Université de Liège

Projet (partie 2)

Bases de données

Sabrina Bonghi (20161420)
Maxime Meurisse (20161278)
Valentin Vermeylen (20162864)

3e année de Bachelier Ingénieur civil

Année académique 2018-2019

1 Architecture du site web

La version définitive du site web est disponible via l'URL :

`student.montefiore.ulg.ac.be/~s161278/`

Les identifiants de connexion sont identiques à ceux de la base de données, à savoir :

- Nom d'utilisateur : **group1**
- Mot de passe : **co/A/vp7qq**

L'arborescence des fichiers composant le site web est présentée à la figure 1.

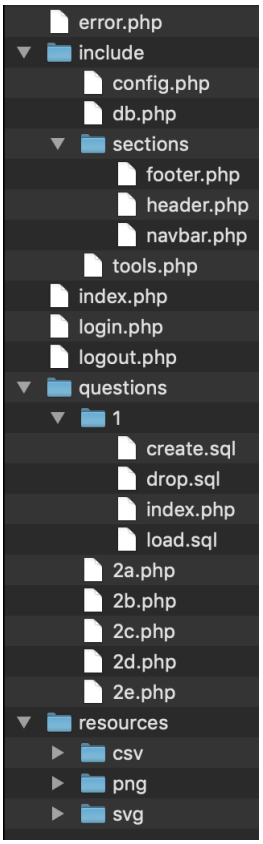

Figure 1 – Arborescence des fichiers composant le site web

Le site web se compose donc de plusieurs fichiers et dossiers.

Le dossier `resources` contient les ressources du site web, à savoir les icônes et logos utilisés (en format `png` et `svg`, situés dans les dossiers du même nom) ainsi que tous les fichiers `csv` contenant les données de la base de données.

Le dossier `questions` contient les fichiers `PHP` contenant les requêtes `SQL` (ainsi que le traitement des données en `PHP` et la structure de la page web en `HTML`) répondant aux différentes questions posées. Le fichier porte le nom de la question à laquelle il répond. Pour la question 1, il s'agit d'un dossier contenant trois fichiers `SQL` et un fichier `PHP` répondant à la question.

Le dossier `include` contient des fichiers contenant des bouts de code (variables, méthodes et mise en forme `HTML`) qui sont utilisés à plusieurs endroits du site web. En particulier, les fichiers se trouvant dans le dossier `sections` contiennent du code `HTML` de mise en forme qui sert de base à toutes les pages visitables du site web. Le fichier `tools.php` contient des méthodes `PHP` utiles, notamment celle vérifiant si un utilisateur est connecté ou non au site web. Le fichier `db.php` contient le script `PHP` permettant de se connecter à la base de données.

Les fichiers `login.php` et `logout.php` permettent la connexion et déconnexion du site web via des scripts `PHP` et formulaire `HTML`.

Le fichier `error.php` permet d'afficher les erreurs (notamment les erreurs 400, 401, 403, 404 et 500) de manière personnalisée.

Finalement, le fichier `.htaccess` permet de modifier certaines propriétés du site web, en particulier le ré-écriture d'URL (afin d'accéder à un fichier sans devoir renseigner son extension) et les redirections vers la page d'erreur personnalisée.

2 Initialiser la base de données

Les requêtes `SQL` permettant d'initialiser la base de données se trouvent dans les fichiers `create.sql` et `load.sql` (dans le dossier `1/`, lui-même dans le dossier `questions/`). Ces requêtes sont appelées dans le fichier `index.php` du même dossier.

Pour initialiser la base de données, il suffit de suivre les étapes suivantes :

Méthode: Initialisation de la base de données

1. Se rendre sur l'URL :

   `student.montefiore.ulg.ac.be/ s161278/questions/1/`

2. Cliquer sur le bouton vert « Initialiser la base de données ».

Contrairement aux autres requêtes `SQL`, celles-ci peuvent être exécutées sans devoir être connecté au site web.

Il existe aussi une requête `SQL` permettant de supprimer toutes les tables (fichier `drop.sql`). Elle peut être exécutée via la même page que la requête d'initialisation et permet, en combinaison avec l'autre requête, de réinitialiser la base de données, c.-à-d. recréer les tables avec les valeurs d'origine fournies dans les fichiers `csv`.

3 Les requêtes SQL

Comme présenté à la section 1, toutes les requêtes `SQL` (exceptées celles de l'initialisation de la base de données) répondant aux différentes questions se trouvent dans les fichiers `PHP` du dossier `questions`.

Ces différentes requêtes, isolées du reste et remises au clair, sont reprises ci-après.

3.1 Question 1

```sql
CREATE TABLE IF NOT EXISTS institutions (
    nom VARCHAR(128) NOT NULL,
    rue VARCHAR(128) NOT NULL,
    numero SMALLINT NOT NULL,
    ville VARCHAR(128) NOT NULL,
    pays VARCHAR(128) NOT NULL,
    PRIMARY KEY (nom)
) ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS auteurs (
    matricule INT NOT NULL,
    nom VARCHAR(64) NOT NULL,
    prenom VARCHAR(64) NOT NULL,
    debut_doctorat SMALLINT NOT NULL,
    nom_institution VARCHAR(128) NOT NULL,
    PRIMARY KEY (matricule),
    FOREIGN KEY (nom_institution) REFERENCES institutions(nom)
) ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS articles (
    url VARCHAR(255) NOT NULL,
    doi BIGINT NOT NULL,
    titre VARCHAR(255) NOT NULL,
    date_publication DATE NOT NULL,
    matricule_premier_auteur INT NOT NULL,
    PRIMARY KEY (url),
    FOREIGN KEY (matricule_premier_auteur) REFERENCES
        auteurs(matricule)
) ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS sujets_articles (
    url VARCHAR(255) NOT NULL,
    sujet VARCHAR(128) NOT NULL,
    PRIMARY KEY (url, sujet),
    FOREIGN KEY (url) REFERENCES articles(url)
) ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS seconds_auteurs (
    url VARCHAR(255) NOT NULL,
    matricule INT NOT NULL,
    PRIMARY KEY (url, matricule),
    FOREIGN KEY (url) REFERENCES articles(url),
    FOREIGN KEY (matricule) REFERENCES auteurs(matricule)
) ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS revues (
    nom VARCHAR(128) NOT NULL,
    impact SMALLINT NOT NULL,
    PRIMARY KEY (nom)
) ENGINE = InnoDB;
```

```
50
51  CREATE TABLE IF NOT EXISTS articles_journaux (
52      url VARCHAR(255) NOT NULL,
53      pg_debut SMALLINT NOT NULL,
54      pg_fin SMALLINT NOT NULL,
55      nom_revue VARCHAR(128) NOT NULL,
56      n_journal INT NOT NULL,
57      PRIMARY KEY (url),
58      FOREIGN KEY (url) REFERENCES articles(url),
59      FOREIGN KEY (nom_revue) REFERENCES revues(nom)
60  ) ENGINE = InnoDB;
61
62  CREATE TABLE IF NOT EXISTS conferences (
63      nom VARCHAR(128) NOT NULL,
64      annee SMALLINT NOT NULL,
65      rue VARCHAR(128) NOT NULL,
66      numero SMALLINT NOT NULL,
67      ville VARCHAR(128) NOT NULL,
68      pays VARCHAR(128) NOT NULL,
69      PRIMARY KEY (nom, annee)
70  ) ENGINE = InnoDB;
71
72  CREATE TABLE IF NOT EXISTS articles_conferences (
73      url VARCHAR(255) NOT NULL,
74      presentation VARCHAR(128) NOT NULL,
75      nom_conference VARCHAR(128) NOT NULL,
76      annee_conference SMALLINT NOT NULL,
77      PRIMARY KEY (url),
78      FOREIGN KEY (url) REFERENCES articles(url),
79      FOREIGN KEY (nom_conference, annee_conference) REFERENCES
            conferences(nom, annee)
80  ) ENGINE = InnoDB;
81
82  CREATE TABLE IF NOT EXISTS participations_conferences (
83      matricule INT NOT NULL,
84      nom_conference VARCHAR(128) NOT NULL,
85      annee_conference SMALLINT NOT NULL,
86      tarif VARCHAR(128) NOT NULL,
87      PRIMARY KEY (matricule, nom_conference, annee_conference),
88      FOREIGN KEY (matricule) REFERENCES auteurs(matricule),
89      FOREIGN KEY (nom_conference, annee_conference) REFERENCES
            conferences(nom, annee)
90  ) ENGINE = InnoDB;
91
92  LOAD DATA LOCAL INFILE
        '/home/s161278/WWW/resources/csv/institutions.csv' INTO TABLE
        institutions
93  CHARACTER SET 'UTF8'
94  FIELDS TERMINATED BY ';'
95  LINES TERMINATED BY '\n'
96  IGNORE 1 LINES;
97
98  LOAD DATA LOCAL INFILE '/home/s161278/WWW/resources/csv/auteurs.csv'
        INTO TABLE auteurs
99  CHARACTER SET 'UTF8'
100 FIELDS TERMINATED BY ';'
```

```
101  LINES TERMINATED BY '\n'
102  IGNORE 1 LINES;
103
104  LOAD DATA LOCAL INFILE '/home/s161278/WWW/resources/csv/articles.csv'
       INTO TABLE articles
105  CHARACTER SET 'UTF8'
106  FIELDS TERMINATED BY ';'
107  LINES TERMINATED BY '\n'
108  IGNORE 1 LINES
109  (`url`, `doi`, `titre`, @DATE_STR, `matricule_premier_auteur`)
110  SET `date_publication` = STR_TO_DATE(@DATE_STR, '%d/%m/%Y');
111
112  LOAD DATA LOCAL INFILE
       '/home/s161278/WWW/resources/csv/sujets_articles.csv' INTO TABLE
       sujets_articles
113  CHARACTER SET 'UTF8'
114  FIELDS TERMINATED BY ';'
115  LINES TERMINATED BY '\n'
116  IGNORE 1 LINES;
117
118  LOAD DATA LOCAL INFILE
       '/home/s161278/WWW/resources/csv/seconds_auteurs.csv' INTO TABLE
       seconds_auteurs
119  CHARACTER SET 'UTF8'
120  FIELDS TERMINATED BY ';'
121  LINES TERMINATED BY '\n'
122  IGNORE 1 LINES;
123
124  LOAD DATA LOCAL INFILE '/home/s161278/WWW/resources/csv/revues.csv'
       INTO TABLE revues
125  CHARACTER SET 'UTF8'
126  FIELDS TERMINATED BY ';'
127  LINES TERMINATED BY '\n'
128  IGNORE 1 LINES;
129
130  LOAD DATA LOCAL INFILE
       '/home/s161278/WWW/resources/csv/articles_journaux.csv' INTO
       TABLE articles_journaux
131  CHARACTER SET 'UTF8'
132  FIELDS TERMINATED BY ';'
133  LINES TERMINATED BY '\n'
134  IGNORE 1 LINES;
135
136  LOAD DATA LOCAL INFILE
       '/home/s161278/WWW/resources/csv/conferences.csv' INTO TABLE
       conferences
137  CHARACTER SET 'UTF8'
138  FIELDS TERMINATED BY ';'
139  LINES TERMINATED BY '\n'
140  IGNORE 1 LINES;
141
142  LOAD DATA LOCAL INFILE
       '/home/s161278/WWW/resources/csv/articles_conferences.csv' INTO
       TABLE articles_conferences
143  CHARACTER SET 'UTF8'
144  FIELDS TERMINATED BY ';'
```

```
145  LINES TERMINATED BY '\n'
146  IGNORE 1 LINES;
147
148  LOAD DATA LOCAL INFILE
         '/home/s161278/WWW/resources/csv/participations_conferences.csv'
         INTO TABLE participations_conferences
149  CHARACTER SET 'UTF8'
150  FIELDS TERMINATED BY ';'
151  LINES TERMINATED BY '\n'
152  IGNORE 1 LINES;
```

Listing 1 – Requête SQL de la question 1

3.2 Question 2a

Cette requête a été construite en PHP. En effet, selon les contraintes de recherche renseignées ou non par l'utilisateur, la requête n'est pas la même.

Dans ce script, la variable `tab_constraints` est un tableau contenant toutes les contraintes renseignées par l'utilisateur. Chaque contrainte est sous forme d'un tableau où le premier élément est l'attribut et le second sa valeur.

```
 1  <?php
 2
 3  $sql = "SELECT * FROM $table";
 4
 5  if(!empty($tab_constraints)) {
 6      $sql .= " WHERE ";
 7
 8      /// on tient compte des contraintes, s'il y en a
 9      foreach($tab_constraints as $constraint) {
10          if(is_numeric($constraint[1])) {
11              /// contrainte d'egalite
12              $sql .= "$constraint[0] = $constraint[1] AND ";
13          } else {
14              /// contraintes de contenance
15              $sql .= "$constraint[0] LIKE '%$constraint[1]%' AND ";
16          }
17      }
18
19      $sql .= "1";
20  }
21
22  ?>
```

Listing 2 – Requête SQL, construite en PHP, de la question 2a

3.3 Question 2b

Dans cette requête, le `x` correspond à la matricule du premier auteur, renseignée par l'utilisateur.

```sql
SELECT url, titre, type, date_publication
FROM
    articles
    NATURAL JOIN
    (
        SELECT url, 'journal' AS type
        FROM articles_journaux
        UNION
        SELECT url, 'conference' AS type
        FROM articles_conferences
    ) AS articles_type
WHERE matricule_premier_auteur = x
ORDER BY date_publication DESC;
```

Listing 3 – Requête SQL de la question 2b

3.4 Question 2c

Cette requête, assez longue, a été construite en PHP. En effet, les requêtes d'ajout dépendent des données renseignées ou non et du type de l'article.

Les requêtes permettant de vérifier les entrées de l'utilisateur n'ont pas été reprises ci-après.

Chaque variable portant le nom d'un attribut contient la valeur de cet attribut (renseignée par l'utilisateur).

L'utilisation des méthodes de PDO concernant les transactions permettent d'assurer la cohérence de la base de données, même en cas d'erreur lors de l'insertion. En effet, il serait inacceptable qu'une partie des données soit insérée dans une table mais que le reste des données ne soit pas insérée à cause d'une erreur. Avec la gestion des transactions, cette situation ne peut pas arriver : soit toutes les données sont insérées, soit rien n'est inséré.

```php
<?php

/* ---------------------------------------- */
/* Verification des contraintes d'integrite */
/* ---------------------------------------- */
$check_annee = date('Y', strtotime($date_publication));

// Les articles d'une conference doivent avoir ete publies l'annee de la conference
if($type == 'conference') {
    if($check_annee != $annee_conference) {
        $error[$i++] = 'L\'article n\'a pas ete publie l\'annee de la conference.';
    }
}

```

```
15  // Les articles publies dans un meme journal doivent avoir ete
        publies la meme annee
16  if($type == 'journal') {
17      $sql = "SELECT date_publication
18              FROM
19                  (
20                      SELECT url, date_publication
21                      FROM articles
22                  ) AS T1
23                  NATURAL JOIN
24                  (
25                      SELECT url
26                      FROM articles_journaux
27                      WHERE n_journal = 51
28                      LIMIT 1
29                  ) AS T2";
30      $request = $connect->query($sql);
31      $response = $request->fetchAll(PDO::FETCH_ASSOC);
32
33      $date_journal = date('Y',
            strtotime($response[0]['date_publication']));
34
35      if($check_annee != $date_journal) {
36          $error[$i++] = 'Les articles publies dans un meme journal
                doivent avoir ete publies la meme annee.';
37      }
38  }
39
40  /// A ce stade, toutes les entrees sont valides et les contraintes
        d'integritees respectees.
41
42  /* ---------------------------------------------- */
43  /* Insertion des donnees dans la base de donnees */
44  /* ---------------------------------------------- */
45  if(empty($error)) {
46      try {
47          $connect->beginTransaction();
48
49          $lock = "LOCK TABLES articles WRITE, sujets_articles WRITE";
50          $sql = "";
51
52          // articles
53          $sql .= "INSERT INTO articles (url, doi, titre,
                date_publication, matricule_premier_auteur) VALUES
                ('$url', $doi, '$titre', '$date_publication',
                $premier_auteur);";
54
55          // sujets_articles
56          foreach($sujets as $sujet) {
57              $sql .= "INSERT INTO sujets_articles (url, sujet) VALUES
                    ('$url', '$sujet');";
58          }
59
60          // seconds_auteurs
61          if(!empty($seconds_auteurs)) {
62              $lock .= ", seconds_auteurs WRITE";
```

```
63
64              foreach($seconds_auteurs as $second_auteur) {
65                  $sql .= "INSERT INTO seconds_auteurs (url, matricule)
                        VALUES ('$url', $second_auteur);";
66              }
67          }
68
69          // articles_journaux
70          if($type == 'journal') {
71              $lock .= ", articles_journaux WRITE";
72
73              $sql .= "INSERT INTO articles_journaux (url, pg_debut,
                    pg_fin, nom_revue, n_journal) VALUES ('$url',
                    $page_debut, $page_fin, '$revue', $n_journal);";
74          }
75
76          // articles_conferences
77          if($type == 'conference') {
78              $lock .= ", articles_conferences WRITE";
79
80              $sql .= "INSERT INTO articles_conferences (url,
                    presentation, nom_conference, annee_conference)
                    VALUES ('$url', '$presentation', '$nom_conference',
                    $annee_conference);";
81          }
82
83          $lock .= ";";
84
85          $connect->query($lock);
86          $connect->query($sql);
87          $connect->commit();
88
89          $success = 'L\'article a ete ajoute a la base de donnees !';
90      } catch(Exception $e) {
91          $connect->rollback();
92
93          $error = 'Une erreur est survenue lors de l\'insertion des
                donnees dans la base de donnees.';
94      }
95
96      $connect->query('UNLOCK TABLES');
97  }
98
99  ?>
```

Listing 4 – Requête SQL, construite en PHP, de la question 2c

3.5 Question 2d

Cette requête SQL se compose de plusieurs types de JOIN imbriqués.

Le premier NATURAL JOIN entre T1 et T2 (et renommé T3) permet de récupérer les participations aux conférences de chaque auteur.

Le deuxième `NATURAL JOIN` entre `T1` et `T2` (et renommé `T4`) permet de récupérer les auteurs de chaque article de conférence.

Le `LEFT JOIN` entre `T3` et `T4` permet d'associer à chaque auteur ayant participé à une conférence l'(les) article(s) qu'il y a écrit. Le `LEFT JOIN` permet de conserver la participation d'un auteur à une conférence même s'il n'y a pas écrit d'article.

Il suffit ensuite de sélectionner les auteurs qui, pour chacune de leur participation à une conférence, ont au moins un article associé.

```sql
SELECT matricule, nom, prenom
FROM
    (
        SELECT T3.matricule, T3.nom, T3.prenom, T3.nom_conference,
            T3.annee_conference, T4.url
        FROM
            (
                SELECT matricule, nom, prenom, nom_conference,
                    annee_conference
                FROM
                    (
                        SELECT matricule, nom_conference,
                            annee_conference
                        FROM participations_conferences
                    ) AS T1
                    NATURAL JOIN
                    (
                        SELECT matricule, nom, prenom
                        FROM auteurs
                    ) AS T2
                ORDER BY matricule ASC
            ) AS T3
            LEFT JOIN
            (
                SELECT url, matricule, nom_conference,
                    annee_conference
                FROM
                    (
                        SELECT url, matricule_premier_auteur AS
                            matricule
                        FROM articles
                    ) AS T1
                    NATURAL JOIN
                    (
                        SELECT url, nom_conference, annee_conference
                    FROM articles_conferences
                    ) AS T2
                ORDER BY matricule ASC
            ) AS T4
            ON T3.matricule = T4.matricule AND T3.nom_conference =
                T4.nom_conference AND T3.annee_conference =
                T4.annee_conference
    ) AS T5
GROUP BY matricule
HAVING COUNT(nom_conference) = COUNT(url)
ORDER BY matricule AS;
```

Listing 5 – Requête SQL de la question 2d

3.6 Question 2e

Cette requête SQL se compose de plusieurs NATURAL JOIN imbriqués.

Le premier (entre T1 et T2) a pour but de récupérer les URL des articles ayant été écrits aux 5 conférences les plus populaires depuis 2012.

Le deuxième (entre T3 et T4) a pour but de récupérer les sujets de tous ces articles.

Il suffit ensuite de sélectionner ces sujets, compter le nombre de fois que chaque sujet apparait et les classer par ordre décroissant d'apparition.

```sql
SELECT sujet, COUNT(sujet) AS used
FROM
    (
        SELECT sujet
        FROM
            (
                SELECT *
                FROM sujets_articles
            ) AS T3
            NATURAL JOIN
            (
                SELECT url
                FROM
                    (
                        SELECT nom_conference, annee_conference,
                            COUNT(matricule) AS popularity
                        FROM participations_conferences
                        WHERE annee_conference >= 2012
                        GROUP BY nom_conference, annee_conference
                        ORDER BY popularity DESC
                        LIMIT 5
                    ) AS T1
                    NATURAL JOIN
                    (
                        SELECT url, nom_conference, annee_conference
                        FROM articles_conferences
                    ) AS T2
            ) AS T4
    ) AS T5
GROUP BY sujet
ORDER BY used DESC;
```

Listing 6 – Requête SQL de la question 2e