SQL - LID

Interrogation des Données

Dimitri

24/12/2012

24 décembre 2012

Sommaire :

I. Projection.

Pour afficher toutes les colonnes d'une table il faut avoir recours à la requête suivante :

SELECT * FROM table ;

Ce qui nous donne par exemple :

```
1  SELECT *
2  FROM client;
```

Pour afficher uniquement les colonnes souhaitées il faut avoir recours à la requête suivante : SELECT COLUMN1, COLUMN2 FROM table ;

Ce qui nous donne par exemple :

```
1  SELECT code_prod, désignation
2  FROM produits;
```

Pour afficher les colonnes souhaitées sans duplication des lignes égales (doublons), il faut avoir recours à la requête suivante : SELECT DISTINCT COLUMN FROM table ;

Ce qui nous donne par exemple :

```
1  SELECT DISTINCT num_com
2  FROM commandes;
```

Il possible aussi d'afficher des colonnes mais de changer le nom apparaissant pour rendre plus compréhensible la lecture de la colonne, pour ce faire on utilise la requête suivante : SELECT COLUMN AS alias FROM table ;

Ce qui nous donne par exemple :

```
1  SELECT num_com AS NUM, date_com AS Date
2  FROM commandes;
```

Il est possible d'afficher le contenu d'une colonne tout en effectuant des opérations sur les données à afficher et ainsi afficher ce résultat à la place, on a recours à la requête suivante : SELECT expression FROM table ;

Ce qui nous donne par exemple :

```
1  SELECT prix_HT * 1.196 AS prix_TTC
2  FROM produits;
```

II. Restriction.

La syntaxe générale pour les restrictions est différente car elle nécessite l'ajout d'une ligne de code supplémentaire définie par la fonction WHERE.

La syntaxe ressemblera donc à ceci :

```
1  SELECT *
2  FROM table
3  WHERE condition;
```

La condition peut s'écrire avec :

- Des opérateurs de comparaison et autres :

  =, <>, <, >, <=, >=, LIKE, IN, NULL, IS NOT NULL.

- Des opérateurs logiques :

  AND, OR, NOT, XOR.

Pour faire des conditions incluant du texte il faut encadrer le texte par des « ' ».

Il existe un autre type de condition qui est la condition BETWEEN (entre x et y).

La syntaxe d'utilisation est la suivante :

```
1  SELECT *
2  FROM table
3  WHERE condition
4  BETWEEN valeur1 AND valeur2;
```

Pour réaliser une sélection dans une liste on utilise l'opérateur IN, la syntaxe générale est la suivante :

```
1  SELECT *
2  FROM table
3  WHERE attribut IN (val1, val2, val3, ...);
```

Pour réaliser une sélection approximative on utilise l'opérateur LIKE, la syntaxe générale est la suivante :

```
1  SELECT *
2  FROM table
3  WHERE attribut LIKE 'chaîne";
```

Remarque :

- % : remplace n'improte quelle chaîne (même vide) (*).
- _ : remplace n'importe quel caractère (?).

Pour réaliser une sélection sur un élément non évalué (vide), on a recours la syntaxe suivante :

```
1  SELECT *
2  FROM table
3  WHERE attribut IS NULL;
```

Remarque : Il est possible de faire l'inverse, sélectionner toutes les lignes où l'attribut sélectionné a été évalué, on utilise donc à la place de l'opérateur IS NULL, l'opérateur :
IS NOT NULL.

III. Tri.

Il est possible en plus de l'affichage d'effectuer un tri pour l'affichage des valeurs, par exemple on peut afficher une liste de nom et trier l'affichage pour afficher les noms dans l'ordre alphabétique.

La syntaxe générale est la suivante :

```
1  SELECT *
2  FROM table
3  ORDER BY attribut
```

Remarque importante : l'attribut de tri doit obligatoirement figurer dans la ligne du SELECT.

Pour trier de façon « croissante » on utilise la valeur ASC mais celle-ci est facultative.

Pour trier de façon « décroissante » on utilise la valeur DESC.

Petit récapitulatif général pour les 3 parties vus :

```
1  SELECT [DISTINCT] <champ1>,<champ2>,<champX> | *
2  FROM <table1>
3  [WHERE <condition-de-sélection>]
4  [AND | OR <condition-de-sélection>]
5  <ORDER BY <champ-à-trier> ASC | DESC]
```

Légende :

- < > : Éléments à remplacer.
- [ ] : Éléments optionnels.
- | : ou (soit l'un soit l'autre).

IV. Fonctions d'agrégation.

Il est possible en SQL de compter le nombre de lignes où un attribut est non nul (vide).

La syntaxe générale est la suivante :

```
1  SELECT COUNT (attribut)
2  FROM <table>
3  [WHERE <condition>]
```

Pour compter le nombre de lignes renvoyées par une requête, la syntaxe est la suivante :

```
1  SELECT COUNT (*)
2  FROM <table>
3  WHERE <condition>;
```

Ce qui donne par exemple :

```
1  SELECT COUNT (*) AS nbCommandes
2  FROM Commandes;
```

La syntaxe pour compter un nombre de valeurs différentes et non nulles d'un attribut est la suivante :

```
1  SELECT COUNT (DISCTINCT <attribut>)
2  FROM <table>
3  [WHERE <condition>];
```

Différentes fonctions permettent par exemple parmi les valeurs sélectionnées de relever uniquement la valeur la plus basse ou alors la plus grande.

Fonction MIN : donne la valeur minimale de la colonne <attribut> des lignes qui satisfont la condition. Syntaxe :

```
1  SELECT MIN (<attribut>)
2  FROM table
3  WHERE <condition>;
```

Fonction MAX : Donne la valeur maximale de la colonne <attribut> des lignes qui satisfont la <condition>. Syntaxe :

```
1  SELECT MAX (<attribut>)
2  FROM table
3  WHERE <condition>;
```

Fonction SUM : Donne la somme des valeurs contenues dans <attribut> qui satisfont la <condition>. Syntaxe :

```
1  SELECT SUM (<attribut>)
2  FROM table
3  WHERE <condition>;
```

Fonction AVG : Donne la moyenne des valeurs contenues dans <attribut> qui satisfont la <condition>. Syntaxe :

```
1  SELECT AVG (<attribut>)
2  FROM table
3  WHERE <condition>;
```

V. Produit cartésien.

Cela consiste à afficher chaque ligne d'une table à côté de chaque ligne d'une autre table. La syntaxe générale est la suivante :

```
1  SELECT <attribut1>, <attribut2>
2  FROM <table1>, <table2>;
```

VI. Sur nommage des tables.

De la même façon que l'on peut donner des alias aux colonnes, ce qui a une influence sur l'affichage, il est possible de surnommer les tables dans la requête pour que celle-ci soit plus aisée à rédiger (il n'y a pas d'influence sur l'affichage, si sur la table physique).

La syntaxe est la suivante :

```
1  SELECT <T1.attribut1>, <T2.attribut2>
2  FROM table1 T1, table2 T2;
```

VII. Préfixer les noms de colonnes.

- On peut préfixer tout attribut par le nom de sa table.
- On doit préfixer un attribut lorsqu'il peut y avoir ambiguïté.

La syntaxe générale est la suivante :

```
1  SELECT <table1.attribut1>, <table1.attribut2>, <table2.attribut1>
2  FROM table1, table2;
```

VIII. La jointure.

Il y a différentes syntaxe pour effectuer une jointure ou plusieurs :

Syntaxe 1 :

```
1  SELECT * | <liste d'attributs>
2  FROM table1
3  INNER JOIN table2 ON attribut1=attribut2;
```

Syntaxe 2 :

```
1  SELECT * | <surnomTable.liste d'attributs>
2  FROM Table1 T1
3  INNER JOIN Table2 T2 ON <condition>;
```

Syntaxe 3 :

```
1  SELECT * | <liste d'attributs>
2  FROM Table1
3  INNER JOIN Table2 ON <condition>
4  INNER JOIN Table3 ON <condition>
5  [WHERE <condition>];
```

IX. Regroupement.

Permet de regrouper des valeurs sur une valeur définie.

Par exemple : Dans un tableau affichant des numéros de commandes avec les quantités associées, il est possible d'afficher une seule ligne pour chaque numéro de commande et afficher la somme totale des quantités pour chaque numéro de commande.

La syntaxe générale pour le regroupement est la suivante :

```
1  SELECT agrégation 1
2  FROM table
3  [INNER JOIN ...]
4  [WHERE ...]
5  [GROUP BY agrégation 2
6  [HAVING condition]]
7  [ORDER BY ...];
```

Remarque : La fonction de condition WHERE se place avant la fonction GROUP BY, puis la fonction HAVING se place après la fonction GROUP BY, la fonction de condition HAVING est propre à la fonction GROUP BY.

X. Imbrication.

Utilité :

- Réaliser l'intersection entre deux ensembles de tuples.
- Réaliser la différence entre deux ensembles de tuples.
- Comparer une colonne avec le résultat d'une fonction agrégative portant sur cette même colonne.

Syntaxe générale en guise d'exemple :

```
1  SELECT <attributs> | * | fonction agrégation
2  FROM table
3  WHERE <condition sur une colonne> OPERATEUR (
4                                        SELECT colonne
5                                        FROM ...);
```