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: **14**

Names: Perrotta Lucie, Phan Hoang Kim Lan, Nguyen Tim

Contents

1. Deliverable 1

Assumptions

The assumptions we made were that we are supposed to delete and to create some tables from the data to optimize the efficiency and the size of our DB.
Here are our assumptions:

- If one person has written thousands of books, it would be a loss of space to copy their name in every issue whey worked in. Instead, we would create a table “artist” with this person inside, and just this person’s id to all the books they have written
- The table “genre” that we have created is not essential but we did this for some efficiency purpose. This is faster to get our genres in a table as an attribute.

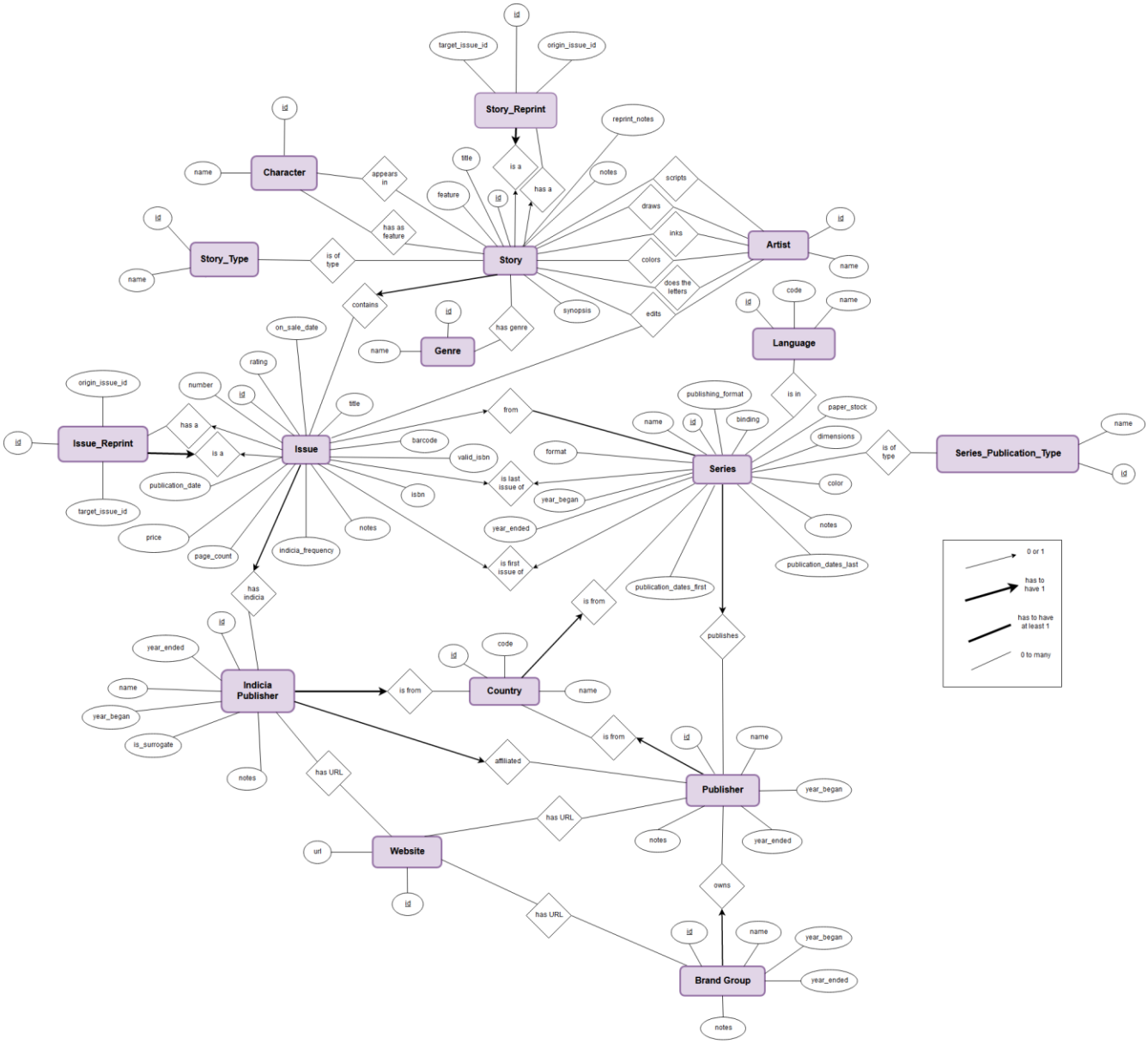
Entity Relationship Schema

Description

We translated the given dataset description into entities. We add some entities to better sort data :

- **Artist** : id, name which describe the different persons for a story
- **Genre** : id, name which describe the different genres of stories
- **Website** : id, url which describe the different URLs of the indicia publishers, the publishers and the brand groups
- **Character** : id, name which describe the different characters in a story

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/



We then linked the “artist” table to the story table through multiples relations :

- *scripts* : describing the story author(s)
- *draws* : describing the artist(s) who did the drawing
- *inks* : describing the artist(s) who did the inking
- *colors* : describing the artist(s) who added color to non-colored artwork
- *does the letters* : describing the creator(s) or studio(s) that did the lettering/typesetting
- *edits:* describing the story editor(s)

We also created an entity “character” and linked it to the story entity with the “has as feature” and “appears in” relations, since some famous characters are featured in many stories (superheroes typically).

Similarly, we created an entity Genre, since a story may have many genre, common with other series.

We created an entity “website” as the brands/incidia/publisher might share common websites.

Relational Schema

ER schema to Relational schema

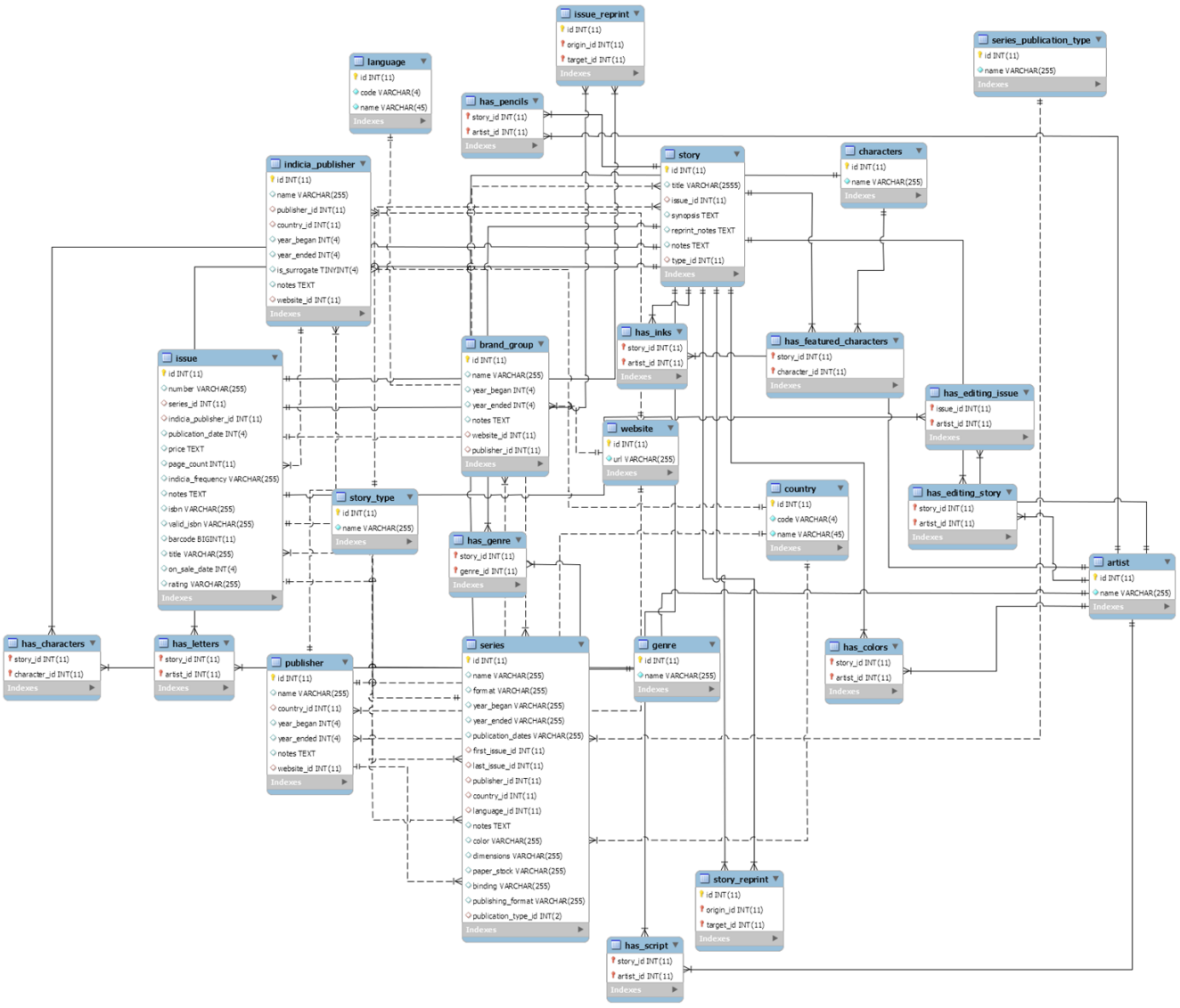
The translation straightly follows a certain logic. For each entity in the ER model we create a table. The arrow types have been translated in relations tables and foreign keys.

- For each relation 1 to 1 between entity A and entity B, we directly replaced the B’s name in A’s attributes by an id B_id, and set it as a foreign key pointing on B’s id itself.

- For each relation from 1 to n, we created an intermediate table “has_...” containing relations. A relation is composed of both ids of the two entities connected through this relation, and the id of the relation. This relation table contains then 2 foreign keys, pointing on both entities related. We hence have 7 “has tables”, for connecting for instance artists with the story they’ve been working on, or also for connecting the many prices an issue could have.

We then deleted all the “artist fields, cost field, etc…”, i.e. “inks, pencils etc.” since entities are directly connected through the relation.

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/



DDL

```
-- -----------------------------------------------------
-- Table `mydb`.`artist`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`artist` (
  `id` INT(11) NOT NULL,
  `name` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `mydb`.`country`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`country` (
  `id` INT(11) NOT NULL,
  `code` VARCHAR(4) NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `mydb`.`website`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`website` (
  `id` INT(11) NOT NULL,
  `url` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `mydb`.`publisher`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`publisher` (
  `id` INT(11) NOT NULL,
  `name` VARCHAR(255) NULL DEFAULT NULL,
  `country_id` INT(11) NULL DEFAULT NULL,
  `year_began` INT(4) NULL DEFAULT NULL,
  `year_ended` INT(4) NULL DEFAULT NULL,
  `notes` TEXT NULL DEFAULT NULL,
  `website_id` INT(11) NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `country_id_idx` (`country_id` ASC),
  INDEX `website_id` (`website_id` ASC),
  CONSTRAINT `country_id_publisher`
    FOREIGN KEY (`country_id`)
    REFERENCES `mydb`.`country` (`id`)
    ON DELETE SET NULL
    ON UPDATE NO ACTION,
  CONSTRAINT `website_id_publisher`
    FOREIGN KEY (`website_id`)
    REFERENCES `mydb`.`website` (`id`)
    ON DELETE SET NULL
    ON UPDATE NO ACTION)
ENGINE = InnoDB
```

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

```
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `mydb`.`brand_group`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`brand_group` (
  `id` INT(11) NOT NULL,
  `name` VARCHAR(255) NULL DEFAULT NULL,
  `year_began` INT(4) NULL DEFAULT NULL,
  `year_ended` INT(4) NULL DEFAULT NULL,
  `notes` TEXT NULL DEFAULT NULL,
  `website_id` INT(11) NULL DEFAULT NULL,
  `publisher_id` INT(11) NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `publisher_id_idx` (`publisher_id` ASC),
  INDEX `website_id` (`website_id` ASC),
  CONSTRAINT `publisher_id_brand`
    FOREIGN KEY (`publisher_id`)
    REFERENCES `mydb`.`publisher` (`id`)
    ON DELETE SET NULL
    ON UPDATE NO ACTION,
  CONSTRAINT `website_id_brand`
    FOREIGN KEY (`website_id`)
    REFERENCES `mydb`.`website` (`id`)
    ON DELETE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `mydb`.`characters`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`characters` (
  `id` INT(11) NOT NULL,
  `name` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `mydb`.`genre`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`genre` (
  `id` INT(11) NOT NULL,
  `name` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `mydb`.`indicia_publisher`
-- -----------------------------------------------------
```

```sql
CREATE TABLE IF NOT EXISTS `mydb`.`indicia_publisher` (
  `id` INT(11) NOT NULL,
  `name` VARCHAR(255) NULL DEFAULT NULL,
  `publisher_id` INT(11) NULL DEFAULT NULL,
  `country_id` INT(11) NULL DEFAULT NULL,
  `year_began` INT(4) NULL DEFAULT NULL,
  `year_ended` INT(4) NULL DEFAULT NULL,
  `is_surrogate` TINYINT(4) NULL DEFAULT NULL,
  `notes` TEXT NULL DEFAULT NULL,
  `website_id` INT(11) NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `publisher_id_idx` (`publisher_id` ASC),
  INDEX `country_id_idx` (`country_id` ASC),
  INDEX `website_id` (`website_id` ASC),
  CONSTRAINT `country_id_indicia`
    FOREIGN KEY (`country_id`)
    REFERENCES `mydb`.`country` (`id`)
    ON DELETE SET NULL
    ON UPDATE NO ACTION,
  CONSTRAINT `publisher_id_indicia`
    FOREIGN KEY (`publisher_id`)
    REFERENCES `mydb`.`publisher` (`id`)
    ON DELETE SET NULL
    ON UPDATE NO ACTION,
  CONSTRAINT `website_id_indicia`
    FOREIGN KEY (`website_id`)
    REFERENCES `mydb`.`website` (`id`)
    ON DELETE SET NULL
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `mydb`.`language`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`language` (
  `id` INT(11) NOT NULL,
  `code` VARCHAR(4) NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `mydb`.`series_publication_type`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`series_publication_type` (
  `id` INT(11) NOT NULL,
  `name` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `mydb`.`series`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`series` (
  `id` INT(11) NOT NULL,
```

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

```
  `name` VARCHAR(255) NULL DEFAULT NULL,
  `format` VARCHAR(255) NULL DEFAULT NULL,
  `year_began` VARCHAR(255) NULL DEFAULT NULL,
  `year_ended` VARCHAR(255) NULL DEFAULT NULL,
  `publication_dates` VARCHAR(255) NULL DEFAULT NULL,
  `first_issue_id` INT(11) NULL DEFAULT NULL,
  `last_issue_id` INT(11) NULL DEFAULT NULL,
  `publisher_id` INT(11) NULL DEFAULT NULL,
  `country_id` INT(11) NULL DEFAULT NULL,
  `language_id` INT(11) NULL DEFAULT NULL,
  `notes` TEXT NULL DEFAULT NULL,
  `color` VARCHAR(255) NULL DEFAULT NULL,
  `dimensions` VARCHAR(255) NULL DEFAULT NULL,
  `paper_stock` VARCHAR(255) NULL DEFAULT NULL,
  `binding` VARCHAR(255) NULL DEFAULT NULL,
  `publishing_format` VARCHAR(255) NULL DEFAULT NULL,
  `publication_type_id` INT(2) NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `first_issue_id_idx` (`first_issue_id` ASC),
  INDEX `last_issue_id_idx` (`last_issue_id` ASC),
  INDEX `publisher_id_idx` (`publisher_id` ASC),
  INDEX `country_id_idx` (`country_id` ASC),
  INDEX `language_id_idx` (`language_id` ASC),
  INDEX `publication_type_id` (`publication_type_id` ASC),
  CONSTRAINT `country_id_series`
    FOREIGN KEY (`country_id`)
    REFERENCES `mydb`.`country` (`id`)
    ON DELETE SET NULL
    ON UPDATE NO ACTION,
  CONSTRAINT `first_issue_id_series`
    FOREIGN KEY (`first_issue_id`)
    REFERENCES `mydb`.`issue` (`id`)
    ON DELETE SET NULL
    ON UPDATE NO ACTION,
  CONSTRAINT `language_id_series`
    FOREIGN KEY (`language_id`)
    REFERENCES `mydb`.`language` (`id`)
    ON DELETE SET NULL
    ON UPDATE NO ACTION,
  CONSTRAINT `last_issue_id_series`
    FOREIGN KEY (`last_issue_id`)
    REFERENCES `mydb`.`issue` (`id`)
    ON DELETE SET NULL
    ON UPDATE NO ACTION,
  CONSTRAINT `publication_type_id_series`
    FOREIGN KEY (`publication_type_id`)
    REFERENCES `mydb`.`series_publication_type` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `publisher_id_series`
    FOREIGN KEY (`publisher_id`)
    REFERENCES `mydb`.`publisher` (`id`)
    ON DELETE SET NULL
    ON UPDATE NO ACTION)
ENGINE = InnoDB
```

```
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `mydb`.`issue`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`issue` (
  `id` INT(11) NOT NULL,
  `number` VARCHAR(255) NULL DEFAULT NULL,
  `series_id` INT(11) NULL DEFAULT NULL,
  `indicia_publisher_id` INT(11) NULL DEFAULT NULL,
  `publication_date` INT(4) NULL DEFAULT NULL,
  `price` TEXT NULL DEFAULT NULL,
  `page_count` INT(11) NULL DEFAULT NULL,
  `indicia_frequency` VARCHAR(255) NULL DEFAULT NULL,
  `notes` TEXT NULL DEFAULT NULL,
  `isbn` VARCHAR(255) NULL DEFAULT NULL,
  `valid_isbn` VARCHAR(255) NULL DEFAULT NULL,
  `barcode` BIGINT(11) NULL DEFAULT NULL,
  `title` VARCHAR(255) NULL DEFAULT NULL,
  `on_sale_date` INT(4) NULL DEFAULT NULL,
  `rating` VARCHAR(255) NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `indicia_publisher_id` (`indicia_publisher_id` ASC),
  INDEX `series_id` (`series_id` ASC),
  CONSTRAINT `indicia_publishier_id_issue`
    FOREIGN KEY (`indicia_publisher_id`)
    REFERENCES `mydb`.`indicia_publisher` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `series_id_issue`
    FOREIGN KEY (`series_id`)
    REFERENCES `mydb`.`series` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `mydb`.`story_type`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`story_type` (
  `id` INT(11) NOT NULL,
  `name` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `mydb`.`story`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`story` (
  `id` INT(11) NOT NULL,
  `title` VARCHAR(2555) NULL DEFAULT NULL,
  `issue_id` INT(11) NULL DEFAULT NULL,
  `synopsis` TEXT NULL DEFAULT NULL,
  `reprint_notes` TEXT NULL DEFAULT NULL,
  `notes` TEXT NULL DEFAULT NULL,
  `type_id` INT(11) NULL DEFAULT NULL,
```

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

```
  PRIMARY KEY (`id`),
  INDEX `issue_id_idx` (`issue_id` ASC),
  INDEX `type_id_idx` (`type_id` ASC),
  CONSTRAINT `issue_id_story`
    FOREIGN KEY (`issue_id`)
    REFERENCES `mydb`.`issue` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `type_id_story`
    FOREIGN KEY (`type_id`)
    REFERENCES `mydb`.`story_type` (`id`)
    ON DELETE SET NULL
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `mydb`.`has_characters`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`has_characters` (
  `story_id` INT(11) NOT NULL,
  `character_id` INT(11) NOT NULL,
  PRIMARY KEY USING BTREE (`story_id`, `character_id`),
  INDEX `character_id_idx` (`character_id` ASC),
  INDEX `story_id_idx` (`story_id` ASC),
  CONSTRAINT `character_id_characters`
    FOREIGN KEY (`character_id`)
    REFERENCES `mydb`.`characters` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT `story_id_characters`
    FOREIGN KEY (`story_id`)
    REFERENCES `mydb`.`story` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `mydb`.`has_colors`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`has_colors` (
  `story_id` INT(11) NOT NULL,
  `artist_id` INT(11) NOT NULL,
  PRIMARY KEY (`story_id`, `artist_id`),
  INDEX `story_id_idx` (`story_id` ASC),
  INDEX `artist_id_idx` (`artist_id` ASC),
  CONSTRAINT `artist_id_colors`
    FOREIGN KEY (`artist_id`)
    REFERENCES `mydb`.`artist` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT `story_id_colors`
```

```
    FOREIGN KEY (`story_id`)
    REFERENCES `mydb`.`story` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `mydb`.`has_editing_issue`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`has_editing_issue` (
  `issue_id` INT(11) NOT NULL,
  `artist_id` INT(11) NOT NULL,
  PRIMARY KEY (`issue_id`, `artist_id`),
  INDEX `artist_id_editing_issue` (`artist_id` ASC),
  CONSTRAINT `artist_id_editing_issue`
    FOREIGN KEY (`artist_id`)
    REFERENCES `mydb`.`artist` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT `issue_id_editing_issue`
    FOREIGN KEY (`issue_id`)
    REFERENCES `mydb`.`issue` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;


-- -----------------------------------------------------
-- Table `mydb`.`has_editing_story`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`has_editing_story` (
  `story_id` INT(11) NOT NULL,
  `artist_id` INT(11) NOT NULL,
  PRIMARY KEY (`story_id`, `artist_id`),
  INDEX `artist_id_editing` (`artist_id` ASC),
  CONSTRAINT `artist_id_editing`
    FOREIGN KEY (`artist_id`)
    REFERENCES `mydb`.`artist` (`id`)
    ON DELETE CASCADE,
  CONSTRAINT `story_id_editing`
    FOREIGN KEY (`story_id`)
    REFERENCES `mydb`.`story` (`id`)
    ON DELETE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;


-- -----------------------------------------------------
-- Table `mydb`.`has_featured_characters`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`has_featured_characters` (
  `story_id` INT(11) NOT NULL,
  `character_id` INT(11) NOT NULL,
  PRIMARY KEY USING BTREE (`story_id`, `character_id`),
  INDEX `character_id_feat` (`character_id` ASC),
  CONSTRAINT `character_id_feat`
    FOREIGN KEY (`character_id`)
    REFERENCES `mydb`.`characters` (`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

```
    ON DELETE CASCADE,
  CONSTRAINT `story_id_feat`
    FOREIGN KEY (`story_id`)
    REFERENCES `mydb`.`story` (`id`)
    ON DELETE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;


-- -----------------------------------------------------
-- Table `mydb`.`has_genre`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`has_genre` (
  `story_id` INT(11) NOT NULL,
  `genre_id` INT(11) NOT NULL,
  PRIMARY KEY (`genre_id`, `story_id`),
  INDEX `genre_id_idx` (`genre_id` ASC),
  INDEX `story_id_idx` (`story_id` ASC),
  CONSTRAINT `genre_id_genre`
    FOREIGN KEY (`genre_id`)
    REFERENCES `mydb`.`genre` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT `story_id_genre`
    FOREIGN KEY (`story_id`)
    REFERENCES `mydb`.`story` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `mydb`.`has_inks`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`has_inks` (
  `story_id` INT(11) NOT NULL,
  `artist_id` INT(11) NOT NULL,
  PRIMARY KEY (`story_id`, `artist_id`),
  INDEX `stpry_Id_idx` (`story_id` ASC),
  INDEX `artist_id_idx` (`artist_id` ASC),
  CONSTRAINT `artist_id_inks`
    FOREIGN KEY (`artist_id`)
    REFERENCES `mydb`.`artist` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT `story_id_inks`
    FOREIGN KEY (`story_id`)
    REFERENCES `mydb`.`story` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
```

```
-- -----------------------------------------------------
-- Table `mydb`.`has_letters`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`has_letters` (
  `story_id` INT(11) NOT NULL,
  `artist_id` INT(11) NOT NULL,
  PRIMARY KEY (`story_id`, `artist_id`),
  INDEX `artist_id_letters` (`artist_id` ASC),
  CONSTRAINT `artist_id_letters`
    FOREIGN KEY (`artist_id`)
    REFERENCES `mydb`.`artist` (`id`)
    ON DELETE CASCADE,
  CONSTRAINT `story_id_letters`
    FOREIGN KEY (`story_id`)
    REFERENCES `mydb`.`story` (`id`)
    ON DELETE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8
COLLATE = utf8_bin;


-- -----------------------------------------------------
-- Table `mydb`.`has_pencils`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`has_pencils` (
  `story_id` INT(11) NOT NULL,
  `artist_id` INT(11) NOT NULL,
  PRIMARY KEY (`story_id`, `artist_id`),
  INDEX `id_artist_idx` (`artist_id` ASC),
  INDEX `story_id_idx` (`story_id` ASC),
  CONSTRAINT `artist_id_pencils`
    FOREIGN KEY (`artist_id`)
    REFERENCES `mydb`.`artist` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT `story_id_pencils`
    FOREIGN KEY (`story_id`)
    REFERENCES `mydb`.`story` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `mydb`.`has_script`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`has_script` (
  `story_id` INT(11) NOT NULL,
  `artist_id` INT(11) NOT NULL,
  PRIMARY KEY (`story_id`, `artist_id`),
  INDEX `story_id_idx` (`story_id` ASC),
  INDEX `artist_id_idx` (`artist_id` ASC),
  CONSTRAINT `artist_id_script`
    FOREIGN KEY (`artist_id`)
    REFERENCES `mydb`.`artist` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT `story_id_script`
    FOREIGN KEY (`story_id`)
    REFERENCES `mydb`.`story` (`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

```
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `mydb`.`issue_reprint`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`issue_reprint` (
  `id` INT(11) NOT NULL,
  `origin_id` INT(11) NOT NULL,
  `target_id` INT(11) NOT NULL,
  PRIMARY KEY (`id`, `origin_id`, `target_id`),
  INDEX `origin_issue_id_idx` (`origin_id` ASC),
  INDEX `target_issue_id_idx` (`target_id` ASC),
  CONSTRAINT `origin_issue_id_issue_reprint`
    FOREIGN KEY (`origin_id`)
    REFERENCES `mydb`.`issue` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT `target_issue_id_issue_reprint`
    FOREIGN KEY (`target_id`)
    REFERENCES `mydb`.`issue` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `mydb`.`story_reprint`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`story_reprint` (
  `id` INT(11) NOT NULL,
  `origin_id` INT(11) NOT NULL,
  `target_id` INT(11) NOT NULL,
  PRIMARY KEY (`id`, `origin_id`, `target_id`),
  INDEX `origin_id_idx` (`origin_id` ASC),
  INDEX `target_id_idx` (`target_id` ASC),
  CONSTRAINT `origin_id_reprint`
    FOREIGN KEY (`origin_id`)
    REFERENCES `mydb`.`story` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT `target_id_reprint`
    FOREIGN KEY (`target_id`)
    REFERENCES `mydb`.`story` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
```

Deliverable 2

Design evolved a bit from milestone 1. Tables we added are has_featured_character(story_id, character_id) as well as has_editing(story_id, artist_id).

Parsing and cleaning of the data

We did a lot of cleaning (probably too much, which made the DB filling very slow). We choose to use a local database, using **wamp server** (windows local server) and **phpmyadmin**. Thus we are parsing the csv file using php. Once parsed, the resulting csv files are imported into phpmyadmin to complete our tables.
We had to parse multiple times the csv's given to us. Here modifications we had to apply on entries:

- **Null values:** Values such as "Null", "none", "[none]", "[nn]", "?", "(unknown)", "None", etc. are deleted and replace with NULL (parseNullValue function).
- **Names** are full of information between parenthesis or brackets (such as (signed), (translator)…). We delete them to be able to rely them (instead of having twice the same author). Also to make them match even more, we construct a comparative string, which is the name of author without spaces, dot and hyphen. These can permit us to avoid some duplicata due to syntax error.
- Dates are difficult to retrieve from given csv file, because of its non-uniform format (getDateFromYear function). We assume for example:
    - "1870's" become "1870"
    - "July 10 1870" become "1870"
    - "1870-07-10" become "1870"
- **Id:** For fields where integer is required, if we retrieve either an integer or replace it with a NULL value (getInt function)
- **Websites** come from publisher, indicia_publisher and brand_group csv file. We get the url values and add it in a new website table. We never add twice the same website. Once done, we process publisher, indicia_publisher and brand_group csv files and write into a new csv file their values plus their url values changed into website ids (which become a foreign key referencing the website table).
- **Artist, Character, Features character, Genre** : If a story or an issue has an artist (for script, letters, inks, editing…) we add in an artist csv file the found artist (id, artist name) and we also add in a has_ csv file the pairs (story_id, artist_id).
  The same idea is also applied to characters (features and characters from story) and genres.

While parsing, some columns lengths and types had to change. For example:

- isbn , rating, number of issue becoming varchar
- synospsis of story becoming text
- dates becoming integer
- some column lengths has to be bigger because of some special entries

The "*functions.php*" file, used in every php parsing files, gathers defined below parsing methods.

[functions.php]

```php
<?php

function parseNullValue($s) {

    if(empty($s) && $s!='0') return true;
```

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

```
    $nullValues = ['NULL', '[nn]', 'nn', 'none', '[none]','?', '(unknown)','None'];
    foreach($nullValues as $n) {
        if ($s === $n) {
            return true;
        }
    }
    return false;
}

function parseDoubleQuote($s) {

    if(parseNullValue($s)) return "NULL";

    $res = str_replace('"', '\"', $s);
    $res = str_replace('\\\\"', '\"', $res);
    return '"'.$res.'"';
}


function getDateFromYear($year) {

    if(parseNullValue($year)) return "NULL";

    $res = preg_replace("/[^\d\s-]/" , "", $year);
    $res1 = preg_split("/[\s-]/" , $res);

    for ($i = 0; $i < sizeof($res1); $i++) {
        if(strlen($res1[$i])==4) {
            return $res1[$i];
        }
    }

    // no date with 4 digits
    return "NULL";
}

function getInt($i) {

    if(parseNullValue($i)) return "NULL";

    // suppress [ ] char
    $i = preg_replace("/\[*\]*/" ,"", $i);

    return $i;
}

// return last index used in csv - useful for assigning id
function getLastIndex($file) {
    $index;
    while((!feof($file)) && ($val = fgetcsv($file))){
        $index = $val[0];
    }
    return (empty($index)) ? 0 : $index+1;
}
```

```php
// return words separated by delimiter
function parseNames($s, $delimiter=";"){
    // get rid of first and last double quotes
    $string = substr($s, 1, -1);
    $array = explode($delimiter, $string);
    for($i = 0; $i< sizeof($array); $i++){
        $array[$i] = ltrim($array[$i]);
    }
    return array_filter($array, function($value) { return $value !== ''; });
}

// return true if $s in contained in $csv file (opened) (at position $pos), false otherwise
function isInCsv($file, $s, $pos){

    rewind($file);

    if(empty($s)) {
        return false;
    }

    while(! feof($file)){
        $val = fgetcsv($file);
        if($val[$pos]==$s) {
            return $val[0];
        }
    }
    return false;
}

// return true if $s in contained in $csv file (opened) (at position $pos), false otherwise
function isInCsvName($file, $s, $pos){

    rewind($file);

    if(empty($s)) {
        return false;
    }

    $string = parseToCompare($s);

    while(! feof($file)){
        $content = fgetcsv($file);
        $val = parseToCompare($content[$pos]);
        if($val==$string) {
            return $content[0];
        }
    }
    return false;
}

// modify string so that it can match even with the following differences : whitespaces, dot, dash
(essentially for names)
// in csv -> will keep first occurence
function parseToCompare($s){
    $res = preg_replace("/\s/", "", $s);
    $res = preg_replace("/\-/", "", $res);
    $res = preg_replace("/\./", "", $res);
    $res = strtolower($res);
    return $res;
}
```

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/

```php
// delete from $s all content between () or [] or ?
function parseComments($s) {
    $res = preg_replace("/\[.*\]/", "", $s);
    $res = preg_replace("/\(.*\)/", "", $res);
    $res = preg_replace("/\?/", "", $res);

    if(empty(preg_replace("/\s/","",$res))){
        return "NULL";
    }

    $res = trim($res);
    return $res;
}
```

[website_tocsv.php]

```php
<?php
include("functions.php");

$files = array("comics/brand_group.csv" => 5 , "comics/indicia_publisher.csv" => 8, "comics/pub-
lisher.csv" => 6);
$csv = fopen("comics/website.csv", "a+"); // write into this sql to import

$index = getLastIndex($csv);


// get websites from all given files
foreach ($files as $f => $pos) {
  $file = fopen($f,"r");
  $val = fgetcsv($file); // avoid url column name

  while(! feof($file)){
    $val = fgetcsv($file);

      $url = $val[$pos];

      // if the websie
      if(!parseNullValue($url)){
        if(isInCsv($csv, $url,1)===false) {
          $add = $index . ",".$url."\n";
          fwrite($csv, $add);
          $index++;
        }
      }
    }
  }
  fclose($file);

}

fclose($csv);
?>
```

[website_to_id.php]

```php
<?php
include("functions.php");

$files = array("comics/brand_group.csv" => 5 , "comics/indicia_publisher.csv" => 8, "comics/pub-
lisher.csv" => 6);
$filesnames = array("comics/brand_group.csv" => "comics/brand_group_id.csv" , "comics/indicia_pub-
lisher.csv" => "comics/indicia_publisher_id.csv", "comics/publisher.csv" => "comics/pub-
lisher_id.csv");
$csv = fopen("comics/website.csv", "r"); // write into this sql to import

$index = getLastIndex($csv);

// get websites from all given files
foreach ($files as $f => $pos) {
  $file = fopen($f,"r");
  $out = implode(",",fgetcsv($file))."\n";

  while(! feof($file) && ($i<$max)){
    $val = fgetcsv($file);

      $url = $val[$pos];

      if(!parseNullValue($url)){
        $index = isInCsvName($csv, $url,1);
        $val[$pos] = $index;
      }
      $out .= implode(",",$val)."\n";

  }
  fclose($file);
  file_put_contents($filesnames[$f], $out);
}

fclose($csv);
?>
```

[genre_tocsv.php]

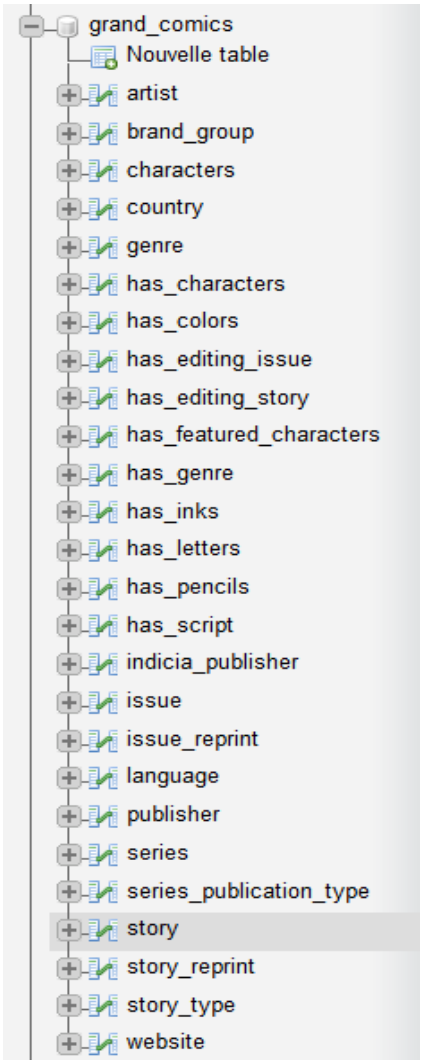
```php
<?php
include("functions.php");

$file = fopen("comics/story.csv","r");
$csv = fopen("comics/genre.csv", "a+");
$has_csv = fopen("comics/has_genre.csv", "a+");

$index = getLastIndex($csv);

fgetcsv($file);

while(! feof($file)){
  $val = fgetcsv($file);

    $id = getInt($val[0]);
    $genre = parseDoubleQuote($val[10]);

    if($genre!="NULL"){
      $genre_array = parseNames($genre);
      foreach ($genre_array as $p){
        $p = parseComments($p);
        $exist = isInCsvName($csv, $p,1);
        if(!is_numeric($exist)) {
```

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/

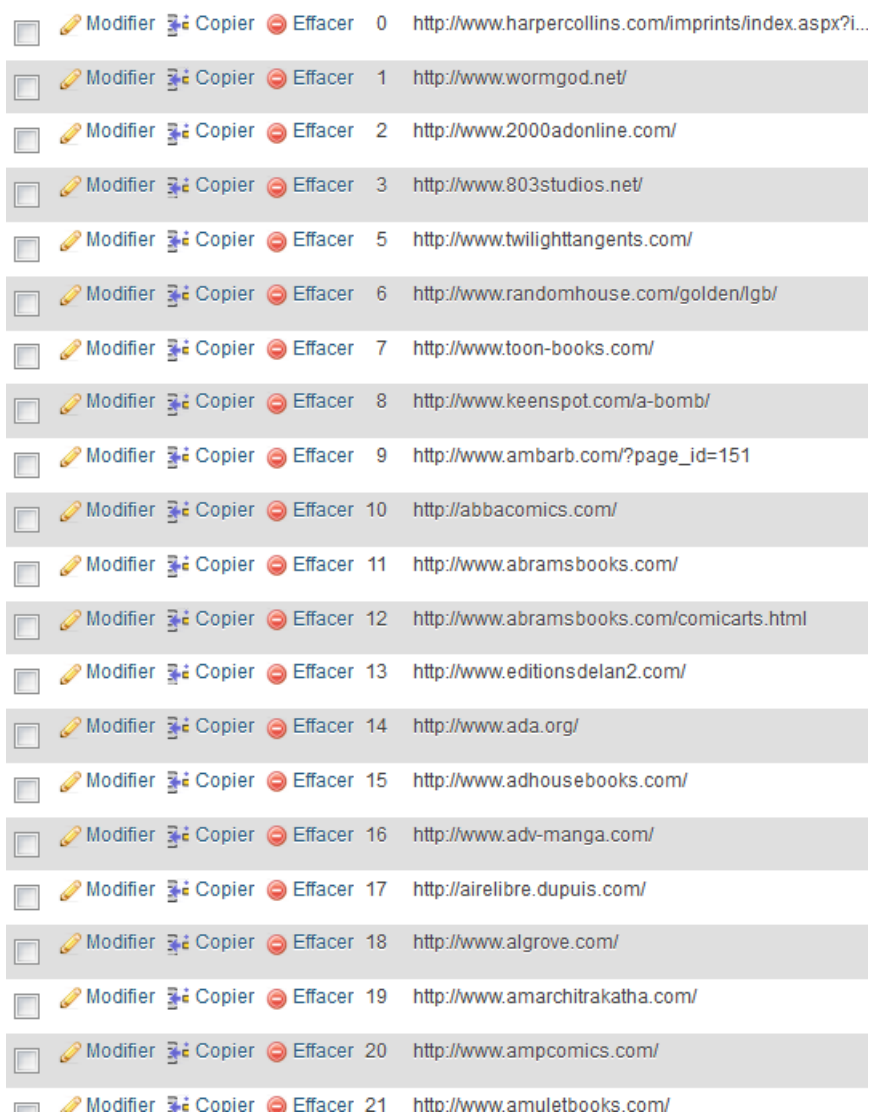
ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

```
          $add = $index . ",".$p."\n";
          fwrite($csv, $add);

          // has_
          $query = $id.",".$index ."\n";
          fwrite($csv, $add);

          $index++;
        }
        else {

          $query = $id.",".$exist ."\n";
          fwrite($has_csv, $query);
        }
      }
    }

}

fclose($file);
fclose($csv);

?>
```

Screenshots of the database, as seen from the phpMyAdmin interface



All tables



Website table

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

Modifier Copier Effacer	6	0
Modifier Copier Effacer	7	0
Modifier Copier Effacer	8	0
Modifier Copier Effacer	9	0
Modifier Copier Effacer	990	0
Modifier Copier Effacer	998	0
Modifier Copier Effacer	1011	0
Modifier Copier Effacer	1015	0
Modifier Copier Effacer	1022	0
Modifier Copier Effacer	1024	0
Modifier Copier Effacer	1048	0
Modifier Copier Effacer	1070	0
Modifier Copier Effacer	1105	0
Modifier Copier Effacer	1161	0
Modifier Copier Effacer	1174	0
Modifier Copier Effacer	1187	0
Modifier Copier Effacer	1215	0

Has_genre table

Modifier Copier Effacer	1	aa	Afar
Modifier Copier Effacer	2	ab	Abkhazian
Modifier Copier Effacer	3	af	Afrikaans
Modifier Copier Effacer	4	am	Amharic
Modifier Copier Effacer	5	ar	Arabic
Modifier Copier Effacer	6	as	Assamese
Modifier Copier Effacer	7	ay	Aymara
Modifier Copier Effacer	8	az	Azerbaijani
Modifier Copier Effacer	9	ba	Bashkir
Modifier Copier Effacer	10	be	Byelorussian
Modifier Copier Effacer	11	bg	Bulgarian
Modifier Copier Effacer	12	bh	Bihari
Modifier Copier Effacer	13	bi	Bislama
Modifier Copier Effacer	14	bn	Bengali; Bangl
Modifier Copier Effacer	15	bo	Tibetan

Language table

Query Implementation

Here are the 8 queries we were asked to implement in MySQL. Those are the last versions of our SQL queries (we didn't keep the old non-optimized queries).

PLEASE NOTE that regarding all the queries in milestone 2 and 3, the results may be incomplete. The reason is, we couldn't fill the DB fast enough in time because we made the choice to clean the data too much. For example retrieving every characters/artists, inserting them into a table and looking into the whole table to find a similar name each time took us a lot of time, and sadly we couldn't fill the DB with all the data. Most of the data has been added though. The results of the queries may be incomplete for that reason.

For this first query, we first select all the brand name possessing at least one indicia publisher from Belgium. We choose to get the name of the brand_group and the number of Belgian indicia it possesses. Then we simply sort the resulting table by number of indicia and get the name only. Note that we don't need to go through the Publisher table, which makes us gain some time.

```
-- a)
SELECT T.name
FROM    (
        SELECT  B.name,
                COUNT(*) AS bid
        FROM    brand_group B,
                indicia_publisher I,
                country C
        WHERE   C.name = 'Belgium' AND
                C.id = I.country_id AND
                I.publisher_id = B.publisher_id
        GROUP BY B.name
        ) AS T
ORDER BY T.bid
```

name
Schetter uitgever
Uitgeverij den Gulden Engel
InDruk
MM; Millennium
Seed
De Bezige Bij
Galatea
Novedi
Biloan
De Schaar

For the query b), we simply use the chain AND rule to get all publishers from Denmark (id 56), in a straightforward fashion.

```
-- b)
SELECT  P.id, P.name
FROM    publisher P,
        series S
WHERE   S.country_id = 56 AND
        S.publisher_id = P.id AND
        S.publication_type_id = 1
```

id	name
6330	Williams
5346	Egmont
7428	Forlaget Zoom
5363	Forlaget Carlsen
4551	Interpresse
6367	Aben Maler
6367	Aben Maler
6367	Aben Maler
6367	Aben Maler
6367	Aben Maler
6349	Bramsen & Hjort
8572	Forlaget Fabel

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

For querry c), the fashion is similar to b), we simply apply the chain rule, to get series from Switzerland (id 40) and published in a magazine.

```
-- c)
SELECT  S.name
FROM    series S,
        series_publication_type T
WHERE   T.name = 'magazine' AND
        T.id = S.publication_type_id AND
        S.country_id = 40
```

name
Micky Maus Zeitung
Tip Top
Junior
Melanie
Comixene

Here in d) we simply want to get all issues from 1990, sorted by year. Note that for simplicity purposes, all the dates have been converted to years stored as integers, since most of the dates simple consists as a year, often followed by text such as “circa”. It is hence easier to work with int(4) formatted years.

```
-- d)
SELECT  I.publication_date,
        COUNT(*)
FROM    issue I
WHERE   I.publication_date >= 1990
GROUP BY I.publication_date
```

publication_date	COUNT(*)
1990	4985
1991	4991
1992	5155
1993	5617
1994	5674
1995	5518
1996	5041
1997	4721
1998	4453
1999	4006
2000	3911

For the query e), we simply do a left join between the indicias and the series (to avoid going through the publisher table). Then we just ask for names resembling Dc comics.

```
-- e)
SELECT  I.name AS name,
        COUNT(I.id) AS nb
FROM    indicia_publisher I
        LEFT JOIN series S
        ON S.publisher_id =
          I.publisher_id
WHERE   I.name LIKE
'%DC_comics%'
GROUP BY I.name
```

name	nb
Dark Horse Comics Inc. and DC Comics	21
Dark Horse Comics Inc.; DC Comics	12
DC Comics	6503
DC Comics / Gareb Shamus Enterprises	2
DC Comics and Dark Horse Comics Inc.	21
DC Comics and Gareb Shamus Enterprises Inc. DBA Wizard Entertainment	4
DC Comics Inc.	6456
DC Comics; Top Cow Productions Inc.	6456
Marvel Comics Group and DC Comics Inc.	6657
Marvel Comics; DC Comics	31
Marvel Entertainment Group Inc. and DC Comics Inc.	6657

For query f), we first select all stories that have been reprinted at least once, and then regroup them by original story. Finally, we count how many times each original story has been reprinted and sort them according to that. We only print the names of the stories for aesthetical purposes.

```
-- f)
SELECT  S.title
FROM    story S,
        story_reprint R
WHERE   S.id = R.origin_id
GROUP BY R.origin_id
ORDER BY COUNT(R.origin_id)
```

title
The Legion of Super-Heroes
A Financial Fable
The Legion of Super-Villains!
The Million Dollar Debut of Batgirl!
The Super-Key to Fort Superman
The Man Behind the Red Hood!
The Strange Experiment of Dr. Erdel
1981 -- A Flash Odyssey
The Super-Duel in Space
The Case of the Chemical Syndicate

This query (g) was interesting since it uses the chain rule in a particular fashion. What we are seeking are artists who contributed to every part of the making of some story. That is, we want all artist who did color, write, draw and ink a story. We simply want an artist who did all 4 on a same story and a story who had all 4 done by a single artist.

```
-- g)
SELECT  distinct A.name
FROM    artist A,
        has_script SC,
        has_pencils P,
        has_colors C,
        has_inks I,
        story S
WHERE   A.id = SC.artist_id AND
        A.id = P.artist_id AND
        A.id = C.artist_id AND
        A.id = I.artist_id AND
        S.id = SC.story_id AND
        S.id = P.story_id AND
        S.id = C.story_id AND
        S.id = I.story_id
```

name
The Donaldson Brothers
Charles Nelan
R. F. Outcault
J.R. Hager
M. E. Brady
Bob McCay

For the last query h), we wanted Batman to be a non-featured character of a non-reprinted story. For that purpose, we seek all stories which were not reprinted, that is, which are not featured in the reprint table. We hence look for the story in the reprint table and expect the number of its occurrences to be 0.
Then, we simply say we want Batman to be in the non-featured characters. Note the utilization of the command LIKE in order to seek for all strings containing “Batman”.

```
-- h)
SELECT  distinct S.title
FROM    story S,
        characters C1,
        characters C2,
        has_characters HS,
        has_featured_characters HFC
WHERE   0 = (   SELECT COUNT(distinct R.origin_id)
                FROM story_reprint R
                WHERE S.id = R.origin_id
            ) AND
        HS.character_id = C1.id AND
        HS.story_id = S.id AND
        C1.name LIKE '%Batman%' AND
        HFC.character_id = C2.id AND
```

The Justice Society Raises $1000000 for War Orphans!
$1000000 for War Orphans! (conclusion)

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

```
        HFC.story_id = S.id AND
        C2.name NOT LIKE '%Batman%'
```

Deliverable 3

Modifications

ER model, DDL statements, relational scheme had been changed below. Deliverable 2's queries have been directly changed above.

Assumptions

Concerning the parsing, we really had to do strong cleaning. For example, all the dates have been converted to years in order to make the correct operations with the SQL queries. We also assumed that using lots of AND in queries was cleaner and easier to understand and read than multiples JOIN one into each other. We tested both, and the runtimes were pretty much the same. We hence have used almost no JOIN instructions in the queries. We also made the assumptions that we were allowed to use additional indexes on other columns, frequently used in the queries to improve performance.

Query Implementation

Query A

```
SELECT   S.name,
    T2.nbi
FROM   (
       SELECT   I.series_id,
           COUNT(*) AS nbi
      FROM   issue I,
            (
            SELECT distinct S.issue_id
            FROM   story S
            WHERE   S.type_id <>
                  (
                  SELECT   S.type_id
                  FROM   story S
                   GROUP BY S.type_id
                  ORDER BY COUNT(*) DESC LIMIT 1
                  )
              ) as T
        WHERE  I.id = T.issue_id
      GROUP BY I.series_id
      ) as T2,
      series S
WHERE   S.id = T2.series_id
ORDER BY T2.nbi DESC
```

This query is pretty slow as we check each of the conditions one by one in inner and inner queries.

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

name	nbi
Commando	3503
Donald Duck & Co	1881
Love Story Picture Library	1395
Kamp-serien	1332
Four Color	1249
Gespenster Geschichten	1146
Jukan	756
Detective Comics	722
Action Comics	702
Batman	677

Query B

```
SELECT  P.name
FROM    publisher P
WHERE   (SELECT COUNT(distinct SP.name)
         FROM    series_publication_type SP,
                 series S
         WHERE   SP.id = S.publication_type_id AND
                 S.publisher_id = P.id
         ) = 3
```

Here we simply check that for one publisher P, we can count 3 different publication_types, 3 being the total number of them. This hence is the same as checking if P has published all different types of media.

name
Casterman
DC
Marvel
Sergio Bonelli Editore
Atlas Publishing
Western
Classics/Williams
Åhlén & Åkerlunds
Hemmets Journal
Fantagraphics

Query C

```
SELECT C.name
FROM    (
        SELECT  HC.character_id,
                COUNT(*) as nch
        FROM    story_reprint SR,
                artist A,
                has_script HS,
                has_characters HC
        WHERE   HC.story_id = SR.origin_id AND
                HC.story_id = HS.story_id AND
                HS.artist_id = A.id AND
                A.name LIKE '%Alan_Moore%'
        GROUP BY HC.character_id
        ) as T,
        characters C
WHERE   C.id = T.character_id
ORDER BY T.nch DESC LIMIT 10
```

This query is straight-forward. We want the writer (script artist) to be Alan Moore, and get back to the characters table through our has_script and

name
Captain Angleterre
Captain Empire
Kommandant Englander
Lord Mandragon
CAMEOS: Merlyn
Meggan
King Arthur
Betsy Braddock
Black Knight
Many Captain Britain Corps Member

has_characters tables. We then list all the characters from Alan Moore, then count how many how them appear, and sort by the number of occurences.

Query D

```
SELECT  distinct A.name
FROM    artist A,
        has_script HS
WHERE   HS.artist_id = A.id AND
        (
        SELECT  COUNT(HS.artist_id)
        FROM    has_pencils HP,
                story S
        WHERE   HS.story_id = S.id AND
                (S.title LIKE '%natur%' OR
                S.synopsis LIKE '%natur%')
        ) = (
        SELECT  COUNT(HS.artist_id)
        FROM    has_pencils HP,
                story S
        WHERE   HP.artist_id = HS.artist_id AND
                HS.story_id = S.id AND
                HP.story_id = S.id AND
                (S.title LIKE '%natur%' OR
                S.synopsis LIKE '%natur%')
        )
```

name
Gustave Dor
Wilhelm Busch
Harry Rogers
The Donaldson Brothers
Richard Doyle
Abby Langdon Alger
Charles T. Brooks
J.C. Saxe
Henry Liddell
Charles Jay Taylor

Here we have to do 2 things. Check all the artists that have done nature related stores, and all the artists that have *drawn* nature stories. Those are the 2 queries that we want to be equal here. Note that we assumed that a nature related story is a story featuring something looking like “natur-“ in its title or synopsis. We then look for artists for which the number of nature stories they worked on is the same than the nature stories they drew.

Query E

```
SELECT  L.name,
        COUNT(*) as nb
FROM    series SE LEFT JOIN
        (
        SELECT  P.id,
                COUNT(P.id)
        FROM    series S,
                publisher P
        WHERE   P.id = S.publisher_id
        GROUP BY P.id DESC
        ORDER BY COUNT(P.id) DESC LIMIT 10
        ) as T ON T.id = SE.publisher_id,
        language L
WHERE   SE.language_id = L.id
GROUP BY L.id
ORDER BY nb DESC LIMIT 3
```

name	nb
English	57702
German	14209
Dutch	7223

The query was not really clear so we assumed we had to return the 3 most popular languages among the top-10 publisher all together. That is, if we kind of merge the top-10 publishers, what are the 3 most popular languages? So for the top-10 publishers we simply printed all the languages that they all published, all together, and then counted how many of each did occur. And finally we selected the 3 with the most occurrences.

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

Query F

```
SELECT  T.name,
        T.num
FROM    (
        SELECT  distinct L.name,
                COUNT(*) as num
        FROM    language L,
                series SE,
                story ST,
                issue I
        WHERE   L.id = SE.language_id AND
                SE.id = I.series_id AND
                I.id = ST.issue_id AND
                SE.publication_type_id = 2 AND
                (SELECT COUNT(*)
                FROM story_reprint SR
                WHERE SR.target_id = ST.id)=0
        GROUP BY L.name
        ) as T
WHERE   T.num >= 10000
ORDER BY T.num DESC
```

name	num
English	565813
Norwegian	47276
Italian	22723
Dutch	18194

Here we first check that the story is original, that is, it does not exist in the reprint table as a reprinted story. We then check that it is in a magazine (type 2), and we get the languages of all such stories. Then we group by languages and get all those appearing most than 10'000 times.

Query G

```
SELECT  distinct STT.name
FROM    series SE,
        story ST,
        issue I,
        story_type STT
WHERE   STT.id = ST.type_id AND
        I.id = ST.issue_id AND
        SE.id = I.series_id AND
        SE.country_id <> 51 AND
        SE.publication_type_id = 2
```

name
cover
comic story
foreword introduction preface afterword
text story
text article
cartoon
activity
advertisement
letters page
illustration
promo (ad from the publisher)

Here we want a magazine series (we have hardcoded the magazine type as type 2) that is not italian (the Italian country code is 51 so we want all other numbers than 51). Then we simply go through a couple tables to get to the story_type table and return all of them.

Query H

```
SELECT  A.name
FROM    artist A,
        has_script HS,
        story S,
        issue I,
        indicia_publisher IP
WHERE   A.id = HS.artist_id AND
```

```
        HS.story_id = S.id AND
        S.type_id = 5 AND
        S.issue_id = I.id AND
        I.indicia_publisher_id = IP.id
GROUP BY A.name
HAVING COUNT(*) > 1
```

name
Art Helfant
Ben Allen
Bob Kane
Capt. Roscoe Fawcett
Charles J. Dunn
Chet Smith
Clyde Lewis
David B. Icove
Ed Reed
Ed Wheelan
Falcon Mathieu

Here we first take all artists writers of cartoons, and we count for how many indicia they've worked. We simply take all those that have more than one (= many) indicia publishers.

Query I

```
SELECT  distinct BG.name,
        COUNT(*) as ipn
FROM    indicia_publisher IP,
        brand_group BG
WHERE   BG.publisher_id = IP.publisher_id
GROUP BY BG.name
ORDER BY ipn DESC LIMIT 10
```

name	ipn
Marvel	194
Disney Comics	124
Malibu	123
Shadowline	122
CrossGen	118
Soleil	114
Star Comics	113
Atlas	112
Legendary	111
Marvel; New Universe [white box]	110

Here it is straight-forward. We link brand_groups and indicia_publisher and we group_by brand group tp have the number of indicia it is associated with. We take the 10 having the most indicia publishers.

Query J

```
SELECT  T.name,
        AVG(years) AS average_years
FROM    (
        SELECT  distinct I.name,
                (S.year_ended - S.year_began) AS years
        FROM    series S,
                indicia_publisher I
        WHERE   S.year_began < S.year_ended AND
                S.year_began > 0 AND
                S.year_ended > 0 AND
                S.publisher_id = I.publisher_id
        ) AS T
GROUP BY T.name
```

name	average_years
Åhlén & Åkerlunds Förlags AB (Albert Bonnier)	14.894736842105264
DrMaster Publications Inc.	2
12 Gauge Comics	1
12 Gauge Comics LLC	1
20th Century Comic Corp.	13.72
21 Publishing Corp.	7.714285714285714
215 Ink.	2
299 Lafayette St. Corp.	1
299 Lafayette Street Corp.	1
3 Finger Prints Inc.	2

Now, to have the average series length we compute the difference in years between the first issue of the series and the last one. We want the last issue's year to be greater than the initial one, and both to be positive (useless yet secure check). Then we use the commence AVG to compute the average of all those differences.

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

Query K

```
SELECT  I.name,
        COUNT(*) as nb
FROM    series S,
        indicia_publisher I
WHERE   S.first_issue_id = S.last_issue_id
AND
        S.publisher_id = I.publisher_id
GROUP BY I.name
ORDER BY nb DESC LIMIT 10
```

name	nb
Hercules Publishing Corp.	6956
Malibu Comics Entertainment Inc.	4434
Homage Comics	4427
DC Comics	3726
Educational Comics Inc.	3723
WildStorm Productions	3696
M. C. Gaines	3696
Wonder Woman Publishing Company Inc.	3694
DC Comics; Top Cow Productions Inc.	3694
More Fun Inc.	3694

Straight-forward again, we group by indicias and choose those having the most series (we also want the series to have one issue, i.e. its first and last issue are the same).

Query L

```
SELECT  S.id,
        IP.name,
        COUNT(*) as nb
FROM    story S,
        has_script HS,
        issue I,
        indicia_publisher IP
WHERE   HS.story_id = S.id AND
        S.issue_id = I.id AND
        I.indicia_publisher_id = IP.id
GROUP BY S.id, IP.name
ORDER BY nb DESC LIMIT 10
```

id	name	nb
61362	National Comics Publications Inc.	4
61359	National Comics Publications Inc.	4
14247	Timely Publications	4
8980	Picture Publications Inc.	4
61361	National Comics Publications Inc.	4
61363	National Comics Publications Inc.	4
61360	National Comics Publications Inc.	4
61364	National Comics Publications Inc.	4
61365	National Comics Publications Inc.	4
23399	Timely Comics Inc.	4

For a given story, here we take all the indicia and all the script writers that have worked for each of them. We then simply take the 10 indicia with the most writers.

Query M

```
SELECT  distinct C.name
FROM    characters C,
        has_characters HC ,
        has_featured_characters HFC,
        story S,
        issue I,
        indicia_publisher IP
WHERE   HC.character_id = C.id AND
        HFC.character_id = C.id AND
        (HC.story_id = S.id OR HFC.story_id = S.id) AND
        (S.issue_id = I.id OR I.id = S.issue_id) AND
        (IP.id = I.indicia_publisher_id OR I.indicia_publisher_id) AND
        IP.name LIKE '%Marvel%DC%'
```

name
Max
Mr. Jones
Mr. Robinson
Daral
Elfa the Evil
Uncle Sam
Mary Jane
Foxy Grandpa
Frank

We were not exactly sure how to understand the query so we looked for all the characters from indicia publishers containing both names Marvel and DC. We assumed that we can't really know if a character is a Marvel one since it can only appear and crossovers and still be only a Marvel character, so we thought only checking the crossover condition would be enough (adding other conditions lead to empty results anyway),

Query N

```
SELECT  T.name
FROM    (
        SELECT  S.name,
                COUNT(*) as inb
        FROM    series S,
                issue I
        WHERE   I.series_id = S.id
        GROUP BY S.id
        ) AS T
ORDER BY T.inb DESC LIMIT 5
```

name
Commando
Fliegende Bl&#65533;tter
Spirou
Robbedoes
The Beano

Here we simply check that an issue belongs to a series and take the series with the most issues. Then we do a second select to keep the column with the issues name only.

Query O

```
SELECT  S.title,
        COUNT(*) as nb
        FROM   story S,
        story_reprint SR
        WHERE  S.issue_id = ".$id." AND
        S.id = SR.origin_id
        GROUP BY S.title
        ORDER BY nb DESC LIMIT 1
```

title	nb
Tintin au Congo	2

Here we select the issue_id with the correct id. Here the id is a parameter given by the php code, which is the cause for the double quotes, receiving *.$id* as a parameter. For example, if $id is equal to 68, here is what we get as a screenshot.

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

Query Analysis

Selected Queries (and why)

Note that, in addition to the particular improvements explained below, we also added a couple more indexes to our database in order to gain more time. Those indexes are B-trees. All the improved optimized queries times shown below are computed with the additional indexes activated. These indexes are:

story.issue_id
story.title
series.name
series.publisher_id
issue.series_id
issue.title
issue.indicia_publisher_id
indicia.publisher_id

Query B

Initial Running time: **0.2004** seconds
Optimized Running time: **0.1351** seconds
Explain the improvement: Since we know that there are only 3 different series types, we can simply check that a publisher has publisher 3 different types of medias, instead of checking if all 3 types of media are present in the publisher's series. This avoids all access to the "series_publication_type" table, and remove a whole sub-query from the final query.

Initial plan

```
SELECT  P.name
 FROM   publisher P
 WHERE  (SELECT COUNT(distinct SP.name)
        FROM    series_publication_type SP,
                series S
        WHERE   SP.id = S.publication_type_id AND
                S.publisher_id = P.id
        ) =
        (SELECT COUNT(distinct SP.name)
        FROM    series_publication_type SP)
```

Improved plan

```
SELECT  P.name
FROM    publisher P
WHERE   (SELECT COUNT(distinct SP.name)
        FROM    series_publication_type SP,
                series S
        WHERE   SP.id = S.publication_type_id AND
                S.publisher_id = P.id
        ) = 3
```

Query G

Initial Running time: **0.0331** seconds
Optimized Running time: **0.0213** seconds
Explain the improvement: Since the magazines have fixed id 2 and Italy has fixed country id 53, we can directly hardcore them, which let us not load tables “country” and “series_publication_type”.

Initial plan

```
SELECT  distinct STT.name
        country C,
FROM    language L,
        series SE,
        story ST,
        issue I,
        story_type STT
WHERE   NOT C.name = 'Italy' AND
        C.id = SE.country_id AND
        SE.id = I.series_id AND
        I.id = ST.issue_id AND
        SE.publication_type_id = 2 AND
        STT.id = ST.type_id
```

Improved plan

```
SELECT  distinct STT.name
FROM    series SE,
        story ST,
        issue I,
        story_type STT
WHERE   STT.id = ST.type_id AND
        I.id = ST.issue_id AND
        SE.id = I.series_id AND
        SE.country_id <> 51 AND
        SE.publication_type_id = 2
```

Query I

Initial Running time: **0.2387** seconds
Optimized Running time: **0.1510** seconds
Explain the improvement: We avoided all accesses to the “publisher” table since bith brand_group and indicia_publisher share a common index “publishier_id”. Also, we merged our double query in one single query.

Initial plan

```
SELECT  T.name,
        COUNT(*) as ipn
FROM    (
        SELECT  distinct BG.name,
                IP.id
        FROM    indicia_publisher IP,
                publisher P,
                brand_group BG
        WHERE   BG.publisher_id = P.id AND
                IP.publisher_id = P.id
        ) AS T
GROUP BY T.name
ORDER BY ipn DESC LIMIT 10
```

Improved plan

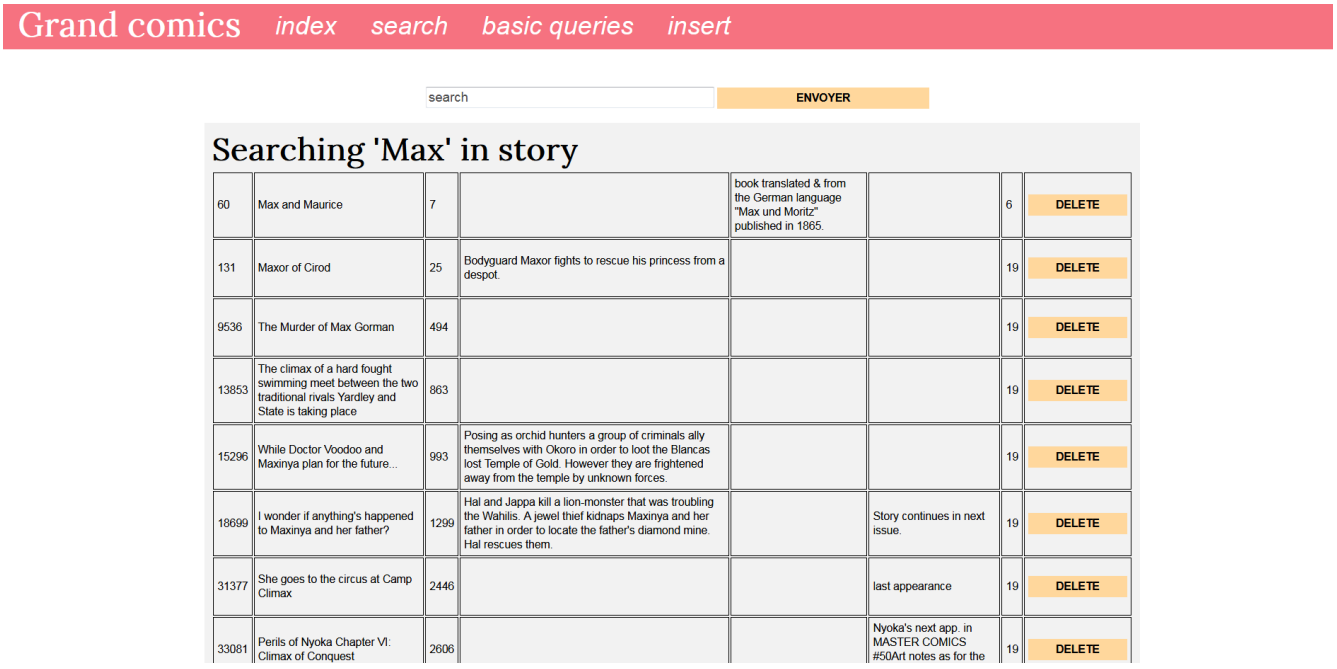
DIAS: Data-Intensive Applications and Systems Laboratory
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL:http://dias.epfl.ch/

```
SELECT  distinct BG.name,
        COUNT(*) as ipn
FROM    indicia_publisher IP,
        brand_group BG
WHERE   BG.publisher_id = IP.publisher_id
GROUP BY BG.name
ORDER BY ipn DESC LIMIT 10
```

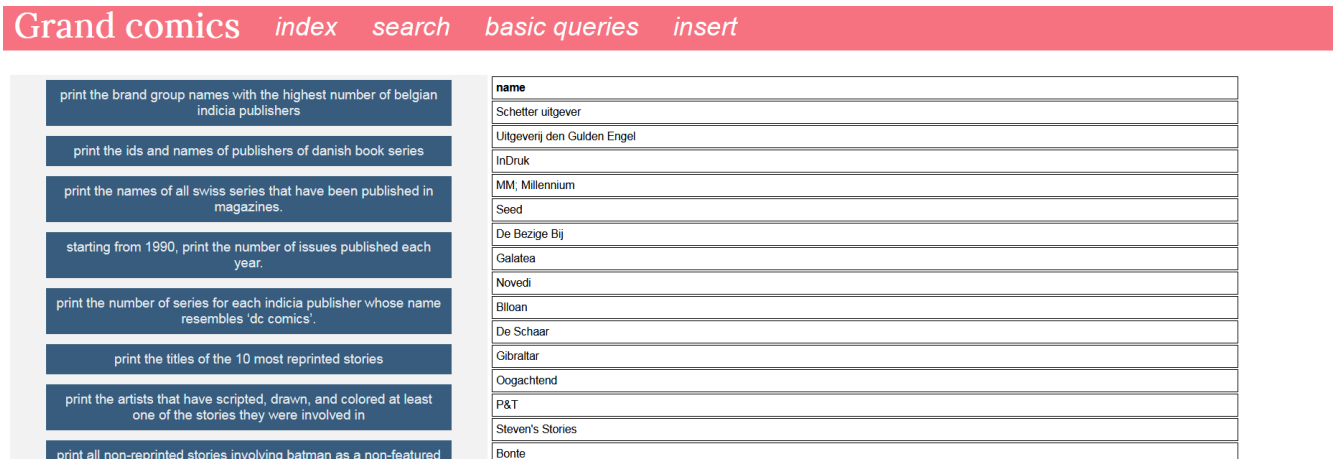
Interface

Design logic Description

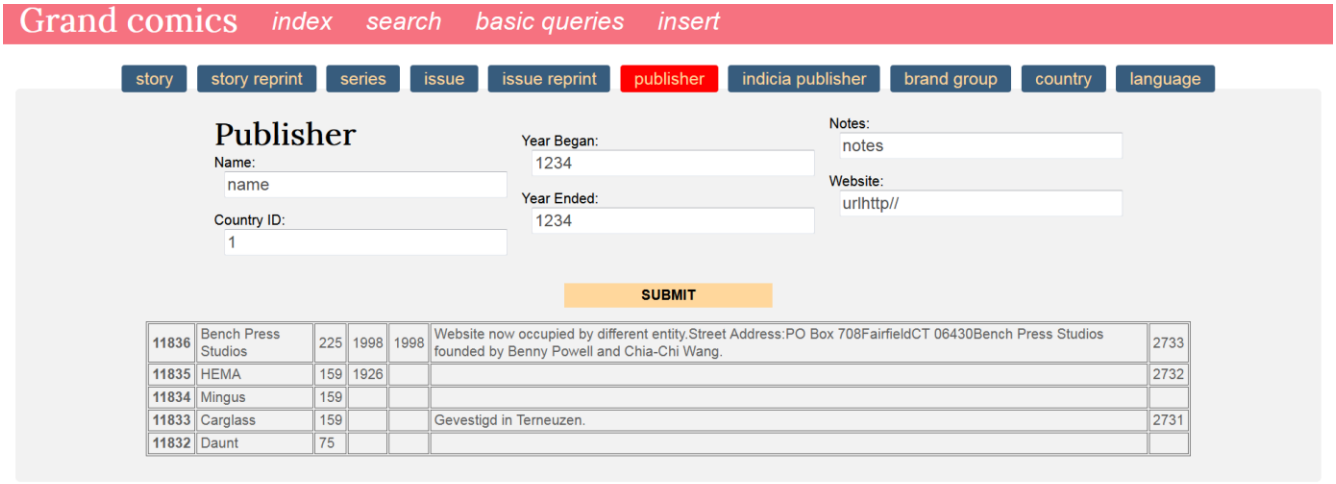
Our interface contains three main pages :



- **search/delete**: this page let the user searches into the database (we choose one column of main tables such as story's title, artist's name, character's name, series' name, etc..). Once the research is done, the possibility to delete a specific entry is also possible.



- **basic queries**: Here buttons are clickable and send a predefined SQL query to out database manager. The result is then displayed on the side, along with the needed time to execute the query.



- **insert**: With tables represented as tabs, the user can insert an entry, field by field. Some constraints are added, so that the user cannot insert anything. For example, primary and foreign keys must be respected, while years has to be a number. Error feedback happens if the fields are not well filled. Last five added entries for each table are also displayed.

General Comments

Data were first transformed into SQL insert commands to be then submit to the database manager. However, that process was too tedious and not scalable, so inserting data that way took *ages*. We then change data into new csv to be directly imported, which was faster to execute.

We chose what table to use and create together.

Kim parsed and inserted the data into the database, while building the interface.

Lucie wrote the queries for milestone 2 and 3, and the SQL DDL + relational model for the tables.

Tim drew the ER model.