CS 121 Final Project Reflection

Due: March 16th, 11:59PM PST

Note: This specification is currently for 22wi, subject to minor changes for 23wi.

This reflection document will include written portions of the Final Project. Submit this as **`reflection.pdf`** with the rest of the required files for your submission on CodePost.

For ease, we have highlighted any parts which require answers in **blue**.

Student name(s):

Sahil Jain, Eric Han

Student email(s):

sjain3@caltech.edu, eyhan@caltech.edu

Part L0. Introduction

Answer the following questions to introduce us to your database and application; you may pull anything relevant from your Project Proposal and/or README.

Database/Application Overview

What application did you design and implement? What was the motivation for your application? What was the dataset and rationale behind finding your dataset?

Database and Application Overview Answer (3-4 sentences) :

Our idea is to develop a client application for interacting with data on the on-time performance of various airlines/flights. Specifically, such an application can be invoked to get information about delay/cancellation patterns on flights based on weather conditions, day of the month, time of the day, seasons etc. Based on this collected information and historical patterns, the application could also provide users with valuable information on which airline has the least delay, what origin/destination pair would cause the most delay to help a flier plan their flights to minimize chances of delay/cancellation. However, it could also be used to get more general information about the airplane industry, such as the routes airlines tend to fly the most, how many routes are included for each airline, etc.

Data set (general or specific) Answer:

We found the data on the github link provided (https://github.com/awesomedata/awesome-public-datasets). The specific link for the airline on-time performance dataset is https://community.amstat.org/jointscsg-section/dataexpo/dataexpo2009. This dataset, although

large, is very thoughtfully organized, with a separate spreadsheet for each year of data(from 1987 to 2008), in addition to extra spreadsheets for more specific details about carriers, airports, plane data, and variable descriptions. With all of this data, we can not only provide a lot of useful queried information to potential users about flight cancellations, but also make relatively accurate predictions and recommendations on what flights to take based on historical patterns.

Due to the extreme size of this dataset, we decided to only use data from January and February 2008. After additional cleanup to remove duplicates/extraneous records, we ended up with 993 total routes in the database and dataset. As was the case in assignment 3, a separate csv file was made for the data in each table of the database.

***Client user(s) Answer*:**

Passengers, who want information about general flight trends in terms of what seasons/time of day/day of the week might be best for minimizing delays. They could also request more general information about what airports are more crowded, what routes are busiest etc. However, since they cannot actually add/change/delete routes or flights, **passengers will only query data, not submit requests to insert/update data**.

Admin user(s) Answer:

The admin will be the central research group/company collecting the information on flight delays and trends across multiple airlines and years. They will have significantly more permissions than the clients, being able to approve requests and modify table structures/add table attributes as need-be, in addition to querying and inserting/updating/deleting information.

Part A. ER Diagrams

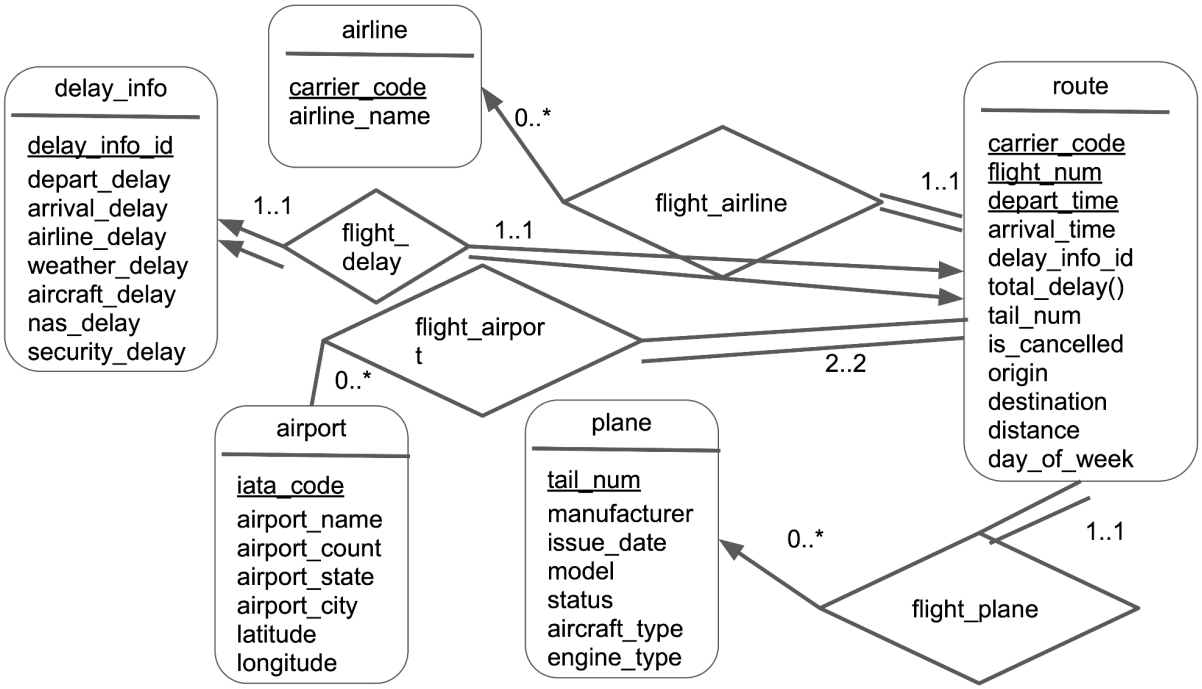
As we've practiced these past few weeks, the ER model is essential for designing your database application, and we expect you to iterate upon your design as you work through the ER and implementation steps. In this answer, you should provide a full ER diagram of your system. Your grade will be based on correct representation of the ER model as well as readability, consistency, and organization.

Notes: For this section **only**, we will allow (and encourage) students to share their diagrams on Discord (**#er-diagram-feedback**) to get feedback from other students on their ER diagrams given a brief summary of your dataset and domain requirements. This is offered as an opportunity to test your ER diagrams for accuracy and robustness, as another pair of eyes can sometimes catch constraints that are not satisfied or which are inconsistent with your specified domain requirements.

Requirements:

- Entity sets, relationship sets, and weak entity sets should be properly represented (also, do not use ER symbols not taught in class)
- Mapping cardinalities should be appropriate for your database schema, and in sync with your DDL
- Participation constraints should be appropriate and in sync with your DDL (total, partial, numeric)
- Use specialization where appropriate (e.g. *purchasers* and *travelers* inheriting from a *customers* specialization in A6)
- Do not use degrees greater than 3 in your relationships, do not use more than one arrow in ternary relationships.
- Use descriptive attributes appropriately
- Underline primary keys and dotted-underline discriminators
- Expectations from A6 still apply here
- Note: You do not need ER diagrams for views

ER Diagrams:



Part B. DDL (Indexes)

As mentioned in Part B, you will need to add at least one index at the bottom of your **`setup.sql`** and show that it makes a performance benefit for some query(s).

Here, describe your process for choosing your index(es) and show that it is used by at least one query, which speeds up the performance of the same query on a version of the same table without that index. You may find lec14-analysis.sql and Lecture 14 slides on indexes useful for strategies to choose and test your indexes. **Remember that indexes are already created in MySQL for PKs and FKs, so you should not be recreating these.**

Index(es):

An index on the total_delay attribute of route: CREATE INDEX idx_route ON route (total_delay);

Justification and Performance Testing Results:

From queries.sql, it is clear that the majority of our queries utilize the total_delay attribute to output the average total delay based on various attributes such as manufacturer/model, origin/destination etc. Thus, choosing an index on total_delay allows us to speed up all such queries, especially since total_delay is not a primary key or foreign key. In fact, the sp_all_routes_update_total_delay procedure, which updates the total_delay parameter for all of the routes initially added via loading the data, takes 0.66 seconds after adding the index instead of 0.99 seconds it took before.

Furthermore, as a testing example via EXPLAIN, the first query in queries.sql, which gets the average delay across all routes for each airline, was run without and with the index:

Before Index:

```
+----+-------------+---------+------------+--------+-----------------------------------------------------------------+---------+---------+
-----------------------------+------+----------+-------+

| id | select_type | table   | partitions | type   | possible_keys                                                   | key     | key_len | ref
| rows | filtered | Extra |

+----+-------------+---------+------------+--------+-----------------------------------------------------------------+---------+---------+
-----------------------------+------+----------+-------+

|  1 | SIMPLE      | route   | NULL       | index  | PRIMARY,tail_num,delay_info_id,origin_code,destination_code |
PRIMARY | 20      | NULL                |  993 |   100.00 | NULL  |

|  1 | SIMPLE      | airline | NULL       | eq_ref | PRIMARY                                                         | PRIMARY | 14      |
flightdb.route.carrier_code |    1 |   100.00 | NULL  |

+----+-------------+---------+------------+--------+-----------------------------------------------------------------+---------+---------+
-----------------------------+------+----------+-------+
```

After Index:

```
+----+-------------+---------+------------+--------+-----------------------------------------------------------------------------+---------+-
--------+-----------------------------+------+----------+-------+

| id | select_type | table   | partitions | type   | possible_keys                                                               | key     | key_len | ref
| rows | filtered | Extra |

+----+-------------+---------+------------+--------+-----------------------------------------------------------------------------+---------+-
--------+-----------------------------+------+----------+-------+

|  1 | SIMPLE      | route   | NULL       | index  |
PRIMARY,tail_num,delay_info_id,origin_code,destination_code,idx_route | PRIMARY | 20      | NULL
|  993 |   100.00 | NULL  |

|  1 | SIMPLE      | airline | NULL       | eq_ref | PRIMARY                                                                     | PRIMARY | 14      |
flightdb.route.carrier_code |    1 |   100.00 | NULL  |

+----+-------------+---------+------------+--------+-----------------------------------------------------------------------------+---------+-
--------+-----------------------------+------+----------+-------+
```

As can be seen from these EXPLAIN outputs, idx_route is now factored into the performance of the query, with idx_route clearly present in the possible_keys section of the after index query. Thus, indeed adding the index improved performance of this query, indicating the benefit of adding this index.

Part C. Functional Dependencies and Normal Forms

Requirements (from Final Project specification):

- Identify *at least 2 non-trivial functional dependencies* in your database
- Choose and justify your decision for the normal form(s) used in your database for at least 4 tables (if you have more, we will not require extra work, but will be more lenient with small errors). BCNF and 3NF will be the more common NF's expected, 4NF is also fine (but not 1NF).
    - Your justification will be strengthened with a discussion of your dataset breakdown, which we expect you to run into trade-offs of redundancy and performance.
- For two of your relations having at least 3 attributes (each) and at least one functional dependency, prove that they are in your chosen NF, using similar methods from A7.
    - If you have identified functional dependencies which are not preserved under a BCNF decomposition, this is fine
- Expectations from A7 still apply here.

Functional Dependencies:

airport(IATA_code) -> airport(airport_name)

airline(carrier_code) -> airline(airline_name)

route(carrier_code, flight_num, depart_time) -> route(delay_info_id)

Normal Forms Used (with Justifications):

Airline:

BCNF used

Here, we have chosen to use the Boyce-Codd Normal Form in our table which maps information about an airline's carrier_code to its name. For this table, the only functional dependency which exists is the dependency carrier_code -> airline_name. Furthermore, the carrier_code of an airline will be the primary key of the table, making it a super key. Therefore, we feel that BCNF is the most appropriate choice.

Airport:

3NF used

Here, we have chosen to use the 3-Normal Form in our table which stores the associated information for each airport. While the 3NF form is slightly weaker than the BCNF form, we ultimately decided that this was the best option for our database and made the most sense for our implementation. Because we needed to store the exact longitudinal and latitudinal coordinates, which could also be used to uniquely identify an airport but is not a primary key, our table has the functional relation

(latitude, longitude) -> airport_name, etc. However, latitude and longitude are both candidate keys in this table, so by the new condition of the Third Normal Form, our table would hold.

Plane:

BCNF used

Here, we have chosen to use the Boyce-Codd Normal Form in the table which stores the associated information for each aircraft that will be used in the airline's flights. The only functional dependencies that exist here are those between the tail number and the other attributes. Since tail_num is the primary key of this table, this means that the only non-trivial functional dependencies are those that include a superkey. Therefore, we felt that BCNF was most suitable.

Delay_info:

BCNF used

Here, we have again used the BCNF in the table which stores the information regarding delay times for each flight. Since each flight will have a delay_info_id, functional dependencies exist in this table between delay_info_id and all other attributes. However, there are no other non-trivial functional dependencies. Delay_info_id is also the primary key of this table, meaning BCNF, which would be the strongest form that we could use in this case, would be best.

NF Proof 1:

Plane

Start with schema

plane(tail_num, manufacturer, issue_date, model, status, aircraft_type, engine_type)

and

F = {tail_num -> manufacturer, tail_num -> issue_date, tail_num -> model, tail_num -> status, tail_num -> aircraft_type, tail_num -> engine_type}

For all functional dependencies in F, of form a -> b, a is a superkey of plane since tail_num is the primary key of plane.

NF Proof 2:

Delay_info

Start with schema

delay_info(delay_info_id, depart_delay, arrival_delay, airline_delay, weather_delay, aircraft_delay, nas_delay, security_delay)

And

F = {delay_info_id -> depart_delay, delay_info_id -> arrival_delay, delay_info_id -> airline_delay, delay_info_id -> weather_delay, delay_info_id -> aircraft_delay, delay_info_id -> nas_delay, delay_info_id -> security_delay, }

For all functional dependencies in F, of form a -> b, a is a superkey of delay_info since delay_info_id is the primary key of delay_info

---

Part G. Relational Algebra

Requirements (from Final Project specification, Part G):

- Minimum of 3 non-trivial queries (e.g. no queries simply in the form **`SELECT <x> FROM <y>`**)
- At least 1 group by with aggregation
- At least 3 joins (across a minimum of 2 queries)
- At least 1 update, insert, and/or delete
    - This may be equivalent to said SQL statements elsewhere (e.g. queries or procedural code), but are not required to be; in other words, you can write these independent of other sections
- Appropriate projection/extended projection use
- Computed attributes should be renamed appropriately
- Part of your grade will come from overall demonstration of relational algebra in the context of your schemas; obviously minimal effort will be ineligible for full credit; it is difficult to formally define "obviously minimal", but refer to A1 and the midterm for examples of what we're looking for
- Above each query, briefly describe what it is computing; we will use this to grade for correctness based on what the query is supposed to compute; lack of descriptions will result in deductions, since we have no idea otherwise of what the query is intended to do.

Below, provide each of your RA queries following their respective description.

1. Query to get the average total delay (in minutes) across all flight routes for each airline:

$$\Pi_{airline\_name,\, avg\_delay}({}_{carrier\_code}G_{average(total\_delay)\ as\ avg\_delay}(route \bowtie airline))$$

Related SQL Query in queries.sql:

```sql
SELECT airline_name, AVG(total_delay) AS avg_delay

FROM route NATURAL LEFT JOIN airline

GROUP BY carrier_code;
```

2. Query to get the number of times each origin/destination airport pair/route is present in the database, with the names of the airports present in the result instead of the IATA code to amplify user experience

$$\Pi_{a1.port\_name\ as\ origin\_name,\, a2.port\_name\ as\ dest\_name,\, num\_routes}({}_{origin\_code,\, dest\_code}G_{count(*)\ as\ num\_routes}$$
$$(\sigma_{route.origin\_code = a1.port\_code \wedge route.origin\_code = a2.port\_code}\ (route \times \rho_{a1}(airport) \times \rho_{a2}(airport))))$$

Related SQL Query in queries.sql:

```
SELECT a1.port_name AS origin_name, a2.port_name AS dest_name,
   COUNT(*) AS num_routes
FROM (route JOIN airport AS a1
   ON route.origin_code = a1.port_code)
   JOIN airport AS a2
   ON route.destination_code = a2.port_code
GROUP BY origin_code, destination_code;
```

3. Query to update the status of all planes with model '717-200' to become 0 (representing that they have become inactive) due to a grounding of all aircraft with this model.

$$plane \leftarrow \Pi_{tail\_num,\, manufacturer,\, issue\_date,\, model,\, 0,\, aircraft\_type,\, engine\_type}(\sigma_{model="717-200"}(plane)) \cup \sigma_{model\neq"717-200"}$$

Part L1. Written Reflection Responses

Challenges and Limitations

List any problems (at least one) that came up in the design and implementation of your database/application (minimum 2-3 sentences)

Answer:

While designing and implementing our database, we had to consider how to represent the delays associated with a particular flight route in the database. When we initially tried to implement the flight routes, we decided to include all delay attributes (weather_delay, airline_delay, etc) within the route database itself. However, this quickly became an issue since the route database was unnecessarily large and included attributes that it largely had no need for when implementing most queries. Thus, we decided to instead create another database called delay_info and create a one-to-one mapping between delay_info and route, with delay_info storing all specific information about all the different types of delays for a particular route. In addition, since the total delay for a route was frequently used in queries on route, we decided to include a total_delay attribute in the route database that would automatically be populated with the total delay via the associated delay_info row via a trigger.

Future work

If you are particularly eager for a certain application (have your own start-up in mind?), it is easy to over-scope a final project, especially one that isn't a term-long project. You can list any stretch goals you might have "if you had the time" which staff can help give feedback on prioritizing (2-3 sentences).

Answer:

1. Import data from all years 1987 to 2008 from the dataset (12 gb) into the database to make more accurate predictions on when delays would more likely occur (instead of just using data from a couple years)
2. Generate data on pilots and make predictions for delays based on the pilots (i.e. certain pilots may be better at flying, thus avoiding delays etc).
3. Suggesting where and when new routes for airlines should be based on delay patterns from previous years.

Collaboration (Required for Partners)

This section is required for projects which involved partner work. Each partner should include 2-3 sentences identifying the amount of time they spent working on the project, as well as their specific responsibilities and overall experience working with a partner in this project.

Partner 1 Name: Eric Han
Partner 1 Responsibilities and Reflection:
I spent a total of 12 hours total working on this project. I focused mostly on developing the python application and password management/permissions via SQL. Overall, this project was a pretty good experience, and I particularly enjoyed learning how to incorporate MySQL data and queries into a python application.

Partner 2 Name: Sahil Jain
Partner 2 Responsibilities and Reflection:
I spent a total of 11.5 hours total working on this project. I focused mostly on developing the database schema, queries, and procedural SQL, including ER diagrams, DDL, and functional dependency analysis. I thoroughly enjoyed the project and was excited to apply all of our knowledge from this term to a real-world application.

Other comments

This is the first time CS 121 has had a Final Project, and we would appreciate your feedback on whether you would recommend this in future terms, as well as what you found most helpful, and what you might find helpful to change.

Answer:

I think it was overall really helpful to have the python application template when implementing app.py. Furthermore, it was particularly enjoyable to develop a UI and password-enabled application, and serves as a real-world example of how mysql can be eloquently employed as an efficient database management software. However, in future terms, I would recommend reducing the length of the project slightly, as it is a huge time commitment during an already busy final's week.