Assignment: The Relational Data Model and SQL

COMPSCI 2DB3: Databases–Winter 2023

Deadline: February 19

Department of Computing and Software
McMaster University

Please read the *Course Outline* for the general policies related to assignments.

Plagiarism is a *serious academic offense* and will be handled accordingly.
All suspicions will be reported to the *Office of Academic Integrity*
(in accordance with the Academic Integrity Policy).

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. Only *discuss or share* any parts of your submissions with your TA or instructor. You are *responsible for protecting* your work: you are strongly advised to password-protect and lock your electronic devices (e.g., laptop) and to not share your logins with partners or friends! 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 all submitted work.

Late submission policy. Late submissions will receive a late penalty of 20% on the score per day late (with a five hour grace period on the first day, e.g., to deal with technical issues) and submissions five days (or more) past the due date are not accepted. In case of technical issues while submitting, contact the instructor *before* the deadline.

Description

The local artists have been further working on parts of the art library proposal. Their focus has mainly been on the subscription services and on providing a donation system. With respect to the subscription services, the main goal is to generate sufficient revenue to support the art library long term while also providing support for local artists and for art projects in the local community (e.g., arts-in-school engagement).

To develop these features, the artists already contacted a consultant that put in some legwork. In specific, the consultant provided a two-part design. The first part describes the subscription services via a documented E-model. The second part describes a high-level sketch of the relational schema necessary to support the donation system.

Part one: The subscription services

The proposed subscription model supports two disjoint types of membership accounts:

1. a personal account held by a single private person (e.g., a community member interested in art); or
2. an organization account (e.g., held by a local company or not-for-profit organization).

Organization accounts are managed by one-or-more *representatives* (with personal accounts) that can manage the organization account (e.g., borrow art pieces in the name of the organization). Furthermore, organization accounts have a *type* (e.g., company, local government, not-for-profit, etc.).

Members can log in to their account via their unique e-mail address and a password. According to the consultant, the password should be stored as a hashed value that incorporates a salt value.

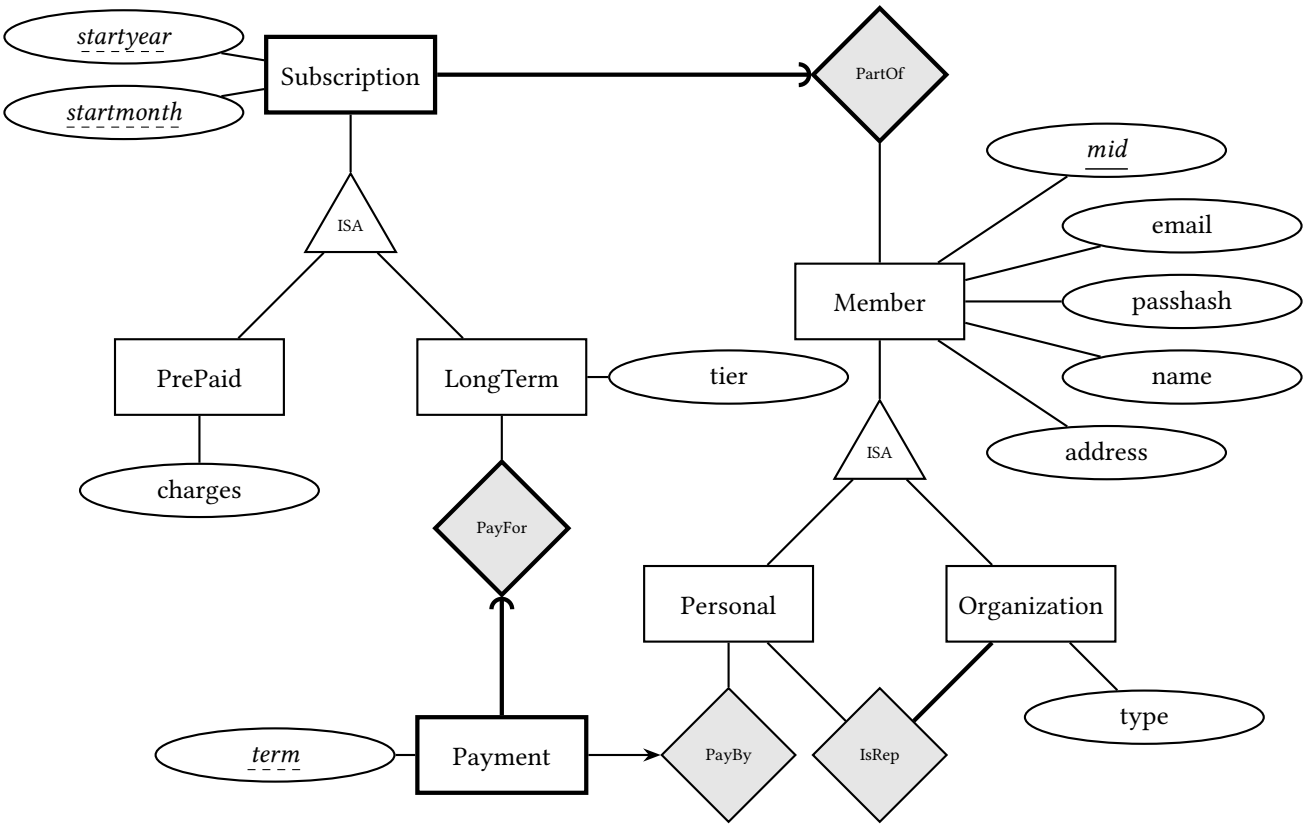

Figure 1: ER-Diagram for the subscription features.

Each account can hold a single subscription at any single time. That subscription can be a *pre-paid subscription* that allows the account to borrow a fixed number of art pieces or a long-term subscription. Long-term subscriptions belong to a tier that determines the subscription cost and how many art pieces can be borrowed simultaneously. The consultant noted that the diagram does not enforce that an account can only hold a single long-term subscription at any single time. For long-term subscriptions, a payment history covering each payment installment is maintained via the *payment terms* that have been paid (term one representing the first month of the subscription, term two the second month, and so on). For each payment term, the paying account is also stored in case of a organization account (to keep track which representative paid). In addition, the system maintains a history of all subscriptions held by an account.

The ER-Diagram for the subscription features can be found in Figure 1.

Part two: The donation system

The consultant sketched the following layout of the database maintaining donor events and donations, but indicated that some details still need to be figured out:

- A *table* **DonorEvent**(*eid*, name, date, description, private).
  This table provides basic information on donor events: their name, date, and description. Donor events will be listed on the library website if the event is not private.
- A *table* **DonorInvite**(*eid*, *mid*, confirmed, amount, $\text{referredBy}_{\text{optional}}$)
  This table provides direct invites to members (identified by the member identifier *mid* of the Member

entity, you can assume there is a table `member` with primary key *mid*). Each direct invite of a member *m*, comes with an *amount* of people they can refer. In specific, a direct-invite member *m* can refer another member *o*. In that case, the invite of the other member *o* will refer to the member identifier *m* via the optional *referredBy* attribute (which must imply that the invite for member *o* for event *e* refers to an invite of member *m* for event *e*). The other member *o* cannot refer other members.

- A *view* **Referrals**(eid, mid, memberName, rid, referredName) to easily retrieve, for each event with identifier *eid*, the list of direct invite members and the members they invited.
- The consultant wants a table **donation** that stores, per donation, the event during which the donation was given, the member that donated, the amount donated, the person to which the donation is attributed (optional), and an attribution message (optional). The consultant had difficulties coming up with a proper design for this table, but noted that people can make several donations during an event.
- A *constraint* that the referrals for an event by a direct-invite member *m* do not exceed the amount of referrals allocated to *m* (as recorded in the **DonorInvite** 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 SQL constructs that are not on the slides or in the book);
4. must work when using IBM Db2 (on `cs2db3.cas.mcmaster.ca`): 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).