

# CS-323 : Deliverable 1

This is our first deliverable for the *Bookshelf* project of the introduction to database course (spring 2016 - Pr. Anastasia Ailamaki), which sums up our ER-model and schema. 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

<b>1</b>	<b>Justifications</b>	<b>2</b>
<b>2</b>	<b>ER model</b>	<b>3</b>
<b>3</b>	<b>SQL schema</b>	<b>5</b>
<b>4</b>	<b>Appendix : Relational Model</b>	<b>11</b>

## 1 Justifications

We started to take a look at the given data and tried to understand how tables were 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.

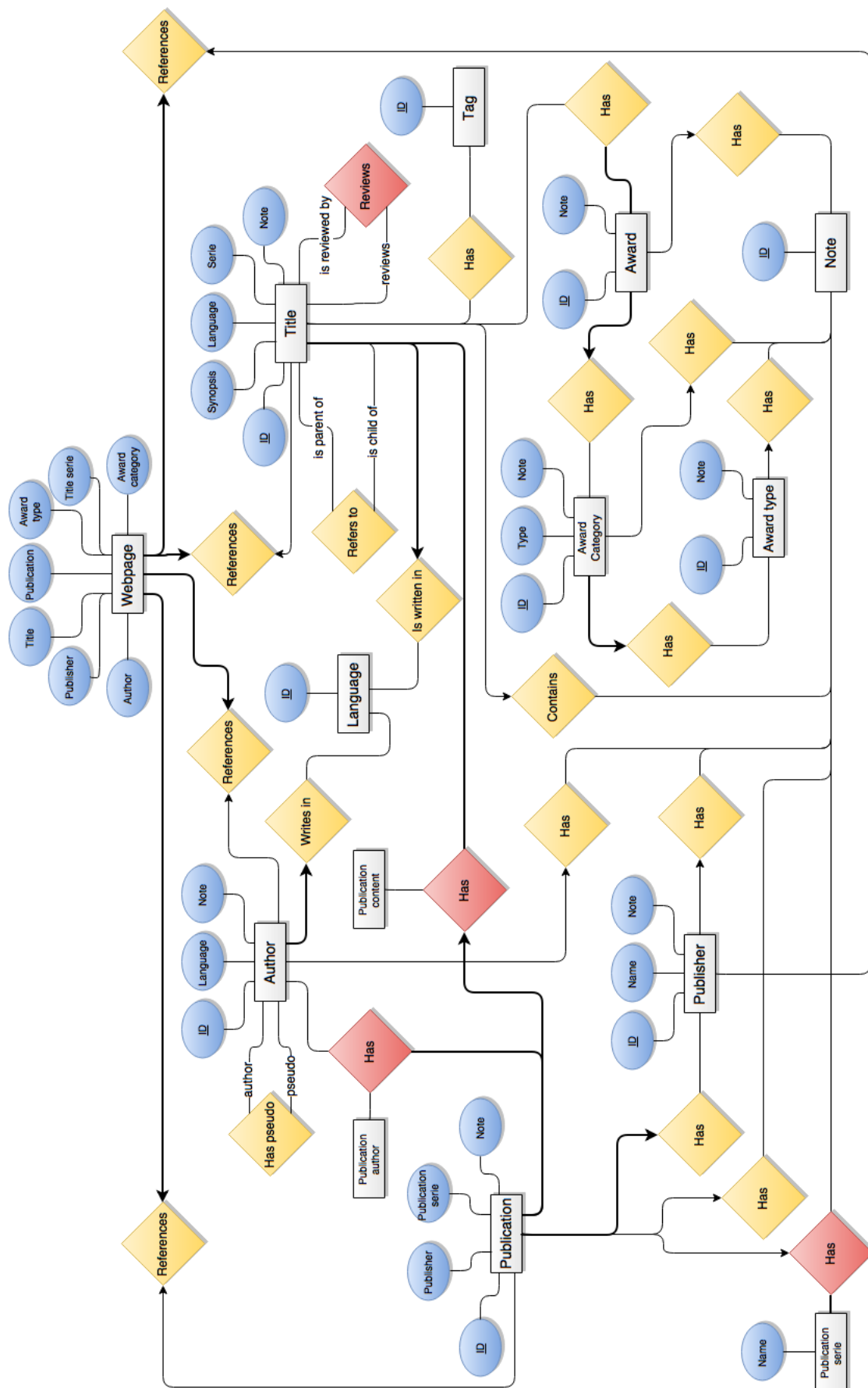
## 2 ER model

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).



## 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          PUBLICATION_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 (pseudonym) REFERENCES authors (id) ON DELETE CASCADE;
ALTER TABLE authors ADD FOREIGN KEY (note_id) REFERENCES notes (id) ON DELETE SET NULL;	

#### 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 (pub_series_id) REFERENCES publications_series (id) ON DELETE SET NULL;
ALTER TABLE publications ADD FOREIGN KEY (note_id) REFERENCES notes (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 titles_tags
ADD FOREIGN KEY (tag_id)
REFERENCES tags (id)
ON DELETE CASCADE;
```

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

```
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

