UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Formation SQL:

Le Data Manipulation Language (DML)

Utopios® Tous droits réservés

Introduction

Ce module est centré sur le **DML** *(Data Manipulation Language)* ou **langage de manipulation de données**
Il s'agit d'un **sous-langage** du langage SQL. Il est, comme son parent, un **langage de requête**.

Ses rôles sont principalement l'**insertion**, la **suppression** et **la modification** des données au sein d'une base. Il permet d'**interagir directement** avec les données dans une BDD établie mais il ne permet ni la création, ni la structuration de nos bases de données en elles-mêmes. (voir sous-langage **DDL**)

Utopios® Tous droits réservés

Avant-Propos

Cette section traite du **langage DML**, il nous permet la modification de données mais **pas la création de tables** dans lesquelles nous pouvons les stocker, rôle qui revient à un autre langage.
Une base de données est donc fourni avec le cours pour permettre la manipulation du langage, elle se prénomme **dql_bdd.sql**
Son implémentation dépendra de l'interface utilisée.

Si vous avez des connaissances sur d'autres sous-langages du SQL, n'hésitez pas à suivre le cours avec vos propres bases de données.

Utopios® Tous droits réservés

Pré-requis

La compréhension et l'utilisation du **langage DQL** est **fortement** recommandé **avant** d'aborder ce cours sur le **DML**. En effet, le DQL nous permet, grâce à la clause **SELECT**, de **voir** les manipulations qui ont été effectuées avec nos clauses de DML.

L'utilisation du DQL n'est pas obligatoire pour utiliser le DML, mais ils sont souvent appris conjointement et le DQL est une étape nécessaire dans notre parcours d'apprentissage pour constater la bonne utilisation de nos clauses.

Utopios® Tous droits réservés

Choisir sa base de données

Avant d'envoyer des requêtes, il me faut choisir dans quelle base de données je souhaite travailler, nos deux premières requêtes seront:

1. Pour afficher la liste des BDD disponibles:

```
SHOW DATABASES;
```

2. Pour utiliser celle de mon choix:

```
USE nom_de_la_bdd;
```

Utopios® Tous droits réservés

Voir les tables présentes dans la BDD

Une fois que je suis dans la base de données de mon choix, je peux consulter la liste des tables présentes:

- Pour MySQL/MariaDB:

```
SHOW TABLES;
```

Utopios® Tous droits réservés

Le CRUD (Create, Read, Update, Delete)

L'acronyme informatique anglais CRUD désigne les quatre opérations de base pour la persistance des données, en particulier le stockage d'informations en base de données.

Opération	SQL	Action
CREATE	INSERT	Ajouter un/des enregistrement(s)
READ	SELECT	Rechercher
UPDATE	UPDATE	Mettre à jour
DELETE	DELETE	Supprimer

Utopios® Tous droits réservés

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

INSERT INTO

Utopios® Tous droits réservés

INSERT INTO...

En SQL, la clause **INSERT INTO** est utilisée pour **ajouter des données** à une table existante.
Elle se construit de la façon suivante:

```
INSERT INTO nom_de_table (colonne1, colonne2, colonne3, ...)
VALUES (valeur1, valeur2, valeur3, ...);
```

Attention: Assurez-vous que les types de données correspondent entre les valeurs insérées et les colonnes de la table.

Utopios® Tous droits réservés

Syntaxes

Insertion de données dans toutes les colonnes :

```
INSERT INTO employees VALUES (1, 'John Doe', 'Manager', 50000);
```

Spécification des colonnes (recommandé) :

```
INSERT INTO employees (employee_id, employee_name, job_title, salary)
VALUES (1, 'John Doe', 'Manager', 50000);
```

Remarque: *Il est possible d'insérer plusieurs lignes d'un coup en les séparant par des virgules*

Utopios® Tous droits réservés

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

UPDATE

Utopios® Tous droits réservés

UPDATE

En SQL, la clause **UPDATE** est utilisée pour **mettre à jour des données** dans une table existante.
Elle se construit de la façon suivante:

```
UPDATE nom_de_table
SET colonne1 = valeur1, colonne2 = valeur2, ...
WHERE condition;
```

- **Colonne:** Les colonnes que vous souhaitez mettre à jour.
- **Valeur:** Les nouvelles valeurs que vous souhaitez assigner.
- **WHERE:** La condition pour spécifier les lignes à mettre à jour. Si cette clause est omise, toutes les lignes de la table seront UPDATE.

Utopios® Tous droits réservés

Exemples

- **Mise à jour de toutes les lignes dans une table :**

```
UPDATE employes
SET salaire = salaire * 1.1;
```

Cela augmente tous les salaires de 10%.

- **Mise à jour avec une condition :**

```
UPDATE employes
SET statut = 'Manager'
WHERE departement = 'Informatique';
```

Cela met à jour le statut des employés dont le département est "Informatique" pour les définir en tant que "Manager".

Utopios® Tous droits réservés

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

DELETE

Utopios® Tous droits réservés

DELETE

En SQL, la clause **DELETE** est utilisée pour **supprimer des données** dans une table existante.
Elle se construit de la façon suivante:

```
DELETE FROM nom_de_table
WHERE condition;
```

- nom_de_table: Le nom de la table dont vous souhaitez supprimer des données.
- WHERE: La condition pour spécifier les lignes à supprimer. **Si cette clause est omise, toutes les lignes de la table seront supprimées.**

Utopios® Tous droits réservés

Exemples

- Suppression de toutes les lignes dans une table :

```
DELETE FROM employes;
```

Cela supprime toutes les lignes de la table "employes".

- Suppression avec une condition :

```
DELETE FROM employes
WHERE date_embauche < '2022-01-01';
```

Cela supprime les employés embauchés avant le 1er janvier 2022.

Utopios® Tous droits réservés

DELETE/TRUNCATE

La commande **TRUNCATE** est également allouée à la suppression.

```
TRUNCATE TABLE nom_de_table;
```

Mais, contrairement au DELETE, elle sert généralement à supprimer la **table entière** plutôt qu'une suppression fine via des conditions.

TRUNCATE est plus performant mais également plus dangereux. L'action sera **irréversible**, contrairement au DELETE qu'on peut intégrer au sein de transactions (voir TCL)

Utopios® Tous droits réservés

Exercice: Résumé sur le langage DML

Considérez la table "Students" avec les colonnes suivantes :

```
CREATE TABLE Students (
    student_id INT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    age INT,
    grade VARCHAR(10)
);
```

Utopios® Tous droits réservés

Exercice: Résumé sur le langage DML

Partie 1: INSERT INTO

Ajoutez trois nouveaux étudiants à la table:

```
Prénom: Maria, Nom: Rodriguez, Âge: 21, Note: B
Prénom: Ahmed, Nom: Khan, Âge: 19, Note: A
Prénom: Emily, Nom: Baker, Âge: 22, Note: C
```

Partie 2: UPDATE

Mettez à jour la note de l'étudiant ayant le prénom "Maria" pour la changer de "B" à "A".

Augmentez l'âge de tous les étudiants de 1 an.

Utopios® Tous droits réservés

Exercice: Résumé sur le langage DML

Partie 3: DELETE

Supprimez l'étudiant ayant le prénom "Emily" de la table.
Supprimez tous les étudiants dont l'âge est inférieur à 20 ans.

Partie 4: TRUNCATE

Videz complètement la table "Students" en utilisant la commande TRUNCATE.

Utopios® Tous droits réservés

Correction

```sql
-- Partie 1: INSERT INTO
INSERT INTO Students (student_id, first_name, last_name, age, grade)
VALUES
    (1, 'Maria', 'Rodriguez', 21, 'B'),
    (2, 'Ahmed', 'Khan', 19, 'A'),
    (3, 'Emily', 'Baker', 22, 'C');

-- Partie 2: UPDATE
UPDATE Students
SET grade = 'A'
WHERE first_name = 'Maria';

UPDATE Students
SET age = age + 1;

-- Partie 3: DELETE
DELETE FROM Students WHERE first_name = 'Emily';
DELETE FROM Students WHERE age < 20;
TRUNCATE TABLE Students;
```

Utopios® Tous droits réservés