQL qui permet
la création de la table

Le nom de la
table

```
SQL PLUS > CREATE TABLE Employés
2 NumEmployé NUMBER (10),
3 Nom VARCHAR(10),
4 Salaire NUMBER(10,2),
5 Fonction VARCHAR(10),
6 Date_Recrutement date,
7 Code_Service NUMBER (10),
8 Code_Departement NUMBER (10),
9 );
```

L'ensemble des attributs
avec leurs types

Remarque:

- ✓ Un attribut déclaré "**NOT NULL**" doit nécessairement être renseigné (rempli) lors de l'insertion d'un tuple.
- ✓ Ne pas mettre de virgule avant la dernière parenthèse fermante de l'instruction **CREATE**.

→ **Deuxième forme :** Elle consiste à créer une table à l'aide d'une autre table (ou de plusieurs tables)

SYNTAXE:

```
CREATE TABLE <Nom_table>  
[(Liste de noms de colonnes avec éventuellement Null ou Not Null)]  
AS < REQUÊTE > ;
```

Remarque: Une table peut donc être créée et alimentée à partir d'une requête.

Exemple : Soit la table Employés_enseignant à partir de la table Employés

```
SQL PLUS > CREATE TABLE Employés_enseignant  
2 AS  
3 ( SELECT NumEmployé, Nom ,salaire ,fonction  
4 FROM Employés  
5 WHERE fonction ='enseignant') ;
```

3. La commande ALTER TABLE

Alter Table : corriger une déclaration mathématique se fait en utilisant une gomme; pour une table déjà créée, la correction doit faire appel à cette commande,

→ **Ajouter une colonne à une table:** consiste en l'ajout de colonnes à la table.

SYNTAXE:

```
ALTER TABLE <Nom_table> ADD  
(définition_d'une_colonne);
```

Exemple : Ajouter le numéro de téléphone des employés

```
SQL PLUS > ALTER TABLE Employés  
2 ADD ( Tel varchar(8) );
```

→ **Supprimer une colonne d'une table**

SYNTAXE:

```
ALTER TABLE <Nom_table> DROP COLUMN (nom_colonne);
```

Exemple : Ajouter le numéro de téléphone des employés

```
SQL PLUS > ALTER TABLE Employés  
2 DROP COLUMN Tel ;
```

→ **Modification du type des colonnes existantes**

SYNTAXE:

```
ALTER TABLE < Nom_table> Modify (redéfinition des  
colonnes);
```

Dans la définition des colonnes, on peut redéfinir le type d'une colonne existante.

Exemple : le numéro de téléphone d'un employé est défini par 10 chiffres

```
SQL PLUS > ALTER TABLE Employés  
2 Modify (tel char(10));
```

4. La commande DROP TABLE

Drop Table : La suppression d'une table déjà créée se fait à travers cette commande.

SYNTAXE:

```
DROP TABLE < Nom_table> ;
```

Exemple : Supprimer la table Employés_Enseignant

```
SQL PLUS > DROP TABLE Employé_Enseignants
```

Remarque: La suppression d'une table supprime aussi tous ces synonymes (c'est un lien)

Rappel : synonyme

Syntaxes :

- CREATE SYNONYM <Nom_synonyme> FOR <Nom_table> ;
- DROP SYNONYM <Nom> ; -- supprime seulement le synonyme

5. Les contraintes d'intégrités

➔ La contrainte de type Not NULL

➤ Définition de la contrainte Not Null lors de la création de la table

Une colonne dans une table peut être spécifiée « not null ». Il n'est, alors pas possible d'insérer la valeur « null » dans cette colonne. Par défaut, une colonne est nullable. Ainsi, dans la table créée ci-dessous, la valeur null peut-être insérer dans les colonnes Prénom et Adresse.

Exemple: Soit la commande de création de la table « Employés » :

```
SQL PLUS > CREATE TABLE Employés (  
2 Nom varchar(50) not null,  
3 Prenom varchar(30),  
4 Adresse varchar(250)  
5 );
```

➤ Ajouter la contrainte “not null” à une colonne

SYNTAXE:

```
ALTER TABLE <Nom_table> MODIFY < nom_attribut > Not Null ;
```

Exemple: Ajouter la contrainte NOT NULL à l'attribut Prénom de la table Employés

```
SQL PLUS > ALTER TABLE Employés modify Prénom Not Null ;
```

➤ Supprimer la contrainte “not null” à une colonne

SYNTAXE:

```
ALTER TABLE <Nom_table> MODIFY < nom_attribut > Null ;
```

Exemple: Supprimer la contrainte NOT NULL à l'attribut Prénom de la table Employés

```
SQL PLUS > ALTER TABLE Employés modify Prénom Null ;
```

→ **Contrainte de type « Unique Key »**

La contrainte de type “unique” ne permet pas de doublons sur la colonne concernée.

Exemple:

```
SQL PLUS > CREATE TABLE Employé
2 NumEmployé NUMBER (10) UNIQUE,
3 Nom VARCHAR(10),
4 Salaire NUMBER(10,2) ,
5 Fonction VARCHAR(10),
6 Date_Recrutement date,
7 Code_Service NUMBER (10),
8 Code_Departement NUMBER (10 ));
```

Dans ce cas, on peut pas avoir deux employés différents avec le même numéro d'employé (NumEmployé)

→ **La contrainte de type « Primary Key »**

Une contrainte de type “Primary Key” combine les contraintes UNIQUE et Not NULL.

Reamarque: Une table ne peut avoir qu'une seule contrainte “Primary key”.

• **Définition de la clé primaire lors de la création de la table**

➤ **Forme 1:**

```
SQL PLUS > CREATE TABLE Employé
2 NumEmployé NUMBER (10) PRIMARY KEY,
3 Nom VARCHAR(10),
4 Salaire NUMBER(10,2) ,
5 Fonction VARCHAR(10),
6 Date_Recrutement date,
7 Code_Service NUMBER (10),
8 Code_Departement NUMBER (10 ));
```

➤ **Forme 2:**

```
SQL PLUS > CREATE TABLE Employé
2 NumEmployé NUMBER (10),
3 Nom VARCHAR(10),
4 Salaire NUMBER(10,2) ,
5 Fonction VARCHAR(10),
6 Date_Recrutement date,
7 Code_Service NUMBER (10),
8 Code_Departement NUMBER (10 )
9 CONSTRAINT PK_Num PRIMAY KY (NumEmployé);
```

Explication de la ligne 9 de la commande:

CONSTRAINT: C'est mot clé de SQL qui permet de créer la contrainte nommée **PK_Num** (ce nom est donné par utilisateur),

PRIMARY KEY: mot clé SQL pour définir le type de la contrainte à créer il doit être suivi par le nom de l'attribut (ou les attribut) sur lequel vous définissez la clé primaire.

➤ **Forme 3:**

```
SQL PLUS > CREATE TABLE Employé
2 NumEmployé NUMBER (10),
3 Nom VARCHAR(10),
4 Salaire NUMBER(10,2) ,
5 Fonction VARCHAR(10),
6 Date_Recrutement date,
7 Code_Service NUMBER (10),
8 Code_Departement NUMBER (10 )
9 PRIMAY KY (NumEmployé);
```

Ici l'est le SQGBD qui va attribuer un nom a votre contrainte.

- **Définition de la clé primaire après avoir créer la table**

SYNTAXE:

```
ALTER TABLE <Nom_Table>
ADD CONSTRAINT <Nom_Contrainte>
PRIMAY KEY ( <Nom_attribut> );
```

Exemple:

```
SQL PLUS > CREATE TABLE Employé
2 NumEmployé NUMBER (10) ,
3 Nom VARCHAR(10),
4 Salaire NUMBER(10,2) ,
5 Fonction VARCHAR(10),
6 Date_Recrutement date,
7 Code_Service NUMBER (10),
8 Code_Departement NUMBER (10 );

SQL PLUS > ALTER TABLE Employé
2 ADD CONSTRAINT PK_num PRIMARY KEY (NumEmployé);
```

Ici, nous avons créé la table Employé sans définir la clé primaire puis nous l'avons ajouté par la commande ALTER TABLE ADD CONSTRAINT

➔ **La contrainte de type « FOREIGN KEY »**

- **Définition de la clé primaire lors de la création de la table**

On suppose ici que la table SERVICE existe déjà (elle a été déjà créée) et que Code_Service est sa clé primaire.

➤ **Forme 1:** ici c'est le SGBD qui va donner le nom à la contrainte.

```
SQL PLUS > CREATE TABLE Employé
2 NumEmployé NUMBER (10) PRIMARY KEY,
3 Nom VARCHAR(10),
4 Salaire NUMBER(10,2) ,
5 Fonction VARCHAR(10),
6 Date_Recrutement date,
7 Code_Service NUMBER (10) FOREIGN KEY REFERENCES SERVICE (Code_Service),
8 Code_Departement NUMBER (10 ));
```

➤ **Forme 2:**

```
SQL PLUS > CREATE TABLE Employé
2 NumEmployé NUMBER (10) PRIMARY KEY,
3 Nom VARCHAR(10),
4 Salaire NUMBER(10,2) ,
5 Fonction VARCHAR(10),
6 Date_Recrutement date,
7 Code_Service NUMBER (10),
8 Code_Departement NUMBER (10 )
9 FOREIGN KEY (Code_Service) REFERENCES SERVICE (Code_Service));
```

➤ **Forme 3:**

```
SQL PLUS > CREATE TABLE Employé
2 NumEmployé NUMBER (10) PRIMARY KEY,
3 Nom VARCHAR(10),
4 Salaire NUMBER(10,2) ,
5 Fonction VARCHAR(10),
6 Date_Recrutement date,
7 Code_Service NUMBER (10),
8 Code_Departement NUMBER (10 )
9 CONSTRAINT fk_Num_serv FOREIGN KEY (Code_Service)
10 REFERENCES SERVICE (Code_Service));
```

- **Définition de la clé primaire après avoir créer la table**

SYNTAXE:

```
ALTER TABLE <Nom_Table_T1>  
ADD CONSTRAINT <Nom_Contrainte>  
FOREIGN KEY (<Nom_attribut_table_T1> ) REFERENCES  
<Nom_Table_T2> ( <Nom_attribut_T2> ) ;
```

```
SQL PLUS > CREATE TABLE Employé  
2 NumEmployé NUMBER (10) ,  
3 Nom VARCHAR(10),  
4 Salaire NUMBER(10,2) ,  
5 Fonction VARCHAR(10),  
6 Date_Recrutement date,  
7 Code_Service NUMBER (10),  
8 Code_Departement NUMBER (10 ));  
  
SQL PLUS > ALTER TABLE Employé  
2 ADD CONSTRAINT FK_num_ser  
3 FOREIGN KEY (Code_Service)  
4 REFERENCES SERVICE (Code_Service);
```

→ La contrainte de type CHECK

Permet d'exprimer des contraintes sur les domaines des valeurs des colonnes.

- Définition de la clé primaire lors de la création de la table

Exemple 1: Ajouter une contrainte sur la table Employé pour que le salaire d'un employé soit toujours entre 15000 et 20000 DA

```
SQL PLUS > CREATE TABLE Employé
2 NumEmployé NUMBER (10) PRIMARY KEY,
3 Nom VARCHAR(10),
4 Salaire NUMBER(10,2) CHECK ( Salaire BETWEEN 15000 AND 20000,
5 Fonction VARCHAR(10),
6 Date_Recrutement date,
7 Code_Service NUMBER (10),
8 Code_Departement NUMBER (10 ));
```

Exemple 2: Ajouter une contrainte sur la table Employé pour que la fonction d'un employé peut être soit Comptable, Enseignant ou Informaticien

```
SQL PLUS > CREATE TABLE Employé
2 NumEmployé NUMBER (10) PRIMARY KEY,
3 Nom VARCHAR(10),
4 Salaire NUMBER(10,2)
5 Fonction VARCHAR(10) CHECK ( Fonction IN ( 'Comptable', 'Enseignat', 'Informaticien'),
6 Date_Recrutement date,
7 Code_Service NUMBER (10),
8 Code_Departement NUMBER (10 ));
```

Exemple 3: Ajouter une contrainte sur la table Employé pour que le salaire soit toujours supérieur à 15000 DA

```
SQL PLUS > CREATE TABLE Employé
2 NumEmployé NUMBER (10) PRIMARY KEY,
3 Nom VARCHAR(10),
4 Salaire NUMBER(10,2) CHECK ( Salaire > 15000),
5 Fonction VARCHAR(10),
6 Date_Recrutement date,
7 Code_Service NUMBER (10),
8 Code_Departement NUMBER (10 ));
```

- **Définition de la clé primaire après avoir créer la table**

SYNTAXE:

```
ALTER TABLE <Nom_Table>  
ADD CONSTRAINT <Nom_Contrainte>  
CHECK (condition) ;
```

Exemple 1: Ajouter une contrainte sur la table Employé pour que le salaire d'un employé soit toujours entre 15000 et 20000 DA

```
SQL PLUS > ALTER TABLE Employé  
2 ADD CONSTRAINT CK_Salaire  
3 CHECK ( Salaire BETWEEN 15000 AND 20000);
```

Exemple 2: Ajouter une contrainte sur la table Employé pour que la fonction d'un employé peut être soit Comptable, Enseignant ou Informaticien

```
SQL PLUS > ALTER TABLE Employé  
2 ADD CONSTRAINT CK_Salaire  
3 CHECK ( Fonction IN ( 'Comptable', 'Enseignant', 'Informaticien');
```

Exemple 3: Ajouter une contrainte sur la table Employé pour que le salaire soit toujours supérieur à 15000 DA

```
SQL PLUS > ALTER TABLE Employé  
2 ADD CONSTRAINT CK_Salaire  
3 CHECK ( Salaire > 15000);
```