Database Application Development
Spring 2020
Metro Card System

Overview

You will form groups of 4-5 to develop a database system for metro cards. You will design the database, insert some sample data, and implement a set of required tasks. Each task needs to be implemented as one or more Oracle PL/SQL procedures. You do NOT need to write a graphic user interface. You also need to provide statements to execute your procedures.

Assumptions:

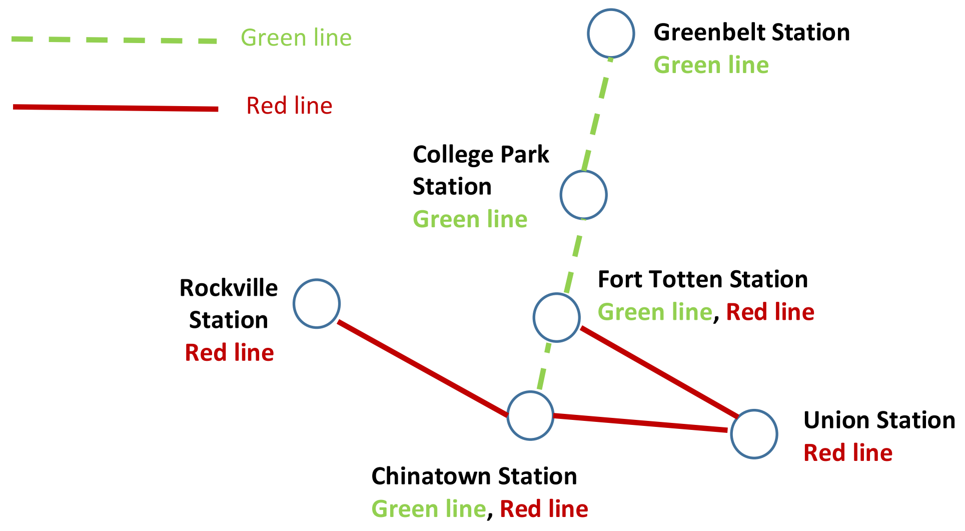
You can make the following assumptions in this project.

1. A passenger can create a rider account. Rider account has account id, email, password, name of passenger, and age of rider.
2. A passenger can buy one or more metro card. The metro card has card id, and balance, and what type of discount is has. These metro cards are linked to the passenger's rider account.
   Each card can be only linked to one account.
3. The system uses a trip table to store trips made using a metro card. This includes a system generated trip id, entrance station id, exit station id, entrance time, exit time, and cost.
4. Each trip costs a fixed amount if there is no discount. Discount is given to children (under 12) and senior (over 65). The discount rate and regular rate are stored in a table. You can use a discount id to represent different types of discount. E.g. 1 for regular, 2 for children, and 3 for senior.
5. The system stores transactions of adding money to a metro card. The transaction needs to store transaction id, metro card id, time of addition, and amount added.
6. There are multiple lines and multiple train stations. Each line has a line ID, line name, and number of stations on the line.
7. Each station has a station ID, station name, address, and status (open or closed).
8. Each line has multiple stations and each station can be on multiple lines. So we use a line_station table to keep track which station is on which line. To specify the order of stations, each row in line_station has a sequence number. E.g., if station A has sequence number 1 on line B, it means station A is the first station of line B. Sequence number 2 means the second station and so on.
9. Each line has multiple schedules. You can think of a schedule as a row in a time table in the real world (a sample time table is given on next page). A schedule has a unique

schedule ID, line ID, and direction (in increasing order sequence number of stations or decreasing order).

10. Schedule_station table stores the scheduled arrival time of a train at a station on a given schedule. E.g., in schedule 1 the train may arrive at station 1 at 7:30 am and arrive at station 2 at 7:40 am, and so on.
11. To represent time during the day such as 7:30 am, we use interval day to second data ty.pe (i.e., count the gap from 12 am). E.g., 7:30 am is represented as interval '7:30:00.00' hour to second or interval '0 7:30:00.00' day to second.
12. A passenger can transfer to a different line at a station that is on multiple lines.

Here is some sample metro lines and stations:



Sample Time table (this is not a real table in the database)

Green line, direction 1 (in increasing order of station's sequence number)

Greenbelt	College Park	Fort Totten	Chinatown
7:30 am	7:40 am	7:50 am	8:10 am
8:30 am	8:40 am	8:50 am	9:10 am
…			

Green line, direction 2 (in decreasing order of station's sequence number)

Chinatown	Fort Totten	College Park	Greenbelt
7:30 am	7:50 am	8:00 am	8:10 am
8:30 am	8:50 am	9:00 am	9:10 am
….	…	…	…

Required Tasks

There are two types of tasks:

1. individual tasks
2. group tasks.

Individual tasks will be assigned to individual group members and will be graded individually.

Group tasks will be assigned to the whole group and graded group-wise. **The first two deliverables are also graded group-wise.**

Each member of the team should contribute more or less equally. It is unfair for a few members to do most of the work while others do less. You will be asked to evaluate your teammate's effort at the end of the project. The instructor will adjust the grade based on the evaluation. Normally if most of your teammates agree that you do not contribute at all or contribute too little (e.g., your group has 4 members and you contribute only 5%), you may lose up to 80% of your project grade. If your teammates agree that you contribute much more than anyone else (e.g., your group has 4 members and you contribute 40%), you may gain up to 20% of your project grade (but not exceeding 100% of project grade). Multiple peer evaluations will be conducted throughout the semester to determine the contribution of each team member.

Individual Tasks

Each member of the group chooses tasks for one member. If your team has fewer than five members, you don’t have to choose all tasks (e.g., if your team has 4 members, you can pick tasks for member 1 to 4 or 2 to 5 etc.)

Member 1: Account creation and login check.

Task 1: allow a passenger to create a rider account with input including email address, name, password, and age. First check whether email exists. If so, print a message that the account exists. Otherwise, create a new rider account with new account id and age and print out the new account id.

Task 2: login to an account with email address and password. If both matches an existing account, print a message login is successful. If email does not match, print a message no such account. If email matches but password does not match, print wrong password.

Member 2: Tasks for metro card.

Task 3: Create a procedure to allow a passenger to buy a metro card and add it to an existing account. Input includes account id, initial balance (payment).

- The procedure first checks if the input account id exists. If not, print out a message saying that the account does not exist.
- Otherwise the procedure inserts a new row to metro card table with given account id, balance and a new card id (using sequence). In addition, please look up age of the account holder. If holder's age <= 12, set discount id to 2 (child). If holder's age >= 65, set discount id to 3 (senior). Otherwise the discount rate should be 1 (regular).

Task 4: Allow a passenger to add money to an existing card, with input card id and amount. The procedure first checks if there is a metro card with given card id. If not, print a message saying no such card. Otherwise, add input amount to balance of the card and insert a row to metro card transaction table with time as current time. Finally print out new balance.

Member 3: Display of card information.

Task 5: Create a procedure to allow a passenger to list all cards related to an account and print out card id and balance on each card. In case the account id is not valid, print out a message saying invalid account id.

Task 6: Create a procedure to allow a passenger to look up all transactions of a card in a given time period. Input includes card id, start date and end date. In case the input card id is invalid, print a message saying invalid card. Otherwise print out the transaction date and amount added for each transaction during the period.

Member 4: Add a trip.

Task 7: Add information about a trip into the trip table. Input includes card id, entrance station id, exit station id, entrance time and exit time.

- The procedure first verifies that the card id is valid. If not print an error message invalid card.
- Otherwise the procedure computes the cost of the trip based on discount id of the card and insert a new trip (with generated trip id) and print out generated trip id, computed cost, entrance station name and exit station name.
- Finally, the procedure updates the balance of the card by deducting cost of the trip from its balance.

Member 5: Display available trips.

Task 8: Create a procedure that given a time (just hours and minutes) and station name, list all schedules departing that station within X minutes after the given time. Here X is an input. The procedure first checks whether the input station name matches any existing station. If not, it prints an error message saying wrong station name. Otherwise it prints out schedule ID, line name, direction (1 or 2), scheduled arrival time at the station.

Hint: use interval day to second for input time and X.

Group Tasks

Task 9: write a PL/SQL function that given the name of an origin station, the name of a destination station, and ID of the line these two stations are on (assuming they are on the same line), returns the direction (1 for increasing order and 2 for decreasing order). You can assume that all station names and line id are valid.

Task 10: Create a procedure that prints out schedules a passenger can take from an origin station to a destination station around a given time assuming both stations are on the same line.

- Input includes a start time (you can use interval day to second type), a time gap (interval data type), the names of the origin and destination station, and a line name (assuming these two stations are on the same line),
- If any of the station name is invalid, the procedure prints out a message saying wrong station. If the line name is invalid, print a message saying wrong line name.
- If all names are valid, the procedure finds schedules on that line that is in the right direction (from the origin station to the destination station). You can use Task 9 to check this.
- The procedure also makes sure the arrival time at the origin station is between the start time and start time plus the gap. E.g., if start time is 7:30 am and gap is one hour, the schedule arrival time at origin station should be 7:30 to 8:30.
- Finally, the procedure prints out the schedule ID and scheduled arrival time at origin and destination station for each schedule.

Task 11: Find transfer stations.

Create a PL/SQL procedure with the input as a passenger's origin and destination station names. You can assume they are on two different lines and there is at least one station that the passenger can transfer between these two lines.

- The procedure first checks whether the two station names exist and whether they are on different lines. If not, print out an error message explaining the problem.
- Otherwise, the procedure prints out the transfer station name and line name and direction (1 or 2) from origin to transfer station, and line name and direction from transfer station to destination. For example, suppose input is Greenbelt (origin) and Rockville (destination). There are two possible transfer stations, Chinatown or Fort Totten. So your procedure should print out 'Take Green line in direction 1 transfer at Chinatown, and then take Red line in direction 2' and 'Take Green line in direction 1 transfer at Fort Totten and then take Red line in direction 2'.

- The procedure can use the function created in task 9 to decide direction.

Task 12: Create a procedure to print out total number of accounts, total number of cards, total spending (cost of all trips), average number of trips per account, the station appears most often as the entrance station in trips, and the station appears most often as the exit station in trips.

Deliverables:

There will be 4 deliverables. Delayed submission will result in a penalty of 30% of your score (e.g., if your score for part 2 is 20 but you are late, your score will be 14) within the first week after the deadline and will not be accepted after that. The final presentation (D3) is due at lab session time and no delay is allowed. All team members are required to participate and demo their own components.

Deliverable I (Group Deliverable)

1. **Due 2/24**. Project Management Schedule.
   a. Include team members and a timeline showing each phase of your project with its activities and time duration, for the entire effort.
   b. It is expected that every member should participate in all phases of the project.
   c. Please specify which task is assigned to which member (for group tasks it is still possible to assign a lead for each task).
   d. Activities should include system design, populating tables, writing code, testing code, running example queries, writing documents, preparing for presentation, etc. Smaller milestones shall be set for deliverable 3 and 4.
   e. This deliverable will be graded based on whether items a) to d) are included and whether the schedule is reasonable (e.g., enough time such as 2-3 weeks are left for testing and integration).

2. Design Document, which includes the following:
   a. ER diagram of the database. You do not have to follow exact notations of ER diagram, but need to show tables, columns, primary keys, and foreign key links.
   b. SQL statements to create database tables and to insert some sample data (at least 3 rows per table). Please include drop table and drop sequence statements before create table, create sequence and insert.
   c. Specification for each required task. The specification should include a description of input parameters and output (usually printing a message), and a

few test cases (normally there should be one normal case and a few special cases). You do not need to implement any of these procedures at this point.

Deliverable II (Individual Deliverable):

3. **Due 3/30**. Individual Member tasks of each member

Deliverable III (Group Deliverable)

4. Due **4/13**. All Group tasks

Deliverable IV (Group Deliverable)

5. **Due 4/27.**
   Please also upload final code through blackboard. The code should include:
   a. Drop table and sequence statements to drop tables if they exist (remember to use cascade constraints).
   b. Create table statements and create sequence statements
   c. Insert statements
   d. Create procedure statements (with code for the procedures). Each task can be implemented as one PL/SQL procedure (in the procedure you may call other procedures or functions). Please include some comments in your code explaining the major steps. You should use create or replace to avoid procedure name conflict.
   e. Test script to show that all your tasks work correctly. The script shall include some examples to test different cases. E.g., for task 1, one example for new user (email is not in database) and one example for existing user (using existing email). Please include:
      i. PL/SQL script to call the appropriate PL/SQL procedure for this task. E.g., exec procedure-name (parameter values)
      j. Explanation of what should be the correct output. The output could be updated tables (you can have some select statement to show the updated tables), some print out, etc.
      k. Make sure you have tested your examples from beginning to end. Remember that database tables may have been changed in the process. So, you may need to start with a clean database (i.e., right after you execute all the drop table, create table, and insert statements).

Deliverable V (Project Demo)

6. Due **5/4**. Presentation of database design and demonstration of each task.
   To demo each task, you need to prepare a couple of test cases, usually one normal case and the rest as special cases. For each test case, you need to be able to explain why your answer is correct (this can be typically done by showing some tables or screen output).

Grading Guidelines:
What I look for while grading demo:

- Clarity of presentation
- Timeliness and smoothness of demo.
- Correctness of each demoed task including explanation.

For deliverable :

- Existence of code
- Comments: Both descriptive and inline for every procedure/function
- Software quality
- Whether it is correct (giving correct results).
- Whether it is complete and clear.
- Efficiency of code. You shall not use too many SQL statements, and you shall put as much work as possible in SQL. For example, if you can do a join, do not use two select statements and then do a join in your program.
- Whether it has considered all special cases such as whether a user has already registered in task 1.

Tips:

1. Start early. Do not wait until last month to start coding. Do not wait until one week before the demo to start putting things together. Experiences show that more than 50% of time should be devoted to testing and putting things together.
2. Learn how to debug SQL and PL/SQL code. You can insert screen output statements to check intermediate results. Oracle also returns error messages and error code. You can google the error messages and error code to find possible causes. You may also use Oracle SQL Developer, which allows you to insert break points during debugging.
3. It is highly recommended to use SQL Developer rather than the web interface for the project.
4. Use homework, in class exercises, and programs in slides as templates of your PL/SQL program. For example, if you need to write a cursor, find a cursor example and use it as a starting point.