Databases Project – Spring 2017

Prof. Anastasia Ailamaki

Team No: **14**

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

Contents

Deliverable 1 2

Deliverable 1 2

Assumptions 2

Entity Relationship Schema 2

Schema 2

Description 2

Relational Schema 3

ER schema to Relational schema 3

DDL 5

Deliverable 2 15

Assumptions 15

Parsing and cleaning of the data 16

Query Implementation 26

Deliverable 3 29

Assumptions 29

Query Implementation 29

Query Analysis 29

Selected Queries (and why) 29

Query 1 29

Query 2 29

Query 3 29

Interface 30

Design logic Description 30

Screenshots 30

General Comments 30

# 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:

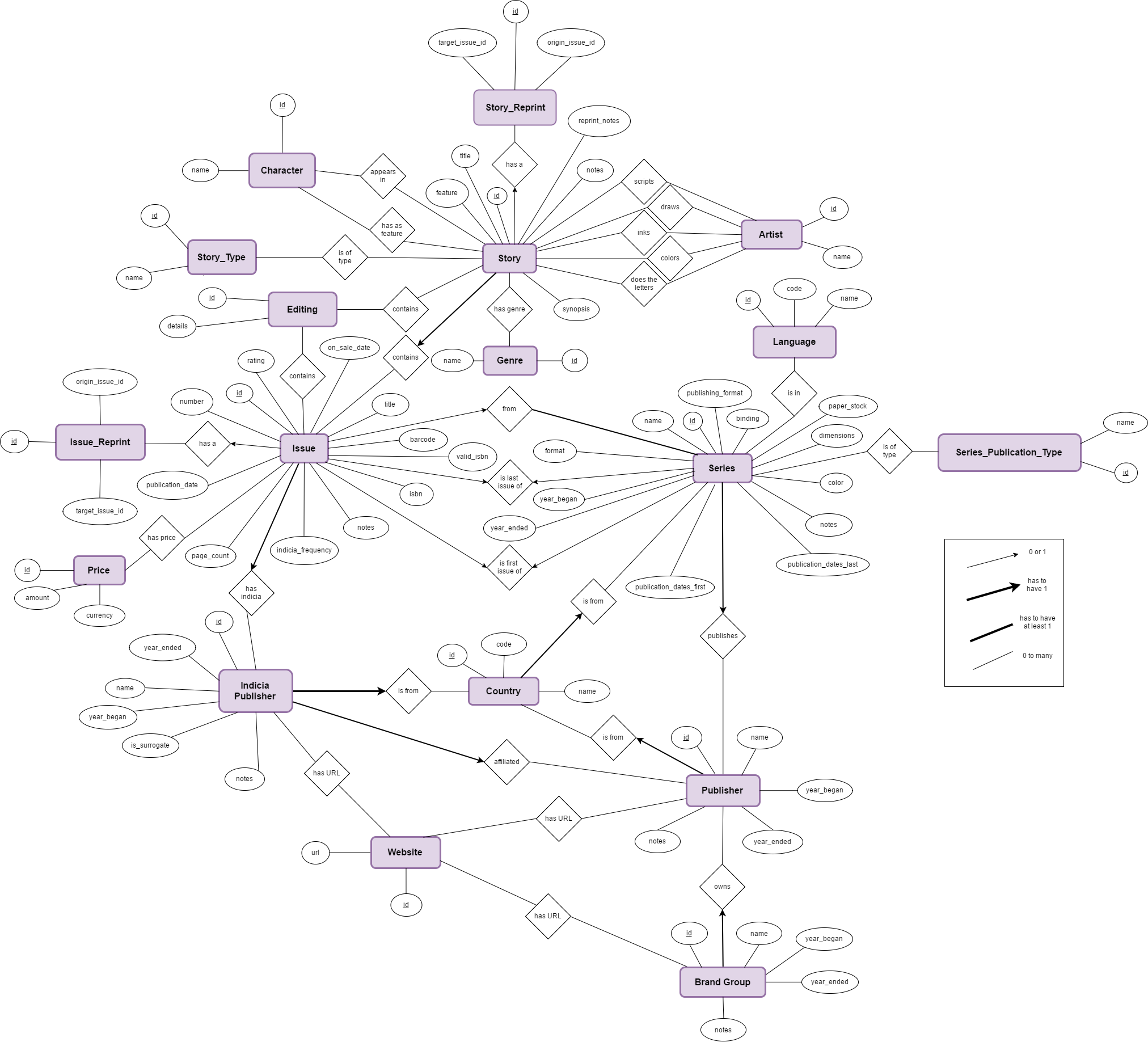
* If one person has written thousands of books, it would be a loss of space to copy their name in every issue whey worked in. Instead, we would create a table “artist” with this person inside, and just this person’s id to all the books they have written
* Having just the price as an attribute for “issue” 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 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
* **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
* *editing:*

Similarly, 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.

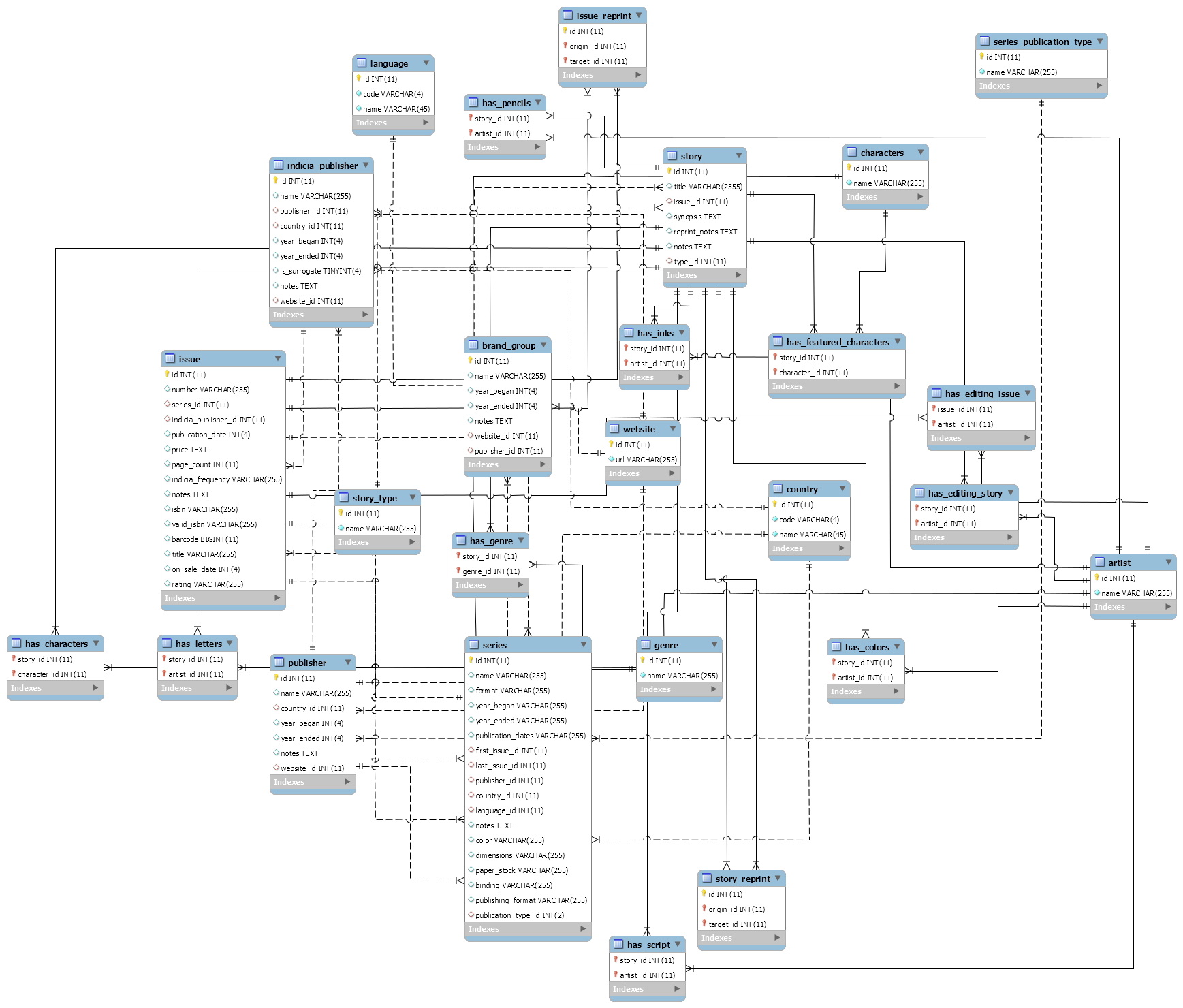
## Relational Schema

### 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.



### 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**

**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`***

***-- -----------------------------------------------------***

**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**

**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;**

***-- -----------------------------------------------------***

***-- 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).

## 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

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];

}

}

// 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;

}

// 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\_tocsv.php]

<?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

$index = getLastIndex($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

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

$index = getLastIndex($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

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)) {

$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);

}

}

}

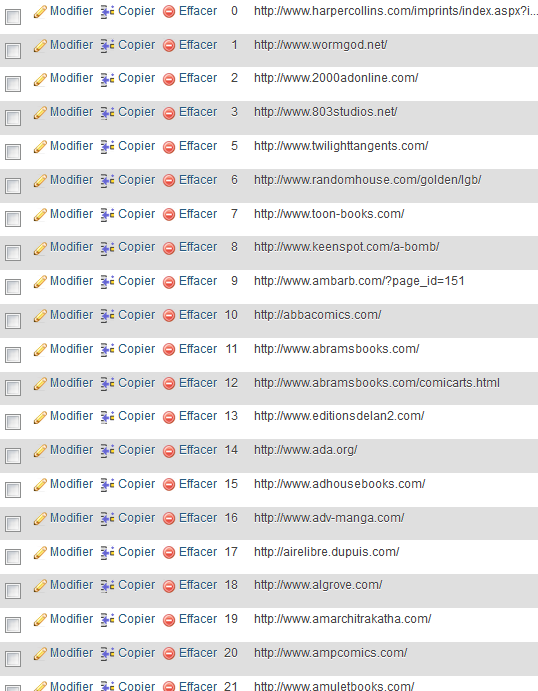
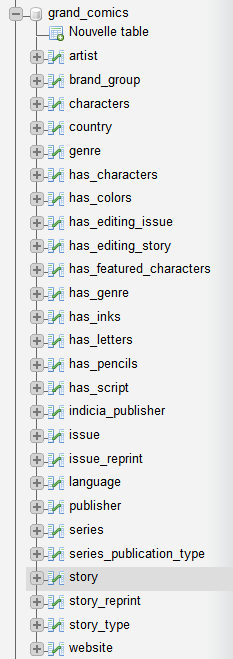
}

fclose($file);

fclose($csv);

?>

**Screenshots of the database, as seen from the phpMyAdmin interface**



All tablesWebsite table

Has\_genre table 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**

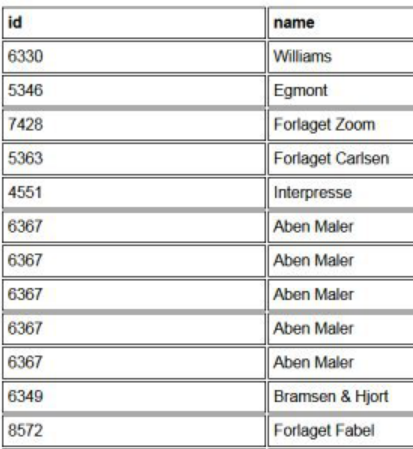
**C**.**id** **=** **I**.**country\_id** **AND**

**I**.**publisher\_id** **=** **B**.**publisher\_id**

**GROUP BY** **B**.**name**

) **AS** T

**ORDER BY** **T**.**bid**

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

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,

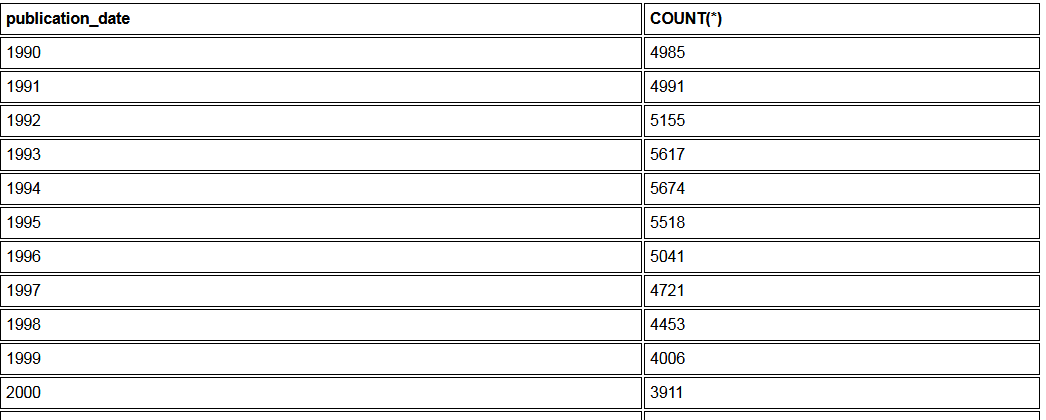
series\_publication\_type T

**WHERE** **T**.**name** **=** 'magazine' **AND**

**T**.**id** **=** **S**.**publication\_type\_id** **AND**

**S**.**country\_id** **=** **40**

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**,

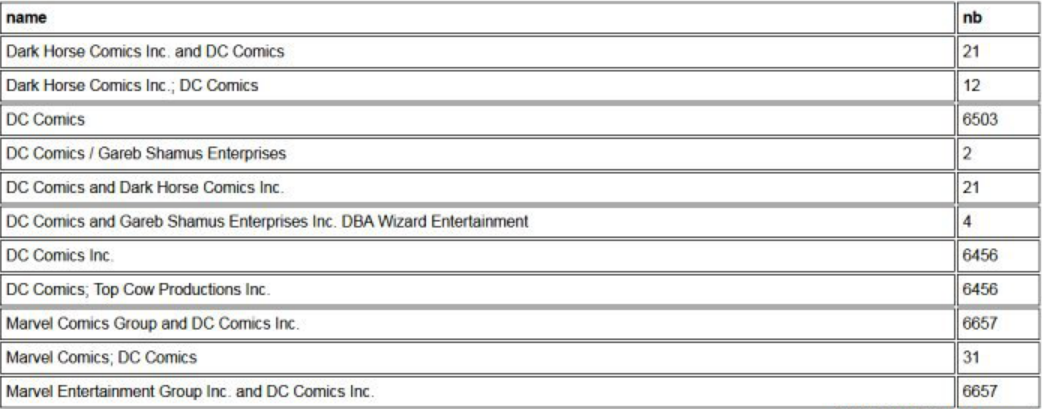
**COUNT**(**\***)

**FROM** issue I

**WHERE** **I**.**publication\_date** **>=** 1990

**GROUP BY** **I**.**publication\_date**

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,

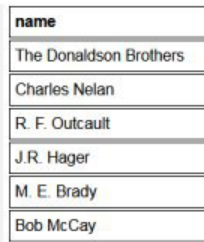
story\_reprint R

**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**

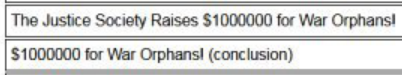
**S**.**id** **=** **I**.**story\_id**

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** **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**

**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**

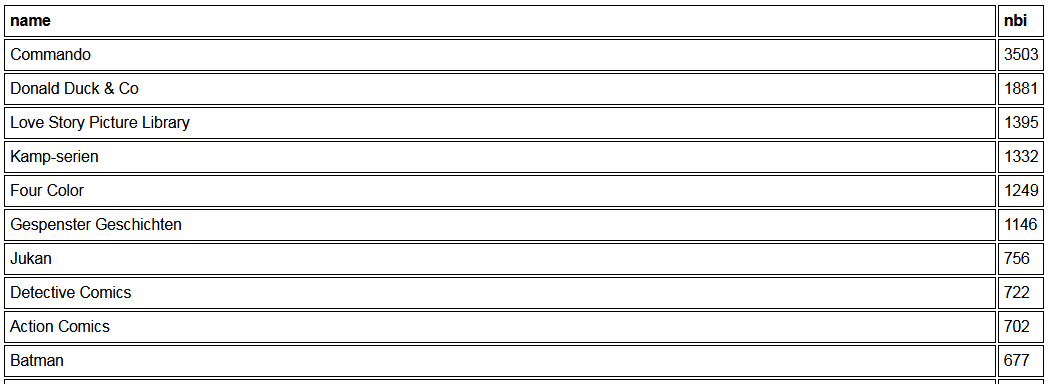
) **as** T2,

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.



#### 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**

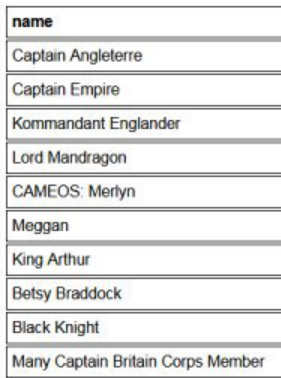
) **=** 3

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**

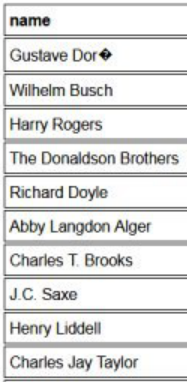
**ORDER BY** **T**.**nch** **DESC** **LIMIT** 10

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**

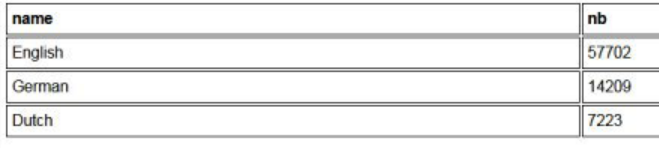
**S**.**synopsis** **LIKE** '%natur%')

)

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. (We only had one result, probably because the DB is not completely full).

#### 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,

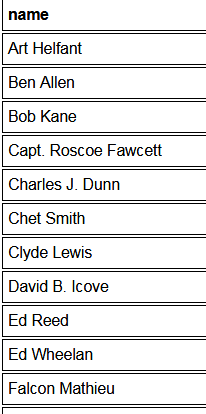
has\_script HS,

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**

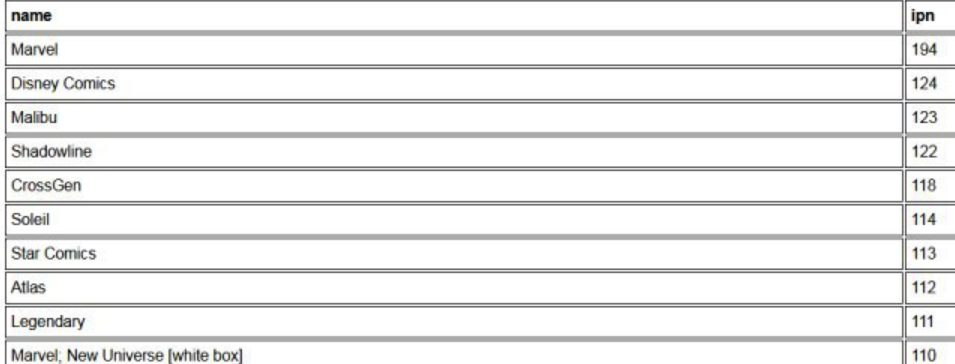
**HAVING** **COUNT**(**\***) **>** 1

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**

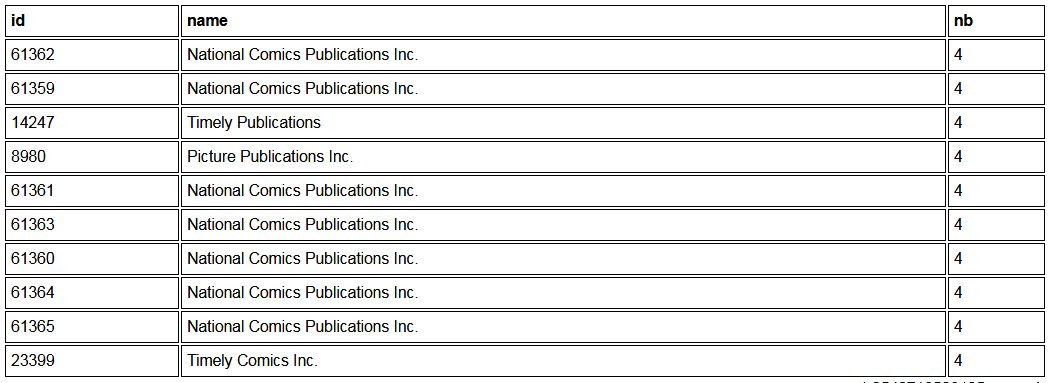
**S**.**publisher\_id** **=** **I**.**publisher\_id**

**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**

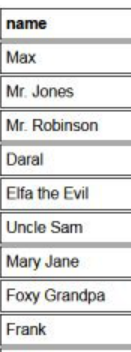
**I**.**indicia\_publisher\_id** **=** **IP**.**id**

**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 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**

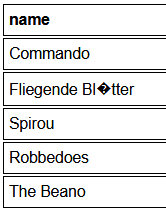
(**S**.**issue\_id** **=** **I**.**id** **OR** **I**.**id** **=** **S**.**issue\_id**) **AND**

(**IP**.**id** **=** **I**.**indicia\_publisher\_id** **OR** **I**.**indicia\_publisher\_id**) **AND**

**IP**.**name** **LIKE** '%Marvel%DC%'

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

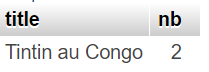
**ORDER BY** **T**.**inb** **DESC** **LIMIT** 5

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

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.

## 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**

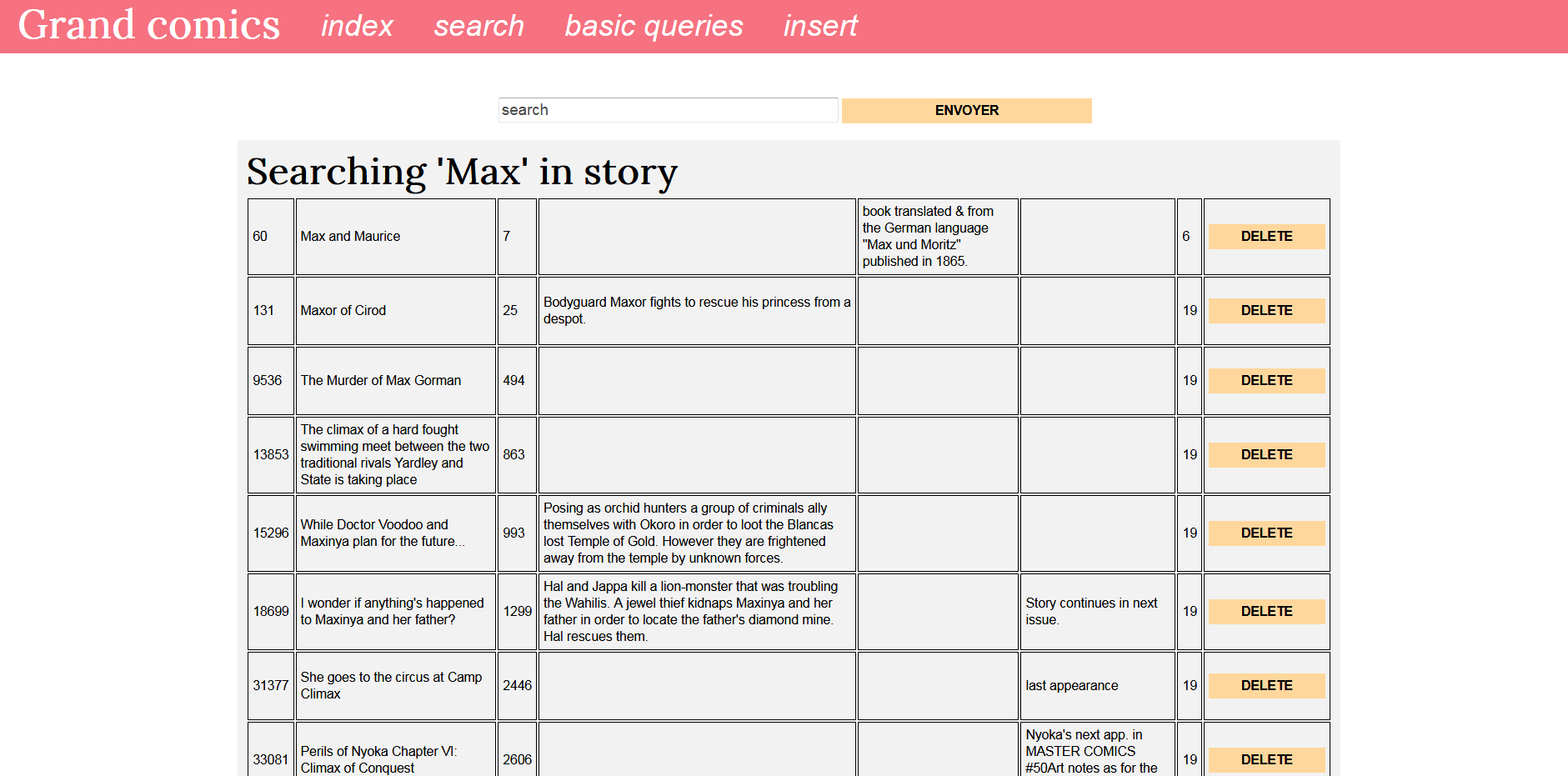
**GROUP BY** **BG**.**name**

**ORDER BY** ipn **DESC** **LIMIT** 10

# Interface

### Design logic Description

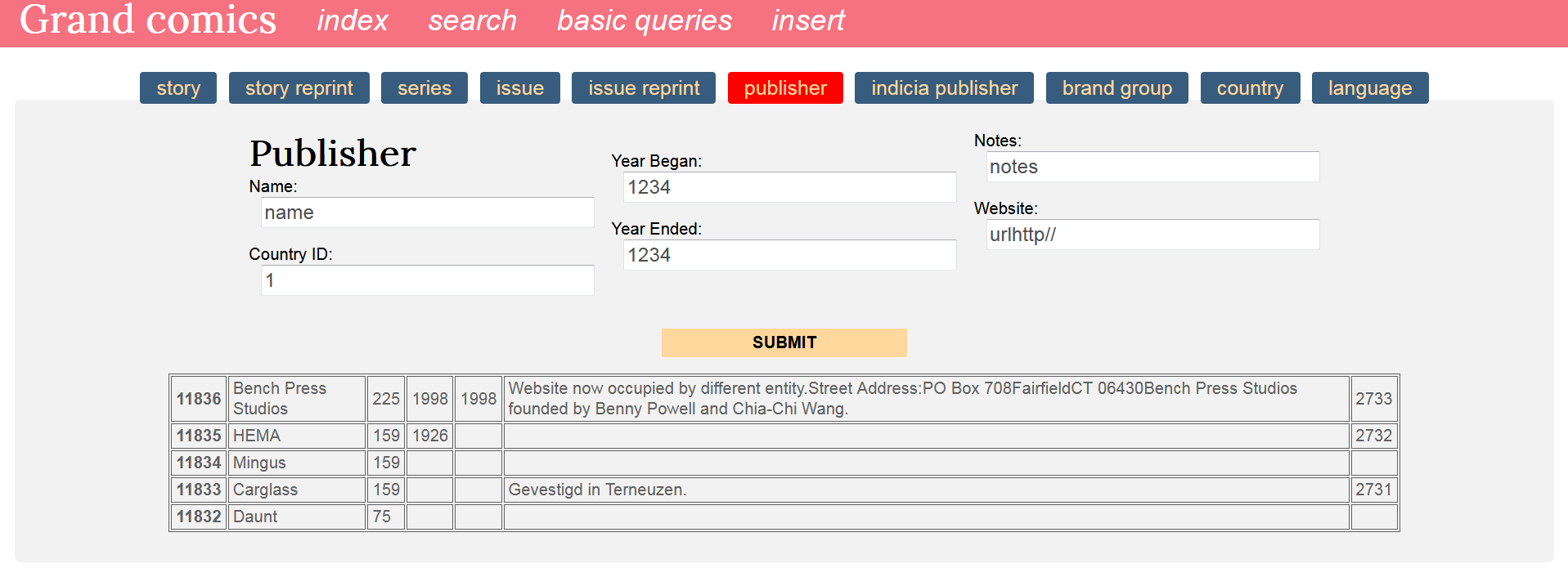
Our interface contains three main pages :



* **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.