



UNIVERSITÉ DE LIÈGE

Projet (partie 2)

Bases de données

Sabrina BONGHI (20161420)
Maxime MEURISSE (20161278)
Valentin VERMEYLEN (20162864)

3^e 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.

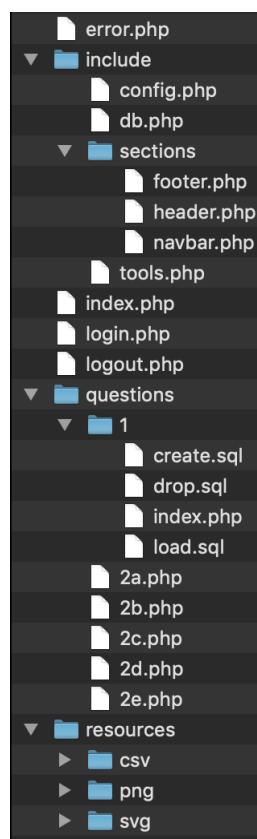


Figure 1 – Arborescence des fichiers composant le site web

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).

```
1 CREATE TABLE IF NOT EXISTS institutions (
2     nom VARCHAR(128) NOT NULL,
3     rue VARCHAR(128) NOT NULL,
4     numero SMALLINT NOT NULL,
5     ville VARCHAR(128) NOT NULL,
6     pays VARCHAR(128) NOT NULL,
7     PRIMARY KEY (nom)
8 ) ENGINE = InnoDB;
9
10 CREATE TABLE IF NOT EXISTS auteurs (
11     matricule INT NOT NULL,
12     nom VARCHAR(64) NOT NULL,
13     prenom VARCHAR(64) NOT NULL,
14     debut_doctorat SMALLINT NOT NULL,
15     nom_institution VARCHAR(128) NOT NULL,
16     PRIMARY KEY (matricule),
17     FOREIGN KEY (nom_institution) REFERENCES institutions(nom)
18 ) ENGINE = InnoDB;
19
20 CREATE TABLE IF NOT EXISTS articles (
21     url VARCHAR(255) NOT NULL,
22     doi BIGINT NOT NULL,
23     titre VARCHAR(255) NOT NULL,
24     date_publication DATE NOT NULL,
25     matricule_premier_auteur INT NOT NULL,
26     PRIMARY KEY (url),
27     FOREIGN KEY (matricule_premier_auteur) REFERENCES
28         auteurs(matricule)
29 ) ENGINE = InnoDB;
30
31 CREATE TABLE IF NOT EXISTS sujets_articles (
32     url VARCHAR(255) NOT NULL,
33     sujet VARCHAR(128) NOT NULL,
34     PRIMARY KEY (url, sujet),
35     FOREIGN KEY (url) REFERENCES articles(url)
36 ) ENGINE = InnoDB;
37
38 CREATE TABLE IF NOT EXISTS seconds_auteurs (
39     url VARCHAR(255) NOT NULL,
40     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
193 CHARACTER SET 'UTF8'
194 FIELDS TERMINATED BY ';'
195 LINES TERMINATED BY '\n'
196 IGNORE 1 LINES;
197
198 LOAD DATA LOCAL INFILE '/home/s161278/WWW/resources/csv/auteurs.csv'
        INTO TABLE auteurs
199 CHARACTER SET 'UTF8'
200 FIELDS TERMINATED BY ';'
201 LINES TERMINATED BY '\n'
202 IGNORE 1 LINES;
203
204 LOAD DATA LOCAL INFILE '/home/s161278/WWW/resources/csv/articles.csv'
        INTO TABLE articles
205 CHARACTER SET 'UTF8'
206 FIELDS TERMINATED BY ';'
207 LINES TERMINATED BY '\n'
208 IGNORE 1 LINES
209 (`url`, `doi`, `titre`, @DATE_STR, `matricule_premier_auteur`)
210 SET `date_publication` = STR_TO_DATE(@DATE_STR, '%d/%m/%Y');
211
212 LOAD DATA LOCAL INFILE
        '/home/s161278/WWW/resources/csv/sujets_articles.csv' INTO TABLE
        sujets_articles
213 CHARACTER SET 'UTF8'
214 FIELDS TERMINATED BY ';'
215 LINES TERMINATED BY '\n'
216 IGNORE 1 LINES;
217
218 LOAD DATA LOCAL INFILE
        '/home/s161278/WWW/resources/csv/seconds_auteurs.csv' INTO TABLE
        seconds_auteurs
219 CHARACTER SET 'UTF8'
220 FIELDS TERMINATED BY ';'
221 LINES TERMINATED BY '\n'
222 IGNORE 1 LINES;
223
224 LOAD DATA LOCAL INFILE '/home/s161278/WWW/resources/csv/revues.csv'
        INTO TABLE revues
225 CHARACTER SET 'UTF8'
226 FIELDS TERMINATED BY ';'
227 LINES TERMINATED BY '\n'
228 IGNORE 1 LINES;
229
230 LOAD DATA LOCAL INFILE
        '/home/s161278/WWW/resources/csv/articles_journaux.csv' INTO
        TABLE articles_journaux
231 CHARACTER SET 'UTF8'
232 FIELDS TERMINATED BY ';'
233 LINES TERMINATED BY '\n'
234 IGNORE 1 LINES;
235
236 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
143     '/home/s161278/WWW/resources/csv/articles_conferences.csv' INTO
144     TABLE articles_conferences
145 CHARACTER SET 'UTF8'
146 FIELDS TERMINATED BY ';'
147 LINES TERMINATED BY '\n'
148 IGNORE 1 LINES;
149
150 LOAD DATA LOCAL INFILE
151     '/home/s161278/WWW/resources/csv/participations_conferences.csv'
152     INTO TABLE participations_conferences
153 CHARACTER SET 'UTF8'
154 FIELDS TERMINATED BY ';'
155 LINES TERMINATED BY '\n'
156 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] ==
13             $bdd_infos['articles'][3]) {
14             $sql .= "$constraint[0] = '$constraint[1]' AND ";
15         }
16
17         /// contrainte de contenance
18         else {

```

```

18         $sql .= "$constraint[0] COLLATE UTF8_GENERAL_CI LIKE
19             '$constraint[1]%' AND ";
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 déverrouille les tables.

```
1  <?php
2  /* ----- */
3  /* Vérification des contraintes d'intégrité */
4  /* ----- */
5
6  /* [...] */
7
8  /// Matricule(s) du(des) second(s) auteur(s)
9  if(!empty($seconds_auteurs)) {
10     $seconds_auteurs = explode(',', $seconds_auteurs);
11
12     foreach($seconds_auteurs as &$e) {
13         $e = trim($e);
14
15         if($e == $premier_auteur) {
16             $error[$i++] = 'Le premier auteur ne peut pas être
17                             également second auteur.';
18         }
19
20         $sql = "SELECT matricule FROM auteurs WHERE matricule = $e";
21         $request = $connect->query($sql);
22         $check = $request->rowCount();
23
24         if($check != 1) {
25             $error[$i++] = 'Un matricule de second auteur n\'est pas
26                             valide.';
27         }
28     }
29 }
30
31 /* [...] */
32
33 $check_annee = date('Y', strtotime($date_publication));
34
35 // Les articles d'une conference doivent avoir ete publies l'annee de
36 // la conference
37 if($type == 'conference') {
38     if($check_annee != $annee_conference) {
39         $error[$i++] = 'L'article n\'a pas été publié l'année de la
40                         conférence.';
41     }
42 }
43
44 // Les articles publies dans un meme journal doivent avoir ete
45 // publies la meme annee
46 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     $date_journal = date('Y',
59         strtotime($response[0]['date_publication']));
60
61     if($check_annee != $date_journal) {
62         $error[$i++] = 'Les articles publiés dans un même journal
63             doivent avoir été publiés la même année.';
64     }
65
66     /// À ce stade, toutes les entrées sont valides et les contraintes
67     d'intégrités respectées.
68
69     /* ----- */
70     /* Insertion des données dans la base de données */
71     /* ----- */
72     if(empty($error)) {
73         try {
74             $connect->beginTransaction();
75
76             $lock = "LOCK TABLES articles WRITE, sujets_articles WRITE";
77             $sql = "";
78
79             // articles
80             $sql .= "INSERT INTO articles (url, doi, titre,
81                 date_publication, matricule_premier_auteur) VALUES
82                 ('$url', $doi, '$titre', '$date_publication',
83                 $premier_auteur);";
84
85             // sujets_articles
86             foreach($sujets as $sujet) {
87                 $sql .= "INSERT INTO sujets_articles (url, sujet) VALUES
88                     ('$url', '$sujet');";
89             }
90
91             // seconds_auteurs
92             if(!empty($seconds_auteurs)) {
93                 $lock .= ", seconds_auteurs WRITE";
94
95                 foreach($seconds_auteurs as $second_auteur) {
96                     $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-

rence 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.

```
1  SELECT matricule, nom, prenom
2  FROM
3      (
4          SELECT T3.matricule, T3.nom, T3.prenom, T3.nom_conference,
5              T3.annee_conference, T4.url
6          FROM
7              (
8                  SELECT matricule, nom, prenom, nom_conference,
9                      annee_conference
10                 FROM
11                     (
12                         SELECT matricule, nom_conference,
13                             annee_conference
14                         FROM participations_conferences
15                     ) AS T1
16                 NATURAL JOIN
17                 (
18                     SELECT matricule, nom, prenom
19                     FROM auteurs
20                 ) AS T2
21             ORDER BY matricule ASC
22         ) AS T3
23     LEFT JOIN
24     (
25         SELECT url, matricule, nom_conference,
26             annee_conference
27         FROM
28             (
29                 SELECT url, matricule_premier_auteur AS
30                     matricule
31                 FROM articles
32             ) AS T1
33             NATURAL JOIN
34             (
35                 SELECT url, nom_conference, annee_conference
36                 FROM articles_conferences
37             ) AS T2
38             ORDER BY matricule ASC
39         ) AS T4
40     ON T3.matricule = T4.matricule AND T3.nom_conference =
41         T4.nom_conference AND T3.annee_conference =
42         T4.annee_conference
43     ) AS T5
44 GROUP BY matricule
45 HAVING COUNT(nom_conference) = COUNT(url)
46 ORDER BY matricule ASC;
```

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 apparaît 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 5^e et 6^e 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.

```
1 SELECT sujet, COUNT(sujet) AS used
2 FROM
3     (
4         SELECT sujet
5         FROM
6             (
7                 SELECT *
8                 FROM sujets_articles
9             ) AS T3
10        NATURAL JOIN
11        (
12            SELECT url
13            FROM
14                (
15                    SELECT nom_conference, annee_conference,
16                           COUNT(matricule) AS popularity
17                    FROM participations_conferences
18                    WHERE annee_conference >= 2012
19                    GROUP BY nom_conference, annee_conference
20                    ORDER BY popularity DESC
21                    LIMIT 5
22                ) AS T1
23            NATURAL JOIN
24            (
25                SELECT url, nom_conference, annee_conference
26                FROM articles_conferences
27            ) AS T2
28        ) AS T4
29    ) AS T5
30 GROUP BY sujet
31 ORDER BY used DESC;
```

Listing 6 – Requête SQL de la question 2e