LANGAGE SQL

SQL

Pour communiquer avec une base de données, il faut lui envoyer des commandes ou des instructions appelées requêtes. Que ce soit pour la création, la suppression d'une table, la modification, l'insertion ou la sélection de données, le langage standard de requêtes est SQL (Standard Query Language).

SQL s'appuie sur les opérateurs de l'algèbre relationnelle définit en 1970 par Codd, mathématicien, chercheur chez IBM. Le langage SQL est basé sur le concept de relation de la théorie des ensembles.

SQL

- **SQL est un langage de définition de données (LDD, ou DDL (*en anglais*), Data definition language) : il permet de modifier la structure d'une base de données relationnelle ;**

- **SQL est un langage de manipulation de données (LMD ou *DML*(*en anglais*), Data manipulation language) : il permet de consulter, de supprimer et d' effectuer des mises à jour sur le contenu d'une base de données relationnelle ;**

- **SQL est un langage de contrôle de données (LCD, ou DCL (*en anglais*), Data control language) : il permet de gérer les privilèges des utilisateurs de la base de données ;**

SQL

- **SQL un langage de contrôle de transactions (LCT, ou en *anglais* TCL, Transaction control language) : il permet de gérer les transactions, c'est-à-dire rendre atomique divers ordres enchaînés en séquence;**
- **SQL intègre aussi d'autres modules destinés notamment à écrire des routines (procédures, fonctions ou déclencheurs) et interagir avec des langages externes.**

LANGAGE DE DEFINITION DE DONNEES

Création de table

Une table est un ensemble de lignes et de colonnes. La création consiste à définir(*en fonction de l'analyse*) le nom de ces colonnes, leur format (TYPE), la valeur par défaut à la création de la ligne (DEFAULT) et les règles de gestions appliquant à la colonne (CONSTRAINT).

Création de table

Syntaxe :

```
CREATE TABLE table
        (
            -- définition des colonnes
                colonne type [ NOT NULL [UNIQUE] ]
                    [ DEFAULT valeur ]
                    [ PRIMARY KEY ]
                    [ REFERENCES table ]
                    [ CHECK condition ] ,
                ... .
                -- contraintes de table
                    [ PRIMARY KEY (liste de colonnes) ],
                    [ UNIQUE (liste de colonnes) ] ,
                ... ,
                [ FOREIGN KEY (liste de colonnes) REFERENCES table
                [ ON DELETE {RESTRICT | CASCADE | SET NULL} ]
                [ ON UPDATE {RESTRICT | CASCADE | SET NULL} ] ,
                ... ,
                [ CHECK condition ] ,
                ...
        )
```

Création de table

Avec:

table: C'est tout simplement le nom de la table que vous allez créer

colonne : Le nom de la colonne ou du champ de la table

type: Le type de données que contient la colonne

contraintes:Les contraintes pour la colonne et pour la table

Création de table

Principaux types de données :

- **INTEGER: Ce type permet de stocker des entiers signés codés sur 4 octets.**
- **BIGINT: Ce type permet de stocker des entiers signés codés sur 8 octets.**
- **REAL: Ce type permet de stocker des réels comportant 6 chiffres significatifs codés sur 4 octets.**
- **DOUBLEPRECISION: Ce type permet de stocker des réels comportant 15 chiffres significatifs codés sur 8 octets.**

Création de table

types de données (suite):

- **NUMERIC**[(***précision***,[***longueur***])]:Ce type de données permet de stocker des données numériques à la fois entières et réelles avec une précision de 1000 chiffres significatifs. Longueur précise le nombre maximum de chiffres significatifs stockés et précision donne le nombre maximum de chiffres après la virgule.
- **CHAR** (***longueur***):Ce type de données permet de stocker des chaînes de caractères de longueur fixe. Longueur doit être inférieure à 255.
- **VARCHAR** (***longueur***):Ce type de données permet de stocker des chaînes de caractères de longueur variable. Longueur doit être inférieure à 2000.

Création de table

types de données (suite):

- **DATE: Ce type de données permet de stocker des données constituées d'une date.**
- **TIMESTAMP: Ce type de données permet de stocker des données constituées d'une date et d'une heure.**
- **BOOLEAN: Ce type de données permet de stocker des valeurs Booléenne.**
- **MONEY: Ce type de données permet de stocker des valeurs monétaires.**
- **TEXT: Ce type de données permet des stocker des chaînes de caractères de longueur variable.**

Création de table

Contraintes d'intégrité

NOT NULL : valeur nul impossible

UNIQUE: unicité d'un attribut

PRIMARY KEY : clé primaire

FOREIGN KEY: clé étrangère

CHECK: plage ou liste de valeurs

DEFAULT valeur: Permet de spécifier la valeur par défaut de l'attribut.

Une contrainte qui ne fait référence qu'à une seule colonne de la table peut faire partie intégrante de la définition de colonne

Création de table

SQL permet de spécifier les actions à entreprendre pour le maintien de l'intégrité référentielle, lors d'une suppression ou d'une modification d'un tuple référencé.

CASCADE cascader les suppressions ou modifications

Par ex. si on supprime un produit dans la table PRODUIT, toutes les ventes correspondantes seront supprimées dans la table VENTE

SET NULL rendre nul les attributs référençant

Par ex. si on modifie la référence d'un produit dans la table PRODUIT, toutes les références correspondantes seront modifiées dans la table VENTE

RESTRICT rejet de la mise à jour c'est l'option par défaut

Création de table

Exemple1 :

```
CREATE TABLE client
   (
         IdCli CHAR(4) PRIMARY KEY ,
         nom_cli CHAR(20) ,
         ville CHAR(30)
        CHECK (ville IN ('Thies', 'Dakar', 'Louga')) ,
   )
```

Création de table

Exemple 2 :

```
CREATE TABLE produit
    (
        IdPro CHAR(6) PRIMARY KEY ,
        nom_pro CHAR(30) NOT NULL UNIQUE ,
        marque CHAR(30) ,
        prix DEC(9,2) ,
        qstock SMALLINT
        CHECK (qstock BETWEEN 0 AND 100) ,
          -- contrainte de table
        CHECK (marque <> 'IBM' OR qstock < 10)

    )
```

Création de table

Exemple 3 :

```
CREATE TABLE  vente
   (
                IdCli  CHAR(4) NOT NULL
                REFERENCES  client ,
                 IdPro CHAR(6) NOT NULL ,
                 date DATE  NOT NULL ,
                 qte SMALLINT
                CHECK (qte BETWEEN 1 AND 10) ,
                   -- contrainte de table
                   PRIMARY KEY (IdCli, IdPro, date) ,
                 FOREIGN KEY (IdPro) REFERENCES produit
                 ON DELETE CASCADE  ON  UPDATE CASCADE
 )
```

CRÉATION D'INDEX

La commande CREATE INDEX permet de créer des index multi-colonne

Syntaxe :

CREATE [UNIQUE] INDEX index

ON table (colonne [ASC|DESC], ...)

- **L'option UNIQUE permet d'assurer l'unicité d'une clé**

Ex.: CREATE UNIQUE INDEX index1 ON client(Nom)

CRÉATION D'INDEX

- **Les index permettent d'accélérer les recherches**
- **Le système détermine sa stratégie d'accès en fonction des index existants**
- **Les index sont automatiquement mis à jour**
- **Il est indispensable de créer les index appropriés pour accélérer le traitement des requêtes**

CRÉATION D'INDEX

- **Il ne faut cependant pas créer des index sur n'importe quel colonne ou groupe de colonnes, car les mises à jour seraient ralenties inutilement par la maintenance de ces index**
- **Un index est supprimé par la commande DROP INDEX**

Modification d’une Table

ALTER TABLE [nom_schema].nom_table alter_specification

alter_specification:

ADD [COLUMN] column_definition[FIRST | AFTERcol_name]

| CHANGE [COLUMN] old_col_namecolumn_definition [FIRST| AFTER col_name]

| MODIFY [COLUMN] column_definition[FIRST | AFTER col_name]

| DROP [COLUMN] col_name

| DROP PRIMARY KEY

Modification d’une Table

Exemple :

Ajout d'une colonne

ALTER TABLE client ADD prénom VARCHAR(50) DEFAULT ‘Jean' ;

Modification d'une colonne

ALTER TABLE client MODIFY prénom VARCHAR(25) DEFAULT ‘Fatou';

Renommez une colonne

ALTER TABLE client CHANGE prénom prénom_cli VARCHAR (25) DEFAULT 'toto‘;

Modification d’une Table

Exemple :

Suppression d'une colonne

ALTER TABLE client DROP prénom_cli ;

Renommez une table

ALTER TABLE client RENAME personne;

LANGAGE DE MANIPULATION DES DONNEES

Langage de manipulation des données

SELECT, INSERT, UPDATE et DELETE sont les 4 commandes de manipulation des données en SQL

Exemple :

Recherche SELECT

```
SELECT P.prix
FROM produit P
WHERE P.IdPro = 'p1'
```

Langage de manipulation des données

Ajout INSERT

```
INTO client (IdCli, nom_cli, ville)
VALUES ('c4', 'Fatou', 'Dakar')
```

Mise à jour UPDATE

```
UPDATE produit P
SET P.prix = P.prix * 1.20
WHERE P.IdPro = 'p2'
```

Suppression DELETE

```
DELETE
FROM produit P
WHERE P.IdPro = 'p4'
```

LA COMMANDE SELECT

La commande SELECT permet de rechercher des données à partir de plusieurs tables ; le résultat est présenté sous forme d'une table réponse

- **Expression des projections**

Q1 Donner les noms, marques et prix des produits

Expression des projections

SELECT P.nom_pro, P.marque, P.prix

FROM produit P

Synonyme de nom de table (ou alias)

- **On peut introduire dans la clause FROM un synonyme (alias) à un nom de table en le plaçant immédiatement après le nom de la table**
- **Les noms de table ou les synonymes peuvent être utilisés pour préfixer les noms de colonnes dans le SELECT**
- **Les préfixes ne sont obligatoires que dans des cas particuliers (par ex. pour une auto-jointure) ; leur emploi est cependant conseillé pour la clarté Synonyme de nom de table (ou alias)**

Expression des projections

- **On peut introduire dans la clause FROM un synonyme (alias) à un nom de table en le plaçant immédiatement après le nom de la table**
- **Les noms de table ou les synonymes peuvent être utilisés pour préfixer les noms de colonnes dans le SELECT**
- **Les préfixes ne sont obligatoires que dans des cas particuliers (par ex. pour une auto-jointure) ; leur emploi est cependant conseillé pour la clarté**

Un alias est utilisé par SQL comme une variable de parcours de table (dite variable de corrélation) désignant à tout instant une ligne de la table

Expression des projections

Q2 Donner les différentes marques de produit

SELECT DISTINCT P.marque

FROM produit P

Contrairement à l'algèbre relationnelle, SQL n'élimine pas les doublons

Pour éliminer les doublons il faut spécifier DISTINCT

Expression des projections

Q3 Donner les références des produits et leurs prix majorés de 20%

```
SELECT P.IdPro, P.prix * 1.20
FROM produit P
```

Il est possible d'effectuer des opérations arithmétiques (+, -, *, /) sur les colonnes extraites

Expression des projections

Q4 Donner tous les renseignements sur les clients

SELECT *

FROM client

Une étoile (*) permet de lister tous les attributs

Expression des restrictions

Q5 Donner les noms des produits de marque IBM

```
SELECT P.nom_pro
FROM produit P
WHERE P.marque = 'IBM'
```

La condition de recherche (qualification) est spécifiée après la clause WHERE par un prédicat

Expression des restrictions

Un prédicat simple peut-être :

- un prédicat d'égalité ou d'inégalité (=, <>, <, >, <=, >=)

- un prédicat LIKE

- un prédicat BETWEEN

- un prédicat IN

- un test de valeur NULL

- un prédicat EXISTS

- un prédicat ALL ou ANY

Un prédicat composé est construit à l'aide des connecteurs AND, OR et NOT

Expression des restrictions

Exemples

Q6 Lister les clients dont le nom comporte la lettre A en 2ième position

```
SELECT *
FROM client C
WHERE C.nom LIKE '_A%'
```

Le prédicat LIKE compare une chaîne avec un modèle

(_) remplace n'importe quel caractère

(%) remplace n'importe quelle suite de caractères

Expression des restrictions

Q7 Lister les produits dont le prix est compris entre 5000F et 12000F

```
SELECT *
FROM produit P
WHERE P.prix BETWEEN 5000 AND 12000
```

Le prédicat BETWEEN teste l'appartenance à un intervalle

Expression des restrictions

Q8 Lister les produits de marque IBM, DELL ou Nokia

```
SELECT *
FROM produit P
WHERE P.marque IN ('IBM', 'DELL', 'Nokia')
```

Le prédicat IN teste l'appartenance à une liste de valeurs

Expression des restrictions

Q9 Lister les produits dont le prix est inconnu

```
SELECT *

FROM produit P

WHERE P.prix IS NULL
```

La valeur NULL signifie qu'une donnée est inconnue

Expression des restrictions

Q10 Lister les produits de marque IBM dont le prix est inférieur à 12000F

```
SELECT *
FROM produit P
WHERE P.marque = 'IBM' AND P.prix < 12000
```

Le connecteur AND relie les 2 prédicats de comparaison

Tri du résultat d'un SELECT

La clause ORDER BY permet de spécifier les colonnes définissant les critères de tri. Le tri se fera d'abord selon la première colonne

spécifiée, puis selon la deuxième colonne etc...

Exemple

Q11 Lister les produits en les triant par marques et à l'intérieur

d'une marque par prix décroissants

```
SELECT *
FROM produit P
ORDER BY P.marque, P.prix DESC
```

L'ordre de tri est précisé par ASC (croissant) ou DESC (décroissant) ; par défaut ASC

Expression des jointures

Le produit cartésien s'exprime simplement en incluant plusieurs tables après la clause FROM

La condition de jointure est exprimée après WHERE

Exemples :

Q12 Donner les références et les noms des produits vendus

```
SELECT P.IdPro, P.nom_pro
FROM produit P , vente V
WHERE P.IdPro = V.IdPro
```

Expression des jointures

Q13 Donner les noms des clients qui ont acheté le produit de nom ‘PC'

```
SELECT C.nom_cli
FROM client C , produit P, vente V
WHERE V.IdCli = C.IdCli
AND V.IdPro = P.IdPro AND P.nom_pro = 'PC'
```

Auto-jointure

Q14 Donner les noms des clients de la même ville que Therry

```
SELECT C2.nom_cli
FROM client C1 , client C2
WHERE C1.ville = C2.ville AND C1.nom_cli = 'Therry '
        AND C2.nom_cli <> 'Therry '
```

Cet exemple utilise, pour le couplage des villes, la jointure de la table Client avec elle-même (auto-jointure)

Pour pouvoir distinguer les références ville dans les 2 copies, il faut introduire 2 alias différents C1 et C2 de la table client

Jointures externes

La jointure externe permet de retenir lors d'une jointure

les lignes d'une table qui n'ont pas de correspondant

dans l'autre table, avec des valeurs nulles associées

On distingue jointure externe gauche, droite et complète

selon que l'on retient les lignes sans correspondant des 2 tables

ou seulement d'une SQL2 offre la possibilité de spécifier les

jointures externes au niveau de la clause FROM selon la syntaxe

suivante :

Jointures externes

FROM table1 [NATURAL] [{LEFT|RIGHT}] JOIN table2

[ON (liste de colonnes = liste de colonnes)]

NATURAL signifie jointure naturelle, c.a.d l'égalité des attributs de même nom

Jointures externes

Q15 Lister tous les clients avec le cas échéant leurs achats

```
SELECT C.IdCli, C.nom_cli, C.ville ,V.IdPro, V.date, V.qte
FROM client C NATURAL LEFT JOIN vente V
```

Sous-requêtes

SQL permet l'imbrication de sous-requêtes au niveau de la clause WHERE d'où le terme "structuré" dans Structured Query Language

Les sous-requêtes sont utilisées :

- **dans des prédicats de comparaison**

(=, <>, <, <=, >, >=)

- **dans des prédicats IN**
- **dans des prédicats EXISTS**
- **dans des prédicats ALL ou ANY**

Sous-requêtes

Une sous-requête dans un prédicat de comparaison doit se réduire à une seule valeur ("singleton select")

- **Une sous-requête dans un prédicat IN, ALL ou ANY**

doit représenter une table à colonne unique

- **L'utilisation de constructions du type "IN sous-requête" permet d'exprimer des jointures de manière procédurale ... ce qui est déconseillé !!**

Sous-requêtes

Exemple

Q16 Donner les noms des clients qui ont acheté le produit 'P1'

- **Avec sous-requête**

```
SELECT C.nom_cli
FROM client C
WHERE IdCli IN
        (
            SELECT V.IdCli
            FROM vente V
            WHERE V.IdPro = 'P1'
        )
```

Sous-requêtes

- **Avec jointure**

```
SELECT C.nom_cli
FROM client C , vente V
WHERE C.IdCli = V.IdCli
        AND V.IdPro = 'P1'
```

- **De préférence, utiliser la jointure**

Sous-requêtes

- **Requêtes quantifiées**
- **Le prédicat EXISTS**

Il permet de tester si le résultat d'une sous-requête est vide ou non

Q17 Donner les noms des produits qui n'ont pas été acheté

```
SELECT P.nom_pro
FROM produit P
WHERE  NOT EXISTS
                ( SELECT *
                  FROM vente V
                  WHERE V.IdPro = P.IdPro )
```

Sous-requêtes

Q18 Donner les noms des produits qui ont été achetés par tous les clients de Louga

```
SELECT P.nom_pro
FROM produit P
WHERE NOT EXISTS
        (
            SELECT *
            FROM client C
            WHERE C.ville = ' Louga' AND NOT EXISTS
                    (
                        SELECT *
                        FROM vente V
                        WHERE C.IdCli = V.IdCli
                         AND V.IdPro = P.IdPro
                    )
        )
```

Sous-requêtes

- **Le prédicat ALL ou ANY**

Ils permettent de tester si un prédicat de comparaison est vrai pour tous (ALL) ou au moins un (ANY) des résultats d'une sous-requête

Q19 Donner les numéros des clients ayant acheté un produit en quantité supérieure à chacune des quantités de produits achetées par le client 'c1'

```
SELECT V.IdCli
 FROM vente V
 WHERE V.qte >= ALL
            (
                SELECT W.qte
                 FROM vente W
                 WHERE W.IdCli = 'c1'
            )
```

Sous-requêtes

Q20 Donner les numéros des clients ayant acheté un produit en quantité supérieure à au moins l'une des quantités de produits achetées par le client 'c1'

```
SELECT V.IdCli
 FROM vente V
 WHERE V.qte >= ANY
            (
                SELECT W.qte
                 FROM vente W
                  WHERE W.IdCli = 'c1'
             )
```

Sous-requêtes

- **Expression des unions**

SQL permet d'exprimer l'opération d'union en connectant des SELECT par des UNION

Q21 Donner les nos des produits de marque IBM ou ceux achetés par le client no 'c1'

```
SELECT P.IdPro
FROM produit P
WHERE P.marque = 'IBM'
UNION
SELECT V.IdPro
FROM vente V
WHERE V.IdCli = 'c1'
```

Fonctions de calculs

SQL fournit des fonctions de calcul opérant sur l'ensemble des valeurs d'une colonne de table

***COUNT* nombre de valeurs**

***SUM* somme des valeurs**

***AVG* moyenne des valeurs**

***MAX* plus grande valeur**

***MIN* plus petite valeur**

Fonctions de calculs

Q22 Donner le nombre total de clients

```
SELECT count ( IdCli )
FROM client
```

Q23 Donner le nombre total de clients ayant acheté des produits

```
SELECT count ( distinct IdCli )
FROM vente
```

Fonctions de calculs

- **On peut faire précéder l'argument du mot clé DISTINCT pour indiquer que les valeurs redondantes doivent être éliminées avant application de la fonction La fonction spéciale COUNT (*) compte toutes les lignes dans une table**
- **Les valeurs nulles ne sont pas prises en compte, sauf pour COUNT(*)**
- **Si l'argument est un ensemble vide, COUNT renvoie la valeur 0, les autres fonctions renvoyant la valeur NULL**

Fonctions de calculs

Exemples :

Q24 Donner le nombre total de 'P1' vendus

```
SELECT sum ( V.qte )
FROM vente V , produit P
WHERE P.IdPro = V.IdPro
AND P.nom = 'P1'
```

Fonctions de calculs

Q25 Donner les noms des produits moins chers que la moyenne des prix de tous les produits

```
SELECT P1.nom_pro
FROM produit P1
WHERE P1.prix <
        (
        SELECT avg ( P2.prix )
        FROM produit P2
        )
```

La clause GROUP BY

Exemples :

Q26 Donner pour chaque référence de produit la quantité totale vendue

```
SELECT V.IdPro, sum ( V.qte )
FROM vente V
GROUP BY V.IdPro
```

Q27 Donner la quantité totale achetée par chaque client (0 pour ceux qui n'ont rien acheté)

```
SELECT C.IdCli, sum ( V.qte )
FROM client C NATURAL LEFT JOIN vente V
GROUP BY C.IdCli
```

La clause GROUP BY

La clause GROUP BY permet de partitionner une table en plusieurs groupes

- **Toutes les lignes d'un même groupe ont la même valeur pour la liste des attributs de partitionnement spécifiés après GROUP BY**
- **Les fonctions de calcul opèrent sur chaque groupe de valeurs**

La clause HAVING

La clause HAVING permet de spécifier une condition de restriction des groupes

- **Elle sert à éliminer certains groupes, comme WHERE sert à éliminer des lignes**

Exemples

Q28 Donner les noms des marques dont le prix moyen des produits est < 5000F

```
SELECT P.marque, AVG ( P.prix )
FROM produit P
GROUP BY P.marque
HAVING AVG ( P.prix ) < 5000
```

La clause HAVING

Q29 Donner les références des produits achetés en qte > 10 par plus de 50 clients

```
SELECT P.marque, AVG ( P.prix )
FROM vente V
WHERE V.qte > 10
GROUP BY V.IdPro
HAVING COUNT (*) > 50
```

La forme générale de SELECT

SELECT **[*DISTINCT*] liste d'attributs, expressions**

FROM **liste de tables ou vues**

WHERE **qualification**

GROUP BY **attributs de partitionnement**

HAVING **qualification de groupe**

ORDER BY **liste de colonnes [*ASC* | *DESC*]**

La commande INSERT

La commande INSERT permet d'ajouter de nouvelles lignes à une table

INSERT

INTO table [(liste de colonnes)]

{VALUES (liste de valeurs) | requête}

La commande INSERT

- **Insertion d'une seule ligne**

Q31 Ajouter le client ('c10', 'Ibrahima', 'Louga') dans la table client

```
INSERT
INTO client (IdCli, nom_cli, ville)
VALUES ('c10', 'Ibrahima', 'Louga')
```

La commande INSERT

- **Insertion de plusieurs lignes**

Q32 Ajouter dans une table « temp » de même schéma que la table Vente, toutes les ventes qui sont antérieures au 01- Jan-1994

```
INSERT
INTO temp (IdCli, IdPro, date, qte)
SELECT V.no_cli, V.IdPro, V.date, V.qte
FROM vente V
WHERE V.date < '01-jan-1994'
```

La commande UPDATE

La commande UPDATE permet de changer des valeurs d'attributs de lignes existantes

UPDATE table

SET liste d'affectations

[WHERE qualification]

- **L'absence de clause WHERE signifie que les changements doivent être appliqués à toutes les lignes de la table cible**

La commande UPDATE

Exemples

Q33 Augmenter de 20% les prix de tous les produits

UPDATE produit

SET prix = prix * 1.2

La commande UPDATE

Q34 Augmenter de 50% les prix des produits achetés par des clients de Nice

```
UPDATE produit
SET prix = prix * 1.5
WHERE EXISTS
(
SELECT *
FROM vente V , client C
WHERE V.IdCli = C.IdCli
AND C.ville = 'Nice'
)
```

La commande DELETE

La commande DELETE permet d'enlever des lignes dans une table

DELETE

FROM table

[WHERE qualification]

- **L'absence de clause WHERE signifie que toutes les lignes de la table cible sont enlevées**

La commande DELETE

Exemples

Q35 Supprimer les ventes antérieures au 2014-11-14

```
DELETE
FROM vente
WHERE date < ' 2014-11-14 '
```

La commande DELETE

Q36 Supprimer les ventes des clients de Nice antérieures au 2014-02-14

```
DELETE
FROM vente
WHERE date < ' 2014-02-14 '
AND IdCli IN
(
SELECT C.IdCli
FROM client C
WHERE C.ville = 'Nice'
)
```