SQL (Structured Query Language)

Le SQL est un langage normalisé, mais il en existe plusieurs normes. La plupart des SGBD reprennent la norme et y ajoute des fonctionnalités propres à leur SGBD **(Non portable)**.

Il est donc important d'éviter les fonctionnalités spécifiques à un environnement si on sait que le SGBD sera changé par la suite.

La structure du SQL

- DML : Data Manipulation Language
  - Opérations *CRUD* : Create, Read, Update, Delete
    - *SELECT* : Read
    - *UPDATE* : Update
    - *INSERT* : Create
    - *DELETE* : Delete
- DDL : Data Definition Language
  - Défininitions des données
    - *CREATE* : Créer
    - *DROP* : Supprimer
    - *ALTER* : Modifier
- DCL : Data Control Language
  - Sécurité
    - *GRANT*
    - *REVOKE*
  - Transaction
    - *BEGIN*
    - *COMMIT*
    - *ROLLBACK*

Règles syntaxiques

- Un espace est égal à une suite d'espaces
- Délimiteur d'instruction : **;**
- Commentaires
  - – – : pour les commentaires sur une ligne
  - /* */ : pour les commentaires sur plusieurs lignes
- Délimiteur de chaîne : **'** *(single quote)*
- Délimiteur d'identifiants : **"** *(Attention à la casse)*

Le DML

SELECT

```
SELECT {* / col1, [col2, [col3...]]}
    FROM tableName
    [{INNER / LEFT / RIGHT} JOIN tableToJoin on (condition)]
    [WHERE (condition)]
    [GROUP BY col1, [col2, [col3...]]
    [HAVING (condition)] ]
```

UPDATE

UPDATE modifie un tuple qui existe déjà.

```
UPDATE tableName
    SET col1 = expression1,
        [col2 = expression2,
        [col3 = expression3...]]
    [WHERE (condition)]
```

Attention, sans le *WHERE*, tous les tuples sont modifiés ! Souvent, on prendra une clé primaire dans la condition.

Si une contrainte d'intégrité est violée lors d'un UPDATE, aucune ligne n'est modifiée. (Atomicité de l'instruction : Tout est fait ou rien n'est fait)

L'intégrité en lecture Lorsqu'on effectue un SELECT et un UPDATE en même temps, l'un des deux se fait toujours avant l'autre pour ne pas avoir d'incohérence. On travaille sur un état figée de la base de données.

SQLCode (Return Code) Lors d'une requête, trois cas peuvent se passer :

- RC = 0 : Tout s'est bien passé
- RC > 0 : Warning (Exemple, aucune ligne modifiées lors d'un UPDATE)
- RC < 0 : Erreur
  - Sémantique
  - Syntaxique
  - Droits insuffisants

DELETE

DELETE supprime un/des tuple(s) respectant la condition. Si aucune condition fournie, tous les tuples sont supprimées si les contraintes le permettent.

```
DELETE FROM tableName
    [WHERE (condition)]
```

INSERT

INSERT insère un/des tuple(s) dans une table.

```
INSERT INTO tableName [ (colNamesList) ]
    { VALUES(colValues) /
      SELECT STATEMENT }
```

On peut spécifier une liste de colonnes pour l'ordre des colonnes dans le champ VALUES, il est recommandé de procéder de cette manière pour éviter les erreurs. À la place de spécifier des valeurs, on peut également placer un SELECT reprenant toutes les colonnes décrites dans la liste des colonnes (colNamesList).

DCL

Les transactions

Une transaction est un ensemble minimum d'instructions qui fait passer la base de données d'un état cohérent vers un autre état cohérent. Une transaction a un début et une fin.

Une transaction doit respectr le principe *ACID* : * A : Atomicity + Une transaction est indivisible, tout est effectué ou rien n'est effectué. * C : Consistency * I : Isolation + Une transaction est isolée des autres transactions. (Pas d'effets pervers) * D : Durability

Attention, une transaction ne fonctionne que sur le DML.

COMMIT COMMIT termine et valide une transaction.

```
COMMIT;
```

ROLLBACK ROLLBACK termine et annule une transaction.

```
ROLLBACK;
```

GRANT

GRANT sert à octroyer des privilèges à un utilisateur ou un groupe d'utilisateur.

```
GRANT { ALL / liste-privilege }
    ON object_name
    TO {public / liste-role}
    [WITH GRANT OPTION]
```

- WITH GRANT OPTION : Sert à donner le droit de donner les mêmes privilèges donnés.
- Droits : SELECT, DELETE, DROP, UPDATE, INSERT, ALTER
- References : Le droit de faire une foreign key sur une table pas à moi.
- Execute : Utiliser une fonction

REVOKE

```
REVOKE {ALL / liste-privileges}
    ON name_object
    FROM {liste user-col}
    [CASCADE CONSTRAINTS]
```

Le DDL

Le DDL est la partie qui définit le schéma des données. C'est à dire, création de table, supression, modification de table, ajout de contraintes, etc.

Création d'une table (CREATE TABLE)

```
CREATE TABLE tableName (
    colName1 type[(size)] [CONSTRAINT] [DEFAULT value] [NOT NULL] [,
    colName2 type[(size)] [,
    colName3 type[(size)] [
    ...]]]
    [,
    CONSTRAINTS ...
    ]
);
```

Types de données

- Les chaînes de longueurs fixes :
  - CHARACTER[(n)] / CHAR[(n)]
- Les chaînes de longueurs variables :
  - CHARACTER VARYING(n) / VARCHAR(n)
- Les chaînes de bits :
  - BIT(n)
  - BIT VARYING(n)
- Les nombres :
  - NUMBER(precision, scale)
  - DECIMAL(precision, scale)
    - precision est le nombre de chiffres
    - scale est le nombre de chiffres après la virgule
  - FLOAT / REAL / DOUBLE PRECISION
  - INT / INTEGER
- Les moments :
  - DATE : Date
  - TIME : Moment (heure, minute, seconde, etc)
  - TIMESTAMP : DATE + TIME
- LOB (Large Objects) :
  - Destiné aux stockages volumineux.
  - CLOB : Character Large Object
  - BLOB : Binary Large Object

Les contraintes d'intégrités :

- PRIMAREY KEY : Clé primaire
- UNIQUE : Clé candidate
- FOREIGN KEY : Clé étrangère
- CHECK : Condition à respectée

CONSTRAINT :

```
[CONSTRAINT constraintName] newConstraint
```

où newConstraint est une des contraintes cité ci dessus.

```
CONSTRAINT PK_tableName PRIMAREY KEY(colName(s))

CONSTRAINT U_tableName_Col1 UNIQUE(colName(s))

CONSTRAINT check_condition CHECK(condition)

CONSTRAINT FK_col1_otherTable_col2
    FOREIGN KEY col1
    REFERENCES otherTable(col2)
    [ON DELETE {RESTRICT / SET NULL / CASCADE }]
```

Préciser le nom d'une contrainte est fortement recommandé pour pouvoir la retrouver facilement.

Dans un CREATE TABLE, une contrainte peut être précisé après un attribut et n'est effective que sur cet attribut ou bien après la déclaration de tous les attributs et on peut alors définir des contraintes de table.

L'option ON DELETE du FOREIGN KEY Cette option sert à gérer les conflit d'une clé étrangère lorsqu'on veut supprimer un tuple.

- RESTRICT : Option par défaut, refuse la supression et lance une erreur.
- SET NULL : Met les valeurs référencant la valeur à supprimer à NULL.
- CASCADE : Toutes les références sont supprimées. *Très dangereux !*

L'option DEFERRABLE Lors d'une transaction, on peut différer le contrôle des contraintes. Les contraintes seront toutes vérifiées à la fin de la transaction et non à chaque instructions de cette transaction. Cela permet de modifier la base de données et la mettre dans un état incohérent tout en pouvant la remettre dans un état cohérent avant la fin de la transaction.

```
DEFERRABLE [INITIALLY {DEFERRED / IMMEDIATE}]
```

Quand on ne précise rien, l'option par défaut du système est appliquée.

- DEFERRED : On attend la fin de la transaction (COMMIT) pour vérifier.
- IMMEDIATE : Les contraintes sont vérifiées à chaques instructions.

On peut modifier la valeur d'une contrainte *définie DEFERRABLE* avec :

```
SET CONSTRAINTS {constraintsList / ALL} {IMMEDIATE / DEFERRED}
```

Attention : ALL modifie les contraintes définies avec DEFERRABLE !

Mdofication de table (ALTER TABLE)

```
ALTER TABLE table_name
    {ADD col_name TYPE |
     DROP col_name |
     ADD CONSTRAINT ... |
     DROP CONSTRAINT constraint_name |
     MODIFY col_name ... }
```

MODIFY n'est pas accepté partout.

Commentaires

Commentaires sur tables

```
COMMENT ON TABLE table_name IS 'comment'.
```

Commentaires sur colonnes

```
COMMENT ON COLUMN table_name.col_name IS 'comment'.
```

Le catalogue

Tables :

- SYSTABLES : Table système qui reprend toutes les tables de la BD.
- SYSCOLUMNS
- SYSINDEXES
- SYCONSTRAINTS

Vues

Créer une vue

```
CREATE VIEW view_name [(col1, col2, ...)]
    AS

    SELECT STATEMENT
    [WITH CHECK OPTION]
```

WITH CHECK OPTION vérifie que les conditions du **SELECT STATEMENT** lors du'ne modification de tuple dans la vue.

Supprimer une vue

```
DROP VIEW view_name
```

Index

Créer un index

```
CREATE [UNIQUE] INDEX index_name
    ON table_name (col1, col2, ...)
```

Supprimer un index

```
DROP INDEX index_name
```