Introduction application web

Master IMST "Système d'information et d'adchivage numérique" (IMST-SIAN)

Fabien Rico

Univ. Claude Bernard Lyon 1

21 octobre 2022

Plan

Introduction

Définition (Base de données)

Une *base de données* est un système permettant de stocker des informations reliées entre elles de manière plus ou moins structurée. Les données sont alors accessibles au moyen d'un logiciel.

Définition (SGBD)

Les *Système de gestion de base de données* sont les logiciels qui organisent l'acces à ces données, leur manipulation et effectuent le stockage.

Exemple : BdD relationnelles

- Très structurées, les données sont des n-upplets stockées dans des tableaux
  - Chaque ligne est un *enregistrement* ;
  - Les colonnes sont des *attributs* typés.
- Les relations entres les enregistrements sont matérialisées dans la base et peuvent être maintenues et vérifiées.
- Utiles pour maintenir une cohérence entre les données
- Demande une centralisation de certaines opérations, il est difficile de répartir les opérations sur plusieurs serveurs
- Limite dans la taille des données gérées

Exemple : BdD NoSQL

- Beaucoup moins structurées
  - Les enregistrements sont stockés dans des structures qui peuvent être plus complexes qu'un tableau ;
  - Les shémas sont adaptables ou absents.
- Le système n'est pas relationnel.
- Il est plus facile de répartir les données.
- Utilisés dans les bases les plus grandes.
- Il est plus difficile ou couteux de gérer les relations entre enregistrements.

Le language SQL

Définition (Structured Query Language)

Le SQL est le langage d'interogation et de manipulation des bases de données relationnelles. Il permet

- la manipulation de données (`SELECT`, `UPDATE`, `INSERT`, `DELETE`)
- la définitions des tables (`CREATE`, `ALTER`, `DROP`, ...)
- le contrôle de l'acces au données (`GRANT`, `REVOKE`)
- le controle des transaction (`COMMIT`, `ROLLBACK`)

Les différents SGBD implémentent plus ou moins ce langage et avec des nuances dans les résultats. Cela occasionne des problèmes lors du changement de SGBD dans un programme. Nous étudierons essentiellement les lectures de données.

Requêtes de lecture

```
SELECT ... FROM ... WHERE ... ORDER BY ... LIMIT ...
```

Si on suppose une requête simple sur une seule table de la base, la requête fait 3 opérations simples :

- *la projection* : pour chaque élément de la table on ne récupère que certains champs grâce au `SELECT` ;
- *le filtrage* : chaque élément est conservé ou non grâce aux conditions du champ `WHERE` ;
- *le tri* : les résultats sont ordonnés grâce à `ORDER BY` et paginés grâce à `LIMIT`.

La projection

```
SELECT nomduchampbase AS nomutilise,
       OPERATION(nomautrechamp) AS autrenom ...
```

- On sélectionne les champs interessants.
- A cette occasion, on peut renomer les champs grâce à `AS`.
- Il est aussi possible d'utiliser des opérations pour construire des valeurs.

Filtrage

```
WHERE champ_str1 = 'une valeur'
  AND (champ_str2 LIKE '%toto%'
       OR champ_int > 10)
```

- La condition est une formule logique sur les champs des enregistrements.
- Plusieurs opérateur existent :
  - Les comparaisons =, !=, >, >=,...
  - Les comparaisons avec carractères générique `LIKE`, `NOT LIKE`
    - `%` remplace n'importe quel chaine de carractère ;
    - `_` remplace n'importe quel carractère.
  - Les expressions régulières `REGEXP`, `NOT REGEXP`

Les tris

```
ORDER BY champs1 ASC,
         champs2 DESC
LIMIT numdebut, nbres
```

- `ORDER BY` permet d'ordonner les résultats selon les valeurs de différents champs
  - `ASC` signifie par ordre croissant
  - `DESC` signifie par ordre décroissant
- `LIMIT` permet de donner a position du premier résultat voulu et le nombre de résultat demandés
  - par exemple `LIMIT 5, 4` signifie 4 valeurs à partir de la numéro 5
  - cela permet de paginer (lire les valeurs par groupes)

TPSql exercices sur les requêtes simples

Requête sur plusieurs tables

```
... FROM table1, table2, ...
```

- Au départ, c'est un produit cartésien :
  - par exemple 2 tables de 100 enregistrements donnent 10 000 possibilités
  - 3 tables de 100 enregistrements donnent 1 000 000
  - ...
- On filtre souvent par des clefs : ce sont les *jointures*
  - Avoir les achats réalisés par les clients demande de lister les achats et de relier l'identifiant de chaque acheteur avec ses informations.
  - Avoir les autrices de tous les livres demande pour chaque livre de le relier à une liste d'autrices.

Jointure

Définition (jointure)

C'est l'association de 2 tables par l'utilisation d'un lien logique. Souvent ce lien est matérialisé par la présence d'un identifiant des enregistrements commun dans les deux tables.

Par exemple dans une bibliothèques, les exemplaires d'un ouvrage :

- un ouvrage peu avoir plusieurs exemplaires ;
- un exemplaire est forcement associé à un ouvrage ;
- pour le matérialiser, il suffit d'ajouter un champ « identifiant de l'ouvrage » dans chaque enregistrement de la table exemplaires.

Comment faire la jointure

- En utilisant les filtres

```
SELECT * FROM ouvrages, exemplaires
       WHERE ouvrages.id = exemplaires.id_ouvrage
```

- Directement dans le `FROM`

```
SELECT *
  FROM ouvrages JOIN exemplaires
               ON ouvrages.id=exemplaires.id_ouvrage
 WHERE ...
```

Mais que se passe-t-il si des ouvrages référencés n'ont pas d'exemplaire dans la base ?

Jointure externe

La seconde méthode permet de mieux spécifier la jointure :

- Jointure interne, seuls les enregistrements présents dans les 2 tables sont retournés

```
SELECT * FROM tabA JOIN tabB ON tabA.id_A=tabB.id_A WHERE 1
```

- Jointure externe à gauche, les enregistrements de la table A sans correspondance dans la table B sont retournés en plus

```
SELECT * FROM tabA LEFT JOIN tabB ON tabA.id_A=tabB.id_A WHERE 1
```

- Jointure externe à droite, les enregistrements de la table B sans correspondance dans la table A sont retournés en plus

```
SELECT * FROM tabA RIGHT JOIN tabB ON tabA.id_A=tabB.id_A WHERE 1
```

- Jointure externe complete, les enregistrements d'une tables sans correcpondance dans l'autre sont retournés

```
SELECT * FROM tabA FULL JOIN tabB ON tabA.id_A=tabB.id_A WHERE 1
```

Exercice sur les jointures simples

Quel est le cout du calcul ?

Supposons une bibliothèque référençant 10000 ouvrages et 20000 exemplaires de ces ouvrages.
Si on effectue la jointure, combien faut-il faire d'opération de comparaison ?

Notion d'index

Définition (Index)

Un index est une structure de données associée à un ou plusieurs attributs dans une table. Elle est entretenue par le SGBD à chaque modification et permet de retrouver rapidement les données en fonction de ce ou ces champs.
Cela accélère les opérations de recherche, de tri, de jointure...

- Il y a plusieurs façon de faire des indexes.
- Il y a des indexes primaires, uniques, normaux.
- Des exemples dans la vie courante ?
- Pourquoi ne crée-t-on pas un index sur tous les attributs ?

Comment représenter les associations ?

Les associations ou relations entre les objets stockés doivent être représentées dans la base. On il y en a différent type selon la cardinalité :

- *Relation 1-1* : Un ouvrage a un titre unique et le titre est associé à un unique ouvrage.
- *Relation 1-n* : Un ouvrage a des exemplaires mais chaque exemplaire est associé à un seul ouvrage.
- *Relation n-n* : Un ouvrage a une ou plusieurs autrices, et une autrice peut avoir écrit plusieurs ouvrages.

Représentation des relations

- Pour les relations 1-1, on fusionne les tables, par exemple, le titre fait partie des attributs de la table ouvrage.
- Pour les relations 1-n, on peut ajouter l'identifiant de la première table dans les attributs de la seconde. Par exemple, chaque exemplaire aura l'identifiant de l'ouvrage correspondant.
- Pour les relation n-n, on peut utiliser une table annexe qui contient comme attributs les identifiants des 2 tables. Par exemple la table `A_ecrit` donc chaque enregistrement contient l'identifiant d'un auteur et d'un ouvrage.

Exercices