SQL (suite)

- Manipulation: CRUD
  Insertion, Suppression, mise à jour, lecture
- Triggers
- Vues

Manipulation CRUD

INSERT INTO
(Create) pour insérer des tuple

SELECT FROM
(Read) pour lire des tuples

UPDATE
(Update) pour mettre à jour des tuples

DELETE FROM
(Delete) pour supprimer des tuples

Insertion

- Insérer des tuples dans une table
- Syntaxe:
  - **INSERT INTO table VALUES ('valeur 1', 'valeur 2', ...)**
  - **INSERT INTO table (nomAtt1, nomAtt2, ...)**
    VALUES
    ('valeur 1', 'valeur 2', ...)
    ('valeur 1', 'valeur 2', ...)
  - **INSERT INTO table (nomAtt1, nomAtt2, ...)**
    SELECT col1, col2, ...
    FROM table

Insertion

```
INSERT INTO Employe
VALUES (102, 'DUPONT', 35, 'Ingenieur',
120, '10/01/2002', 3500, 0, 25);
```

```
INSERT INTO Employe(num, nom, fonc)
VALUES (104, 'DURAND', 'Comptable');
```

Insertion à partir d'une sélection

```
INSERT INTO Employe(num, nom, fonc)
    SELECT F.numero, F.nom, F.fonction
    FROM Employe_2   F
    WHERE F.dept=4;
```

Suppression

- Supprimer des tuples dans une table
- Syntaxe:
  - **DELETE FROM `table`**
    WHERE condition
  - **DELETE FROM `table`**
    WHERE att1 IN (
    SELECT att1
    FROM table
    WHERE <condition>
)

Suppression

```
DELETE FROM Employe
WHERE nom = 'DUPONT' ;
```

```
DELETE FROM Employe
WHERE dept in (SELECT  num
                FROM Departement D
                WHERE D.nom='R&D') ;
```

Mise à jour

- Mettre à jour les attributs de tuples
- Syntaxe:
  - **UPDATE table**
    SET colonne_1 = 'valeur 1', colonne_2 = 'valeur 2', colonne_3 = 'valeur 3'
    WHERE <condition>

  - **UPDATE table**
    SET colonne1 = valeur 1, colonne2 = valeur 2,
    WHERE colonn_1 IN
    SELECT att1
    FROM table
    WHERE <condition>
)

Mise à jour

```
UPDATE Employe
SET dept=25
WHERE  dept=45 ;
```

```
UPDATE Employe
SET salaire=salaire*1.1
WHERE  dept in (SELECT  num
                FROM Departement D
                WHERE D.nom='R&D') ;
```

Triggers

Définition de triggers

- Les triggers sont des objets de la base de données
- Ils sont attachés à une table
- Ils permettent de déclencher l'exécution d'instructions lorsqu'une modification est effectuée

Définition de triggers

Un trigger spécifie un évènement, un moment, une action, une condition,
pour surveiller la base de données

L'action doit être exécutée automatiquement si la condition est satisfaite

CREATE TRIGGER *nom_trigger*
quand evenement* ON *table
[FOR EACH ROW [WHEN *condition*]]
action;

Définition de triggers

- ***Nom_trigger***: Nom qui permet de d'identifier le trigger
- ***Quand***: BEFORE/AFTER
- ***Evenement***: INSERT/UPDATE/DELETE
- ***Table***: Table à laquelle le trigger est rattaché
- ***condition***: Condition pour exécution du trigger
- ***Action***: instruction à réaliser

Définition de triggers

FOR EACH ROW

trigger de niveau ligne : pour que la règle soit déclenchée une fois pour chaque ligne affectée par l'évènement déclencheur

sans cette clause : trigger de niveau instruction : règle déclenchée une seule fois pour chaque évènement

Définition de triggers

- Il ne peut exister qu'un seul trigger par combinaison ***moment/evenement*** par table
- Ex. pour une table FILM un seul trigger BEFORE UPDATE
- ***Donc pour chaque table, un nombre max de triggers possible !***

Définition de triggers

- Dans la partie *instructions* du Trigger, MySQL met a disposition deux mots clés: **OLD** et **NEW**
- ***OLD***: correspond aux valeurs de la ligne avant le traitement
- ***NEW***: correspond aux valeurs de la ligne après traitement
- Conséquence: **OLD** et **NEW** n'existent que dans le cas d'un UPDATE

Définition de triggers

- Exemple:
  - INSERT INTO Client(id, nom, prenom, code) Values(122, Toto, Alain, 97180)
  - Dans le trigger

    NEW.id -> 122
    NEW.nom -> Toto
  - Evidemment, OLD n'est pas défini

Définition de triggers

- Exemple:
  - UPDATE Client
    SET code = 97200
    WHERE id=122
  - Dans le Trigger
    OLD.code -> 97180
    NEW.code -> 97200
    OLD.nom = NEW.nom -> TOTO

Exemple

on suppose la table DeptSalaires créée :

DeptSalaires (numDept, nbreEmp, totalSal);

Mettre en place un trigger qui garantisse que l'attribut *totalSal* de la table *DeptSalaires* soit toujours égal au total des salaires de tous les employes du département

Rappel:

Employe(num, nom, age, fonction, sup, datent, salaire, comm, #numDept)

Departement(numDept, nom, noresp, datecre, loc)

Exemple trigger

```
CREATE TRIGGER TotalSalaires
AFTER INSERT ON Employe
FOR EACH ROW
WHEN new.dept IS NOT NULL AND SALAIRE
    IS NOT NULL
UPDATE DeptSalaires
SET total_sal=total_sal + new.salaire
WHERE num_dept=new.dept ;
```

Vues

Définition des vues

- Une vue est une table virtuelle dérivée de tables de base :
  - On stocke seulement la définition de vue
  - Son contenu est généré dynamiquement

CREATE VIEW *nom_de_vue*
(nom_de_colonne {,nom_de_colonne }) ;
AS sous-selection ;

Exemple de vues

- Exemple

```
CREATE VIEW EmployeDept(num, nom,dept)
AS
SELECT    E.num, E.nom, D.nom
FROM      Employe E, Departement D
WHERE     E.dept=D.num ;
```

Propriétés des vues

- Une vue est toujours supposée à jour
- Elle n'est générée que lorsqu'une réquête est spécifiée sur elle
- Une requête sur une vue est traduite en requête sur les tables de base
- Une table temporaire correspondant à la vue est créée