L'ORDRE SELECT

Table des matières

I. SYNTAXE

Cette commande permet la RECHERCHE d'informations en sélectionnant les données selon divers critères.

```
SELECT <NOMS DE COLONNES OU EXPRESSIONS> FROM
        <NOMS DE TABLES>
WHERE <CONDITIONS DE RECHERCHE>
GROUP BY <NOMS DE COLONNE DU SELECT> HAVING
        <CONDITION DE RECHERCHE> ORDER BY <NOM OU
POSITION DE COLONNE
DANS L'ORDRE SELECT>
```

Le résultat de la clause SELECT est une table, sous-ensemble de la (ou des) table(s) de départ :

- dont les colonnes dépendent des rubriques citées après SELECT (colonnes directement issues de la table d'origine, valeurs calculées, constantes, etc)
- dont les lignes satisfont tant par leur contenu que pour leur présentation, aux options suivant; le cas échéant, le nom de la table.

EXEMPLE : Lister le contenu de la table EMPLOYES

```
SELECT * FROM EMPLOYES
```

NOEMP	PRENOM	NOM	WDEPT	SALAIRE
00010	PAUL	BALZAC	A00	1200
00020	ANNE	MARTIN	B00	1500
00030	ANDRE	BOULIN	C01	2100
00040	PIERRE	DURAND	E01	1530
00050	MARIE	VIALA	A01	1650

L'astérisque (*) inséré dans la liste de sélection permet d'extraire toutes les colonnes d'une table.

II. SELECT AVEC EXPRESSION

Une ou plusieurs expressions peuvent suivre le mot-clé SELECT. Une expression spécifie une valeur. Elle peut être :

- Un nom de colonne
- Une constante (numérique ou chaîne de caractères)
- Une fonction (CURRENT_TIMESTAMP, GETDATE() ...)
- Une combinaison de ces valeurs séparées :
  - par l'opérateur de concaténation + (chaînes de caractères)
  - par l'un des opérateurs arithmétiques binaires (*, /, +, - évalués dans cet ordre) ou unaires (+, -) (numériques).

  Si l'un des opérandes a la valeur null, le résultat de l'expression est null.
- Une suite d'expressions séparées par des parenthèses

EXEMPLE : Lister le nom et le salaire en centimes de chaque employé.

La lisibilité des noms de colonne peut être améliorée en utilisant le mot clé AS : les noms de colonne par défaut seront remplacés par des **alias** dans la liste de sélection**.**

```
SELECT NOM + ' ' + PRENOM AS 'NOM DU SALARIE', SALAIRE * 100 AS
SALAIRE,'CENTIMES '
FROM EMPLOYES
```

NOM DU SALARIE	SALAIRE	
BALZAC PAUL	1200000	CENTIMES
MARTIN ANNE	1500000	CENTIMES
BOULIN ANDRE	2100000	CENTIMES
DURAND PIERRE	1530000	CENTIMES

III. SELECT OPTION DISTINCT

```
SELECT [ALL] nomcol1 FROM nomtable
```

par opposition à

```
SELECT DISTINCT nomcol1 FROM nomtable
```

L'option ALL est prise par défaut, toutes les lignes sélectées figurent dans le résultat.

L'option DISTINCT permet de ne conserver qu'un exemplaire de chaque ligne en double.

SELECT **WDEPT** FROM EMPLOYES

```
WDEPT
----------
A00
B00
C01
E01
A01
E01
C01
B00
```

SELECT **DISTINCT WDEPT** FROM EMPLOYES

```
WDEPT
----------
A00
B00
C01
E01
A01
```

IV. SELECT CONDITIONNELLE AVEC WHERE

La clause **WHERE** permet de préciser les conditions de recherche sur les lignes de la table.

Les conditions peuvent contenir une liste illimitée de prédicats – expressions renvoyant la valeur TRUE (vrai), FALSE (faux) ou UNKNOWN (inconnu) -, combinés à l'aide des opérateurs logiques **AND** ou **OR** .

Pour spécifier la condition de recherche dans la clause WHERE, on utilise indifféremment l'un des opérateurs conditionnels ci-après :

Description	Opérateurs conditionnels
Opérateurs de comparaison	**=, <>, !=, >, >=, !>, <, <=, !<**
Comparaisons de plage	**BETWEEN** et **NOT BETWEEN**
Comparaisons de listes	**IN** et **NOT IN**
Comparaisons de chaîne de caractères	**LIKE** et **NOT LIKE**
Valeurs inconnues	**IS NULL** et **IS NOT NULL**

Quelques exemples :

Rechercher dans la table EMPLOYES, les données concernant les employés qui travaillent dans le départements A00 ou E01

```
SELECT *
FROM EMPLOYES
WHERE WDEPT = 'A00' OR WDEPT = 'E01'
```

NOEMP	PRENOM	NOM	WDEPT	SALAIRE
00010	PAUL	BALZAC	A00	1200
00040	PIERRE	DURAND	E01	1530
00120	JULIEN	DELMAS	E01	1834

Lister le prénom et le nom des employés dont le salaire est compris entre 1200 et 1600

```
SELECT PRENOM, NOM
FROM EMPLOYES
WHERE SALAIRE BETWEEN 1200 AND 1600
```

PRENOM	NOM	SALAIRE
PAUL	BALZAC	1200
ANNE	MARTIN	1500
PIERRE	DURAND	1530
ARTHUR	ZOLA	1600

```
SELECT *
FROM EMPLOYES
WHERE NOEMP IN ('00010', '00020', '00050'' '00100')
```

L'opérateur LIKE de la clause WHERE conjointement aux caractères génériques, permet de comparer des chaînes de caractères inexactes.

Employé dont le nom commence par B

```
SELECT *
FROM EMPLOYES
WHERE NOM LIKE 'B%'
```

Caractères génériques	Description
%	N'importe quelle chaîne comprise entre zéro et plusieurs caractères
-(trait de soulignement)	N'importe quel caractère unique
[ ]	N'importe quel caractère unique dans la plage (par exemple [s-w] ou [aeiouy])
[^]	N'importe quel caractère unique n'appartenant pas à la plage (par exemple [^s-w] ou [^aeiouy])

LIKE 'BAL%' tous les noms commençant par les lettres BAL
LIKE '%BAL%' tous les noms contenant les lettres BAL
LIKE '--LZ--' tous les noms de 6 caractères contenant LZ en 3ème et 4ème positions
LIKE '[S-V]ENT' tous les noms de 4 lettres se terminant par les lettres ENT et commençant par n'importe quelle lettre comprise entre S et V.

V. LES FONCTIONS

SQL offre la possibilité d'intégrer dans les expressions des fonctions retournant :

- Une valeur dépendant du contenu de colonnes (fonction sur les colonnes)
- Une valeur dépendant d'opérandes (fonctions scalaires)

Exemple : Lister le salaire maximum du département E01

```
SELECT MAX (SALAIRE)
FROM EMPLOYES
WHERE WDEPT = 'E01'
```

Lorsqu'une fonction d'agrégation est exécutée, SQL effectue la synthèse des valeurs d'une colonne spécifique pour la table complète ou pour des groupes de lignes de la table (clause GROUP BY) : une valeur d'agrégation unique est alors générée pour la table complète ou pour chaque groupe de lignes.

```
SELECT AVG(SALAIRE)
FROM EMPLOYES
```

```
SELECT COUNT (*)
FROM EMPLOYES
```

VI. LES FONCTIONS DU SELECT

A. La clause ORDER BY

La clause ORDER BY permet de préciser une séquence de tri pour le résultat d'une requête.

•••• ASC séquence croissante (valeur par défaut)

•••• DESC séquence décroissante

```
SELECT *
FROM EMPLOYES
ORDER BY WDEPT ASC, NOM DESC
```

La clause ORDER BY doit être la dernière clause dans l'ordre SELECT

B. La clause GROUP BY

Ces options permettent de définir et de traiter des groupes.

Un groupe est formé à partir d'un ensemble de lignes d'une table ayant une ou plusieurs caractéristiques communes.

L'intérêt d'un groupe est de conserver la trace des éléments qu'il contient, par exemple pour les dénombrer ou effectuer des opérations telles que somme ou moyenne.

```
SELECT WDEPT, AVG (SALAIRE), MIN (SALAIRE)
FROM EMPLOYES
WHERE NOEMP > 00010
GROUP BY WDEPT
```

Il y a autant de groupes que de valeurs de WDEPT distinctes.

GROUP BY est suivi du nom d'une ou plusieurs colonnes présentes dans le SELECT appelées colonnes de regroupement.

La liste de colonnes suivant SELECT ne peut comporter que les noms des colonnes de regroupement, ou des noms de fonctions.

C. La clause HAVING

La clause HAVING est utilisée en conjonction avec la clause GROUP BY.

La clause HAVING agit comme critère de sélection pour les groupes renvoyés avec la clause GROUP BY.

Exemple : Quel est le salaire moyen et le salaire minimum des employés à l'intérieur de chaque département pour les n°employés > 00010 ?

Lister uniquement les groupes pour lesquels la moyenne est supérieure à 16 000

```
SELECT WDEPT, AVG (SALAIRE), MIN (SALAIRE)
FROM EMPLOYES
WHERE NOEMP > 00010
GROUP BY WDEPT
HAVING AVG (SALAIRE) >= 16000
```

La condition de recherche suivant HAVING ne peut porter que sur des colonnes de regroupement définies par la clause GROUP BY, ou sur des fonctions.

Il ne faut pas confondre les clauses WHERE et HAVING.

- WHERE permet de sélectionner des lignes avant la formation des groupes.
- HAVING permet de ne retenir que certains des groupes constitués par la clause GROUP BY.

D. L'opération de jointure

Une jointure est une opération qui permet d'interroger plusieurs tables pour obtenir un jeu de résultats unique intégrant des lignes et des colonnes de chaque table.

La plupart des conditions de jointure sont basées sur la clé primaire d'une table et la clé étrangère d'une autre table. Quand la jointure se fait sur des tables ayant des noms de colonne identiques, les noms en double doivent être préfixés par leur nom de table ou un nom associé.

Il existe trois types de jointure : les jointures internes, les jointures externes et les jointures croisées.

```
SELECT nom_colonne1, nom_colonne2, ... nom_colonne n
FROM nom_table1
[INNER | {LEFT | RIGHT | FULL} [OUTER]} JOIN
nom_table2 ON conditions _recherche
```

• Le mot clé JOIN et ses options spécifient les tables à joindre et la manière de les joindre.

• Le mot clé ON spécifie les colonnes communes aux tables.

La plupart des conditions de jointure sont basées sur la clé primaire d'une table et la clé étrangère d'une autre table.

```
SELECT NOM, PRENOM, SALAIRE, NOMDEPT
FROM EMPLOYES
INNER JOIN DEPART
ON WDEPT = NODEPT
```

Les informations extraites font parties de la table Employes et de la table Depart

Pour être clair, on préférera écrire :

```
SELECT E.NOM, E.PRENOM, E.SALAIRE, D.NOMDEPT
FROM EMPLOYES as E
INNER JOIN DEPART as D
ON E.WDEPT = D.NODEPT
```

La jointure interne renvoie uniquement les lignes en correspondance.

E. Utilisation des jointures externes gauche ou droite

• La jointure externe gauche permet d'afficher toutes les lignes de la première table nommée (table située à gauche de la clause JOIN).

• La jointure externe droite permet d'afficher toutes les lignes de la seconde table nommée (table située à droite de la clause JOIN).

Exemple : Liste des employés avec le nom du département dans lequel ils sont affectés.

```
SELECT E.NOM, E.PRENOM, E.SALAIRE, D.NOMDEPT
FROM EMPLOYES as E
LEFT JOIN DEPART as D
ON E.WDEPT = D.NODEPT
```

La jointure externe renvoie toutes les lignes de la table EMPLOYES. La colonne NOMDEPT contient la valeur NULL pour les départements n'existant pas dans la table DEPART.

F. Jointure de plus de deux tables

```
SELECT colonne_1, colonne_2, ..., colonne_n
FROM Table_1
JOIN Table_2
ON Table_1.colonne_i = Table_2.colonne_j
JOIN Table_3
ON Table_2.colonne_x = Table_3.colonne_y
```

G. Auto-jointure

Exemple : Liste des employés ayant le même prénom

```
SELECT A.NOM, NODEPT FROM EMPLOYES as A
JOIN EMPLOYES as B
ON A.PRENOM = B.PRENOM
WHERE A.NOM <> B.NOM;
```