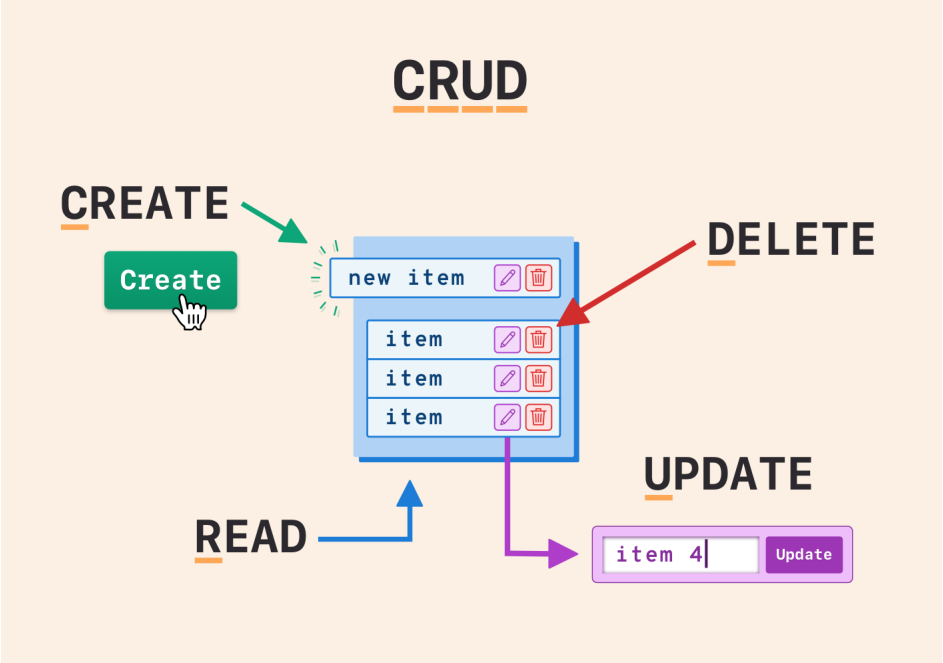
Ajouter des données dans une base de données

CRUD



INSERT INTO (create)

La commande `INSERT INTO` est utilisée pour insérer de nouvelles lignes dans une table d'une base de données MySQL. Cette commande vous permet de spécifier les valeurs pour chaque colonne de la nouvelle ligne que vous souhaitez insérer.

Voici un exemple de syntaxe pour insérer une nouvelle ligne dans une table:

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

Dans cet exemple, `nom_de_la_table` est le nom de la table dans laquelle vous souhaitez insérer une nouvelle ligne. `colonne1`, `colonne2`, `colonne3`, … sont les noms des colonnes pour lesquelles vous souhaitez spécifier des valeurs. `valeur1`, `valeur2`, `valeur3`, … sont les valeurs que vous souhaitez insérer pour chaque colonne.

Par exemple, si vous avez une table `clients` avec les colonnes `nom`, `prenom` et `age`, et que vous souhaitez insérer une nouvelle ligne avec les valeurs “Dupont”, “Jean” et 42, vous pouvez utiliser la commande suivante:

```
INSERT INTO clients (nom, prenom, age)
VALUES ('Dupont', 'Jean', 42);
```

Cette commande ajoutera une nouvelle ligne à la table `clients` avec les valeurs spécifiées pour chaque colonne.

Il est également possible d'insérer plusieurs lignes en une seule commande en utilisant la syntaxe suivante:

```
INSERT INTO nom_de_la_table (colonne1, colonne2, colonne3, ...)
VALUES (valeur1a, valeur2a, valeur3a, ...),
       (valeur1b, valeur2b, valeur3b, ...),
       ...
       (valeur1n, valeur2n, valeur3n, ...);
```

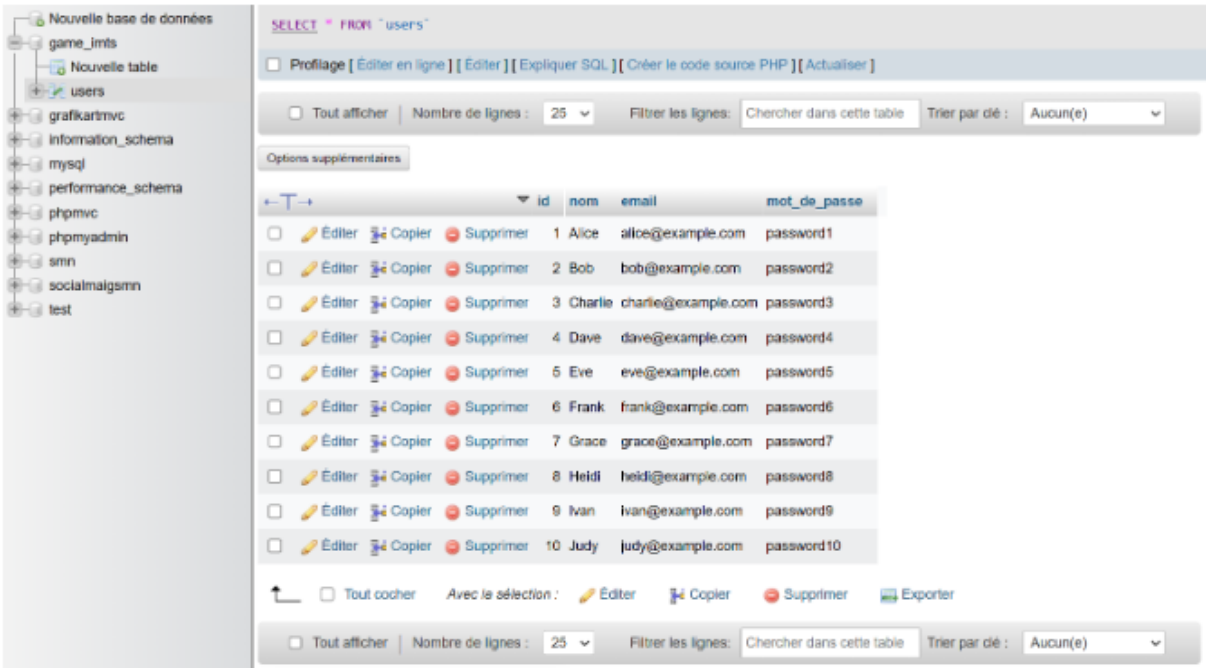
Dans cet exemple, chaque paire de parenthèses représente une ligne à insérer dans la table. Vous pouvez spécifier autant de lignes que vous le souhaitez en ajoutant des paires de parenthèses supplémentaires.

Exemple

Voici un exemple de code MySQL qui insère 10 utilisateurs avec des données fictives dans la table `users` :

```
INSERT INTO users (nom, email, mot_de_passe)
VALUES
('Alice', 'alice@example.com', 'password1'),
('Bob', 'bob@example.com', 'password2'),
('Charlie', 'charlie@example.com', 'password3'),
('Dave', 'dave@example.com', 'password4'),
('Eve', 'eve@example.com', 'password5'),
('Frank', 'frank@example.com', 'password6'),
('Grace', 'grace@example.com', 'password7'),
('Heidi', 'heidi@example.com', 'password8'),
('Ivan', 'ivan@example.com', 'password9'),
('Judy', 'judy@example.com', 'password10');
```

Ce code utilise la commande `INSERT INTO` pour insérer des données dans la table `users`. Les colonnes `nom`, `email` et `mot_de_passe` sont spécifiées après le nom de la table, et les valeurs à insérer sont spécifiées après le mot-clé `VALUES`. Chaque ligne de valeurs représente un utilisateur différent.



A titre d'entraînement vous pouvez essayer d'ajouter de nouveaux utilisateurs sans email, sans nom et sans mot de passe.

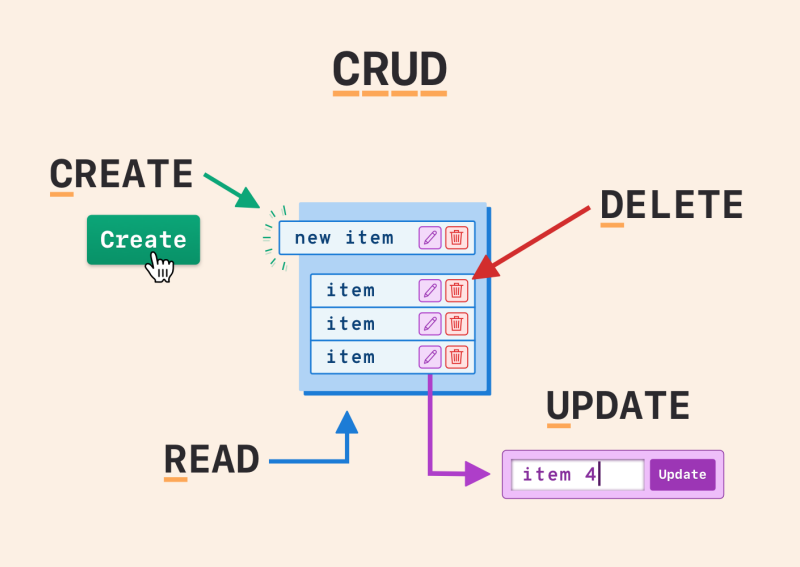
Résumé :

→ La commande INSERT INTO ajoute de nouvelle données dans la table.

→ **Attention :** les champs marqués comme **NOT NULL** doivent être impérativement remplis.

Lire les données dans la base de données

CRUD

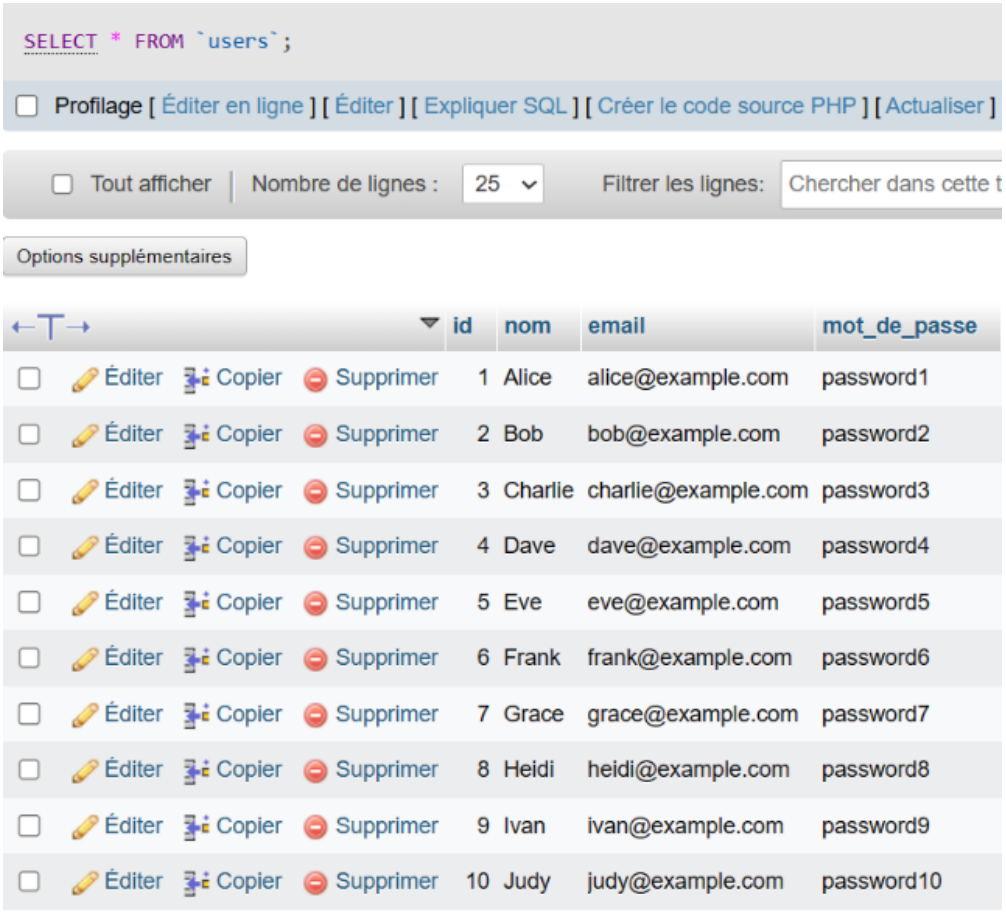


Select (Read)

Pour lire toutes les données de la table `users`, vous pouvez utiliser la commande `SELECT` avec l'astérisque (`*`) pour sélectionner toutes les colonnes. Voici un exemple de code MySQL qui lit toutes les données de la table `users` :

```
SELECT * FROM users;
```

Cette commande utilise la clause `FROM` pour spécifier la table à partir de laquelle les données doivent être lues, et l'astérisque (`*`) pour sélectionner toutes les colonnes de cette table. Le résultat de cette commande sera une liste de toutes les lignes de la table `users`, avec toutes les colonnes pour chaque ligne.



```
SELECT * FROM `users`;
```

Profilage [Éditer en ligne] [Éditer] [Expliquer SQL] [Créer le code source PHP] [Actualiser]

Tout afficher | Nombre de lignes : 25 Filtrer les lignes: Chercher dans cette t

Options supplémentaires

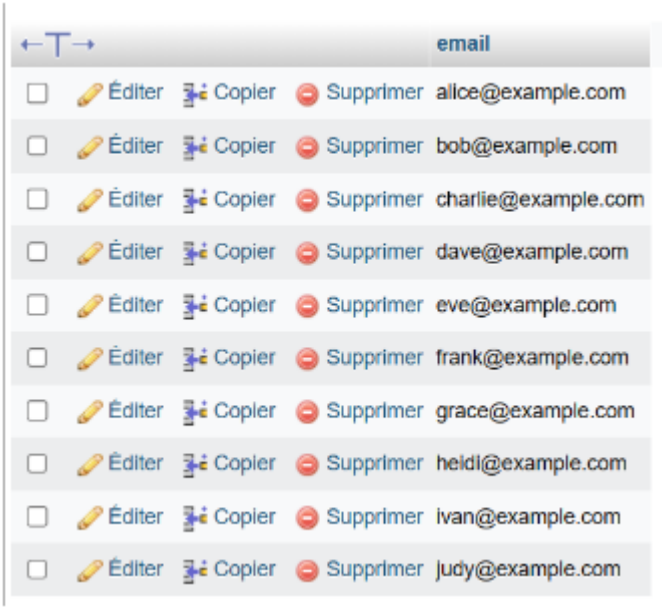
	id	nom	email	mot_de_passe
Éditer Copier Supprimer	1	Alice	alice@example.com	password1
Éditer Copier Supprimer	2	Bob	bob@example.com	password2
Éditer Copier Supprimer	3	Charlie	charlie@example.com	password3
Éditer Copier Supprimer	4	Dave	dave@example.com	password4
Éditer Copier Supprimer	5	Eve	eve@example.com	password5
Éditer Copier Supprimer	6	Frank	frank@example.com	password6
Éditer Copier Supprimer	7	Grace	grace@example.com	password7
Éditer Copier Supprimer	8	Heidi	heidi@example.com	password8
Éditer Copier Supprimer	9	Ivan	ivan@example.com	password9
Éditer Copier Supprimer	10	Judy	judy@example.com	password10

Lire une seule colonne de la table

Pour lire uniquement la colonne `email` de la table `users`, vous pouvez utiliser la commande `SELECT` en spécifiant le nom de la colonne après le mot-clé `SELECT`. Voici un exemple de code MySQL qui lit uniquement la colonne `email` de la table `users` :

```
SELECT email FROM users;
```

Cette commande utilise la clause `FROM` pour spécifier la table à partir de laquelle les données doivent être lues, et le nom de la colonne `email` pour sélectionner uniquement cette colonne. Le résultat de cette commande sera une liste de toutes les lignes de la table `users`, avec uniquement la colonne `email` pour chaque ligne.



←T→	email
Éditer Copier Supprimer	alice@example.com
Éditer Copier Supprimer	bob@example.com
Éditer Copier Supprimer	charlie@example.com
Éditer Copier Supprimer	dave@example.com
Éditer Copier Supprimer	eve@example.com
Éditer Copier Supprimer	frank@example.com
Éditer Copier Supprimer	grace@example.com
Éditer Copier Supprimer	heidi@example.com
Éditer Copier Supprimer	ivan@example.com
Éditer Copier Supprimer	judy@example.com

Lire 2 colonnes

SELECT nom, email FROM `users`;

←T→	nom	email
Éditer Copier Supprimer	Alice	alice@example.com
Éditer Copier Supprimer	Bob	bob@example.com
Éditer Copier Supprimer	Charlie	charlie@example.com
Éditer Copier Supprimer	Dave	dave@example.com
Éditer Copier Supprimer	Eve	eve@example.com
Éditer Copier Supprimer	Frank	frank@example.com
Éditer Copier Supprimer	Grace	grace@example.com
Éditer Copier Supprimer	Heidi	heidi@example.com
Éditer Copier Supprimer	Ivan	ivan@example.com
Éditer Copier Supprimer	Judy	judy@example.com

La commande `SELECT` est utilisée pour récupérer des données d'une ou plusieurs tables d'une base de données MySQL. Cette commande vous permet de spécifier les colonnes que vous souhaitez récupérer, ainsi que les conditions que les lignes doivent remplir pour être incluses dans les résultats.

Voici un exemple de syntaxe pour récupérer des données d'une table:

```
SELECT colonne1, colonne2, colonne3, ...
FROM nom_de_la_table
WHERE condition;
```

Dans cet exemple, `colonne1`, `colonne2`, `colonne3`, … sont les noms des colonnes que vous souhaitez récupérer. `nom_de_la_table` est le nom de la table à partir de laquelle vous souhaitez récupérer des données. `condition` est une expression qui spécifie les conditions que les lignes doivent remplir pour être incluses dans les résultats.

Par exemple, si vous avez une table `clients` avec les colonnes `nom`, `prenom` et `age`, et que vous souhaitez récupérer toutes les lignes où l'âge est supérieur à 18, vous pouvez utiliser la commande suivante:

```
SELECT nom, prenom, age
FROM clients
WHERE age > 18;
```

Cette commande renverra toutes les lignes de la table `clients` où la valeur de la colonne `age` est supérieure à 18.

Il est également possible d'utiliser des fonctions d'agrégation pour effectuer des calculs sur les données récupérées. Par exemple, pour calculer l'âge moyen des clients, vous pouvez utiliser la commande suivante:

```
SELECT AVG(age)
FROM clients;
```

Cette commande calcule la moyenne des valeurs de la colonne `age` pour toutes les lignes de la table `clients`.

Vous pouvez également utiliser des jointures pour récupérer des données à partir de plusieurs tables en même temps. Par exemple, si vous avez une table `commandes` avec une clé étrangère faisant référence à la table `clients`, vous pouvez utiliser une jointure pour récupérer les noms des clients et les détails de leurs commandes en utilisant la commande suivante:

```
SELECT clients.nom, commandes.details
FROM clients
INNER JOIN commandes
ON clients.id = commandes.client_id;
```

Cette commande effectue une jointure entre les tables `clients` et `commandes` en utilisant la clé étrangère `client_id` dans la table `commandes`. Elle renvoie les valeurs des colonnes `nom` et `details` pour toutes les lignes correspondantes dans les deux tables.

Résumé:

- Select permet de lire les données dans une table.
- On peut lire l'intégralité d'une table avec SELECT * FROM nom_table
- On peut lire uniquement certaines colonnes en indiquant le nom des tables.
- SELECT champ01,champs02 FROM nom_table

Mettre à jour les données

→UPDATE

♦ Updater une ligne

Pour mettre à jour l'adresse e-mail de l'utilisateur avec l'`id` 1 dans la table `users`, vous pouvez utiliser la commande `UPDATE` en spécifiant la colonne à mettre à jour et la nouvelle valeur à utiliser. Voici un exemple de code MySQL qui met à jour l'adresse e-mail de l'utilisateur avec l'`id` 1 dans la table `users` :

```
UPDATE users
SET email = 'alice2@example.com'
WHERE id = 1;
```

Cette commande utilise la clause `SET` pour spécifier la colonne à mettre à jour (`email`) et la nouvelle valeur à utiliser (`'alice2@example.com'`). La clause `WHERE` est utilisée pour spécifier la ligne à mettre à jour (celle où `id = 1`). Cette commande mettra à jour l'adresse e-mail de l'utilisateur avec l'`id` 1 dans la table `users`.

♦ WHERE

La commande `WHERE` en SQL est utilisée pour filtrer les lignes d'une table en fonction d'une condition spécifiée. Elle peut être utilisée avec des commandes telles que `SELECT`, `UPDATE` et `DELETE` pour spécifier les lignes à sélectionner, mettre à jour ou supprimer.

La syntaxe de base de la commande `WHERE` est la suivante :

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

La `condition` peut être une expression qui utilise des opérateurs de comparaison tels que `=`, `<>`, `<`, `>`, `<=`, `>=`, ainsi que des opérateurs logiques tels que `AND`, `OR` et `NOT`. Par exemple, pour sélectionner toutes les lignes de la table `users` où la colonne `age` est supérieure à 18, vous pouvez utiliser la commande suivante :

```
SELECT * FROM users
WHERE age > 18;
```

♦ Updater plusieurs lignes

Pour mettre à jour les mots de passe des utilisateurs `Alice` et `Ivan` dans la table `users` et les changer en `password2`, vous pouvez utiliser la commande `UPDATE` en spécifiant la colonne à mettre à jour, la nouvelle valeur à utiliser et les lignes à mettre à jour. Voici un exemple de code MySQL qui met à jour les mots de passe des utilisateurs `Alice` et `Ivan` dans la table `users` :

```
UPDATE users
SET mot_de_passe = 'password2'
WHERE nom = 'Alice' OR nom = 'Ivan';
```

Cette commande utilise la clause `SET` pour spécifier la colonne à mettre à jour (`mot_de_passe`) et la nouvelle valeur à utiliser (`'password2'`). La clause `WHERE` est utilisée pour spécifier les lignes à mettre à jour (celles où `nom = 'Alice'` ou `nom = 'Ivan'`). Cette commande mettra à jour les mots de passe des utilisateurs `Alice` et `Ivan` dans la table `users`.

Rappelons ici, que changer les mots de passe en une seule valeur fixe n'est pas une pratique sécurisée. Il est recommandé d'utiliser des mots de passe uniques et forts pour chaque utilisateur, et de les stocker de manière sécurisée (par exemple, en utilisant un hachage avec un sel).

♦ Mettre à jour toutes les lignes dans une colonne.

Pour mettre à jour tous les mots de passe dans la table `users` et les changer tous en `password`, vous pouvez utiliser la commande `UPDATE` en spécifiant la colonne à mettre à jour et la nouvelle valeur à utiliser. Voici un exemple de code MySQL qui met à jour tous les mots de passe dans la table `users` :

```
UPDATE users
SET mot_de_passe = 'password';
```

Cette commande utilise la clause `SET` pour spécifier la colonne à mettre à jour (`mot_de_passe`) et la nouvelle valeur à utiliser (`'password'`). Comme aucune clause `WHERE` n'est utilisée, cette commande mettra à jour tous les mots de passe dans la table `users`.

		id	nom	email	mot_de_passe
☐	Éditer Copier Supprimer	1	Alice	alice2@example.com	password
☐	Éditer Copier Supprimer	2	Bob	bob@example.com	password
☐	Éditer Copier Supprimer	3	Charlie	charlie@example.com	password
☐	Éditer Copier Supprimer	4	Dave	dave@example.com	password
☐	Éditer Copier Supprimer	5	Eve	eve@example.com	password
☐	Éditer Copier Supprimer	6	Frank	frank@example.com	password
☐	Éditer Copier Supprimer	7	Grace	grace@example.com	password
☐	Éditer Copier Supprimer	8	Heidi	heidi@example.com	password
☐	Éditer Copier Supprimer	9	Ivan	ivan@example.com	password
☐	Éditer Copier Supprimer	10	Judy	judy@example.com	password

Résumé

◗ UPDATE

Cette commande est utilisée pour mettre à jour les données existantes dans une table. Elle permet de modifier les valeurs d'une ou plusieurs colonnes pour une ou plusieurs lignes de la table. EXemple : UPDATE table_name

◗ SET

Cette clause est utilisée pour spécifier les colonnes à mettre à jour et les nouvelles valeurs à utiliser. Il est possible de mettre à jour plusieurs colonnes en même temps en les séparant par des virgules.

SET column1 = value1, column2 = value2, ...

◗ WHERE

Cette commande est utilisée pour filtrer les lignes d'une table en fonction d'une condition spécifiée.
Elle peut être utilisée avec des commandes telles que SELECT, UPDATE et DELETE pour spécifier les lignes à sélectionner, mettre à jour ou supprimer.

Supprimer des données

→ DELETE

La commande `DELETE` est utilisée pour supprimer des lignes d'une table dans une base de données MySQL. Cette commande vous permet de spécifier les conditions que les lignes doivent remplir pour être supprimées.

Voici un exemple de syntaxe pour supprimer des lignes d'une table:

```
DELETE FROM nom_de_la_table
WHERE condition;
```

Dans cet exemple, `nom_de_la_table` est le nom de la table à partir de laquelle vous souhaitez supprimer des lignes. `condition` est une expression qui spécifie les conditions que les lignes doivent remplir pour être supprimées.

Par exemple, si vous avez une table `clients` avec les colonnes `nom`, `prenom` et `age`, et que vous souhaitez supprimer toutes les lignes où l'âge est inférieur à 18, vous pouvez utiliser la commande suivante:

```
DELETE FROM clients
WHERE age < 18;
```

Cette commande supprimera toutes les lignes de la table `clients` où la valeur de la colonne `age` est inférieure à 18.

Il est important de noter que la commande `DELETE` supprime définitivement les lignes de la table, il est donc recommandé de l'utiliser avec précaution. Si vous souhaitez simplement masquer des lignes sans les supprimer définitivement, vous pouvez utiliser une colonne supplémentaire pour marquer les lignes comme étant "supprimées" sans réellement les supprimer de la table.

Voici un exemple, pour supprimer l'utilisateur `Ivan` de la table `users`, vous pouvez utiliser la commande `DELETE` en spécifiant la ligne à supprimer. Voici un exemple de code MySQL qui supprime l'utilisateur `Ivan` de la table `users` :

```
DELETE FROM users
WHERE nom = 'Ivan';
```

Cette commande utilise la clause `FROM` pour spécifier la table à partir de laquelle les données doivent être supprimées, et la clause `WHERE` pour spécifier la ligne à supprimer

(celle où `nom = 'Ivan'`). Cette commande supprimera l'utilisateur `Ivan` de la table `users`.

⏵ Vider une table

Pour vider la table `users`, vous pouvez utiliser la commande `DELETE` sans spécifier de condition dans la clause `WHERE`. Voici un exemple de code MySQL qui vide la table `users` :

```
DELETE FROM users;
```

Cette commande utilise la clause `FROM` pour spécifier la table à partir de laquelle les données doivent être supprimées. Comme aucune clause `WHERE` n'est utilisée, cette commande supprimera toutes les lignes de la table `users`, la vidant complètement.

⏵ Supprimer une table

Pour supprimer la table `users`, vous pouvez utiliser la commande `DROP TABLE`. Voici un exemple de code MySQL qui supprime la table `users` :

```
DROP TABLE users;
```

Cette commande utilise la clause `TABLE` pour spécifier la table à supprimer. Une fois cette commande exécutée, la table `users` sera supprimée de la base de données et toutes les données qu'elle contient seront perdues.

⏵ Supprimer une base de données

Pour supprimer la base de données `GAME_IMTS`, vous pouvez utiliser la commande `DROP DATABASE`. Voici un exemple de code MySQL qui supprime la base de données `GAME_IMTS` :

```
DROP DATABASE GAME_IMTS;
```

Cette commande utilise la clause `DATABASE` pour spécifier la base de données à supprimer. Une fois cette commande exécutée, la base de données `GAME_IMTS` sera supprimée et toutes les données qu'elle contient seront perdues.

Résumé

```
DELETE FROM users
WHERE nom = 'Ivan';

DELETE FROM users;

DROP TABLE users;

DROP DATABASE GAME_IMTS;
```