# Bases de données (SQL)

Skander Zannad et Judicaël Courant

2014-03-26

## 1 Le modèle logique (MLD)

On a représenté des données par des tables. Par exemple, pour les films :

| titre | date |
|---|---|
| Gran Torino | 2008 |
| The good, the Bad and the Ugly | 1966 |
| Study in Pink | 2010 |
| Schindler's List | 1993 |
| Dr Strangelove | 1964 |
| Invictus | 2009 |

Au départ, on a identifié les films par leur titre. On dit que le champ titre de la table «films» est une *clé primaire* pour signifier que :

- tout film a un titre ;
- pour un titre donné, il y au plus un film ayant ce titre ;
- on fera référence à tout film par son titre.

Une clé primaire peut être constituée de plusieurs champs. Exemple pour la table «personnes», le couple (nom, prénom). On dit qu'il s'agit d'une clé primaire *composée*.

| nom | prénom | datenaissance |
|---|---|---|
| Kubrick | Stanley | 1928 |
| Spielberg | Steven | 1946 |
| Eastwood | Clint | 1930 |
| Cumberbatch | Benedict | 1976 |
| Freeman | Martin | 1971 |
| Leone | Sergio | 1929 |
| McGuigan | Paul | 1963 |
| Sellers | Peter | 1925 |

Dans la table «joue», on fait référence à ces clés. On dit que les champs (nom, prénom) de la table «joue» sont une *clé étrangère* car ils font référence à une clé primaire de la table «personnes».

| nom | prenom | titre | nom (de personnage) |
|---|---|---|---|
| Eastwood | Clint | The good, the Bad and the Ugly | Blondie |
| Eastwood | Clint | Gran Torino | Walt Kowalski |
| Cumberbatch | Benedict | Study in Pink | Sherlock Holmes |
| Freeman | Martin | Study in Pink | Dr John Watson |
| Selers | Peters | Dr Strangelove | Dr Strangelove |
| Selers | Peters | Dr Strangelove | Group Capt. Lionel Mandrake |
| Selers | Peters | Dr Strangelove | President Merkin Muffley |

# 2 Vers le modèle physique (MPD)

Pour implanter cette base de données on peut choisir n'importe quel langage de programmation.

**Mais** tous les langages ne sont pas aussi adaptés à la modélisation et l'interrogation des données.

## 2.1 SQL

**SQL**[1] : Standard Query Language
Langage spécialisé :

- Langage de définition de données (LDD ou DDL), pour définir le format des données ;
- Langage de manipulation de données (LMD ou DML) pour ajouter ou supprimer des données ;
- Instruction SELECT pour effectuer des requêtes (recherches) sur les données.

C'est **le** standard actuel (décliné en différentes versions), implanté dans les systèmes de gestion de bases de données.

## 2.2 Définition de tables

Nous allons utiliser SQLite, un outil très dépouillé pour la manipulation d'une base de données.

En SQlite, une base de données est stocké dans un fichier (et tout fichier contient une unique base de données).

1. Prononcer comme le mot anglais *sequel*.

### 2.2.1 Lancement de SQLite

En ligne de commande, taper

`sqlite3` *f*

pour lancer l'interface (textuelle) de sqlite3.
Cette commande :

- ouvre la base de données contenue dans le fichier *f* ;
- crée cette base si ce fichier n'existe pas.

### 2.2.2 Table «personne»

On va créer une table «personne» avec 4 champs :

- `id`
- `nom`
- `prenom`
- `datenaissance`

Il reste à préciser le *type* des données que contiendront ces champs.

### 2.2.3 Types de données en SQL

Nombreux types de données, dont :

**VARCHAR(**$n$**)** Une chaîne de caractères d'au plus $n$ caractères ;

**INT ou INTEGER**

**REAL** nombre flottant

**DECIMAL(**$p$**,** $s$**)** décimaux sur $p$ chiffres, dont $s$ après la virgule.

**DATE** une date (exemple : 2014-03-27)

**TIME** une heure de la journée (exemple 15 :43 :07, précision en général $10^{-7}s$).

**DATETIME** Couple des deux précédents (ex : 2014-03-27 15 :43 :07).

## 2.3 Définition des tables

Fait l'objet de la partie de SQL appelée DDL (Data Definition Language). Les deux opérations les plus importantes :

**CREATE** Pour créer une table ;

**DROP** Pour effacer une table («DROP TABLE *nomtable* ;»).

Utilisation :

```
CREATE TABLE PERSONNE (
  id INTEGER,
  nom VARCHAR(50) NOT NULL,
  prenom VARCHAR(50) NOT NULL,
  datenaissance DATE,
  PRIMARY KEY(id));
```

NB :

- ne pas oublier de terminer les commandes par des « ;»
- noter la déclaration de la clé primaire.

On crée de même la table «personnage» :

```
CREATE TABLE PERSONNAGE (
  id INTEGER,
  nom VARCHAR(50) NOT NULL,
  PRIMARY KEY(id)
);
```

et la table «joue»

```
CREATE TABLE JOUE (
  idacteur INTEGER,
  idfilm INTEGER,
  idpersonnage INTEGER,
  PRIMARY KEY(idacteur, idfilm, idpersonnage),
  FOREIGN KEY(idacteur) REFERENCES PERSONNE,
  FOREIGN KEY(idfilm) REFERENCES FILM,
  FOREIGN KEY(idpersonnage) REFERENCES PERSONNAGE
);
```

Notez les déclarations de clés étrangères.

```
CREATE TABLE FILM (
  id INTEGER,
  titre VARCHAR(50) NOT NULL,
  date DATE,
  idrealisateur INTEGER,
  PRIMARY KEY(id),
  FOREIGN KEY(idrealisateur) REFERENCES PERSONNE
);
```

## 2.4 Data Manipulation Language (DML)

### 2.4.1 Insertions de valeurs

On peut insérer une donnée :

```
INSERT INTO PERSONNE (nom, prenom, datenaissance)
  VALUES('Kubrick', 'Stanley', date('1928-07-26'));
```

NB : on n'a pas donné de valeur pour `id`.

- Pour une clé primaire entière, la base de données donne une valeur si on ne la fournit pas ;
- On peut indiquer lors de la création de la table une valeur par défaut pour les champs manquants, sinon la valeur `NULL` est mise par défaut, sauf si on l'a interdit lors de la création de la table.

Exemples :

```
INSERT INTO PERSONNE (prenom, nom)
  VALUES ('Clint', 'Eastwood');
```

est accepté (pas de date de naissance renseignée).

NB : pas besoin de mettre les champs dans le même ordre que dans la déclaration de la table.

```
INSERT INTO PERSONNE (nom) VALUES ('Spielberg');
```

déclenche une erreur :

```
Error: PERSONNE.prenom may not be NULL
```

### 2.4.2 Modification de valeurs

```
UPDATE PERSONNE
SET datenaissance = date('1930-05-31')
WHERE prenom = 'Clint' AND nom = 'Eastwood';
```

### 2.4.3 Suppression de valeurs

```
DELETE FROM PERSONNE WHERE prenom='Clint';
```

### 2.4.4 Vérification des contraintes de clés primaires

```
INSERT INTO PERSONNE (id, prenom, nom)
  VALUES (3, 'Steven', 'Spielberg');
INSERT INTO PERSONNE (id, prenom, nom)
  VALUES (3, 'Benedict', 'Cumberbatch');
```

Première insertion acceptée, la deuxième donne :

```
Error: PRIMARY KEY must be unique
```

## 2.5 Requêtes

Les requêtes SQL sont introduites par le mot-clé `SELECT`.

Une requête retourne une liste de $n$-uplets.

### 2.5.1 Lister tous les éléments d'une table

```
SELECT * FROM personne;
```

| id | nom | prenom | datenaissance |
|---|---|---|---|
| 1 | Kubrick | Stanley | 1928-07-26 |
| 2 | Spielberg | Steven | 1946-12-18 |
| 3 | Eastwood | Clint | 1930-05-31 |
| 4 | Cumberbatch | Benedict | 1976-07-19 |
| 5 | Freeman | Martin | 1971-09-08 |
| 6 | Leone | Sergio | 1929-01-03 |
| 7 | McGuigan | Paul | 1963-09-19 |
| 8 | Sellers | Peter | 1925-09-08 |

### 2.5.2 Lister certaines colonnes d'une table

```
SELECT nom, prenom FROM PERSONNE;
```

| nom | prenom |
|---|---|
| Kubrick | Stanley |
| Spielberg | Steven |
| Eastwood | Clint |
| Cumberbatch | Benedict |
| Freeman | Martin |
| Leone | Sergio |
| McGuigan | Paul |
| Sellers | Peter |

### 2.5.3 Lister certaines lignes d'une table

Si on sait que Clint Eastwood est le réalisateur no 3 :

```
SELECT titre, date FROM FILM
WHERE idrealisateur = 3;
```

| titre | date |
|---|---|
| Gran Torino | 2008 |
| Invictus | 2009 |

Et sinon :

```
SELECT FILM.titre, FILM.date FROM FILM, PERSONNE
  WHERE idrealisateur = PERSONNE.id
  AND nom = 'Eastwood' ;
```

| titre | date |
|---|---|
| Gran Torino | 2008 |
| Invictus | 2009 |

### 2.5.4 Ensembles ou listes ?

```
SELECT nom, prenom FROM PERSONNE, JOUE
WHERE id = idacteur;
```

| nom | prenom |
|---|---|
| Eastwood | Clint |
| Eastwood | Clint |
| Cumberbatch | Benedict |
| Freeman | Martin |
| Sellers | Peter |
| Sellers | Peter |
| Sellers | Peter |

SELECT ne retourne pas un ensemble de $n$-uplets mais en fait une liste de $n$-uplets : il peut y avoir des doublons. On peut les supprimer :

```
SELECT DISTINCT nom, prenom FROM PERSONNE, JOUE
WHERE id = idacteur;
```

| nom | prenom |
|---|---|
| Eastwood | Clint |
| Cumberbatch | Benedict |
| Freeman | Martin |
| Sellers | Peter |

### 2.5.5 Tri des résultats

On peut trier la liste retournée :

```
SELECT DISTINCT nom, prenom FROM PERSONNE, JOUE
WHERE id = idacteur
ORDER BY nom ASC, prenom ASC;
```

| nom | prenom |
|---|---|
| Cumberbatch | Benedict |
| Eastwood | Clint |
| Freeman | Martin |
| Sellers | Peter |

### 2.5.6 Requêtes agrégats

```
SELECT COUNT(*) AS nbfilms FROM FILM;
```

| nbfilms |
|---|
| 6 |

```
SELECT MIN(date) AS 'date premier film' FROM FILM;
```

| date premier film |
|---|
| 1964 |

On peut calculer des cumuls. Par exemple, le nombre de films que les différents réalisateurs ont tournés :

```
SELECT nom, prenom, COUNT(*) as nbfilms
FROM PERSONNE, FILM
WHERE idrealisateur = PERSONNE.id
GROUP BY PERSONNE.id;
```

| nom | prenom | nbfilms |
|---|---|---|
| Kubrick | Stanley | 1 |
| Spielberg | Steven | 1 |
| Eastwood | Clint | 2 |
| Leone | Sergio | 1 |
| McGuigan | Paul | 1 |

Pour compter dans combien de films chaque acteur a joué :

```
SELECT nom, prenom, COUNT(*) FROM PERSONNE, JOUE
WHERE PERSONNE.id=idacteur
GROUP BY PERSONNE.id;
```

| nom | prenom | COUNT(*) |
|---|---|---|
| Eastwood | Clint | 2 |
| Cumberbatch | Benedict | 1 |
| Freeman | Martin | 1 |
| Sellers | Peter | 3 |

Aïe, ce n'est clairement pas ce qu'on voulait (Peter Sellers n'a joué que dans un film de notre base de données). Pourquoi ?

Façon possible de corriger (compliquée)

```
SELECT nom, prenom, COUNT(*) AS nbfilms
FROM PERSONNE, FILM
WHERE EXISTS
 (SELECT * FROM joue
  WHERE PERSONNE.id=idacteur AND FILM.id=idfilm)
GROUP BY PERSONNE.id;
```

| nom | prenom | nbfilms |
|---|---|---|
| Eastwood | Clint | 2 |
| Cumberbatch | Benedict | 1 |
| Freeman | Martin | 1 |
| Sellers | Peter | 1 |

La condition WHERE porte sur les données qu'on va ensuite agréger. On peut aussi mettre des conditions sur les agrégats à retenir :

```
SELECT nom, prenom, COUNT(*) AS nbroles
FROM personne, joue
WHERE personne.id=idacteur
GROUP BY personne.id
HAVING COUNT(*) >= 2;
```

| nom | prenom | nbroles |
|---|---|---|
| Eastwood | Clint | 2 |
| Sellers | Peter | 3 |

### 2.5.7 Exercices

Donner des requêtes pour :

- trouver les acteurs qui sont aussi des réalisateurs ;
- trouver les acteurs nés le 1er janvier 1930 ou après ;
- compter les acteurs nés avant le 1er janvier 1940 ;
- trouver les acteurs qui ont joué au moins dans un film où jouait Martin Freeman ;
- trouver les réalisateurs qui ont dirigé Clint Eastwood.