CS5200: Homework 3: Physical Modeling and SQL

Fall 2024

1 Overview

This assignment asks you to construct physical models (i.e., SQL tables) given a logical model, and to write SQL queries based on those physical models. By completing this assignment, you will practice the following skills:

- Reading and understanding a logical model
- Constructing a physical model from a logical model
- Using `CREATE TABLE` statements to construct tables with primary keys, default values, and data-integrity constraints
- Writing SQL queries for given table definitions

For this assignment, you should submit a single .zip file, containing two .sql files, orders.sql and students.sql, that contain your answers for part I and part II, respectively. For each part, you will begin by defining the physical model (using `CREATE TABLE` statements), and then write a number of SQL queries.

For full credit, the grader should be able to open your file in MySQL Workbench and execute it with no errors.

If you find it helpful in testing your queries, you may add records to the database beyond those explicitly requested below, but this is not required. If you do this, please add comments to indicate to the graders that the records are for testing and not part of your answer to any of the questions below.

PLEASE NOTE: this assignment is due at the beginning of class on Wednesday, Oct 16, and we will *not* accept late work for this assignment. This allows us to discuss the answers to these questions during the midterm review in class on the 16th and 17th.

YOU MAY NOT DISABLE MySQL's CONSTRAINT CHECKING FOR THIS ASSIGNMENT. IF YOU DO, WE WILL DEDUCT 50 POINTS FROM YOUR GRADE. (If you don't know what this means or how to disable constraint checking, you don't need to worry; it's not the kind of thing you're likely to do accidentally.) Our justification for the somewhat draconian penalty for disabling constraint checking: while it is possible to disable constraint checking in MySQL, this is a global setting—it affects all schemas running on the server and all .sql files. Therefore, if any submission disables constraint checking, it could affect the grading of other students' assignments as well.

Within each SQL file, delete and recreate the schema specified in each problem, and use SQL comments to label each query with the corresponding question. In MySQL, `--␣` (that is, 2 hyphens and a following space) introduces a line comment. For example, orders.sql might look something like

```
DROP SCHEMA IF EXISTS HW3_Orders;
CREATE SCHEMA HW3_Orders;
USE HW3_Orders;

-- CREATE TABLE statements go here
```

```
-- Problem 1:
SELECT ...;

-- Problem 2:
SELECT ...;
```

2 Questions

2.1 Part I: Orders

Your solution for this part should create and use a schema named `HW3_Orders`.

Write `CREATE TABLE` statements for each relation in the logical model in figure 1 on page 3, including primary key constraints, foreign key constraints, `AUTO_INCREMENT` and `NOT NULL` constraints where appropriate, and choosing appropriate data types for each attribute. Your constraints should reflect the cardinality and modality annotations on the diagram in figure 1.

For this model, attributes whose names end in "Number" or "ID" should be integers; these are generally artificial primary keys (or foreign keys). Attributes whose names end in "Rate" or "Cost" should be fractional numbers (like 0.0825 or 19.95), and quantities should be integers. Attributes whose names end in "Code" should be strings.

Then, using the physical model you created above, write SQL statements to perform the following operations or answer the following questions:

1. Add a new customer to the database: Robert Mitchell, 123 Main St, Anytown, IL, 51236. You will also need to add corresponding records to the StateCodes table to ensure that the referential integrity constraints are met.
2. List each order number, customer name, city, state, and zip code, in order by date (earliest orders first). Your query should not include any columns other than those specified.
3. For each customer, compute the total amount that customer has spent. Your query's result should have 3 and only 3 columns: customerID, customer name, and total amount spent.
4. For each item in the product catalog (i.e., the Item table), list the number of customers who have ordered that item at least once. Your query's result should have 2 and only 2 columns: item number and the number of customers who have ordered that item. (If no customer has ordered a particular item, that item should not appear in your results.)
5. For each customer zipcode and state, list the total order value. Include summary rows for each state. Your query's result should have 3 and only 3 columns: zipcode, state, and total order value. (Include only those zipcodes and states for which we have orders.)

2.2 Part II: Students

Your solution for this part should create and use a schema named `HW3_Students`.

First, write `CREATE TABLE` statements for each relation in the logical model in figure 2, on page 4. Note that the figure is only *part* of the data model you should have submitted for HW2; I've simplified and reduced it somewhat. In this diagram, attribute names in parentheses are considered to be optional (i.e., can be `NULL`); all other attributes are required.

Your table definitions must include primary key constraints, foreign key constraints, `AUTO_INCREMENT` and `NOT NULL` constraints where appropriate, and you must choose appropriate data types for each attribute. Attributes whose names end in "number"and "ID" are integer artificial keys; grades should be integers, and department names should be strings. A course catalog number should be a string, such as `'CS5200'`.

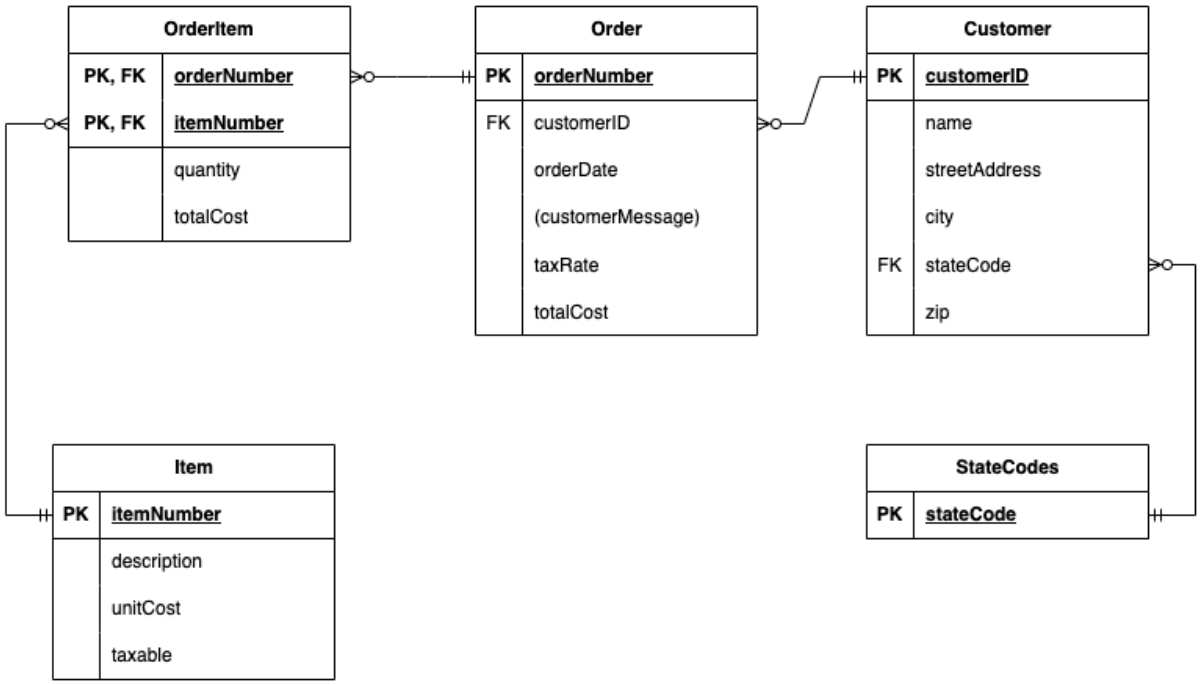

Figure 1: Orders logical model

Then, write SQL statements that perform the following actions or answer the following questions:

1. Insert records into the database to indicate that student John Smith has enrolled in the university with ID #2475343 and an enrollment date of September 1, 2023.
2. Register the student from the previous question for CS5200; this student does not yet have a grade for this course. You will need to insert additional records into other tables to ensure that the relational integrity constraints remain valid.
3. List all students (ID#, first name, last name) enrolled in CS5010.
4. For each student, compute their average grade for all classes. Expected columns: studentID, average.
5. For each faculty member in the 'English' department, count the number of messages they have sent. Expected columns: idNumber, firstName, lastName, numberOfMessages.
6. List the 5 courses with the largest enrollments. Expected columns: catalogNumber, numberOfStudents.
7. List all students who have registered for a course but not yet received a grade in that course. Expected columns: studentID, firstName, lastName, catalogNumber.
8. For each faculty member, list the number of courses they are teaching. Expected columns: idNumber, firstName, lastName, numberOfCourses. If a faculty member is not currently teaching any courses, their name should appear in the result with numberOfCourses = 0.

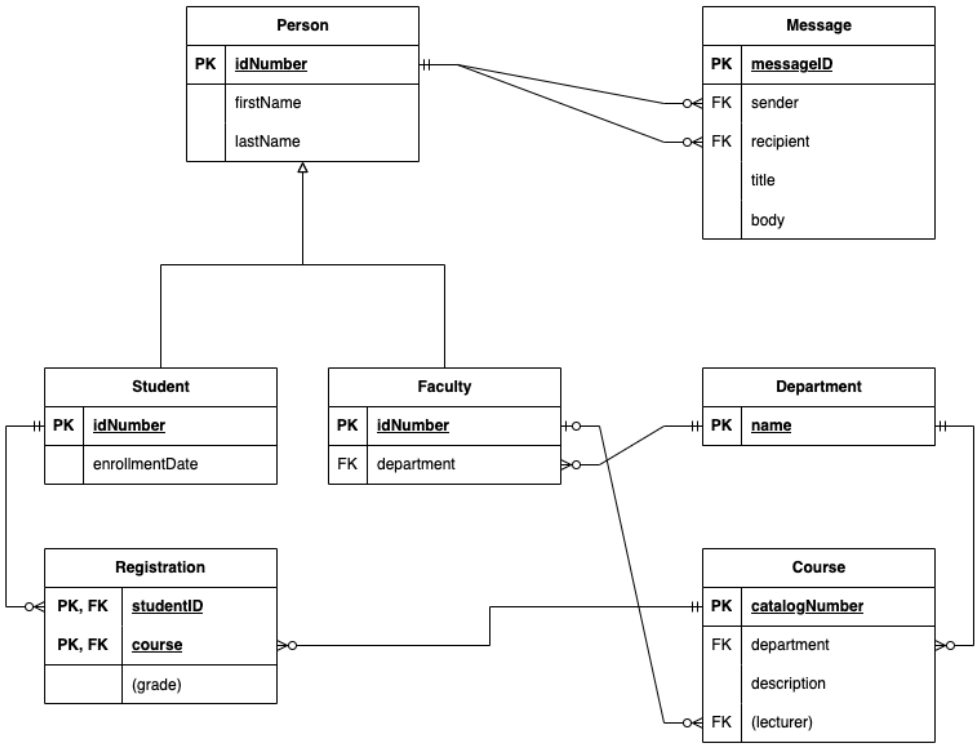

Figure 2: Students logical model