

SQL - Cours 5

Le langage SQL

Ikbel GUIDARA

ikbel.guidara@univ-lyon1.fr

Manipulation des Tables

Manipulation des tables

- Pour créer une table sans modèle, il faut préciser au moins :
 - les noms
 - les types
 - pour certains types la taille de la colonne
 - les contraintes

```
CREATE TABLE [IF NOT EXISTS] Table (  
    nomColonne1 type [DEFAULT valeur] [contrainteColonne],  
    nomColonne2 type [DEFAULT valeur] [contrainteColonne],  
    ...  
    nomColonneN type [DEFAULT valeur] [contrainteColonne],  
[contrainte],  
[contrainte], ...  
);
```

Supprimer une table

```
DROP TABLE Table;
```

```
DROP TABLE emp;
```

Renommer une table

```
RENAME ancien_nom_table TO nouveau_nom_table;
```

```
RENAME emp TO employe;
```

Types de données

- **CHAR (N)** : chaîne de caractères de longueur fixe = N
- **VARCHAR2(N)** : chaîne de caractères de longueur variable mais inférieure à N
- **NUMBER (X), Number(X,Y)**
 - X est le nombre de chiffres au total
 - Y est le nombre de chiffres après la virgule
- **DATE**

```
CREATE TABLE PUF(  
    NP Number(10),  
    NU Number(10),  
    NOMP Varcher2(40),  
    Ville Varchar2(35),  
    Quantite Number(10) DEFAULT 10  
);
```

DEFAULT: Pour donner une valeur par défaut pour une colonne si la valeur est omise à l'insertion d'une ligne

Types de contraintes

- **PRIMARY KEY** : clé primaire
- **FOREIGN KEY** et **REFERENCES** : clé étrangère
- **UNIQUE** : les valeurs d'une colonne ne sont pas redondantes
- **CHECK** : domaine de définition d'une colonne (l'ensemble des valeurs possibles)
- **NOT NULL** : la valeur de la colonne est obligatoire
- Les contraintes peuvent être définies à deux niveaux
 - Niveau colonne
 - Niveau externe à la colonne: obligatoire si la contrainte concerne plusieurs colonnes

Syntaxe (réduite) – Création d'une Table avec contraintes

```
CREATE TABLE [IF NOT EXISTS] $table_name (  
    $column_name $type [k times] ,  
    Constraint $pk_cstr primary key (column_1, ...),  
    Constraint $fk_cstr foreign key (column_1, ...) references $table_name,  
    Constraint $un_cstr unique (column_1,...)  
    Constraint $ck_cstr check ($condition_expression)  
    Constraint $nn_cstr not null (column_name)  
);
```

2 méthodes pour déclarer les contraintes

```
CREATE TABLE PUF(  
  NP Number PRIMARY KEY,  
  NU Number REFERENCES P(NU),  
  NOMP Varcher2(40) UNIQUE,  
  Ville Varchar2(35) CHECK (Ville in ('Lyon', 'Marseille', 'Paris', 'Dijon')),  
  Quantite Number NOT NULL DEFAULT 10  
);
```

```
CREATE TABLE PUF(  
  NP Number,  
  NU Number,  
  NOMP Varcher2(40),  
  Ville Varchar2(35),  
  Quantite Number DEFAULT 10  
  
  Constraint pk_np PRIMARY KEY(NP),  
  Constraint fk_nu FOREIGN KEY(NU) REFERENCES P(NU),  
  Constraint un_nom UNIQUE NOMP,  
  Constraint ck_ville CHECK (Ville in ('Lyon', 'Marseille', 'Paris', 'Dijon')),  
  Constraint nn_qte NOT NULL Number  
);
```

Création de tables en plusieurs étapes

Remarque : création de la table en plusieurs étapes

Il est possible d'ajouter les contraintes ou de les modifier après la création de la table en utilisation la commande :

ALTER TABLE \$table_name add pour ajoute des contraintes ou des nouvelles colonnes

ALTER TABLE \$table_name modify pour modifier les définition d'une colonne ou d'une contrainte

ALTER TABLE \$table_name drop pour supprimer une colonne ou une contrainte

Ajout, modification, suppression d'une colonne

AJOUT

```
ALTER TABLE Table ADD  
(colonne1 type ..., Colonne2 type ... );
```

```
ALTER TABLE F ADD (TEL varchar2(30) );
```

MODIFICATION

```
ALTER TABLE Table MODIFY  
(colonne1 type ..., Colonne2 type ... );
```

```
ALTER TABLE F MODIFY (TEL varchar2(30) );
```

SUPPRESSION

```
ALTER TABLE Table DROP  
(colonne1, Colonne2...);
```

```
ALTER TABLE F DROP TEL;
```


Manipulation de données

Manipulation des données

- SQL offre trois possibilités pour modifier une BDD
 - INSERT : ajout de lignes dans la table
 - UPDATE: mise à jour de lignes
 - DELETE: Suppression de lignes

Insertion

- Insertion de tuples : INSERT INTO VALUES

```
INSERT INTO nomTable (att1,..., attn)  
VALUES (val1,...,valn)
```

```
INSERT INTO Etudiant(idEtud, nom, prénom, age)  
VALUES (E001,Perron, Jean, 25)
```



Etudiant

idEtud	nom	prénom	age
E001	Perron	Jean	25

- Il y a autant d'expressions que de colonnes
- La liste des colonnes à renseigner est facultative. Quand elle est omise, toutes les colonnes sont alors renseignées

Remarque : insertion avec un résultat d'interrogation

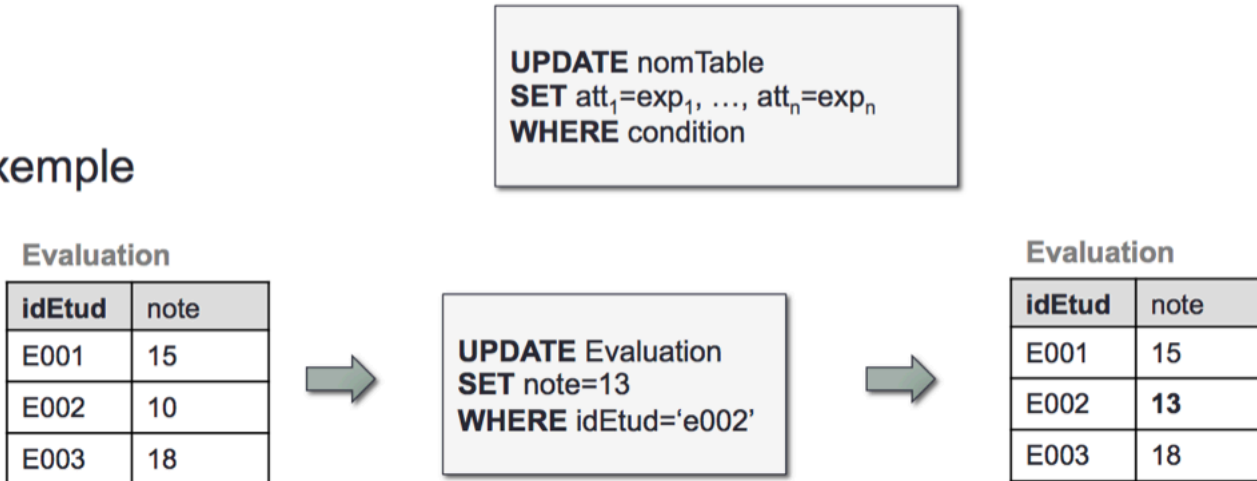
Il est possible d'insérer un ensemble de tuples dans une table à partir du résultat d'une requête select. Comme le montre la requête suivante :

```
> insert into table_name [(coln1,coln2 ... , colnn)]  
select val1,val2,...,valn from ...
```

Modification

- Modification des données : UPDATE

- Exemple



Remarque : modification avec un résultat d'interrogation

De la même manière que pour l'insertion, il est possible d'utiliser le résultat d'une interrogation pour faire un update d'une ou plusieurs lignes, comme le montre la requête ci-dessous : *

```
> Update $table_name
  set col1,col2 ..., coln = (select val1,val2 ... valn from ...)
  [where $condition_expression]
```

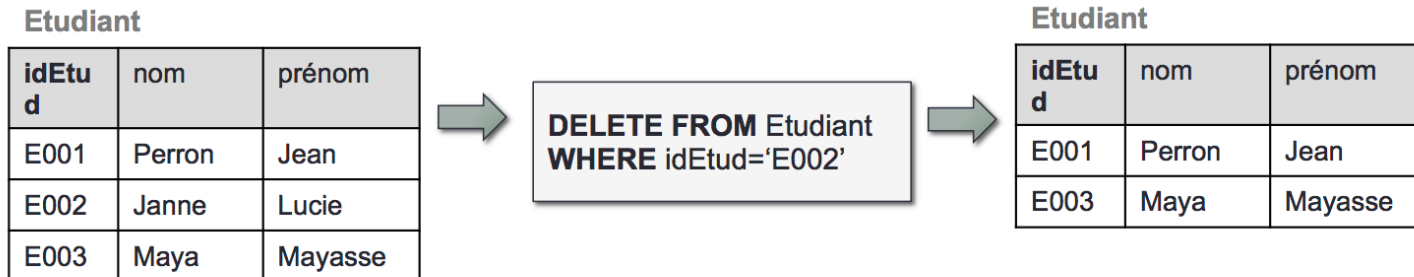
NB : la requête d'interrogation doit retourner une ligne au plus

Suppression

- Suppression des données : DELETE

DELETE FROM nomTable
WHERE condition

- Exemple



- La clause de condition peut être le résultat d'une requête

Les vues

Les vues

- Ce sont des **tables virtuelles** présentant le résultat d'un **SELECT**
- La vue ne stocke pas les données, mais fait référence à une ou plusieurs tables d'origine à travers une requête **SELECT**, qui est ré-exécutée chaque fois qu'on utilise la vue

```
CREATE VIEW nom_vue AS SELECT liste_attributs FROM table  
WHERE condition ;
```

```
DROP VIEW nom_vue ;
```