# Chapitre 4
# Requêtes : mises à jour, création de tables

# Chapitre 4 : plan

# LMD

La **mise à jour** des tables fait partie, comme l'**interrogation** de la 'manipulation de données'.

L'ensemble du langage d'interrogation et de mise à jour est appelé :
**LMD = Langage de Manipulation de Données**

# LMD

La partie **interrogation** du LMD ne comporte qu'une seule instruction :

**SELECT**

Rappel : l'instruction DESCRIBE (non SQL) ne concerne pas les données, mais le schéma relationnel (intention, structure de la table)

# LMD : mise à jour

La partie **mise à jour** du LMD comporte quatre instructions :

**INSERT --** insertion de nouveaux tuples

**DELETE --** suppression de tuples

**TRUNCATE --** suppression de tuples

**UPDATE --** modification de tuples déjà insérés

# LMD : mise à jour

Pour l'insertion d'un t-uple l'instruction de base est :

**INSERT INTO** nomTable
**VALUES (**v1,v2,...**)**
**;**

- - où le nombre de valeurs et leur type doivent correspondre exactement au résultat du **DESCRIBE**
- - les tuples sont insérés un par un

# LMD : mise à jour

L'instruction :

**INSERT INTO** nomTable **(** nomColonne **)**
**VALUES** **(** nouvelleValeur **)**
**;**
-- permet d'insérer un tuple avec une valeur précise dans une colonne, toutes les autres étant mises à **NULL**

mais il est possible aussi d'insérer les **NULL** en utilisant la méthode précédente

# LMD : mise à jour

Suppression de t-uple :

**DELETE FROM** nomTable
**WHERE** condition
**;**

-- supprime les tuples de la « table » qui vérifient la « condition »

# LMD : mise à jour

Vidage de toute la table :

**DELETE FROM** Clients ;

-- supprime, si c'est possible, tous les clients enregistrés, et sinon ne fait rien

**TRUNCATE TABLE** Clients ;

-- même effet, a l'avantage d'être plus rapide et utilise moins de ressources.

# LMD : mise à jour

Ceci étant l'avantage de la commande DELETE est de pouvoir :

Supprimer certains t-uples

**DELETE FROM** Clients
**WHERE** NumClient = 2365 - - clé ;

-- supprime le client de clé primaire 2365, à condition que ce client ne soit pas référencé dans une autre table

# LMD : mise à jour

Modification de t-uples :

**UPDATE** nomTable
**SET** nomColonne **=** nouvelleValeur
**WHERE** condition
**;**

-- modifie les tuples de la « table » qui vérifient la « condition » en remplaçant l'ancienne valeur de « colonne » par « nouvelleValeur »

# LMD : mise à jour

Exemple de modification de t-uples :

```
UPDATE  Employes
SET     Salaire = ( SELECT MAX ( Salaire )
                    FROM   Employes
                  )
;
```

-- tous les employés auront le salaire maximum

# LMD : mise à jour

Modification d'un t-uple :

```
UPDATE  Employes
SET     Salaire  =  1.3  *  Salaire
WHERE   NomEmploye = 'moi'  -- clé
;

-- j'ai augmenté mon salaire de 30%
```

# LDD : création des tables

Le langage de création des tables est le **LDD : Langage de Définition des Données**

Il comporte quatre instructions :

- **CREATE** : création des objets
- **DROP** : suppression des objets
- **ALTER** : modification des objets
- **RENAME** : renommage des objets

# LDD et LMD

Les mots - clés employés caractérisent le niveau LMD ou LDD :

- **SELECT ... <---------> DESCRIBE**

- **INSERT INTO ... <---------> CREATE TABLE ...**
- **DELETE FROM ... <-------> DROP TABLE ...**
- **UPDATE ... <---------> ALTER TABLE ...**

# LDD : création des tables

Syntaxe de base :

```
CREATE TABLE nomTable (
                    nomCol1  <<type de col1>>,
                    nomCol2  <<type de col2>>,
                      ...
                     nomColn  <<type de coln>>
                     )
;
```

# LDD : création des tables

-- Traduction du schéma relationnel typé :

-- Produits (code : [1..9999], libelle : car[30])

```
CREATE TABLE Produits (
                    code  NUMBER(4) ,
                    libelle VARCHAR2(30)
                          )
 ;
 --  attention ! Pas de ligne blanche dans un
                    CREATE
```

# LDD : contraintes de clé

Clé primaire :

Attention, le schéma relationnel précédent :
Produits (code : [1..9999], libelle : car[30])
est **incorrect**.
**Toute relation doit contenir une clé primaire !**
On pourra remplacer ce schéma par :

Produits ([code : [1..9999]](1), libelle : car[30])

d'où la nouvelle traduction --->

# LDD : contraintes de clé

Codage en colonne de la clé primaire :

-- Produits ([code : [1..9999]](1), libelle : car[30])

```
CREATE TABLE Produits (
                code  NUMBER(4)
                CONSTRAINT pkProduits PRIMARY KEY,
                libelle VARCHAR2(30)
                                )
;
```

# LDD : contraintes de clé

Codage en table de la clé primaire :

-- Produits ([code : [0..9999]](1), libelle : car[30])

```
CREATE TABLE Produits (
                    code  NUMBER(4) ,
                    libelle VARCHAR2(30),
                    CONSTRAINT    pkProduits
                    PRIMARY KEY  (code)       - - subtile différence !
                                 )
;
```

# LDD : contraintes de clé

Dans le cas d'une clé primaire multiattribut, seul le codage en table est possible :

-- Etudiants ([nom : car[20], prenom : car[15]](1))

```
CREATE TABLE Etudiants  (
                    nom       VARCHAR2(20),
                    prenom  VARCHAR2(20),
                   CONSTRAINT   pkEtudiants
                   PRIMARY KEY (Nom, Prenom)
                                )
;
```

# LDD : contraintes de clé

Clé primaire :

Dans cette définition, **le SGBD vous garantit :**

1/ qu'aucune valeur de la clé ne sera **NULL**

2/ que toutes les valeurs de la clé seront **différentes**

Il n'est pas prévu de codage pour une **clé secondaire ;**

il faudra donc si nécessaire, **coder séparément ces deux contraintes**

# LDD : contraintes de clé

Codage en colonne de clé étrangère mono attribut :

-- ALivrer (...,unProduit=@Produits[code] : [1..9999]],...)

```
CREATE TABLE ALivrer (...
                      unProduit  NUMBER(4)
                      CONSTRAINT   fkProduit
                      REFERENCES Produits(code)
                      , ...
                      )
 ;
```

# LDD : contraintes de clé

Codage en table de clé étrangère mono attribut :

```
-- Alivrer (...,unProduit=@Produits[code]
          : [1..9999]],...)
CREATE TABLE ALivrer (...
                              unProduit  NUMBER(4)
                              , ... ,
                              CONSTRAINT   fkProduit
                              FOREIGN KEY ( UnProduit )
                              REFERENCES Produits(code)
                              , ...
                              )
;
```

# LDD : contraintes de clé

Codage en table obligatoire pour les clés étrangères multi attributs :

Projets (...,(nomChef ,prenomChef)=
@Personnes[nom : car[20], prenom:car[15]], …)

```
CREATE TABLE Projets (...
                      nomChef  VARCHAR2(20),
                      prenomChef  VARCHAR2(15),
                      CONSTRAINT   fkChef
                      FOREIGN KEY(nomChef,prenomChef)
                      REFERENCES Personnes(nom,prenom)
                      , ...
                     )
;
```

# LDD : contraintes de clé

Le SGBD vous garantit que :

- ou bien une des colonnes **du n-uplet** de la clé étrangère est **NULL**

- ou bien que la **valeur du n-uplet** (ici le couple) **existe dans la clé primaire de la table 'étrangère' (référencée)**

# LDD : contraintes de clé

## Remarque

Il n'est pas possible :

- de référencer autre chose qu'une **clé primaire**
- de donner un **type différent** à la clé étrangère

Par contre, il est possible d'**auto-référencer,** c'est-à-dire de référencer la clé primaire **de la même table**

# LDD : contraintes de clé

Première table :

| PK1 | PK2 |
|---|---|
| 10 | 20 |
| 35 | 75 |

| PK | FK1 | FK2 | |
|---|---|---|---|
| 1 | 10 | 20 | existe |
| 2 | 4783 | NULL | acceptée sans exister |
| 3 | NULL | NULL | ..... |
| 4 | NULL | 75 | ..... |
| 5 | 10 | 75 | refusée |

# LDD : autres contraintes

En plus de **PK** (colonne ou table) et **FK** (table ou base)

**NN** : obligation de présence (**colonne**)

**UQ** : unicité (colonne ou table)

**CK** : contrôle de saisie (colonne ou table)

# LDD : autres contraintes

**Contrainte NN**

**NOT NULL** : obligation de présence
ne peut porter que sur une seule **colonne**

implicite dans **PK**
facultative pour FK

# LDD : autres contraintes

Exemple de contrainte NN :

-- Etudiants (.... ,numTel : car[10](NN), ....)

```
CREATE TABLE Etudiants  (.... ,
                         numTel      VARCHAR2(10)
                         CONSTRAINT  nnNumTel
                         NOT NULL,
                         ,.....
                         )
;
```

# LDD : autres contraintes

## Autre exemple : FK + NN

-- ALivrer (...,unProduit=@Produits[code] : [1..9999]] (NN),...)

```
CREATE TABLE ALivrer (… ,
                    unProduit  NUMBER(4)
                    CONSTRAINT   fkProduit REFERENCES Produits(code)
                    CONSTRAINT   nnfkProduit NOT NULL
                                        , ...
                                       )
;
```

# LDD : autres contraintes

**Contrainte UQ :**

**UNIQUE** : unicité
peut porter sur une ou plusieurs colonnes
implicite dans **PK**
en général fausse pour FK

sert pour coder une « clé secondaire »

# LDD : autres contraintes

Exemple de contrainte UQ :

-- Etudiants (....  ,mel : car[30](UQ), ....)

```
CREATE TABLE Etudiants  (....,
                              mel  VARCHAR2(30)
                              CONSTRAINT   uqMel UNIQUE,
                                  )
;
```

# LDD : autres contraintes

**Contrainte CK :**

**CHECK** ( ) : contrôle de saisie
peut porter sur une ou plusieurs colonnes

précondition pour entrer dans la base de données

# LDD : autres contraintes

Exemple de contrainte CK :

-- Etudiants (....  ,codePostal : car[5](CK), ....)

```
CREATE TABLE Etudiants  (....,

                codePostal VARCHAR2(5),
                CONSTRAINT   ckCodePostal
                CHECK ( codePostal LIKE '56%' ),
                                ....
                                )
;
```

# LDD : autres contraintes

**Niveau de contraintes :**

**PK** : table
**FK** : base
**NN** : colonne
**UQ** : table
**CK** : table

# LDD : suppression de table

**DROP TABLE** nomTable
**;**

n'est possible que s'il n'existe aucune table contenant une FK vers la PK de la table à supprimer.

# LDD : renommer une table

**RENAME** nomTable **TO** nouveauNom
**;**

évidemment le nom doit être **nouveau** !

# LDD : ajout de contrainte hors NN

**ALTER TABLE** nomTable
    **ADD CONSTRAINT** nomContrainte
    <contrainte>
;

valable pour PK, FK , UQ , CK
tout sauf NN

# LDD : ajout de contrainte hors NN

**Exemple :**

```
ALTER TABLE  Etudiants
    ADD CONSTRAINT  fkResidence
                FOREIGN KEY ( Residence )
                REFERENCES Ville(nom)
;
```

# LDD : ajout de contrainte NN

```
ALTER TABLE  nomTable
    MODIFY   nomColonne   <<type>>
    CONSTRAINT   nnNomColonne
    NOT NULL
;
```

# LDD : ajout de contrainte NN

**Exemple :**

```
ALTER TABLE  Etudiants
    MODIFY    codePostal      VARCHAR2(5)
                  CONSTRAINT   nnCodePostal
                      NOT NULL
;
```

**- - les autres contraintes, s'il en existe, sont conservées**

# LDD : suppression de contrainte

**ALTER TABLE** nomTable
**DROP CONSTRAINT** nomContrainte
**;**

**Exemple :**

**ALTER TABLE** Etudiants
**DROP CONSTRAINT** nnCodePostal
**;**

# LDD : activation / désactivation de contrainte

**ALTER TABLE** nomTable
**DISABLE CONSTRAINT** nomContrainte
**;**

**ALTER TABLE** nomTable
**ENABLE CONSTRAINT** nomContrainte
**;**