Bases de données, algèbre relationnelle et SQL

Delphine Salvy

Séminaire Cr@ns

23 octobre 2018

I. Les bases de données

Introduction

Une base de données (BDD) est un ensemble de données stockées sur des supports (informatiques) qu'un groupe d'*utilisateurices* peut interroger (en faisant des *requêtes*) ou modifier.

Conception d'une base de données : un mauvais exemple

titre	realisateur
La vie est belle	Benigni
La vie est belle	Capra
Beau-père	Blier
L'argent de poche	Truffaut
L'amour en fuite	Truffaut
Domicile conjugal	Trufaut
Mauvaise graine	Wilder
Les cinq secrets du désert	Wilder

Table – Base de données d'une DVDthèque

Conception d'une base de données : un mauvais exemple

titre	realisateur
La vie est belle	Benigni
La vie est belle	Capra
Beau-père	Blier
L'argent de poche	Truffaut
L'amour en fuite	Truffaut
Domicile conjugal	Trufaut
Mauvaise graine	Wilder
Les cinq secrets du désert	Wilder

Table – Base de données d'une DVDthèque

- Problèmes d'unicité (La vie est belle)

Conception d'une base de données : un mauvais exemple

titre	realisateur
La vie est belle	Benigni
La vie est belle	Capra
Beau-père	Blier
L'argent de poche	Truffaut
L'amour en fuite	Truffaut
Domicile conjugal	Trufaut
Mauvaise graine	Wilder
Les cinq secrets du désert	Wilder

Table – Base de données d'une DVDthèque

- Problèmes d'unicité (La vie est belle)
- Présence de redondances (Truffaut)

Conception d'une base de données : un bon exemple

films		
id	titre	real
1	La vie est belle	1
2	La vie est belle	3
3	Beau-père	2
4	L'argent de poche	5
5	L'amour en fuite	5
6	Domicile conjugal	5
7	Mauvaise graine	4
8	Les cinq secrets du désert	4

Table – Table films

realisateurs	
id	nom
1	Benigni
2	Blier
3	Capra
4	Wilder
5	Truffaut

Table – Table realisateurs

Conception d'une base de données : un bon exemple

films		
id	titre	real
1	La vie est belle	1
2	La vie est belle	3
3	Beau-père	2
4	L'argent de poche	5
5	L'amour en fuite	5
6	Domicile conjugal	5
7	Mauvaise graine	4
8	Les cinq secrets du désert	4

Table – Table films

realisateurs	
id	nom
1	Benigni
2	Blier
3	Capra
4	Wilder
5	Truffaut

Table – Table realisateurs

- Présence d'une clé primaire
- Mise en relation de deux tables (= relations)

Un peu de vocabulaire

- La table `films` possède trois attributs : `id, titre, real`
- Chaque attribut prend des valeurs dans un domaine. L'attribut `id` est un entier (INTEGER), l'attribut `titre` est une chaîne de caractères (CHAR, VARCHAR, TEXT, ...).
- Schéma relationnel

  ```
  films(id : entier, titre : texte, real : entier)
  realisateurs(id : entier, nom : texte)
  ```

Résumé : grands principes de la conception d'une base de données

Éviter les ambiguïtés Utiliser une clé primaire, généralement un entier (INTEGER PRIMARY KEY) incrémenté de 1 à chaque insertion (AUTOINCREMENT).

Éviter les redondances Pour diminuer les risques d'erreur, pouvoir modifier plus facilement, ... Pour cela

- Ne jamais stocker une information que l'on peut obtenir avec les informations déjà présentes ;
- Séparer les informations dans plusieurs tables.

II. Installer SQL

SGBD & SQL

- Pour accéder à la base de données et réaliser ses requêtes, l'utilisateurice peut utiliser des formulaires, ou un langage conçu pour les bases de données : Structured Query Language (SQL).
- Le Système de Gestion de Base de Données (SGBD) est un logiciel qui s'occupe de répondre aux requêtes, réaliser les insertions et les suppressions, ...
- Parmi les principaux, on compte MySQL (open source, GPL ou propriétaire selon les cas), PostgreSQL (libre), Oracle Database (propriétaire), Microsoft SQL Server (propriétaire), ...

Tests avec SQLite

1. Téléchargez SQLite depuis le site officiel `https://www.sqlite.org/`
2. Téléchargez la base de données `https://perso.crans.org/salvy/vertebres.db` [1]
3. Lancez SQLite et tapez `.open chemin` où `chemin` est le chemin vers le fichier téléchargé.
4. Vous pouvez dès à présent tester `.tables` et `.dump`.

1. Crédit : Franz Ridde

III. L'algèbre relationnelle en SQL

Première commande

Syntaxe

```
SELECT *
FROM table
```

Example

```
SELECT *
FROM groupes
```

Première commande

Syntaxe

```
SELECT *
FROM table
```

Example

```
SELECT *
FROM groupes
```

Le SQL est un langage déclaratif.

Projection

Definition

Une *projection* consiste à ne garder que certains attributs (colonnes) de la base de données.

Syntaxe

```
SELECT attribut1, attribut2, ...
FROM table
```

Example

```
SELECT nomg
FROM groupes
```

Exercice

Question

Afficher l'identifiant et le nom des différents vertébrés.

Attention : pensez à finir toutes vos commandes par un point virgule sur SQLite !

```
etats(ide : entier, nome : texte)
groupes(idg : entier, nomg : texte)
vertebres(id : entier, groupe : entier, nom : texte,
        protection : texte, etat : entier)
```

Exercice

Question

Afficher l'identifiant et le nom des différents vertébrés.

Attention : pensez à finir toutes vos commandes par un point virgule sur SQLite !

```
etats(ide : entier, nome : texte)
groupes(idg : entier, nomg : texte)
vertebres(id : entier, groupe : entier, nom : texte,
        protection : texte, etat : entier)
```

Réponse

```
SELECT id, nom
FROM vertebres
```

Sélection

Definition

Une *sélection* consiste à ne garder que certaines entrées (lignes) de la table vérifiant une certaine condition.

Syntaxe

```
SELECT attribut(s)
FROM table
WHERE condition
```

Example

```
SELECT nom
FROM vertebres
WHERE id <= 10
```

Exercice

Question

Afficher l'identifiant et le nom des vertébrés du groupe 4.

Exercice

Question

Afficher l'identifiant et le nom des vertébrés du groupe 4.

Réponse

```
SELECT id, nom
FROM vertebres
WHERE groupe = 4
```

Union, intersection, différence

Definition

L'*union* de deux relations *de même schéma* (en SQL : avec le même nombre d'attributs) la relation obtenue en gardant le même schéma et en réunissant toutes les lignes *sans doublons*. On définit de même l'*intersection* et la *différence* ($R_1 \setminus R_2$).

Syntaxe

```
SELECT attribut(s)
FROM table1
UNION / INTERSECT / EXCEPT
SELECT attribut(s)
FROM table2
```

Example

```
SELECT id, nom
FROM vertebres
UNION
SELECT *
FROM groupes
```

Exercice

Question

Afficher les identifiants de `vertebres` et `groupes`. Que remarquez-vous ?

Exercice

Question

Afficher les identifiants de `vertebres` et `groupes`. Que remarquez-vous ?

Réponse

```
SELECT id
FROM vertebres
UNION
SELECT idg
FROM groupes
```

Les doublons sont supprimés !

Produit cartésien

Definition

Le *produit cartésien* $R_1 \times R_2$ de deux relations R_1 et R_2 la relation obtenue en prenant chaque occurrence de R_1 et en l'associant à chaque occurence de R_2.

Syntaxe

```
SELECT attribut(s)
FROM table1, table2
```

Example

```
SELECT id, nom, groupe, idg, nomg
FROM vertebres, groupes
```

Si R_1 et R_2 ont un attribut de même nom, on les préfixe par le nom de leur table.

Jointure

Definition

La *jointure* de deux relations R_1 et R_2 est la relation obtenue en sélectionnant les occurrences de $R_1 \times R_2$ qui vérifient une certaine condition.

Syntaxe

```
SELECT attribut(s)
FROM table1 JOIN table2
ON table1.attribut1 = table2.attribut2
```

Example

```
SELECT id, nom, groupe, idg, nomg
FROM vertebres JOIN groupes
ON vertebres.groupe = groupes.idg
```

Exercice

Question

Afficher tous les noms de vertébrés dont l'état est "En danger".

Exercice

Question

Afficher tous les noms de vertébrés dont l'état est "En danger".

Réponse

```
SELECT nom
FROM vertebres JOIN etats
ON vertebres.etat = etats.ide
WHERE nome = "En danger"
```

Jointures multiples

Syntaxe

```
SELECT attribut(s)
FROM table1
JOIN table2 ON table1.attribut1 = table2.attribut2
JOIN table3 ON table1.attribut3 = table3.attribut4
```

Example

```
SELECT id, nom, groupe, idg, nomg, ide, etat, nome
FROM vertebres
JOIN groupes ON vertebres.groupe = groupes.idg
JOIN etats  ON vertebres.etat = etats.ide
```

Exercice

Question

Afficher tous les oiseaux dont l'état est "En danger".

Exercice

Question

Afficher tous les oiseaux dont l'état est "En danger".

Réponse

```
SELECT nom
FROM vertebres
JOIN groupes ON vertebres.groupe = groupes.idg
JOIN etats ON vertebres.etat = etats.ide
WHERE nome = "En danger" AND nomg = "Oiseau"
```

Fonctions d'agrégation

Definition

Les fonctions d'agrégation agissent sur un attribut (ou toute la table) et renvoient un résultat unique pour toutes les lignes (ou groupes de lignes) sélectionnées.

Syntaxe

```
SELECT fonction(attribut)
FROM table
```

Pour compter toutes les lignes de la table, on utilise `COUNT(*)`. `COUNT(attribut)` compte le nombre de valeurs qui ne valent pas NULL dans la colonne.

Example

```
SELECT MAX(id) - MIN(id) + 1
FROM vertebres
```

Exercice

Question

Combien y a-t-il de groupes de vertébrés ?

Exercice

Question

Combien y a-t-il de groupes de vertébrés ?

Réponse

```
SELECT COUNT(*)
FROM groupes
```

Groupes (agrégats)

La clause GROUP BY permet de former des *groupes* dans une table.

Syntaxe

```
SELECT attribut, fonction(attribut)
FROM table
GROUP BY attribut
HAVING condition
```

Attention ! L'attribut sur lequel porte la fonction d'agrégation doit avoir la même valeur pour toutes les entrées du groupe.

Example

```
SELECT nomg, COUNT(*)
FROM groupes JOIN vertebres
ON vertebres.groupe = groupes.idg
GROUP BY nomg
```

Exercice

Question

Afficher le plus grand nombre de vertébrés dans un même groupe.

Exercice

Question

Afficher le plus grand nombre de vertébrés dans un même groupe.

Réponse

```
SELECT MAX(nbvertebres)
FROM
        (
        SELECT COUNT(*) AS nbvertebres
        FROM vertebres
        JOIN groupes ON vertebres.groupe = groupes.idg
        GROUP BY idg
        )
```

Ordonner les réponses

La clause ORDER BY permet de trier ce que renvoie la requête.

Syntaxe

```
requête
ORDER BY attribut ASC / DESC
```

Example

```
SELECT nom
FROM vertebres
ORDER BY nom ASC
```

Exercice

Question

Afficher les groupes par nombre décroissant de vertébrés en leur sein.

Exercice

Question

Afficher les groupes par nombre décroissant de vertébrés en leur sein.

Réponse

```
SELECT nomg
FROM
        (
        SELECT nomg, COUNT(*) AS nbvertebres
        FROM vertebres
        JOIN groupes ON vertebres.groupe = groupes.idg
        GROUP BY idg
        )
ORDER BY nbvertebres DESC
```

Un dernier exercice - question

Question

Afficher le groupe avec le plus grand nombre de vertébrés.

Un dernier exercice - réponse

Réponse

```
SELECT nomg
FROM vertebres
JOIN groupes ON vertebres.groupe = groupes.idg
GROUP BY idg
HAVING COUNT(*) =
        (
        SELECT MAX(nbvertebres)
        FROM
        (
        SELECT COUNT(*) AS nbvertebres
        FROM vertebres
        JOIN groupes ON vertebres.groupe = groupes.idg
        GROUP BY idg
        )
        )
```

IV. Conclusion

Conclusion

Forme générale

```
SELECT attributs, fonction(attribut)
FROM table1
JOIN table2 ON table1.attribut1 = table2.attribut2
WHERE conditions
GROUP BY attributs
HAVING conditions
ORDER BY attribut ASC/DESC
```

Référence : Franz Ridde, *Base de données*