The Database Life-Cycle

Team Project Due: Friday, Dec 11, 5pm

Project Synopsis

Every so often in the evolution of technology a novel system is introduced that radically changes the way business is done. Two such disruptive systems that emerged in relatively recently are Airbnb and Uber. In this project your team will choose one of these two businesses and model a portion of the data model needed to support the functioning of the business. Starting from user-stories, you will develop a domain model. From the domain model you will derive relations which will be normalized to a schema in BCNF. You will implement the schema using SQL's DDL. With a combination of SQL's DML and Python (via psycopg2) you will complete the system design by implementing transactions that realize the user stories.

Project Steps

1. Choose between Airbnb or Uber [5 points]

Research and form a high level perspective of the business model, functioning, and revenue stream of either Airbnb or Uber. Take time studying the origin, current functioning and competitors to both of these businesses before making the choice of which of these two companies you would like to model for this project.

Write a brief paragraph (2-5 sentences) discussing the core mission of the business.

2. Identify 3 Users and 10 User Stories [30 points]

By definition a stakeholder is a *role* who care about the success of the system. For example, an Airbnb stakeholder would be the owner of the rental property. For Uber a stakeholder would be the person requesting transportation. While a business normally has numerous stakeholders, you want to identify three user stakeholders. Users directly interact with the core database i.e., the database is needed to meet their main business goal (request transportation, list a house for rent housing etc.). An insurer would be a stakeholder but does not participate in what would normally be viewed as the core business.

Enumerate the 3 types of users you have identified. Give a 1-2 word description of the role (e.g., driver), followed by short description (1-2 sentences) of the role e.g., who exactly is an "driver" in your terminology.

Given the 3 users you have identified, list at least 10 user stories which represent their desired functional requirements. Make sure each type of user (role) has at least 2 user stories associated with them. This accounts for 6 user stories. You can associate the remaining 4 user stories with any user you wish. Create a table of user stories along the below format. Even though the <reason> for desiring to attain the <goal> may sometimes be obvious, it is a good thought exercise to explicitly list the reason for goal attainment.

As a <role>	I want <goal>	So that <reason>

Each of these user stories will be realized by a collection of one or more DML statements with the database (a transaction). Of the 10 user stories you have identified at <u>least 3 should be non-trivial</u> i.e., will require multiple interactions with the database.

3. The Conceptual Model [60 points]

The 10 user stories (which are realized as SQL DML statements—SELECT, INSERT, UPDATE etc.) drive the architecture of the database. Naturally all the information required to realize the user stories need to be in the database. Identify the domain entities, their attributes, and relationships between the entities. Using UML draw a conceptual model that represents this subset of the real domain of Airbnb or Uber. Recall that these entities + attributes + associations will be used to realize the user stories. Hence during this design phase you may iterate between steps 2 and 3 a few times aligning the needs of the user stories with the data supported by the domain model. As with the user-stories, make the domain model realistic and meaningful.

Your domain model should have at least 3 association classes and 2 generalizations.

4. Relational Schema [20 points]

Convert the conceptual model into a relational database schema. For each entity and relation in the conceptual model write the equivalent relation. Note that when writing a relation you only need to indicate the relation name and it's attributes (no data types or other constraints). Indicate the primary key by underlining it and the foreign key by a double underline. You now have the first draft of your relational schema.

We've discussed in class how association classes, under certain conditions of multiplicities of the association (i.e., when one end of the association is either 1 or 0..1) can be absorbed into one (or both) of the participating classes. If you have such associations and association classes, show them in the conceptual model but indicate how this simple optimization has been done in the relational schema. You now have the second draft of your relational schema.

5. Functional Dependencies [20 points]

For each relation you obtain from 4, specify the functional dependencies (FDs). Given the FDs + Relation indicate what normal form each relation satisfies.

6. Normalization [20 points]

Unless your initial design is trivial, it is highly likely that your first draft of the relational schema will have relations that are in 1NF or 2NF. Decompose all relations so that they, and consequentially the full schema, are in BCNF. This is your final relational schema (which started with step 4).

7. Setup the DB [50 points]

Call your database `db_project`. Write two SQL scripts `create.sql` and `initialize.sql` which when executed will create all the tables and populate them by reading values from corresponding CSV files. Do not hard code the initial values in INSERT statements. Note that you are initializing the DB by connecting the various tables via foreign keys. You may find it useful to draw pictures of instantiated tables as you connect them with concrete values of primary and foreign keys. Load each table with 5-10 records.

Write a SQL script `show_all.sql` which when executed will show the contents of all the tables in your database.

8. Queries 70+60 = 130 points

For each of the user stories, write a corresponding query that realizes that user-story. You would have identified 10 user stories of which 3 are non-trivial (requires more than 1 interaction with the database). Produce the following:

`simple_queries.sql` - will execute 7 queries

`complex_query_1.py`

```
complex_query_2.py
complex_query_3.py
```

Recall that a complex query, by our definition, requires multiple DML statements where SQL interacts with Python. As with assignment 4, we test each complex query separately with the equivalent of

```
% python complex_query_1.py  postgres
```

These Python scripts are for demonstration purposes. Hence it is acceptable to hard code values in the Python scripts to demonstrate the idea.

9. Overall quality of the deliverables [20 points]

Professionalism, clarity and effort matters. Ensure your work is organized according to the requirements (above) and the report is well formatted, spell/grammar checked etc. Your SQL and Python scripts need to be organized and commented.

Adding up the above points we get 355 points which will be scaled to the weight of the project 20%.

10. Peer evaluation. [+/- ½ grade point]

Each team member will submit an assessment of what their contributions and team members contributions to the to the project were. The peer evaluation will be kept confidential by the instructor.

Deliverables, Rubric and Grading policy

Each component of the final deliverable has been tagged with point values. The below check list will help to ensure all components are accounted for.

- Report in PDF
  - ✓ Team# + Names of all Team members
  - ✓ Chosen Project's (Airbnb or Uber) core mission
  - ✓ 3 stakeholders + table of 10 user stories
  - ✓ Diagram of initial conceptual model in UML
  - ✓ Relational schema
  - ✓ Functional dependencies
  - ✓ Normalized schema

- SQL scripts
  - ✓ create.sql
  - ✓ initialize.sql
  - ✓ show_all.sql
  - ✓ simple_queries.sql

- Python scripts for complex queries
  - ✓ complex_query_1.py
  - ✓ complex_query_2.py
  - ✓ complex_query_3.py

- Peer evaluation [+/- ½ grade point]

As with the assignments in this class, SQL scripts will be tested along the following lines:

```
% psql –U postgres –f create.sql
```

and Python scripts along the lines of:

```
% python complex_query_1.py postgres
```

Zip the pdf report + SQL queries + Python script and submit the zip bundle to the Blackboard assignment box. Each time will only submit one zip bundle with the name of their team e.g., `J3.zip` or `S11.zip`.

Each class member will submit their peer evaluations in a separate assignment box.