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/

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

□ Having just the price as an attribute for “issue” is not possible because there exists different currencies for the prices of the issues. So we added a table “price” related to the “issue” table to take into account the currency and the amount in this currency.

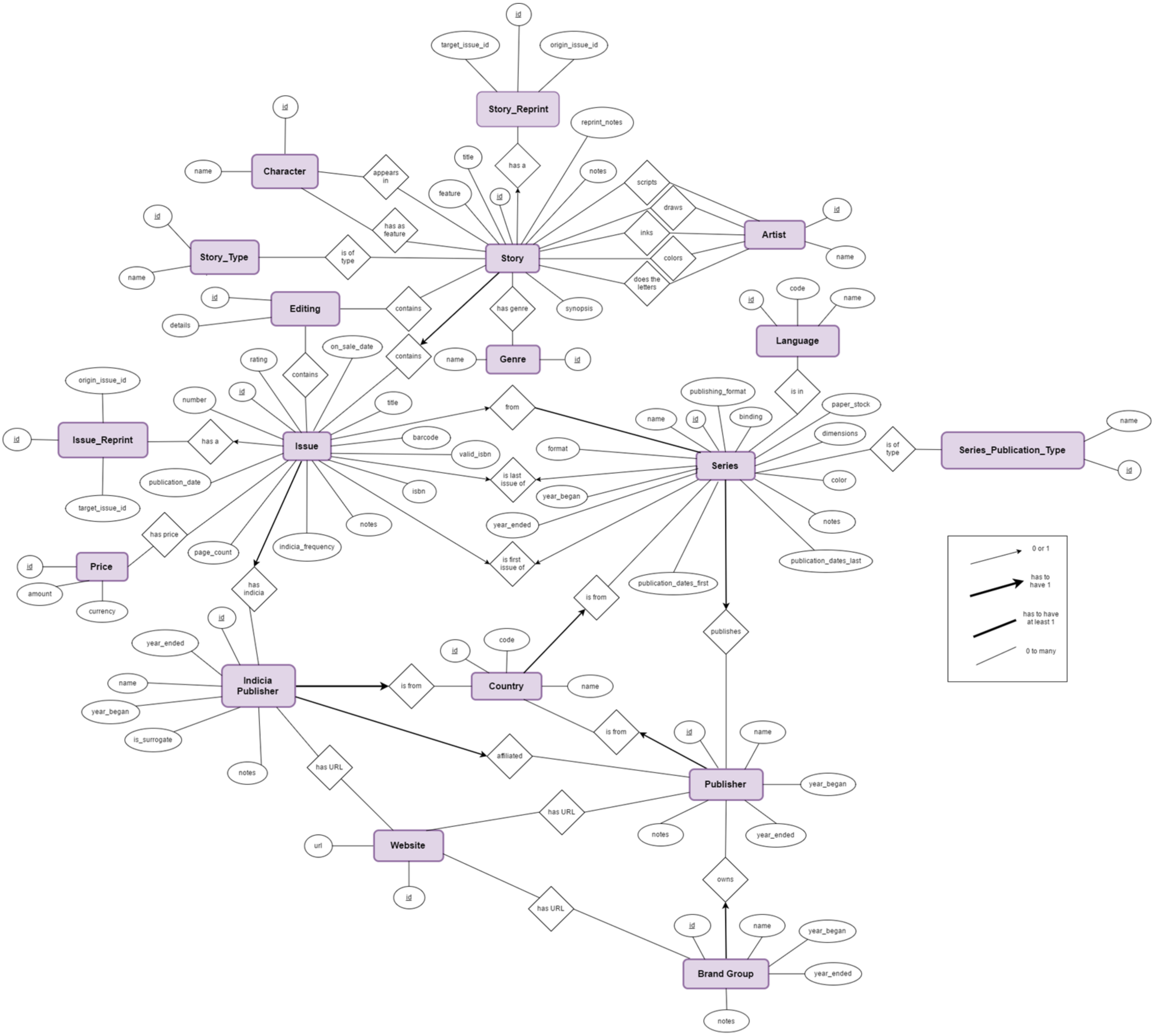
□ The table “genre” that we have created is not essential but we did this for some efficiency purpose. This is faster to get our genres in a table as as an attribute.

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
- **Price** : id, amount, currency which describe the different prices for an issue
- **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



- **Editing** : id, details which describe the different editing details about a story or an issue

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

Similarily, we linked the price entity to the issue entity through the “has_price” relationship, since a single issue can have many prices, expressed in many currencies.

We also created a table “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 share common websites.

Relational Schema

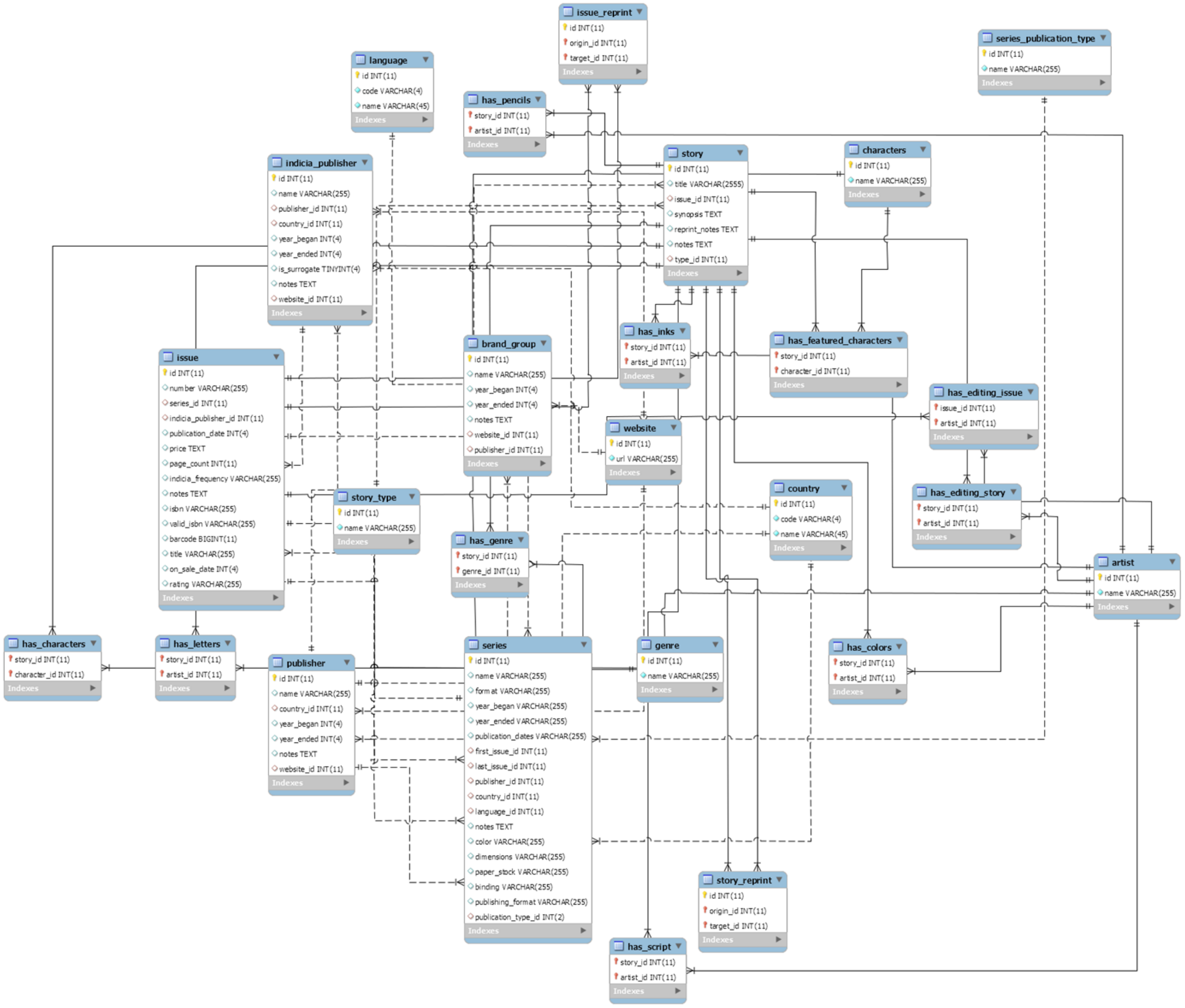
ER schema to Relational schema

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.

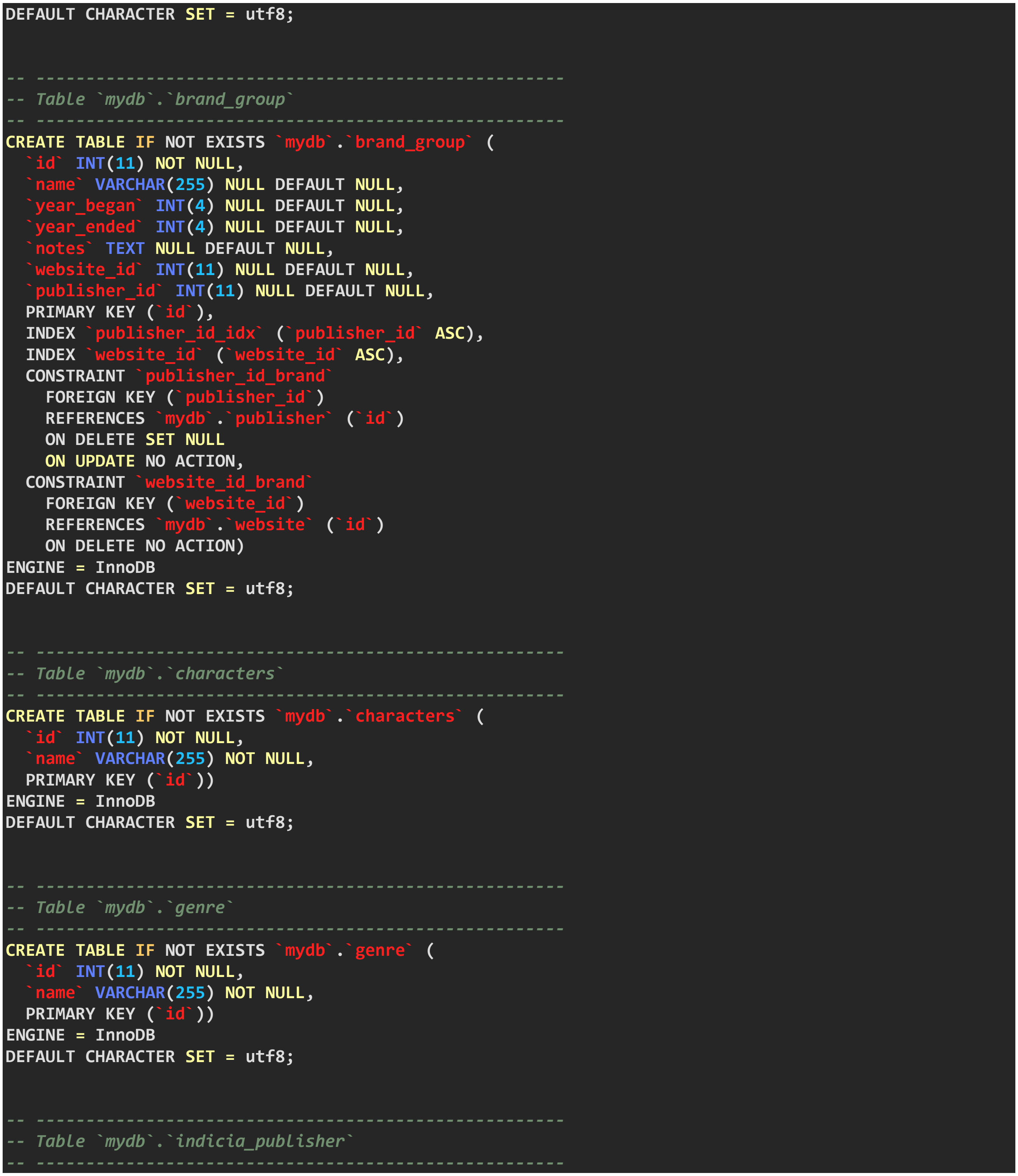
- For each relation from 1 to n, we created an intermediate table “has_...” containing relations. A relation is composed of both ids of the two entities connected through this relation, and the id of the relation. This relation table contains then 2 foreign keys, pointing on both entities related. We 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.


issue_reprint
id INT(11)
origin_id INT(11)
target_id INT(11)
Indexes
series_publication_type
id INT(11)
name VARCHAR(255)
Indexes
language
id INT(11)
code VARCHAR(4)
name VARCHAR(45)
Indexes
has_pencils
story_id INT(11)
artist_id INT(11)
Indexes
story
id INT(11)
title VARCHAR(2555)
issue_id INT(11)
synopsis TEXT
reprint_notes TEXT
notes TEXT
type_id INT(11)
Indexes
characters
id INT(11)
name VARCHAR(255)
Indexes
indicia_publisher
id INT(11)
name VARCHAR(255)
publisher_id INT(11)
country_id INT(11)
year_began INT(4)
year_ended INT(4)
is_surrogate TINYINT(4)
notes TEXT
website_id INT(11)
Indexes
has_inks
story_id INT(11)
artist_id INT(11)
Indexes
has_featured_characters
story_id INT(11)
character_id INT(11)
Indexes
issue
id INT(11)
number VARCHAR(255)
series_id INT(11)
indicia_publisher_id INT(11)
publication_date INT(4)
price TEXT
page_count INT(11)
indicia_frequency VARCHAR(255)
notes TEXT
isbn VARCHAR(255)
valid_isbn VARCHAR(255)
barcode BIGINT(11)
title VARCHAR(255)
on_sale_date INT(4)
rating VARCHAR(255)
Indexes
brand_group
id INT(11)
name VARCHAR(255)
year_began INT(4)
year_ended INT(4)
notes TEXT
website_id INT(11)
publisher_id INT(11)
Indexes
has_editing_issue
issue_id INT(11)
artist_id INT(11)
Indexes
website
id INT(11)
url VARCHAR(255)
Indexes
country
id INT(11)
code VARCHAR(4)
name VARCHAR(45)
Indexes
has_editing_story
story_id INT(11)
artist_id INT(11)
Indexes
story_type
id INT(11)
name VARCHAR(255)
Indexes
has_genre
story_id INT(11)
genre_id INT(11)
Indexes
artist
id INT(11)
name VARCHAR(255)
Indexes
has_characters
story_id INT(11)
character_id INT(11)
Indexes
has_letters
story_id INT(11)
artist_id INT(11)
Indexes
publisher
id INT(11)
name VARCHAR(255)
country_id INT(11)
year_began INT(4)
year_ended INT(4)
notes TEXT
website_id INT(11)
Indexes
series
id INT(11)
name VARCHAR(255)
format VARCHAR(255)
year_began VARCHAR(255)
year_ended VARCHAR(255)
publication_dates VARCHAR(255)
first_issue_id INT(11)
last_issue_id INT(11)
publisher_id INT(11)
country_id INT(11)
language_id INT(11)
notes TEXT
color VARCHAR(255)
dimensions VARCHAR(255)
paper_stock VARCHAR(255)
binding VARCHAR(255)
publishing_format VARCHAR(255)
publication_type_id INT(2)
Indexes
genre
id INT(11)
name VARCHAR(255)
Indexes
has_colors
story_id INT(11)
artist_id INT(11)
Indexes
story_reprint
id INT(11)
origin_id INT(11)
target_id INT(11)
Indexes
has_script
story_id INT(11)
artist_id INT(11)
Indexes

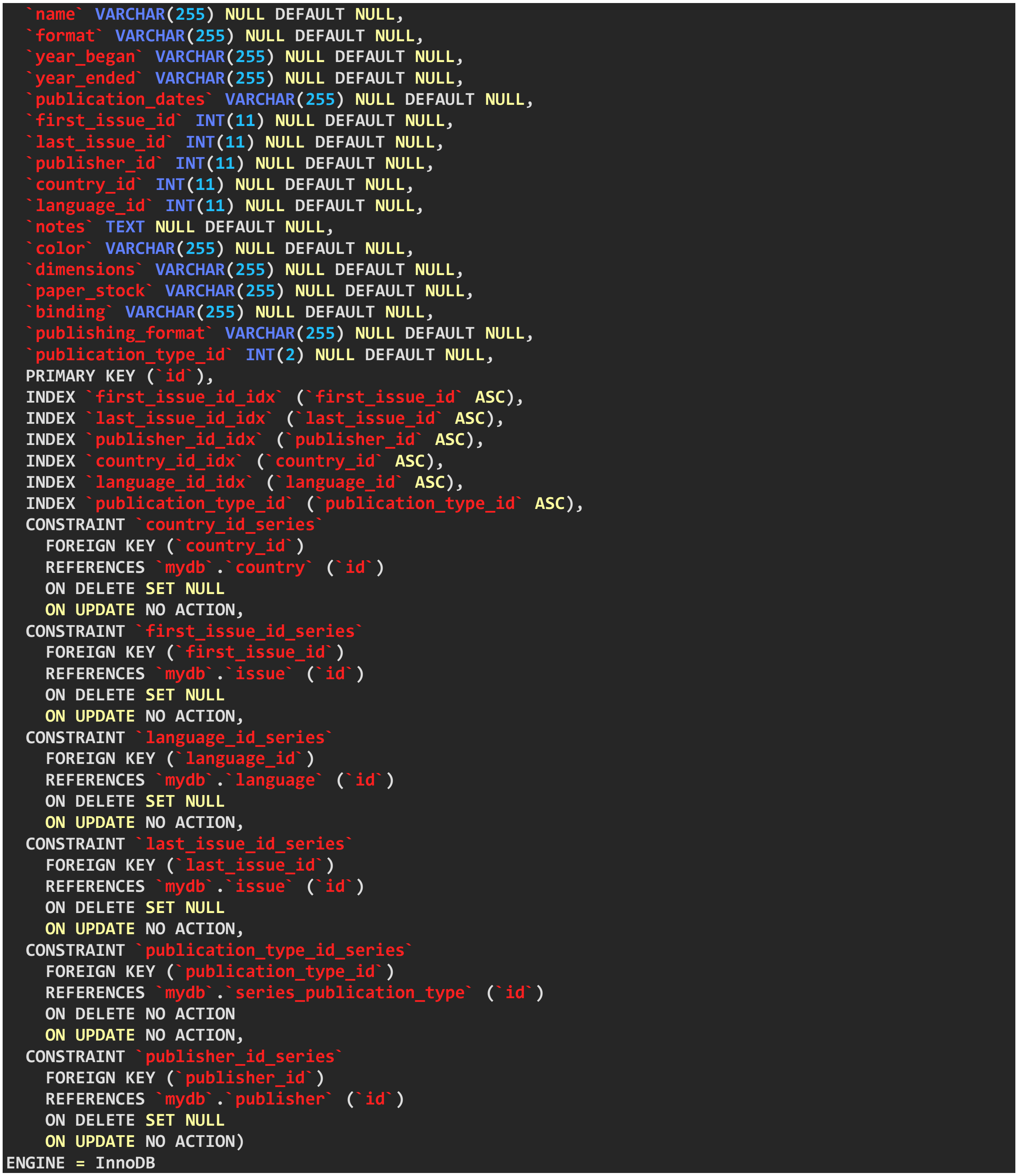
DDL

```sql
- -----------------------------------------------------
-- 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
```



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

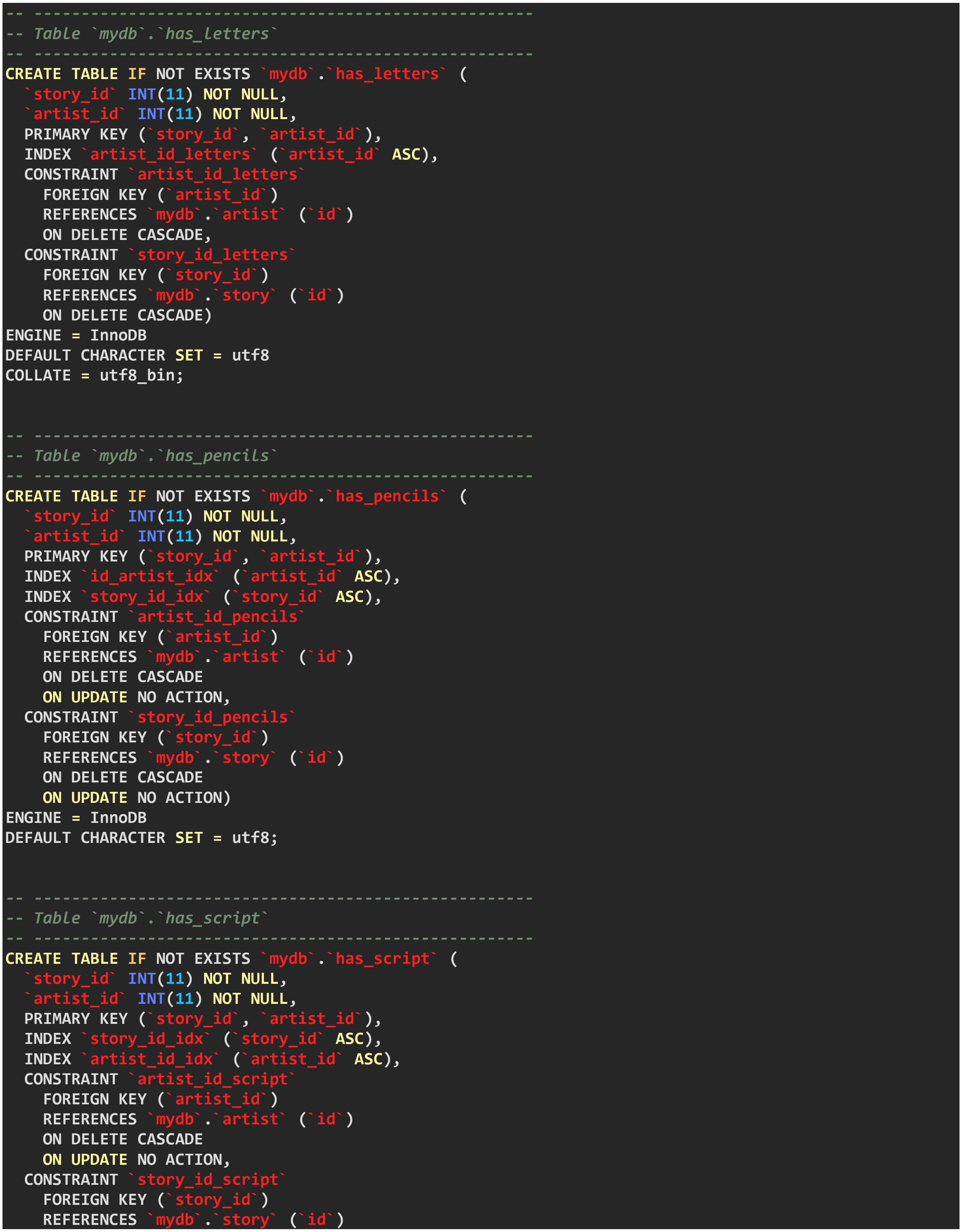
```sql
CREATE TABLE IF NOT EXISTS `mydb`.`indicia_publisher` (
  `id` INT(11) NOT NULL,
  `name` VARCHAR(255) NULL DEFAULT NULL,
  `publisher_id` INT(11) NULL DEFAULT NULL,
  `country_id` INT(11) NULL DEFAULT NULL,
  `year_began` INT(4) NULL DEFAULT NULL,
  `year_ended` INT(4) NULL DEFAULT NULL,
  `is_surrogate` 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,
```



```
  `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
```

```sql
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,
  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`)
    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;
```



```sql
-- -----------------------------------------------------
-- 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`)
    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).

Assumptions

Parsing leads to lots of assumptions. Here the ones we had to assume :

- 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 duplicata or obtain full names of artists (our actual version uniquely keep first found entry but should be modify so that we keep the longer and most detailed ones).
- Dates are difficult to retrieve from given csv file, because of its non-uniform format. We assume:
  - “1870’s” become “1870”
  - “July 10 1870” become “1870”
  - “1870-07-10” become “1870”
- Stories with null titles are deleted.

Parsing and cleaning of the data

We choose to use a local database, using **wamp server** (windows local server) and **phpmyadmin**. Thus we are parsing the csv file using php.

We had to parse multiple times the csv’s given to us.
For the beginning, we created scripts which create sql commands from csv data. During the process, we get rid of null values such as “Null”, “none”, “[nn]”, “?”, etc. We also had to change some column types such as:

- isbn , rating, number of issue becoming varchar
- synospsis of story becoming text
- dates becoming integer

TODO KL changer les commentaires sur les fichers php et beaucoup parler du cleaning

We used the following php commands to do the parsing. This is the “*functions.php*” file, used in every php parsing files.

[functions.php]

```
<?php

function parseNullValue($s) {

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

    $nullValues = ['NULL', '[nn]', 'nn', 'none', '[none]','?', '(unknown)','None'];
    foreach($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];
        }
    }

    $month=1; $day=1;
    $hour=0; $minute=0; $second=0;

    // no date with 4 digits
    return "0";
}

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;
}

// 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;
}


// 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 :
Websites come from publisher, indicia_publisher and brand_group csv file. We get the url values, parse null 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). These csv files are then process into sql commands to import them into our database.

[website_tocsv.php]

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

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

echo "<h1>Websites</h1>";

$index = getLastIndex($csv);

$min = 0;
$max = 10000000;


// get websites from all given files
foreach ($files as $f => $pos) {
 $i = 0;
 echo "begin with " . $f;
 $file = fopen($f,"r");
 $val = fgetcsv($file); // avoid url column name

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

  if($i > $min){

   $url = $val[$pos];

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

  if($i==$max){
   return;
  }
 }
 fclose($file);

}

fclose($csv);
?>
```

[website_to_id.php]

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

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

echo "<h1>Websites to id</h1>";
echo "<h2>Don't forget to run websites script first !!</h2>";

$index = getLastIndex($csv);

$min = 0;
$max = 1000000000;

// get websites from all given files
foreach ($files as $f => $pos) {
 $i = 0;
 $file = fopen($f,"r");
 $out = implode(",",fgetcsv($file))."\n";

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

  if($i > $min){

   $url = $val[$pos];

   if(!parseNullValue($url)){
    $index = isInCsvName($csv, $url,1);
    $val[$pos] = $index;
   }

   //var_dump($val);
   $out .= implode(",",$val)."\n";
```

```
      if($i>$max) break;
    }

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

fclose($csv);
?>
```

[website.php]

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

$file = fopen("comics/website.csv","r");
$mysql = fopen("website.sql", "w"); // write into this sql to import


/*
website id ->> mettre dans table website -> remettre foreign key
year -> date
*/

$min = 0;
$max = 100000;
$i = 0;

/*
  0 => string 'id' (length=2)
  1 => string 'code' (length=4)
  2 => string 'name' (length=4)
*/

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

    if($i > $min){

      $id = getInt($val[0]);
      $url = parseDoubleQuote($val[1]);

      $query = 'INSERT INTO website(id,url) VALUES(
      '.$id.', '.$url.'
      );';

      //print_r($query);
      fwrite($mysql,$query);

      $s1 = $con->query($query);
      /*var_dump($s1);*/
      if($i==$max){
        break;
      }
    }

  }
  fclose($file);
  ?>
```

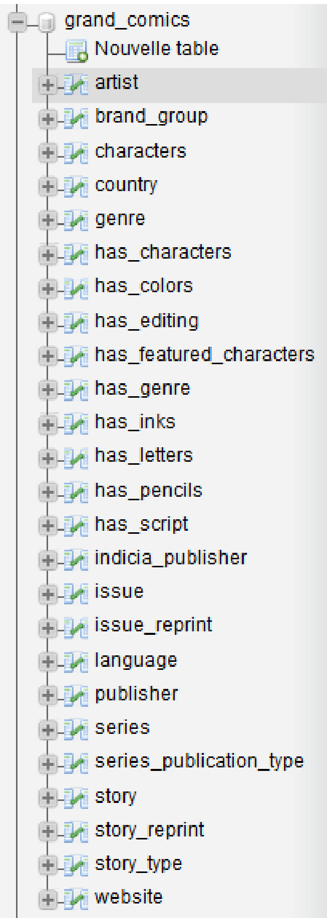
Artist, Characters, Genre :
Artists come from story csv file. We get names, parse null values, and add it in a new artist table.
If a story has an artist (for script, letters, inks…) we add in an artist csv file the found artist (id, artist name) and we also add in a has_ csv file the pairs (story_id, artist_id). These csv files are then process into sql commands to import them into our database.

This same idea is also applied to characters (features and characters from story) and genres.

[genre_tocsv.php]

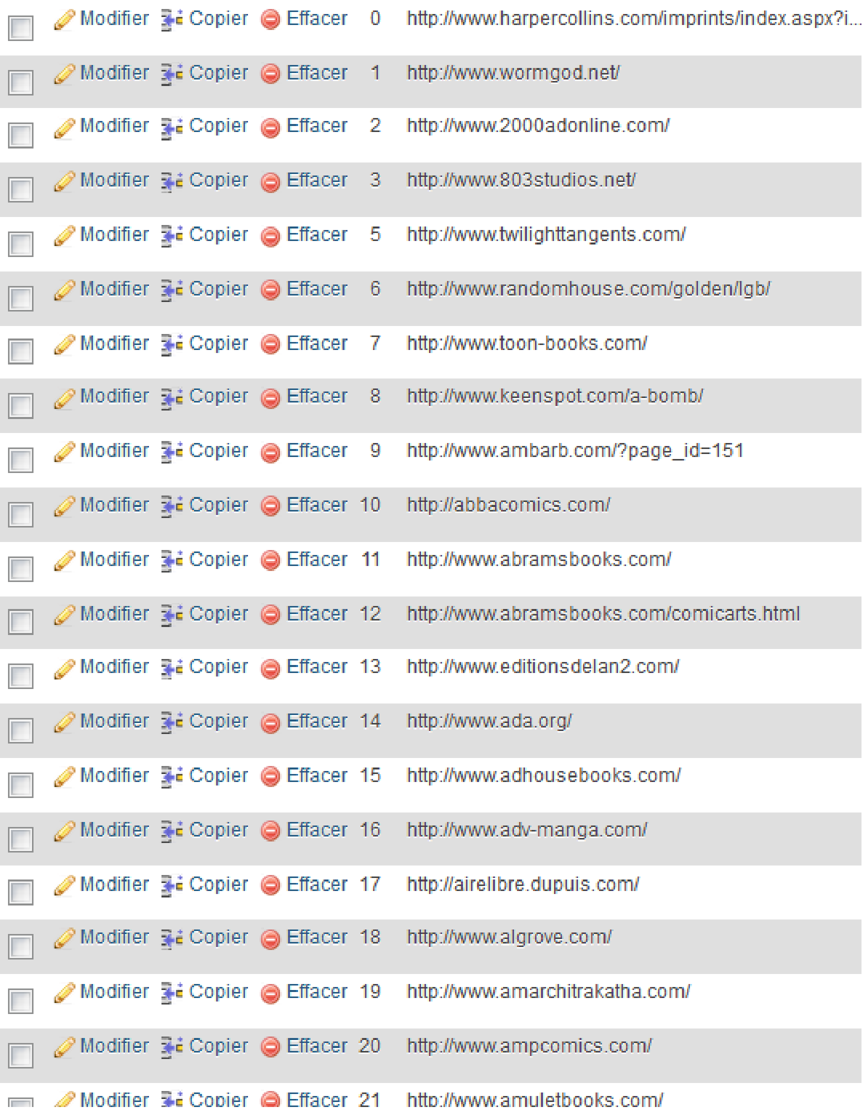
```
<?php
include("db.php");
include("functions.php");

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

echo "<h1>Genre</h1>";

$index = getLastIndex($csv);

$min = 0;
$max = 10000;
$i = 0;

var_dump(fgetcsv($file));

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

  if($i > $min){

    $id = getInt($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)) {
          $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);
        }
      }
    }

    if($i==$max){
      break;
    }
  }
}

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

?>
```

Screenshots of the database, as seen from the phpMyAdmin interface



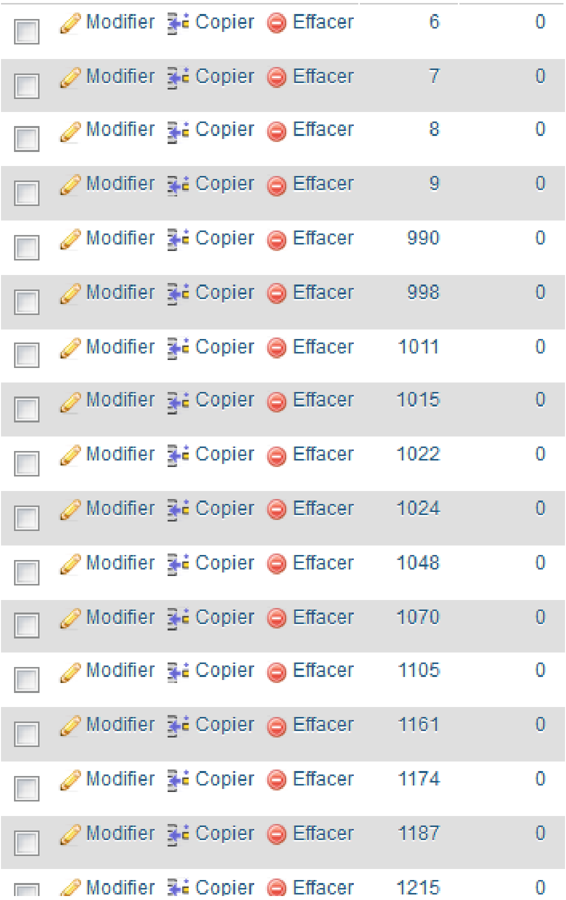
- grand_comics
  - Nouvelle table
  - artist
  - brand_group
  - characters
  - country
  - genre
  - has_characters
  - has_colors
  - has_editing
  - has_featured_characters
  - has_genre
  - has_inks
  - has_letters
  - has_pencils
  - has_script
  - indicia_publisher
  - issue
  - issue_reprint
  - language
  - publisher
  - series
  - series_publication_type
  - story
  - story_reprint
  - story_type
  - website

All tables



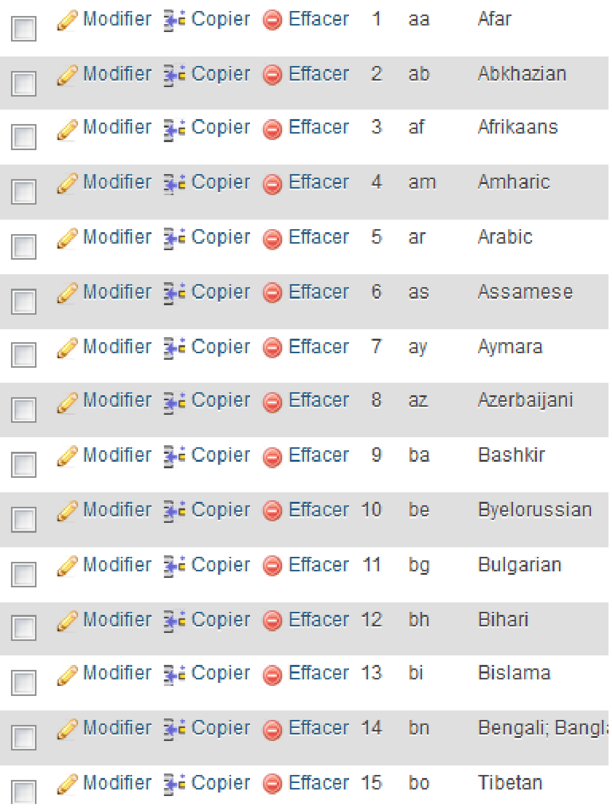
☐	Modifier	Copier	Effacer	0	http://www.harpercollins.com/imprints/index.aspx?i...
☐	Modifier	Copier	Effacer	1	http://www.wormgod.net/
☐	Modifier	Copier	Effacer	2	http://www.2000adonline.com/
☐	Modifier	Copier	Effacer	3	http://www.803studios.net/
☐	Modifier	Copier	Effacer	5	http://www.twilighttangents.com/
☐	Modifier	Copier	Effacer	6	http://www.randomhouse.com/golden/lgb/
☐	Modifier	Copier	Effacer	7	http://www.toon-books.com/
☐	Modifier	Copier	Effacer	8	http://www.keenspot.com/a-bomb/
☐	Modifier	Copier	Effacer	9	http://www.ambarb.com/?page_id=151
☐	Modifier	Copier	Effacer	10	http://abbacomics.com/
☐	Modifier	Copier	Effacer	11	http://www.abramsbooks.com/
☐	Modifier	Copier	Effacer	12	http://www.abramsbooks.com/comicarts.html
☐	Modifier	Copier	Effacer	13	http://www.editionsdelan2.com/
☐	Modifier	Copier	Effacer	14	http://www.ada.org/
☐	Modifier	Copier	Effacer	15	http://www.adhousebooks.com/
☐	Modifier	Copier	Effacer	16	http://www.adv-manga.com/
☐	Modifier	Copier	Effacer	17	http://airelibre.dupuis.com/
☐	Modifier	Copier	Effacer	18	http://www.algrove.com/
☐	Modifier	Copier	Effacer	19	http://www.amarchitrakatha.com/
☐	Modifier	Copier	Effacer	20	http://www.ampcomics.com/
☐	Modifier	Copier	Effacer	21	http://www.amuletbooks.com/

Website table



□	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.

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
```

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

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

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

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

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  COUNT(*)
```

```
FROM    issue I
WHERE   I.publication_date >= 1990
GROUP BY I.publication_date
```

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
```

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)
```

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
```

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  S.title
FROM    story S,
        characters C,
        has_characters HS
WHERE   0 = (   SELECT COUNT(distinct R.origin_id)
                FROM story_reprint R
                WHERE S.id = R.origin_id
            ) AND
        HS.character_id = C.id AND
        HS.story_id = S.id AND
        C.name LIKE '%Batman%' AND
        S.features NOT LIKE '%Batman%'
```

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/

Deliverable 3

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

TODO LUCIE QUERIES A AND M

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.

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 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%')
        )
```

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
```

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.

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
```

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
```

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
```

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
```

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
```

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.

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
```

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
```

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 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
```

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 = 68 AND
        S.id = SR.origin_id
GROUP BY S.title
ORDER BY nb DESC LIMIT 1
```

TODO REPLACE 68 BY ACTUAL PHP

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

```
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

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

Screenshots

<Provide some initial screen shots of your interface>

General Comments

<In this section write general comments about your deliverable (comments and work allocation between team members>