BMEG3120 Assignment, Fall 2013

In this assignment, you need to design and implement a database for a flight management system.

Basic Information to be Captured:

- **Airport:** For each airport, its name and country.
- **Airline:** For each airline, its name and country.
- **Flight:** For each flight, its flight number, airline, departing and destination airports, mileage, and departure and arrival time (including dates and hours).
- **Passenger:** For each passenger, her/his passport number, country, and name.
- **Travel History:** The flight records of all passengers.

Requirements. In addition to all the facts of common sense (e.g., a country can have multiple airports, different customers can have the same name, the mileage of a flight never changes, etc.), your design must adhere to the following constraints:

- Every flight belongs to a unique airline.
- The origin and destination of each flight are fixed throughout the history.
- No two passport numbers from the same country can be identical.

You need to implement your design in ORACLE to prevent the following errors:

- The origin and destination airports of a flight are the same.
- The arrival time of a flight precedes its departure time.

To do so, you will need to specify `CHECK` clauses in your table creation statements. See the appendix for information about `CHECK`.

What to Submit: Email the TA (*jhgan@cse.cuhk.edu.hk*) a report in pdf (with your student id as the file name) that contains all the information below:

- The minimal cover of the set of functional dependencies that your design is based on.
- Your ER diagram.
- The tables of your design, which must be dependency preserving. Furthermore, all tables must be in 3NF.
- ORACLE statements for creating your tables.
- SQL statements to accomplish the tasks below:
  - **Task 1:** Find all the countries where the HK citizen with HKID A123456(7) has ever landed.
  - **Task 2:** For each passenger, display her/his name, total mileage traveled, and total flight time.

- **Task 3:** For each airline, display its name, and the total number of distinct passengers it has ever served.

- Screenshots of the above queries' outputs in ORACLE. For this purpose, you need to insert into your database some test tuples that allow the instructor to observe the correctness of your query results. All the test tuples should also be described in your report.

Appendix

Formats of Time Values: ORACLE supports a data type called *date* for attributes, as illustrated in the example below:

create table dummy (id integer, d date)

The following statement inserts a tuple into the table:

insert into dummy values (1, to_date ('2010-2-12 10:20:30', 'YYYY-MM-DD HH24:MI:SS'))

"To_date" is a function that converts string '2010-2-12 10:20:30' to a date. Note how the function's second parameter 'YYYY-MM-DD HH24:MI:SS' specifies the date's format. In ORACLE, you can do subtraction between two dates, which will return a real value in the unit of days (e.g., 1.5 means a day and a half). For example, after the above insertion, the following query

select d − to_date('2010-2-11 4:20:30', 'YYYY-MM-DD HH24:MI:SS') from dummy

returns 1.25.

Furthermore, ORACLE supports direct comparisons of dates. Specifically, if d_1 and d_2 are two date values with d_1 preceding d_2, then $d_1 < d_2$ returns true while $d_1 > d_2$ returns false.

Clause `CHECK` in `CREATE TABLE` Statements: This clause permits you to define a simple constraint that needs to hold on a table at all times. Whenever the table is to be updated, the database will examine the constraint automatically, and allows the update to proceed only if the constraint will not be violated. You need to learn the syntax of this clause from this page: *http://www.techonthenet.com/oracle/check.php*.