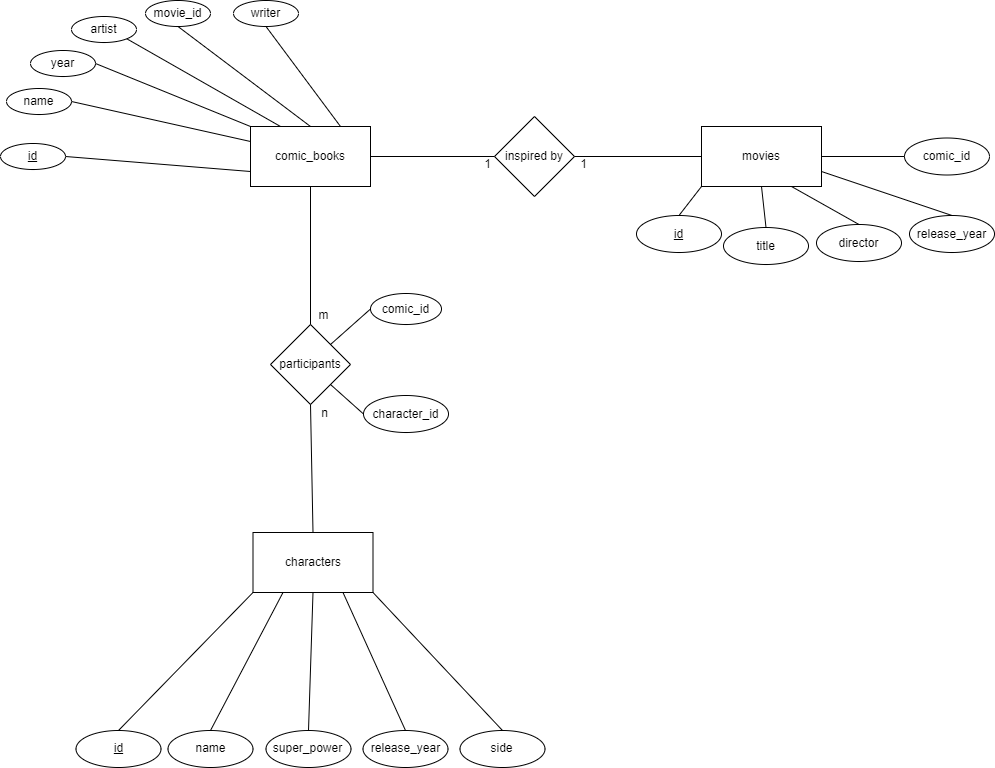
# Programming Assignment 2 Report

|  |  |
| --- | --- |
| Student(s): | Irmantas Venckus – [iv222dn@student.lnu.se](mailto:iv222dn@student.lnu.se)  Hussein Lakis – hl223fk@student.lnu.se |

## Project Idea

For our assignment we have designed a database, which stores information about comic books, its characters and the movies that are based on them. The targeted audience are the comic book/ movie fans which are interested in finding out more information about their favorite movie/comic book. Examples of usage: finding out how many movies does a particular hero has, finding out what kind of characters are mentioned in a comic book.

## Schema Design



The E/R diagram contains 3 main entities comic\_books, movies and characters. The main entity comic\_books is connected by 2 relationships. The first one is connection with characters it is connected by a relationship “participant”, which links each comic book with characters that are defined in it by their id numbers. The described relationship has a many to many relation, because a single comic book can contain multiple characters and a single character can be featured in multiple books. The second relationship is “inspired by” which is a one to one relationship. Though in reality it is a many to many relationship and we are aware of it, but since it is a limited database and our used examples do not contain these scenarios we decided to use a one to one relation.

because there can be multiple movies inspired by a single comic book and a single movie can be inspired by multiple comic books. Unlike the participants

## SQL Queries

Here you present and discuss the most interesting queries. Make sure you have 5 of them at least and check the specification in the assignment sheet. One example is found below:

1. **List the names of character names from a given answer which is comic book id.** The following query combines 3 tables (characters, comic\_books, participants). The table participants contain combinations of character id and comic book id, which indicate that a particular character is inside that comic book. Using this table (Participants), we can combine all 3 tables together. The given answer should be an id of a comic book, which will indicate a specific row. From that row we print only the name, which we need.

SELECT ch.name FROM participants p

INNER JOIN comic\_books co ON co.id = p.comic\_id

INNER JOIN characters ch ON ch.id = p.character\_id

WHERE p.comic\_id = answer

2. **Print the characters (having at least one comic book appearance) and their number of appearances.** For this query we combine 2 tables characters and participants based on their id. We know for a fact, that the participants table stores the ids of every character appearance once in a comic book. By using this information, we can count how many times a character has appeared in all our collected comic books. Since we are asking for 2 different values, which are not from identical rows we used group by character name.

SELECT ch.name, count(ch.id) FROM characters ch

INNER JOIN participants p ON ch.id = p.character\_id

GROUP BY ch.name

3. **List all the description of a superpower on the given answer, which is expected to be a name.** For this query we use a View, which gives us a table of character names and their superpowers. Using that, we match the answer, which should be a name and search for its row in the table and list its superpower.

CREATE VIEW allPowers AS SELECT name, super\_power FROM characters ch

SELECT super\_power FROM allPowers WHERE name = answer

4. **Print the comic books where the wanted character appears.** For this query we combine 3 tables (participants, comic\_books, characters) we do this exactly as we did in the 1 st query by using the middle table participants to link both tables. The final result gives us a table from which we can list all the comic book names according to the answer given by the user.

SELECT co.name FROM participants p

INNER JOIN comic\_books co ON co.id = p.comic\_id

INNER JOIN characters ch ON ch.id = p.character\_id

WHERE answer = p.character\_id

5. **Show what comic books some movies are inspired from.** This query combines 2 table (comic\_books and movies) by seraching a comic book id in the movies table. If a movie attribute comic\_id finds its match in the comic\_book table, we print both of their names.

SELECT m.title, co.name FROM movies m, comic\_books co

WHERE m.comic\_id = co.id

6. **Print the anticipated marvel characters for the movies in the database based on the comic books.** This query combines all 4 tables combining (characters, participants, comic\_books, movies) the characters, participants, comic\_books tables are combined exactly as in the previous examples and for the movies we used one of its atributes comic\_id to find its link. The final result prints out movies and what character is expected to be in.

SELECT m.title, ch.name FROM participants p

INNER JOIN characters ch ON p.character\_id = ch.id

INNER JOIN comic\_books cb ON cb.id = p.comic\_id

INNER JOIN movies m ON m.comic\_id = cb.id

7. **Show the average release year and the number of characters of each side.** For this query we print out all the existing side names in the table using “GROUP BY side”, calculate their average release using AVG year and count how many characters there are on that side using COUNT.

SELECT side, AVG(release\_year), COUNT(side) FROM characters GROUP BY side

## Discussion and Resources

Here you can write anything you might think it is important and provide the link to the required resources. For example:

One issue presented when trying to implement queries is the using the argument *Writer* in the relation Comic books when the comic books would have two writers instead of one. For example some tuples have Stan Lee and another writer as the value Stan Lee, Larry Lieber, another would have Stan Lee, Steve Dikto and another would only have Stan Lee as the writer. If we tried to execute SELECT name FROM comic\_books WHERE writer = “Stan Lee”, we wouldn’t have 3 comic books printed out but only 1.

Another problem is the relationship between the relations Comic\_books and Movies. Here it is presented as one–to–one but in reality, it should be many–to–many since many movies can be inspired by many different comic books. We are aware of this, but since our database did not contain alot of data and we did not want to add a 5th table we kept it simple and left it as a one to one.

Source code: [https://github.com/HealinGhost/comicbooks]

Video demonstration: [https://youtu.be/\_2Brm4bS4us]

# Changelog

|  |  |
| --- | --- |
| Person | Task |
| Irmantas | Created Git repository |
| Irmantas | Wrote the idea worked for the report |
| Hussein | Inserted Data in csv files (manually) |
| Irmantas | Created Database and inserted tables with the data |
| Hussein | Came up with queries |
| Hussein | Created GUI |
| Hussein–Irmantas | Video montage |