Assignment: The Relational Data Model and SQL

COMPSCI 2DB3: Databases–Winter 2024

Deadline: February 26, 2024

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 are aware of the *Plagiarism and Academic Dishonesty* policy of this course outlined in this section, that you are aware of the Academic Integrity Policy, and that you have completed the submitted work entirely yourself. Furthermore, 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

Local farmers are discussing setting up an online co-op farmers market to make healthy in-season produce directly available to consumers for fair prices for both consumers and the farmers. In addition, the farmers market should provide a review system for produce. To figure out the requirements of building such a system, the farmers have contacted a consultant that provided some initial design on the data model of the online market and review system.

Part one: The online market

To enable a farmers market, the farmers are looking into setting up a digital market place.

In this marketplace, we have two types of users: *consumers* and *producers*. Producers can also partake in the system as consumers (of produce of other producers). All users can log in to their account via their unique e-mail address and a password. Consumers have a delivery address. Producers have a public seller name and need to be vetted by the co-op to assure that only legitimate farmers are providing their produce. In addition, producers can put up offers for produce. Producers can place active orders every week of the year and offers expire after a week. Consumers can place orders in which they buy one-or-more offers at a given date (which should fall within the week in which all parts of the sale are active).

The ER-Diagram for the online market can be found in Figure 1.

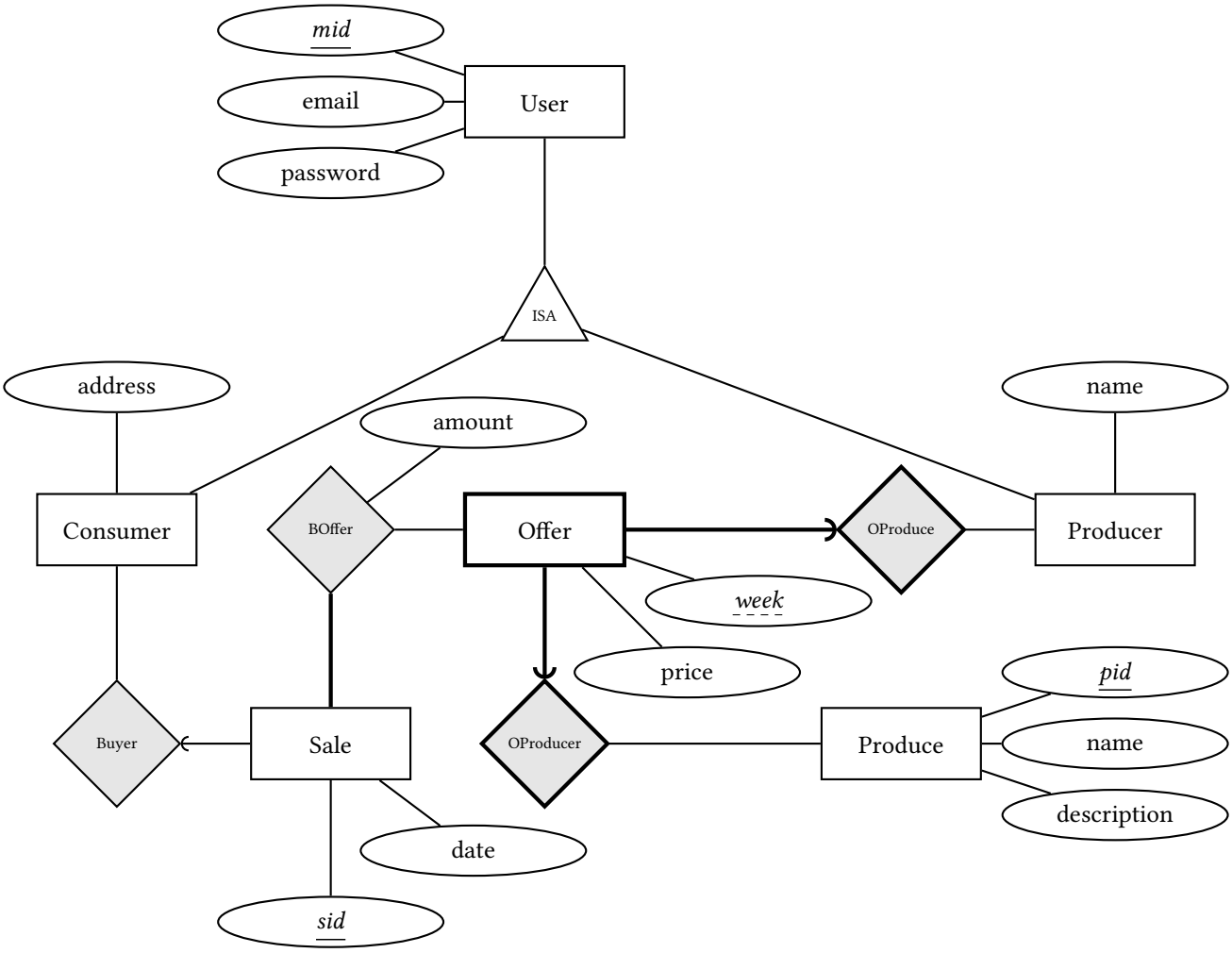

Figure 1: ER-Diagram for the online market.

Part two: The review system

In addition, users can review produce they have ordered before. Reviews can be either short form (a score) or long form (a score with a description). The consultant sketched the following layout of the database maintaining these reviews, but indicated that some details still need to be figured out:

- A *table* **Review**(*rid*, sid, uid, score, date, $\text{description}_{\text{optional}}$).
  This table provides a basic review with identifier *rid*. The review is on the sale identified by sale identifier *sid* and is written by the user with user identifier *uid* (which can be either the consumer or a producer that partook in the sale). Every review has a score and a date of writing. In addition, reviews can optionally have a long-form description.
- A *table* **ReviewVote**(*rid*, *uid*, voteType)
  This table provides votes on the usefulness of reviews. All users (consumers and producers) can vote whether a review is helpful or not. In this table, *rid* refers to a review, *uid* refers to a user, and *voteType* can represent an upvote, neutral, or a downvote.
- A *view* **SellerScore**(pid, score, num_reviews) to easily retrieve a summary of all reviews for a producer: the summary should contain the *average* score the sales of the producer received and the number of reviews the sales of the producer have received. For this review, we only consider reviews on sales written by the consumer that placed the sale.
- The consultant wants a table **comment** that can store user content for produce. Each comment should have a title (e.g., "great for summer salads") and a description (e.g., a summer-salad recipe). Each comment is written by a user on a given date and a user can write multiple comments. Users can not only put comments directly on produce, but also react on the comments of other users.
- A *constraint* that reviews are *only* written by either the consumer or a producer that partook in the sale that is reviewed.

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 your student number and MacID in both the `PDF` and the plain-text file;
3. 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;
4. 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);
5. 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).