CS-323 : Project delivrable

This is first deliverable for the *Bookshelf* project of the introduction to database course (spring 2016 - Pr. Anastasia Ailamaki), which sums up our work. It also contains some justifications.

March 2016 - Team n3

Jeremy Hottinger
259573
jeremy.hottinger@epfl.ch

Aurlien Soccard
235746
aurelien.soccard@epfl.ch

To Stocco
235744
teo.stocco@epfl.ch

Contents

I Delivrable 1 **3**

1 Justifications **3**

2 ER model **3**

3 SQL schema **5**
3.1 Entities . . . 5
3.2 Relations . . . 7
3.3 Foreign keys . . . 8
3.3.1 Authors . . . 8
3.3.2 Publication authors . . . 8
3.3.3 Publications . . . 8
3.3.4 Publication contents . . . 9
3.3.5 Publishers . . . 9
3.3.6 Publication series . . . 9
3.3.7 Titles . . . 9
3.3.8 Reviews . . . 9
3.3.9 Webpages . . . 10
3.3.10 Title awards . . . 10
3.3.11 Title tags . . . 10
3.3.12 Awards . . . 10
3.3.13 Award categories . . . 10
3.3.14 Award types . . . 11

4 Appendix : Relational Model **11**

II Delivrable 2 **13**

5 Modifications and improvements **13**
5.1 Justifications . . . 13
5.2 New ER Model . . . 14

6 Queries implementation **15**
6.1 For every year, output the year and the number of publications for said year . . . 15
6.2 Output the names of the ten authors with most publications . . . 15
6.3 What are the names of the youngest and oldest authors to publish something in 2010? . . . 16
6.4 How many comics (graphic titles) have publications with less than 50 pages, less than 100 pages, and more (or equal) than 100 pages? . . . 16
6.5 For every publisher, calculate the average price of its published novels (the ones that have a dollar price) . . . 17
6.6 What is the name of the author with the highest number of titles that are tagged as science fiction? 18
6.7 List the three most popular titles (i.e., the ones with the most awards and reviews) . . . 18

7 Interface **19**
7.1 Screenshots . . . 19
7.1.1 Menu . . . 19
7.1.2 Selection . . . 19
7.1.3 Preset . . . 19
7.1.4 Search . . . 20
7.1.5 Insert . . . 20
7.1.6 Result . . . 21
7.2 Implementation details . . . 21

III Delivrable 3 **22**

Part I
Delivrable 1

1 Justifications

We started to take a look at the given data and tried to understand how tables where connected at a first glance by drawing a really simplified diagram. Once this was done, we started to look at more deeply to the data : how are they stored? May they be empty? Are they all relevant? Our major decision was not to drop any table, but only add and remove some fields to entities.

First, for everything that is related to publications, we quickly figured out how things were working. Therefore, we quickly decided to drop some of the unused id (such as its publication author or its publication content). As a primary key, we made the choice to inflate it using constraints. Furthermore, we also decided to separate the price into two fields, one for the amount and one for the currency, this solution being much more efficient while looking for some price range for instance.

Even though both publications and titles have each one a related type field we did not find it relevant to split them into different tables and/or explicit a IS-A relationship because of the way the datas are gathered.

Another part of the work was to analyse how tables were connected to remove potential redundancy. For instance, an award has a category but also a type, and a category has a type. These two types being always the same, we needed to drop out some content, what we've done by removing the type entry in a award. We also realise that, for instance in the author table, among the 80'000 birth places, only 10% were unique so we have started thinking about changing this attribute into a placeID and create a places table. There were some advantages (no redundancy, less space consumed) but some drawbacks (2 queries instead of one for each author) that finally made us stay with the given configuration (however, if another would be using a place, we will have done that).

Furthermore, to write properly the creation of the table, we needed to know exactly what entry may be null, and which ones could not. This was done by inspecting carefully the data and reading correctly the instructions. Then, the dilemma was for notes and web pages tables. Indeed, for note, we only have two entries : the ID and the raw note, whereas for a website, there are an ID, an URL but also some other IDs to relate it to other entities : among these 7 IDs, only one The type of each entry is explained in the following paragraph. Our final has been not the change the structure of these two tables since there is no WebpageID entry in tables as there is for note, but whether a web page is directly connected to the ID of its corresponding entity. Therefore, it implies to add an entry into these 7 tables, and we decided not to do this for practical reasons. Besides, we also thought about simply adding a simply URL entry into the note table, but this solution was also not satisfactory since before insert we would need to do a map from web page to note (not that hard) but since primary keys are only unique in one table, an author and a publication may have the same ID and this would imply much more work to know which website belongs to who.

As publications and titles both have a type we used an enum as a compromise between having a raw string field and creating normalisation for it. This avoids the redundancy and increase data consistency. Note that it would not be an option if the dataset usage would include adding new types (normalising is better in this case).

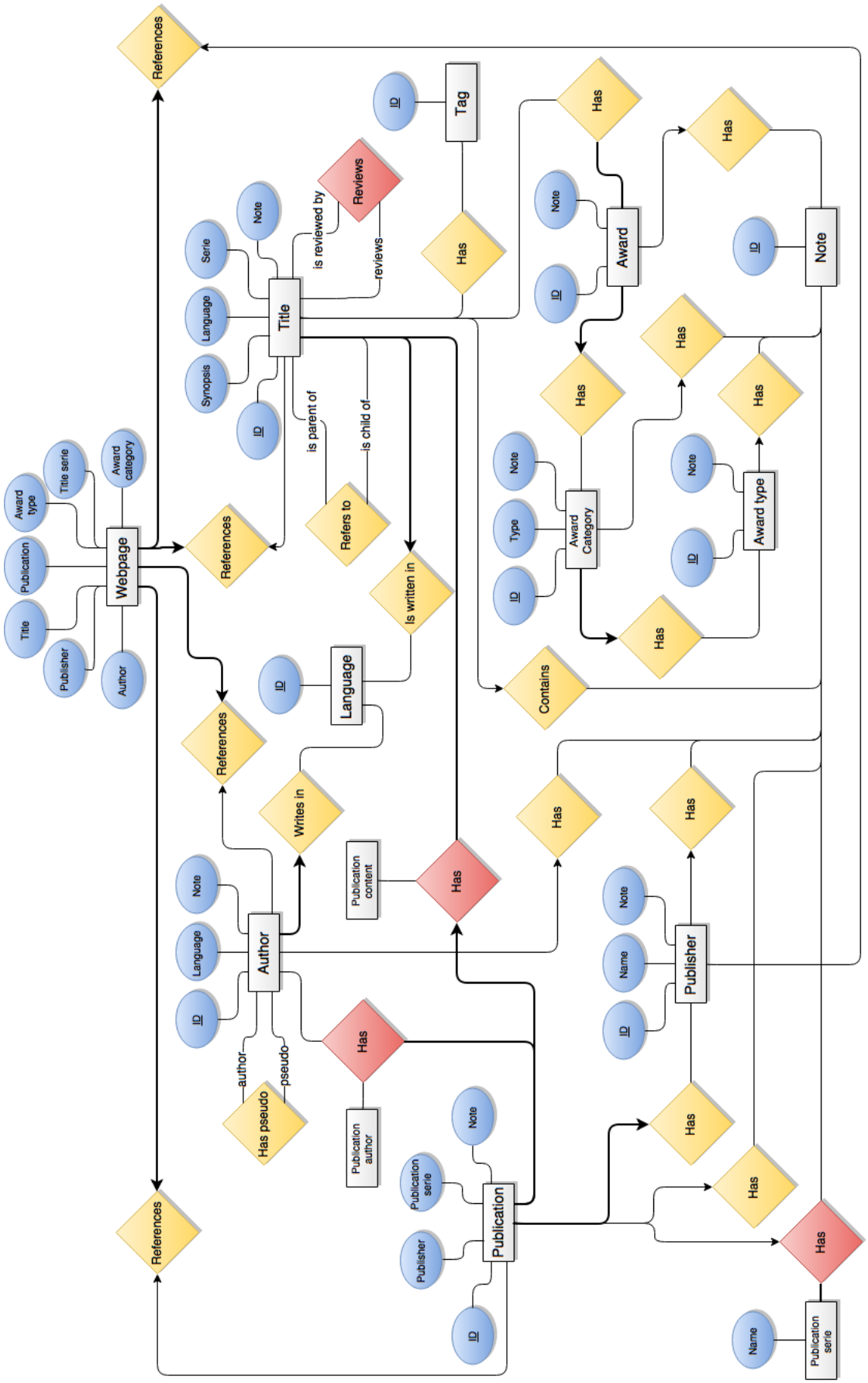
Last but not least, some intensive parsing has also been performed to know exactly how many characters were required for instance for each field, since we have encountered some difficulties with Cyrillic. We quickly realised that we should definitely parse these values before inserting them, it presents the advantages of being less memory consuming (6 characters become 1) and also the process of conversion is only done one time.

2 ER model

This entity-relation model only contains fields which are either directly use for relation or as primary key (underline then). For a more complete view of the table, refers to the next section. Fields are coloured in blue, direct relationships in yellow, relationships through a table in red.
Note : If you find this diagram too small, a larger version is available online at the following address :

`https://documents.epfl.ch/users/s/so/soccard/private/DBMS`

(access restricted to the db2016 group as defined in the EPFL AD directory).


References
Tag
ID
Has
Reviews
is reviewed by
reviews
Note
Serie
Language
Synopsis
Title
ID
is parent of
is child of
Refers to
Award
Note
ID
Has
Award type
Title serie
Award category
Publication
Title
Publisher
Author
Webpage
References
Is written in
Language
ID
Contains
Writes in
Publication content
Author
Language
ID
author
pseudo
Has pseudo
Publication author
Publication serie
Publisher
Publication
Name
Publication serie

3 SQL schema

3.1 Entities

```
CREATE TYPE PUBLICATION_TYPE AS ENUM ('ANTHOLOGY', 'COLLECTION', 'MAGAZINE',
        'NONFICTION', 'NOVEL', 'OMNIBUS', 'FANZINE', 'CHAPBOOK');

CREATE TYPE TITLE_TYPE AS ENUM ('ANTHOLOGY', 'BACKCOVERART', 'COLLECTION',
        'COVERART', 'INTERIORART', 'EDITOR', 'ESSAY', 'INTERVIEW', 'NOVEL',
        'NONFICTION', 'OMNIBUS', 'POEM', 'REVIEW', 'SERIAL', 'SHORTFICTION',
        'CHAPBOOK');

CREATE TABLE authors
(
  id           INT PRIMARY KEY NOT NULL,
  name         VARCHAR(256)    NOT NULL,
  legal_name   VARCHAR(256),
  last_name    VARCHAR(256),
  pseudonym    INT, -- fk
  birth_place  VARCHAR(256),
  birth_date   DATE,
  death_date   DATE,
  email        VARCHAR(256),
  image        VARCHAR(256),
  language_id  INT, -- fk
  note_id      INT -- fk
)

CREATE TABLE publications
(
  id              INT PRIMARY KEY        NOT NULL,
  title           VARCHAR(256)           NOT NULL,
  date_pub        DATE                   NOT NULL,
  publisher_id    INT                    NOT NULL, -- fk
  pages           INT,
  preface         INT,
  packaging_type  VARCHAR(16)            NOT NULL,
  type            PUBLICATON_TYPE    NOT NULL,
  isbn            BIGINT,
  cover           VARCHAR(256),
  price           FLOAT,
  currency        VARCHAR(8),
  pub_series_id   INT, -- fk
  pub_series_num  INT,
  note_id         INT -- fk
)
```

```
CREATE TABLE titles
(
  id           INT PRIMARY KEY NOT NULL,
  title        VARCHAR(256)    NOT NULL,
  translator   VARCHAR(256),
  synopsis     INT, -- fk
  note_id      INT, -- fk
  series_id    INT, -- fk
  series_num   INT,
  story_length VARCHAR(256),
  type         TITLE_TYPE,
  parent       INT             NOT NULL DEFAULT 0, -- fk
  language_id  INT, -- fk
  graphic      BOOLEAN         NOT NULL
)

CREATE TABLE languages
(
  id     INT PRIMARY KEY NOT NULL,
  name   VARCHAR(256)    NOT NULL,
  code   CHAR(3)         NOT NULL UNIQUE,
  script BOOLEAN
)

CREATE TABLE notes
(
  id   INT PRIMARY KEY NOT NULL,
  note TEXT            NOT NULL
)

CREATE TABLE webpages
(
  id                     INT PRIMARY KEY NOT NULL,
  author_id              INT, -- fk
  publisher_id           INT, -- fk
  title_id               INT, -- fk
  url                    VARCHAR(256)    NOT NULL UNIQUE,
  publications_series_id INT, -- fk
  award_type_id          INT, -- fk
  title_series_id        INT, -- fk
  award_category_id      INT -- fk
)

CREATE TABLE tags
(
  id   INT PRIMARY KEY NOT NULL,
  name VARCHAR(256)    NOT NULL
)

CREATE TABLE titles_series
(
  id      INT PRIMARY KEY NOT NULL,
  title   VARCHAR(256)    NOT NULL,
  parent  INT DEFAULT 0, -- fk
  note_id INT -- fk
)
```

```
CREATE TABLE awards
(
  id          INT PRIMARY KEY NOT NULL,
  title       VARCHAR(256)    NOT NULL,
  date        DATE            NOT NULL,
  category_id INT             NOT NULL, -- fk
  note_id     INT -- fk
)

CREATE TABLE awards_categories
(
  id      INT PRIMARY KEY NOT NULL,
  name    VARCHAR(256)    NOT NULL,
  type_id INT             NOT NULL, -- fk
  ordr    INT,
  note_id INT -- fk
)

CREATE TABLE awards_types
(
  id          INT PRIMARY KEY NOT NULL,
  code        CHAR(2) UNIQUE,
  name        VARCHAR(256)    NOT NULL,
  note_id     INT, -- fk
  awarded_by  VARCHAR(256)    NOT NULL,
  awarded_for VARCHAR(256)    NOT NULL,
  short_name  VARCHAR(256)    NOT NULL UNIQUE,
  poll        BOOLEAN         NOT NULL,
  non_genre   BOOLEAN         NOT NULL
)

CREATE TABLE publishers
(
  id      INT PRIMARY KEY NOT NULL,
  name    VARCHAR(512)    NOT NULL,
  note_id INT -- fk
)

CREATE TABLE publications_series
(
  id      INT PRIMARY KEY NOT NULL,
  name    VARCHAR(512)    NOT NULL,
  note_id INT -- fk
)
```

3.2 Relations

```
CREATE TABLE publications_authors
(
  publication_id INT NOT NULL, -- fk
  author_id      INT NOT NULL, -- fk
  CONSTRAINT pk_publications_authors PRIMARY KEY (publication_id, author_id)
)

CREATE TABLE titles_awards
(
  title_id INT NOT NULL, -- fk
  award_id INT NOT NULL, -- fk
  CONSTRAINT pk_titles_awards PRIMARY KEY (title_id, award_id)
)
```

```
CREATE TABLE titles_tags
(
  title_id INT NOT NULL, -- fk
  tag_id   INT NOT NULL, -- fk
  CONSTRAINT pk_titles_tags PRIMARY KEY (title_id, tag_id)
)

CREATE TABLE reviews
(
  title_id  INT NOT NULL, -- fk
  review_id INT NOT NULL, -- fk
  CONSTRAINT pk_reviews PRIMARY KEY (title_id, review_id)
)

CREATE TABLE publications_contents
(
  title_id       INT NOT NULL, -- fk
  publication_id INT NOT NULL, -- fk
  CONSTRAINT pk_publications_contents PRIMARY KEY (title_id, publication_id)
)
```

3.3 Foreign keys

3.3.1 Authors

```
ALTER TABLE authors
ADD FOREIGN KEY (language_id)
REFERENCES languages (id)
ON DELETE SET NULL;

ALTER TABLE authors
ADD FOREIGN KEY (note_id)
REFERENCES notes (id)
ON DELETE SET NULL;
```

```
ALTER TABLE authors
ADD FOREIGN KEY (pseudonym)
REFERENCES authors (id)
ON DELETE CASCADE;
```

3.3.2 Publication authors

```
ALTER TABLE publications_authors
ADD FOREIGN KEY (publication_id)
REFERENCES publications (id)
ON DELETE CASCADE;
```

```
ALTER TABLE publications_authors
ADD FOREIGN KEY (author_id)
REFERENCES authors (id)
ON DELETE CASCADE;
```

3.3.3 Publications

```
ALTER TABLE publications
ADD FOREIGN KEY (publisher_id)
REFERENCES publishers (id)
ON DELETE SET NULL;

ALTER TABLE publications
ADD FOREIGN KEY (note_id)
REFERENCES notes (id)
ON DELETE SET NULL;
```

```
ALTER TABLE publications
ADD FOREIGN KEY (pub_series_id)
REFERENCES publications_series (id)
ON DELETE SET NULL;
```

3.3.4 Publication contents

```
ALTER TABLE publications_contents
ADD FOREIGN KEY (title_id)
REFERENCES titles (id)
ON DELETE CASCADE;
```

```
ALTER TABLE publications_contents
ADD FOREIGN KEY (publication_id)
REFERENCES publications (id)
ON DELETE CASCADE;
```

3.3.5 Publishers

```
ALTER TABLE publishers
ADD FOREIGN KEY (note_id)
REFERENCES notes (id)
ON DELETE SET NULL;
```

3.3.6 Publication series

```
ALTER TABLE publications_series
ADD FOREIGN KEY (note_id)
REFERENCES notes (id)
ON DELETE SET NULL;
```

3.3.7 Titles

```
ALTER TABLE titles
ADD FOREIGN KEY (synopsis)
REFERENCES notes (id)
ON DELETE SET NULL;
```

```
ALTER TABLE titles
ADD FOREIGN KEY (series_id)
REFERENCES titles_series (id)
ON DELETE SET NULL;
```

```
ALTER TABLE titles
ADD FOREIGN KEY (parent)
REFERENCES titles (id)
ON DELETE SET DEFAULT;
```

```
ALTER TABLE titles
ADD FOREIGN KEY (language_id)
REFERENCES languages (id)
ON DELETE SET NULL;
```

```
ALTER TABLE titles
ADD FOREIGN KEY (note_id)
REFERENCES notes (id)
ON DELETE SET NULL;
```

3.3.8 Reviews

```
ALTER TABLE reviews
ADD FOREIGN KEY (title_id)
REFERENCES titles (id)
ON DELETE CASCADE;
```

```
ALTER TABLE reviews
ADD FOREIGN KEY (review_id)
REFERENCES titles (id)
ON DELETE CASCADE;
```

3.3.9 Webpages

```
ALTER TABLE webpages
ADD FOREIGN KEY (author_id)
REFERENCES authors (id)
ON DELETE CASCADE;

ALTER TABLE webpages
ADD FOREIGN KEY (title_id)
REFERENCES titles (id)
ON DELETE CASCADE;

ALTER TABLE webpages
ADD FOREIGN KEY (award_type_id)
REFERENCES awards_types (id)
ON DELETE CASCADE;

ALTER TABLE webpages
ADD FOREIGN KEY (award_category_id)
REFERENCES awards_categories (id)
ON DELETE CASCADE;
```

```
ALTER TABLE webpages
ADD FOREIGN KEY (publisher_id)
REFERENCES publishers (id)
ON DELETE CASCADE;

ALTER TABLE webpages
ADD FOREIGN KEY (publications_series_id)
REFERENCES publications_series (id)
ON DELETE CASCADE;

ALTER TABLE webpages
ADD FOREIGN KEY (title_series_id)
REFERENCES title_series (id)
ON DELETE CASCADE;
```

3.3.10 Title awards

```
ALTER TABLE titles_awards
ADD FOREIGN KEY (title_id)
REFERENCES titles (id)
ON DELETE CASCADE;
```

```
ALTER TABLE titles_awards
ADD FOREIGN KEY (award_id)
REFERENCES awards (id)
ON DELETE CASCADE;
```

3.3.11 Title tags

```
ALTER TABLE titles_tags
ADD FOREIGN KEY (title_id)
REFERENCES titles (id)
ON DELETE CASCADE;

ALTER TABLE title_series
ADD FOREIGN KEY (parent)
REFERENCES title_series (id)
ON DELETE SET DEFAULT;
```

```
ALTER TABLE titles_tags
ADD FOREIGN KEY (tag_id)
REFERENCES tags (id)
ON DELETE CASCADE;

ALTER TABLE title_series
ADD FOREIGN KEY (note_id)
REFERENCES notes (id)
ON DELETE SET NULL;
```

3.3.12 Awards

```
ALTER TABLE awards
ADD FOREIGN KEY (category_id)
REFERENCES awards_categories (id)
ON DELETE SET NULL;
```

```
ALTER TABLE awards
ADD FOREIGN KEY (note_id)
REFERENCES notes (id)
ON DELETE SET NULL;
```

3.3.13 Award categories

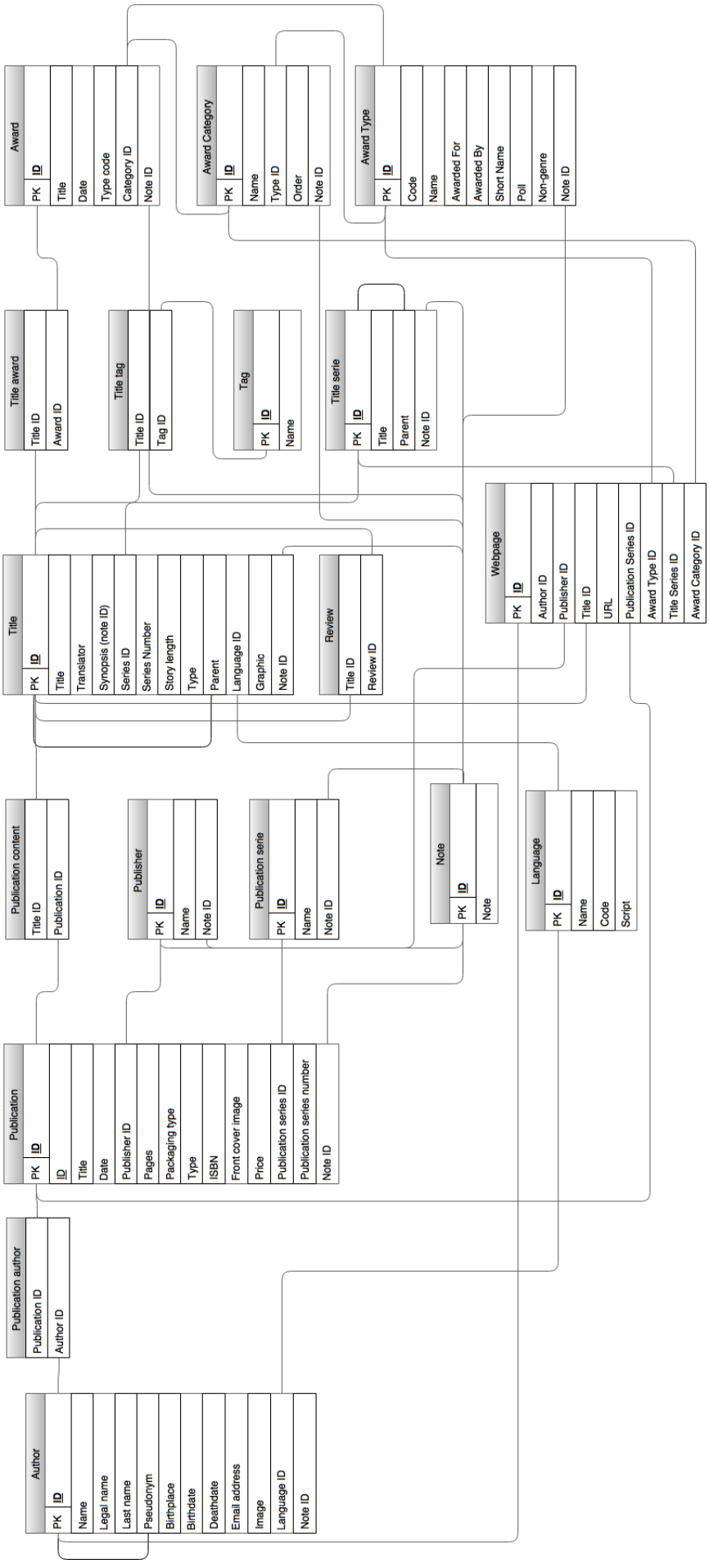
```
ALTER TABLE awards_categories
ADD FOREIGN KEY (type_id)
REFERENCES awards_types (id)
ON DELETE SET NULL;
```

```
ALTER TABLE awards_categories
ADD FOREIGN KEY (note_id)
REFERENCES notes (id)
ON DELETE SET NULL;
```

3.3.14 Award types

```
ALTER TABLE awards_types
ADD FOREIGN KEY (note_id)
REFERENCES notes (id)
ON DELETE SET NULL;
```

4 Appendix : Relational Model


Award
PK ID
Title
Date
Type code
Category ID
Note ID
Award Category
PK ID
Name
Type ID
Order
Note ID
Award Type
PK ID
Code
Name
Awarded For
Awarded By
Short Name
Poll
Non-genre
Note ID
Title award
Title ID
Award ID
Title tag
Title ID
Tag ID
Tag
PK ID
Name
Title serie
PK ID
Title
Parent
Note ID
Webpage
PK ID
Author ID
Publisher ID
Title ID
URL
Publication Series ID
Award Type ID
Title Series ID
Award Category ID
Title
PK ID
Title
Translator
Synopsis (note ID)
Series ID
Series Number
Story length
Type
Parent
Language ID
Graphic
Note ID
Review
Title ID
Review ID
Publication content
Title ID
Publication ID
Publisher
PK ID
Name
Note ID
Publication serie
PK ID
Name
Note ID
Note
PK ID
Note
Language
PK ID
Name
Code
Script
Publication
PK ID
ID
Title
Date
Publisher ID
Pages
Packaging type
Type
ISBN
Front cover image
Price
Publication series ID
Publication series number
Note ID
Publication author
Publication ID
Author ID
Author
PK ID
Name
Legal name
Last name
Pseudonym
Birthplace
Birthdate
Deathdate
Email address
Image
Language ID
Note ID

Part II
Delivrable 2

5 Modifications and improvements

5.1 Justifications

We are now considering `award_category` as a weak entity of `award_type` since it doesnt make sense for a category to exist without type. Therefore, the primary key of the category is now not only `cat_ID` but also a tuple containing `cat_ID` and `type_ID`. This leads us to store again in our award table the `type_ID` attribute to make the connection between awards and their categories.
After analyzing the data, the graphic attribute may be null, we lead us to remove the NOT NULL constraint on this attribute.
While inserting, we removed some of the notes entries due to bad formatting or empty content. Advanced data such as dates or combined page numbers have been parse the best possible. When that was not sufficient the field might have been replaced by null or replaced by the nearest value (e.g. 0/0/xxxx to 1/1/xxxx) as discussed in class. The translator field of titles has been extracted into a many to many relationship. The parser also been adapted to manage \\t as an escape character for tabulation.

In the DDL and according to the data, the following changes have been made :

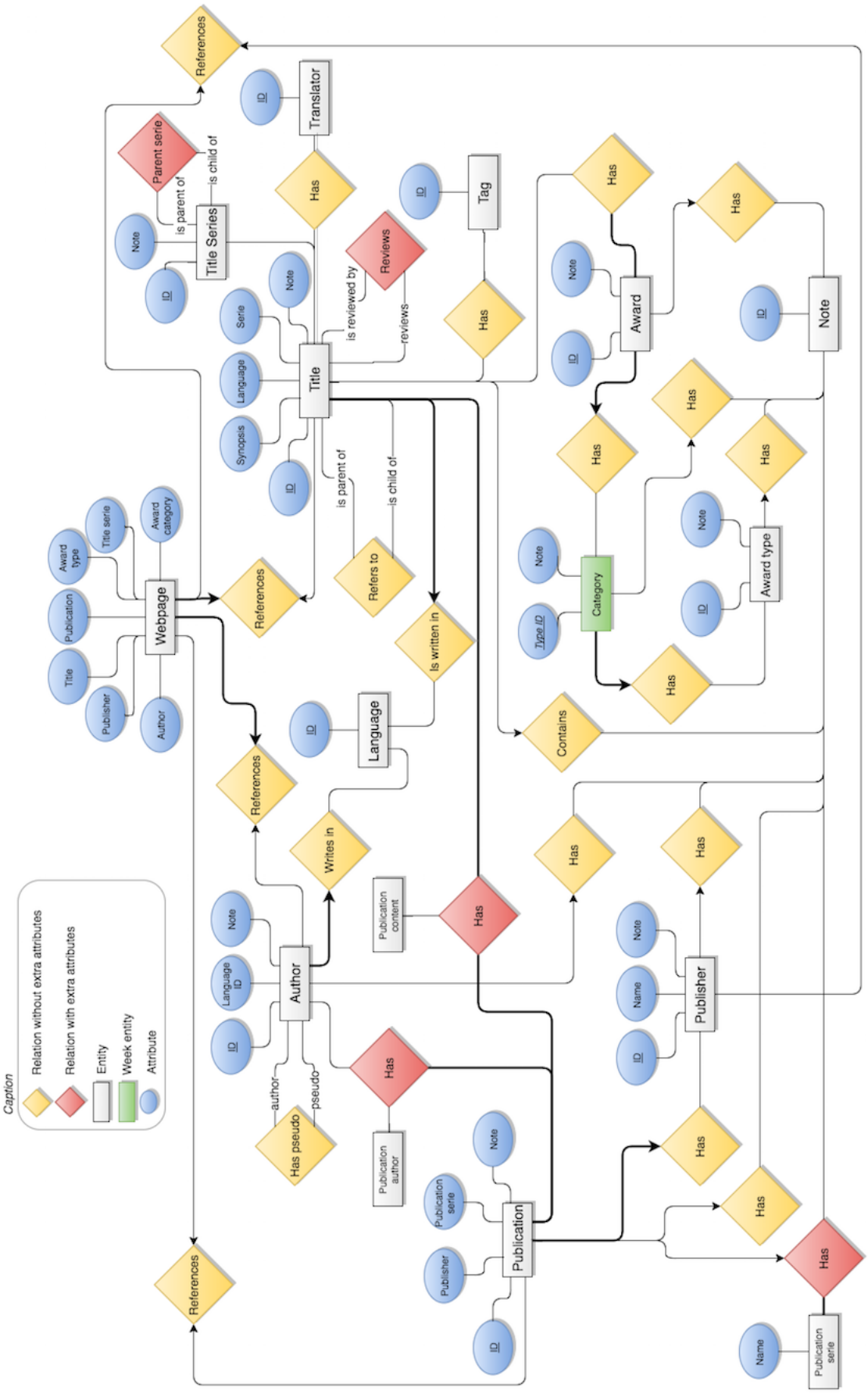
- `publisher_id` in publication is now nullable
- `type` in publication is now nullable
- `parent` in titles is now nullable
- `url` in webpages is now not unique, as many url could reference different items
- `code` in `award_type` is now not unique, as two awards could share the same initials
- drop table statements have been added to make easier the deletion and creation of the database
- dead foreign key values removal queries have been added to allow the creation of corresponding foreign keys
- a constraint ensuring at least of the entities is linked in each webpages entries has been added for not having a page referencing no entity

About cascade justifications, we decided with one of the TAs to only specify the behaviour for deleting as updating is not involved in the project. If not specified otherwise, we set the deleted field to null as they entities lifetime are independent of each others :

- `authors_id` and `pseudonym` are linked to the same person, thus requiring a cascade deletion.
- all many to many relationship have a cascade deletion, as there is no meaning of a two entity mapping missing one of the entity.
- `webpages` have a cascade deletion whenever one the referring entity is deleted.
- `awards_categories` has a cascade deletion as it is a weak entity.

The enumerations values have been kept as inline values for flexibility and simplicity reasons as discussed with one of the teaching assistants. The full inserts are batched and executed in about 12 minutes.

5.2 New ER Model



6 Queries implementation

6.1 For every year, output the year and the number of publications for said year

1. Description of logic : We are using the `DATE_PART` function provided by Postgresql to extract the year of the date stored for each publication. Then we simply do a GROUP BY on the year and count the number of entries.

2. SQL statement :

```
SELECT DATE_PART('year', p.date_pub) AS year, COUNT(*) AS count
FROM publications p
GROUP BY DATE_PART('year', p.date_pub)
ORDER BY year ASC;
```

3. Results (partial):

year	count
2005	8667
2006	8166
2007	11970
2008	14204
2009	15526
2010	19150
2011	22470
2012	22317
2013	22923
2014	21219
2015	17206
2016	1590
8888	445
9999	4

6.2 Output the names of the ten authors with most publications

1. Description of logic : This query simply joins the authors and their publications and then GROUP BY on the authors name. We just have to count the number of entries, ORDER BY that count and set a limit to 10.

2. SQL statement :

```
SELECT a.name, COUNT(*) AS count
FROM publications_authors p
INNER JOIN authors a ON p.author_id = a.id
GROUP BY a.name
ORDER BY count DESC
LIMIT 10;
```

3. Results :

name	count
uncredited	2912
Isaac Asimov	2389
Edgar Rice Burroughs	2287
Robert A. Heinlein	1878
Arthur C. Clarke	1512
Andre Norton	1505
Stephen King	1504
Robert Silverberg	1481
Philip K. Dick	1398
Terry Pratchett	1354

6.3 What are the names of the youngest and oldest authors to publish something in 2010?

1. Description of logic : Slightly more complicated this query uses nested subqueries to first select publications of 2010, then to select the maximum and minimum birth dates of author that have some publication of 2010.

2. SQL statement :

```
SELECT a.name
FROM authors a
WHERE a.birth_date = (
SELECT MAX(a.birth_date) AS max
FROM publications_authors pa
JOIN authors a ON pa.author_id = a.id
WHERE pa.publication_id IN (
  SELECT pub.id
  FROM publications pub
  WHERE DATE_PART('year', pub.date_pub) = '2010'
)) OR
a.birth_date = (
SELECT MIN(a.birth_date) AS max
FROM publications_authors pa
JOIN authors A ON pa.author_id = a.id
WHERE pa.publication_id IN (
  SELECT pub.id
  FROM publications pub
  WHERE DATE_PART('year', pub.date_pub) = '2010'
));
```

3. Results :

name
Robert Henryson
Gavin Douglas
Greg Kurzawa
Laramie Sasseville
Brooke Vaughn
Pancham Yadav
Euripides
Aubrey Smith
Augustin Lardy
Livy
Michel Saint-Romain
Gan Bao
Timothy F. Mitchell
Gottfried von Strassburg

6.4 How many comics (graphic titles) have publications with less than 50 pages, less than 100 pages, and more (or equal) than 100 pages?

1. Description of logic : After joining titles and publications, filtering them using the graphic field of titles and setting the condition on the number of pages, we just have to count the number of entries.

2. SQL statement :

```
SELECT COUNT(*)
FROM titles t
JOIN publications_contents AS pc ON t.id = pc.title_id
JOIN publications AS p ON pc.publication_id = p.id
WHERE t.graphic = 'YES'
AND p.Pages < 50;
```

```
-----------------------------------------------

SELECT COUNT(*)
FROM titles t
JOIN publications_contents AS pc ON t.id = pc.title_id
JOIN publications AS p ON pc.publication_id = p.id
WHERE t.graphic = 'YES'
AND p.Pages < 100;

-----------------------------------------------

SELECT COUNT(*)
FROM titles t
JOIN publications_contents AS pc ON t.id = pc.title_id
JOIN publications AS p ON pc.publication_id = p.id
WHERE t.graphic = 'YES'
AND p.Pages >= 100;
```

3. Results :

count
153

count
202

count
194

6.5 For every publisher, calculate the average price of its published novels (the ones that have a dollar price)

1. Description of logic : This query joins the publications and authors via the `publications_authors` table. Then it filters the publications to keep only the ones that have a price in dollar. Finally it groups everything on the authors name and do an average on the price.

2. SQL statement :

```
ELECT pr.name, AVG(p.price) AS average
FROM publications p
JOIN publishers pr ON p.publisher_id = pr.id
WHERE p.currency = '$'
GROUP BY pr.name
ORDER BY average DESC;
```

3. Results (partial, 10 first):

name	avg
Library Fellows of the Whitney Museum of American Art	2200
The Pennyroyal Press	1000
Pickering & Chatto	795
Epic Ink Books	500
The Book Club of California	375
Tradition Books	357
Charnel House	344.88235294117646
Subterranean Press & PS Publishing	300
Sirius Science Fiction	262.5
Kluwer Academic Publishers	259

6.6 What is the name of the author with the highest number of titles that are tagged as science fiction?

1. Description of logic : This query use a subquery. This subquery links authors with their publications, the publications with their titles and finally the titles with their tags. A simple filter on tags to keep only the ones that have the keyword `sf` (meaning science fiction), then group by authors name, count the entries for each authors and order by this counter. Then the main query is there to keep the name without the counter.

2. SQL statement :

```
SELECT a.name
FROM authors a
JOIN (
        SELECT a.name AS n, COUNT(*) AS count_sf
        FROM authors a
        JOIN publications_authors pa ON a.id = pa.author_id
        JOIN publications p ON pa.publication_id = p.id
        JOIN publications_contents pc ON pc.publication_id = p.id
        JOIN titles t ON pc.title_id = t.id
        JOIN titles_tags tt ON t.id = tt.title_id
        JOIN tags ON tt.tag_id = tags.id
        WHERE tags.name '%science-fiction%'
        GROUP BY a.name
        ORDER BY count_sf DESC
        LIMIT 1)
c ON c.n = a.name;
```

3. Results :

name
Maurice Level

6.7 List the three most popular titles (i.e., the ones with the most awards and reviews)

1. Description of logic : Here we order the titles on their number of awards and reviews and then simply sum them and order them on that total. Note that union is mandatory as otherwise we will miss some datas (e.g. all titles that only appear to have awards or titles).

2. SQL statement :

```
SELECT t.title, ts.count_awards + ts.count_reviews as total
FROM (
  SELECT ta.title_id, 0 AS count_reviews, COUNT(ta.award_id) AS count_awards
  FROM titles_awards ta
  GROUP BY ta.title_id
  UNION
  SELECT r.title_id, COUNT(r.review_id) AS count_reviews, 0 AS count_awards
  FROM reviews r
  GROUP BY r.title_id
) as ts
  INNER JOIN titles t on t.id = ts.title_id
ORDER BY total DESC
LIMIT 3;
```

3. Results :

title	total
The Wonderful Wizard of Oz	100
Sky Island	54
The Sea Fairies	48

7 Interface

7.1 Screenshots

7.1.1 Menu

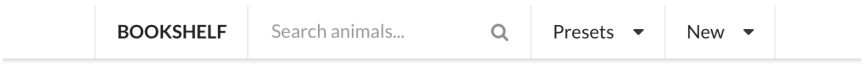


Figure 1: Menu view

7.1.2 Selection

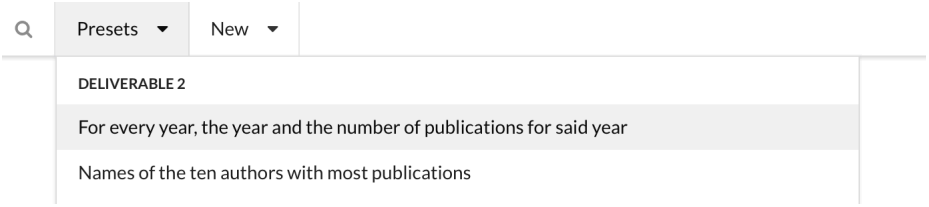


Figure 2: Selection view

7.1.3 Preset

Figure 3: Preset view

7.1.4 Search

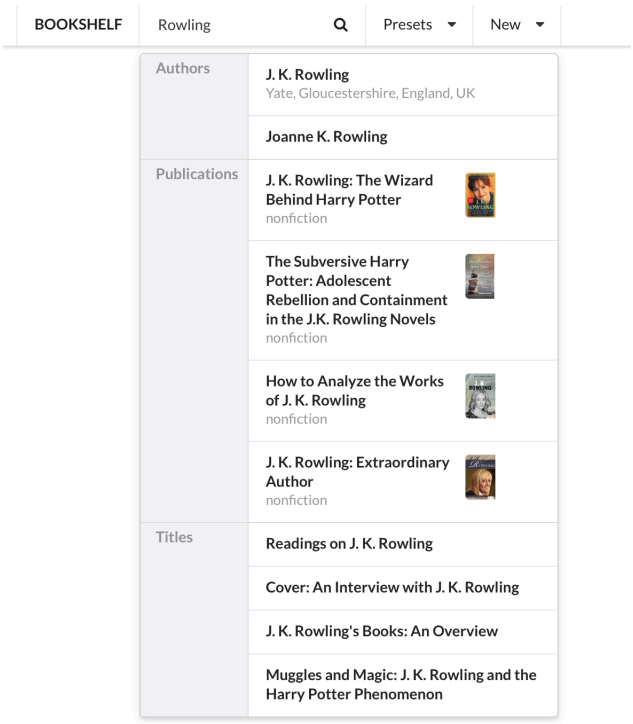


Figure 4: Search view

7.1.5 Insert

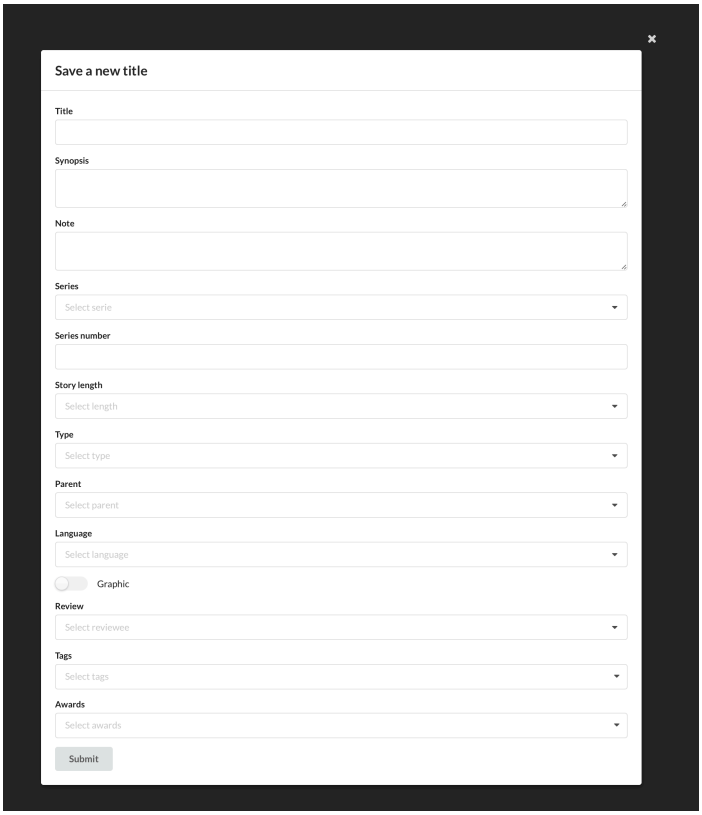


Figure 5: Insert view

7.1.6 Result

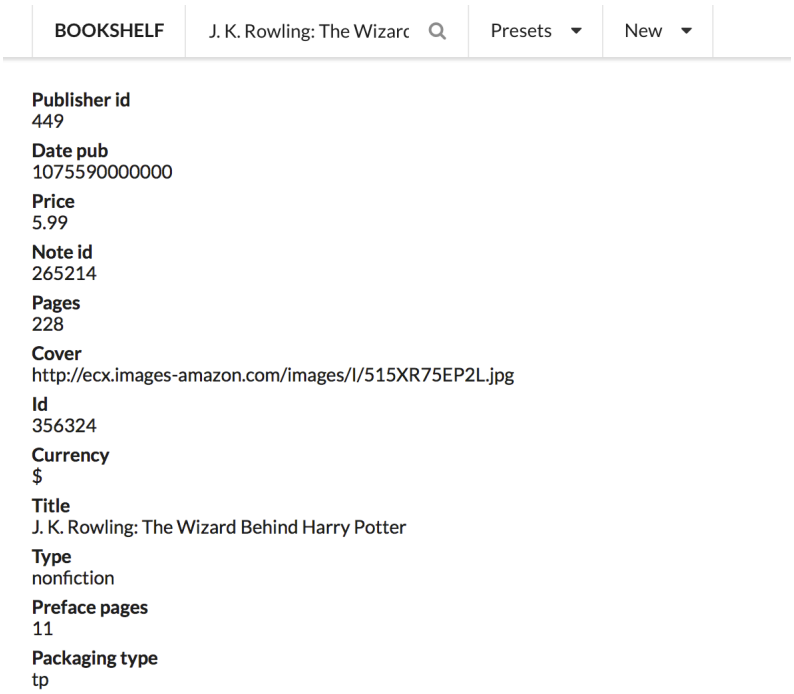


Figure 6: Result view

7.2 Implementation details

The search field is using ajax to access our backend server in Scala (cf. search). For each search query, we run in parallel 3 queries using LIKE statements to find all relevant results and send it be aggregated to the requester. We limit the number of result to 4 for visibility reason. The user is able to always enter the main field (either name or title), a secondary field for more information about the entry (but not searchable) and optionally a picture of the entry if relevant. Once the entry is select more information about the result in a formatted way (cf. result).

Part III
Delivrable 3