# Chapitre 4
# Utilisation d'un SGBD –SQL

(SQL : Structured Query Language)

## Introduction

- SQL est un langage créé par Codd pour mettre en œuvre son modèle relationnel
- Langage le plus diffusé au sein des SGBD relationnels (DB2, Oracle, MySQL, …)
- C'est un langage de définition (LDD) et de manipulation (LMD) de données.
- Langage normalisé.

## Oracle

- TP : Oracle SQLPLUS
- Multiplateforme
- Interfaces
  - Ligne de commande
  - ODBC, JDBC
  - Connections pour Java
- Documentation Oracle sqlplus :
  - http://www.oracle.com/pls/db92/homepage
  - http://www.oracle.com/pls/db92/db92.sql_keywords?

## Langage SQL

1) **Langage de Manipulation des Données :**
   - SELECT pour l'interrogation d'une ou plusieurs tables.
   - INSERT pour l'ajout de lignes (tuples) dans une table.
   - UPDATE pour la modification de lignes.
   - DELETE pour la suppression de lignes
2) **Langage de définition des données**
   - CREATE / DROP TABLE
   - CREATE VIEW
3) **Langage de contrôle des données**
   - GRANT / REVOKE
   - BEGIN / END TRANSACTION
   - COMMIT / ROLLBACK.

## Exemple de base de données

- Base de données Concert

```
Salle(idS, nom, ville, nbplaces)
Groupe(idG, nom, nationalité, genre)
Concert(idC, idS, idG, date)
```

- SQL> select * from groupe;

| IDG | NOM | NATIONALITE | GENRE |
|---|---|---|---|
| 1 | AC/DC | Australo-britannique | Rock |
| 2 | M | Francais | Chanson Francaise |
| 3 | Eiffel | Francais | Rock |
| 4 | Rolling stones | US | Rock |
| 5 | Gojira | Francais | Death Metal |

## Base de données Concert

- SQL> select * from salle;

| IDS | NOM | VILLE | NBPLACES |
|---|---|---|---|
| 1 | Laiterie | Strasbourg | 1000 |
| 2 | Zenith | Strasbourg | 2000 |
| 3 | Grillen | Colmar | 500 |
| 4 | 112 | Terville | 300 |

- SQL> select * from concert;

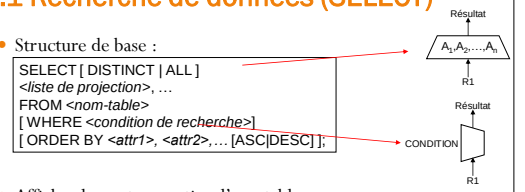
| IDC | IDS | IDG | DATE |
|---|---|---|---|
| 1 | 1 | 1 | 11-NOV-07 |
| 2 | 3 | 2 | 12-DEC-08 |
| 3 | 1 | 1 | 23-FEV-11 |
| 4 | 4 | 3 | 26-MAR-11 |

# 1. Langage de manipulation des données (LMD)

## 1.1 Recherche de données (SELECT)

- Structure de base :

```
SELECT [ DISTINCT | ALL ]
<liste de projection>, …
FROM <nom-table>
[ WHERE <condition de recherche>]
[ ORDER BY <attr1>, <attr2>,… [ASC|DESC] ];
```




- Afficher le contenu entier d'une table:

```
SELECT * FROM <nom-table>;
```

## Alias d'attributs et de tables

- Alias d'un attribut

```
SELECT <expr-i> as <alias-attr-1>, <expr-j> as <alias—attr-2>, …
FROM …
WHERE …
ORDER BY … ;
```

- Alias d'une table

```
SELECT …
FROM <nom-table-1> <alias-table-1>,<nom-table-2> <alias-table-2>
WHERE …
ORDER BY … ;
```

## Conditions de recherche : Expressions logiques [ WHERE <*condition de recherche*>]

- Une condition de recherche est une expression logique : chaque ligne de la table interrogée est renvoyée par la requête **si cette ligne satisfait la condition spécifiée**.
- **Comparaisons**
  - Arithmétique (=, <, >, <=, <=)
  - Textuelle (<expr> LIKE <modèle> ex: nom LIKE 'r%'
    nom LIKE 'dupon_'
  - Sur intervalle (BETWEEN <expr-1> AND <expr-2>)
  - Sur liste (<expr-1> [NOT] IN (<expr-2>, <expr-3>, …))
- **Opérateurs logiques** :
  - NOT <expr>
  - <expr1> AND <expr2> (permet d'imposer plusieurs conditions)
  - <expr1> OR <expr2>

## Conditions de recherche : Expressions logiques [ WHERE <*condition de recherche*>]

- La valeur NULL

```
<attribut> IS NULL
<attribut> IS NOT NULL
```

- Renvoie de la première expression non nulle parmi les expressions entre parenthèses :

```
COALESCE(<expr-1>,<expr-2>,…)
```

## 1.2 Expressions

- Expressions Mathématiques
  - Les opérateurs arithmétiques sont disponibles

```
-, +, *, /
```

  - Quelques fonctions mathématiques

```
ABS, SIN, COS, TAN, ASIN, ACOS, ATAN, COT, EXP, LN, LOG
PI, POW, RAND, ROUND, SIGN, SQRT …
```

## Expressions Chaînes de caractère

- Opérations sur les chaînes :
  - `UPPER(expression_caractere)` : écrit en majuscules
  - `LOWER(expression_caractere)` : écrit en minuscules
  - `SUBSTR(expression_caractere,pos_deb,nbcar)` : extrait une sous-chaine de « nbcar » caracteres a partir de la position « deb »
- Concaténation, suppression des espaces a gauche ou a droite ???

## Support du temps : Date, heure

- Type de données date avec opérations
  - DATE, TIME, TIMESTAMP
  - Exemples :

```
SQL> select sysdate from dual;
SYSDATE
---------
10-FEV-11

SQL> select current_timestamp from dual;
CURRENT_TIMESTAMP
---------------------------
10-FEV-11 02.57.30.767319 AM +08:00
```

## Affichage des dates

- Fonction TO_CHAR(date,<format>)
  <format> :
  - 'DD/MM/YY' : jour, mois et annee sur deux chiffres'DD/MON/YYYY' : mois sur trois lettres
  - DAY : nom du jour
  - D : numero du jour dans la semaine
  - DD : numero du jour dans le mois
  - DDD : numero du jour dans l'annee
- Exemples :

```
SQL> select to_char(sysdate,'DD/MM/YYYY')
  from dual;

TO_CHAR(SY
 ----------
15/10/2008
```

## Affichage des dates

- Quelques exemples :

```
SQL> select to_char(sysdate,'DAY DD MONTH YEAR')
as "DATE" from dual;

 DATE
 ----------------------------------------
THURSDAY 10 FEVRIER TWO THOUSAND ELEVEN
```

/* Programmation de l'annee 2011 */

```
SQL> select nom, ville,to_char(date,'DD/MM/YY')
 from concert where date like '%11';
```

## Traitement des dates

- EXTRACT(champ FROM source)

```
SQL> select extract (year from current_date)
  as ANNEE from dual;

     ANNEE
     ------------
     2011
```

- CAST (expression AS type | domaine)

```
SQL> update consomme set date_cons =cast('16-
  MAR-97' as date) where nb=10 and nv= 10;
  (1 row updated)
```

## 1.3 Agrégations

- Agregat = partitionnement horizontal d'une relation.
  Exemples :
  - Connaitre le nombre total de concerts à Strasbourg en 2011
  - Connaitre le prochain concert de Gojira à la laiterie
- Syntaxe complète :

```
SELECT <liste de projection>
FROM <liste de tables>
WHERE <condition de recherche>]
GROUP BY <attribut de partitionnement>
HAVING <critere de restriction>];
```

## Fonctions de calcul et agrégats

- Fonctions de calcul :
  - COUNT : nombre de valeurs d'un ensemble
  - SUM : somme des valeurs d'un ensemble
  - AVG : valeur moyenne d'un ensemble
  - MAX : valeur maximum d'un ensemble
  - MIN : valeur minimum d'un ensemble
- Exemples :

## 1.4 Interrogation sur plusieurs tables

- Permet de faire une requête sur plusieurs tables (lien clé primaire / étrangère)
- 2 méthodes
  1. Sous-requêtes (subquery)
  2. Jointure
- En général :
  - Les sous-requêtes sont préférables pour comparer des agrégations à d'autres valeurs.
  - Les jointures sont idéales pour afficher des résultats provenant de plusieurs tables.

## Sous-requêtes

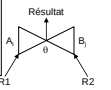
- Création d'une requête SELECT dans la clause WHERE

```
SELECT …
FROM …
WHERE <expr-i>, <expr-j>, …
<comparaison> | IN | NOT IN
( SELECT <expr-r>, <expr-s>, … FROM …
WHERE …
)
ORDER BY …
;
```

## Jointure interne

- Ajout d'une condition de jointure dans la clause WHERE :

```
SELECT <expr-i>, <expr-j>, …
FROM <nom-table-1>, <nom-table-2>, …
WHERE <expr-condition-jointure>
ORDER BY <nom-attr-k>, <nom-attr-l>, …;
```




**La condition doit être du type : Ai θ Bj**

Ai un attribut de la première table

Bj un attribut de la deuxième table

## Autres jointures

- **Produit cartésien (=Jointure sans qualification)**

  Exemple :

```
select * from Artiste, Concert;
```

- **Jointure externe (outer-join)**

```
SELECT …
FROM R1 [NATURAL] [{LEFT | RIGHT}] JOIN R2
WHERE …
```

- **Autre syntaxe jointure externe droite ou gauche : …=… (+)**

  /* afficher le nom des groupes ainsi que leurs concerts si il y en a */

```
SQL> select g.nom, c.salle, c.ville, c.date
from groupe g, concert c
where g.idG=c.idG(+);
```

## Expression des unions

- Exemple : nom des groupes ayant plus d'un concert ou aucun concert

```
(select g.nom from groupe g, concert c
 where g.idG=c.idG group by c.idG
 having count(c.idG)>1)
UNION
(select nom from groupe where idG not in
 (select idG from concert));
```

## 1.5 Modification des données

- Insertion d'une ligne (tuple)

```
INSERT INTO <nom-table> (<nom-attr-1>, <nom-attr-2>,…)
VALUES (<valeur-1>,<valeur-2>,…);
```

(Les valeurs non spécifiées sont insérées avec la valeur NULL.)

- Si on donne la valeur de tous les champs on peut simplifier par:

```
INSERT INTO <nom-table>
VALUES (<valeur-1>,<valeur-2>,…);
```

- Insertion multiple à l'aide d'une requête

```
INSERT INTO <nom-table>    (<nom-attr-1>, <nom-attr-2>,…)
SELECT …;       /* la requête  */
```

/* permet d'insérer un commentaire*/

## Suppression / Modification

- Modification du contenu :

```
UPDATE <nom-table1>, …
SET<nom-attr-i> = <expr-1|NULL>,
<nom-attr-j> = <expr-2>,
…
WHERE <expr-condition>;
```

- Suppression du contenu :

```
DELETE FROM <nom-table>
WHERE <expr-condition>;
```

## Option 'ON DELETE CASCADE'

- En cas de suppression de tuples dans une table dont la clé primaire est utilisée comme référence (clé étrangère) dans une autre table :
  =>Par défaut la suppression est interdite (violation de contrainte référentielle)
- **Exemple :**

```
SQL> delete from groupe where IDG =2;

ERROR at line 1:
ORA-02292: integrity constraint
(SLEBRE.FK_CONCERT_GROUPE) violated - child
record found
```

=> Erreur générée par le fait que la table concert contient des tuples faisant référence au groupe n°2 (IDG =2).

## Option 'ON DELETE CASCADE'

- Avec l'option ON DELETE CASCADE à la création de la contrainte d'intégrité référentielle, la suppression des tuples référencés dans une autre table entraine la suppression des tuples référençants.
- **Exemple :**

```
SQL> alter table emp drop constraint
emp_deptno_fk;
Table altered.

SQL> alter table emp add constraint
emp_deptno_fk foreign key (deptno) references
dept(deptno) on delete cascade;
Table altered.
```

## Option 'ON DELETE CASCADE'

- Suppression dans la table groupe :

```
SQL> delete from groupe where IDG=2;
no rows selected
```

- Cela a entrainé la suppression des tuples dans la table groupe :

```
SQL> select * from concert where IDG=2;
no rows selected
```

- Avec l'option 'on delete cascade' dans la contrainte d'intégrité référentielle, les tuples de la table concert concernant les concert du groupes 'M' ont automatiquement été supprimés :

```
SQL> select * from concert where nom like 'M';
no rows selected
```

## 2. Le langage de définition de données (LDD)

## Langage de définition de données (LDD)

- Instructions SQL permettant d'agir sur les éléments constituant d'un schéma de base de données relationnelle :
  - Tables
  - Vues
  - Index
- Les instructions principales permettront de :
  - Créer
  - Modifier
  - Supprimer
  - Renommer

## Gestion de la Base de données

- Création :

```
CREATE DATABASE <nom de la bd>;
```

- Suppression :

```
DROP DATABASE [IF EXISTS] <nom de la bd>;
```

## Création d'une table

- Création :

```
CREATE TABLE <nom_table>
(<def_colonne>*
[<def_contrainte_table>*]);
```

- Avec :
  - `<def_colonne>::=`

```
<nom colonne> <type| nom_domaine> <clause defaut>
[CONSTRAINT nom_contrainte
<NOT NULL | UNIQUE |PRIMARY KEY| CHECK (condition)
| REFERENCES nom_table (liste_colonnes)>]
```

  - `<def_contrainte_table> ::=`

```
CONSTRAINT nom_contrainte
< UNIQUE (liste_colonnes)
| PRIMARY KEY (liste_colonnes) | CHECK (condition)
| FOREIGN KEY (liste_colonnes)
REFERENCES nom_table (liste_colonnes) >
```

## Création/suppression d'une table

- Exemple :

```
create table gaulois (
gauno number(3) primary key,
nom varchar2(15) not null, sexe char(1),
metier varchar2(15), vilno number(2),
constraint CK_sexe check (sexe in ('M', 'F')),
constraint fk_gaulois_vilno foreign key
  (vilno) references village
);
```

- Suppression d'une table :
  DROP TABLE [IF EXISTS] <nom-table>;
- Supprimer une table malgré ses contraintes (clé étrangère)
  DROP <nom-tablet> cascade constraints;

## Option d'un attribut

- PRIMARY KEY
  - Clé primaire, doublons et valeur NULL interdites
- NOT NULL
  - Valeur NULL interdite
- UNIQUE
  - Doublons interdits
- DEFAULT <valeur>
  - Spécifie une valeur par défaut

## Définition de vues

- Création de vue :

```
CREATE [OR REPLACE] VUE <nom_vue>
AS <spécification de question>
[WITH CHECK OPTION]
```

- Suppression de vue :

```
DROP VIEW <nom_vue>;
```

- Exemple :

```
SQL> create view rock(idG, nom,
  nationalité) as(select idG, nom,
  nationalité from Groupe where
  upper(genre)='ROCK');
View created.
```

## Modifier le schéma d'une table

- Ajouter un attribut :

```
ALTER TABLE <nom-table>
ADD COLUMN <nom-attr> <type> [ <options> ] ;
```

- Ajouter une contrainte :

```
ALTER TABLE <nom-table>
ADD CONSTRAINT<def-constr> ;
```

- Modifier la définition d'une colonne :

```
ALTER TABLE <nom-table>
MODIFY <nom-col> <nouveau-type-col> ;
```

- Retirer la contrainte 'not null' de la définition d'une colonne :

```
ALTER TABLE <nom-table>
MODIFY <nom-col> null;
```

## Modifier le schéma d'une table

- Supprimer un attribut :

```
ALTER TABLE <nom-table>
DROP COLUMN <nom-attr>;
```

- Supprimer une contrainte :

```
ALTER TABLE <nom-table>
DROP CONSTRAINT <nom-constr>;
```

- ATTENTION :
  - Supprimer la contrainte de clé primaire :

```
ALTER TABLE <nom-table>
DROP PRIMARY KEY;
```

  - Supprimer uns contrainte d'unicité :

```
ALTER TABLE <nom-table> DROP UNIQUE (<nom-attr>) ;
```

## Modifier le schéma d'une table

- Renommer une table :

```
RENAME <nom-table> TO <nouveau-nom-table>;
```

- Renommer une colonne :

```
ALTER TABLE <nom-table>
RENAME COLUMN <nom-col> TO <nouveau-nom-col>;
```

## Exemples

- `create table dept (`
  `deptno number(2) primary key,`
  `loc varchar2(15) not null);`
- `alter table dept add (dname varchar2(15));`
- `alter table dept add constraint dept_dname_uk UNIQUE (dname);`
- `insert into dept(deptno,dname,loc) values (1,'info','Strasbourg');`
- `insert into dept(deptno,dname,loc) values (2,'ventes','Barcelone');`
- `insert into dept(deptno,dname,loc) values (3,'musique','Londres');`

## Démarche

Les étapes précédentes (conceptuelle, logique) ont conduit à la définition du schéma relationnel et d'un ensemble de contraintes d'intégrité. A partir de ce schéma on appliquera les étapes suivantes :

1. Création des tables relationnelles
   - A l'aide du LDD créer la structure et les contraintes
   - Le choix des types de données (domaines) est important
2. Chargement des données
   - A l'aide du LMD insérer les données tuple par tuple ou avec des outils d'importations propre à la BD
   - Faire attention aux contraintes lors d'importation de gros volumes
3. Réalisation des requêtes

## Lister des éléments de la BDD

- Lister les tables du schéma de l'utilisateur courant :
  **`SELECT table_name FROM user_tables;`**
- Lister les tables accessibles par l'utilisateur :
  **`SELECT table_name FROM all_tables;`**
- Lister toutes les tables (il faut être ADMIN) :
  **`SELECT table_name FROM dba_tables;`**
- Lister tous les objets :
  **`SELECT * FROM user_objects;`**
- Lister tous les utilisateurs :
  **`SELECT username FROM all_users order by username ;`**
  **`DESC all_users;`**
- Qui suis-je ? **`SHOW USER;`**

# 3. Langage de contrôle des données

## Gestion des états stables de la BDD

- Enregistrer les modifications (les rendre permanentes )

  **COMMIT;**

  Rq : **COMMIT** rend les modifications visibles par les autres utilisateurs.
- Enregistrer l'état courant de la BDD

  **SAVEPOINT <savepoint-name>;**
- Revenir à l'état de la BDD au moment du dernier COMMIT (undo)

  **ROLLBACK;**
- Revenir a un point de sauvegarde :

  **ROLLBACK TO SAVEPOINT <savepoint-name>;**

## Gestion des droits d'accès

- Donner un privilege :
  **GRANT < privilege > ON < table_name > TO <user_name> [WITH GRANT OPTION];**
- Retirer un privilege :
  **REVOKE < privilege > ON < table_name > FROM <user_name>;**
  avec
  - **< privilege > = SELECT, UPDATE, INSERT, DELETE, ALTER, ALL**
  - **WITH GRANT OPTION** = donne le droit de transmettre le privilège