SQL

Juillet 2014

Jérôme GUY

SQL

I. INTRODUCTION

SQL (Structured Query Language) est un langage de

- définition de données (LDD, ou en anglais DDL Data Definition Language),
- manipulation de données (LMD, ou en anglais DML, Data Manipulation Language),
- de contrôle de données (LCD, ou en anglais DCL, Data Control Language),

pour les bases de données relationnelles.

Le modèle relationnel a été inventé par E.F. Codd (Directeur de recherche du centre IBM de San José) en 1970, suite à quoi de nombreux langages ont fait leur apparition :

- IBM Sequel (Structured English Query Language) en 1977
- IBM Sequel/2
- IBM System/R
- IBM DB2

Ce sont ces langages qui ont donné naissance au standard SQL, normalisé en 1986 par l'ANSI pour donner SQL/86. Puis en 1989 la version SQL/89 a été approuvée.

La norme SQL/92 a désormais pour nom SQL 2.

SQL est un langage de définition de données

SQL est un langage de définition de données (LDD), c'est-à-dire qu'il permet de créer des tables dans une base de données relationnelle, ainsi que d'en modifier ou en supprimer.

SQL est un langage de manipulation de données

SQL est un langage de manipulation de données (LMD), cela signifie qu'il permet de sélectionner, insérer, modifier ou supprimer des données dans une table d'une base de données relationnelle.

SQL est un langage de protections d'accès

Il est possible avec SQL de définir des permissions au niveau des utilisateurs d'une base de données.

On parle de DCL (Data Control Language).

Typologie du langage

Il est possible d'inclure des requêtes SQL dans un programme écrit dans un autre langage (en langage C par exemple), ainsi que d'envoyer directement les requêtes SQL telles quelles au SGBD.

Il est possible d'ajouter des commentaires grâce :

- au caractère **--** (deux tirets). Tous les caractères situés après sur la même ligne ne seront pas interprétés
- aux délimiteurs **/* et */**. Tous les caractères compris entre les délimiteurs sont considérés comme des commentaires

Sensibilité à la casse

Le langage SQL n'est pas sensible à la casse, on peut aussi bien écrire les instructions en minuscules qu'en majuscule.

Toutefois, cette insensibilité à la casse n'est que partielle dans la mesure où la différenciation entre minuscules et majuscules existe au niveau des identificateurs d'objets.

Par exemple lors d’une requête la recherche est sensible à la casse !

II. Création d'une vue

Une vue est une table virtuelle, les données ne sont pas stockées dans une table de la base de données

Il est possible de rassembler des informations provenant de plusieurs tables.

On parle de "vue" car il s'agit simplement d'une représentation des données dans le but d'une exploitation visuelle.

Les données présentes dans une vue sont définies grâce à une clause *SELECT*

A. Créer une vue

CREATE VIEW Nom_de_la_Vue
(colonnes)
AS SELECT ...

Exemple :

CREATE VIEW Vue
(colonneA,colonneB,colonneC,colonneD)
AS SELECT colonne1, colonne2, colonneI, colonneII
FROM Nom_table1 Alias1,Nom_tableII AliasII
WHERE Alias1.colonne1 = AliasII.colonneI
AND Alias1.colonne2 = AliasII.colonneII

CREATE VIEW *Ma_vue*
(Prenom , Nom)
AS SELECT first_name, last_name
from customer;

B. Supprimer une vue

DROP VIEW *Nom_de_la_vue*

C. Intérêts des vues

La vue représente un intermédiaire entre la base de données et l'utilisateur.

Avantages :

- sélection des données à afficher
- restriction d'accès à la table pour l'utilisateur, donc sécurité des données accrue
- un regroupement d'informations au sein d'une entité

III.Les types de données

A. Types alphanumériques

TYPE	DESCRIPTION	Min	Max	Exemples
VARCHAR2 (taille)	Chaîne de caractères de longueur variable	1 car.	2000 car.	'A' 'Bonjour DD'
NVARCHAR2 (taille)	Chaîne de caractères de longueur variable utilisant le jeu de car. National (unicode)			René
~~**LONG (taille)**~~	~~Chaîne de caractères de longueur variable.~~	~~1 car.~~	~~2 giga car.~~	~~'AAAHHHHHH...HHH'~~
CHAR (taille)	Chaîne de caractères de longueur fixe	1 car.	255 car.	'AZERTY' 'W'
NCHAR (taille)	Chaîne de caractères de longueur fixe utilisant le jeu de car. national (unicode)			

nom VARCHAR2(50),

CHAR : la chaine de caractère est remplie par des espaces, ex : « martin » si 15 caractères

B. Types numériques

TYPE	DESCRIPTION	Minimum	Maximum	Exemple
NUMBER [(taille[,precision])]	Numérique. (prec<=38, exp max -84 +127)	10exp-84	10exp127	10.9999

Ex : age NUMBER(2),

C. Types dates & heures

Type	Description	Minimum	Maximum	Exemples de valeurs
DATE	Date (du siècle à la seconde)	01/01/-4712 (avant J.C)	31/12/9999	'10-FEB-04' '10/02/04' '10/DEC/2007 10:31:01'
TIMESTAMP (fractional_seconds_precision)	Year, month, and day values of date, as well as hour, minute, and second values of time, where fractional_seconds_precision is the number of digits in the fractional part of the SECOND datetime field. Accepted values of fractional_seconds_precision are 0 to 9. The default is 6. 10-DEC-07 10.32.47.797201 AM			
TIMESTAMP (fractional_seconds_precision) WITH LOCAL TIME ZONE	All values of TIMESTAMP WITH TIME ZONE, with the following exceptions : Data is normalized to the database time zone when it is stored in the database. When the data is retrieved, users see the data in the session time zone. *TIMESTAMP [(fractional_seconds_precision)] WITH TIME ZONE*			
INTERVAL YEAR (year_precision) TO MONTH	Stores a period of time in years and months, where year_precision is the number of digits in the YEAR datetime field. Accepted values are 0 to 9. The default is 2			
INTERVAL DAY (day_precision) TO SECOND (fractional_seconds_precision)	Stores a period of time in days, hours, minutes, and seconds, where day_precision is the maximum number of digits in the DAY datetime field. Accepted values are 0 to 9. The default is 2. fractional_seconds_precision is the number of digits in the fractional part of the SECOND field. Accepted values are 0 to 9. The default is 6.			

Ex : Date_naissance DATE,

D. Autres types

TYPE	DESCRIPTION	Minimum	Maximum	Exemples
RAW (taille)	Données binaires devant être entrées en notation hexadécimale. Taille : 1 à 255 caractères	1 octet	2000 octets	
LONG RAW (taille)	Données binaires devant être entrées en notation hexadécimale.	1 octet	2 GO	
ROWID	Type réservé à la pseudo-colonne ROWID. Ne peut être utilisé.			
CLOC NCLOB	LOB de type caractère mono byte ou multi-bytes utilisant le jeu de car. national	1 octet	4 giga car.	
BLOB	BLOB ! gros objet binaire	1 octet	4 giga car.	
BFILE	pointeur vers un fichier binaire externe	1 octet	4 giga car.	
UROWID [(size)]	Adresse logique d'une ligne (chaîne de caractère en base 64) d'une table organisée en index Base 64 string representing the logical address of a row of an index-organized table. The optional size is the size of a column of type UROWID. The maximum size and default is 4000 bytes.			

E. Contraintes de données

- Valeur minimum , Valeur maximum
- Valeur par défaut, Valeur obligatoire
- Valeur unique, Clef primaire, Index secondaire
- Format ou modèle (par exemple 3 caractères majuscules suivi de 2 caractères numériques)
- Table de référence (recopie d'une valeur d'une table dans un champ d'une autre table en sélectionnant par la clef) aussi appelé CHECK en SQL
- Liste de choix

IV.DDL : " Data Definition Language "

Permet de créer des bases de données, des tables, des index, des contraintes…

CREATE, ALTER, DROP

Créer, modifier, supprimer un élément de la base.

A. Créer une table

Règles de nommages des tables : A-Z, a-z, 0-9, _

CREATE TABLE Clients
(num number(3) primary key,
nom VARCHAR2(50) not null,
prenom VARCHAR2(50),
age number(2));

1. Not Null

Pour empêcher un champ de rester vide, il faut utiliser la clause NOT NULL

CREATE TABLE Personnes
(Nom CHAR(20) ***NOT NULL****,*
Prénom CHAR(20));

2. Unique

Pour qu'un champ soit indexé sans doublons, il faut utiliser la clause UNIQUE :

CREATE TABLE Personnes
(Nom CHAR(20) ***UNIQUE****,*
Prénom CHAR(20));

3. Clef Primaire

CREATE TABLE Personnes
(Nom CHAR(20) ***PRIMARY KEY****,*
Prénom CHAR(20));

CREATE TABLE Personnes
(Nom CHAR(20) ***CONSTRAINT*** *clé_primaire PRIMARY KEY,*
Prénom CHAR(20));

CONSTRAINT permet d'attribuer le nom "clé_primaire" à la clé ainsi créée.

a) Clef primaire sur plusieurs champs

CREATE TABLE Personnes
(Nom CHAR(20),
Prénom CHAR(20),
CONSTRAINT essai_index PRIMARY KEY(Nom, Prénom));

a) Clef primaire sur champs déjà créés

Pour un champ :

```
CREATE UNIQUE INDEX essai_index
ON Personnes (Nom);
```

Dans le cas d'un index sur deux champs :

```
CREATE UNIQUE INDEX essai_index
ON Personnes (Nom, Prénom);
```

Pour supprimer un index :

```
DROP INDEX Personnes.essai_index;
```

4. Valeur par défaut

DEFAULT

```
CREATE TABLE Customer
(First_Name char(50),
Last_Name char(50),
Address char(50) DEFAULT 'inconnue',
City char(50) DEFAULT 'Grenoble',
Country char(25),
Birth_Date date);
```

```
Date_facture DATE DEFAULT CURRENT_DATE,
```

Current_date : date du jour

5. Conditions de validité

CHECK

```
quantity    NUMBER  CHECK (quantity > 0)
COULEUR CHAR(16) CHECK (VALUE IN ('blanc', 'noir', 'rouge', 'vert', 'bleu'))
NOM CHAR(32) NOT NULL CHECK (SUBSTRING(VALUE, 1, 1) <> ' ' AND UPPER(VALUE) = VALUE),
Email  char(50) NOT NULL, check (Email LIKE "%@%")
```

B. Modifier une table

1. Ajouter un champ

```
ALTER TABLE Clients
ADD Naissance DATE;
```

2. Modifier le type de données

```
ALTER TABLE Personnel
MODIFY contrat varchar2(20);
```

C. Supprimer une table

```
DROP TABLE Personnes;
```

D. Vider une table

TRUNCATE *TABLE Nom_de_la_table ;*

E. Supprimer un champ

ALTER TABLE Personnes
DROP Column Naissance;

F. Renommer une table

ALTER TABLE *ancien_nom* ***RENAME*** *to nouveau_nom;*

V. DML : " Data Manipulation Language "

Pour traiter les données.

INSERT, UPDATE, DELETE, SELECT

Insérer, modifier, supprimer et extraire des données.

A. Insérer

1. Dans l'ordre des champs

INSERT INTO table VALUES ('valeur 1', 'valeur 2', ...) ;

INSERT INTO Clients
VALUES ('2','Durand','Jules','49');

2. En nommant les champs

INSERT INTO table (nom_colonne_1, nom_colonne_2, ...
VALUES ('valeur 1', 'valeur 2', ...) ;

INSERT INTO Clients (num, age, nom)
VALUES (12, 25, 'Sylvie') ;

Si on veut la valeur par défaut ou un champ vide :

Values(20, default, 'Pierre')
Values(12, null, 'Jean ')

3. Plusieurs champs

INSERT INTO clients (prenom, nom,num , age)
VALUES ('Rébecca', 'Armand', 10, 24);
INSERT INTO clients (prenom, nom,num , age)
VALUES ('Aimée', 'Hebert',11, 36);
INSERT INTO clients (prenom, nom,num , age)
VALUES ('Marielle', 'Ribeiro',13, 27); …

B. Modifier, mettre à jour

Update

UPDATE table
SET nom_colonne_1 = 'nouvelle valeur'
WHERE condition

***UPDATE** Personnel*
***SET** nom = 'MARTIN'*
***WHERE** nom = 'LONGUET'*

1. Un enregistrement

UPDATE client
SET rue = '49 Rue Ameline',
ville = 'Saint-Eustache-la-Forêt',
code_postal = '76210'
WHERE id = 2

2. La table entière

UPDATE client
SET pays = 'FRANCE'

C. Supprimer

Delete

***DELETE** FROM table*
WHERE condition

DELETE FROM Client
WHERE age < 10 ;

D. Select

Les tables existantes

*Select * from user_tables ;*

*SELECT [DISTINCT ou ALL] * ou liste des champs*
FROM nom de table ou de la vue
[WHERE prédicats]
[GROUP BY ordre des groupes]
[HAVING condition]
[ORDER BY] liste de colonnes

***SELECT** : Spécification des colonnes du résultat*

***FROM** : Spécification des tables sur lesquelles porte l'ordre*

***WHERE** : Filtre portant sur les données (conditions à remplir pour que les lignes soient présentes dans le résultat)*

***GROUP BY** : Définition d'un groupe (sous ensemble)*

***HAVING** : Filtre portant sur les résultats (conditions de regroupement des lignes)*

***ORDER BY** : Tri des données du résultat*

---

1. Select From

***SELECT** num,nom*
***FROM** client;*

---

2. Where

SELECT num,nom
FROM client
***WHERE** nom = 'Durand';*

Attention les données sont sensibles à la casse ! Durand <> durand

3. Tous les champs de la table, étoile

*SELECT **
FROM client
WHERE nom = 'Durand';

* : rapatrie tous les champs de la table

4. Distinct

*SELECT **DISTINCT** PRENOM*
FROM client;

Distinct supprime les doublons

5. AS

*SELECT num, nom **AS** Commercial*
FROM client;

As : renomme le champ

6. Concaténation ||

*SELECT prenom **||** ' ' **||**NOM as prenom_nom*
FROM Client ;

7. Opérateurs + - * /

*SELECT Prix_ht * 1.206 AS Tarif_ttc*
FROM Tarif;

8. Order By

ORDER BY colonne1 | 1 [ASC ou DESC] [, colonne2 | 2 [ASC ou DESC] ...

```
SELECT nom, prenom
FROM client
ORDER BY nom, prenom;

ou

SELECT nom, prenom
FROM client
ORDER BY   1, 2
```

---

9. And, Or, Opérateurs de comparaison

WHERE valeur1 [NOT et] = ou < ou <= ou > ou >= ou <>valeur2 [OR ou AND ...]

```
SELECT nom, prenom ,age
FROM client
WHERE age >= 30 AND nom <'E';
```

Conseil pour une meilleure lecture :

```
WHERE        age  >= 30'
             AND
             nom <'E' ;
```

---

10. Not

```
SELECT nom, prenom
FROM client
WHERE NOT nom = 'Durand';
```

---

11. Null, Not Null

Vide, Renseigné

```
SELECT nom, prenom, adresse
FROM client
WHERE adresse IS NULL;

SELECT nom, prenom, adresse
FROM client
WHERE adresse IS NOT  NULL;
```

12. In

SELECT nom, prenom
FROM client
*WHERE nom **IN** ('Durand','Martin');*

13. Beetween

Alpha-numérique

SELECT nom, prenom
FROM client
*WHERE nom **BETWEEN** 'A' **AND** 'E' ;*

Numérique:

SELECT nom, prenom, age
FROM client
*WHERE age **BETWEEN** 30 **AND** 40 ;*

14. Like : jokers

% : 0 ou plusieurs caractères

_ : un caractère unique

SELECT nom, prenom
FROM client
*WHERE nom **LIKE 'B%'** ;*

15. Upper, Lower

Upper : majuscule

Lower : minuscule,

*SELECT **UPPER**(nom), **LOWER**(prenom)*
FROM client;

16. Extract

EXTRACT (YEAR ou MONTH ou DAY FROM nom de colonne)

Extrait le jour, mois ou année d'une date

17. Fonctions d'agrégations : moyenne, max, min, somme, compte

SELECT **AVG**(TRF_CHB_PRIX) as MOYENNE,

MAX(TRF_CHB_PRIX) as MAXI,

MIN(TRF_CHB_PRIX) as MINI,

SUM(TRF_CHB_PRIX) as TOTAL,

COUNT(TRF_CHB_PRIX) as NOMBRE

FROM TJ_TRF_CHB

WHERE TRF_DATE_DEBUT = '2001-01-01'

select avg (num),
from personnel;

select avg (num), qualification
from personnel
group by qualification;

18. Fonctions textuelles utiles

CHAR_LENGTH() permet de compter le nombre de caractères

CONCAT() concaténer plusieurs chaînes de caractères

LENGTH() retourner la longueur d'une chaîne

LOWER() transformer la chaîne pour tout retourner en minuscule

LTRIM() supprimer les caractères vides au début de la chaîne

REPEAT() répéter le texte un nombre de fois défini

REPLACE() remplacer des caractères par d'autres caractères

RTRIM() supprimer les caractères vides en fin d'une chaîne de caractère

STRCMP() comparaison binaire de 2 chaînes

SUBSTR() retourne un segment de chaîne

TRIM() supprime les caractères vides en début et fin de chaîne

UPPER() tout retourner en majuscule

E. Types de jointures

Il y a plusieurs méthodes pour associer 2 tables ensemble. Voici la liste des différentes techniques qui sont utilisées :

- **INNER JOIN :** jointure interne pour retourner les enregistrements quand la condition est vrai dans les 2 tables. C'est l'une des jointures les plus communes.
- **CROSS JOIN :** jointure croisée permettant de faire le produit cartésien de 2 tables. En d'autres mots, permet de joindre chaque lignes d'une table avec chaque lignes d'une seconde table. Attention, le nombre de résultats est en général très élevé.
- **LEFT JOIN (ou LEFT OUTER JOIN) :** jointure externe pour retourner tous les enregistrements de la table de gauche (LEFT = gauche) même si la condition n'est pas vérifié dans l'autre table.
- **RIGHT JOIN (ou RIGHT OUTER JOIN) :** jointure externe pour retourner tous les enregistrements de la table de droite (RIGHT = droite) même si la condition n'est pas vérifié dans l'autre table.
- **FULL JOIN (ou FULL OUTER JOIN) :** jointure externe pour retourner les résultats quand la condition est vrai dans au moins une des 2 tables.
- **SELF JOIN :** permet d'effectuer une jointure d'une table avec elle-même comme si c'était une autre table.
- **NATURAL JOIN :** jointure naturelle entre 2 tables s'il y a au moins une colonne qui porte le même nom entre les 2 tables SQL
- **UNION JOIN :** jointure d'union

http://sql.sh/cours/jointures

F. Opérations ensemblistes : union, intersect, except

1. Union

SELECT … FROM … WHERE …

UNION

SELECT … FROM … WHERE …

2. Intersect

SELECT … FROM …WHERE …

INTERSECT

SELECT … FROM … WHERE …

3. Except (minus)

SELECT a,b FROM table1 WHERE …

EXCEPT

SELECT c,d FROM table2 WHERE …

G. Résumé des opérateurs

Opérateurs de comparaisons	= <> < <= > >=
Connecteurs logiques	{OR | AND}
Opérateur de négation	NOT
Parenthèses	(...)
Opérateurs mathématiques	+ - * /
Comparaison logique	IS [NOT] {TRUE | FALSE | UNKNOWN}
Comparaison avec valeur	IS [NOT] NULL
Intervalle	BETWEEN val_basse AND val_haute
Comparaison partielle de chaîne de caractères	LIKE motif
Comparaison à une liste de valeur	[NOT] IN (liste)

VI.DCL : " Data Control Language "

Pour gérer les droits d'accès aux tables.

GRANT, REVOKE

Attribuer et de révoquer des droits.

- ***DELETE***: privilège de supprimer les données d'une table
- ***INSERT***: privilège d'ajouter des données à une table
- ***SELECT***: privilège d'accéder aux données d'une table
- ***UPDATE***: privilège de mettre à jour les données d'une table

VII. Exercices pratiques

Créer la table Personnel

```
CREATE TABLE Personnel
(num number(3) primary key,
titre VARCHAR2(15),
nom VARCHAR2(50) not null,
prenom VARCHAR2(50),
NomJeuneFille VARCHAR2(10),
Rue VARCHAR2(100),
DateNaissance date,
SSociale VARCHAR2(20),
CP VARCHAR2(5),
DateEntrée date,
DateSortie date,
SituationFamiliale VARCHAR2(15),
Qualification VARCHAR2(50),
Contrat VARCHAR2(15)
);
```