

Introduction au langage SQL

Les bases de données relationnelles

Le langage SQL est un langage universel destiné à travailler sur des bases de données relationnelles. Nous considérerons ici qu'une base de données relationnelle est constituée d'un ensemble de tables, chaque table pouvant être décrite sous la forme d'un tableau où les colonnes correspondent à des variables et les lignes à des enregistrements.

Par exemple, le tableau ci-dessous donne la valeur des variables production (P1), consommation intermédiaire (P2) et valeur ajoutée (B1) de plusieurs secteurs d'activité (AE).

AE	P1	P2	B1
A01	800	300	500
A02	500	400	100
A02	600	450	150
A03	200	50	150
B01	700	400	300
B01	900	500	400

La première ligne du tableau donne le nom des différentes variables, les autres lignes correspondent aux enregistrements. Dans une table, le nombre d'enregistrements n'est pas fixé et est seulement limité par les capacités de la mémoire, mais la contrainte fondamentale est que chaque enregistrement doit avoir la même structure, c'est-à-dire comporter les mêmes variables et cela dans le même ordre. Ainsi, si l'on ajoutait un nouvel enregistrement à cette table, il devrait impérativement comprendre les variables AE, P1, P2 et B1 dans cet ordre.

La commande SELECT

La commande SELECT est l'une des plus importantes du langage SQL, c'est elle qui permet de sélectionner les variables que l'on veut lire dans une table. Elle est toujours accompagnée au minimum de la clause FROM qui permet de préciser sur quelle table on va travailler. La commande SELECT doit être suivie de la liste des variables retenues, les variables étant séparées par des virgules. La clause FROM doit être suivie du nom de la table. Un ensemble de commandes SQL se nomme une requête et chaque requête doit se terminer par un point-virgule.

Par exemple, si la table ci-dessus s'appelle PROD la requête suivante :

```
SELECT AE, B1  
FROM PROD  
;
```

renvoie :

AE	B1
A01	500
A02	100
A02	150
A03	150
B01	300

B01	400
-----	-----

On peut aussi avec la commande SELECT sélectionner des combinaisons de variables et leur donner un nom nouveau (un alias) en utilisant la clause AS, c'est un moyen simple de faire certains calculs avec SQL. Par exemple :

```
SELECT AE, 2*P1 AS DP1, P2+B1 AS NP1
FROM PROD
;
```

Renvoie :

AE	DP1	NP1
A01	1600	800
A02	1000	500
A02	1200	600
A03	400	200
B01	1400	700
B01	1800	900

La commande SELECT * permet de sélectionner l'ensemble des variables sans avoir besoin de les détailler dans une liste.

La clause WHERE

La clause WHERE permet de sélectionner les enregistrements correspondant à un certain critère. La clause WHERE vient obligatoirement après la clause FROM et les expressions alphanumériques doivent être encadrées par de simples quotes (et non des guillemets). Par exemple :

```
SELECT AE, B1
FROM PROD
WHERE AE='A02'
;
```

renvoie :

AE	B1
A02	100
A02	150

Ou encore :

```
SELECT AE, B1
FROM PROD
WHERE P1>700
;
```

renvoie :

AE	B1
A01	500

B01	400
-----	-----

Il est possible de réaliser des critères complexes en utilisant notamment les opérateurs AND, OR et NOT. Par exemple :

```
SELECT AE, B1
FROM PROD
WHERE AE='A02' AND P1>500
;
```

renvoie :

AE	B1
A02	150

On peut également travailler avec des listes en utilisant l'opérateur IN. La liste des valeurs que peut prendre la variable est placée entre parenthèses, chaque élément de la liste étant séparé des autres par des virgules et placé entre quotes s'il correspond à une valeur alphanumérique. Par exemple :

```
SELECT AE, B1
FROM PROD
WHERE AE IN ('A01','B01')
;
```

renvoie :

AE	B1
A01	500
B01	300
B01	400

La clause GROUP BY

On peut également utiliser dans des requêtes SQL des fonctions d'agrégation comme la somme SUM, le nombre COUNT, la moyenne AVG, la valeur maximale MAX ou la valeur minimale MIN. Ce sont ces fonctions d'agrégation qui font tout l'intérêt de SQL pour le comptable national. La fonction COUNT(*) permet de compter tous les enregistrements d'une table. Ces fonctions d'agrégation peuvent être utilisées sur l'ensemble de la table ou sur des regroupements. Dans ce dernier cas, il faudra utiliser la clause GROUP BY qui permet de regrouper les enregistrements par critères. Cette clause GROUP BY doit se placer après la clause WHERE. Par exemple, la requête suivante permet de calculer la somme de la variable B1 pour chaque modalité de l'activité AE :

```
SELECT AE, SUM(B1)
FROM PROD
GROUP BY AE
;
```

Elle renvoie :

AE	B1
A01	500
A02	200
A03	150
B01	700

La commande INSERT INTO

Pour entrer des données dans une table SQL on peut utiliser la commande INSERT INTO. Celle-ci peut s'utiliser de deux manières différentes. La première consiste à introduire les nouveaux enregistrements un par un. Ainsi la commande INSERT INTO doit être suivie en combinaison avec la clause VALUES. La commande INSERT INTO doit être suivie par le nom de la table puis, de manière facultative par la liste des variables mise entre parenthèses. La clause VALUES doit être suivie de la liste des valeurs, mise entre parenthèses, chaque élément étant séparé par une virgule. Les variables alphanumériques doivent être mises entre quotes. Par exemple :

```
INSERT INTO Prod (AE, P1, P2, B1)
VALUES ('A03', 2000, 800, 1200)
;
```

L'ordre de la liste des valeurs doit correspondre à celui de la liste des variables. Lorsque la liste des variables a été omise dans la commande INSERT INTO l'ordre de la liste des valeurs doit correspondre à celui des variables tel qu'il a été défini au moment de la création de la table.

La deuxième manière d'utiliser la commande INSERT INTO est d'introduire dans une table une série de valeurs à partir d'une autre table. La commande INSERT INTO sera alors utilisée en liaison avec la commande SELECT FROM. Par, exemple, si nous disposons d'une table nommée Production contenant les mêmes variables que la table Prod, la requête suivante permettra d'insérer dans la table Prod tous les enregistrements de la table Production :

```
INSERT INTO Prod
SELECT * FROM Production
;
```

La commande DELETE FROM

Il est souvent utile de détruire des enregistrements d'une table. Cela peut se faire avec la commande DELETE FROM qui est le plus souvent associée à la clause WHERE. La commande DELETE FROM doit être suivie du nom de la table, la clause WHERE fonctionne comme avec la commande SELECT. Par exemple, la requête suivante élimine de la table Prod tous les enregistrements pour lesquels la variable P1 est inférieure à 1000 :

```
DELETE FROM Prod
WHERE P1<1000
;
```

Lorsque la clause WHERE est omise c'est l'ensemble des enregistrements de la table qui est supprimé.

Les jointures

L'une des grandes forces de SQL est de permettre le travail avec plusieurs tables. Par exemple,

désirions calculer des valeurs à partir d'indices de prix. Nous pouvons stocker nos indices de prix dans une table PRIX qui se présentera de la manière suivante :

AE	INDICE
A01	110
A02	120
A03	100
B01	120

Pour obtenir les valeurs de la production, nous pouvons combiner la table PROD qui contient les volumes de la production et la table PRIX qui contient les indices de prix. Nous allons, pour cela, réaliser une jointure. Cette jointure va créer de nouveaux enregistrements qui reprendront les variables sélectionnées de chacune des deux tables. Chaque enregistrement de la première table sera associé à chaque enregistrement de la deuxième table, c'est-à-dire que la jointure réalise un produit cartésien. Dans notre exemple, la table PROD comporte 6 enregistrements et la table PRIX 4 enregistrements. La jointure va donc générer $6 \times 4 = 24$ enregistrements.

Cependant, il n'est en général, pas intéressant de réaliser toutes les combinaisons possibles entre les deux tables. Ainsi, dans notre exemple, nous allons associer un enregistrement du tableau PROD à l'enregistrement de la table PRIX qui a la même activité afin d'obtenir une valeur qui a un sens. Aussi, la jointure devra-t-elle être accompagnée d'une clause établissant un lien entre les deux tables afin de n'associer que des enregistrements de même activité. Dans la commande SELECT on fera apparaître les variables des deux tables préfixées du nom des tables, dans la clause FROM les noms des deux tables apparaîtront séparées d'une virgule. Ainsi, par exemple, la requête suivante :

```
SELECT PROD.AE, PROD.P1, PRIX.INDICE, PROD.P1*PRIX.INDICE/100 AS VALEUR
FROM PROD, PRIX
WHERE PROD.AE=PRIX.AE
;
```

renvoie :

AE	P1	INDICE	VALEUR
A01	800	110	880
A02	500	120	600
A02	600	120	720
A03	200	100	200
B01	700	120	840
B01	900	120	1080

On peut également utiliser des alias de nom de tables dans les requêtes pour alléger leur écriture. L'utilisation d'alias est même obligatoire lorsque les noms de tables ont une extension, par exemple en .csv. Ainsi la requête précédente peut encore s'écrire :

```
SELECT P.AE, P.P1, X.INDICE, P.P1*X.INDICE/100 AS VALEUR
FROM PROD P, PRIX X
WHERE P.AE=X.AE
;
```

L'alias est défini dans la clause WHERE et il apparaît après le nom de la table, séparé par un espace.

Ce texte n'engage que son auteur : Francis Malherbe