

Initiation à SQL : Introduction

Le SQL (Structured Query Language) permet d'interroger une base de données, d'en modifier des informations. C'est un langage universel d'interrogation des bases de données, qui permet à différents systèmes d'échanger des données entre eux.

ACCESS peut être interrogé en SQL via un macro langage qui « cache » le SQL (voir à ce sujet le Menu « Requêtes », « Spécifique SQL » « SQL direct »).

Normalisation ISO :

- ❑ norme SQL1 (1986, 1989)
- ❑ norme SQL2 (1992)
- ❑ nouvelle norme en préparation SQL3

Dans la suite, le code SQL est représenté en **ARIAL 12**, les commentaires en *Italique*.

1 Les requêtes simples

Soit 3 tables :

- ❑ **Eleves** (NomElv, AdrElv, VilleElv),
- ❑ **Matieres** (NomMat, Coef, Intitule),
- ❑ **Notes** (NomElv, NomMat, Date, Note).

Attribut souligné = *clé primaire*

1.1 L'interrogation simple

- ❑ *Liste des élèves.*

```
SELECT NomElv      (ce qui doit être affiché ?)
FROM Eleves;       (dans quelle table rechercher l'information ?)
```

- ❑ *Liste des matières avec leurs coefficients.*

```
SELECT NomMat, Coef
FROM Matieres;
```

1.2 La close WHERE

Elle permet de spécifier la ou les conditions que doivent remplir les lignes choisies.

- ❑ *Liste des élèves habitant Toulon.*

```
SELECT NomElv
FROM Eleves
WHERE VilleElv = 'Toulon';
```

- ❑ *Liste des matières pour lesquelles l'élève "Denis" a eu au moins une note.*

```
SELECT NomMat
FROM Matiere
WHERE NomElv = 'Denis';
```

Remarque : Dans la close WHERE, on ne peut utiliser que des propriétés qui sont dans la table sélectionnée

1.3 La close GROUP BY

- ❑ *Liste des élèves par ville.*

```
SELECT NomElv, VilleElv
FROM Eleves
GROUP BY VilleElv;
```

1.4 La close HAVING

Elle ne s'utilise qu'avec le GROUP BY et permet de donner la ou les conditions que doivent remplir ces groupes.

- ❑ *Liste des élèves regroupés par ville où habitent plus de 10 élèves.*

```
SELECT NomElv, VilleElv
FROM Eleves
GROUP BY VilleElv
HAVING Count(NomElv) > 10;
```

- ❑ *Liste des matières où plus de 35 notes ont été données.*

```
SELECT NomMat
FROM Notes
GROUP BY NomMat
HAVING Count(Note) > 35;
```

1.5 La close ORDER BY

Elle permet de spécifier l'ordre dans lequel vont être affichées les lignes.

- ❑ *Liste des matières dans l'ordre alphabétique.*

```
SELECT NomMat
FROM Matieres
ORDER BY NomMat;
```

- ❑ *Liste des matières par ordre décroissant de coef., puis par ordre alpha. de nom.*

```
SELECT NomMat
FROM Matieres
ORDER BY Coef Desc, NomMat Asc;
```

1.6 Récapitulatif

SELECT	<i>noms des colonnes à afficher</i>
FROM	<i>nom de la table où se trouvent les colonnes susmentionnées</i>
WHERE	<i>condition(s) à remplir par les lignes</i>
GROUP BY	<i>condition(s) de regroupement des lignes</i>
HAVING	<i>condition(s) à remplir par le groupe</i>
ORDER BY	<i>ordre (Asc, Desc) d'affichage</i>

2 Les requêtes multi-tables

Soit 4 tables :

- ❑ Eleves (RefElv, NomElv, PreElv, VilleElv, ClasseElv),
- ❑ Classes (NomCla, Niveau),
- ❑ Cours (RefElv, NomMat, NbHeure),
- ❑ Matieres (NomMat).

2.1 Requêtes où les données sélectionnées sont dans plusieurs tables

- ❑ *Liste des élèves avec leur niveau.*

```
SELECT NomElv, PreElv, Niveau
FROM Eleves, Classes
WHERE Eleves.ClasseElv = Classes.NomCla;
```

- ❑ *Liste des élèves et nom des cours qu'ils suivent pendant plus de 3 heures.*

```
SELECT NomElv, NomMat
FROM Eleves, Cours
WHERE (Eleves.ClasseElv = Cours.Nomcla) AND (Cours.NbHeure > 3)
      (il faut faire d'abord les jointures puis les sélections)
```

2.2 Requêtes où les données proviennent d'une table mais où la condition de sélection est faite sur une table différente

- ❑ *Liste des élèves de 1ère.*

```
SELECT NomElv, PreElv
FROM Eleves
WHERE ClasseElv IN (SELECT NomCla FROM Classes WHERE Niveau =
'1ère');
```

- ❑ *Liste des élèves qui font de la Peinture pendant plus de 2 heures.*

```
SELECT NomElv, PreElv
FROM Eleves
WHERE RefElv IN (SELECT RefElv FROM Cours WHERE (NomMat =
'Peinture') AND (NbHeure > 2));
```

- ❑ *Liste des élèves habitant Toulon et suivant des cours de Macramé et de niveau BTS.*

```
SELECT NomElv, PreElv
FROM Eleves
WHERE (VilleElv = 'Toulon')
AND (RefElv IN (SELECT RefElv FROM Cours WHERE (NomMat = 'Macramé')))
AND (ClasseElv IN (SELECT NomCla FROM Classes WHERE (Niveau = 'BTS')));
```

2.3 Ce qu'il faut éviter

- ❑ *Liste des élèves de niveau Terminale.*

```
SELECT NomElv, PreElv
FROM Eleves, Classes
WHERE (Eleves.ClasseElv = Classes.NomCla) AND (Niveau = 'Terminale');
```

3 Les Jointures

3.1 La jointure LEFT OUTER JOIN

Soit 2 tables :

- ❑ **Clients** (NomCli, AdrCli, CPCLi, VilleCli, CodeCom),
- ❑ **Commerciaux** (CodeCom, NomCom).

On utilise **LEFT OUTER JOIN** pour créer une jointure externe gauche.

Prenons un cas simple avec deux tables : La jointure externe gauche se compose de tous les enregistrements de la

première table (celle de gauche) et de ceux la deuxième table (celle de droite) seulement à la condition qu'il existe une valeur correspondante aux enregistrements de la première table (celle de gauche).

- ❑ *Liste des Clients avec le nom de leurs commerciaux.*

```
SELECT NomCli, CPCli, VilleCli, NomCom
FROM Clients LEFT OUTER JOIN Commerciaux ON Clients.CodeCom =
Commerciaux.CodeCom;
```

Avec les tables suivantes :

NomCli	AdrCli	CPCli	VilleCli	CodeCom
SA J. Minet	5 av Paul Doumer	33000	Bordeaux	2
SARL Tony	3 rue de la Cloche	17000	La Rochelle	3

et

CodeCom	NomCom
1	Pierre Dubois
2	Paul Renaudin
3	Jacques William
4	Guillaume Roland

On obtient le résultat suivant :

NomCli	CPCli	VilleCli	NomCom
SA J. Minet	33000	Bordeaux	Paul Renaudin
SARL Tony	17000	La Rochelle	Jacques William

3.2 La jointure RIGHT OUTER JOIN

Soit 2 tables :

- ❑ Clients (NomCli, AdrCli, CPCli, VilleCli, CodeCom),
- ❑ Commerciaux (CodeCom, NomCom).

On utilise **RIGHT OUTER JOIN** pour créer une jointure externe droite.

Prenons un cas simple avec deux tables : La jointure externe droite se compose de tous les enregistrements de la seconde table (celle de droite) et de ceux la première table (celle de gauche) seulement à la condition qu'il existe une valeur correspondante aux enregistrements de la deuxième table (celle de droite).

- ❑ *Liste des Clients avec le nom de leurs commerciaux.*

```
SELECT NomCli, CPCli, VilleCli, NomCom
FROM Commerciaux RIGHT OUTER JOIN Clients ON Clients.CodeCom =
Commerciaux.CodeCom;
```

Fournit un résultat identique à la précédente

3.3 La jointure INNER JOIN

Soit 2 tables :

- ❑ Eleves (RefElv, NomElv, PreElv, ClasseElv),
- ❑ Classes (NomCla, Niveau).

On utilise **INNER JOIN** pour fusionner les enregistrements de deux tables lorsqu'un champ commun contient des valeurs identiques.

- ❑ *Liste des élèves avec leurs niveaux.*

```
SELECT RefElv, NomElv, PreElv, NomCla, Niveau
FROM Eleves INNER JOIN Classes ON Eleves.ClasseElv =
Classes.NomCla;
```

4 Création de tables

L'instruction CREATE TABLE permet de créer une table de nom « étudiant » avec les attributs « nom », « date-naiss », « numero », « nom-ent » avec un index sur le numero..

```
CREATE TABLE Etudiant (Nom char (20), date-naiss date, numero int,  
nom-ent char (40), CONSTRAINT index PRIMARY KEY (numero)) ;
```

5 Insertion, Modification et Suppression d'enregistrements dans des tables

Soit 1 table :

❑ Eleves (NomElv, AdrElv, VilleElv).

5.1 Insertion de données : INSERT INTO

Insérer l'élève "Tony" qui habite "av Paul Doumer" à "Toulon".

```
INSERT INTO Eleves (NomElv, AdrElv, VilleElv) VALUES ('Tony', 'av Paul  
Doumer', 'Toulon');
```

5.2 Mise à jour de données : UPDATE

Modifier le nom de l'élève "Tony" en "Antony".

```
UPDATE Eleves SET NomElv = 'Antony' (Nouvelle valeur)  
WHERE NomElv = 'Tony'; (Ancienne valeur)
```

5.3 Suppression de lignes de tables : DELETE

Supprimer l'élève "Tony".

```
DELETE FROM Eleves  
WHERE NomElv = 'Tony';
```

6 Les fonctions

Il existe cinq agrégats (fonctions prédéfinies) : Avg, Count, Sum, Min, Max.

Une fonction s'applique à l'ensemble des valeurs d'une colonne d'une table (sauf pour la fonction Count). elle produit une valeur unique.

6.1 AVG

La commande AVG permet de calculer la moyenne d'un champ

Soit une table

❑ Commandes (NumCmd, DateCmd, Désignation, FraisPort).

La moyenne des frais de port pour les commandes dont les frais sont inférieurs 150F.

```
SELECT AVG (FraisPort)  
FROM Commandes  
WHERE FraisPort < 150;
```

6.2 COUNT

La commande COUNT permet de compter les lignes.

Soit une table :

❑ Eleves (NomElv, AdrElv, VilleElv).

Le nombre d'élèves est.

```
SELECT COUNT(NomElv)  
FROM Eleves;
```

6.3 Le SUM

La commande SUM fait la somme des valeurs d'un champ.

Soit une table Acomptes (DateAcpt, NomOvr, Montant).

La somme des acomptes de chaque ouvrier.

```
SELECT NomOvr, SUM(Montant)
FROM AComptes
GROUP BY NomOvr;
```

7 Fonctions diverses de sélection sur un attribut

7.1 BETWEEN

BETWEEN s'utilise avec la close WHERE.

Soit une table

❑ Acomptes (DateAcpt, NomOvr, Montant).

La liste des acomptes versés entre le 01/02/98 et le 28/02/98.

```
SELECT * (Affiche tout les champs de la table Acomptes)
FROM AComptes
WHERE DateAcpt BETWEEN '01/02/98' AND '28/02/98';
```

7.2 DISTINCT

La commande DISTINCT permet de supprimer les lignes en doublons.

Soit une table :

❑ Eleves (NomElv, AdrElv, VilleElv).

La liste des villes où habitent les élèves.

```
SELECT DISTINCT VilleElv
FROM Eleves;
```

7.3 LIKE

L'instruction LIKE permet de rechercher des occurrences dans les chaînes de caractères. Elle s'utilise avec la close WHERE.

Soit une table :

❑ Eleves (NomElv, AdrElv, VilleElv).

La liste des élèves dont le nom commence par "c" ou "C".

```
SELECT * (Affiche tout les champs de la table Elèves)
FROM Eleves
WHERE NomElv LIKE 'C%' OR NomElv LIKE 'c%';
```

Notez que le caractère « % » qui sert de « joker » pour toute chaîne de 0 à n caractères. Le caractère « _ » underscore sert de joker pour un caractère unique. Le code suivant :

```
SELECT * (Affiche tout les champs de la table Elèves)
FROM Eleves
WHERE NomElv LIKE '%C%' OR NomElv LIKE '%c%';
```

Renvoie la liste des élèves dont le nom contient un « C »

8 Fonctions booléennes

8.1 UNION

UNION permet de fusionner deux tables de même schéma par exemple CLIENTS et FOURNISSEURS

```
SELECT * FROM CLIENTS
UNION
SELECT * FROM FOURNISSEURS
```

8.2 INTERSECT

```
SELECT * FROM CLIENTS
INTERSECT
SELECT * FROM FOURNISSEURS
```