



Sommaire

1	Introduction .....	2
2	Présentation du modèle du cours .....	2
3	Le langage algébrique .....	3
3.1.1	La Projection (Sélection de colonnes) .....	3
3.1.2	La Sélection (Sélection de lignes) .....	3
3.1.3	La Jointure .....	4
3.1.4	Tri .....	4
3.1.5	Les autres opérations .....	4
3.1.6	Compositions des opérations .....	5
4	Le langage SQL .....	5
4.1	Présentation .....	5
4.2	Historique .....	6
4.3	Les conventions .....	6
4.4	Le langage d'interrogation des données LID .....	7
4.4.1	La clause SELECT .....	7
4.4.2	La clause WHERE .....	7
4.4.3	La notation pointée .....	8
4.4.4	La jointure .....	9
4.4.5	Les Sous interrogations ou Select imbriqués .....	11
4.4.6	Les opérateurs UNION et INTERSECT .....	12
4.4.7	Les fonctions d'agrégation: .....	13
4.4.8	Les clauses GROUP BY ET HAVING: .....	13
4.4.9	La clause ORDER BY : .....	14
4.4.10	Les pseudonymes ou alias .....	15
4.4.11	Les fonctions .....	15
4.4.12	Les champs calculés .....	16
5	Exercices .....	16



## 1 Introduction

Dès le début de l'informatique, on a voulu construire des systèmes pour effectuer des calculs (équations différentielles, calcul matriciel,...). Aujourd'hui, la tendance actuelle est la gestion de grandes quantités d'informations. Un des soucis majeur a été de gérer et de traiter ces informations de façon simple. Dans ce but, deux méthodes ont été créées : Une mathématique (le langage algébrique), et une plus opérationnelle pour les utilisateurs (ou les informaticiens) le langage SQL.

## 2 Présentation du modèle du cours

Pour les exemples nous nous appuyerons sur le modèle relationnel suivant :

VEHICULE (Code, Type, Marque, Puissance) → les types de véhicule

GARAGE (Code, Nom, Adresse, Ville, Cpostal, Marque) → les garages

PLAQUE (Immat, CodeV#, Nom, Prenom, Adresse, Ville, Cpostal, Date) → véhicules immatriculé

CodeV# de PLAQUE est clé étrangère et référence Code de VEHICULE.

La représentation Type Access

### Le jeu d'essai :

Table VEHICULE

Code	Type	Marque	Puissance
1	Espace	Renault	100
2	Clio	Renault	60
3	Megane	Renault	80
4	Twingo	Renault	50
5	106	Peugeot	45
6	206	Peugeot	60
7	306	Peugeot	80
8	807	Peugeot	110
9	407	Peugeot	130
10	C4	Citroen	85
11	C5	Citroen	100
12	C3	Citroen	45
13	C8	Citroen	115
14	Yaris	Toyota	55
15	Rav4	Toyota	95

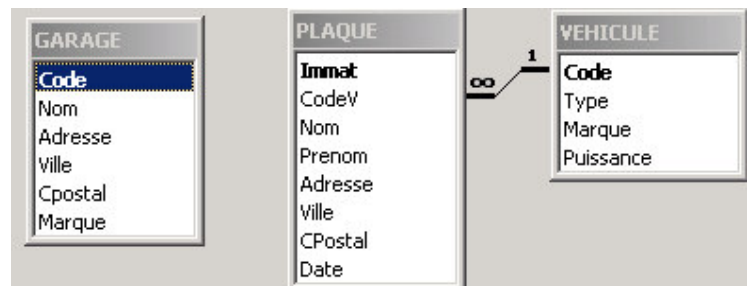




Table GARAGE

Code	Nom	Adresse	Ville	Cpostal	Marque
1	Dupont	36 Route de Paris	Bar Le Duc	55000	Renault
2	Durand	75 Rue de Par là	Nancy	54000	Peugeot
3	Asterix	10 Allée le Ciel	Ailleurs	80100	Renault
4	Obelix	25 Allée du Ciel	Ailleurs	80100	Citroen
5	Le Barde	35 Va en enfer	ICI	81010	Toyota

Table PLAQUE

Immat	CodeV	Nom	Prenom	Adresse	Ville	CPostal	Date
150XY52	1	Lucien	Raymond	36 par ici	Saint Dizier	52000	01/10/2005
200XY55	2	Traban	Fernand	60 rue du lac	Nancy	54000	20/02/2005
300ATX57	3	Tournesol	Luc	80 Avenue de Strasbourg	Metz	57000	18/12/2004
220ACD80	4	Dupond	Louis	35 rue de Moscou	Amiens	80000	20/10/2005
300ATZ81	5	Etienne	Marcel	20 rue de Toulouse	Albi	81990	12/07/2004
400XZ55	6	Durand	Maurice	35 rue de Belfort	Chambley	54120	18/02/2004
20AAA80	7	Lenoir	Louis	70 Rue de Bellefontaine	Amiens	80000	27/05/2005
2875ATX57	8	Leblanc	Lucien	80 Route de Vienne	Albi	81990	12/07/2004

	BTS IG 1 <sup>ère</sup> année ALSI	Chapitre 8 - Cours	
	<i>L'accès aux données</i>		

### 3 Le langage algébrique

C'est un langage théorique conçu par le mathématicien américain Codd en 1970. Il fait appel à des opérations élémentaires qui manipulent les tables ainsi que leurs données.  
Il prépare la conception de requêtes qui seront traduites en SQL.

Il est composé de 2 types d'opérateurs :

- **Relationnels** : Sélection, Projection, Jointure et Division.
- **Ensemblistes** : Union, Intersection, Différence.
  - Le résultat de l'opération est une table.
  - Chaque opération est décomposée.
  - Il n'existe pas de norme de représentation pour représenter ce langage.

Les principales fonctions exploitées sont la projection, la sélection, la jointure et le tri.

#### ***3.1.1 La Projection (Sélection de colonnes)***

Elle permet de sélectionner certains attributs de la relation.  
Aucun doublon n'apparaît.

Ex :

RESULTAT = PROJ (GARAGE , Ville)

Fournit une table composée des Villes de la table GARAGE.

#### ***3.1.2 La Sélection (Sélection de lignes)***

Elle consiste à sélectionner des enregistrements qui vérifient les critères énoncés dans la requête.

Ex :

RESULTAT = SEL (GARAGE , Marque = "Renault")

Fournit une table composée des Garages dont la marque est "Renault"



### 3.1.3 La Jointure

Elle consiste à rapprocher 2 tables ayant un champ commun. On obtient une relation dont les données sont celles obtenues par concaténation des données des 2 tables et qui vérifient le critère de jointure.

Ex :

RESULTAT = JOINT(VEHICULE, GARAGE, GARAGE.Marque = VEHICULE.Marque)

Fournit une table associant à chaque garage de la marque la voiture de la marque correspondante.

G.Code	Nom	Adresse	Ville	Cpostal	G.Marque	V.Code	Type	V.Marque	Puiss.
1	Dupont	36 Route de Paris	Bar Le Duc	55000	Renault	4	Twingo	Renault	50
1	Dupont	36 Route de Paris	Bar Le Duc	55000	Renault	3	Megane	Renault	80
1	Dupont	36 Route de Paris	Bar Le Duc	55000	Renault	2	Clio	Renault	60
1	Dupont	36 Route de Paris	Bar Le Duc	55000	Renault	1	Espace	Renault	100
2	Durand	75 Rue de Par là	Nancy	54000	Peugeot	9	407	Peugeot	130
2	Durand	75 Rue de Par là	Nancy	54000	Peugeot	8	807	Peugeot	110
2	Durand	75 Rue de Par là	Nancy	54000	Peugeot	7	306	Peugeot	80
2	Durand	75 Rue de Par là	Nancy	54000	Peugeot	6	206	Peugeot	60
2	Durand	75 Rue de Par là	Nancy	54000	Peugeot	5	106	Peugeot	45
3	Asterix	10 Allée le Ciel	Ailleurs	80100	Renault	4	Twingo	Renault	50
3	Asterix	10 Allée le Ciel	Ailleurs	80100	Renault	3	Megane	Renault	80
3	Asterix	10 Allée le Ciel	Ailleurs	80100	Renault	2	Clio	Renault	60
3	Asterix	10 Allée le Ciel	Ailleurs	80100	Renault	1	Espace	Renault	100
4	Obelix	25 Allée du Ciel	Ailleurs	80100	Citroen	13	C8	Citroen	115
4	Obelix	25 Allée du Ciel	Ailleurs	80100	Citroen	12	C3	Citroen	45
4	Obelix	25 Allée du Ciel	Ailleurs	80100	Citroen	11	C5	Citroen	100
4	Obelix	25 Allée du Ciel	Ailleurs	80100	Citroen	10	C4	Citroen	85
5	Le Barde	35 Va en enfer	ICI	81010	Toyota	15	Rav4	Toyota	95
5	Le Barde	35 Va en enfer	ICI	81010	Toyota	14	Yaris	Toyota	55

### 3.1.4 Tri

Tri d'une relation suivant un champ par ordre croissant (ASC) ou décroissant (DESC).

Ex :

RESULTAT = TRI (GARAGE, Cpostal, Ville)

Fournit une table des Garages triée par code postal puis Ville

### 3.1.5 Les autres opérations

Toutes les opérations permettant de manipuler des données existent en langage algébrique :

- Union, Intersection, Division
- Les fonctions d'agrégats : Somme, Moyenne, Max ...



### 3.1.6 Compositions des opérations

Il est possible de composer les différentes opérations afin de créer une requête. Chaque opération créant une table intermédiaire, cette table sera prise en compte pour l'opération suivante :

Ex :

TABLE1 = JOINT (VEHICULE , PLAQUE, Code = CodeV)

TABLE2 = PROJ (TABLE1, Nom, Prenom, Date, Type, Marque, Immat)

RESULTAT = TRI (TABLE2,Date)

Fournit la liste triée par date des Immatriculations ...

Nom	Prenom	Date	Type	MArque	Immat
Durand	Maurice	18/02/2004	206	Peugeot	400XZ55
Leblanc	Lucien	12/07/2004	807	Peugeot	2875ATX57
Etienne	Marcel	12/07/2004	106	Peugeot	300ATZ81
Tournesol	Luc	18/12/2004	Megane	Renault	300ATX57
Traban	Fernand	20/02/2005	Clio	Renault	200XY55
Lenoir	Louis	27/05/2005	306	Peugeot	20AAA80
Lucien	Raymond	01/10/2005	Espace	Renault	150XY52
Dupond	Louis	20/10/2005	Twingo	Renault	220ACD80

## 4 Le langage SQL

### 4.1 *Présentation*

Le langage algébrique est un pseudo-langage qui permet de formuler des requêtes sur une Base de Données, mais ce pseudo-langage n'est pas un langage compréhensible par l'ordinateur.

Le SQL (Structured Query Language) est un langage informatique compris par la majorité des bases de données. Grâce à ce langage, on pourra comme en algèbre relationnelle interroger une BD (formuler des requêtes) mais aussi créer, modifier ou supprimer des données. On pourra même gérer la sécurité de la BD.

Le SQL est donc à la fois ::

- Un langage d'interrogation de données (LID)
- Un langage de manipulation de données (LMD)
- Un langage de définition de données (LDD)
- Un langage de contrôle des données (LCD)

Ce langage est très répandu dans les SGBD et est pratiquement universel en matière de langage de SGBD. Il est donc très important de le connaître.

Ce langage peut être utilisé de façon interactive, dans un environnement qui interprète chaque commande SQL après sa saisie ou bien incorporé à un programme pour traiter les accès à la base de données.

Les commandes et la syntaxe du langage sont en anglais

Nous verrons cette année le LID, le LDD et le LMD. Le LCD sera vu en 2<sup>ème</sup> année.



Certains langages comme Access proposent des assistants permettant d'effectuer des requêtes. Néanmoins, ces assistants génèrent des requêtes en langage SQL. De plus, certaines requêtes complexes ne peuvent être réalisées à l'aide d'assistants.

#### 4.2 Historique

Développé initialement dans les années 70, SQL n'a été normalisé dans sa première version qu'en 1986. Cette première norme, trop restrictive a été peu suivie et chaque SGBD a développé son propre langage, ce qui rendait difficile le portage d'une application d'une base à une autre. La véritable révolution a eu lieu par l'adoption de la norme SQL2 en 1992. Les principales phases de SQL sont les suivantes :

SQL 1 ou SQL SQL86 - SQL89 représente la référence de base. Cette version développée au départ par IBM présentait :

- Des requêtes compilées puis exécutées depuis un programme d'application.
- Des types de données simples (entiers, réels, chaînes de caractères de taille fixe)
- Des opérations ensemblistes restreintes (UNION).

SQL 2 ou SQL92 est le standard actuel. Il présente comme caractéristiques :

- Des requêtes dynamiques: exécution différée ou immédiate
- Des types de données plus riches (intervalles, dates, chaînes de caractères de taille variable)
- Différents types de jointures:
- Des opérations ensemblistes plus nombreuses : différence, intersection, union
- Le renommage des attributs dans la clause SELECT

SQL3 (en cours de normalisation ) présente comme modifications :

- SQL devient un langage de programmation :
- Des extensions orientées-objet
- ...

#### 4.3 Les conventions

Ces conventions ne sont que des habitudes, néanmoins sont importantes afin de bien lire les requêtes :

- Les "termes" SQL s'écrivent en majuscule
- Les noms des tables s'écrivent en majuscule
- Les noms des champs : 1<sup>ère</sup> lettre en majuscule, le reste en minuscule
- Le passage à la ligne dans une requête permet de différencier les opérations

Ex :

```
SELECT PLAQUE.Nom, Prenom
      FROM VEHICULE , GARAGE, PLAQUE
      WHERE VEHICULE.Marque=GARAGE.Marque
      AND VEHICULE.Code=PLAQUE.CodeV
      AND Puissance>=100
      AND GARAGE.Ville="Ailleurs";
```

est plus lisible que

```
SELECT PLAQUE.Nom, Prenom FROM VEHICULE , GARAGE, PLAQUE WHERE
VEHICULE.Marque=GARAGE.Marque AND VEHICULE.Code=PLAQUE.CodeV AND
Puissance>=100 AND GARAGE.Ville="Ailleurs";
```



#### 4.4 *Le langage d'interrogation des données LID*

La structure de base d'une interrogation est formée des 3 clauses suivantes :

**SELECT** liste champ(s)  
**FROM** liste table(s)  
**WHERE** condition(s)  
;

- La clause **SELECT** désigne la liste des champs devant figurer dans le résultat.
- La clause **FROM** indique le nom de la ou des table(s) ou vue(s) impliquée(s) dans l'interrogation.
- La clause **WHERE** correspond aux conditions de sélection des champs.
- La requête se termine par un point virgule.
- Chaque nom de champ ou de table est séparé par une virgule.

##### 4.4.1 *La clause SELECT*

**SELECT** [ALL, DISTINCT, UNIQUE] liste champs  
**FROM** nom table  
**WHERE** conditions

- La liste des champs peut comporter des noms de champs, des fonctions SQL prédéfinies, des expressions arithmétiques (/,\*,-,+).
- **DISTINCT**, **UNIQUE** signifie que les enregistrements en double dans le résultat sont supprimés.
- **ALL**, par défaut, renvoie tous les enregistrements, même doubles.
- Le symbole \* à la place de la liste des champs sélectionne tous les champs.

##### Exemple :

```
SELECT Nom, Adresse FROM GARAGE ;
```

```
SELECT DISTINCT Ville FROM GARAGE ;
```

```
SELECT * FROM GARAGE ;
```

##### 4.4.2 *La clause WHERE*

La clause **WHERE** permet de sélectionner des enregistrements de la table selon des critères bien précis. Dans cette clause, on peut utiliser des prédicats :

- Le prédicat de comparaison : =, <, >, <>, <=, >=
- Le prédicat **(NOT) BETWEEN** :

```
SELECT Nom, Prenom  
FROM PLAQUE
```



WHERE Date BETWEEN #01/01/2004# AND #31/12/2004#;  
Liste des plaques obtenues en 2004

Attention, la syntaxe ci-dessus est propre à ACCESS.

- Le prédicat **(NOT) IN** :

```
SELECT Type  
FROM VEHICULE  
WHERE Puissance IN (60, 80, 100);
```

Les Types de Véhicules dont la puissance est 60 80 et 100 Cv

- Le prédicat **(NOT) LIKE** :

```
SELECT Nom, Prenom  
FROM PLAQUE  
WHERE Cpostal LIKE "5%";
```

Nom et Prenom des propriétaires de voiture dont le code postal commence par 5

Remarque : % remplace n'importe quel caractère, \_ un caractère particulier.  
Attention, sous Access, % → \* et \_ → ?

- Le prédicat **(NOT) NULL** :

```
SELECT Nom, Prenom  
FROM PLAQUE  
WHERE Cpostal IS NOT NULL;
```

Vérification si le code postal est nul

- Le prédicat composé qui mélange tous les prédicats ci-dessus dans une même requête.

```
SELECT Type  
FROM VEHICULE  
WHERE Puissance IN (60, 80, 100)  
AND Marque <> "Renault";
```

#### ***4.4.3 La notation pointée***

Cette notation permet de préciser le nom de la table sur laquelle portent les champs de la requête, et ce surtout en cas de requête sur plusieurs tables quand les champs sont identiques.

Exemple :

```
SELECT VEHICULE.Marque, Nom FROM VEHICULE, GARAGE ....
```





#### 4.4.4 La jointure

Pour réaliser l'interrogation, on a besoin de champs figurant dans plusieurs tables. Il faut donc fusionner les tables soit en utilisant le produit cartésien, soit la jointure.

#### Le produit cartésien :

SELECT Type, Nom FROM VEHICULE, GARAGE ;

Cette requête va associer chaque Type de VEHICULE au Nom du GARAGE :

Type	Nom
Espace	Dupont
Espace	Durand
Espace	Asterix
Espace	Obelix
Espace	Le Barde
Clio	Dupont
Clio	Durand
Clio	Asterix
Clio	Obelix
Clio	Le Barde
Megane	Dupont
Megane	Durand
...	...

#### La jointure :

Le principe de la jointure est de créer un lien entre tables ayant au moins un champs en commun. Les conditions de jointure s'expriment dans la clause Where.

- L'équi-jointure associe un enregistrement de la table T1 à un de la table T2, si on vérifie l'égalité de valeur entre les champs communs aux deux tables.

SELECT Type, VEHICULE.Marque, Nom, Adresse  
FROM VEHICULE, GARAGE  
WHERE VEHICULE.Marque=GARAGE.Marque ;

Type	Marque	Nom	Adresse
Twingo	Renault	Dupont	36 Route de Paris
Megane	Renault	Dupont	36 Route de Paris
Clio	Renault	Dupont	36 Route de Paris
Espace	Renault	Dupont	36 Route de Paris
407	Peugeot	Durand	75 Rue de Par là
807	Peugeot	Durand	75 Rue de Par là
306	Peugeot	Durand	75 Rue de Par là
206	Peugeot	Durand	75 Rue de Par là
106	Peugeot	Durand	75 Rue de Par là
Twingo	Renault	Asterix	10 Allée le Ciel
Megane	Renault	Asterix	10 Allée le Ciel
...	...	...	...



- La théta-jointure est une jointure dont la condition est une comparaison entre deux colonnes appartenant à deux tables différentes qui utilise un autre opérateur que l'égalité.

Ex :

```
SELECT Nom
FROM GARAGE, VEHICULE
WHERE GARAGE.Marque<>VEHICULE.Marque
AND Type="Espace";
```

Les garages commercialisant des véhicules de marque autre que celle du véhicule de type 'espace'

- La jointure de façon générale
  - Dans la clause WHERE on peut trouver une ou plusieurs jointures et un ou plusieurs prédicats de sélection.

Ex :

```
SELECT Nom, Adresse
FROM VEHICULE , GARAGE
WHERE VEHICULE.Marque=GARAGE.Marque
AND Ville="Ailleurs"
AND Type="Espace" ;
```

Cette requête donne le nom et l'adresse du garage traitant les véhicules de type Espace et exerçant à "Ailleurs"

- Une interrogation peut nécessiter plus de deux tables. Dans ce cas il faut que chaque relation soit identifiée dans la jointure.

Ex :

```
SELECT PLAQUE.Nom, Prenom
FROM VEHICULE , GARAGE, PLAQUE
WHERE VEHICULE.Marque=GARAGE.Marque
AND VEHICULE.Code=PLAQUE.CodeV
AND Puissance>=100
AND GARAGE.Ville="Ailleurs";
```

On recherche tous les propriétaires de véhicules > ou égal à 100 Cv, achetés dans un garage d'Ailleurs.

- La jointure entre 2 tables peut mettre en jeu plus d'un attribut commun (clé étrangère double, triple...)

```
SELECT .....
FROM t1, t2
WHERE t1.champ1=t2.champ1
AND t1.champ2=t2.champ2
AND .....
```



- L'auto-jointure

La jointure d'une table avec elle-même est nécessaire lorsqu'une interrogation a besoin de travailler sur plusieurs occurrences de la même table.

Le problème est que l'on travaille sur la même table et donc on aura à utiliser les mêmes noms de champs. Il faut donc une méthode pour différencier les différents champs. La solution est de nommer les tables de façon différente. Ce nommage n'existe que pendant la requête.

Ex : Quels garages commercialisent la même marque que le garage 'dupont' ?

```
SELECT G2.Nom, G2.Adresse
FROM GARAGE G1, GARAGE G2
WHERE G1.Marque=G2.Marque
AND G1.Nom="Dupont" ;
```

L'inconvénient de la requête ci-dessus est qu'elle liste également le garage Dupont qui n'est pas demandé. Il faut donc exclure le garage Dupont :

```
SELECT G2.Nom, G2.Adresse
FROM GARAGE G1, GARAGE G2
WHERE G1.Marque=G2.Marque
AND G1.Nom="Dupont"
AND G2.Nom <> "Dupont" ;
```

#### ***4.4.5 Les Sous interrogations ou Select imbriqués***

Une sous interrogation est une commande SELECT qui est située à l'intérieur de la clause WHERE d'un autre SELECT. Elle peut prendre l'une des formes suivantes :

#### **WHERE [expr] [opérateur] {ALL|ANY} (commande SELECT)**

On compare une expression au résultat retourné par la commande SELECT. Ce résultat devant être du même type que celui de l'expression de la clause WHERE.

**ALL** : la condition de comparaison est vérifiée pour toutes les valeurs retournées par le SELECT

**ANY** : la condition de comparaison est vérifiée pour au moins une des valeurs retournées par le SELECT

Les opérateurs utilisables sont les opérateurs arithmétiques vus en début de cours.

Ex :

```
SELECT nom, prenom
FROM VEHICULE V, PLAQUE P
WHERE V.Code=P.CodeV
AND Puissance > ALL
(SELECT Puissance FROM VEHICULE WHERE Marque="Renault") ;
```

Liste des personnes possédant un véhicule dont la puissance est supérieure à celle de tous les véhicules Renault.



**WHERE [expr] (NOT) IN (commande SELECT)**

Dans ce cas, le IN est équivalent à ANY, c'est-à-dire que la condition est vraie pour au moins une des valeurs retournées par le SELECT.

Ex :

```
SELECT Nom, prenom
FROM PLAQUE , VEHICULE
WHERE Code = CodeV
AND Type IN
      (SELECT Type FROM VEHICULE WHERE Marque="renault") ;
```

Liste des personnes ayant au moins un véhicule de marque 'Renault'

On aurait pu écrire .....AND type = ANY (SELECT.....

**WHERE (NOT) EXISTS (commande SELECT)**

Le prédicat WHERE est vrai, si la sous interrogation retourne au moins une ligne.

Ex :

```
SELECT Code, Type
FROM VEHICULE
WHERE NOT EXISTS
      (SELECT * FROM PLAQUE WHERE CodeV = Code) ;
```

La requête renvoie la liste des véhicules qui ne sont pas immatriculés

**WHERE (NOT) UNIQUE (commande SELECT)**

Le prédicat WHERE est vrai, si la sous interrogation retourne une ligne et une seule.

REM non implanté sous Access

***4.4.6 Les opérateurs UNION et INTERSECT***

L'opérateur UNION est le mot-clé placé entre deux commandes SELECT pour réaliser l'union ensembliste de deux tables.

Ex :

```
SELECT Type, Marque
FROM VEHICULE V
WHERE Puissance >100
UNION
SELECT Type, Marque
FROM VEHICULE V
WHERE Puissance <50;
```

Liste des voitures de plus de 100 Cv et de moins de 50 Cv



Important : pour réaliser l'union, les champs sélectionnés dans les 2 requêtes doivent être compatibles.

Si les tables unies sont des sous-ensembles de la même table, l'opérateur OR placé entre les deux conditions était identique.

```
SELECT Type, Marque
FROM VEHICULE V
WHERE Puissance >100
OR Puissance <50;
```

Le principe est identique pour l'intersection.

#### 4.4.7 Les fonctions d'agrégation:

Les fonctions d'agrégation retournent une valeur obtenue par l'application d'une fonction au groupe de valeurs sélectionné par la clause WHERE d'une commande SELECT.

Ex :

```
SELECT AVG(puissance)
FROM VEHICULE
WHERE Marque="Renault";
```

Retourne la moyenne de la puissance des véhicules de marque 'Renault'

```
SELECT MAX(puissance) ...
```

Retourne la puissance maximum ...

Fonctions d'agrégation :

FONCTIONS	VALEUR DE RETOUR
AVG([DISTINCT]x )	Moyenne de toutes les valeurs de x
SUM([DISTINCT]x )	Somme de toutes les valeurs de x
MAX(x)	Valeur max de x
MIN(x)	Valeur min de x
COUNT(*)	Nombre de ligne
COUNT([DISTINCT] x)	Nombre de valeur de x (unique si DISTINCT)

**Rem** : Distinct dans la fonction d'agrégat ne fonctionne pas sous Access

#### 4.4.8 Les clauses GROUP BY ET HAVING:

On utilise la clause GROUP BY pour produire une ligne résultat pour chaque groupe d'enregistrements issu d'une ou plusieurs tables.

Ex :

```
SELECT Marque, count(*), AVG(Puissance)
FROM VEHICULE
GROUP BY Marque ;
```

Fournit pour chaque marque le nombre de types de véhicules et la moyenne de leur puissance

Marque	Expr1001	Expr1002
Citroen	4	86,25
Peugeot	5	85
Renault	4	72,5
Toyota	2	75



La clause HAVING permet de définir une condition devant être vérifiée par le groupe.

Ex :

```
SELECT Marque, count(*), MAX(Puissance)
FROM VEHICULE
GROUP BY Marque
HAVING count(*) > 2 ;
```

Fournit pour chaque marque commercialisant plus de 2 types de véhicules la puissance maxi.

Marque	Expr1001	Expr1002
Citroen	4	115
Peugeot	5	130
Renault	4	100

#### **4.4.9 La clause ORDER BY :**

On l'utilise pour trier les résultats des interrogations sur un ou plusieurs champs figurant dans la clause SELECT.

ORDER BY champ [ASC|DESC]

Ex :

```
SELECT Marque, Type, Puissance
FROM VEHICULE
ORDER BY Puissance DESC , Marque
```

Liste des véhicules triés par puissance (décroissante) puis par marque.

Marque	Type	Puissance
Peugeot	407	130
Citroen	C8	115
Peugeot	807	110
Citroen	C5	100
Renault	Espace	100
Toyota	Rav4	95
...	...	...

Il est possible également de désigner les champs de l'ORDER BY par leur ordre dans le select :

```
SELECT Marque, Type, Puissance
FROM VEHICULE
ORDER BY 3 DESC, 2
```



#### 4.4.10 Les pseudonymes ou alias

Ils permettent de renommer des champs ou des tables dans une commande SELECT. Ceci est intéressant dans les cas suivants :

- Remplacer un nom de champ peu significatif.
- Abréger un nom de table ou de champ.
- Distinguer deux noms de champ ou de tables identiques (vu avec l'auto-jointure)..

Ex :

```
SELECT Marque, count(*) AS Quantite, AVG(Puissance) AS Moyenne  
FROM VEHICULE V  
GROUP BY Marque ;
```

Marque	Quantite	Moyenne
Citroen	4	86,25
Peugeot	5	85
Renault	4	72,5
Toyota	2	75

Dans cette requête, la table VEHICULE a comme Alias V. Pour les tables, le AS est facultatif.

#### 4.4.11 Les fonctions

On peut, dans la clause SELECT appliquer des fonctions aux champs, comme par exemple des fonctions de manipulation de texte.

Ex :

```
SELECT UPPER(Nom), LOWER(Prenom)  
FROM PLAQUE  
ORDER BY 1 ASC ;
```

Retourne le nom en majuscule et le prénom en minuscule des propriétaires de véhicules.

**REM :** Sous Access, les mots clés sont UCASE et LCASE.

Il existe des fonctions numériques permettant notamment de calculer les cosinus, sinus, tangente, valeur absolue, racine carrée....

#### Les fonctions sur les dates :

Year(date) donne l'année  
Month (date) donne le mois (1 → 12)  
Day (date) donne le jour (1 → 31)



#### 4.4.12 Les champs calculés

On peut dans la clause SELECT, effectuer des calculs entre les champs d'une ou plusieurs tables.

Ex :

```
SELECT Type, Puissance/10 AS DIVISE  
FROM VEHICULE;
```

Calcule la puissance divisée par dix et l'affiche dans un champ DIVISE

Type	DIVISE
Espace	10
Clio	6
Megane	8
Twingo	5

Il est possible de la même façon de calculer un prix TTC, un montant de TVA ...

De la même manière, on peut concaténer des champs dans un seul :

```
SELECT Nom || ' et ' || Prenom AS personne  
FROM PLAQUE
```

Les champs Nom et Prénom seront concaténés avec la chaîne ' et ' entre les deux.

**REM :** Sous Access, le caractère de concaténation est &

De même sous Access, on utilise la double cote " ( simple cote sur les autres SGBD) pour les chaînes de caractères.

## 5 Exercices

1. Liste (immat, code, type) des voitures Renault immatriculées en 2004
2. Liste des clients (nom, prenom) résidant dans la même ville que les garagistes (donnez le nom des garagistes)
3. liste (type marque puissance) des véhicules dont la puissance est supérieur ou égal à la moyenne de la puissance des véhicules
4. Liste des véhicules (Type) dont le type commence par une lettre
5. Liste des clients (nom, prenom) qui habitent dans une rue
6. Nombre de véhicules (marque, quantité) par marque immatriculé
7. Liste des véhicules (type marque puissance) trié par puissance décroissante et par marque (ordre alphabétique)
8. Liste des véhicules (Code, type) ayant été achetés le deuxième semestre de 2004
9. Quantité de véhicules (Marque, quantité) par marque trié par quantité croissante
10. Listez les clients (Nom, prénom, type de voiture) qui sont dans la même Ville que Dupond Louis
11. Donnez la puissance moyenne par marque
12. Donnez la liste des garages (Nom, Adresse, Marque) qui n'ont aucune voiture immatriculée