Sommaire

1 Présentation

1.1 Introduction

Dès le début de l'informatique, l'idée était de construire des systèmes permettant d'effectuer des calculs mathématiques complexes (résolutions d'équations, calcul matriciel,...).
Aujourd'hui, la tendance est la gestion de grandes quantités d'informations. Un des soucis majeur est et a été de gérer et de traiter ces informations de façon simple. Dans ce but, une méthode standard a été mise au point pour les utilisateurs avancés (ou les informaticiens) : le langage SQL.

1.2 Le langage SQL

1.2.1 Présentation

Le SQL (Sructured Query Language) est un langage informatique compris par la majorité des bases de données. Grâce à ce langage, on peut interroger une BDD (formuler des requêtes) mais aussi créer, modifier ou supprimer des données. On peut même gérer la sécurité de la BDD.

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) (qui n'est pas l'objet de ce cours)

Ce langage est pratiquement universel, et les rares acteurs du domaine ne l'implémentant pas sont pénalisés. Il est donc 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 afin d'automatiser et de masquer les accès à la base de données.

Les commandes et la syntaxe du langage sont en anglais

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

1.2.2 Historique

Développé initialement dans les années 70, SQL n'a été normalisé dans sa premiére version qu'en 1986. Cette premiére norme, trop restrictive a été peu suivi 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

1.2.3 Les conventions

Ces conventions ne sont que des habitudes, néanmoins elles sont importantes afin de rendre lisible les requêtes :

- Les "termes" SQL s'écrivent en majuscule
- Les noms des tables s'écrivent en majuscule
- Les noms des champs : 1ére lettre en majuscule, le reste en minuscule. Attention, un nom ne comprend pas de caractères spéciaux, pas d'accent et pas d'espace. Seul le caractère _ est autorisé

- 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";
```

2 Le modèle de donnée utilisé

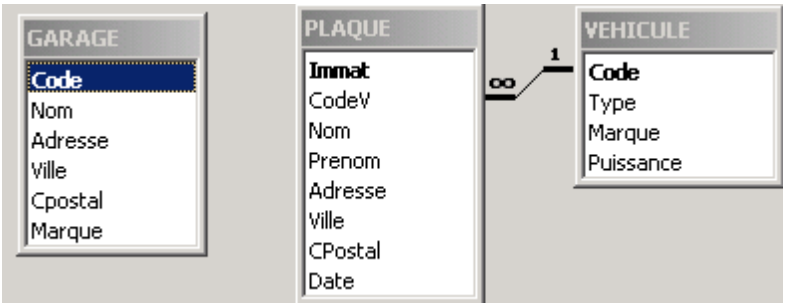
Le modèle de donnée correspond à la gestion de voitures. Une liste de voiture est ou a été disponible à la vente.
Les voitures faisant l'objet d'une vente sont alors immatriculées et appartiennent à un client.
D'autre part, des garages sont spécialisés dans une marque, et normalement ne s'occupent que des véhicules de leur marque.

Je vous propose le modèle relationnel suivant qui n'est volontairement pas optimisé, ainsi que les données correspondantes.

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

3 Le langage d'interrogation des données LID

3.1 Présentation

Ce langage est un des plus utilisé. Il permet l'exploitation de la base de données.

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. On parle également des champs projetés.
- La clause FROM indique le nom de la ou des table(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.

3.2 La clause SELECT

SELECT [DISTINCT] 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** (ou UNIQUE) signifie que les enregistrements en double dans le résultat sont supprimés. Par défaut, tous les enregistrements même en double sont renvoyés.
- Le symbole ***** à la place de la liste des champs sélectionne tous les champs de la table.

Exemple :

SELECT Nom, Adresse FROM GARAGE ;

SELECT DISTINCT Ville FROM GARAGE ;

SELECT * FROM GARAGE ;

3.3 La clause WHERE

La clause WHERE permet de sélectionner des enregistrements de la table selon des critères bien précis. Cette sélection s'applique sur chacun des enregistrements de la ou des tables. Le but est de filtrer un certain nombre d'enregistrement à partir de la valeur d'un ou de plusieurs attribut des tables. Dans cette clause, on peut utiliser des prédicats :

- Le prédicat de comparaison : **=, <, >, <>, <=, >=**
- Le prédicat **BETWEEN** (ou NOT BETWEEN):
    - Attention, pour comparer un champs de type chaîne de caractère à une valeur, il faut encadrer celle-ci par des apostrophes (apostrophes double sous Access)
    - Pour une valeur entière, la valeur est donnée sans encadrement
    - Le séparateur décimal est le point .
    - Une date est encadrée d'une apostrophe dans le format de la base (français ou américain). Sous Access, le caractère d'encadrement est le #

Exemple :

- Le prédicat de **comparaison** :

```
SELECT Nom, Prenom
    FROM PLAQUE
    WHERE Date > '01/01/2004' ;
```

Liste des voitures immatriculées après le 1er janvier 2004

- Le prédicat **BETWEEN**:

Ce prédicat permet de tester si un attribut se trouve entre deux valeurs.

```
SELECT Nom, Prenom
    FROM PLAQUE
    WHERE Date BETWEEN '01/01/2004' AND '31/12/2004';
```

Liste des voitures immatriculées en 2004

- 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** :

Ce prédicat est utilisé sur un attribut chaîne de caractère. Il permet de vérifier l'affirmation ressemble à. Il est possible en conjonction à une valeur d'utiliser des caractères joker :

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

Nom et Prenom des propriétaires de voiture dont le code postal commence par 5 et continue par n'importe quoi.

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;
```

Liste des noms, prénoms de la table plaque où le code postal est renseigné.

- 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';
```

Liste des véhicules (Type dont la puissance est 60, 80 ou 100 CV et dont la marque est Renault

3.4 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 le nom des champs sont identiques.

Exemple :

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

Ici il y a ambiguïté sur le champs marque puisqu'il existe dans les deux tables. La notation pointée est donc obligatoire

3.5 La jointure

Pour réaliser l'interrogation, on a souvent 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 champ en commun. Les conditions de jointure s'expriment dans la clause WHERE. La jointure est souvent faite entre clé primaire et clé étrangère. Le principe de la jointure est de vérifier l'égalité de valeur entre des champs commun de 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
…	…	…	…

```
SELECT Immat, Type, Nom
        FROM VEHICULE, PLAQUE
        WHERE Code = CodeV
```

Cette requête va associer au nom de chaque propriétaire de voiture le type de la voiture ainsi que l'immatriculation correspondante.

- 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.

3.6 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] (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.

Les opérateurs utilisables sont les opérateurs arithmétiques vus précédemment.

Ex :

```
SELECT nom, prenom
        FROM VEHICULE , PLAQUE
        WHERE Code=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, 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'

3.7 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 …

La clause distinct utilisée dans une fonction d'agrégat permet de ne prendre en compte que les valeurs différentes de l'attribut spécifié.

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

Rem : Distinct dans la fonction d'agrégat ne fonctionne pas sous Access.
Le mot clé DISTINCT est facultatif. S'il est utilisé, cela signifie que seules les valeurs différentes seront prises en compte.

3.8 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. On parle également de regroupement.

Ex :

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

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

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

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

Attention, la clause HAVING s'applique sur une fonction d'agrégat. Elle intervient après un group BY puisqu'elle amène une restriction sur les champs issus du regroupement

De même le GROUP BY s'applique après le WHERE

3.9 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
```

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

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

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
```

Attention, il n'y a qu'une seule fois le mot clé ORDER BY

3.10 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
    GROUP BY Marque ;
```

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

Dans cette requête, le résultat des fonctions d'agrégat a été renommé :

- Quantite pour le nombre de véhicules concerné
- Moyenne pour la moyenne de la puissance des véhicules concernés

Les Alias sur les tables sont surtout utilisés dans le cas de requêtes multi-tables afin de simplifier l'écriture.

Ex :

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

Dans cette requête, la table VEHICULE a comme Alias V, la table GARAGE G. Pour les tables, le AS est facultatif. Attention, une fois que les tables sont renommés, les noms de base n'existent plus dans la requête.

3.11 Les fonctions

On peut, dans la clause SELECT, dans la clause WHERE appliquer des fonctions aux champs, comme par exemple des fonctions de manipulation de texte, d'extraction de dates ....

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 à 12)
Day (date) donne le jour (1 à 31)

3.12 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
        WHERE Marque = 'Renault';
```

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 &