SQL, a beginner's guide

1 Le programme

On se limite volontairement à une description applicative des bases de données en langage SQL. Il s'agit de permettre d'interroger une base présentant des données à travers plusieurs relations. On ne présente pas l'algèbre relationnelle ni le calcul relationnel.

Notions	**Commentaires**
Vocabulaire des bases de données : tables ou relations, attributs ou colonnes, domaine, schéma de tables, enregistrements ou lignes, types de données.	On présente ces concepts à travers de nombreux exemples. On s'en tient à une notion sommaire de domaine : entier, flottant, chaîne ; aucune considération quant aux types des moteurs SQL n'est au programme. Aucune notion relative à la représentation des dates n'est au programme ; en tant que de besoin on s'appuie sur des types numériques ou chaîne pour lesquels la relation d'ordre coïncide avec l'écoulement du temps. Toute notion relative aux collations est hors programme ; on se place dans l'hypothèse que la relation d'ordre correspond à l'ordre lexicographique usuel. `NULL` est hors programme.
Clé primaire.	Une clé primaire n'est pas forcément associée à un unique attribut même si c'est le cas le plus fréquent. La notion d'index est hors programme.
Entités et associations, clé étrangère.	On s'intéresse au modèle entité–association au travers de cas concrets d'associations $1-1, 1-*, *-*$. Séparation d'une association $*-*$ en deux associations $1-*$. L'utilisation de clés primaires et de clés étrangères permet de traduire en SQL les associations $1-1$ et $1-*$.
Requêtes `SELECT` avec simple clause `WHERE` (sélection), projection, renommage `AS`. Utilisation des mots-clés `DISTINCT`, `LIMIT`, `OFFSET`, `ORDER BY`.	Les opérateurs au programme sont `+`, `-`, `*`, `/` (on passe outre les subtilités liées à la division entière ou flottante), `=`, `<>`, `<`, `<=`, `>`, `>=`, `AND`, `OR`, `NOT`.
Opérateurs ensemblistes `UNION`, `INTERSECT` et `EXCEPT`, produit cartésien.	
Jointures internes T_1 `JOIN` T_2 ... `JOIN` T_n `ON` ϕ. Autojointure.	On présente les jointures en lien avec la notion de relations entre tables. On se limite aux équi-jointures : ϕ est une conjonction d'égalités.
Agrégation avec les fonctions `MIN`, `MAX`, `SUM`, `AVG` et `COUNT`, y compris avec `GROUP BY`.	Pour la mise en œuvre des agrégats, on s'en tient à la norme SQL99. On présente quelques exemples de requêtes imbriquées.
Filtrage des agrégats avec `HAVING`.	On marque la différence entre `WHERE` et `HAVING` sur des exemples.
Mise en œuvre	
La création de tables et la suppression de tables au travers du langage SQL sont hors programme. La mise en ?uvre effective se fait au travers d'un logiciel permettant d'interroger une base de données à l'aide de requêtes SQL. Récupérer le résultat d'une requête à partir d'un programme n'est pas un objectif. Même si aucun formalisme graphique précis n'est au programme, on peut décrire les entités et les associations qui les lient au travers de diagrammes sagittaux informels. Sont hors programme : la notion de modèle logique *vs* physique, les bases de données non relationnelles, les méthodes de modélisation de base, les fragments DDL, TCL et ACL du langage SQL, les transactions, l'optimisation de requêtes par l'algèbre relationnelle.	

2 Introduction

• SQL (Structured Query Language) est un langage de programmation qui permet de
On marque la différence entre WHERE et HAVING sur des exemples.

— créer,
— administrer,
— modifier/structurer, et
— requeter (c'est à dire effectuer des demandes sur une base de données)

des bases de données relationnelles.

Une ***base de données relationnelle*** est un ensemble de tables (ou relations).

Une ***table*** est un tableau ***bi-dimentionnel*** :
— les colonnes sont les attributs de la table (ou ***champs***)
— les lignes sont les entrées (ou ***enregistrements***)
comme par exemple :

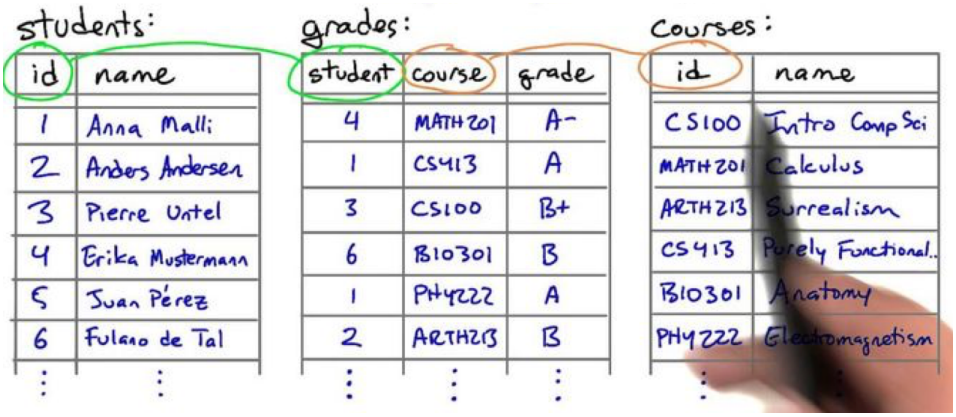

FIGURE 1 – Exemple de base de données

Dans une base de données relationnelle les données sont « décomposées » en tables selon une procédure de *normalisation*, afin de séparer les données qui évoluent a priori de manière indépendante. Par exemple les étudiants et les cours d'une université, les produits que vend une entreprise et ses clients, ou encore les types de moteurs et les types de véhicules d'un constructeur automobile, etc.

SQL est le langage le plus universellement répandu dans les systèmes de gestion de bases de données (Data Base Management Systems) relationelles.

Il existe une variété de langages SQL, les plus répandues et partageant toutes les commandes essentielles sont présentés à la figure 2

FIGURE 2 – Langages SQL

• Cours et tutoriel en ligne

— https://sqlbolt.com/
— https://sql.sh/

- **Quelques sources de données**
  - Le catalogue des données ouvertes du gouvernement OpenDataGouv : https://www.data.gouv.fr/fr/datasets/catalogue-des-donnees-de-data-gouv-fr/
  - les données ouvertes de la Banque mondiale : https://donnees.banquemondiale.org/
  - les données de la comission européenne : https://ec.europa.eu/eurostat/data/database
  - les données de la commission européenne : https://ec.europa.eu/eurostat/data/database
  - les données de la banque de France : http://webstat.banque-france.fr/fr/#/home
  - Relational Dataset Repository : https://relational.fit.cvut.cz/search

- **Convertisseurs vers SQL**

  Ces liens permettent de convertir des fichiers xls, csv en SQL.
  - SQLizer : https://sqlizer.io/
  - Code Beautify : https://codebeautify.org/csv-to-sql-converter

- **Théorie des bases de données**

  Les bases de données relationnelles reposent sur ***l'algèbre relationnelle*** et le ***théorème de Codd*** (E. F. Codd, "Relational completeness of data base sublanguages", Database Systems, Prentice-Hall, 1972, pp. 65-98).

3 Environnements de développement

Pour exécuter des commandes SQL, vous aurez typiquement trois options :
- en ligne de commande, en vous connectant à un serveur SQL (local ou remote)
- en utilisant un logiciel d'administration et de gestion SQL. Pour l'installation, sur le lien https://dev.mysql.com/downloads/mysql/, télécharger "MySQL Installer". Exécuter ce dernier programme et dans le menu des composantes à installer, choisissez "MySQL server" et "MySQL Workbench"
- en python, avec la librairie mysql, pour automatiser ces commandes.

4 Les requêtes élémentaires SQL

4.1 Généralités

La connection à serveur mySQL se fait :
- en ligne de commande :
  `mysql -u user_name -p`
  où
  `user_name`
  est le nom de l'utilisateur, puis en entrant son mot de passe.
- avec workbench : en lançant l'application, puis en choisissant le serveur auquel se connecter.

Le langage SQL est case insensitive mais il est d'usage d'écrire toute instruction SQL en majuscule.

Toute instruction SQL doit se terminer par un point virgule ;.

Pour noter des commentaire dans le code : #.

La commande `SHOW DATABASES` ; liste les tables présentes dans la base de donnée.

La commande `USE nom_base` ; (ou `CONNECT nom_base` ;) connecte à la table `nom_base`. Pour se déconnecter : `RESETCONNECTION` ;.

Lorsque l'on est connecté à une base de donnée, la commande `SHOW TABLES;` liste les tables de la base de donnée, et la commande `DESCRIBE nom_table` ; liste les caractéristiques de la table `nom_table` (nom et types des colonnes, clés, etc.)

Une table `nom_table` dans une base `nom_base` est reconnue par SQL comme `nom_base.nom_table`.

```
# liste les tables présentes dans la base de donnée
SHOW DATABASES;


# connection à la bdd trucmuche
```

```
USE trucmuche_db;

# liste des tables de la bdd
SHOW TABLES;

# caractéristiques de la table Album
DESCRIBE Album;
```

4.2 Requêtes SELECT

La commande fondamentale `SELECT` réalise une extraction de données à partir de tables d'une base de donnée.

• Extraction de toutes les colonnes d'une table

```
SELECT
   *
FROM
   nom_table
```

• Extraction de colonnes spécifiques

```
SELECT
   colonne_1,
   colonne_2,
   ...
FROM
   nom_table
```

• Extractions de colonnes d'une table, avec `LIMIT` et `OFFSET`

— La commande `LIMIT n` permet de limiter le nombre de lignes extraites aux n premières de la table,
— la commande `OFFSET m` permet de décaler l'extraction pour obtenir les lignes $m+1$ à $m+n$.

```
# extraction des 10 première lignes
SELECT
   colonne_1,
   colonne_2,
   ...
FROM
   nom_table LIMIT 10
```

```
# extraction des lignes 4 à 13
SELECT
   colonne_1,
   colonne_2,
   ...
FROM
   nom_table LIMIT 10 OFFSET 3
```

- **Requêtes avec conditions de filtre `WHERE`**

```
SELECT
   colonne_1,
   colonne_2,
   ...
FROM
   nom_table
WHERE
   condition_1
   AND/OR condition_2
   AND/OR ...;
```

où les conditions peuvent s'exprimer avec les opérateurs du tableau suivant.

Opérateur	Condition	Exemple
`=`	Comparaison exacte d'une chaîne de caractères. Sensible à la casse. (**simple égal !**)	`nom_col = "France"`
`!=` ou `<>`	Différent de, sensible à la casse.	`nom_col != "France"`
`LIKE`	Comparaison exacte d'une chaîne de caractères. Insensible à la casse.	`nom_col LIKE "france"`
`NOT LIKE`	Insensible à la casse.	`nom_col NOT LIKE "France"`
`%`	Utilisé n'importe où dans une chaîne pour comparer à une suite de caractères. Seulement avec `LIKE` et `NOT LIKE`	`nom_col LIKE "%Fr%"` Prendra en considération `"France"` mais aussi `"refroidi"`
`_`	Utilisé n'importe où dans une chaîne. Seulement avec `LIKE` et `NOT LIKE`	`nom_col LIKE "Fra_"` Prendra `France` mais pas `Fra`
`IN(...)`	La chaîne de caractères existe dans la liste	`nom_col IN("France","Belgique"`
`NOT IN(...)`		`nom_col NOT IN("France","Belgique"`

- **Requêtes triées avec ORDER BY**

```
SELECT
   colonne_1,
   colonne_2,
   ...
FROM
   nom_table
WHERE
   condition(s)
ORDER BY
   colonne ASC/DESC
```

où `ASC` (resp. `DESC`) classe par ordre numérique/lexicographique croissant (resp. décroissant).

- **Fonctions d'agrégation et expressions**

```
SELECT
AGG_FUNC(colonne_ou_expression) AS description, ...
FROM
```

```
   nom_table
WHERE
   condition(s);
```

où les fonctions d'agrégation les plus courantes sont présentées dans le tableau suivant et où une expression est une opération effectuée sur la colonne, par exemple `colonne_radian * 180 / 3.14 AS degre`.

Fonction	Description
COUNT(*)	Donne le nombre de lignes totales
COUNT(Colonne)	Donne le nombre de lignes dans la colonne spécifiée avec une valeur non NULL.
MIN(Colonne)	Donne la plus petite valeur numérique de la colonne spécifiée
MAX(Colonne)	Donne la plus grande valeur numérique de la colonne spécifiée
AVG(Colonne)	Donne la moyenne des éléments de la colonne spécifiée
SUM(Colonne)	Donne la somme des éléments de la colonne spécifiée

Dans l'exemple précédent la fonction d'agrégation s'applique à toutes les lignes de la table, mais on peut également l'appliquer à des groupes de lignes sélectionnés par la commande GROUP BY :

```
SELECT
   AGG_FUNC(colonne_ou_expression) AS description
FROM
   nom_table
WHERE
   condition(s)
GROUP BY
   colonne
```

dans ce cas la fonction d'agrégation s'applique à chaque groupe de lignes ayant la même valeur dans la colonne spécifiée.

On peut finalement appliquer des filtres/tris supplémentaires sur les groupes avec la commande `HAVING`.

```
SELECT
   colonne,
   AGG_FUNC(colonne_ou_expression) AS alias_du_résultat_agrégé,
   ...
FROM
   nom_table
WHERE
   condition(s)
GROUP BY
   column
HAVING
   condition(s)_sur_les_groupes;
```

Remarque. il faut écrire les commandes WHERE, GROUP BY, et HAVING dans cet ordre.

Pour finir voici une requête SELECT prenant en compte ce que nous venons de voir :

```
SELECT
   DISTINCT colonne,
   AGG_FUNC(colonne_ou_expression),
   ...
FROM
   nom_table
WHERE
   condition(s)
GROUP BY
```

```
   colonne
HAVING
   condition(s)_groupes
ORDER BY
   colonne ASC/DESC    LIMIT nb_groupes OFFSET nb_décalage;
```

4.3 Jointures INNER JOIN

Nous avons vu jusqu'à présent des requête sur une seule table.

Dans une bdd relationnelle les données sont décomposées en tables selon une procédure de normalisation, afin de séparer les données qui évoluent de manière indépendante (par exemple les clients d'une entreprise et les produits mis en vente, ou les types de moteurs et les types de véhicules d'un constructeur automobile, etc.).

Des tables qui partagent des informations sur une même entité doivent avoir une clé primaire, `PRIMARY KEY` (typiquement un entier), qui identifie cette entité de manière unique dans la base de donnée. Une opération de jointure permet de combiner les lignes de deux tables différentes à partir d'une clé unique.

Considérons les tables Album et Artist ci-dessous (figure 3) :

AlbumId	Title	ArtistId
1	For Those About To Rock We Salute You	1
2	Balls to the Wall	2
3	Restless and Wild	2
4	Let There Be Rock	1
5	Big Ones	3
6	Jagged Little Pill	4
7	Facelift	5
8	Warner 25 Anos	6
9	Plays Metallica By Four Cellos	7
10	Audioslave	8

ArtistId	Name
1	AC/DC
2	Accept
3	Aerosmith
4	Alanis Morissette
5	Alice In Chains
6	Antônio Carlos Jobim
7	Apocalyptica
8	Audioslave
9	BackBeat
10	Billy Cobham

FIGURE 3 – Tables Album et Artist

Utilisons la clé `ArtistID` pour afficher par artiste (de la table `Artist`) la liste des albums référencés (dans la table Album) :

```
SELECT
   Artist.Name AS nom_artiste,
   Album.Title AS titre_album
FROM
   Artist
INNER JOIN
   Album
      ON Artist.ArtistId=Album.ArtistID
ORDER BY
   nom_artiste,
   titre_album;
```

pour obtenir :

Remarque. Il est fortement recommandé d'écrire les champs complets que l'on souhaite extraire, i.e. de la forme `nom_table.nom_colonne`, et d'utiliser des alias.

Ici par exemple nous avons utilisé `Artist.Name` - et pas `Name` - ainsi que l'alias `nom_artiste`.

```
SELECT
   students.name AS etudiant,
   courses.name AS cours,
   grades.grade AS note
FROM
```

```
   students
INNER JOIN
   grades
      ON students.id=grades.student
INNER JOIN
   courses
      ON grades.course=courses.id
ORDER BY
   etudiant,
   cours;
```

- **Requête** `SELECT` **complète**

```
SELECT
   DISTINCT colonne(s),
   AGG_FUNC(colonne_ou_expression),
   ...
FROM
   table_1
INNER JOIN
   table_2
      ON table_1.colonne = table_2.colonne
WHERE
   condition(s)
GROUP BY
   colonne
HAVING
   condition(s)_groupes
ORDER BY
   colonne ASC/DESC    LIMIT nb_lignes OFFSET nb_décalage;
```