Le Langage S.Q.L. (Structured Query Language)

1ère Partie : Les requêtes de sélection

1/ Introduction

Les requêtes de sélection correspondent à des interrogations sur une base de données afin d'en extraire des informations.

Le langage DQL (Data Query Language) propose à cet effet une commande essentielle dans SQL, l'instruction *SELECT*.

L'instruction *SELECT* est associée à diverses clauses obligatoires ou optionnelles en vue d'affiner la sélection de données.

Les clauses principales de l'instruction *SELECT* permettent de désigner une table (*FROM*), d'appliquer des critères de sélection (*WHERE*), de regrouper des informations (*GROUP BY*) selon d'éventuelles conditions (*HAVING*), d'ordonner ces mêmes informations (*ORDER BY*) dans un ordre descendant (*DSC*) ou ascendant (*ASC*) ou encore de joindre plusieurs tables (*JOIN*).

2/ Les commandes SELECT et FROM

Les commandes *SELECT* et *FROM* sont utilisées pour sélectionner des tables dans une base de données.

```
SELECT * | { [ALL | DISTINCT] nom_colonne,...,nom_colonneN }
FROM nom_table
WHERE Condition
```

Pour **sélectionner tous les champs d'une table**, il suffit d'**utiliser une étoile** (*) à la place des noms de champs.
Il est possible de **sélectionner de un à plusieurs champs en séparant les noms de champs ou de colonnes par une virgule**.
Les clauses *ALL* par défaut et *DISTINCT* permettent respectivement de sélectionner tous les enregistrements d'une colonne et de ne prendre en compte chaque enregistrement distinct, soit sans doublons.
La clause *FROM* cible une ou plusieurs tables à partir desquelles l'extraction des données doit être opérée.

```
SELECT nom_table.nom_colonne,..., nom_tableN.nom_colonneN
FROM nom_table,..., nom_tableN
```

La clause *WHERE* pose une condition dans la sélection des informations.

```
SELECT * FROM employe WHERE nom = 'DUPONT'
```

Les requêtes de sélection peuvent également être utilisées comme sous-requêtes dans une clause conditionnelle *WHERE*.

```
SELECT tab1.nom_champ,..., tab1.nom_champN
FROM nom_table AS tab1
WHERE tab1.nom_champ = (SELECT tab2.nom_champ
                        FROM nom_table AS tab2
                        WHERE tab2.nom_champ = valeur)
```

Les sous-requêtes permettent de sélectionner un premier jeu d'enregistrements dans une table tierce qui servira de condition de sélection dans la requête principale.

Exemple

Librairie			
ISBN	**Livre**	**Editeur**	**Prix**
1565926978	ORACLE SQL : THE ESSENTIAL REFERENCE (EN ANGLAIS)	O'REILLY	330.00
1565929489	ORACLE SQL* LOADER : THE DEFINITIVE GUIDE (EN ANGLAIS)	O'REILLY	286.00
2212092857	INITIATION A SQL - COURS ET EXERCICES CORRIGES	EYROLLES	183.00
2744009296	SQL	CAMPUSPRESS FRANCE	62.00
2744090034	MAITRISEZ SQL	WROX PRESS	286.00
2840725029	SQL	ENI	140.00
2840828987	KIT DE FORMATION MICROSOFT SQL SERVER 2000 ADMINISTRATION SYSTEME	MICROSOFT PRESS	284.00

```
SELECT Livre FROM Librairie

' retourne
ORACLE SQL : THE ESSENTIAL REFERENCE (EN ANGLAIS)
ORACLE SQL* LOADER : THE DEFINITIVE GUIDE (EN ANGLAIS)
INITIATION A SQL - COURS ET EXERCICES CORRIGES
SQL
MAITRISEZ SQL
SQL
KIT DE FORMATION MICROSOFT SQL
      SERVER 2000 ADMINISTRATION SYSTEME
```

3/ Les commandes DISTINCT et ALL

La commande *DISTINCT* sélectionne chaque élément distinct d'une colonne de données.

La commande *ALL* indique que les doublons peuvent apparaître dans le résultat d'une requête, elle constitue une commande par défaut de *SELECT*.

```
SELECT DISTINCT | ALL nom_champ FROM nom_table
```

Exemple

```
SELECT DISTINCT Livre FROM Librairie

' retourne
ORACLE SQL : THE ESSENTIAL REFERENCE (EN ANGLAIS)
ORACLE SQL* LOADER : THE DEFINITIVE GUIDE (EN ANGLAIS)
INITIATION A SQL - COURS ET EXERCICES CORRIGES
SQL
MAITRISEZ SQL
KIT DE FORMATION MICROSOFT SQL
      SERVER 2000 ADMINISTRATION SYSTEME


SELECT ALL Livre FROM Librairie

' retourne
ORACLE SQL : THE ESSENTIAL REFERENCE (EN ANGLAIS)
ORACLE SQL* LOADER : THE DEFINITIVE GUIDE (EN ANGLAIS)
INITIATION A SQL - COURS ET EXERCICES CORRIGES
SQL
SQL
MAITRISEZ SQL
```

4/ Les commandes d'alias

Les alias concourent à améliorer la lisibilité d'une requête. Il existe deux types d'alias : les alias de tables et les alias de champs. Les alias peuvent également s'appliquer à une fonction d'agrégation retournant des données sous forme de colonnes.

```
SELECT Alias_table.nom_champ AS Alias_champ
FROM nom_table AS Alias_table
```

La clause *AS* affectant un alias à une table ou une colonne, peut être **remplacer par un simple espace blanc**.

```
SELECT Alias_table.nom_champ Alias_champ
FROM nom_table Alias_table
```

Le langage SQL prévoit également les synonymes de tables, de vues ou de tout autre objet. Les synonymes permettent de dissimuler un nom de table par un autre afin d'éviter qu'un utilisateur accède nommément à un objet qui ne lui appartient guère.

```
CREATE [ PUBLIC ] SYNONYM nom_synonyme FOR objet
```

Les synonymes peuvent être publics (*PUBLIC*) donc accessibles dans toute la base de données, et partant, par l'ensemble de ses utilisateurs **ou privés** donc destinés au seul usage de son créateur et des éventuels bénéficiaires de privilèges d'accès.

Exemple

```
SELECT L.Editeur Edi,
SUM(L.Prix) F
FROM Librairie AS L
GROUP BY Edi
HAVING F > 200

' retourne
```

O'REILLY	616.00
O'REILLY	616.00
WROX PRESS	286.00
MICROSOFT PRESS	284.00

5/ La commande WHERE

La commande *WHERE* pose une condition dans la requête de sélection.

```
SELECT nom_champ
FROM nom_table
WHERE condition
```

L'**expression conditionnelle** peut être construite à l'aide de **plusieurs prédicats** constitués d'**opérateurs de comparaisons** (= | <> | != | > | > = | !> | < | <= | !<) et séparées par des **opérateurs booléens** *AND*, *OR* ou *NOT*.

```
SELECT nom_champ
FROM nom_table
WHERE champ >= valeur AND
                ( champ2 = valeur2 OR champ3 != valeur3)
```

La condition consiste à qualifier des enregistrements en fonction de son résultat booléen.

- **Si la valeur est *True***, l'information est sélectionnée.
- **Si la valeur est *False***, l'information n'est pas prise en compte.

La clause *WHERE* peut accueillir **une sous-requête de sélection limitant ainsi les enregistrements sélectionnables** dans la requête principale.

Exemple

```
SELECT Livre FROM Librairie WHERE Prix < 200

' retourne
INITIATION A SQL - COURS ET EXERCICES CORRIGES
SQL
SQL
```

Remarques :

- Les chaînes de caractères sont mentionnées entre guillemets (‘’) ou entre cottes (‘), les dates entre dièses (#) et les valeurs numériques directement sans aucun signe particulier.
- L’instruction **BETWEEN** permet de tester des plages de valeurs (souvent utilisée pour les dates), exemple : WHERE date_commande BETWEEN #01-01-2001# AND #07-01-2001#

6/ La clause HAVING

La clause *HAVING* spécifie un critère de recherche pour un groupe ou une fonction d'agrégation. *HAVING* est généralement utilisé avec *GROUP BY*, sinon *HAVING* se comporte à l'instar de *WHERE*.

```
SELECT nom_champ,
Fonction_Agregation
FROM nom_table
GROUP BY nom_champ2
HAVING "Condition"
```

Exemple

```
SELECT Editeur,
SUM(Prix)
FROM Librairie
GROUP BY Editeur
HAVING SUM(Prix) > 200

' retourne
O'REILLY            616.00
WROX PRESS          286.00
MICROSOFT PRESS 284.00
```

7/ La commande GROUP BY

La commande *GROUP BY* spécifie des groupes dans lesquels les lignes de sortie doivent être placées et calcule une valeur de résumé pour chacun des groupes si des fonctions d'agrégation sont employées avec la commande *SELECT*.

```
SELECT nom_champ, Fonction_Agregation
FROM nom_table
GROUP BY nom_champ2
```

L**e critère de la clause *GROUP BY*** peut être soit **le nom**, soit **le numéro** d'une des colonnes sélectionnées par la requête.

Exemple

```
SELECT Editeur, SUM(Prix)
FROM Librairie
GROUP BY Editeur

' retourne
O'REILLY            616.00
```

EYROLLES	183.00
CAMPUSPRESS FRANCE	62.00
WROX PRESS	286.00
ENI	140.00
MICROSOFT PRESS	284.00

8/ La commande ORDER BY

La commande *ORDER BY* permet de classer alphabétiquement (ou trier numériquement ou chronologiquement) les données retournées par la requête de sélection.

```
SELECT nom_champ, nom_champ2,..., nom_champN
FROM nom_table
ORDER BY nom_champ [ ASC | DSC { nom_champ2 ASC | DSC } ]
```

Le critère de la clause *ORDER BY* peut être soit **le nom**, soit **le numéro** d'une des colonnes sélectionnées par la requête.
La clause *ASC* par défaut ou *DSC* provoque respectivement un tri dans un ordre ascendant (0 ... 9 et A ... Z) ou descendant (Z ... A et 9 ... 0).

Exemple

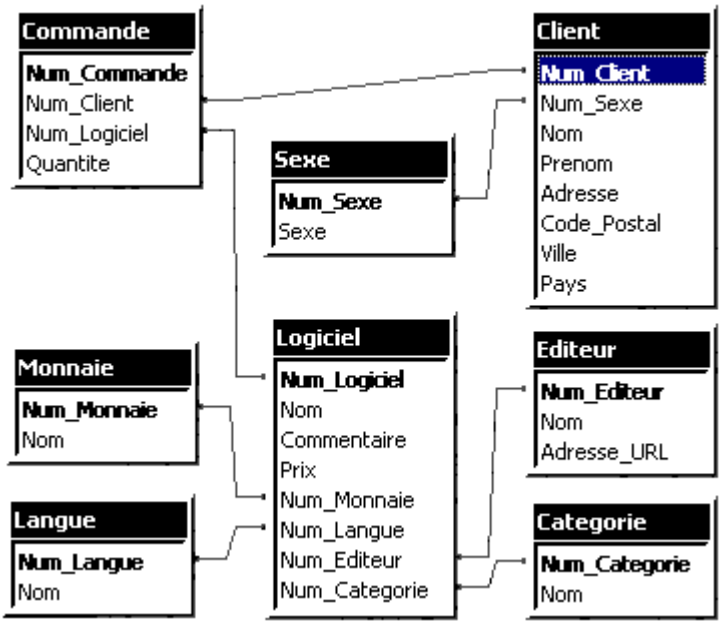
```
SELECT Editeur,
SUM(Prix)
FROM Librairie
ORDER BY Editeur

' retourne
```

CAMPUSPRESS FRANCE	62.00
ENI	140.00
EYROLLES	183.00
MICROSOFT PRESS	284.00
O'REILLY	616.00
WROX PRESS	286.00

9/ Les commandes de jointures de table

les requêtes de sélection demeurent les plus intéressantes, lorsqu'elles s'appliquent à plusieurs tables. Mais pour cela, il est nécessaire d'exécuter des commandes de jointure afin de pouvoir extraire des données de ces tables liées.

Il existe de nombreuses manières de joindre des tables dépendant souvent de l'implémentation utilisée. Néanmoins, la plupart des commandes de jointures accomplissent leurs opérations sur des **colonnes communes aux différentes tables**. Ainsi, les jointures sont possible uniquement sur des tables possédant des colonnes correspondantes comme une clé étrangère se relierait à une clé primaire.



Des tables mises en relation les unes avec les autres, peuvent facilement délivrer des informations recoupées. Si aucune instruction de jointure n'est fournie dans une requête de sélection, alors cette dernière renverra la totalité des lignes présentes dans les tables concernées.

Une jointure d'égalité symbolisée par *EQUIJOIN* ou *INNER JOIN* crée une relation de correspondance entre des tables.

```
SELECT l.Nom, e.Nom, c.Nom
FROM Logiciel AS l, Editeur AS e, Categorie AS c
WHERE l.Num_Editeur = e.Num_Editeur,
      AND l.Num_Categorie = c.Num_Categorie
```

Ou :

```
SELECT l.Nom, e.Nom, c.Nom
FROM Logiciel AS l
INNER JOIN
ON Editeur AS e, Categorie AS c
```

Autre exemple :

```
SELECT cl.Nom, cl.Prenom, l.Nom,
          (l.Prix * cmd.Quantite) AS Montant, m.Nom
FROM Logiciel l, Client cl, Commande cmd, Monnaie m
WHERE cmd.Num_Client = cl.Num_Client
      AND l.Num_Logiciel = cmd.Num_Logiciel
      AND l.Num_Monnaie = m.Num_Monnaie
```

Une jointure de non-égalité *NON-EQUIJOIN* permet de sélectionner des enregistrements dont les valeurs de colonnes ne correspondent pas.

```
SELECT s.Sexe, cl.Nom, cl.Prenom
FROM Logiciel AS l, Commande AS cmd, Client AS cl
WHERE cl.Num_Client != cmd.Num_Client,
      AND cl.Num_Sexe != s.Num_Sexe
```

Les jointures externes *OUTER JOIN* sont employées pour retourner tous les enregistrements, y compris ceux ne possédant aucune correspondance. Ces jointures peuvent s'appliquer à gauche (*LEFT*), à droite (*RIGHT*) ou sur les deux (*FULL*).

```
SELECT cl.Nom, cl.Prenom, cmd.Num_Commande
FROM Client AS cl, Commande AS c
WHERE cl.Num_Client (+) = cmd.Num_Commande
```

Ou :

```
SELECT cl.Nom, cl.Prenom, cmd.Num_Commande
FROM Client AS cl
```

```
LEFT OUTER JOIN
ON Commande AS c
```

Les jointures réflexives sont utilisées pour joindre une table à elle-même.

```
SELECT cl1.Nom, cl1.Prenom, cl2.Nom, cl2.Prenom
FROM Client AS cl1, Client AS cl2
WHERE cl1.Num_Client = cl2.Num_Client
```

Ou :

```
SELECT cl1.Nom, cl1.Prenom, cl2.Nom, cl2.Prenom
FROM Client AS cl1
SELF JOIN
ON Client AS cl2
```

10/ Les fonctions d'agrégation

Les fonctions d'agrégation accomplissent un calcul sur plusieurs valeurs et retournent un résultat et sont souvent utilisées avec les commandes *GROUP BY* et *SELECT*.

Les calculs effectués par ces fonctions consiste à faire sur une colonne, la somme, la moyenne des valeurs, le décompte des enregistrements ou encore l'extraction de la valeur minimum ou maximum.

Attention : Hormis la fonction *COUNT*, **les fonctions d'agrégation ne tiennent pas compte des valeurs *NULL***.

Fonction	Description
AVG	retourne la moyenne des valeurs d'un groupe.
BINARY_CHECKSUM	retourne la valeur totale de contrôle binaire calculée à partir d'une ligne d'une table ou d'une liste d'expressions.
CHECKSUM	retourne la valeur de *checksum* calculée dans une ligne d'une table ou dans une liste d'expressions.
CHECKSUM_AGG	retourne le *checksum* des valeurs d'un groupe.
COUNT	retourne le nombre d'éléments figurant dans un groupe.
COUNT_BIG	retourne le nombre d'éléments figurant dans un groupe.
MAX	retourne la valeur maximale de l'expression.
MIN	retourne la valeur minimale de l'expression.
SUM	retourne la somme de toutes les valeurs
STDEV	retourne l'écart type de toutes les valeurs de l'expression spécifiée.
STDEVP	retourne l'écart type de remplissage pour toutes les valeurs de l'expression spécifiée.
VAR	retourne la variance de toutes les valeurs de l'expression spécifiée.
VARP	retourne la variance de remplissage pour toutes les valeurs de l'expression spécifiée.

Exemple

```
SELECT COUNT(Livre)
FROM Librairie

' retourne
7

SELECT SUM(Prix)
FROM Librairie

' retourne
1 571
```