**IS 620 Advanced Database Projects**

Fall 2020

Group Project

Contact Tracing System

## Overview

You will be assigned into groups of three to four people in this project. Please read the whole document carefully before starting your project.

Your assignment is to design a contact tracing system. You will design the database, insert some sample data, and implement a set of required features. Each feature will be implemented as one or more Oracle PL/SQL procedures/functions. You do **NOT** need to write a graphic user interface.

**Assumptions:**

You can make the following assumptions in this project.

1. The database stores information about houses, each with a house ID, address, and zip code.
2. The database stores information about person, including person ID, house ID where the person lives, person name, phone number, and status where 1 means currently tested positive (i.e., the most recent test result is positive), 0 means currently tested negative (i.e., the most recent test result is negative), and null (no test so far so status is unknown).
3. The database stores information about a person's all past tests, including person ID, test date, and result (1 means positive and 0 negative).
4. The database stores information about events (could be gathering, party, etc.). Each event has an event ID, event name, date, and address.
5. The database stores information about a person participated in an event.
6. The database stores information about flights, including flight ID, flight date, flight number (a varchar type e.g., 'DL 345' is a delta 345 flight).
7. The database stores IDs of people on a specific flight.

**Individual Features:** There are 8 features grouped in feature sets for member 1, 2, 3, and 4. Each member need to implement feature set for one member. If your group only has 3 members then only 3 feature sets need to be implemented (e.g., your group can choose features for member 1,2 and 3).

**Member 1:**

**Feature 1:** add a person to the database. Input includes name of person, home address and zip code, phone#, and set status to null. Please do the following:

1) check if a person already exists with same name and phone number. If so print a message: 'person already exists!'

2) otherwise, check whether there is a house with same address and zip. If so, just insert a row into person table with input name, phone number, the id of house with that address and zip, and status as null. Print a message saying person inserted with existing house and print out newly assigned person.

3) if the house does not exist as in step 2), insert a row into house table with a new house id, input address and zip. Next insert a row into person table. Print a message saying a new house created with the newly assigned house id, and a person inserted with the new person id.

**Feature 2:** List people's names and phone numbers for those whose current status is positive. Then for each such person P, list names, phone numbers, and status of people who live in the same house as P (do not print information for P again).

**Member 2:**

**Feature 3:** Enter a new test result and update current status. Input includes person ID, test date and test result.

1. First check whether there is a person with the input person ID. If not please print an error message.
2. Next check whether an existing test exists with the same person ID and test date, if so just update the test result. Otherwise insert a new row with input person ID, test date, and test result into person\_test.
3. Finally check whether this is the latest test result for the same person. if so please update the status of that person in person table to the new result.

**Feature 4:** List people's names and phone numbers for those whose current status is positive. Then for each such person P, list names and phone numbers of people who were on the same flight with this person in the last X days where X is an input parameter. Please do not include P in the result. You can use sysdate as current date.

**Member 3:**

**Feature 5:** Enter a new event. Input includes event date, name and address. First check whether an event with the same name, date and address exists. If so print a message that the event exists. Otherwise insert a row into event table with input date, name and address and print out event id.

**Feature 6:** List people's names and phone numbers for those whose current status is positive. Then for each such person P, list names and phone numbers of people who have attended the same event with that person in the last X days. X is an input parameter. Do not include P in the result.

**Member 4:**

**Feature 7:** Enter a list of people attending an event. Input includes event ID and list of person ids.

1. First check whether an event with the input ID exists. If not print a message saying the event does not exist.
2. Next check each input person ID, if there is no person with that ID, print a message saying the person does not exist.
3. Otherwise for each input person ID, check whether the person id and event id combination are already in event\_person table. If so print a message 'No need to insert'.
4. If the combination does not exist, insert a row into person\_event table for that input person ID and event ID.

Hint: use varray data type as input parameter.

**Feature 8:** Given the name and phone number of a person, print out test dates and test results of this person.

1. Please first check whether the person table has such a person with the input name and phone number. If not, please print an error message 'No such person'.
2. If the person does exist, print out the results sorted by test date in descending order.

**Group features:** all group features need to be implemented.

**Feature 9:** (This is a complex feature and the amount of work is equivalent to two features). Given a person ID and a test date D that person was tested positive, print out the suspected chain of transmission. The input also includes two X and Y (both are number of days and used in 2 b and 2 c below). Steps 2) to 4) below finds the people who may get infected by the input person.

1) First please check whether there exists a person A with the input person ID and tested positive on the input date D, if not print a message in valid input;

2) otherwise, find close contacts of the person A. Close contact includes the following 3 cases:

a) Those living in the same house as the person A

b) Those attended the same event between the input date test date D minus X and D. For example if D = date '2020-10-1', and X = 30 day, the event date should be between date '2020-9-1' and date '2020-9-30'

c) Those on the same flight with the person A between date D minus X and D (X as input parameter).

Note that person A should not be counted as his or her own close contact.

1. print out names of people who may have get infected by person A. Such person B is a close contact of A (found in step 2) and tested positive between D+1 to D + Y days (Y as input parameter) but never tested positive before D (i.e., B is A's close contact and tested positive shortly after A tested positive). Please also print why the person B qualified as close contact of A (in the same house, on same flight, or attend same event with A) and the date when B tested positive between date D to D+Y. If there are more than one such test dates pick the earliest in the range. It is fine to print the same person twice if that person qualified as A's close contact in multiple ways (e.g., both in the same house and took the same flight).

For example, if person Susan is Jeff's close contact and the input date D is date '2020-9-30', Y is 30 days. Then Susan will be in the result if she tested positive between date '2020-9-30' and date '2020-10-30' (D to D+Y), but never tested positive before date '2020-9-30'.

4) For each unique person B identified in step 3), continue step 2) to 3) to print close contacts of B using B's person ID, the date B tested positive between date D to D+Y, and X and Y as input.

For example, let X=Y=30 days, and Jeff's ID is in initial input and D = date '2020-9-30' (i.e., jeff tested positive on D). If Susan is Jeff's close contact and Susan tested positive on date '2020-10-15', the program should look for Susan's close contact with new D=date '2020-10-15' (Susan's test date). If a person Alice is Susan's close contact (e.g., on the same flight as Susan from date '2020-9-16' to '2020-10-15'), and Alice tested positive on a date between date '2020-10-15' and date '2020-11-14', then Alice will be returned next as she may get the virus from Susan. Next we will use Alice's person ID and test date as input and continue.

5) Stop when no new person is found.

Hint: You can implement step 2,3,4,5 as a recursive procedure (the procedure will call itself inside its body) to find the possible transmission chain. You can write another procedure to implement step 1) and call the recursive procedure. Be careful do not let the recursive procedure run infinitely. E.g., the following mistakes will lead to infinite execution: if you include input person A in result of step 4) then your code will run infinitely because you assumed A infected A, which will never stop. Another case is you forgot to check the test date condition where if A infect B, B's test date must > A's test date. Otherwise you may get infinite loop (A may infect B, B may infect A, and so on).

You can use two varray to store close contact's person id and test date, or use a temporary table.

**Feature 10:** Case Statistics. Please print out the following statistics:

1) total number of people who has positive status in the database, and number of such people in each zip code.

2) total number of distinctive people tested positive in the last X days where X is an input. If a person tested multiple times and more than one test was positive in the time range, that person only counted as one.

3) accumulated number of people tested positive in each zip code. Accumulate count includes every distinctive person who was tested positive at least once.

**Feature 11:** Daily stats. Given a start date and end date, print out daily statistics in this period.

1) Print out daily accumulated positive cases during the period. For each date, the daily accumulated positive cases equals number of distinctive people tested positive by that date.

2) Print out daily new cases. A new case is someone who is not tested or never tested positive but tested positive on that date.

3) Print out daily current cases. Current cases include those who tested positive in the last test on or before the current date.

4) Print out daily accumulated recovered cases. Recovered case includes everyone who tested negative in the last test on or before that date, but tested positive before.

**Feature 12:** Find zip codes as hot spots. Hot spots are where new cases in a period of X days increase more than a threshold. Input includes D (a date), X (number of days) and Y (threshold). Do the following:

1) Compute number of new cases for each zip code between date D-X+1 day and D where a new case in a date range is a person who lives in the zip code, and tested positive during the date range (D-X+1 to D), but never tested positive before D-X+1.

2) for each zip code, compute the number of new cases in the same zip code in the date range of D-2\*X+1 to D-X (day).

3) check if the count in step 1) is greater or equal to the count in step 2) \* (1+Y). If so print out this zip code as hot spot. Note that if count in step 2) is zero then that zip code also qualifies if count in step 1) is not zero.

**Deliverables:**

There will be 4 deliverables. D1, D2, D4 will be due midnight of the due date. D3 is a demo. 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). The final presentation (D3) is due at class time and no delay is allowed.

1. 10%. Due 9/16. Project Management Schedule.
   1. Include team members and a timeline showing each phase of your project with its tasks and time duration, for the entire effort.
   2. It is expected that every member should participate in all phases of the project. For example, every member should be involved in writing the code.
   3. There are 8 individual features and 4 group features. Please indicate which group member will do which set of individual features and how your group plan to work on group features.
   4. Tasks 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.
   5. This deliverable will be graded based on whether items a) to d) are included and whether the schedule is reasonable (e.g., enough time is left for testing and integration).
2. 25%. Due 10/14. Design Document which includes the following:
   1. ER diagram of the database. You don’t have to follow exact notations of ER diagram, but need to show tables, columns, primary keys, and foreign key links.
   2. SQL statements to create database tables and to insert some sample data (at least 5 rows per table).
   3. Specification for each required feature. The specification should include a description of input parameters and output (usually screen outputs), and an example of how a user can use this feature (e.g., exec XXX(…) where XXX is the procedure name). You don’t need to implement any of these features at this point.
3. 30%. Due 12/14. Demonstration. Your work will be demonstrated to the class in real time, where you will present the design of your system and you will run a demo. You don’t need to submit anything.
4. 35% Due 12/14. The code should include:
   1. Drop table statements to drop tables if they exist (remember the order of drop should be inverse of create). Drop sequence statements.
   2. Create table statements, create sequence statements.
   3. Insert statements
   4. Create procedure statements (with code for the procedures). Each feature can be implemented as one or multiple 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.
   5. Demo script to show that all your features work correctly. The script shall include some examples to test different cases. E.g., for feature 1, one example for new user (email is not in database) and one example for existing user (using existing email). Please include:
      1. PL/SQL script to call the appropriate PL/SQL procedure for this feature. E.g., exec procedure-name(parameter values)
      2. 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.
      3. 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).

**Grading Guidelines**

What I look for while grading software code (deliverable 4):

1. Existence of code and whether all code can be compiled without any error.
2. Comments: Both descriptive and inline for every procedure/function
3. Software quality
   1. Whether it is correct (giving correct results).
   2. Whether it is complete and clear.
   3. 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.
   4. Whether it has considered all special cases such as whether a user has already registered in Feature 1.

Regarding the presentation of your project: Each student must participate in the project demonstration by presenting to the entire class some slides. You will be graded on:

1. Timeliness of presentation
2. Presentation Style
3. Demo (running the code)

For the demo, you will be graded on the following items:

1. Existence of tables and data. You need to have at least 5 rows in each table.
2. The correctness of features. This can be shown by checking whether the screen output is correct and the database has been updated correctly.

Each member of the team shall 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). A peer evaluation will be conducted at the end of the semester to determine the contribution of each team member.

Tips:

1. Be aware of you and your team members’ strengths and limitations. Some of you may have very little programming experiences, and some of you do. So each team shall have at least 2 people who are good at programming. Each team can discuss how to assign the workload fairly and at the same time matching the strengths of team members. Usually easier features can be assigned to members without much programming experience (they should take more responsibility on non programming part to ensure equal contribution). However, every member should write some code (3 features at minimal).
2. 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. Past experiences show that more than 50% of time shall be devoted to testing and putting things together.
3. Learn how to debug SQL and PL/SQL code. Most of time the error is from the SQL part of your code. So you can test SQL part separately (e.g., by copy & paste the SQL statement in a cursor and replace PL/SQL variables/parameters with values). 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.
4. It is highly recommended to use SQL Developer rather than the web interface for the project.
5. 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.
6. Make sure special cases are handled.
7. At demo time, different data in the database may lead to different results. So usually you will start with a standard database (with a fixed set of tables and rows), and keep track of the sequence of the demo (e.g., a course can only be scheduled if it has been added first).