

INFO-H-303 Bases de données

Projet : DBLP

Quentin Delhay

Année académique 2012–2013

1 Modélisation

author (Name)
publication (key, Title, Publisher, url, ee, Year, mdate, type)
pub_author (AuthorName, PubID)
publisher (Name)
publication_publisher (PublicaKey, PublishName)
pub_book (PubID, Volume, Series, isbn)
editor (Name)
book_editor (EditorName, BookID)
pub_incollection (PubID, Chapter, Pages)
incollection_in_book (IncolID, BookID)
pub_article (PubID, Volume, Number, Pages)
journal (Title)
article_in_journal (ArtID, Journal)
pub_master_thesis (PubID, School)
pub_phd_thesis (PubID, School, isbn)
admin (Name, password)

- Les héritages ont été gérés comme suit : toutes les entités ont été traduites en relation, et les relations découlant des entités filles ont une clé supplémentaire « PubID » pointant vers l'id de la publication à laquelle elles correspondent. Ce choix permet d'éviter une redondance certaine, offre un parcours plus simple de toutes les publications ainsi qu'un nombre de tables réduit puisqu'il n'a pas fallu distribuer l'association « author - publication » et « publisher - publication ». En revanche, il faudra parcourir deux tables pour obtenir toutes les informations concernant une publication.
- L'entité binaire « author - Writes - publication » a été remplacée par une relation « pub_author ». Sa clé composite est créée à partir de l'id de la publication et du nom de l'auteur. Il en va de même pour l'entité « editor - Edits - Book » qui est remplacée par la relation « book_editor », ainsi que « publisher - publishes - publication » par « publication_publisher ».
- L'éventuelle multiplicité des auteurs apparaîtra sous forme de redondance des entrées des relations sus-citées.
- L'association « incollection - isPartOf - book » est remplacée par la table « incollection_in_book » dont l'attribut « IncID » représente l'ID de l'incollection et « BookID » l'ID du book dont incollection fait partie.

- L'attribut *pub_author.AuthorName* correspond à l'attribut *author.Name*, et *pub_author.PubID* à *publication.PubKey*.
- L'attribut *publication_publisher.PublicaKey* correspond à l'attribut *publication.PubKey*, et *publication_publisher.PublishName* à *publisher.Name*.
- L'attribut *book_editor.BookID* correspond à l'attribut *publication.PubKey*, et *book_editor.EditorName* à *editor.Name*.
- L'attribut *incollection_in_book.IncolID* correspond à l'attribut *pub_incollection.PubID*, et *incollection_in_book.BookID* à *pub_book.PubID*.
- L'attribut *article_in_journal.ArtID* correspond à l'attribut *pub_article.PubID*, et *article_in_journal.Jou* à *journal.Title*.

**INFO-H-303 Bases de données
Quentin Delhaye 2012-2013
Diagramme entités associations**

Year ne peut être plus tard que mdate.
DBLP ne donne pas l'isbn de toutes les publications.
Une publication peut avoir plusieurs auteurs.
Les homonymes et pseudonymes ne sont pas gérés.
Un Book est une publication, mais n'a pas d'auteur (Author), à la place, elle a un éditeur (Editor).
Une publication n'a qu'au plus un publisher.
Publication.ee est un lien absolu vers une version électronique du document, tandis que publication.url est soit un lien relatif à partir de dblp.org, soit équivalent à publication.ee.
On suppose que deux journaux ou deux publishers ne peuvent avoir le même nom.

DBLP ne donne pas l'isbn de toutes les publications.

Une publication peut avoir plusieurs auteurs.

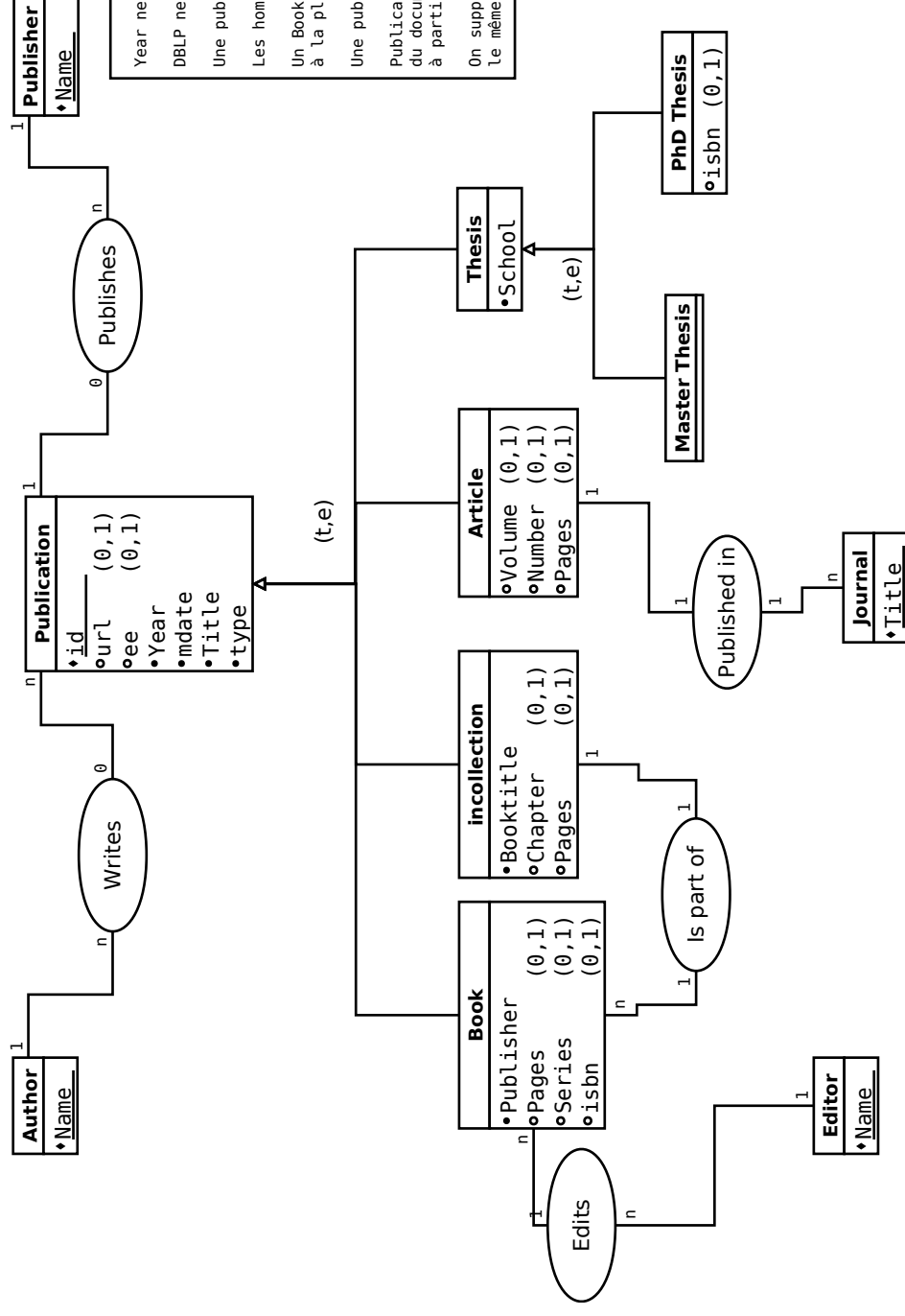
Les homonymes et pseudonymes ne sont pas gérés.

Un Book est une publication, mais n'a pas d'auteur (Author), à la place, elle a un éditeur (Editor).

Une publication n'a qu'au plus un publisher.

Publication.ee est un lien absolu vers une version électronique du document, tandis que publication.url est soit un lien relatif à partir de dblp.org, soit équivalent à publication.ee.

On suppose que deux journaux ou deux publishers ne peuvent avoir le même nom.



Admin
♦ <u>id</u>
• password

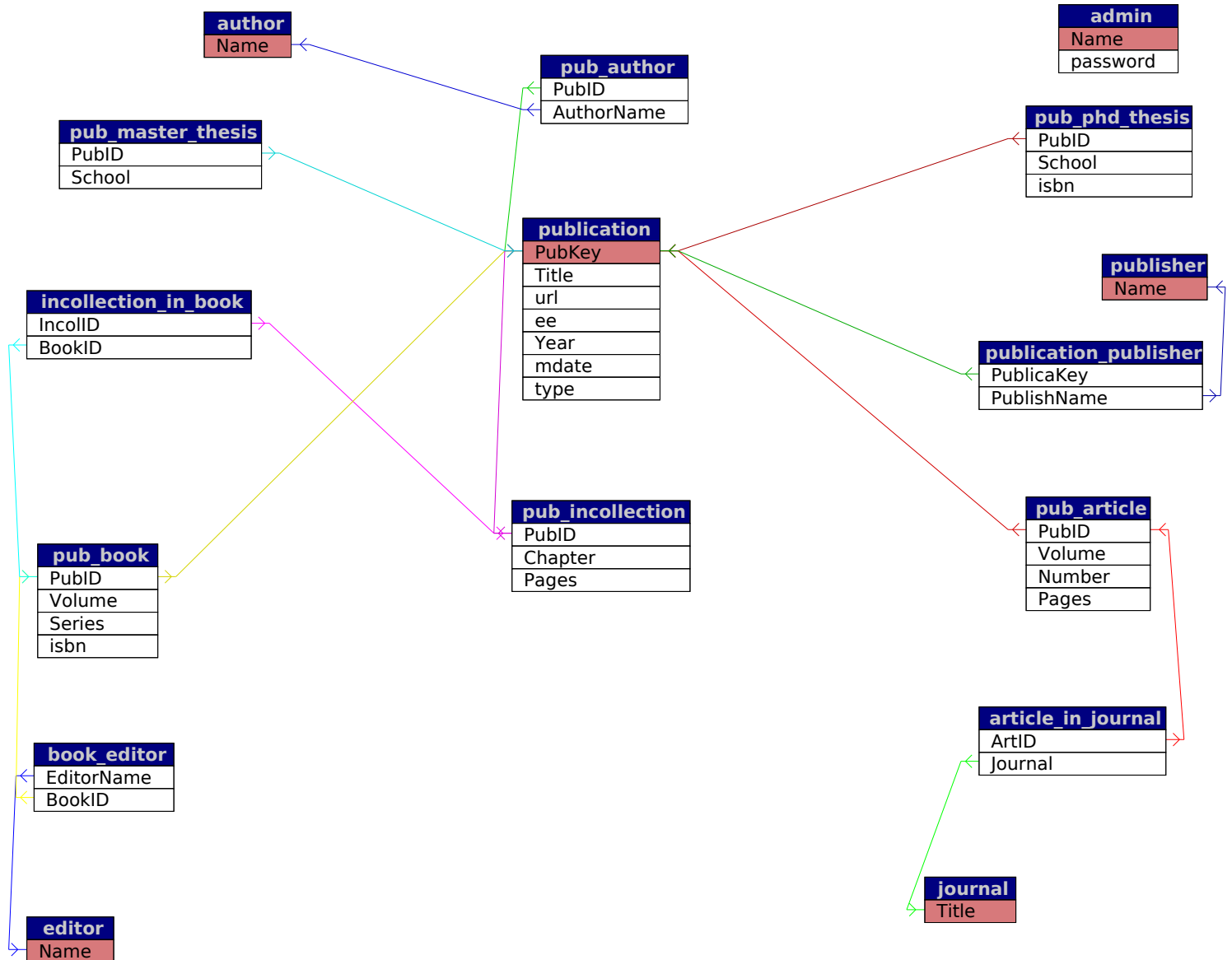


Table	Action	Rows
<input type="checkbox"/> admin	Browse Structure Search Insert Empty Drop	2
<input type="checkbox"/> article_in_journal	Browse Structure Search Insert Empty Drop	~292,851
<input type="checkbox"/> author	Browse Structure Search Insert Empty Drop	~379,629
<input type="checkbox"/> book_editor	Browse Structure Search Insert Empty Drop	831
<input type="checkbox"/> editor	Browse Structure Search Insert Empty Drop	668
<input type="checkbox"/> incollection_in_book	Browse Structure Search Insert Empty Drop	17,387
<input type="checkbox"/> journal	Browse Structure Search Insert Empty Drop	1,179
<input type="checkbox"/> publication	Browse Structure Search Insert Empty Drop	~327,451
<input type="checkbox"/> publication_publisher	Browse Structure Search Insert Empty Drop	3,207
<input type="checkbox"/> publisher	Browse Structure Search Insert Empty Drop	109
<input type="checkbox"/> pub_article	Browse Structure Search Insert Empty Drop	~313,722
<input type="checkbox"/> pub_author	Browse Structure Search Insert Empty Drop	~925,718
<input type="checkbox"/> pub_book	Browse Structure Search Insert Empty Drop	3,224
<input type="checkbox"/> pub_incollection	Browse Structure Search Insert Empty Drop	17,463
<input type="checkbox"/> pub_master_thesis	Browse Structure Search Insert Empty Drop	0
<input type="checkbox"/> pub_phd_thesis	Browse Structure Search Insert Empty Drop	2,884
16 tables	Sum	~2,286,325

Figure 1: Aperçu de l'importance des tables.

2 Extraction des données

La technologie utilisée est JAVA couplée d'une part à la librairie SAX pour parcourir le fichier XML, et d'autre part à JDBC pour la connexion à la base de données.

Les tables sont d'abord créées successivement dans la classe `SQLHandler`. La classe `EntriesHandler` se charge ensuite de *parser* le fichier XML. Les données retenues sont les *article*, *book*, *incollection*, *phdthesis* et *masterthesis*. Lorsque le *parser* tombe sur une autre entité, un booléen est activé et les étapes suivantes seront ignorées. Les années sont gérées de façon similaire : lorsque la balise *year* est détectée, on vérifie que son contenant entre bien dans l'intervalle 2008-2011, sinon le booléen est aussi activé.

Arrivé à la balise de fin d'entité, les différentes requêtes sont construites et envoyées dans la table via un objet `SQLHandler`. Ces requêtes sont exécutées dans un ordre tel que les contraintes de clés étrangères sont respectées, c'est-à-dire qu'on insère d'abord l'entité dans *publication* avant de l'envoyer dans sa table plus spécifique, comme *pub_book* par exemple. Tout au long du parcours de l'entité, les éditeurs et auteurs éventuels ont été placé dans une liste et sont là aussi envoyés dans leurs tables respectives. À la fin de chaque passage, toutes les variables sont réinitialisées.

Remarques : Ce code, bien que relativement efficace pour parcourir le fichier XML (quelques minutes pour copier les données voulues dans un autre fichier), a besoin de plusieurs dizaines d'heures pour parvenir à insérer les quelques 2,5 millions d'entrées dans la base SQL. Si le matériel relativement poussif sur lequel il a été exécuté y est largement en cause, on aurait sans doute pu rendre l'exécution plus rapide en groupant des insertions dans une seule requête SQL.

3 Requêtes spéciales

3.1 R1 : Tous les auteurs qui ont chacun publié au moins une fois chaque année entre 2008 et 2010 inclus

Calcul relationnel

```

a ← (publication ⋈PubKey=PubID pub_author)
b ←  $\sigma_{Year=2008}$  (a)
c ←  $\sigma_{Year=2009}$  (a)
d ←  $\sigma_{Year=2010}$  (a)
result ←  $\pi_{AuthorName}$  (b  $\cap$  c  $\cap$  d)

```

Calcul tuple

```

{ pa1.AuthorName | pub_author(pa1)  $\wedge$   $\exists$  pa2 pa3 pub1 pub2 pub3 ( pub_author(pa2)
 $\wedge$  pub_author(pa3)  $\wedge$  publication(pub1)  $\wedge$  publication(pub2)
 $\wedge$  publication(pub3)  $\wedge$  pa1.PubID=pub1.PubKey  $\wedge$  pa2.PubID=pub2.PubKey
 $\wedge$  pa2.PubID=pub2.PubKey  $\wedge$  pub1.Year='2008'
 $\wedge$  pub2.Year='2009'  $\wedge$  pub3.Year='2010'  $\wedge$  pa1.AuthorName=pa2.AuthorName
 $\wedge$  pa1.AuthorName=pa3.AuthorName) }

```

SQL

```

SELECT DISTINCT pa1.AuthorName
FROM pub_author pa1, pub_author pa2, pub_author pa3,
publication pub1, publication pub2, publication pub3
WHERE pa1.PubID=pub1.PubKey AND pa2.PubID=pub2.PubKey AND pa3.PubID=pub3.PubKey
AND pub1.Year='2008' AND pub2.Year='2009' AND pub3.Year='2010'
AND pa1.AuthorName=pa2.AuthorName AND pa1.AuthorName=pa3.AuthorName

```

3.2 R2 : Les auteurs qui ont écrit au moins deux articles pendant la même année

Calcul relationnel

```

a ←  $\pi_{PubKey,Year}$  (publication ⋈PubKey=PubID pub_article)
b ←  $\pi_{PubKey,Year,AuthorName}$  (a ⋈PubKey=PubID pub_author)
c ← (b ⋈PubKey $\neq$ PubID $\wedge$ Year=Year $\wedge$ AuthorName=AuthorName b)
result ←  $\pi_{AuthorName}$  (c)

```

Calcul tuple

$\{pa.AuthorName \mid pub_author(pa) \wedge \exists pub1\ pub2\ art1\ art2\ (publication(pub1) \wedge publication(pub1) \wedge pub_article(art2) \wedge pub_article(art2) \wedge pub1.Year=pub2.Year \wedge pub1.PubKey=art1.PubID \wedge pub2.PubKey=art2.PubID \wedge pub1.PubKey \neq pub2.PubKey)\}$

SQL

```
SELECT pa.AuthorName
FROM pub_author pa
WHERE pa.PubID
IN (
  SELECT pub1.PubKey
  FROM publication pub1, publication pub2
  WHERE pub1.PubKey != pub2.PubKey
  AND pub1.Year = pub2.Year
  AND pub1.PubKey
  IN (
    SELECT art.PubID
    FROM pub_article art
  )
  AND pub2.PubKey
  IN (
    SELECT art.PubID
    FROM pub_article art
  )
)
```

3.3 R3 : Les auteurs Y qui sont à une distance 2 d'un auteur X. Un auteur Y est à une distance 1 d'un auteur X si ces deux auteurs ont co-écrit un article

Calcul relationnel

$h \leftarrow \pi_{AuthorName, PubID} (pub_author \bowtie_{PubID=PubID} pub_article)$
 $a \leftarrow \sigma_{AuthorName='Y'} (h)$
 $b(PubY, AuthorY) \leftarrow a$
 $c \leftarrow \pi_{PubY, AuthorY, AuthorName} (b \bowtie_{PubID=PubY \wedge AuthorName \neq AuthorY} h)$
 $d(PubY, AuthorY, AuthorInt) \leftarrow c$
 $e \leftarrow \pi_{PubY, AuthorY, AuthorInt, PubID} (d \bowtie_{PubID \neq PubY \wedge AuthorName \neq AuthorY \wedge AuthorName \neq AuthorInt} h)$
 $f(d(PubY, AuthorY, AuthorInt, PubInt) \leftarrow e$
 $result \leftarrow \pi_{AuthorName} (f \bowtie_{PubID \neq PubY \wedge AuthorName \neq AuthorY \wedge AuthorName \neq AuthorInt \wedge PubID=PubInt} h)$

Calcul tuple

$\{pa1.AuthorName \mid pub_author(pa1) \wedge \exists pa2\ pa3\ pa4\ art1\ art2\ (pub_author(pa2) \wedge pub_author(pa3) \wedge pub_author(pa4) \wedge pub_article(art1) \wedge pub_article(art2) \wedge pa1.PubID=art1.PubID \wedge pa2.PubID=art1.PubID \wedge pa3.PubID=art2.PubID \wedge pa4.PubID=art2.PubID \wedge pa4.AuthorName='Y' \wedge pa3.AuthorName=pa2.AuthorName)$

$$\wedge \text{art2.PubID} \neq \text{art1.PubID} \wedge \text{pa1.AuthorName} = \text{pa2.AuthorName} \\ \wedge \text{pa3.AuthorName} = \text{pa4.AuthorName})\}$$

SQL

```
SELECT DISTINCT pa1.AuthorName
FROM pub_author pa1, pub_author pa2, pub_author pa3, pub_author pa4,
pub_article art1, pub_article art2
WHERE pa1.PubID = art1.PubID
AND pa2.PubID = art1.PubID
AND pa3.PubID = art2.PubID
AND pa4.PubID = art2.PubID
AND pa4.AuthorName = 'Sandra Haseloff'
AND pa3.AuthorName = pa2.AuthorName
AND art2.PubID != art1.PubID
AND pa1.AuthorName != pa2.AuthorName
AND pa3.AuthorName != pa4.AuthorName
```

3.4 R4 : Les articles qui n'ont aucun docteur parmi leurs auteurs, où un docteur est défini comme un auteur pour lequel la base de données contient une thèse de doctorat dont il est le seul auteur

Calcul relationnel

Sélection des docteurs :

$$a \leftarrow \pi_{AuthorName, PubID} (\text{pub_author} \bowtie_{PubID=PubID} \text{pub_phd_thesis}) \\ b(AuthorBis, PubBis) \leftarrow a \\ c \leftarrow (a \bowtie_{PubID \neq PubBis \wedge AuthorName = AuthorBis} b) \\ d \leftarrow \sigma_{AuthorBis = null} (c) \\ e \leftarrow \pi_{AuthorName} (d)$$

Sélection des auteurs d'article :

$$g \leftarrow \pi_{AuthorName} (\text{pub_author} \bowtie_{PubID=PubID} \text{pub_article})$$

Retrait des docteurs :

$$h \leftarrow g - e \\ \text{result} \leftarrow \pi_{PubID} (h \bowtie_{PubID=PubID} \text{pub_article})$$

Calcul tuple

$$\{\text{art.PubID} \mid \text{pub_article}(\text{art}) \wedge \nexists \text{pa} (\text{pub_author}(\text{pa}) \wedge \text{pa.PubID} = \text{art.PubID} \\ \wedge \exists \text{doc1 phd} \nexists \text{doc2} (\text{pub_author}(\text{doc1}) \wedge \text{pub_author}(\text{doc2}) \\ \wedge \text{pub_phd_thesis}(\text{phd}) \wedge \text{pa.Name} = \text{doc1.Name} \wedge \text{doc1.Name} \neq \text{doc2.Name} \\ \wedge \text{doc2.PubID} = \text{phd.PubID} \wedge \text{doc1.PubID} = \text{phd.PubID}))\}$$

SQL

```
SELECT pa1.PubID
FROM Pub_author pa1
WHERE pa1.AuthorName NOT IN (
    SELECT pa.AuthorName
```



```

FROM Pub_author pa WHERE pa.PubID IN (
    SELECT phd.PubID FROM Pub_PhD_Thesis phd
)
GROUP BY pa.PubID
HAVING COUNT(pa.AuthorName)=1
)
AND pa1.PubID IN (
    SELECT art.PubID FROM Pub_Article art
)

```

3.5 R5 : Les acteurs ayant publié dans le plus de journaux différents

SQL

```

SELECT pa1.AuthorName
FROM Pub_author pa1, Article_in_Journal j
WHERE j.ArtID = pa1.PubID
GROUP BY pa1.AuthorName
HAVING COUNT( DISTINCT j.Journal ) >=
ALL (
    SELECT COUNT( DISTINCT j2.journal )
    FROM Pub_author pa2, Article_in_Journal j2
    WHERE j2.ArtID = pa2.PubID
    GROUP BY pa2.AuthorName
)

```

3.6 R6 : Les journaux avec leur nombre total d'articles, le nombre d'articles moyen par an et le nombre d'auteurs moyen par article depuis leur année de création et ce pour tous les journaux dont le nombre de volumes par an est supérieur au nombre de volumes moyen par an des journaux

SQL

```

SELECT aij1.Journal, COUNT( art1.PubID ) AS art_count,
COUNT(art1.PubID)/(2012-MIN_YEAR.minYear) AS art_avg,
COUNT(art1.PubID)/TOTAL_AUTHOR.totalAut AS author_average
FROM article_in_journal aij1, pub_article art1, (
    SELECT MIN(pub.Year) AS 'minYear', aij2.Journal AS 'minYearJour'
    FROM publication pub, article_in_journal aij2
    WHERE pub.PubKey=aij2.ArtID
    GROUP BY aij2.Journal
) MIN_YEAR, -- get minimum year of publication for each journal
(
    SELECT COUNT(pa.AuthorName) AS totalAut
    FROM pub_author pa, pub_article art2
    WHERE pa.PubID=art2.PubID
    GROUP BY pa.PubID
) TOTAL_AUTHOR

WHERE art1.PubID = aij1.ArtID AND MIN_YEAR.minYearJour=aij1.Journal

```

```

GROUP BY aij1.Journal
HAVING MAX( art1.Volume ) > (-- keep only journal above average
    SELECT AVG( volume )
    FROM (
        SELECT art.Volume AS 'volume'
        FROM pub_article art, article_in_journal aij
        WHERE art.PubID = aij.ArtID
        GROUP BY aij.Journal
        HAVING MAX( art.Volume )
    ) VOLUME_PER_JOURNAL -- get average volume per year for each journal
)

```

4 Application

La communication avec la base de données se fait via la bibliothèque PDO de PHP et l’affichage est en HTML, épaulé par un brin de JavaScript. Aucun CSS n’a été écrit, la mise en page est brute de décoffrage sans pour autant manquer de structure.

4.1 Recherche basique et avancée

La page d’accueil propose deux formulaires de recherche : un pour les auteurs et un autre pour les publications.

La recherche d’auteur permet d’entrer un nom, même partiel, d’auteur et affiche les résultat dans un nouvel onglet avec la page `search_author.php`. Chacun des résultats est un hyperlien vers une page donnant les détails de l’auteur sélectionné (`details_author.php`) en affichant toutes ses publications ainsi que leur type. À leur tour, les publications sont des hyperliens vers une page détaillée de la publication. Cette page permet aussi d’afficher tous les co-auteurs de la personne choisie. Ci-dessous, un extrait du code PHP permettant de trouver tous les co-auteurs.

```

$sql='SELECT DISTINCT pa.AuthorName FROM pub_author pa, pub_author pa2 WHERE
pa.PubID=pa2.PubID AND pa2.AuthorName=\'\'.$author.\'\' AND
pa.AuthorName!=pa2.AuthorName';

```

La recherche de publication se passe de façon similaire, sauf que chaque type de publication a son propre script PHP permettant d’afficher les détails.

Les paramètres de recherche avancée s’affichent dynamiquement en fonction du type de publication choisie. Du côté du script PHP, la présence de chaque champ est testée dans le fichier `search_title.php`. En cas de nouvelle entité, un champ est ajouté au tableau `$tableArray`, et deux autre tableaux s’occupent des options subséquentes au WHERE. Les options sont ensuite concaténées comme suit :

```

$qry_param=count($paramArray)>0 ? " WHERE ".implode(' AND ', $paramArray) : "";

```

4.2 Implémentation des requêtes spéciales

Toutes les requêtes spéciales sont accessibles à partir du lien « Special request » sur la page d’accueil. Ce lien redirige vers `special_request.php` qui propose à son tour 6 liens vers les différentes requêtes, avec la possibilité d’introduire un nom d’auteur pour la requête 3.

4.3 Panneau d'administration

Toujours sur la page d'accueil, il est possible de s'identifier afin de faire apparaître un lien vers le panneau d'administration (lui-même protégé par une variable de session, il ne suffit pas d'entrer l'URL pour y avoir accès).

Cet espace permet tout d'abord d'enregistrer un nouvel administrateur. Les mots de passe sont cryptés en SHA-256.

Il permet aussi d'interagir avec la base de données en y insérant, mettant à jour ou supprimant des données. Les trois **radio button** permettant de changer la cible du formulaire. Cette astuce ainsi que les suivantes évitent d'avoir à afficher de nombreux champs redondants. La liste déroulante **Entity type** permet ensuite de sélectionner le type d'entité à administrer, et affiche de nouveaux champs de formulaire en conséquence.

Chaque fonctionnalité, insert, update et delete, a droit à son script PHP. Ci-suivent quelques extraits de code.

Mise à jour :

```
$paramArray=array();
    if(isset($_POST['key'])) //Vérification du champ
        $paramArray[]='PubKey='.$_POST['key'].'\'';
if(count($paramArray)>0) {
    $qry_param=implode(' ', $paramArray);
    $sql='UPDATE publication SET '.$qry_param.
        ' WHERE PubKey='.$_POST['entity_search'].'\'';
    $send=$db->query($sql);
}
```

Insertion :

```
else if($_POST['entity_choice']=="master_thesis") {
    $sql='INSERT INTO pub_master_thesis VALUES(\''
        .$_POST['key'].'\','.$_POST['school'].'\')';
    $send=$db->query($sql);
    if(!$send) {
        echo 'FAILURE';
        backToPrevious(5);//Fonction renvoyant à la page précédente
    }
}
```

En ce qui concerne la suppression, il faut être attentif à l'ordre de suppression sous peine de ne plus respecter les contraintes de clés étrangères. Ainsi, quand on désire supprimer un livre, il faut d'abord supprimer tous ses incollections. Il en va de même avec les article d'un journal qu'on voudrait faire disparaître. Ci-suit un extrait de code montrant comment se débarrasser des incollections d'un livre.

```
$sql='SELECT iib.IncolID FROM incollection_in_book iib WHERE iib.BookID=\''
    .$_POST['entity_search'].'\'';
$result=$db->query($sql);
while($inc=$result->fetch()) {
    $sql='DELETE FROM incollection_in_book WHERE IncolID='.$inc['IncolID'].'\'';
    $db->query($sql);
    $sql='DELETE FROM pub_incollection WHERE PubID='.$inc['IncolID'].'\'';
```

```
$db->query($sql);
$sql='DELETE FROM publication WHERE PubKey=\''.$inc['IncolID'].'\'';
$db->query($sql);
$sql='DELETE FROM pub_author WHERE PubID=\''.$inc['IncolID'].'\'';
$db->query($sql);
$sql='DELETE FROM publication_publisher WHERE PublicaKey=\''
    .$inc['IncolID'].'\'';
$db->query($sql);
}
```