Lab 8: PL/SQL

Due date: Thursday, November 29, midnight

Assignment Preparation

The main part of this assignment is to be done in teams. The remaining part is individual. You can organize yourselves in teams of 2-3 people each. Once you build your team, one member of the team needs to email me at dekhtyar@csc.calpoly.edu. The email message should have the header ``CSC 365: Lab 8". In the body of the message, please put the names and emails of all team members.

Lab Assignment

In a Nutshell

The lab consists of parts:

- **Part 1. (individual)** Fix all queries from Labs 4 and 6.
- **Part 2. (team)** Create a PL/SQL package of functions and procedures for the new AIRLINES dataset. This part tests your ability to extend the functionality of SQL with PL/SQL code for transitive closure (i.e., reachability).

AIRLINES Dataset

The AIRLINES dataset represents information about airlines, airports and connections between airports. There are three .csv files in the dataset.

airlines.csv contains a list of airlines with some information about them.

airports100.csv contains a list of 100 US airports.

flights.csv contains a list of flights (connections) between different airports. Each airline operates 50 routs. Each route connects two different airports *A1* and *A2*. The airline operates two flights on each route: one flight goes from *A1* to *A2* and one — from *A2* to *A1*.

More information about the AIRLINES dataset can be found in its README file. The AIRLINES dataset can be downloads from the class web page:

http://www.csc.calpoly.edu/∼dekhtyar/365-Fall2007/

Note, that the AIRLINES dataset is very simplistic. It does not record information about the arrival and departure times of the flights, neither does it provide any realistic data: all connections are randomly generated.

Part 1

Using the gradesheets from Labs 4 and 6, revisit your submissions and fix your queries to provide correct answers. For Lab 4, you can use the `row-count` file available from the course web page as the guide. If you have questions regarding specific Lab 6 queries, ask the instructor.

There is no deliverable for this part, but queries like those in Lab 4 and Lab 6 will be asked on the final exam, so, this part of the lab goes towards your final exam preparation.

This is an individual task, and should be done outside of the lab time, or after the team part of the lab is completed.

Part 2

You are asked to form groups of 2-3 people. The goal of each group is to build a PL/SQL package which provides important functions and procedures for the AIRLINES database. The functions will be used from SQL queries to the database. The procedures will be called from anonymous PL/SQL blocks.

The list of tasks is outlined below.

1. **Create Database.** Convert the AIRLINES dataset into an Oracle database. Follow the same basic procedures that you used in Labs 2 and 3. Your database schema should match the .csv file columns one-to-one. Create SQL scripts `AIRLINES-setup.sql` (CREATE TABLE statements), AIRLINES-insert.sql (INSERT INTO statements) and AIRLINES-cleanup.sql (DROP TABLE statements). `AIRLINES-insert.sql` should contain **all** INSERT INTO statements for all tables in the AIRLINES database. Make certain you populate databases in the right order (if table X has a foreign key onto table Y, then table Y needs to be populated first).

2. **Create package.** Create and test PL/SQL package `airline_ops`. The package shall consist of the functions and procedures described in the section below. You are allowed to declare any additions variables, constants, data types, functions and procedures, to help you build the mandatory functionality. You are also allowed to include package initialization code.

   Please, make sure you document each component of the package.

   The deliverable is an SQL script `airline_ops.sql` which shall contain `create package` and `create package body` statements.

String Values in .csv Files in AIRLINES dataset

Please note the following **important** detail. All character data in the `airports100.csv` table contains **one trailing space symbol**. This is of particular importance, because **three-letter airlines codes** are used as unique identifiers of the airports. For example, `'MRI'` is the unique identifier of the `Merril Field, Anchorage`. However, in the `airports100.csv` file, this value is `'MRI '`.

You have the following choices:

- strip trailing spaces before data gets inserted into the database.
- store data in the database with trailing spaces. When information is retrieved for the purpose of comparison, string trailing spaces from the retrieved values.
- store date in the database with trailing spaces. When information is retrieved for the purpose of comparison, keep trailing spaces in the data retrieved from the datbase, and *add trailing spaces* to any strings used in comparison.

I recommend the first approach - there is only one place where this needs to be coded. In the other two approaches, the code will have to be written pretty much for every function/procedure.

Package airline_ops

Functions and procedures of the `airline_ops` package will take as input parameters identifying individual airports and airlines. Airports are **always** identified by their three-letter code. **Note**, input parameters will always be **three-letter codes**, *NOT* extended with a trailing space (i.e., `'MRI'`, NOT, `'MRI '`). Airlines may be identified either by their unique Id number, or by their name. This will be stated specifically in each case.

The following functions and procedures shall be implemented in the package `airline_ops`.

`procedure add_flight(Flight_Number in binary_integer, Airline in varchar2, Airport1 in varchar2, Airport2 in varchar2);` This procedure adds *two* flights to the table of flights: a flight from `Airport1` to `Airport2` with filght number `Flight_Number`, and a return flight from `Airport2` to `Airport1` with flight number `Flight_Number`+1. The following must be taken into account:

- `Airline` refers to the values in the third columns of `Airlines.csv` (e.g., `'Delta'` or `'JetBlue'`).
- Flight numbers must be kept unique for each airline. Your code must check if `Flight_Number` and/or `Flight_Number`+1 have already been used in the database, for the given airline. The procedure shall perform no insertions, and return, if a duplicate flight number insertion is attempted for an airline. At the same time, there may be multiple flights between two destinations, even run by the same airline.

`procedure list_destinations(Airport in varchar2);` This procedure takes as input an airport, and outputs, i.e., prints to the output buffer, **all** airports (except for itself) that are reachable form it. For each airport, the information from the first three columns of the `airports100.csv` file needs to be reported.

`procedure list_destinations_airline(Airport in varchar2, Airline in varchar2);` Takes as input the airport code and the airline (values from the third column of the `airlines.csv` file). Prints to the output buffer the list of all (excluding the input airport) destination airports (three first columns from the `airports100.csv`) for all airports reachable from the input airport using only flights of the specified airline.

`procedure list_destinations_legs(Airport in varchar2, connections in binary_integer);` Takes as input the airport code and the number of *connections* or *legs* desired. Prints to the output buffer the list of all (excluding the input airport) destination airports (three first columns from the `airports100.csv`) which can be reached from the input airport in `connections` legs *or fewer*.

That, is, `list_destinations_legs('MRI',2)` shall print all airports reachable from `'MRI'` in one step (i.e., via a direct flight) and all airports reachable from `'MRI'` with one transfer (i.e., in two legs). `'MRI'` itself should be excluded from the output.

`procedure path(Airport1 in varchar2, Airport2 in varchar2);` This procedure takes as input the source (`Airport1`) and the destination (`Airport2`) airports and produces and prints to the output buffer, the list of flights a passenger would need to take to get from `Airport1` to `Airport2`. *Only one path* between the airports needs to be reported. Report each flight on a separate line. Specify the code for the source airport, the code for the destination airport, flight number, and the name of the airline (full name: the second column from the `airlines.csv` file).

Note: there is no requirement that the reported path is the shortest, but you **must** ensure that the reported path does not contain loops.

If `Airport2` is not reachable from `Airport1`, the procedure shall print to the output buffer the appropriate diagnostic statement and graciously return.

`function connected(Airport1 in varchar2, Airport2 in varchar2) returns boolean;` this function returns `true` if there is a path between `Airport1` and `Airport2`, and `false` otherwise.

`function connected_airline(Airport1 in varchar2, Airport2 in varchar2m Airline in varchar2);` This function returns `true` if there is a path between `Airport1` and `Airport2`, that uses only flights by airline `Airline`. It returns `false` otherwise.

`procedure find_airlines(Airport1 in binary_integer, Airport2 in binary_integer);` takes as input the airport codes for the source and the destination airports and prints to the output buffer the names of airlines (full names) a passenger must use to travel between the two airports. (basically, this procedure needs to find a path between the airports, if one exists, and output the names of all airlines whose flights are on the path).

Extra Credit

Extra credit is assessed for the following:

- Implementing the shortest path algorithm for the `path` procedure and for the `find_airlines` procedure. (a 50% markup for the score for each procedure).
- Implementing one or more components of the package using recursion. (25% markup for each recursive procedure/function).
- Implementing, for each procedure of the package, a twin *function* which, instead of printing the result to the output buffer, returns it as a table. (Each twin function earns a bit of extra credit independently of other twin functions). (each function earns the same amount of points as its twin procedure).

Submission Instructions

Email instructor a `.zip` or `.tar.gz` file containing the following files:

- `AIRLINES-setup.sql`
- `AIRLINES-insert.sql`
- `AIRLINES-cleanup.sql`
- `airline_ops.sql`
- `README`

Each file must contain a comment block at the top listing all members of the group.

`README` file shall contain the specifications for the extra credit functionality, if any has been comleted.