ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR

Membre de HONORIS UNITED UNIVERSITIES

École Marocaine des Sciences de l'Ingénieur

Filière: IIR

Les Bases de Données

Prof. Zakaria KHATAR

LANGAGE SQL :
Langage de Manipulation de données (LMD)

1) Les Commandes de manipulation de données :

Les commandes de manipulation de données sont :

- **INSERT** pour insérer dans une table une ligne donnée ou des lignes résultat d'un SELECT.
- **UPDATE** pour modifier dans une table un ou plusieurs champs dans une ou plusieurs lignes.
- **DELETE** pour supprimer dans une table une ou plusieurs lignes.

1-1) La commande d'Insertion

La commande **INSERT** est utilisée pour insérer des lignes dans une table. Elle a trois formes :

```
INSERT INTO nom_table VALUES ( valeur1, valeur2, ..);
```

Exemple :

INSERT INTO employees **VALUES (1235, 'Ziyech', 'Hakim', 15000, 26)**;

Cette forme permet d'insérer une ligne unique, avec une valeur pour toutes les colonnes.

S'il y a un doublon pour une colonne à valeur clé ou unique, l'insertion est refusée.

Table : **Employees**

Num_Emp	Nom	Prénom	Salaire	Age
1235	Ziyech	Hakim	15000	26

1-1) La commande d'Insertion

```
INSERT INTO nom_table (colonne1, colonne2,..)
VALUES ( valeur1, valeur2, ..);
```

Exemple :

```
INSERT INTO employees (Num_Emp, Nom, Prénom, Salaire, Age)
VALUES (1235,'Ziyech', 'Hakim', 30000,26);
```

Cette forme ajoute une ligne avec certaines valeurs définies, les colonnes non définies dans la commande seront remplies avec **NULL**.

S'il y a un doublon pour une colonne à clé primaire ou unique, l'insertion est refusée.

Pour insérer la date système (date actuelle) utiliser **SYSDATE** comme valeur.

1-1) La commande d'Insertion

INSERT INTO nom_table SELECT ...

Cette forme permet d'insérer dans une table les lignes résultats de SELECT.
Le schéma du SELECT doit correspondre à celui de la table (en terme de colonne).

INSERT INTO client **SELECT** Nom, Prénom **FROM** employees **WHERE** Nom ='Ziyech';

Table: **employees**

Num_Emp	Nom	Prénom	Salaire	Age
1235	Ziyech	Hakim	15000	26

Table: **client**

Nom_Client	Prénom_Client	Adresse
Ziyech	**Hakim**	**Null**

1-1) La commande de Modification

La commande **UPDATE** permet de modifier les composants d'une ligne dans une table.

```
UPDATE nom_table SET colonne = valeur, colonne = valeur, ...
WHERE condition;
```

Exemple :

UPDATE employees **SET** Salaire = 30000, Age = 30 **WHERE** Num_Emp=1235;

Num_Emp	Nom	Prénom	Salaire	Age
1235	Ziyech	Hakim	**30000**	**30**

1-1) La commande de Suppression

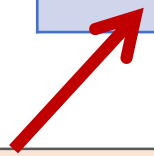
La commande **DELETE** permet de supprimeer des lignes dans une table.

```
DELETE FROM nom_table WHERE condition;
```

Exemple :

DELETE FROM employees **WHERE** Num_Emp=1235;

Num_Emp	Nom	Prénom	Salaire	Age

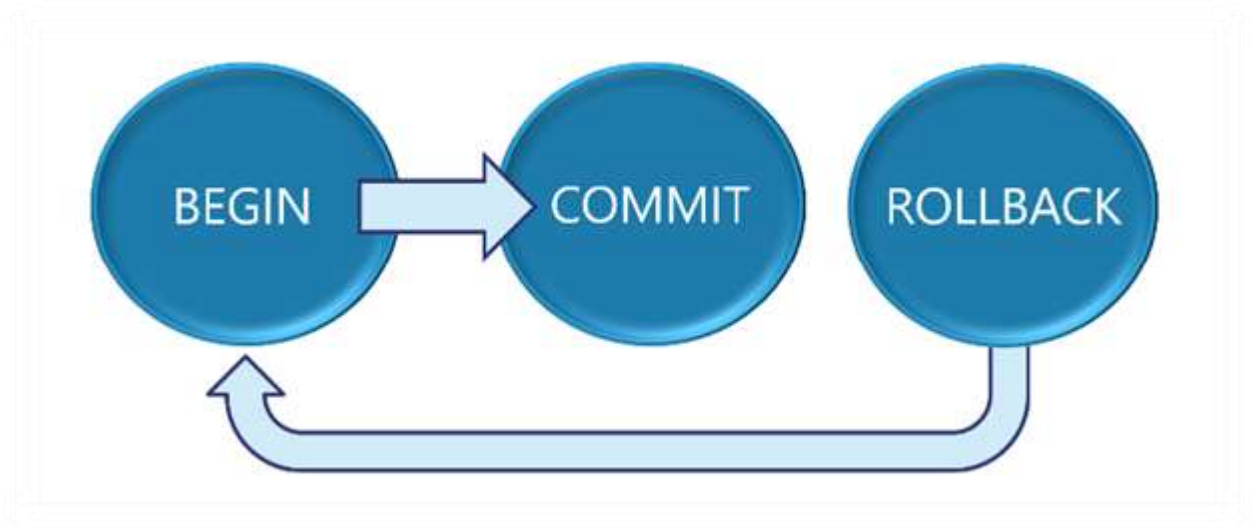


la ligne a été supprimée

Les transactions

1) La gestion des transactions

Les transactions SQL sont utilisées pour exécuter plusieurs instructions SQL en tant que groupe unique, de sorte qu'elles soient **toutes exécutées ou annulées ensemble**, en utilisant les commandes **COMMIT** et **ROLLBACK**, afin de maintenir l'intégrité des données dans la base de données.



1-1) La commande AUTOCOMMIT

La commande **AUTOCOMMIT** est utilisée pour **désactiver la gestion des transactions**. Cela signifie que chaque instruction SQL est traitée comme une transaction individuelle et **est automatiquement validée ou annulée**. Par exemple:

```
-- Désactivez la gestion des transactions
SET AUTOCOMMIT ON;

    -- Mise à jour de la table employees
    UPDATE employees SET salary = salary + 1000 WHERE employee_id = 100;

    -- Mise à jour de la table departments
    UPDATE departments SET budget = budget - 1000 WHERE department_id = 10;
```

Dans cet exemple, chaque instruction SQL est exécutée en tant que **transaction individuelle**, ce qui signifie qu'elles sont **automatiquement validées ou annulées**.

1-1) La commande AUTOCOMMIT

La commande AUTOCOMMIT peut être activée ou désactivée avec la commande suivante :

```
SET AUTOCOMMIT ON;
```

```
SET AUTOCOMMIT OFF;
```

Pour afficher l'état d'**AUTOCOMMIT**, on écrit la commande suivante :

```
SHOW AUTOCOMMIT;
```

1) La gestion des transactions

Pour gérer les transactions on utilise les commandes **COMMIT** et **ROLLBACK** :

❖ **COMMIT :** Permet de **valider** les modifications apportées à la base de données lors d'une transaction. Une transaction est un ensemble d'opérations qui sont effectuées sur la base de données et qui doivent être traitées de manière indivisible : soit toutes les opérations de la transaction sont exécutées, soit aucune n'est exécutée.

❖ **ROLLBACK :** Permet d'**annuler** les modifications apportées à la base de données lors d'une transaction. Si une transaction est annulée, toutes les modifications apportées à la base de données lors de cette transaction sont effacées et la base de données est restaurée à son état précédent.

1-2) La commande COMMIT

L'exemple suivant met à jour la table **employees** en augmentant le salaire de l'employé avec **l'ID 100** de **1000** dollars, et met à jour la table **departments** en diminuant le budget du **département 10** de **1000** dollars. Si toutes les instructions sont exécutées avec succès, la transaction **est validée (**à travers la commande **COMMIT)** et les modifications sont enregistrées dans la base de données.

```
-- Mise à jour de la table employees
UPDATE employees SET salary = salary + 1000 WHERE employee_id = 100;

-- Mise à jour de la table departments
UPDATE departments SET budget = budget - 1000 WHERE department_id = 10;

-- Fin de la transaction
COMMIT;
```

1-3) La commande ROLLBACK

Vous pouvez utiliser la commande ROLLBACK pour annuler toutes les modifications effectuées dans la transaction si une erreur se produit ou si vous souhaitez annuler la transaction. Par exemple:

```
-- Mise à jour de la table employees
UPDATE employees SET salary = salary + 1000 WHERE employee_id = 100;

-- Mise à jour de la table departments
UPDATE departments SET budget = budget - 1000 WHERE department_id = 10;

-- Fin de la transaction
ROLLBACK;
```