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.

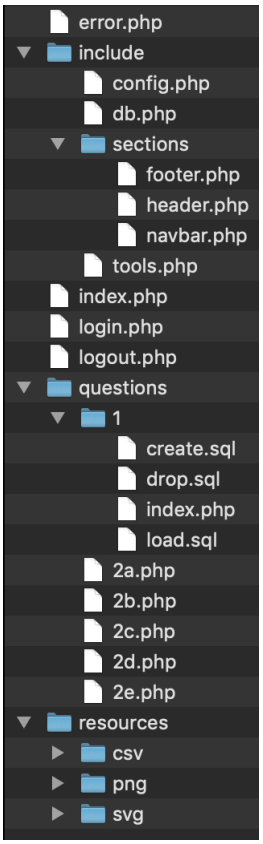

Figure 1 – Arborescence des fichiers composant le site web

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 la 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

On remarque que le code SQL permettant de charger le contenu de la table articles convertit l'attribut date_publication en format DATE. Pour ce faire, on utilise la méthode STR_TO_DATE de SQL (car la date dans le fichier csv n'était pas dans un format de date standard).

L'avantage de cette conversion est qu'il sera beaucoup plus simple de trier les dates (par ordre croissant ou décroissant) par la suite (en effet, un simple ORDER BY date_publication ASC/DESC suffira).

```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,
```

```
40      PRIMARY KEY (url, matricule),
41      FOREIGN KEY (url) REFERENCES articles(url),
42      FOREIGN KEY (matricule) REFERENCES auteurs(matricule)
43  ) ENGINE = InnoDB;
44
45  CREATE TABLE IF NOT EXISTS revues (
46      nom VARCHAR(128) NOT NULL,
47      impact SMALLINT NOT NULL,
48      PRIMARY KEY (nom)
49  ) 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
CHARACTER SET 'UTF8'
FIELDS TERMINATED BY ';'
LINES TERMINATED BY '\n'
IGNORE 1 LINES;

LOAD DATA LOCAL INFILE '/home/s161278/WWW/resources/csv/auteurs.csv'
    INTO TABLE auteurs
CHARACTER SET 'UTF8'
FIELDS TERMINATED BY ';'
LINES TERMINATED BY '\n'
IGNORE 1 LINES;

LOAD DATA LOCAL INFILE '/home/s161278/WWW/resources/csv/articles.csv'
    INTO TABLE articles
CHARACTER SET 'UTF8'
FIELDS TERMINATED BY ';'
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(`url`, `doi`, `titre`, @DATE_STR, `matricule_premier_auteur`)
SET `date_publication` = STR_TO_DATE(@DATE_STR, '%d/%m/%Y');

LOAD DATA LOCAL INFILE
    '/home/s161278/WWW/resources/csv/sujets_articles.csv' INTO TABLE
    sujets_articles
CHARACTER SET 'UTF8'
FIELDS TERMINATED BY ';'
LINES TERMINATED BY '\n'
IGNORE 1 LINES;

LOAD DATA LOCAL INFILE
    '/home/s161278/WWW/resources/csv/seconds_auteurs.csv' INTO TABLE
    seconds_auteurs
CHARACTER SET 'UTF8'
FIELDS TERMINATED BY ';'
LINES TERMINATED BY '\n'
IGNORE 1 LINES;

LOAD DATA LOCAL INFILE '/home/s161278/WWW/resources/csv/revues.csv'
    INTO TABLE revues
CHARACTER SET 'UTF8'
FIELDS TERMINATED BY ';'
LINES TERMINATED BY '\n'
IGNORE 1 LINES;

LOAD DATA LOCAL INFILE
    '/home/s161278/WWW/resources/csv/articles_journaux.csv' INTO
    TABLE articles_journaux
CHARACTER SET 'UTF8'
FIELDS TERMINATED BY ';'
LINES TERMINATED BY '\n'
IGNORE 1 LINES;

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.

L'utilisation de `COLLATE UTF8_GENERAL_CI` juste avant le mot clé `LIKE` dans les contraintes de contenance permet de ne pas tenir compte de la casse lors de la recherche.

```
 1  <?php
 2
 3  /// On crée la requête SQL
 4  $sql = "SELECT * FROM $table";
 5
 6  /// On y ajoute les contraintes s'il y en a
 7  if(!empty($tab_constraints)) {
 8      $sql .= " WHERE ";
 9
10      foreach($tab_constraints as $constraint) {
11          /// contrainte d'égalité
12          if(is_numeric($constraint[1]) || $constraint[0] ==
                $bdd_infos['articles'][3]) {
13              $sql .= "$constraint[0] = '$constraint[1]' AND ";
14          }
15
16          /// contrainte de contenance
17          else {
```

```
18              $sql .= "$constraint[0] COLLATE UTF8_GENERAL_CI LIKE
                    '%$constraint[1]%' AND ";
19          }
20      }
21
22      $sql .= "1";
23  }
24
25  ?>
```

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

3.3 Question 2b

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

```
1  SELECT url, titre, type, date_publication
2  FROM
3      articles
4      NATURAL JOIN
5      (
6          SELECT url, 'journal' AS type
7          FROM articles_journaux
8          UNION
9          SELECT url, 'conference' AS type
10         FROM articles_conferences
11     ) AS articles_type
12 WHERE matricule_premier_auteur = x
13 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.

Pour ce faire, lorsque toutes les données de l'utilisateur ont été traitées, on démarre une transaction (grâce à la méthode correspondante de PDO) et on verrouille en écriture toutes les tables dans lesquelles on va rajouter des données. On tente d'ajouter les données. Si aucune erreur n'est survenue, on effectue un *commit*, sinon on effectue un *rollback* (grâce aux méthodes correspondantes de PDO). Dans tous les cas, on déverouille les tables.

```php
<?php
/* ---------------------------------------- */
/* Vérification des contraintes d'intégrité */
/* ---------------------------------------- */

/* [...] */

/// Matricule(s) du(des) second(s) auteur(s)
if(!empty($seconds_auteurs)) {
    $seconds_auteurs = explode(',', $seconds_auteurs);

    foreach($seconds_auteurs as &$e) {
        $e = trim($e);

        if($e == $premier_auteur) {
            $error[$i++] = 'Le premier auteur ne peut pas être également second auteur.';
        }

        $sql = "SELECT matricule FROM auteurs WHERE matricule = $e";
        $request = $connect->query($sql);
        $check = $request->rowCount();

        if($check != 1) {
            $error[$i++] = 'Un matricule de second auteur n\'est pas valide.';
        }
    }
}

/* [...] */

$check_annee = date('Y', strtotime($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 été publié l\'année de la conférence.';
    }
}

// Les articles publies dans un meme journal doivent avoir ete publies la meme annee
if($type == 'journal') {
```

```
42      $sql = "SELECT date_publication
43              FROM
44                  (
45                      SELECT url, date_publication
46                      FROM articles
47                  ) AS T1
48                  NATURAL JOIN
49                  (
50                      SELECT url
51                      FROM articles_journaux
52                      WHERE n_journal = 51
53                      LIMIT 1
54                  ) AS T2";
55      $request = $connect->query($sql);
56      $response = $request->fetchAll(PDO::FETCH_ASSOC);
57
58      $date_journal = date('Y',
            strtotime($response[0]['date_publication']));
59
60      if($check_annee != $date_journal) {
61          $error[$i++] = 'Les articles publiés dans un même journal
                doivent avoir été publiés la même année.';
62      }
63  }
64
65  /// À ce stade, toutes les entrées sont valides et les contraintes
        d'intégritées respectées.
66
67  /* ---------------------------------------------- */
68  /* Insertion des données dans la base de données */
69  /* ---------------------------------------------- */
70  if(empty($error)) {
71      try {
72          $connect->beginTransaction();
73
74          $lock = "LOCK TABLES articles WRITE, sujets_articles WRITE";
75          $sql = "";
76
77          // articles
78          $sql .= "INSERT INTO articles (url, doi, titre,
                date_publication, matricule_premier_auteur) VALUES
                ('$url', $doi, '$titre', '$date_publication',
                $premier_auteur);";
79
80          // sujets_articles
81          foreach($sujets as $sujet) {
82              $sql .= "INSERT INTO sujets_articles (url, sujet) VALUES
                    ('$url', '$sujet');";
83          }
84
85          // seconds_auteurs
86          if(!empty($seconds_auteurs)) {
87              $lock .= ", seconds_auteurs WRITE";
88
89              foreach($seconds_auteurs as $second_auteur) {
90                  $sql .= "INSERT INTO seconds_auteurs (url, matricule)
```

```
                    VALUES ('$url', $second_auteur);";
91              }
92          }
93
94          // articles_journaux
95          if($type == 'journal') {
96              $lock .= ", articles_journaux WRITE";
97
98              $sql .= "INSERT INTO articles_journaux (url, pg_debut,
                    pg_fin, nom_revue, n_journal) VALUES ('$url',
                    $page_debut, $page_fin, '$revue', $n_journal);";
99          }
100
101         // articles_conferences
102         if($type == 'conference') {
103             $lock .= ", articles_conferences WRITE";
104
105             $sql .= "INSERT INTO articles_conferences (url,
                    presentation, nom_conference, annee_conference)
                    VALUES ('$url', '$presentation', '$nom_conference',
                    $annee_conference);";
106         }
107
108         $lock .= ";";
109
110         $connect->query($lock);
111         $connect->query($sql);
112         $connect->commit();
113
114         $success = 'L\'article a été ajouté à la base de données !';
115     } catch(Exception $e) {
116         $connect->rollback();
117
118         $error = 'Une erreur est survenue lors de l\'insertion des
                données dans la base de données.';
119     }
120
121     $connect->query('UNLOCK TABLES');
122 ?>
```

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 confé-

rénce 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.

Remarque Les résultats de cette requête, affichée sur la page PHP correspondante, ne sont pas toujours identiques. Cela vient du fait que les 5e et 6e conférences les plus populaires ont la même cote de popularité. Dès lors, on suppose que SQL effectue un choix arbitraire entre ces deux conférences pour sélectionner les 5 conférences les plus populaires. Ce choix arbitraire peut donc mener à des résultats différents pour plusieurs exécutions du même script.

```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