**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne

URL: http://dias.epfl.ch/

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

# Databases Project – Spring 2017

Prof. Anastasia Ailamaki

Team No: 2

Names:
Mikael Morales Gonzalez, Niroshan Vijayarasa, Charles Parzy Turlat

## Contents

# Deliverable 1

## *Assumptions*

Using the given data, we identify several attributes that are not filled most of the time, those attributes are marked in red in the ER schema. If future queries don't need those attributes we might want to delete them since they don't bring much information to the user.
By looking at the data, we deduced the relationships between the entities. Furthermore, it allowed us to lessen our constraints that we initially thought.
Entity Relationship Schema

## *Entity Relationship Schema*

### Schema

See annex 1.

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne

URL: http://dias.epfl.ch/

### Description

We found several constraints on the relationship.

- **Story:** Since the Story entity contains an issue_id attribute, we defined a relationship between Issue and Story. By looking at the data, we saw that issue_id is not always filled, hence we conclude that Story is contained in at most one issue. Furthermore, Story always contains a single type.
- **Story_Reprint:** We decided to make a role indicator because it is a relationship between two Stories.
- **Issue_Reprint:** Same reasoning was applied to issue_reprint. We decided to make a role indicator, because it is a relationship between two Issues.
- **Issue:** Issue contains several relationships. For instance it is contained in at most one series because by looking at the data, we saw that series_id is not always given. We apply the same reasoning with indicia_publisher_id.
- **Series:** Series has a lot of relationship. First at all, the country_id is always given and points to only one country. It is the same for the Language and Publisher, they are always given. Series_Publication_Type is not always given, and when it is, there is only one. So we used an at most one constraint. Furthermore, it is the same for first and last issue of a series. first_issue and last_issue are not always given, so we also used at most one constraint.
- **Publisher:** The country_id of Publisher is always given.
- **Indicia Publisher:** An Indicia_publisher is always contained in exactly one publisher and is located in exactly one country.
- **Brand Group:** A Brand Group is always contained in exactly one publisher.

## *Relational Schema*

### ER schema to Relational schema

Since we always have an exactly one or at most one constraint in every relationship, we don't need any intermediate table to represent the relationship. Moreover, defining the primary keys and the foreign keys was straightforward.
The primary keys are always the id of the entity, and the foreign keys the attributes that point to another entity.

### DDL

See annex 2.

## *General Comments*

At the beginning, we started by trying to define the relationships logically, using common sense and what we could expect to have in the real world. But then we noticed that our constraints were too restrictive. So we decided to look at the given data and conclude the relationships from there, this allowed us to lessen our constraints and provide a more flexible design.

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne

URL: http://dias.epfl.ch/

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

The work has been equally split up between the 3 members of the group, each one was given a different task and we met up every Wednesday to evaluate the progress of everyone and to define the tasks for the upcoming weeks.

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne

URL: http://dias.epfl.ch/

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

# Deliverable 2

## *Assumptions*

During this deliverable, we finished to clean the data. We made the following assumptions on the attributes: a date is only composed by a year and there is only one price per Issue record (in one currency). We made this assumptions by looking at the data, and by making statistics. We determined which form this attribute most often has and we apply this form to all the records for the given attributes. When it was impossible to convert the form of an attribute, we just set it to NULL.

Further clarification about the dates, we keep only the ones that are lower than 2017.

Moreover, for every attributes that was filled with a list of name (for example: the artists in Story), we create new tables, one containing the entity listed previously in an attribute, and one mapping this new entities to the previous ones which contained them. For example, we create an table Characters, containing every characters that were appearing in the attribute Character of Story. This table contains the ids and the names of the Characters. And we create another table, Story_Has_Characters, which map a Story (story_id) to a Character (character_id).

## *Data Loading*

## *Query Implementation*

### Query a:

*Description of logic:*

Print the brand group names with the highest number of Belgian indicia publishers.

In this query, we select the Brand_Group name of the Brand_Group whose publisher_id belongs to the list of Publishers which have the highest number of Belgian Indicia Publisher. To make this list, we select all Belgian Indicia Publisher (by checking that the country of the Indicia Publisher is Belgium) and we group them by the publisher_id. So, now we can count the number of Indicia Publisher per Publisher and keep only the publisher_id associated with the maximum number of Belgian Indicia Publisher.

*SQL statement:*

```
SELECT G.name
FROM Brand_Group G
WHERE G.publisher_id IN
        (SELECT I2.publisher_id
         FROM Indicia_Publisher I2, Country C1
         WHERE C1.name = "Belgium" AND I2.country_id = C1.id
         GROUP BY I2.publisher_id
         HAVING COUNT(*) >=
                (SELECT MAX(counted) FROM (SELECT COUNT(I3.id) AS counted
                 FROM Indicia_Publisher I3, Country C2
                 WHERE C2.name = "Belgium" AND I3.country_id = C2.id
                 GROUP BY I3.publisher_id) AS counts));
```

*Result:*

15 lines in 46ms

### Query b:

*Description of logic:*

Print the ids and names of publishers of Danish book series.

For this query, we select the id and the name of the publishers contained in the list containing the publisher id of the publisher which published Danish series.

*SQL statement:*

```
SELECT DISTINCT(P.id), P.name
FROM Publisher P
WHERE P.id IN (SELECT S.publisher_id
                FROM Series S, Country C
                WHERE C.name = "Denmark" AND S.country_id = C.id);
```

*Result:*

112 lines in 18.4ms

## Query c:

*Description of logic:*

Print the names of all Swiss series that have been published in magazines.

For this query, we select the Series whose country_id is the id of « Switzerland », and whose publication_type_id is the id of « magazine »

*SQL statement:*

```
SELECT S.name
FROM Series S, Country C, Series_Publication_Type T
WHERE C.name = "Switzerland" AND
        C.id = S.country_id AND
        T.name = "magazine" AND
        S.publication_type_id = T.id;
```

*Result:*

5 lines in 2.3ms

## Query d:

*Description of logic:*

Starting from 1990, print the number of issues published each year.

For this query, we select all the issues whose publication_date is greater than 1990 and we group them by the publication_date. Finally , we select every publication date and the number of Issue for each publication_date.

*SQL statement:*

```
SELECT I.publication_date, COUNT(I.id)
FROM Issue I
WHERE I.publication_date >= 1990
GROUP BY I.publication_date;
```

*Result:*

28 lines in 507ms

## Query e:

*Description of logic:*

Print the number of series for each indicia publisher whose name resembles 'DC comics'.

For this query, we take the Issues whose Indicia_publisher_id points to an Indicia Publisher whose name is like 'DC comics'. Then, we group this issues by their indicia_publisher_id. And finally we select the number of Issue for each indicia_publisher_id.

*SQL statement:*

```
SELECT I.indicia_publisher_id, IndiPubli.name, COUNT(DISTINCT(I.series_id))
FROM Issue I, Indicia_Publisher IndiPubli
WHERE I.indicia_publisher_id = IndiPubli.id AND IndiPubli.name LIKE 'DC comics'
GROUP BY I.indicia_publisher_id;
```

*Result:*
4 lines in 104ms

## Query f:

*Description of logic:*
Print the titles of the 10 most reprinted stories.

In an intermediate table, we select the number of story_reprint for each story (that has been reprinted). To achieve that, we group story_reprint by their origin_id and we count the number of reprint for each origin_id. Then, we order this table by the number of reprint.
Finally, we select the title and the number of reprint of the 10 stories that have the highest number of reprint.

*SQL statement:*

```
SELECT S.title, count FROM
(SELECT SR.origin_id, COUNT(*) AS count
 FROM Story_Reprint SR
 GROUP BY SR.origin_id
 ORDER BY COUNT(*) DESC) S1
JOIN Story S
ON S.id = S1.origin_id
WHERE S.title IS NOT NULL
LIMIT 10;
```

*Result:*
10 lines in 285ms

## Query g:

*Description of logic:*

Print the artists that have scripted, drawn, and colored at least one of the stories they were involved in.

We select the name of the artists which are in the list containing the artists that have scripted, drawn and colored a same story.

*SQL statement:*

```
SELECT name
FROM Story_Artists A
WHERE A.id IN (SELECT DISTINCT C.artist_id
               FROM Story_Has_Scripts H, Story_Has_Colors C, Story_Has_Pencils P
               WHERE H.story_id = C.story_id AND
                     C.story_id = P.story_id  AND
                     H.artist_id = C.artist_id AND
                     C.artist_id = P.artist_id);
```

*Result:*

4150 lines in 1.55s

## Query h:

*Description of logic:*

Print all non-reprinted stories involving Batman as a non-featured character.

We take the Story which has at least one Character named Batman. We keep only the story that have a non null title. Then, we check that this story is not in the set containing all the reprinted stories. Finally, we check that this story is not in the set containing all the stories which have Batman as feature.

*SQL statement:*

```
SELECT DISTINCT S.title
FROM Story_Characters SC, Story_Has_Characters SHC, Story S
WHERE SC.name = "Batman" AND
      SHC.character_id = SC.id AND
      S.id = SHC.story_id AND
      S.title IS NOT NULL AND
      S.id NOT IN (SELECT SR.origin_id FROM Story_Reprint SR) AND
      S.id NOT IN (SELECT DISTINCT SHF.story_id
FROM Story_Has_Features SHF, Story_Features SF
WHERE SF.name = "Batman" AND SHF.feature_id = SF.id);
```

*Result:*

3110 lines in 232ms

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne

URL: http://dias.epfl.ch/

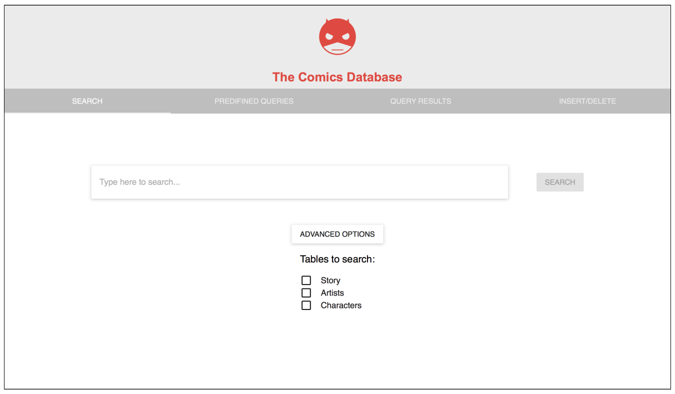
ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

## *Interface*

### Design logic Description

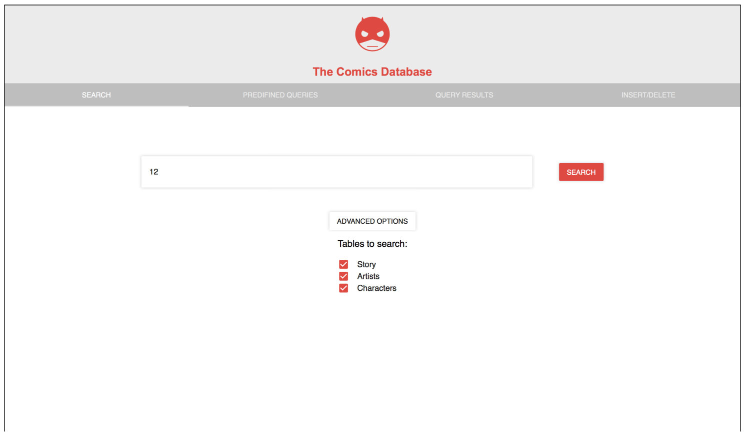
We decided to use React JS for the web app, and php for the server.
For now, we can only search by id. When the user enter an id in the search bar, the web app send the request to the php server, which establishes a connection to the mysql database and makes the SQL request for the story, artist and character identified by the given id. For now, we can only search in the following tables: Story, Story_Artists, Story_Characters.

### Screenshots



Tables to search:

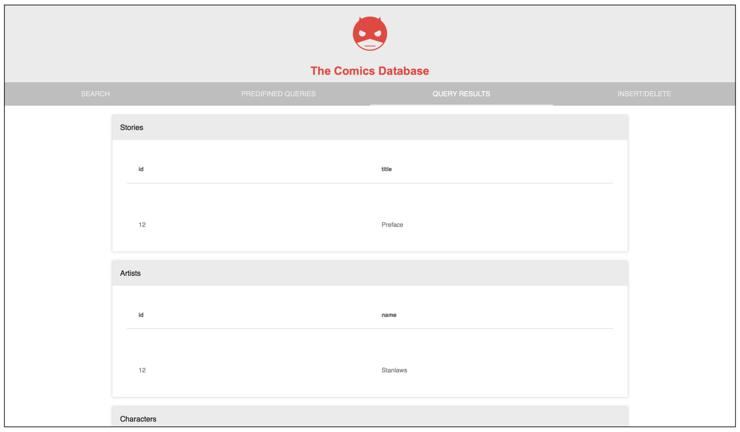
- ☐ Story
- ☐ Artists
- ☐ Characters



12

Tables to search:

- ☑ Story
- ☑ Artists
- ☑ Characters



Stories

| id | title |
|---|---|
| 12 | Preface |

Artists

| id | name |
|---|---|
| 12 | Stanlaws |

Characters

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne

URL: http://dias.epfl.ch/

## *General Comments*

We split the work in three equal parts, containing the set up of the mysql database, the cleaning of the data, the beginning interface and the php server.

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne

URL: http://dias.epfl.ch/

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

# Deliverable 3

## Assumptions

<In this section write down the assumptions you made about the data. Write a sentence for each assumption you made>

## *Query Implementation*

<For each query>

### Query a:

#### *Description of logic:*

<What does the query do and how do I decide to solve it>

#### *SQL statement*

<The SQL statement>

## *Query Analysis*

### Selected Queries (and why)

#### *Query 1*

<Initial Running time:

Optimized Running time:

Explain the improvement:

Initial plan

Improved plan>

#### *Query 2*

<Initial Running time:

Optimized Running time:

Explain the improvement:

Initial plan

Improved plan>

### *Query 3*

<Initial Running time:

Optimized Running time:

Explain the improvement:

Initial plan

Improved plan>

## Interface

### Design logic Description

<Describe the general logic of your design as well as the technology you decided to use>

### Screenshots

<Provide some initial screen shots of your interface>

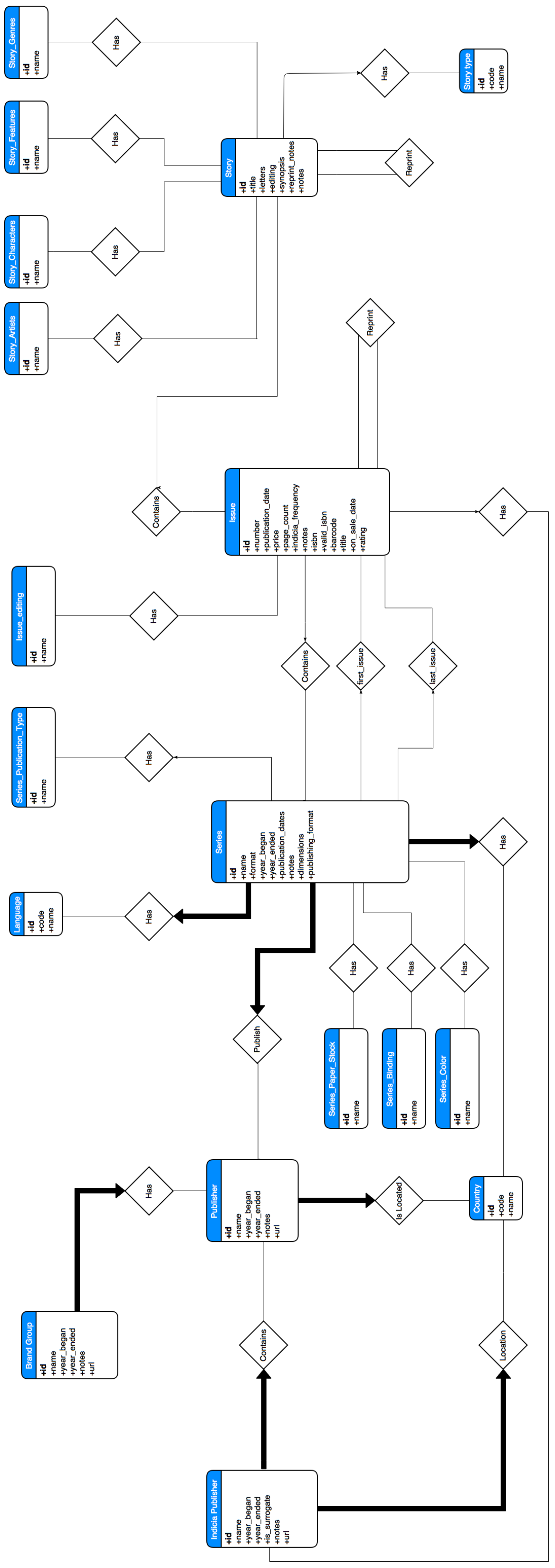
## General Comments

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

URL: http://dias.epfl.ch/

## Annex1



**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne

URL: http://dias.epfl.ch/

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

## Annex2

```
CREATE TABLE Story_Type (
        id INT NOT NULL,
        name VARCHAR(40),
        PRIMARY KEY (id)
);

CREATE TABLE Country (
        id INT NOT NULL,
        code VARCHAR(4),
        name VARCHAR(40),
        PRIMARY KEY (id)
);

CREATE TABLE Series_Publication_Type (
        id INT NOT NULL,
        name VARCHAR(16),
        PRIMARY KEY (id)
);
```

```
CREATE TABLE Language (
        id INT NOT NULL,
        code VARCHAR(4),
        name VARCHAR(40),
        PRIMARY KEY (id)
);

CREATE TABLE Publisher (
        id INT NOT NULL,
        name VARCHAR(128),
        country_id INT NOT NULL,
        year_began INT,
        year_ended INT,
        notes TEXT,
        url VARCHAR(256),
        PRIMARY KEY (id),
        FOREIGN KEY (country_id) REFERENCES Country(id)
);

CREATE TABLE Brand_Group (
        id INT NOT NULL,
        name VARCHAR(128),
        year_began INT,
        year_ended INT,
        notes TEXT,
        url VARCHAR(256),
        publisher_id INT NOT NULL,
        PRIMARY KEY (id),
        FOREIGN KEY (publisher_id) REFERENCES Publisher(id)
);

CREATE TABLE Indicia_Publisher (
        id INT NOT NULL,
        name VARCHAR(128),
        publisher_id INT NOT NULL,
        country_id INT NOT NULL,
        year_began INT,
        year_ended INT,
        is_surrogate INT,
        notes TEXT,
        url VARCHAR(256),
        PRIMARY KEY (id),
        FOREIGN KEY (publisher_id) REFERENCES Publisher(id),
        FOREIGN KEY (country_id) REFERENCES Country(id)
);

CREATE TABLE Issue (
        id INT NOT NULL,
	number VARCHAR(64),
	series_id INT,
	indicia_publisher_id INT,
	publication_date INT,
	price VARCHAR(16),
	page_count INT,
	indicia_frequency VARCHAR(160),
	notes TEXT,
	isbn VARCHAR(64),
	valid_isbn VARCHAR(64),
	barcode VARCHAR(64),
	title VARCHAR(128),
	on_sale_date INT,
	rating VARCHAR(128),
	PRIMARY KEY (id),
/*	FOREIGN KEY (series_id) REFERENCES Series(id), */
	FOREIGN KEY (indicia_publisher_id) REFERENCES Indicia_Publisher(id)
);


CREATE TABLE Issue_Editing (
	id INT NOT NULL,
	name VARCHAR(32),
	PRIMARY KEY (id)
```

```
);

CREATE TABLE Issue_Has_Editing (
        issue_id INT NOT NULL,
        editing_id INT NOT NULL,
        PRIMARY KEY (issue_id, editing_id),
        FOREIGN KEY (issue_id) REFERENCES Issue(id),
        FOREIGN KEY (editing_id) REFERENCES Issue_Editing(id)
);

CREATE TABLE Story (
        id INT NOT NULL,
        title VARCHAR(512),
        issue_id INT,
        letters TEXT,
        editing VARCHAR(512),
        synopsis TEXT,
        reprint_notes TEXT,
        notes TEXT,
        type_id INT,
        PRIMARY KEY (id),
        FOREIGN KEY (issue_id) REFERENCES Issue(id),
        FOREIGN KEY (type_id) REFERENCES Story_Type(id)
);
```

```
CREATE TABLE Series (
        id INT NOT NULL,
        name VARCHAR(256),
        format VARCHAR(256),
        year_began INT,
        year_ended INT,
        publication_dates INT,
        first_issue_id INT,
        last_issue_id INT,
        publisher_id INT NOT NULL,
        country_id INT NOT NULL,
        language_id INT NOT NULL,
        notes TEXT,
        dimensions VARCHAR(256),
        publishing_format VARCHAR(128),
        publication_type_id INT,
        PRIMARY KEY (id),
        FOREIGN KEY (first_issue_id) REFERENCES Issue(id),
        FOREIGN KEY (last_issue_id) REFERENCES Issue(id),
        FOREIGN KEY (publisher_id) REFERENCES Publisher(id),
        FOREIGN KEY (language_id) REFERENCES Language(id),
        FOREIGN KEY (country_id) REFERENCES Country(id),
        FOREIGN KEY (publication_type_id) REFERENCES Series_Publication_Type(id)
);

ALTER TABLE Issue ADD CONSTRAINT series_id FOREIGN KEY (series_id) REFERENCES Series(id);

CREATE TABLE Serie_Binding (
        id INT NOT NULL,
        name VARCHAR(32),
        PRIMARY KEY (id)
);

CREATE TABLE Serie_Has_Binding (
        series_id INT NOT NULL,
        binding_id INT NOT NULL,
        PRIMARY KEY (series_id, binding_id),
        FOREIGN KEY (series_id) REFERENCES Series(id),
        FOREIGN KEY (binding_id) REFERENCES Serie_Binding(id)
);

CREATE TABLE Serie_Colors (
        id INT NOT NULL,
        name VARCHAR(64),
        PRIMARY KEY (id)
);
```

```
CREATE TABLE Serie_Has_Colors (
        series_id INT NOT NULL,
        color_id INT NOT NULL,
        PRIMARY KEY (series_id, color_id),
        FOREIGN KEY (series_id) REFERENCES Series(id),
        FOREIGN KEY (color_id) REFERENCES Serie_Colors(id)
);

CREATE TABLE Series_Paper_Stock (
        id INT NOT NULL,
        name VARCHAR(32),
        PRIMARY KEY (id)
);

CREATE TABLE Series_Has_Paper_Stock (
        series_id INT NOT NULL,
        paper_stock_id INT NOT NULL,
        PRIMARY KEY (series_id, paper_stock_id),
        FOREIGN KEY (series_id) REFERENCES Series(id),
        FOREIGN KEY (paper_stock_id) REFERENCES Series_Paper_Stock(id)
);
```

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne

URL: http://dias.epfl.ch/

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

```
CREATE TABLE Story_Features (
        id INT NOT NULL,
        name VARCHAR(64),
        PRIMARY KEY (id)
);

CREATE TABLE Story_Has_Features (
        story_id INT NOT NULL,
        feature_id INT NOT NULL,
        PRIMARY KEY (story_id, feature_id),
        FOREIGN KEY (story_id) REFERENCES Story(id),
        FOREIGN KEY (feature_id) REFERENCES Story_Features(id)
);

CREATE TABLE Story_Artists (
        id INT NOT NULL,
        name VARCHAR(64),
        PRIMARY KEY(id)
);

CREATE TABLE Story_Has_Inks (
        story_id INT NOT NULL,
        artist_id INT NOT NULL,
        PRIMARY KEY (story_id, artist_id),
        FOREIGN KEY(story_id) REFERENCES Story(id),
        FOREIGN KEY(artist_id) REFERENCES Story_Artists(id)
);

CREATE TABLE Story_Has_Colors (
        story_id INT NOT NULL,
        artist_id INT NOT NULL,
        PRIMARY KEY (story_id, artist_id),
        FOREIGN KEY(story_id) REFERENCES Story(id),
        FOREIGN KEY(artist_id) REFERENCES Story_Artists(id)
);

CREATE TABLE Story_Has_Pencils (
        story_id INT NOT NULL,
        artist_id INT NOT NULL,
        PRIMARY KEY (story_id, artist_id),
        FOREIGN KEY(story_id) REFERENCES Story(id),
        FOREIGN KEY(artist_id) REFERENCES Story_Artists(id)
);

CREATE TABLE Story_Has_Scripts (
        story_id INT NOT NULL,
        artist_id INT NOT NULL,
        PRIMARY KEY (story_id, artist_id),
        FOREIGN KEY(story_id) REFERENCES Story(id),
        FOREIGN KEY(artist_id) REFERENCES Story_Artists(id)
);

CREATE TABLE Story_Genres (
        id INT NOT NULL,
        name VARCHAR(32),
        PRIMARY KEY(id)
);

CREATE TABLE Story_Has_Genres (
        story_id INT NOT NULL,
        genre_id INT NOT NULL,
        PRIMARY KEY(story_id, genre_id),
        FOREIGN KEY(story_id) REFERENCES Story(id),
        FOREIGN KEY(genre_id) REFERENCES Story_Genres(id)
);

CREATE TABLE Story_Characters (
        id INT NOT NULL,
        name VARCHAR(2048),
        PRIMARY KEY(id)
);
```

```
CREATE TABLE Story_Has_Characters (
        story_id INT NOT NULL,
        character_id INT NOT NULL,
        PRIMARY KEY(story_id, character_id),
        FOREIGN KEY(story_id) REFERENCES Story(id),
        FOREIGN KEY(character_id) REFERENCES Story_Characters(id)
);

CREATE TABLE Story_Reprint (
        origin_id INT NOT NULL,
        target_id INT NOT NULL,
        PRIMARY KEY (origin_id, target_id),
        FOREIGN KEY (origin_id) REFERENCES Story(id),
        FOREIGN KEY (target_id) REFERENCES Story(id)
);

CREATE TABLE Issue_Reprint (
        origin_id INT NOT NULL,
        target_id INT NOT NULL,
        PRIMARY KEY (origin_id, target_id),
        FOREIGN KEY (origin_id) REFERENCES Story(id),
        FOREIGN KEY (target_id) REFERENCES Story(id)
);
```