Relational Databases and SQL - Course Project Instructions

Uri Goldstein, Reichman University, January 2022

Version 1.0, 10/02/2022

1. Goal and Background

As students of the course "Relational Databases and SQL" you are tasked with submitting a database implementation project as a major part of your course grade. The goal of the project is to demonstrate your understanding and practical experience of the material of the course. Use this project to show how the skills you learned throughout the semester can be used to build a real, usable database.

The central task in this year's project is to implement specific parts of a database for a Bus Rapid Transit system for Gush Dan in Israel.

You will:

- **Design** the structure of the database.
- **Implement** the design using DDL commands.
- Insert some **sample data** to the database.
- Write several **queries** against the database.
- Create a visual **model** of the structure of the database.

Please read the following sections very carefully.

---

2. Requirements

Below are the specific data requirements to be implemented in your submission for the project.

2.1. Entities

All the relevant business entities of the BRT system are listed below. Your database must represent all these entities according to relational principles that we've learned in the course.

2.1.1 Buses

A bus is a large automobile used for transporting people between places in a municipal area. Buses come in many shapes, sizes, makes and models but in our system the only properties of a bus that matter are:

- License plate number - For example 12-345-67 or 123-45-678.
- Line - At any given moment a bus can only be assigned to a single bus line.
- Capacity - This property is explained in detail in section 2.1.6 below.

2.1.2 Bus Stops

Buses stop to pick up or drop off passengers at predesignated locations called "bus stops". Bus stops are typically small sheds located on main streets.

Bus stops have:

- A unique identifying number like 3476 or 12990.
- A name. For example: "Dizengoff Center North" or "13 Ben Gurion st. ". Stop names are not necessarily unique.
- An address like "13 Ben Gurion st., Tel-Aviv"

Any given bus stop can be a part of zero, one or more bus routes (see below).

2.1.3 Routes

Buses are only allowed to travel along predesignated routes. A route is an **ordered** series of bus stops.

Here is an example of one of the routes of bus line no. 5 in Tel-Aviv:

```
1. HaLohamim Parking/Licensing Office (35246)
2. Licensing Bureau/HaLohamim (36392)
3. Panorama Building/Freigerson (25262)
...
35. Hadash Rabin High School/Namir Road (21534)
36. Tel-Aviv Raqevet Merkaz Park and Go (25937)
```

Important! The ordered numbering of the stops inside a route is a part of the data requirements. It is essential for the implementation of some of the queries that you will need to write.

It is also required that your database will maintain data integrity so that no two stops along a given route share the same number. (e.g., a route cannot have two stops numbered 2).

2.1.4 Lines

Passengers in our BRT system are used to thinking in terms of "bus lines". A line is identified by its "number" which is an alphanumeric value like "89", "27A" or even "405 Express".

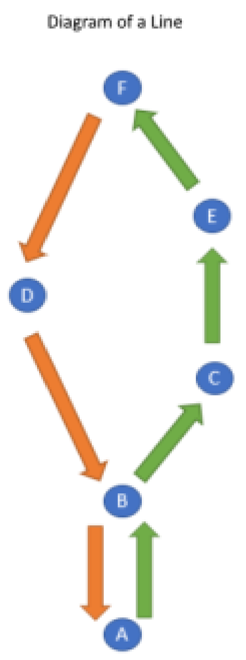
Every line is made up of two routes. One of the routes - typically the one that goes in the general direction of North or East - is called the "up" route. The other route is called "down" and usually goes in the general direction of South or West.

The two routes that make up a line must always form a closed loop together. This means that the first stop for the up route must be the same as the last stop for the down route and vice versa. This rule must be enforced by your database.

For example, the route shown in 2.1.3 above is the up route for Tel Aviv bus line no. 5. The down route will start at "Raqevet Merkaz" and end at "Halohamim Parking/Licensing Office". The stops in the middle of the down route do not necessarily have to be the same as the one in the up route and vice versa.

As another example, consider the diagram on the right. It shows a single line where the "up" route is marked with green arrows and the "down" route is marked with orange arrows. Bus stops are blue dots.

Note that the two routes do not share all stops between them and even the number of stops in each route can be different. However, the two routes share their first and last stops: Stop A is first for "up" and last for "down" and stop F is its opposite.



2.1.5 Tags

Tags are little bits of information that can be assigned to buses. For example, a given bus can be assigned two tags: "electric" and "extra-large" to signify that this bus is electric and can contain many passengers.

A given tag can be assigned to any number of buses. This means that tags and buses have a many-to-many relationship between them.

As shown above each tag is a short string made up of one or more words separated by dashes.

Important! Tag names cannot contain spaces. Tag names are always lowercase. Make sure your database enforces this.

2.1.6 Capacities

Different types of buses can have different capacities to carry passengers. The capacity of a type of bus is the largest the number of passengers (excluding the driver) that are allowed to be present on the bus at any given time.

There are 5 types of capacities:

Name	Value
Small	8
Medium	16
Large	32
Extra Large	50
Double	64

In your database design, please implement capacities based on tags. As tags only have names you will need to come up with a solution to associate specific capacity tags with their numeric values.

2.2 Example Data

As part of your submission, you must include a set of DML SQL commands that populate all your tables with some example data.

The minimum data to include is:

- 10 buses.
- 7 stops.

- 4 routes that make up 2 lines.
- 10 tags (5 of which are the capacity tags).

The example data doesn't have to use real world data. You can invent your own bus stop names, line numbers etc. but keep them consistent with the requirements above.

Every pair is expected to make up and submit their own examples. **Do not use examples from this document**. The one exception here are the 5 capacity tags which you need for query 2.3.2 below.

2.3 Queries

2.3.1 Validating a line's routes

Write a single SELECT query that given a line id returns 1 if and only if the line is a proper loop. 0 should be returned if it isn't.

For the line to be a proper loop two conditions must be true:

a. The first stop of the line's "up" route must be the same as the last stop of its "down" route.

and

b. The first stop of the line's "down" route must be the same as the last stop of its "up" route.

Hint! It is possible (but not necessary) to implement this query without using any JOINs.

2.3.2 Average Capacity per Bus Stop

Write a single SELECT query that given a bus stop id returns the average capacity (as a number) of the bus stop. The average capacity of a bus stop the average of the capacities (as numbers) of all the buses that are assigned to lines that have routes that go through the bus stop.

For example, consider a bus stop called "Cinema City East". There is only one route that stops at this bus stop, let's call it "Route 24 North". This route is part of the line called "Line 24" and this line has 3 buses assigned to it:

- The capacity of bus '12-345-67' is Extra Large (50 passengers).
- The capacity of buses '11-111-11' and '222-22-222' is Large (32 passengers).

In this example the average capacity of the bus stop is therefor: 38.

2.4 Performance

The only requirement in terms of the performance of the database is that the queries above should not require a full table scan. Whenever you find that a query you've written causes a full table scan when run, add the minimum number of indexes to avoid the scan.

---

3. Submissions

3.1 Pairing

You may only submit the project in pairs. You and your pair must register through Moodle. If you need help in finding someone to pair with, please email me.

3.2 Date and Location

The earliest date to submit your work is February 1, 2022.

The **last** date to submit your work is **Sunday, February 27, 2022 at 23:59**.

Submission must be through Moodle. Please do not email submissions as this would prevent us from grading them.

3.3 Contents

You submission should be a single PDF file. The contents of the file should be clearly separated into 4 separate sections detailed below.

Important! Sections A,B and C should only contain **valid**, **runnable** SQL code. If you want to add any text in these sections you must format it as valid SQL comments.

3.3.1 Section A - Schema

The first section in the PDF should contain SQL code that, when run, creates the new database, its tables and related SQL objects (keys, constraints etc.)

3.3.2 Section B - Data

This section should contain SQL code that, when run, populates the database with the example data.

3.3.3 Section C - Queries

This section should contain the SQL queries that are described in 2.3 above. Add clear comments above each query to indicate the question number (2.3.1, 2.3.2 etc.)

3.3.4 Section D - Appendix

The last section, the appendix, is for you to add details about your design decisions or any other information relevant to your submission.

You are encouraged to include a screen shot of an EER Diagram of your database but this is optional and not required.

---

4. Grading

The grade for this project accounts for 60% of the total course grade. The other 40% coming from your home exercises. A minimum grade of 60 in the project is required for passing the course.

Submissions will be graded on the following principles:

4.1 Correctness and Completeness

Submissions are expected to fulfill all business requirements detailed above correctly.

4.2 Relational Design Principles

Submissions are expected to follow relational design principles taught in the course. For example:

- Correctly identifying and implementing entities and their relationships.
- Enforcing referential integrity
- Avoiding the 3 anomalies
- Avoiding unnecessary duplication of data.
- Etc. etc.

4.3 Readability

Keep your code organized and easy to read. Using consistent naming and casing (upper/lower case) is recommended (but, as promised, **not** required).

---

5. Questions are most welcome!

If you have any questions or concerns over the project, its requirements or its submission, please don't hesitate to email me at uri.goldstein@post.idc.ac.il or Efi Pakani (TA) at efraimpaka@gmail.com. We will be more than happy to help!

Good luck!

Uri

- EOF -