SGBD

III.
Le langage d'interrogation SQL

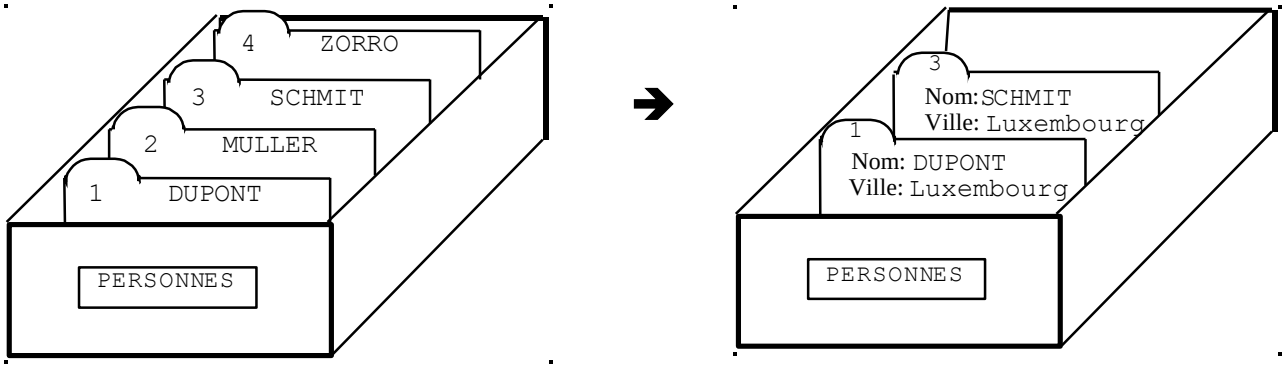
Version: 2.2018

manuel élaboré par
Jean-Marie Ottelé, ECG

e-mail : jean-marie.ottele@education.lu

III. Les langages d'interrogation

Un langage d'interrogation permet d'extraire les données d'un système de gestion de bases de données à l'aide de requêtes (e : queries) formulées.



Il existe 2 langages d'interrogation (e : query language) standard pour les SGBD relationnels :

SQL

SQL = **S**tructured **Q**uery **L**anguage
(Langage d'interrogation structuré)

SQL est un langage textuel aux règles syntaxiques précises. SQL a été développé pour le système R d'IBM. Il a été ensuite adopté comme norme de langage de manipulation de bases de données et a été repris dans la quasi-totalité des SGBD.

La structure de base des requêtes écrites en SQL:

```
SELECT <liste de champs>
FROM <liste de tables>
WHERE <condition>
```

QBE

QBE = **Q**uery **B**y **E**xample
(Interrogation par l'exemple)

QBE est un langage visuel à interface graphique développé par IBM. Pour manipuler les données l'utilisateur n'a qu'à remplir les champs prédéfinis sur l'écran.

Champ/Field		
Table/Table		
Tri/Sort		
Afficher/Show		
Critère/Criteria		

Ces 2 langages relationnels ont en commun d'être non-procéduraux, c.-à-d. l'utilisateur indique uniquement le résultat qui l'intéresse sans préciser comment procéder pour y parvenir.

SQL et QBE peuvent cependant posséder quelques particularités selon le SGBD utilisé.

Sources : SQL2 de C. Marée et G. Ledant - Edition Armand Colin ISBN : 2-200-21411-1

Informatique appliquée à la gestion de C. Moine et B. Herz – Edition Foucher ISBN : 2-216-03481-9

Syntaxe SQL

Tableau des commandes:

SQL interactif			SQL intégré	SQL dynamique
LDD	**LMD**	**LCD**		
CREATE DROP ALTER	**INSERT** **DELETE** **UPDATE** **SELECT**	GRANT REVOKE CONNECT COMMIT ROLLBACK SET	DECLARE CURSOR FETCH	PREPARE DESCRIBE EXECUTE

LDD: Langage de définition des données (Data Definition Language, DDL)

LMD: Langage de manipulation des données (Data Manipulation Language, DML)

LCD: Langage de contrôle des données (Data Control Language, DCL)

Traité en cours: LMD

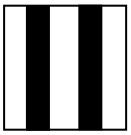
Table : client

numcli	nom	adresse	localité	datenaiss	marié	nbrenfants
1	Schmit	9 Rue G-D Charlotte	Mersch	20.04.1995	☑	1
2	Duront	101 Boulevard Royal	Luxembourg		☐	2
3	Dupont	33 Rue de l'Alzette	Mersch	30.12.1994	☑	3
4	Muller	6 Rue de Schouweiler	Hautcharage	11.11.1995	☑	2
5	Zorro	32 Rue Adolphe	Pétange	04.04.1987	☑	4
6	Aach	1 Rue du 9 Mai 1944	Howald	05.04.2000	☐	0
7	Maier	10 Avenue de la Gare	Mersch	06.12.1997	☐	5
8	Schmit	234 Rue de Luxembourg	Hautcharage	30.10.1991	☐	
9	Schmit	7 Rue de la Gare	Mersch	20.05.1994	☑	6

Créer la structure de la table en choisissant les types appropriés. Entrer les données.

Nom champ	Type
numcli	
nom	
adresse	
localité	
datenaiss	
marié	
nbrenfants	
heurenaiss	

Créer une requête	
Exécuter une requête	

1. La projection  SELECT

La projection permet de ne conserver que les champs (colonnes) intéressants.

Exemple 1: Afficher le nom et la localité de tous les clients.

SQL	QBE		
…… …… ……	Champ/Field		
	Table/Table		
	Tri/Sort		
	Afficher/Show		
	Critère/Criteria		

Résultat de la requête :

Exemple 2: Afficher tous les champs de tous les clients.

SQL	QBE				
…… …… …… ……	Champ/Field				
	Table/Table				
	Tri/Sort				
	Afficher/Show				
	Critère/Criteria				

Résultat de la requête :

2. La sélection  WHERE

La sélection permet de ne conserver que les enregistrements (lignes) respectant une condition vérifiée.

a. La formulation des conditions

La forme d'une condition dépend du type du champ:

Type du champ	Exemple: Affichez tous les champs des clients	Condition
caractère/texte	habitant à Luxembourg	
	n'habitant pas à Luxembourg	
numérique	ayant 2 enfants	
	ayant plus de 2 enfants	
logique/booléen	étant mariés	
	n'étant pas mariés	
date	nés le 5 avril 2000	
	nés avant le 5 avril 2000	
heure	nés avant 9h15	

Attention : au format **numérique** pour le séparateur décimal : ….
au format **date** : …….

b. Les opérateurs de comparaison

- égal (=) ..... • différent de (≠) .....
- supérieur (>) ..... • inférieur (<) .....
- supérieur ou égal à (≥) ..... • inférieur ou égal à (≤) .....

Exemple 1: Afficher le nom de tous les clients habitant à Mersch.

SQL : …….

…….

…….

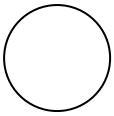
…….

Résultat de la requête :

c. Les opérateurs logiques

Les opérateurs logiques permettent de créer des requêtes composées ou permettent la négation d'une condition.

L'opérateur NOT



L'opérateur NOT inverse la valeur d'une condition.

Exemple 1: Afficher le nom et la localité des clients n'habitant pas à Mersch (utiliser NOT).

SQL : …….

…….

…….

…….

Résultat de la requête :

Exemple 2: Afficher le nom des clients étant célibataires.

SQL : …….

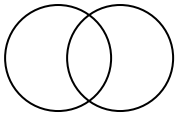
…….

…….

…….

Résultat de la requête :

L'opérateur AND



L'opérateur AND permet de ne considérer les enregistrements vérifiant **simultanément** les deux conditions unies par un ET logique.

Exemple 1: Afficher le nom des clients mariés habitant à Mersch.

SQL : …….

…….

…….

…….

Résultat de la requête :

Exemple 2: Afficher le nom, la localité et le nombre d'enfants des clients ayant 1 à 2 enfants.

SQL : …….

…….

…….

…….

Résultat de la requête :

Exemple 3: Afficher le nom et la localité des clients habitant à Mersch **et** à Pétange.

SQL : …….

…….

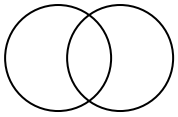
…….

…….

Résultat de la requête :

Remarque :

L'opérateur OR



L'opérateur OR permet de considérer les enregistrements vérifiant une première condition et/ou ceux d'une deuxième condition.

Exemple 1: Afficher le nom et la localité des clients habitant à Mersch ou à Pétange.

SQL : …….

…….

…….

…….

Résultat de la requête :

Exemple 2: Afficher le nom des clients ayant 1 ou 2 enfants.

SQL : …….

…….

…….

…….

Résultat de la requête :

d. L'indétermination d'un champ

Tout champ ayant un contenu indéterminé possède la valeur **`IS NULL.`**

SQL

Exemple 1: Afficher le nom des clients qui n'ont pas donné des renseignements concernant leur date de naissance.

SQL : …….

…….

…….

Résultat de la requête :

Exemple 2: Afficher le nom des clients qui ont donné des renseignements concernant leur date de naissance.

SQL : …….

…….

…….

Résultat de la requête :

Exemple 3: Afficher le nom des clients qui n'ont pas d'enfants.

SQL : …….

…….

…….

Résultat de la requête :

Exemple 4: Afficher le nom des clients qui n'ont pas donné des renseignements concernant le nombre d'enfants.

SQL : …….

…….

…….

Résultat de la requête :

e. Les caractères génériques / les filtres

Les caractères génériques (e: jokers ; d : Platzhalter) permettent de remplacer des caractères inconnus lors d'une recherche. On ne peut utiliser ces caractères génériques que sur des champs de type texte et heure/date.

SQL

?

—

Le point d'interrogation représente n'importe quel caractère situé à cet endroit.

Par exemple, **Dupon?** trouvera: Dupong, Dupond, Dupont, etc.
Du?ont trouvera: Dumont, Dupont, Duront, etc.
????? trouvera tous les mots de 5 caractères

*****

%

L'astérisque représente aucun ou plusieurs caractères.

Par exemple, **Luxemb*** trouvera: Luxembourg, Luxemburg, Luxemburgo, etc.
***bourg** trouvera: Bourg, Bettembourg, Hambourger, Luxembourg, etc.
t trouvera tous les mots contenant au moins un «t»

Quelle condition affiche ? ☒	WHERE nom = "Hôtel du Parc *"	WHERE nom LIKE "Hôtel du Parc *"
Hôtel du Parc	☐	☐
Hôtel du Parc *	☐	☐
Hôtel du Parc **	☐	☐
Hôtel du Parc et du Lac	☐	☐

……..

……..

Remarque: Lorsqu'on veut utiliser le point d'interrogation (?) ou l'astérisque (*) dans sa **fonction propre** (p.ex. nomfilm = "??? ", alors

il faut utiliser: .....

Exemple 1: Afficher les localités des clients habitant dans une localité commençant avec « H ».

SQL
…….. …….. ……..

Résultat de la requête :

Exemple 2: Afficher toutes les données des clients habitant dans une localité contenant la lettre « h ».

SQL : ……

……

……

Résultat de la requête :

Exemple 3: Afficher les localités des clients ayant 6 lettres.

SQL : ……

……

……

Résultat de la requête :

Exemple 4: Afficher les localités ayant comme 4e lettre un « a ».

SQL : ……

……

……

Résultat de la requête :

Exemple 5: Afficher les localités n'ayant pas comme 4e lettre un « a ».

SQL : ……

……

……

Résultat de la requête :

Exemple 6: Afficher le nom des clients ayant comme dernière lettre un « t ».

SQL : ……

……

……

Résultat de la requête :

Exemple 7: Afficher le nom des clients ayant leur adresse dans un boulevard.

SQL : ……

……

……

Résultat de la requête :

Exemple 8: Afficher toutes les données des clients habitant dans une localité contenant minimum 2 fois la lettre « h ».

SQL : ……

……

……

……

Résultat de la requête :

Exemple 9: Afficher le nom de tous les clients ayant dans leur nom 1 ou plusieurs caractères précédant la première lettre « a » .

SQL : …….

…….

…….

…….

Résultat de la requête :

Exemple 10: Afficher les localités de tous les clients habitant dans une localité pouvant comporter plusieurs lettres dont un et un seul « h ».

SQL : …….

…….

…….

…….

Résultat de la requête :

3. Comparaison à une fourchette de valeurs

SQL

L'opérateur BETWEEN ... AND … permet de sélectionner les enregistrements dont le champ spécifié contient une valeur dans un intervalle déterminé.

Exemple 1 : Afficher le nom et le nombre d'enfants des clients ayant 1 à 3 enfants.

SQL : …….

…….

…….

Résultat de la requête :

Exemple 2 : Afficher le nom et le nombre d'enfants des clients n'ayant pas 1 à 3 enfants.

SQL : …….

…….

…….

Résultat de la requête :

Exemple 3: Afficher le nom et la date de naissance des clients nés entre le 12.2.1994 et le 31.12.1995.

SQL : …….

…….

…….

Résultat de la requête :

Exemple 4 : Afficher le nom et la date de naissance des clients nés entre le 12.2.1994 et le 31.12.1995 (sans BETWEEN).

SQL : …….

…….

…….

4. Comparaison à une liste de valeurs

SQL

L'opérateur IN permet de sélectionner les enregistrements dont le champ spécifié contient une valeur apparaissant dans la liste de valeurs suivant l'opérateur.

Exemple 1 : Afficher le nom et le nombre d'enfants des clients ayant 0, 2 ou 4 enfants. (avec IN)

SQL : …….

…….

…….

Refaire le même énoncé sans IN :

SQL : …….

…….

…….

…….

…….

Résultat de la requête :

Remarque :

IN remplace plusieurs …….

Exemple 2: Afficher le nom et la localité des clients habitant à Howald, Mersch ou Pétange.

SQL : …….

…….

…….

Résultat de la requête :

Exemple 3: Afficher le nom et la localité des clients n'habitant pas à Howald, Mersch ou Pétange.

SQL : …….

…….

…….

Résultat de la requête :

5. La priorité des opérateurs logiques

L'ordre d'exécution des expressions est défini par la priorité des opérateurs :

SQL effectue d'abord les **comparaisons** (=, <>, <, >, <=, >=),
puis les **NOT**, puis les **AND** et en dernier les **OR** (et ceci de la gauche vers la droite).

En cas de doute, il est conseillé d'employer des parenthèses afin de grouper les expressions.
p. ex. C1 OR C2 AND (C3 OR C4)

Priorité des opérateurs	arithmétiques	logiques
1	( )	( )
2	Exposant	NOT
3	* /	AND
4	+ -	OR

Exemples:	2 * 3 + 4 = 2 + 3 * 4 = 2 + 3^2 = (2 + 3) * 4 = 2 * 4 + 3 * 4 =	C1 AND C2 OR C3 C1 OR C2 AND C3 C1 OR C2 AND NOT C3 (C1 OR C2) AND C3 C1 AND C3 OR C2 AND C3

Exemple 1: Montrer le nom et la localité des clients habitant ni à Mersch ni à Hautcharage (4 possibilités).

SQL : …….

…….

…….

ou

…….

…….

ou

…….

…….

ou

…….

…….

Résultat de la requête :

Exemple 2: Montrer le numéro, le nom et la localité des clients habitant soit à Mersch, soit à Hautcharage, mais qui sont tous célibataires.

a) SQL : avec IN :

…….

…….

…….

…….

…….

b) SQL : avec AND et OR :

…….

…….

…….

…….

…….

c) SQL : avec AND et OR sans parenthèses:

…….

…….

…….

…….

…….

…….

Résultat de la requête :

Exemple 3: Montrer le nom et le nombre d'enfants des clients n'ayant pas 1 – 3 enfants.

SQL : ……

……

……

……

ou

SQL : ……

……

……

……

……

Résultat de la requête :

Exemple 4: Montrer le nom, la localité, le nombre d'enfants et l'état civil des clients habitant à Mersch ou ayant 2 enfants, et qui sont tous célibataires.

SQL : ……

……

……

……

……

……

Ou

SQL : ……

……

……

……

……

……

Résultat de la requête :

6. Empêcher les répétitions de lignes

Les langages d'interrogation affichent par défaut le résultat d'une requête sans éliminer les répétitions de lignes. Si on veut éliminer ces répétitions (doublons) il faudra le spécifier.

SQL

Exemple 1: Afficher la liste de toutes les localités, en énumérant chaque localité qu'une seule fois.

SQL
…….. ……..

Résultat de la requête :

Exemple 2: Afficher le nom et la localité des clients en énumérant chaque nom et localité qu'une seule fois

SQL
…….. ……..

Résultat de la requête :

Exemple 3: Afficher le nom des clients en énumérant chaque nom qu'une seule fois

SQL
…….. ……..

Résultat de la requête :

Exemple 4: Combien de lignes produisent les commandes ci-dessous :

a)	SELECT nom	
b)	SELECT DISTINCT nom	
c)	SELECT nom, localité	
d)	SELECT DISTINCT nom, localité	
e)	SELECT DISTINCT nom, localité, datenaiss	
f)	SELECT DISTINCT numcli, nom	

7. Le tri / le classement

Le tri permet de classer les enregistrements du résultat d'une requête dans un ordre alphabétique, numérique ou chronologique.

L'ordre de tri

L'ordre de tri correspond à la manière dont on souhaite organiser les données. On peut trier une table en ordre croissant (e: ascending) ou décroissant (e: descending). Le tri se fait par défaut par ordre croissant.

Si on utilise un ordre croissant, le texte est classé de A à Z, les nombres sont classés de 0 à 9 et les dates de la plus ancienne à la plus récente.

SQL

.....

.....

Si on utilise par contre un ordre décroissant, le texte est classé de Z à A, les nombres sont classés de 9 à 0 et les dates de la plus récente à la plus ancienne.

SQL

.....

.....

Remarques :

Dans un système de gestion de base de données relationnel, les enregistrements n'ont pas d'ordre particulier dans une table !

Ce n'est que par des commandes de tri que les enregistrements sont visualisés selon un critère de tri spécifié. On peut utiliser plusieurs critères de tri pour sortir les enregistrements dans un ordre bien précis.

Le résultat du tri peut varier selon le type (texte, numérique, date) du champ trié.

Un annuaire téléphonique est trié sur quels critères ?

Exemple 1: Afficher le nom et la localité de tous les clients, ayant au moins un enfant, triés selon l'ordre croissant des localités.

SQL : …….

…….

…….

…….

Résultat de la requête :

Exemple 2: Afficher les adresses triées de manière croissante de tous les clients.

SQL : …….

…….

…….

…….

Résultat de la requête :

Remarque

Tri sur texte :	Tri sur nombres

Comment faut-il saisir les adresses pour pouvoir trier les rues correctement ?

Exemple 3: Afficher la localité, le nom, l'adresse et le nombre d'enfants de tous les clients triés selon l'ordre décroissant des localités, croissant des noms et décroissant des adresses.

SQL : …….

…….

…….

…….

Résultat de la requête :

8. L'équi-jointure

L'équi-jointure permet de fusionner les enregistrements de 2 tables possédant un domaine commun.

SQL

Table : client

numcli	nom	adresse	localité
1	Schmit	9 Rue G-D Charlotte	Mersch
2	Duront	101 Boulevard Royal	Luxembourg
3	Dupont	33 Rue de l'Alzette	Mersch
4	Muller	6 Rue de Schouweiler	Hautcharage
5	Zorro	32 Rue Adolphe	Pétange
6	Aach	1 Rue du 9 mai 1944	Howald

Table : commande

numcom	date	numcli
1	2.1.2018	2
2	2.1.2018	3
3	5.1.2018	2
4	3.1.2018	4

Dessinez le MLD.

Exemple d'une équi-jointure

Table : client

numcli	nom	adresse	localité
1	Schmit	9 Rue G-D Charlotte	Mersch
2	Duront	101 Boulevard Royal	Luxembourg
3	Dupont	33 Rue de l'Alzette	Mersch
4	Muller	6 Rue de Schouweiler	Hautcharage
5	Zorro	32 Rue Adolphe	Pétange
6	Aach	1 Rue du 9 mai 1944	Howald

Table : commande

numcom	date	numcli
1	2.1.2018	2
2	2.1.2018	3
3	5.1.2018	2
4	3.1.2018	4

Exemple 1 : Créez une équi-jointure entre les tables Client et Commande :

SQL : …….

…….

…….

Résultat de la requête :

numcom	date	Commande . numcli	Client. numcli	nom	adresse	localité
1	2.1.2018	2	2	Duront	101 Boulevard Royal	Luxembourg
2	2.1.2018	3	3	Dupont	33 Rue de l'Alzette	Mersch
3	5.1.2018	2	2	Duront	101 Boulevard Royal	Luxembourg
4	3.1.2018	4	4	Muller	6 Rue de Schouweiler	Hautcharage

Si on a … lignes dans la clé étrangère de l'équi-jointure, alors le résultat de l'équi-jointure présentera … lignes.

Avec 4 clients et 10 commandes, le résultat comportera …… lignes, comme la clé étrangère ……..se trouve dans la table ……..

Pour chaque clé étrangère il faut trouver une clé primaire correspondante.

Exemple 2: Afficher derrière chaque numéro de commande le nom et le numéro du client correspondant.

SQL

……..

……..

……..

……..

Résultat de la requête :

Exercice :

table: commande

numcom	date
1	15.2.2018
2	16.2.2018
3	17.2.2018
4	18.2.2018

table: comprendre

numcom	numpro	quantité
1	1	2
1	3	1
2	2	4
2	1	3
2	6	1
2	5	3

table produit

numpro	nom	prix
1	Ajax	3
2	Bjax	4
3	Cjax	1
4	Djax	5
5	Ejax	6
6	Fjax	7
7	Gjax	2

1. Souligner les clés primaires des 3 tables et entourez les clés étrangères.

2. Equi-jointure entre les tables commande et comprendre ?

   a) Définir l'équi-jointure

   ……..

   ……..

   b) Nombre de lignes : …….

3. Equi-jointure entre les tables comprendre et produit ?

   a) Définir l'équi-jointure

   ……..

   ……..

   b) Nombre de lignes : …….

4. Equi-jointures entre les tables commande, comprendre et produit ?

   a) Définir les équi-jointures

   ……..

   ……..

   ……..

   b) Nombre de lignes : …….

9. L'ALIAS

Un ALIAS est un nom synonyme ou un nom de remplacement d'une expression, d'un champ ou d'une table.
L'utilisation d'un ALIAS peut avoir plusieurs intérêts:

- employer une abréviation du nom d'une table, d'un champ[1] ou d'une expression[1], ce qui simplifie le travail avec de longs noms
- employer un nom pour une expression, ce qui rend le résultat de la requête plus significatif
- une requête comportant un critère portant sur la valeur d'un champ par rapport à la valeur de ce même champ dans un autre enregistrement de la même table: l'auto-jointure

SQL

1) Pas possible avec les versions actuelles d'ACCESS.

Exemple :

```
SELECT prix * quantité AS total
FROM client AS cl, commande AS CO
WHERE cl.num_cli = co.num_cli;
```

10. Fonctions et expressions

10.1 Fonctions numériques

Chaque SGBD propose des fonctions diverses :

Fonction		Exemple	Résultat
Extraire le jour d'une date			
Extraire le mois d'une date			
Extraire l'année d'une date			

Exercice 1 : Afficher les années de naissance des clients.

SQL : ……

……

……

Résultat de la requête :

Exercice 2 : Afficher le nom et la date de naissance des clients nés en 1994.

SQL : ……

……

……

Résultat de la requête :

Exercice 3 : Afficher le nom et la date de naissance des clients nés en avril.

SQL : ……

……

……

Résultat de la requête :

Exercice 4 : Afficher les clients nés un 20 avril.

SQL : ……

……

……

……

Résultat de la requête :

10.2 Expressions numériques

Il est possible de combiner la valeur de différentes valeurs numériques par les opérateurs arithmétiques (* / + -). Les opérateurs * et / ont le plus haut niveau de priorité. Il faut utiliser des parenthèses pour changer cet ordre de calcul.

Exercice 1 : Pour la fête de Saint Nicolas, tout client reçoit un bon de 25,50 € par enfant. Afficher le nom des clients ainsi que le montant de chaque bon.

SQL : ……

……

……

Résultat de la requête :

10.3 Expressions avec dates et des intervalles

Il est possible d'ajouter un intervalle à une date pour obtenir une autre date ou de calculer un intervalle (en jours) par la différence entre deux dates.

p. ex. #1/28/2018# - #1/26/2018# = …….. jours

Exercice 2 : Afficher le nom et l'âge des clients en jours en utilisant la fonction ..... pour connaître la date courante.

SQL : …….

…….

…….

Résultat de la requête :

Exercice 3 : Afficher le nom et l'âge des clients en années.

SQL : …….

…….

…….

Résultat de la requête :

Exercice 4 : Afficher les numéros des commandes passées il y a deux semaines et plus.

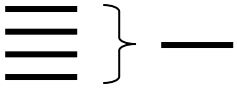
SQL : …….

…….

…….

Résultat de la requête :

10.4 Regroupement - Fonctions de groupe



Les fonctions de groupe (e : aggregate function) effectuent un calcul sur l'ensemble des valeurs d'un champ pour un groupe d'enregistrements. Un groupe est un sous-ensemble d'enregistrements d'une table, pour lesquels la valeur d'un champ reste constante. Le groupe est constitué de l'ensemble des enregistrements sélectionnés.

Les fonctions de groupe les plus courantes sont :

- COUNT() compte le nombre d'occurrences du champ
- SUM() calcule la somme des valeurs du champ (de type numérique)
- AVG() calcule la moyenne des valeurs du champ (de type numérique)
- MAX() recherche la plus grande valeur du champ
- MIN() recherche la plus petite valeur du champ

Exercice 1 : Afficher le nombre moyen d'enfants des clients.

SQL : ……

……

……

Résultat de la requête :

Exercice 2 : Afficher le nombre total d'enfants des clients.

SQL : ……

……

……

Résultat de la requête :

Exercice 3 : Afficher le nombre d'enfants le plus élevé des clients.

SQL : ……

……

……

Résultat de la requête :

Exercice 4 : Afficher le nombre de clients.

SQL : ……

……

……

Résultat de la requête :

Exercice 5 : Afficher l'anniversaire du client le plus jeune.

SQL : ……

……

……

Résultat de la requête :

Exercice 6 : Afficher l'anniversaire du client le plus âgé.

SQL : ……

……

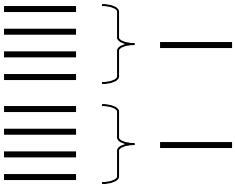
……

Résultat de la requête :

11. Requêtes sur les groupes

Un groupe est un sous-ensemble d'enregistrements d'une table, pour lesquels la valeur d'un champ reste constante.

11.1 La clause GROUP BY



La clause GROUP BY regroupe en une seule ligne des enregistrements qui possèdent des valeurs identiques à l'intérieur de chaque groupe pour les champs spécifiés.

Tous les champs dans la liste de champs du SELECT doivent être inclus dans la clause GROUP BY ou être inclus dans une fonction de regroupement.

Exercice 1 : Compter le nombre de clients par localité. Afficher les localités et les nombres.

SQL : …….

…….

…….

…….

Résultat de la requête :

Exercice 2 : Calculer la moyenne de nombre d'enfants par année de naissance des clients. Afficher les années de naissance, ainsi que les moyennes. Trier les années de manière décroissante.

SQL : …….

…….

…….

…….

Résultat de la requête :

Exercice 3 : Calculer l'âge moyen (en années) des clients en fonction du nombre d'enfants. Afficher le nombre d'enfants, ainsi que l'âge moyen.

SQL : …….

…….

…….

Résultat de la requête :

Exercice 4 : Afficher le nombre total d'enfants par année de naissance des clients.

SQL : …….

…….

…….

Résultat de la requête :

Exercice 5 : Compter le nombre de commandes par client. Afficher le numéro des clients, ainsi que le nombre de commandes.

SQL : …….

…….

…….

Résultat de la requête :

Exercice 6 : Compter le nombre de commandes par client. Afficher le numéro et le nom des clients, ainsi que le nombre de commandes.

SQL : …….

…….

…….

…….

Résultat de la requête :

12.1 La clause HAVING

La condition de la clause **HAVING** s'applique aux **fonctions de groupe**, celle du **WHERE** s'applique aux **champs**.

En pratique, la condition spécifiée dans la clause HAVING porte toujours sur la valeur d'une fonction **calculée** sur un groupe.

Il n'y a **jamais** un HAVING **sans** GROUP BY.

Exercice 1 : Afficher les localités dans lesquelles habitent au moins 2 clients.

SQL : …….

…….

…….

…….

Résultat de la requête :

Exercice 2 : Afficher les localités dans lesquelles la moyenne d'enfants par client est inférieure ou égale à 2.

SQL : …….

…….

…….

…….

Résultat de la requête :

12. La requête imbriquée

Dans des requêtes imbriquées le résultat de la requête la plus imbriquée sert de départ à la requête de niveau immédiatement supérieur. Les requêtes de niveau inférieur sont comprises entre parenthèses. Le SELECT de la requête imbriquée comporte une seule colonne (un champ ou une expression) comportant une ou plusieurs valeurs.

12.1 La requête interne renvoyant une seule valeur

La valeur retournée par la requête interne sert de valeur de comparaison dans la clause WHERE ou HAVING. Une telle requête ne fonctionne que si la requête interne retourne une valeur et une seule. Il faut aussi que cette valeur retournée soit du même type que le champ auquel elle est comparée !

Exercice 1 Afficher le nom et la date de naissance de tous les clients étant plus âgés que le client N°4.

SQL : …….

…….

…….

…….

…….

…….

Résultat de la requête :

Exercice 2 : Refaire exercice 1 en affichant après les données de chaque client en plus le nom et la date de naissance du client N°4.

SQL : …….

…….

…….

…….

…….

…….

Résultat de la requête :

Exercice 3 : Afficher les noms des clients qui ont plus d'enfants que la moyenne.

SQL : ……

……

……

……

……

……

Résultat de la requête :

Exercice 4 : Afficher les noms des clients qui ont plus d'enfants que Zorro.

SQL : ……

……

……

……

……

……

Résultat de la requête :

Exercice 5 : Afficher les noms des clients qui ont plus d'enfants que Schmit.

SQL : ……

……

……

……

……

……

Résultat de la requête :

Remarque :

12.2 La requête interne renvoyant plusieurs valeurs

Les valeurs retournées par la requête interne servent de valeurs dans une clause WHERE comportant un opérateur IN.

Table : client

numcli	nom	adresse	localité
1	Schmit	9 Rue G-D Charlotte	Mersch
2	Duront	101 Boulevard Royal	Luxembourg
3	Dupont	33 Rue de l'Alzette	Mersch
4	Muller	6 Rue de Schouweiler	Hautcharage
5	Zorro	32 Rue Adolphe	Pétange
6	Aach	1 Rue du 9 Mai 1944	Howald
7	Maier	10 Avenue de la Gare	Mersch
8	Schmit	234 Rue de Luxembourg	Hautcharage
9	Schmit	7 Rue de la Gare	Mersch

Table : commande

numcom	date	numcli
1	2.1.2018	2
2	2.1.2018	3
3	5.1.2018	2
4	3.1.2018	4

Exercice 1 : Afficher les noms des clients qui ont le même nombre d'enfants que les Schmit.

SQL : ……

……

……

……

……

……

Résultat de la requête :

Exercice 2 : Afficher le nom et la localité des clients habitant dans une localité où habite au moins un client dont le nom commence par « S »

SQL : ……

……

……

……

……

……

Résultat de la requête :

Exercice 3 : Afficher le nom des clients qui n'ont pas encore passé de commande.

SQL : ……

……

……

……

……

……

Résultat de la requête :

Remarque : Dans le cadre de ce type d'exercices il ne faut jamais utiliser une …… entre la table client et commande !

Ex 4 - Révision

eleves

noeleve	nom	noclasse
1	Dupont	13
2	Duront	12
3	Dumont	
4	Dunont	12
5	Dutont	13
6	Ducont	11

classes

noclasse	nom
11	1GCG1
12	1GCG2
13	1GCG3
14	1GCG4

a) Afficher le nom des élèves sans classe

Nom
Dumont

SQL : ……

……

……

……

b) Afficher le nom des classes sans élèves

Nom
1GCG4

SQL : ……

……

……

……

c) Afficher le nom des classes ayant un ou plusieurs élève(s). Afficher chaque classe qu'une seule fois.

Nom
1GCG1
1GCG2
1GCG3

SQL : ……

……

……

……

d) Afficher le nom de la / des classes ayant un seul élève.

Nom
1GCG1

SQL : ……

……

……

……

……

13. Empêcher les répétitions dans des fonctions de groupe

On peut empêcher les répétitions de valeurs dans des fonctions de groupe à l'aide de:

SQL

Remarque : Ceci n'est pas implémenté dans les versions actuelles d'ACCESS, mais dans BASE.

Exercice 1: Compter le nombre de localités différentes dans la table client

SQL : …….

…….

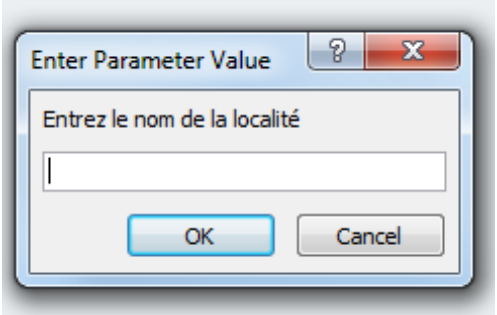
…….

Résultat de la requête :

14. Requêtes paramétrées

On peut automatiser le processus de modification des critères qu'on exécute régulièrement en créant une requête paramétrée. Le SGBD invite l'utilisateur alors à entrer les critères au moment de l'exécution de la requête. L'utilisateur n'a donc plus besoin de modifier la requête pour changer les critères.

Exercice 1: Afficher le nom et l'adresse des clients dont la localité reste à spécifier lors de l'exécution de la requête.

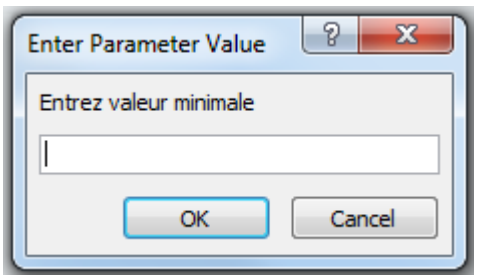


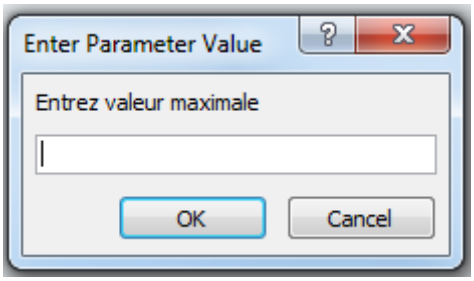
SQL : ……

……

……

Exercice 2: Afficher le nom et le nombre d'enfants des clients dont le nombre d'enfants se situe entre 2 valeurs entrées par paramètre lors de l'exécution de la requête.



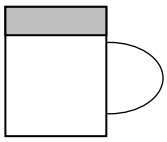


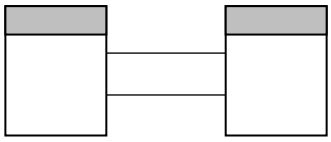
SQL : ……

……

……

15. L'auto-jointure - relation double





L'auto-jointure est une requête comportant un critère portant sur la valeur d'un champ par rapport à la valeur de ce même champ dans un autre enregistrement de la même table. L'auto-jointure est utilisée pour toute relation réflexive.

Exemple 1:

Table: personnel

num_personnel	nom	prénom	num_personnel_chef
1	Schmit	Isabelle	7
2	Muller	Carine	6
3	Feller	Pierre	5
4	Maurer	Lea	3
5	Popov	Alex	
6	Thill	Paul	3
7	Schmitz	Charlotte	5
8	Meier	Lynn	6

a) Dessiner le MCD et le MLD de la table: personnel.
b) Dessiner le MLD des tables "alias": employé et chef.
c) Dessiner l'organigramme de l'entreprise

d) Montrer la liste des noms et prénoms de tout employé ensemble avec le nom et prénom de son chef respectif.

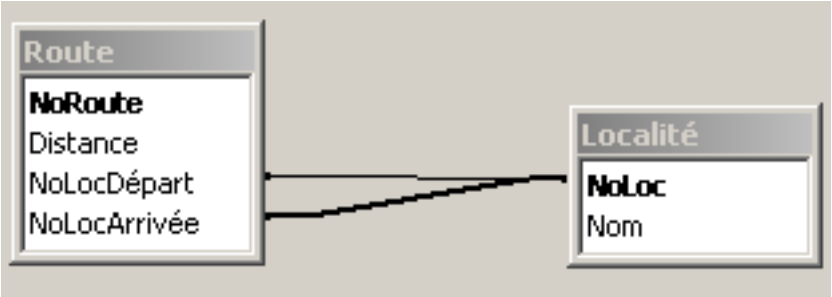
SQL : …….

…….

…….

Résultat de la requête :

Exemple 2: Afficher pour toutes les routes d'une longueur supérieure à 20 km : le numéro de la route, le nom de la localité de départ et le nom de la localité d'arrivée.



Redessiner le MLD avec les tables "alias".

SQL : ……

……

……

……

……

Exemple 3: Afficher le nom des élèves avec le nom de sa classe actuelle et précédente.

Elève
No_Eleve
Nom
NoClassePrécédente
NoClasseActuelle

Classe
NoClasse
Nom

a) Dessiner les relations entre les 2 tables
b) Redessiner le MLD avec les tables "alias".

SQL : ……

……

……

……

……

16. SQL: Règles simplifiées

SELECT

SELECT <tous les champs et expressions à afficher>

- Utiliser un alias lorsque le libellé d'un champ ou d'une expression doit être changé.
  p.ex.

SQL standard	SQL ACCESS
pré_cli prénom	pré_cli AS prénom

- Ajouter le nom de la table devant le nom du champ lorsque le nom d'un champ intervient dans 2 tables utilisées.
  p. ex. `client.nom, fournisseur.nom`

FROM

FROM < toutes les tables utilisées>

- Utiliser un alias lors de l'utilisation d'une auto-jointure ou d'une relation double.

WHERE

WHERE <équijointure(s)> et <condition(s) sur champs>

- Les champs / expressions des 2 côtés des opérateurs de comparaisons (=, <>, <, <=, >, >=) doivent être du même type (numérique, texte, date, logique, etc.).

GROUP BY

SELECT **<champ(s)> + <fonction(s) de groupe>**

…
GROUP BY <champ(s)>

HAVING

SELECT <champs> + <fonction(s) de groupe>
…
GROUP BY <champs>
HAVING <condition(s) sur fonction(s) de groupe>

- Les champs / expressions des 2 côtés des opérateurs de comparaisons (=, <>, <, <=, >, >=) sont de type numérique.

p.ex.
```
SELECT nom, COUNT(*) AS nombre
FROM client
GROUP BY nom
HAVING COUNT(*)>10;
```

ORDER BY

Différence entre `ORDER BY` et `GROUP BY`

num_fact	date_fact	num_cli
1	1.4.2018	1
2	2.4.2018	2
3	2.4.2018	3
4	3.4.2018	1

```
SELECT *
FROM facture
ORDER BY num_cli;
```

num_fact	date_fact	num_cli
1	1.4.2018	1
4	3.4.2018	1
2	2.4.2018	2
3	2.4.2018	3

ORDER BY = tri asc / desc

```
SELECT num_cli, count(*) AS nombre
FROM facture
GROUP BY num_cli;
```

num_cli	nombre
1	2
2	1
3	1

GROUP BY = regroupement

Requête imbriquée

```
…
WHERE <champ> <opérateur de comparaison> (SELECT <1 champ/expression>
                                                                    …)
```

```
…
HAVING <…> <opérateur de comparaison> (SELECT <1 champ/expression>
                                                                    …)
```

Fonctions de groupe

COUNT()

num_fact	date	num_cli
1	1.4.2018	1
2	2.4.2018	2
3	2.4.2018	3
4		1

`SELECT COUNT(*) FROM facture;`	`SELECT COUNT(num_fact) FROM facture;`	`SELECT COUNT(date) FROM facture;`	`SELECT COUNT(numcli) FROM facture;`
Résultat:	**Résultat:**	**Résultat:**	**Résultat:**

```
SELECT COUNT(*) AS nbre_fact, COUNT(DISTINCT num_cli)AS nbre_cli
FROM facture;
```

Résultat:

17. Exercices supplémentaires - Exercice A

table: commande

numcom	date
1	15.2.2018
2	16.2.2018
3	17.2.2018
4	18.2.2018

table: comprendre

numcom	numpro	quantité
1	1	2
1	3	1
2	2	4
2	1	3
2	6	1
3	5	3

table : produit

numpro	nom	prix unitaire
1	Ajax	3 €
2	Bjax	4 €
3	Cjax	1 €
4	Djax	5 €
5	Ejax	6 €
6	Fjax	7 €
7	Gjax	9 €

1) Soulignez les clés des tables ci-dessus. Remplissez les tableaux et formulez les requêtes en SQL suivantes:

2) Afficher toutes les lignes de commande:

numcom	date	numpro	quantité	prix unitaire

.....

.....

.....

.....

.....

3) Afficher toutes les lignes de commande avec le total à payer:

numcom	numpro	total

.....

.....

.....

.....

.....

4) Afficher le nombre de produits différents commandés par commande:

numcom	nombre de produits

……

……

……

……

……

5) Afficher la quantité totale commandée par produit:

numpro	nom	quantité totale

……

……

……

……

……

6) Afficher le total à payer par commande:

numcom	date	total

……

……

……

……

……

7) Afficher le nom des produits qui n'ont pas encore été commandés

nom

.....

.....

.....

.....

.....

8) Afficher le numéro et la date des commandes pour lesquelles on n'a pas encore commandées des produits.

numéro commande	date

.....

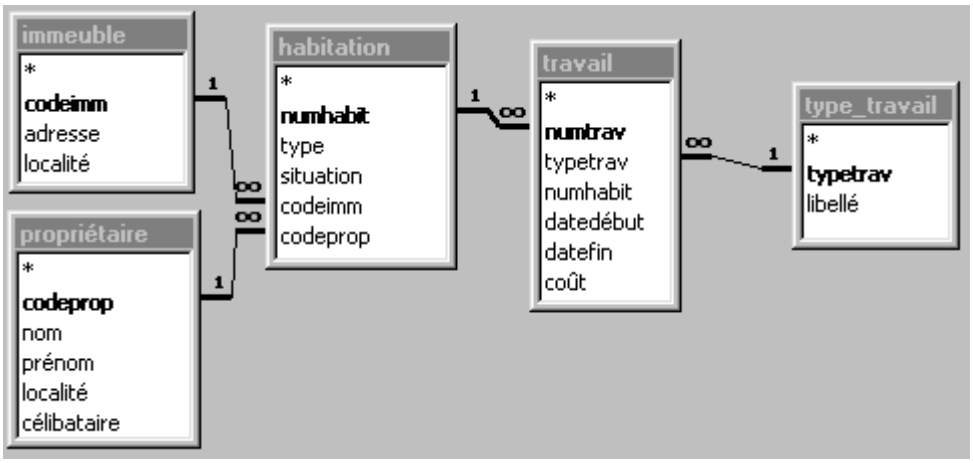
.....

.....

.....

.....

Exercice B



Description des champs:

type défini: studio; appartement; duplex; etc.
situation: rdc, 1er étage, 2e étage, etc.
datedébut est la date à laquelle le travail commence réellement.
datefin est la date à laquelle le travail s'est terminé. Pour les travaux en cours ce champ est laissé vide.
célibataire est un champ logique décrivant si on est célibataire ou pas

Réalisez les requêtes SQL suivantes:

a) Afficher pour tout travail qui est actuellement encore en cours, le libellé du travail, la date de début du travail, ainsi que la localité où ce travail a été effectué.

libellé	datedébut	localité
carrelages	20/11/2018	Luxembourg
électricité	21/12/2018	Mersch
cuisine	22/12/2018	Howald

b) Afficher le libellé de tous les travaux terminés depuis au moins un an

libellé
carrelages
chauffage
électricité
….

c) Calculer les dépenses totales par immeuble pour les travaux terminés en 2018.

codeimm	adresse	localité	Dépenses totales
i2	4 Rue Etroite	Esch	77000
i3	13 Rue des Etoiles	Luxembourg	23000

d) Afficher le libellé des types de travaux ayant coûté en total plus de 100.000 €

libellé	total
divers	500000
maçonnerie	1250000
sanitaire	670000

e) Afficher le nombre d'habitations par propriétaire. Trier de manière décroissante sur le nombre d'habitations et en suite de manière croissante sur le nom

codeprop	nom	nombre
11	Dos Santos	10
7	Miller	4
5	Schmitz	2

f) Afficher le type d'habitation de tous les célibataires dont le nom commence par C ou D. Afficher chaque type qu'une seule fois

type
appartement 2 ch.
duplex
studio
….