Assignment 3: The Relational Data Model and SQL

Jelle Hellings

3DB3: Databases – Fall 2021

Department of Computing and Software
McMaster University

Deadline: October 29, 2021

Cheating and plagiarism. This assignment is an *individual* assignment: do not submit work of others. All parts of your submission *must* be your own work and be based on your own ideas and conclusions. If you *submit* work, then you are certifying that you have completed the work for that assignment by yourself. By submitting work, you agree to automated and manual plagiarism checking of the submitted work.

Cheating and plagiarism are serious academic offenses. All cases of academic dishonesty will be handled in accordance with the Academic Integrity Policy via the Office of Academic Integrity.

Late submission policy. There is a late penalty of 20% on the score per day after the deadline. Submissions five days (or later) after the deadline are not accepted. Do not wait until the deadline to ask questions or raise problems.

Description

Our local cinema chain recently introduced a *subscription service* that gives regular visitors the option to watch a film at their cinema whenever they want for a fixed price per month. To promote this new service, an online community consultant proposed to expand the cinema website with elements that encourage *engagement*. To do so, the consultant proposed to build a social online environment into the cinema ordering website such that subscribers can review and discuss films and make friendships with other subscribers. The consultant already put in some legwork by providing an ER-model for this social online environment and a relational schema to store basic film information.

The consultant provided a two-part design. The first part describes the social media features and the second part describes a high-level sketch of the information maintained per film (which the consultant figured would be of use for other parts of the website as well).

Part one: The social media features

The social media features center around its users (the subscribers). Users have usernames, which are not necessary unique. To distinguish users with the same username, the consultant proposed to assign each such user a different number. Users will log in using their unique email address and a *password*. According to the consultant, the password should not be stored in plain text, but as a pair of a 512-bit (64-byte) hash and a 64-byte salt value. Together, the hash and salt are sufficient to determine whether a login attempt provided the right password, this without providing easy access to the stored password (even if the system gets compromised).

The social media features break down in three types:

- Users can *befriend* other users. If a user X indicates that Y is a friend, then this indicates that X follows the activities of Y, but this does not imply that Y is also a friend of X.
- Users can *review* films. Each review assigns a score to the film and reviews can be revised with more information. Each review can optionally have a *video* component (e.g., as in a vlog) and optionally have a *text* component (e.g., as in a blog).
- Users can place *reactions* on reviews and on other reactions (a reaction is either on a single review or on a single reaction, but not on both).

The ER-Diagram for the social media feature can be found in Figure 1.

Part two: The film information

The consultant sketched the following layout of the database maintaining film information, but indicated that some details still need to be figured out:

- A *table* **person**(<u>id</u>, name, birthdate$_{\text{optional}}$).
  This table provides basic information on people that work on films.
- A *table* **film**(<u>title</u>, <u>year</u>, <u>creator</u>, duration, budget).
  This table represent films. The consultant figured out that films can have the same title and be released in the same year (e.g., in 2005 there was a film *Chaos* directed by Tony Giglio and another film *Chaos* directed by David DeFalco—The consultant didn't see either of them, however). To distinguish between films with the same title and made in the same year, the consultant proposed to use the main director (*creator*, which refers to the id column in the **person** table) of the film.
- A *view* **film_info** to easily retrieve a film (columns title, year, duration, and budget) together with the name of its creator (column creator_name).
- The consultant wants a table **role** that stores, per film, all people that worked on the film and on the role these people had (e.g., actor, writer, director, producer, costume, casting, editor, or makeup). The consultant had difficulties coming up with a proper design for this table, but noted that people can have several roles in the same film.
- A *constraint* that all film creators (mentioned in the **film** table) have the role of director for that film (as recorded in the **role** table).

Assignment

The goal of the assignment is to translate the above description into proper SQL statements that create tables and set up all required constraints. To do so, you proceed in two steps. First, you write a report in which you step-by-step translate the ER diagram of part one (Figure 1) into a relational schema and complete the details of the relational schema of part two. Next, you provide the translation of your relational schema into a sequence of SQL statements that will set up the database. Your submission:

1. must be two files: a `PDF` file with the report and a plain-text file (`txt`) with the SQL statements to set up the database;
2. must include (in the report) a clear description of the steps taken to translate the ER diagram of part one to SQL tables, each choice made during this translation, and each constraint present;
3. must use the constructs presented on the slides or in the book (we do *not* allow any system-specific constructs that are not standard SQL);

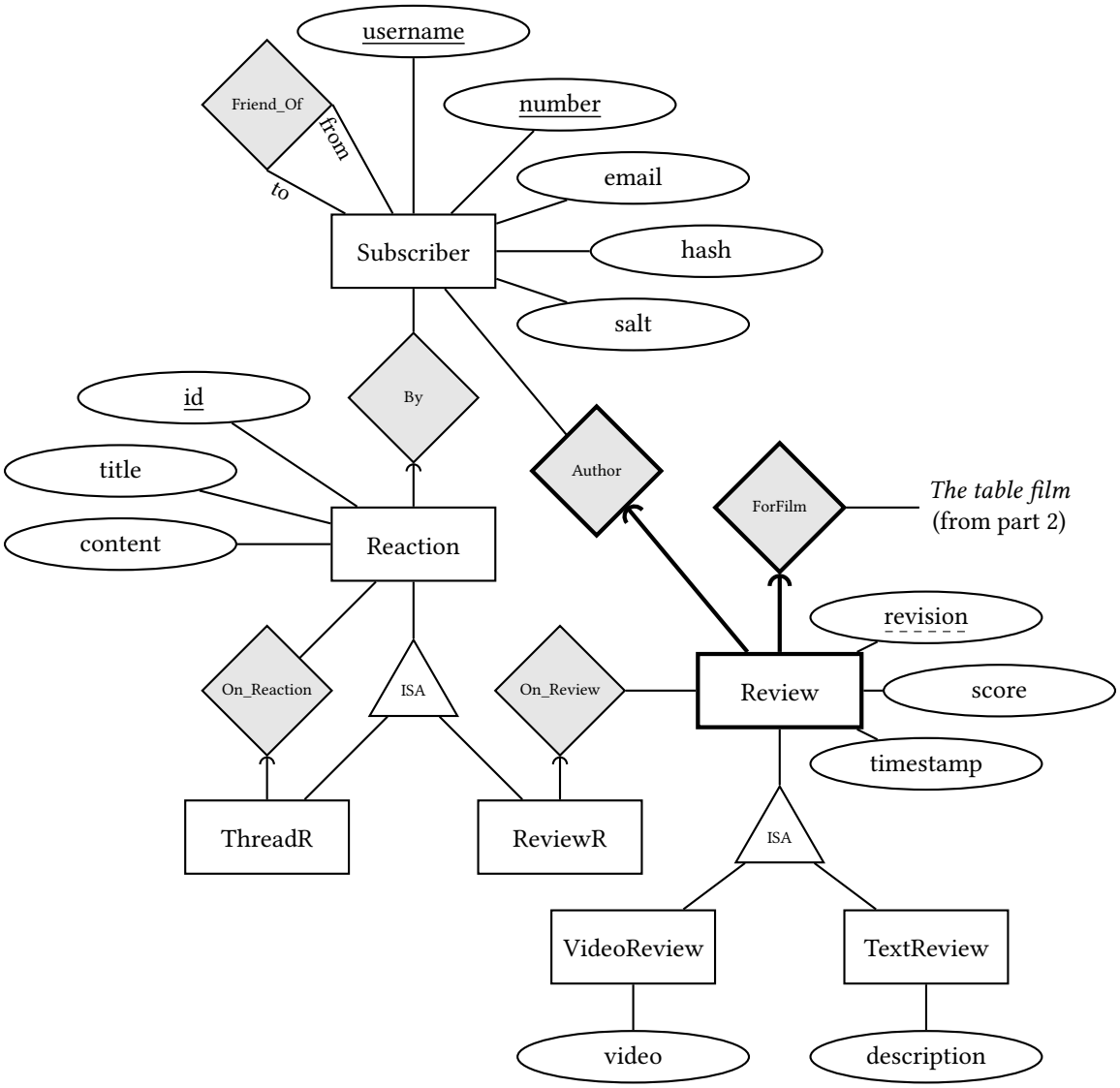

Figure 1: ER-Diagram for the social media features.

4. must work when using DB2 (on `db2srv2`): test this yourself!

Submissions that do not follow the above requirements will get a grade of zero.

As multi-table checks via **CHECK**s and **ASSERTATION**s are not supported by DB2 or any other major DBMS, you do not need to use these constructs in your solution. Make note in your report of all constraints present in the description that you cannot express via the constraint types we have seen for SQL or which can only be expressed via multi-table **CHECK**s and **ASSERTATION**s.

Grading

The final grade will be split between part one (75% of the grade) and part two (25%). While evaluating your work, we will look at:

completeness Does your solution contain all tables and constraints described in the requirements?

correctness Does your solution use the correct SQL syntax and do you take the right decisions in your translations? Are all included constraints correct? Do all excluded constraints have a proper motivation?

presentation Is the report readable? Does the report properly motivate the choices made while translating toward SQL?

The maximum grade for part one is equally determined by the quality of your translation of each of the entities and relationships, whereas the grade for part two is equally determined by the quality of the translation of the five items. Every SQL error will result in a reduction of the overall grade (with the lowest possible grade being zero).