Les bases de données

Renaud Costadoat
Lycée Dorian

Bases de données

Definition

Une Base de données est un gros ensemble d'**informations** structurées **mémorisées** sur un support permanent.

Un système de fichier correspond à cette définition, cependant, l'utilisation directe de fichiers soulève des problèmes :

1. Lourdeur d'accès aux données,
2. Manque de sécurité,
3. Pas de contrôle de concurrence.

Definition

Un Système de Gestion de Bases de Données (SGBD) est un **logiciel** de haut niveau qui permet de manipuler les **informations** stockées dans une base de données.

Utilisation d'un SGBD

L'utilisation d'un SGBD suppose de comprendre (et donc de savoir utiliser) les fonctionnalités suivantes :

1. Définition du schéma de données en utilisant les modèles de données du SGBD,
2. Opérations sur les données : recherche, mises-à-jour, etc,
3. Partager les données entre plusieurs utilisateurs. (Mécanisme de transaction),
4. Optimiser les performances, par le réglage de l'organisation physique des données. Cet aspect relève plutôt de l'administration et ne sera évoqué que dans l'introduction.

Exemple de données

Extrait d'une base de données sur le cinéma.

Film	Année	Nom Réal	Prénom Réal	Année Réal
Fargo	1996	Coen	Joel	1953
Fargo	1996	Coen	Ethan	1957
Pulp Fiction	1994	Tarantino	Quentin	1963
Inglourious Basterds	2009	Tarantino	Quentin	1963
Inside Llewyn Davis	2013	Coen	Joel	1953
Inside Llewyn Davis	2013	Coen	Ethan	1957
RockNRolla	2008	Ritchie	Guy	1968

Avec un cas aussi simple, un ensemble de problèmes potentiels apparaît. Une des principales causes de problème vient du fait qu'il est possible de représenter la même information **plusieurs fois**.

Anomalies liées à l'utilisation d'une base de données

Anomalies lors d'une insertion

Il est possible de faire apparaître plusieurs fois le même film (*Fargo* apparaît deux fois). Une bonne question consiste d'ailleurs à se demander ce qui distingue deux films l'un de l'autre, et à quel moment on peut dire que la même information a été répétée. Peut-il y avoir deux films différents avec le même titre par exemple ?

Anomalies lors d'une modification

La redondance d'information entraîne également des anomalies de mise à jour. En modifiant l'année de naissance de *Quentin Tarantino* sur la ligne *Inglourious Basterds*, cela ne modifie pas la ligne *Pulp Fiction*. Des informations incohérentes apparaissent.

Anomalies lors d'une destruction

Il est impossible de supprimer un film sans supprimer du même coup son metteur en scène. Si on souhaite, par exemple, ne plus voir le film *RockNRolla* figurer dans la base de données, on va effacer du même coup les informations sur *Guy Ritchie*.

Distinctions des tables

Une bonne méthode évitant les anomalies ci-dessus consiste à:

1. Etre capable de représenter individuellement les films et les réalisateurs, de manière à ce qu'une action sur l'un n'entraîne pas systématiquement une action sur l'autre,
2. Définir une méthode d'identification d'un film ou d'un réalisateur, qui permette d'assurer que la même information est représentée une seule fois,
3. Préserver le lien entre les films et les réalisateurs, mais sans introduire de redondance.

Table des films

Id	Film	Année
1	Fargo	1996
2	Pulp Fiction	1994
3	Inglourious Basterds	2009
4	Inside Llewyn Davis	2013
5	RockNRolla	2008

Table des réalisateurs

Id	Nom	Prénom	Année
1	Coen	Joel	1953
2	Coen	Ethan	1957
3	Tarantino	Quentin	1963
4	Ritchie	Guy	1968

Lien entre les tables

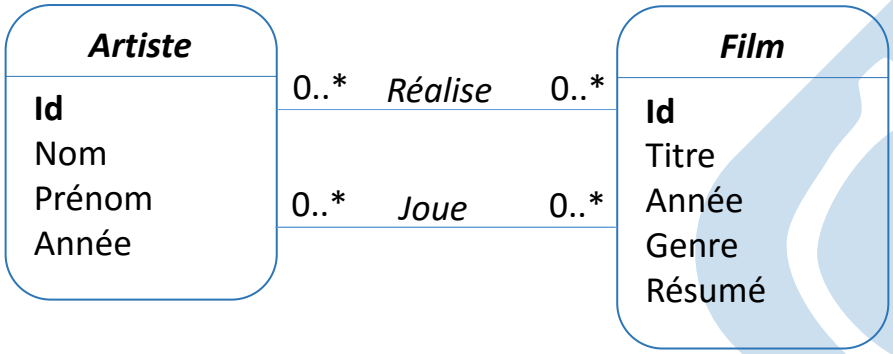
Il reste à représenter le lien entre les films et les metteurs en scène, sans introduire de redondance. Maintenant que nous avons défini les identifiants, il existe un moyen simple pour indiquer quel est le metteur en scène qui a réalisé un film : associer l'identifiant du metteur en scène au film. On ajoute des attributs *IdReal1* et *IdReal2* dans la *Table des films*, et on obtient la représentation suivante.

Table des films

Id	Film	Année	IdReal1	IdReal2
1	Fargo	1996	1	2
2	Pulp Fiction	1994	3	
3	Inglourious Basterds	2009	3	
4	Inside Llewyn Davis	2013	1	2
5	RockNRolla	2008	4	

Schéma base de données

Le schéma suivant permet de décrire la base de donnée et les liens entre les bases.



Les associations sont caractérisées par des cardinalités. La notation *0..** sur le lien *Réalise*, du côté de l'entité Artiste, signifie qu'un film peut être réalisé par plusieurs artistes, ou aucun. Une notation *0..1* signifierai en revanche qu'un film ne peut être réalisé que par au plus un artiste, *1..1* signifierai un seul artiste.

Les **clés** sont indiqués en gras, elles permettent d'identifier de manière unique une instance de la table.

Tables de jointure

La mise en place de tables de jointure permet de retrouver les relations entre les instances des tables.

Table des réalisateurs

Id	IdFilm	IdReal1	IdReal2
1	1	1	2
2	2	3	
3	3	3	
4	5	4	
5	4	1	2

Définition d'un schéma relationnel

Une relation est caractérisée par un **nom** et se compose d'un ensemble de colonnes désignées par un nom d'**attribut**. Dans chaque colonne on trouve des valeurs d'un certain **domaine** (chaînes de caractères, nombres). Enfin on constate que chaque ligne (ou tuple) correspond à une **entité**.

Ainsi pour le schéma relationnel suivant:
Film (titre: string, année: number, genre : string)

- *Film* est le **nom**,
- *titre*, *année* et *genre* sont des **attributs**,
- *sting* et *number* sont des **domaines**.

('Fargo',1996,'Thriller') est un **tuple**.

Types du langage de données SQL

Type	Description	Taille
INTEGER	Type des entiers relatifs	4 octets
SMALLINT	Idem.	2 octets
BIGINT	Idem.	8 octets
FLOAT	Flottants simple précision	4 octets
DOUBLE PRECISION	Flottants double précision	8 octets
REAL	Synonyme de FLOAT	4 octets
NUMERIC (M, D)	Numérique avec précision fixe.	M octets
DECIMAL (M, D)	Idem.	M octets
CHAR(M)	Chaînes de longueur fixe	M octets
VARCHAR(M)	Chaînes de longueur variable	$L+1$ avec $L \leq M$
BIT VARYING	Chaînes d'octets	Longueur de la chaîne
DATE	Date (jour, mois, an)	env. 4 octets
TIME	Horaire (heure, minutes, secondes)	env. 4 octets
DATETIME	Date et heure	8 octets
YEAR	Année	2 octets

Langage de définition de données SQL

Langage de définition de données SQL

```
CREATE TABLE eleve (id INT (4) NOT NULL,
                    email VARCHAR (50) NOT NULL,
                    nom VARCHAR (20) NOT NULL,
                    prenom VARCHAR (20),
                    motDePasse VARCHAR (60) NOT NULL,
                    classe VARCHAR (4) DEFAULT 'PTSI',
                    lv1 INT (1) NOT NULL,
                    PRIMARY KEY (Id),
                    FOREIGN KEY (lv1))
```

Clés primaire (`PRIMARY KEY`)

Une clé est un attribut qui identifie de manière unique un tuple d'une relation.

Clés étrangères (`FOREIGN KEY`)

La norme SQL ANSI permet d'indiquer quelles sont les clés étrangères dans une table, autrement dit, quels sont les attributs qui font référence à une ligne dans une autre table.

Langage de définition de données SQL

Modification d'une table: Ajout d'un champ

```
ALTER TABLE Internaute ADD region VARCHAR(10)
```

Modification d'une table: Modification d'un champ

```
ALTER TABLE Internaute MODIFY region VARCHAR(30) NOT NULL
```


Application

Proposer le code de création d'une table permettant de gérer la lv1. Les attributs seront:

- un *id* (clé primaire),
- un *nom*,
- une *salle*.

Requêtes SQL

La suite va prendre pour exemple la base de données des *Villes de France* dont le schéma relationnel est le suivant:

Ville (ville_id:int, ville_departement:int, ville_nom_reel=: char, ville_code_postal=int, ville_population_2010=int, ville_longitude_dms:int , ville_latitude_dms:int) (d'autres colonnes sont présentes mais elles ne nous intéressent pas ici).

- `FROM` indique la (ou les) tables dans lesquelles on trouve les attributs utiles à la requête,
- `SELECT` indique la liste des attributs constituant le résultat (* pour tout sélectionner),
- `WHERE` indique les conditions que doivent satisfaire les n-uplets de la base pour faire partie du résultat.

Requêtes SQL

Lister les éléments de la base

```
SELECT ville_nom_reel, ville_population_2010
FROM villes_france_free
WHERE ville_code_postal=40990
```

ville_nom_reel	ville_population_2010
Saint-Vincent-de-Paul	2997
Téthieu	656
Gourbera	356
Herm	1044
Mées	1736
Saint-Paul-lès-Dax	12409
Angoumé	288

Requêtes SQL

Modifier l'affichage des attributs.

```
SELECT ville_nom_reel as 'Ville',
ville_population_2010/1000 AS 'Population en milliers'
FROM villes_france_free
WHERE ville_code_postal=40990 and ville_population_2010>1000
```

Ville	Population en milliers
Herm	1.0440
Mées	1.7360
Saint-Vincent-de-Paul	2.9970
Saint-Paul-lès-Dax	12.4090

Requêtes SQL

Recherche dans une table.

```
SELECT ville_nom_reel as 'Ville',ville_code_postal as 'Code Postal'
FROM `villes_france_free`
WHERE `ville_nom_reel`='Saint-Vincent-de-Paul'
```

Ville	Code Postal
Saint-Vincent-de-Paul	33440
Saint-Vincent-de-Paul	40990

Éviter les doublons.

```
SELECT DISTINCT ville_nom_reel as 'Ville'
FROM `villes_france_free`
WHERE `ville_nom_reel`='Saint-Vincent-de-Paul'
```

Ville
Saint-Vincent-de-Paul

Requêtes SQL

Affichage trié:

```
SELECT ville_nom_reel, ville_population_2010
FROM villes_france_free
WHERE ville_code_postal=40990 ORDER BY ville_population_2010
```

ville_nom_reel	ville_population_2010
Angoumé	288
Gourbera	356
Téthieu	656
Herm	1044
Mées	1736
Saint-Vincent-de-Paul	2997
Saint-Paul-lès-Dax	12409

Le tri dans l'ordre descendant s'effectue en ajoutant DESC, à la suite des paramètres de classement.

Requêtes SQL

Calcul de somme:

```
SELECT SUM(ville_population_2010) AS 'Population Totale 40990'
FROM villes_france_free
WHERE ville_code_postal=40990
```

Population Totale 40990
19486

Calcul de moyenne:

```
SELECT AVG(ville_population_2010) AS 'Population Moyenne 40990'
FROM villes_france_free
WHERE ville_code_postal=40990
```

Population Moyenne 40990
2783.7143

Requêtes SQL

Compter les valeurs:

```
SELECT COUNT(ville_nom_reel) AS 'Nombre de villes'
FROM villes_france_free
WHERE ville_code_postal=40990
```

Nombre de villes
7

```
SELECT ville_departement AS 'Département',
COUNT(ville_id) as 'Nb villes'
FROM villes_france_free
WHERE ville_departement > 40
    AND ville_departement < 45
GROUP BY ville_departement
```

Département	Nb villes
41	291
42	327
43	260
44	221

Requêtes SQL

Limiter le nombre de réponse

```
SELECT ville_nom_reel AS Ville,
    ville_population_2010 AS Population
FROM villes_france_free
ORDER BY ville_population_2010 DESC LIMIT 1
```

Ville	Population
Paris	2243833

Rechercher le maximum d'une liste

```
SELECT ville_nom_reel AS Ville,
    ville_population_2010 AS Population
FROM villes_france_free
WHERE ville_population_2010=
    (SELECT max(ville_population_2010)
        FROM villes_france_free)
```

Ville	Population
Paris	2243833

Requêtes SQL

Recherche de plusieurs maximums.

```
SELECT ville_departement AS Département, ville_nom_reel AS Ville,
    ville_population_2010 AS Population
FROM villes_france_free v,
    (SELECT max(ville_population_2010) AS Max FROM villes_france_free
        WHERE ville_departement >=40 AND ville_departement <45
            GROUP BY ville_departement) m
WHERE v.ville_population_2010=m.Max
```

Département	Ville	Population
40	Mont-de-Marsan	31225
41	Blois	46492
42	Saint-Étienne	171260
43	Le Puy-en-Velay	18521
44	Nantes	284970

Requêtes SQL

Utilisation de `HAVING` pour limiter les résultats avec des `SUM` , `AVG` ,...

```
SELECT ville_departement AS Département,
    SUM(ville_population_2010) AS Population
FROM villes_france_free
GROUP BY ville_departement
HAVING SUM(ville_population_2010)<100000
```

Département	Population
48	77082
975	6080

Requêtes SQL

Des tables sont ajoutées afin de gérer les régions et les départements.

```
SELECT departement_nom as Département, departement_code as Numéro
FROM `departement`
WHERE departement_code < 50 AND departement_code >= 40
```

Département	Numéro
Landes	40
Loir-et-Cher	41
Loire	42
Haute-Loire	43
Loire-Atlantique	44
Loiret	45
Lot	46
Lot-et-Garonne	47
Lozère	48
Maine-et-Loire	49

Requêtes SQL

```
SELECT `Id_region`,`NCCENR` AS Nom
FROM `regions`
WHERE `Id_region`>40
```

Id_region	Nom
44	Alsace-Champagne-Ardenne-Lorraine
52	Pays de la Loire
53	Bretagne
75	Aquitaine-Limousin-Poitou-Charentes
76	Languedoc-Roussillon-Midi-Pyrénées
84	Auvergne-Rhône-Alpes
93	Provence-Alpes-Côte d'Azur
94	Corse

Requêtes SQL

```
SELECT * FROM `jointure departement-region` WHERE id_region=75
```

id_departement	id_region
16	75
17	75
19	75
23	75
24	75
33	75
40	75
47	75
64	75
79	75
86	75
87	75

Requêtes SQL

Des tables sont ajoutées afin de gérer les régions et les départements.

```
SELECT v.ville_nom_reel, d.departement_nom, v.ville_population_2010
FROM villes_france_free v
JOIN departement d ON d.departement_code=v.ville_departement
WHERE v.ville_population_2010>400000
ORDER BY v.ville_population_2010 DESC
```

ville_nom_reel	departement_nom	ville_population_2010
Paris	Paris	2243833
Marseille	Bouches-du-Rhône	850726
Lyon	Rhône	484344
Toulouse	Haute-Garonne	441802

Requêtes SQL

```
SELECT v.ville_nom_reel AS Ville, d.departement_nom AS Département,
r.NCCENR AS Région, v.ville_population_2010 AS Population
FROM villes_france_free v
JOIN departement d JOIN regions r JOIN jointure_departement_region j
    ON d.departement_code=v.ville_departement
    AND d.departement_code=j.id_departement
    AND r.Id_region=j.id_region
WHERE v.ville_population_2010>400000
ORDER BY v.ville_population_2010 DESC
```

Ville	Département	Région	Population
Paris	Paris	Ile de France	2243833
Marseille	Bouches-du-Rhône	Provence-Alpes-Côte d'Azur	850726
Lyon	Rhône	Auvergne-Rhône-Alpes	484344
Toulouse	Haute-Garonne	Languedoc-Roussillon-Midi-Pyrénées	441802

Requêtes SQL

Modification d'une table (gestion de la LV1).

Insérer une ligne.

```
INSERT INTO lv1 VALUES (1, 'Anglais', 'B302')
```

Modifier une ligne.

```
UPDATE lv1 SET salle='B306' WHERE Id=1
```

Supprimer une ligne.

```
DELETE FROM lv1 WHERE Id=1
```