CSC 355 Database Systems
Fall Quarter 2018
Final Project

Due Monday, November 19th

In this project, you will work with a modified version of the Star Schema Benchmark (SSBM). This benchmark is widely used in the database systems research community. It combines a realistic data distribution (while maintaining correct data types and cross-column correlations) with a synthetic data generator. The table definitions (including the primary keys and foreign keys) and the data to populate these tables can be downloaded from the following link:

http://dbgroup.cdm.depaul.edu/CSC355_Fall2018/FinalProjectData.zip

There is a total of three parts to this project along with an extra credit part. Each part has multiple tasks. Everything that requires a response in your submission is in **bold**.

Part I. Database Design (40 points)

A. You are tasked with enforcing the following functional dependency on the customer data:

City → Nation

As we've learned in class, this functional dependency can be enforced through decomposition. However, in the schema you are given, the customer table was unnecessarily decomposed (it happens in the real-world) into CUSTOMER_T1 and CUSTOMER_T2. Therefore, this functional dependency must be enforced across multiple tables. Luckily, we learned that triggers can be used in this situation.

Q.PI.A1. Write a trigger that enforces the functional dependency City → Nation for INSERT, UPDATE, and DELETE.

Hint: triggers can only be attached to one table at a time, so the same trigger would have to be attached to each table involved in the FD.

B. You are tasked with tracking the total price of all orders for each customer. Fortunately, someone previously added the column C_TotalOrderPrice column to the customer table. However, one of our employees currently spends 2 hours each day manually updating these entries. Rather than have someone manually update these values, we want to streamline our business, and automatically update this value.

Q.PI.B1. Write a trigger that updates the C_TotalOrderPrice column in the customer table each time a new order is added, updated, or deleted from the LINEORDER table.

C. Due to federal regulations, your company can only fulfill orders on Monday, Tuesday, and Wednesday. You need to guarantee that orders will not occur on Thursday, Friday, Saturday, or Sunday. Luckily for us, Oracle has a function that returns the day of the week for a given date:

```
SELECT TO_CHAR(date '2018-11-01', 'DAY') day FROM animal;
```

returns 'Thursday'. In the data you are given, the LO_Orderdate column in the LINEORDER table is unfortunately an integer (data will not always be clean and friendly in the real-world). Therefore, you will need to convert the integer to the proper date format (Hint: this can be done with SUBSTRING and string concatenation, ||).

Q.PI.C1. Write a trigger that prevents records from being added to the LINEORDER table where the LO_OrderDate is a Thursday, Friday, Saturday, or Sunday.

Hint: You will want to create a view that replicates lineorder (CREATE VIEW Lineorderview AS SELECT * FROM Lineorder), load the data by inserting into LineorderView and use an INSTEAD OF INSERT trigger over that view.

D. Using the DDL file provided and the triggers you created in Parts I.A I.B, and I.C, load the data from the csv files provided.

Q.PI.D1. Report the time to load each table and the number of records for each table using SELECT COUNT(*) FROM [table_name].

Hint: You may want to call executeBatch() every 10,000 records for reasonable performance.

Part II. SQL (21 points)

Write SQL queries for 7 out of the 10 following questions:

A. Compute the average lo_quantity for each c_city and c_nation combination.

B. Find the minimum, maximum, and average discount (in a single query) for suppliers whose address includes at least one comma (“,”) and who are located outside of Canada.

C. For each combination of even year (1992, 1994, …) and odd month (1, 3, …) find the number of different p_color and c_mktsegment values

D. P_container entry always consists of 2 words separated by a space. For each p_container where the 2nd word has exactly 4 characters (e.g., ‘WRAP DRUM’) find the number of unique lo_shippriority and c_mktsegment values. (NOTE: You can use LIKE or REGEXP_LIKE, as you prefer).

E. Find all lineorder entries (lo_orderdate, lo_orderpriority, lo_shippriority, lo_revenue and lo_tax) for suppliers whose city has at least one space (e.g., ‘BRAZIL 5’ but not ‘INDONESIA5’). Your result should be sorted by d_dayofweek in ascending order.

F. Find the dates (d_dayofweek, d_month, d_year) on which at least one lineorder record had higher-than-average lo_quantity. No date listing should appear more than once and your output should be sorted by d_year in decreasing order.

G. Find the c_nation (or s_nation) that has the greatest, combined amount of both suppliers and customers. (e.g., if Canada has 100 suppliers and 100 customers, the total amount would be 200 for Canada).

H. Find all parts where the p_color appears in the p_name. (e.g., the p_color of ‘red’ appears in the p_name ‘red brick’).

I. Find the unique set of customer names that ordered parts with a larger than the average p_size from a supplier with a s_region that is different than c_region.

J. Using a single query, return two values: 1) the total number of parts where the p_type contains ‘COPPER’, and 2) the total number of parts where the p_type contains ‘STEEL’. Your output should contain two columns (COPPER and STEEL) and one record that is the total number of parts for each.

Part III. Query Optimization (39 points)

A. Your company often does part lookups. Lately, you have received complaints that these take a long time. You are tasked to solve this problem. To investigate, you run a query that performs a lookup for a single part.

Q.PIII.A1. Run and report the runtime the following query 2000 times:

```
SELECT P_Type. P_Size
FROM Part
WHERE PName = ‘dodger moccasin’;
```

Hint: Do not time each individual query execution. Your start and end time variables should be outside of the FOR loop in Java, timing the 2000 inserts as a unit.

To speed up the queries you decide to create an index:

Q.PIII.A2. Create an index that would improve the performance of the above query.
Q.PIII.A3. Re-run the same query 2000 times and report the new runtime.
Q.PIII.A4. Was there a difference between the two runtimes? Explain why or why not.

B. Create the following index on the discount column from the LINEORDER table:

```
CREATE INDEX OrderDiscount ON LINEORDER(LO_DISCOUNT);
```

Someone in your company entered the wrong discount and revenue amount for each order (all of the values were off by one). You can fix this by running the following queries.

Query 1:

```
UPDATE LINEORDER
SET LO_Discount = LO_Discount + 1
WHERE LO_Shipmode IN ('RAIL', 'FOB', 'MAIL', 'REG AIR', 'AIR', 'SHIP');
```

Query 2:

```
UPDATE LINEORDER
SET LO_Revenue = LO_Revenue + 100
WHERE LO_Shipmode IN ('RAIL', 'FOB', 'MAIL', 'REG AIR', 'AIR', 'SHIP');
```

Q.PIII.B1. For each query, report the runtime and the number of records updated.
Q.PIII.B2. Was there a difference between the runtime of the two queries? Explain why or why not.

C. Your company deals with many international customers. Therefore, each time someone wants to look up the total revenue for an international customer, a conversion rate must be calculated. To save your employees time, you decide to create a materialized view.

Q.PIII.C1. Create a materialized view that stores the total revenue for each customer, and the conversion rate for at least 5 different (actual) currencies of your choice.

You can assume that original amount is in USD. Also note that you can retrieve the revenue from the LINEORDER table, which is different than the TotalOrderPrice stored in CUSTOMER_T2.

Part IV. Extra Credit (This is optional. Up to +16 points on the final project)

A. Database Decay Paper (+8 points)

Write a good summary and a good discussion for the following research paper:

M. Stonebraker et al., Database Decay and How to Avoid It, 2016.

If you are on DePaul's network, you can access the paper through the IEEE digital library at the following link:

http://ieeexplore.ieee.org/document/7840584/

If you are not on DePaul's network, you can access the paper through DePaul's library with the following steps:

- https://libguides.depaul.edu/az.php?q=IEEE
- With "IEEE" in the search box, select "IEEE Xplore"
- Once on the IEEE website, search "Database Decay and How to Avoid It". It should be the first paper that is returned.

What is a "good" summary?

- Around ½ – 1 page
- What was the research question? What were the findings/results? What are the implications of the results?
- Focus on content, not length

What is a "good" discussion?

- Relate the paper to class and/or your work/interests
- How does this paper relate to the real-world?
- Anything else insightful

B. SQL (+3 points)

Write the additional SQL queries you did not answer from Part II. (+1 point for each correct query, no partial credit).

C. ScorchedEarth Function (+5 points)

Write a PL/SQL function that will self-destruct the database ("*This message will self-destruct in 10 seconds*"). We will drop all of the triggers, procedures, functions and tables defined in the database.

You can find triggers in ALL_TRIGGERS table (**note: you should only be dropping triggers where YOU are the TABLE_OWNER!**). ALL_PROCEDURES stores both functions and procedures (again, do not attempt to drop system owned entries – you won't have permission to do so, but let's not scare the DBAs unnecessarily). USER_TABLES contains the list of all user tables.