Langage de manipulation de données

Pour modifier les données d'une base, l'utilisateur dispose de 3 ordres SQL :

- `INSERT`
- `UPDATE`
- `DELETE`

INSERT

L'ordre `INSERT` permet d'ajouter des lignes dans une table (ou une vue).

Il est composé de deux clauses et répond à la syntaxe suivante :

```
INSERT
INTO NOM_DE_TABLE [(NOM_COLONNE_1 [,.. NOM_COLONNE_n)]
VALUES ('VALEUR_COLONNE_1' [,.. 'VALEUR_COLONNE_n'])
```

- `INTO` : nom de la table dans laquelle on souhaite créer des enregistrements
  La liste des colonnes est optionnelle. Elle permet de préciser l'ordre des colonnes dans lesquelles on va insérer les valeurs (la 1e valeur sera insérée dans la colonne spécifiée en 1e, et ainsi de suite) ; si cette précision n'est pas faite, les valeurs seront insérées dans l'ordre par défaut des colonnes de la table (= lorsque la table a été créée)
- `VALUES` : liste des valeurs à insérer dans chaque colonne spécifiée

Exemples

A partir

- d'une table EMPLOYES recensant les n°, nom, prénom, n° de département (service) et salaire d'employés :

  ```
  EMPLOYES (noemp, nom, prenom, dept, salaire)
  ```

- d'une table DEPART contenant les n° et noms de départements :

  ```
  DEPART (no_dept, nom_dept)
  ```

Exemple 1

On souhaite créer la fiche de l'employé *00140*, ayant pour nom *REEVES*, prénom *HUBERT*, travaillant dans le département *A00*, pour un salaire de *2100€* :

```
INSERT INTO employes (noemp, nom, prenom, dept, salaire)
VALUES ('00140', 'REEVES', 'HUBERT', 'A00', 2100)
```

On donne ici une valeur pour chacun des attributs spécifiés dans l'ordre `INSERT` ; les valeurs de la clause `VALUES` doivent correspondre avec la liste des colonnes.

Exemple 2

Les attributs non spécifiés prennent la valeur `NULL`. En effet, on peut juste insérer l'employé *00140*, de nom *REEVES*, de prénom *HUBERT* dans le département *A00*, sans spécifier son salaire :

```
INSERT INTO employes (noemp, nom, prenom, dept)
VALUES ('00140','REEVES','HUBERT','A00')
```

La colonne *salaire* prendra alors la valeur `NULL`, pour cette ligne. Et si cette colonne n'a pas été spécifiée comme pouvant être nulle lorsque la table a été créée, une erreur d'exécution sera générée.

Exemple 3

La liste des colonnes peut être "omise" ; auquel cas, l'ordre d'insertion des valeurs respectera l'ordre des colonnes par défaut dans la table, et concernera *toutes* les colonnes :

```
INSERT INTO employes
VALUES ('00140', 'REEVES', 'HUBERT', 'A00', 2100)
```

Exemple 4

Plusieurs lignes peuvent être ajoutées dans la table en une seule instruction, avec un seul `VALUES` :

```
INSERT INTO employes
VALUES
('00140','REEVES','HUBERT','A00', 2100),
('00150','JACQUARD','ALBERT','B00', 1800),
('00999','LOPER', 'DAVE', 'C00', 900)
```

Remarque complémentaire

Une colonne ayant une propriété `AUTO_INCREMENT` ne fait habituellement pas partie de la liste des colonnes lors de l'`INSERT`, car sa valeur est attribuée automatiquement.
Lorsqu'on crée un enregistrement sans spécifier l'ordre des colonnes, on ne précise pas non plus la valeur de la colonne auto-incrémentée ; celle-ci sera automatiquement alimentée lorsque la ligne sera ajoutée.

On peut toutefois forcer sa valeur en spécifiant la colonne dans la liste de l'ordre d'ajout.

UPDATE

L'ordre `UPDATE` est utilisé pour modifier des lignes de tables existantes.

Il est composé de deux ou trois clauses et répond à la syntaxe suivante :

```
UPDATE NOM_DE_TABLE
SET NOM_COLONNE_1 = 'NOUVELLE_VALEUR_COLONNE_1' [,.. NOM_COLONNE_n = 'NOUVELLE_VALEUR_COLONNE_n']
[WHERE <condition>]
```

- `SET` : nom des colonnes à modifier, avec attribution d'une nouvelle valeur chacune
- `WHERE` : critère(s) de sélection des lignes à mettre à jour ; si cette clause n'est pas spécifiée, *toutes* les lignes de la table seront mises à jour !

Exemple 1

On modifie le salaire de l'employé *LOPER*, qui gagne désormais 1000 € (au lieu de 900 €):

```
UPDATE employes
SET
salaire = 1000
WHERE nom = 'LOPER'
```

Exemple 2

On modifie plusieurs valeurs (nom, prénom, adresse) de l'employé ayant le matricule *00999* :

```
UPDATE employes
SET
nom = 'LOPER',
prenom = 'Dave',
adresse = '15 avenue Tella'
WHERE noemp = '00999'
```

Exemple 3

On augmente le salaire de 20% de tous les employés :

```
UPDATE employes
SET
salaire = salaire * 1.2
```

> L'absence de clause `WHERE` implique donc la mise à jour de tous les enregistrements de la table !

Exemple 4

On augmente le salaire de 20% de l'employé ayant le matricule *00040* :

```
UPDATE employes
SET salaire = salaire * 1.2
WHERE noemp = '00040'
```

Exemple 5

On modifie le salaire (+20%) de l'employé ayant le matricule *00040*, et son département pour l'affecter au service *A40* :

```
UPDATE employes
SET salaire = salaire * 1.2,
dept = 'A40'
WHERE noemp = '00040'
```

DELETE

L'ordre `DELETE` permet de supprimer une ou plusieurs lignes d'une table.

Il est composé de deux ou trois clauses et répond à la syntaxe suivante :

```
DELETE [FROM] NOM_DE_TABLE
[WHERE <condition>]
```

- `FROM` : nom de la table où des lignes doivent être supprimées
- `WHERE` : critère(s) de sélection des lignes à supprimer ; si cette clause n'est pas spécifiée, *toutes* les lignes de la table seront supprimées, et la table sera vide !

Exemple 1

On supprime **tous** les employés de la table *EMPLOYES* :

```
DELETE FROM employes
```

Exemple 2

On supprime les employés du département *E21* :

```
DELETE FROM employes
WHERE nodept = 'E21'
```

Exemple 3

On supprime les employés du département *E21* qui habitent Amiens :

```
DELETE FROM employes
WHERE nodept = 'E21'
AND ville = 'Amiens'
```