CS 121 Final Project Reflection

Due: March 16th, 11:59PM PST

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): Grace Lu, Jae Yoon Kim

Student email(s): glu@caltech.edu, jaeyoonk@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) :

We created a database application for a grocery store and tried to come up with ways that different stakeholders could manipulate the data in a way that protects the integrity of the data but still allows different types of users to access and edit the database based on their privilege level. The motivation for this is when we walked through a grocery store and marveled at the complexity of a grocery store with the thousands of products and tens of aisles.

Data set (general or specific) Answer:

We were trying to get the dataset from the Instacart official website (https://www.instacart.com/datasets/grocery-shopping-2017) but that was currently down so I retrieved it from a third party online that just stored the same dataset (https://www.p8105.com/dataset_instacart.html). The data is just one huge CSV file so we took the time to parse it into the schemas like with the Spotify data. We wanted to build a grocery store system with this data and have ways to analyze customer purchasing behavior.

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

An average department employee would be able to access the database to view any interesting information with regards to products that are sold. As a client, this user would be able to choose to run query among many pre-made queries. These include finding the most popular item sold per day of the week, and the average number of items in a customer's cart.

Admin user(s) Answer:

The administrator has power that the clients do not. The administrator is able to add new employees and their access username/password. The administrator is able to run all the queries that the client is able to but also those that modify the database.

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.

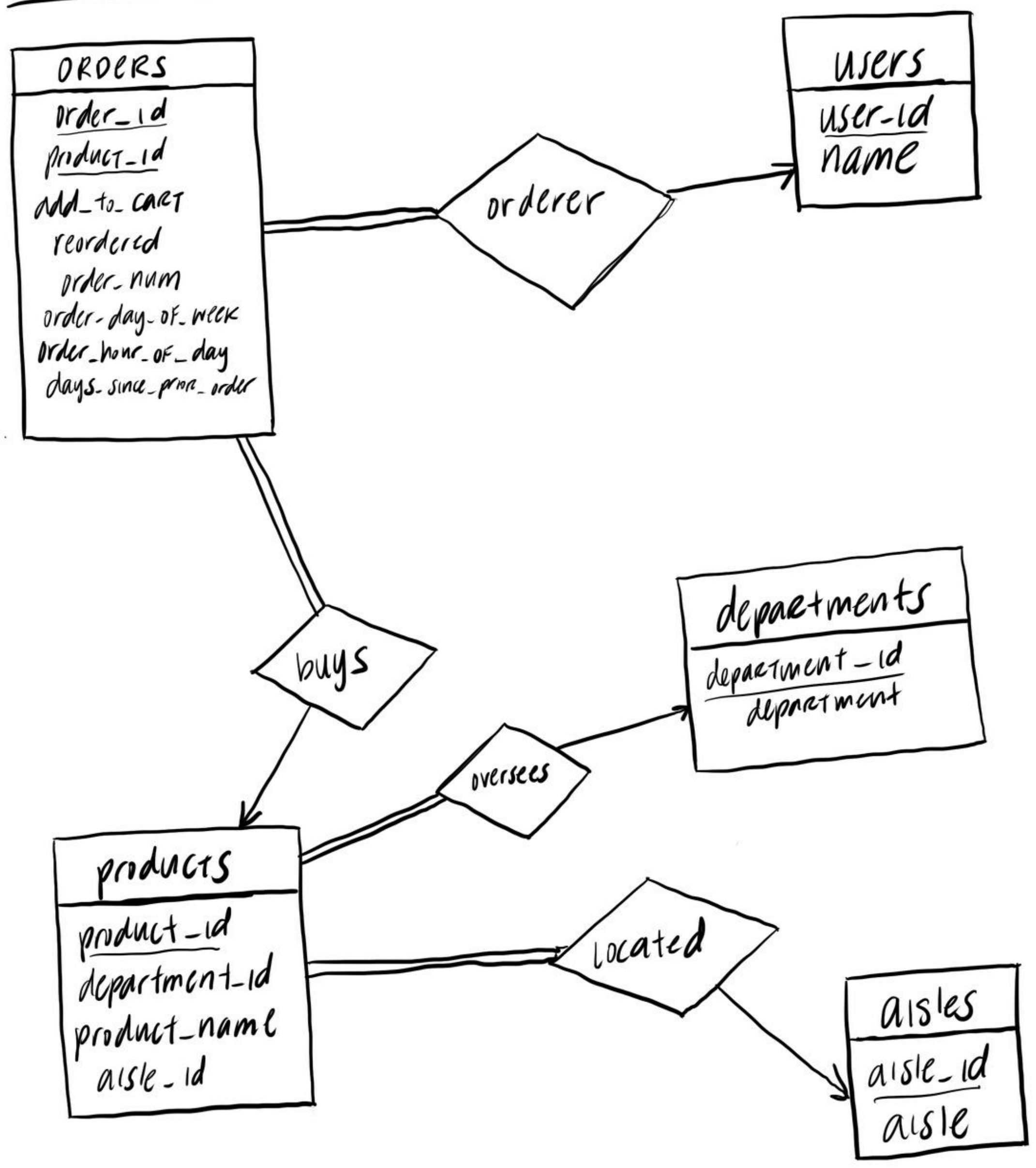
Notes: For this section **only**, we will allow (and encourage) students to share their diagrams on Discord (**#er-diagram-feedback**) to get feedback from other students on their ER diagrams given a brief summary of your dataset and domain requirements. This is offered as an opportunity to test your 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:

ER Diagram


ORDERS
order_id
product_id
add_to_cart
reordered
order_num
order_day_of_week
order_hour_of_day
days_since_prior_order
orderer
users
user_id
name
buys
departments
department_id
department
oversees
products
product_id
department_id
product_name
aisle_id
located
aisles
aisle_id
aisle

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).

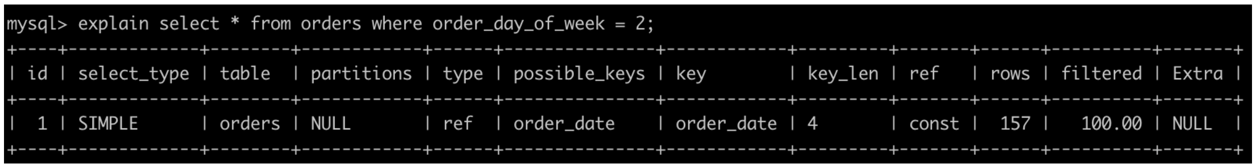
Here, describe your process for choosing your index(es) and show that it is used by at least one query, which speeds up the performance of the same query on a version of the same table without that index. You may find lec14-analysis.sql and Lecture 14 slides on indexes useful for strategies to choose 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):

We created an index called order_day_idx on order_day_of_week in the orders table.

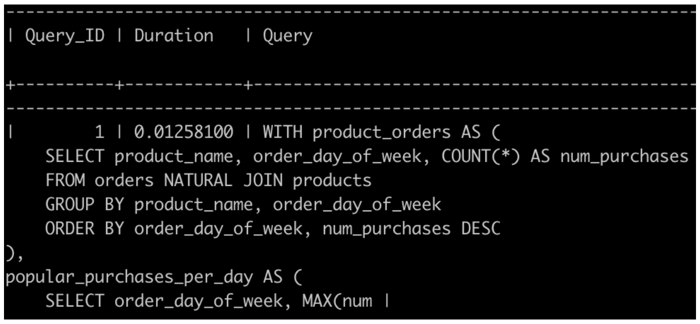
Justification and Performance Testing Results:

This index is useful because a lot of our analysis is based on what are popular products on certain days of the week, so we have to query the order_day_of_week a lot to separate by days.



```
mysql> explain select * from orders where order_day_of_week = 2;
+----+-------------+--------+------------+------+---------------+------------+---------+-------+------+----------+-------+
| id | select_type | table  | partitions | type | possible_keys | key        | key_len | ref   | rows | filtered | Extra |
+----+-------------+--------+------------+------+---------------+------------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | orders | NULL       | ref  | order_date    | order_date | 4       | const |  157 |   100.00 | NULL  |
+----+-------------+--------+------------+------+---------------+------------+---------+-------+------+----------+-------+
```

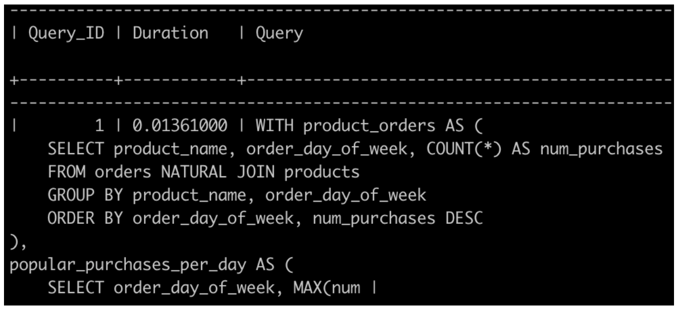
With the index: dd2dl l



```
------------------------------------------------------------------------
| Query_ID | Duration   | Query

+----------+------------+-------------------------------------------
------------------------------------------------------------------------
|        1 | 0.01258100 | WITH product_orders AS (
    SELECT product_name, order_day_of_week, COUNT(*) AS num_purchases
    FROM orders NATURAL JOIN products
    GROUP BY product_name, order_day_of_week
    ORDER BY order_day_of_week, num_purchases DESC
),
popular_purchases_per_day AS (
    SELECT order_day_of_week, MAX(num |
```

Without the index:



```
-----------------------------------------------------------------------
| Query_ID | Duration   | Query

+----------+------------+-----------------------------------------------
-----------------------------------------------------------------------
|        1 | 0.01361000 | WITH product_orders AS (
    SELECT product_name, order_day_of_week, COUNT(*) AS num_purchases
    FROM orders NATURAL JOIN products
    GROUP BY product_name, order_day_of_week
    ORDER BY order_day_of_week, num_purchases DESC
),
popular_purchases_per_day AS (
    SELECT order_day_of_week, MAX(num |
```

Therefore, we can see with the index, it's faster.

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:

- User_id ⟶ name
- Department_id ⟶ department
- Aisle_id ⟶ aisle
- Product_id ⟶ department_id, product_name, aisle_id
- Order_id, product_id ⟶ add_to_cart, reordered, order_num, order_day_of_week, order_hour_of_day, days_since_prior_order

Normal Forms Used (with Justifications):

We used the 3NF normal forms because we're only working with one grocery store data. There's only so many orders a grocery store can handle, so there's not too much data. Therefore, the 3NF does a good job of preserving dependencies, which are important for grocery stores to manage their orders and make sure everything is where it should be and people get the right items. Since the database isn't too big, it shouldn't be too expensive to have repeated information with this normal form.

NF Proof 1:

Let's prove that products(product_id, department_id, product_name, aisle_id) is in 3NF. We can see that the functional dependency here is product_id → department_id, product_name, aisle_id. Using the 3NF synthesis algorithm, we can see that we just do (alpha union beta), which in this case would be (product_id, department_id, product_name, aisle_id). We can clearly see that this contains a candidate key (product_id) so it is valid and cannot be broken down further. Therefore, it is in 3NF form.

NF Proof 2:

Let's prove that orders(order_id, product_id, add_to_cart, reordered, order_num, order_day_of_week, order_hour_of_day, days_since_prior_order) is in 3NF. We can see that the functional dependency here is order_id, product_id → add_to_cart, reordered, order_num, order_day_of_week, order_hour_of_day, days_since_prior_order. Using the 3NF synthesis algorithm, we can see that we just do (alpha union beta), which in this case would be (order_id, product_id, add_to_cart, reordered, order_num, order_day_of_week, order_hour_of_day, days_since_prior_order). We can clearly see that this contains a candidate key (order_id, product_id) so it is valid and cannot be broken down further. Therefore, it is in 3NF form.

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.

Query 1: This query finds the number of visits to that aisle based on the orders in the grocery store. So an order with a produce x, that is located in aisle y, counts as one visit to aisle y. This query includes a group by with aggregation and a join.

most popular aisles w/ num visits

$$temp \leftarrow \Pi_{product\_id, product\_name, aisle\_id, aisle}(orders \bowtie products \bowtie aisles)$$

$$\Pi_{aisle\_id, aisle, num\_aisle\_visits}\left[{}_{aisle\_id, aisle}\mathcal{G}_{count(*)\ as\ num\_aisle\_visits}(temp)\right]$$

Includes aggregation, and Join

Query 2: This query returns the users in the database that are returning users. A returning user is defined as having placed more than one order. This query uses a group by with aggregation and a join.

<u>number of returning customers</u> (placed more than 1 order)

$$temp \leftarrow \Pi_{\substack{user\_id \\ num\_orders \\ \_per\_user}} \left[{}_{\substack{user\_id, \\ order\_id}} \mathcal{G}_{\substack{count(order\_id) \\ as\ num\_orders\_per\_user}} \left[\Pi_{\substack{user\_id, \\ order\_id}} (orders \bowtie user\_orders) \right] \right]$$

$$\Pi_{\substack{user\_id, name \\ num\_orders\_per\_user}} \left[\sigma_{\substack{num\_orders\_ \\ per\_user > 1}} (temp \bowtie users) \right]$$

aggregation & join

Query 3: This query removes the "cake" department in the grocery store and all of the products that are in the "cake" department. This query includes a delete.

remire the "cake" department and all the products in it.

$$temp \leftarrow \Pi_{product\_name}\left[\sigma_{department = \text{"cake"}}(departments \bowtie products)\right]$$

$$products \leftarrow products - (temp \bowtie products)$$

$$departments \leftarrow \sigma_{department \,!= cake}(departments)$$

delete

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:
One problem was that there would be some situations where using ON DELETE CASCADE would make sense but not so much in other situations depending on other circumstances. We had to be judicious with using these foreign key constraints. We spent a lot of time discussing which of the keys needed which constraints because we were aware of the performance cost it would take to implement an increasing number of constraints.

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:
A stretch goal would've been to using React to create a website that accesses the SQL database through the Python command line so that more graphically inclined users could take advantage. This would make it a lot more intuitive for people to use but would take a lot more time than creating the Python command line application. This website may or may not have interactive elements so users don't even have to type anything.

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: Grace Lu
Partner 1 Responsibilities and Reflection:

- I did a lot of the proposal, the ER diagram, DDL, Functional Dependency Theory, Loading part, Relational algebra, and the SQL queries. I also cleaned the data to make it loadable and filled in the reflection. I spent about 20-25 hours on this project. The experience was good–I think we played to our strengths.

Partner 2 Name: Jaeyoon Kim
Partner 2 Responsibilities and Reflection:

- I worked on the proposal, README, setup passwords, grant permissions, the procedures/triggers/UDF, and the Python application. I spent about 20 hours on the project as well. Grace was a great teammate to work with.

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 would have much preferred a final exam. I think for a final project, it had late notice with the amount of work required and it took a lot of time. So I definitely think in the future, it should at least be announced earlier to give people time to think about it and start.