Université de Liège

Seconde partie du projet

INFO0009-2 – Base de données

Simon Bernard (s161519)
Ivan Klapka (s165345)
François Rozet (s161024)

3ème année de Bachelier Ingénieur civil

Année académique 2018-2019

1 Arti-clever

1.1 Emplacement

Le site web *Arti-clever* est actuellement hébergé à l'adresse suivante :

http://www.student.montefiore.ulg.ac.be/ s161024/

1.2 Authentification

Pour accéder à la page d'accueil du site, et par la même occasion tous ses sous-dossiers, il est nécessaire de remplir une requête d'authentification[1] `http`. Ce sont les fichiers `.htaccess` et .htpasswd présents à la racine du site qui permettent ce type de protection. Il est à noter que l'emplacement *absolu* du fichier .htpasswd est renseigné dans le `.htaccess`, ce qui signifie que pour déplacer le site, il est nécessaire de modifier ce fichier.

1.3 Architecture

Lorsque l'authentification est réussie, l'utilisateur est dirigé vers la page d'accueil du site ; qui est en réalité sa seule page. Elle est séparée en plusieurs sections répondant chacune à une des sous-questions.

Cette page d'accueil est un fichier `HTML` dynamisée à l'aide de `JavaScript`. Notamment, tous les accès à la base de données sont réalisés par des requêtes `AJAX` (cf. répertoire `resources/js/`) vers des scripts `PHP` et, en cas de succès, les résultats sont ajoutés dynamiquement au contenu de la page. Cela à pour avantage non négligeable de ne pas devoir recharger la page à chaque requête.

Les scripts `PHP` contenus dans le répertoire `include/`, exécutent des requêtes `SQL`, pour la plupart, dépendantes des arguments fournis par méthode `get`[2]. Lorsque les requêtes ne nécessitent pas d'arguments, elles sont directement rédigées dans des scripts `.sql` contenus dans le répertoire `resources/sql/`.

Le site utilise la bibliothèque de styles Bootstrap.

1.4 Initialisation

Initialiser ou réinitialiser la base de données se fait en trois étapes représentées par les trois scripts `create.sql`, `delete.sql` et `load.sql` dans le répertoire `resources/sql/initialization/`.

Le premier créée les tables si elles n'existent pas, le deuxième supprime les éventuelles données de toutes les tables et le troisième charge les données depuis les `.csv` du répertoire `resources/csv/` dans les tables correspondantes.

1. Les identifiants sont ceux de la base de données.
2. Nous avons préféré la méthode `get` à la méthode `post` car la première permet d'introduire des arguments dans l'URL manuellement.

Bien qu'il soit parfaitement envisageable de copier-coller manuellement ces scripts dans un terminal, nous avons implémenté le script `include/initialize.php` pour le faire à notre place. Il suffit donc de s'y rendre, pour initialiser la base de données.

2 Requêtes

Mis à part pour la question 2.c, les requêtes ne font que lire dans la base de données. Il n'est donc pas nécessaire de vérifier les contraintes d'intégrité.

2.a Recherche par contraintes

La requête est exécutée par le script `include/questions/q2a.php`.

```
18 $sql = "SELECT * FROM $tablename WHERE ";
19
20 foreach ($desc as $attr) {
21     if (isset($_GET[$attr[0]])) {
22
23         $pos = strpos($attr[1], 'int');
24         if (!$pos && $pos !== 0) {
25             $sql .= $attr[0]." COLLATE UTF8_GENERAL_CI LIKE
                   '%".$_GET[$attr[0]]."%'";
26         } else {
27             $sql .= $attr[0]." = ".$_GET[$attr[0]];
28         }
29         $sql .= " AND ";
30     }
31 }
32
33 $sql .= "1;";
```

Listing 1 – `q2a.php` – Construction de la requête `SQL`

Ce dernier reçoit, par l'intermidiaire du script `resources/js/questions/q2a.js`, un tableau (`$_GET`) dont les couples *clé-valeur* définissent les contraintes sur la recherche.

La valeur associée à la clé `table` est le nom de la table dans laquelle sont recherchés les éléments et les autres clés sont les noms des attributs contraints. Il est à noter que seuls sont considérés les couples dont la clé correspond à un attribut de la table.

Si un attribut est un nombre, c.-à-d. si son *type* dans la description (`DESC tablename;`) de la table contient la chaîne `int`, nous utilisons une contrainte d'égalité (`= value`) et, le cas échéant, une contrainte de contenance (`LIKE '%value%'`). Pour ne pas tenir compte de la casse, la commande `COLLATE UTF8_GENERAL_CI` est utilisée.

En finalité, nous obtenons une requête du type

```
SELECT * FROM $tablename WHERE $attr0 = $value0 AND $attr1 COLLATE
    UTF8_GENERAL_CI LIKE '%$value1%' AND ... AND 1;
```

Le `1` terminal n'est normalement pas nécessaire, mais il a été ajouté pour neutraliser le dernier `AND`.

2.b Ensemble de publications

La requête est exécutée par le script `include/questions/q2b.php`. Il est décomposé en deux parties.

2.b.1 Publications

La première partie permet d'obtenir l'ensemble de publications de l'auteur dont le matricule est `$matricule` (initialement `$_GET['matricule']`).

```
16 SELECT titre, date_publication, type, url
17 FROM
18     (
19         SELECT url, titre, date_publication
20         FROM articles WHERE matricule_auteur = $matricule
21     ) AS T1
22     NATURAL JOIN
23     (
24         SELECT url, 'journal' AS type
25         FROM articles_journaux
26         UNION
27         SELECT url, 'conference' AS type
28         FROM articles_conferences
29     ) AS T2
30 ORDER BY date_publication DESC;
```

Listing 2 – `q2b.php` - Première requête `SQL`

La table `T1` est l'ensemble des articles dont il est auteur et la table `T2` lie à chaque article de la base de données un type (`journal` ou `conference`). Dès lors, le `NATURAL JOIN` entre ces deux tables est ce que nous souhaitons.

2.b.2 Seconds auteurs

Ensuite, pour chaque article trouvé, on rechercher ses seconds auteurs à partir de son URL.

```
41 SELECT CONCAT(' ', LEFT(prenom, 1), '.', nom)
42 FROM auteurs
43 NATURAL JOIN
44 (
45     SELECT matricule
46     FROM seconds_auteurs
47     WHERE url = '$url'
48 ) AS T1;
```

Listing 3 – q2b.php – Seconde(s) requête(s) SQL

La table T1 est donc l'ensemble des matricules des seconds auteurs de l'article dont l'URL est $url. Son NATURAL JOIN avec la table articles nous permet d'obtenir leurs nom et prénom que nous concaténons en un seul champ pour faciliter l'affichage.

Ensuite, le champ contenant l'URL dans la table précédente est remplacé par cette nouvelle table et ce pour chaque article.

2.c Nouvel article

La requête est exécutée par le script include/questions/q2c.php. Elle est composée de deux parties.

2.c.1 Pre-transaction

Avant d'initialiser la transaction et de verrouiller les tables pour écriture, certaines contraintes sont vérifiées (cf. lignes 60 à 110).

Il est vérifié que

- L'auteur existe. C.-à-d. vérifier si le résultat de

  ```
  SELECT matricule FROM auteurs WHERE matricule = '$matricule_auteur';
  ```

  n'est pas vide.

- La conférence existe. C.-à-d. vérifier si le résultat de

  ```
  SELECT * FROM conferences WHERE nom = '$nom_conference' AND annee =
      '$annee_conference';
  ```

  n'est pas vide.

- La journal existe. C.-à-d. vérifier si le résultat de

  ```
  SELECT * FROM articles_journaux WHERE nom_revue = '$nom_revue' AND
      n_journal = '$n_journal';
  ```

  n'est pas vide.

  Pour ne pas multiplier les appels à la base de données, cette requête est couplée avec une autre pour obtenir l'année de publication du journal.

```
SELECT date_publication FROM articles NATURAL JOIN $type WHERE nom_revue
    = '$nom_revue' AND n_journal = '$n_journal' LIMIT 1;
```

- L'année de la conférence ou celle du journal (cf. au dessus) est l'année du nouvel article.
- La page de fin d'un article de journal est au moins égale à la page de début. [3]

2.c.2 Transaction

Si la première partie est un succès, nous déclarons le début de la transaction et nous verrouillons les tables nécessaires.

```
116 $pdo->beginTransaction();
117
118 // LOCK
119 $sql = "LOCK TABLES articles WRITE, $type WRITE";
120
121 if (isset($sujets_articles)) {
122     $sql .= ", sujets_articles WRITE";
123 }
124 if (isset($seconds_auteurs)) {
125     $sql .= ", seconds_auteurs WRITE, auteurs READ";
126 }
127
128 $sql .= ";";
```

Listing 4 – q2c.php - Transaction et verrouillage des tables

Dans le cas d'un article de journal sans sujets mais avec des seconds auteurs, nous avons

```
LOCK TABLES articles WRITE, articles_journaux WRITE, seconds_auteurs
    WRITE, auteurs READ;
```

Ensuite, nous vérifions si l'article n'existe pas. C.-à-d. si

```
SELECT * FROM articles WHERE url = '$url' OR doi = '$doi';
```

est bien vide. Cette vérification *doit* être effectuée après le verrouillage pour garentir l'intégrité.

Finalement, les informations liées à l'article peuvent être ajoutées aux tables. Sont choisies les requêtes nécessaires parmi les suivantes :

articles

```
INSERT INTO articles (url, doi, titre, date_publication, matricule_auteur)
    VALUES ('$url', $doi, '$titre', '$date_publication', $matricule_auteur);
```

articles_conferences

3. Il faudrait normalement vérifier que deux articles dans un même journal ne se chevauchent pas. Néanmoins, ce n'est pas une des contraintes de l'énoncé, nous ne l'avons donc pas implémenté.

```
INSERT INTO articles_conferences (url, presentation, nom_conference,
    annee_conference) VALUES ('$url', '$presentation', '$nom_conference',
    $annee_conference);
```

articles_journaux

```
INSERT INTO articles_journaux (url, pg_debut, pg_fin, nom_revue,
    n_journal) VALUES ('$url', $pg_debut, $pg_fin, '$nom_revue',
    $n_journal);
```

sujets_articles

```
INSERT INTO sujets_artilces (url, sujet) VALUES ('$url', '$sujet0'),
    ('$url', '$sujet1'), ...;
```

seconds_auteurs

```
INSERT INTO seconds_auteurs (url, matricule) VALUES ('$url', $matricule0),
    ('$url', $matricule1), ...;
```

Où les `$matriculei` sont différents du matricule de l'auteur principal et sont ceux d'auteurs existants dans la base de données. C.-à-d. que le résultat de

```
SELECT matricule FROM auteurs WHERE matricule = $matriculei;
```

n'est pas vide pour tout `i`.

Le succès de chacune de ces requêtes est vérifié de façon à annuler l'opération en cas d'erreur.

```
ROLLBACK;
```

et la finaliser s'il n'y en a aucune.

```
COMMIT;
```

Dans notre implémentation, ces deux commandes sont remplacées respectivement par les méthodes `rollback()` et `commit()` fournies par `PDO`.

Finalement, après un succès ou un échec, nous déverrouillons les tables :

```
UNLOCK TABLES;
```

2.d Participants actifs

Pour exécuter la requête, le script `resources/sql/questions/q2d.sql` est appelé.

```sql
SELECT matricule, nom, prenom
FROM
(SELECT matricule, nom, prenom FROM auteurs) AS T1
NATURAL JOIN
(
    SELECT *
    FROM
    (
        SELECT DISTINCT matricule, matricule_auteur
        FROM
        (SELECT nom_conference, annee_conference, matricule FROM
            participations) AS T2
        LEFT JOIN
        (
            SELECT nom_conference, annee_conference, matricule_auteur
            FROM
            (SELECT url, nom_conference, annee_conference FROM
                articles_conferences) AS T3
            NATURAL JOIN
            (SELECT url, matricule_auteur FROM articles) AS T4
        ) AS T5
        ON T2.matricule = T5.matricule_auteur AND T2.nom_conference =
            T5.nom_conference AND T2.annee_conference = T5.annee_conference
    ) AS T6
    GROUP BY matricule
    HAVING COUNT(*) = 1 AND matricule_auteur IS NOT NULL
) AS T7;
```

Listing 5 – `q2d.sql`

En premier lieu, nous associons (`LEFT JOIN`) à chaque ligne de la table `participations` (`T2`) les articles de l'auteur participant dans la conférence. Il est possible qu'une ligne de `T2` n'ait aucun ou plusieurs articles associés.

Dans cette table, nous intéresse les matricules des participants qui ne sont *pas* associés à des `NULL`. Nous réduisons donc la table aux deux colonnes de matricule et ne sélectionnons que des lignes distinctes (`DISTINCT`).

Dans cette nouvelle table `T6`, tous les `matricule` figurant dans plusieurs lignes sont obligatoirement associés à une valeur `NULL`. La table `T7` groupe donc par `matricule` tout en supprimant les groupes de plus d'un élément et ceux dont la valeur associée est `NULL`.

Finalement, les nom et prénom des auteurs restants sont récupérés de la table `articles`.

2.e Sujets populaires

Pour exécuter la requête, le script `resources/sql/questions/q2e.sql` est appelé.

```sql
SELECT sujet, COUNT(*) AS popularity
FROM sujets_articles
NATURAL JOIN
(SELECT url, nom_conference, annee_conference FROM articles_conferences)
   AS T1
NATURAL JOIN
(
    SELECT nom_conference, annee_conference, COUNT(*) AS popularity
    FROM participations
    WHERE annee_conference >= 2012
    GROUP BY nom_conference, annee_conference
    ORDER BY popularity DESC
    LIMIT 5
) AS T2
GROUP BY sujet
ORDER BY popularity DESC;
```

Listing 6 – `q2e.sql`

Premièrement (`T2`), nous recherchons les cinq conférences les plus populaires depuis 2012, c.-à-d. celles ayant eu le plus de participations. Il est à noter que si plusieurs conférences ont le même nombre de participants, le tri (`ORDER`) pourrait différer à chaque appel ainsi que les cinq premières conférences [4].

Ensuite, nous récupérons les sujets de chaque article de ces conférences et les trions par nombre d'occurrences.

4. C'est le cas de notre base de données.