CECS 323 Term Project

Spring 2020

**Project Description**

You are designing a database to track airline related information. I'm a frequent flyer and have found that existing airline reservation systems just don't provide the data and reports that I need to handle all my travel so I've hired you to create a specialized airline database just for me. Let's name this system "mifly".

In my travels, I have flown into and out of many different airports. Each of these airports has a name, a location, and an FAA-approved abbreviation. For example: Dulles Airport, Washington DC, IAD or Kennedy Airport, New York, JFK. Sometimes an area is so large that is can be serviced by multiple airports. If my destination again happens to be the Washington DC area, I can fly into Dulles airport, Reagan National Airport or Baltimore Washington International Airport. New York is serviced by Kennedy, La Guardia and Newark.

An airline can fly into and out of many airports. Each airline has a name and is headquartered in one particular city. An airline headquartered in the United States can either fly locally (limited to a portion of the country), domestic (the entire United States) or International (outside of the United States).

An airline can own any number of planes. These planes are made by a specific manufacturer with a specific model number (ex. Boeing 747) and hold a set number of passengers. Each of these airplanes have a tail number assigned by the FAA (ex. TB134). Some of the airplanes are even given names. Recently JetBlue airline had a contest to name each of their planes (ex. Bluebird).

Travel on an airplane is referred to as a flight. Each flight is arranged to leave a particular airport and return to a different airport. The flight is identified by the airline and a number assigned by the airline itself. This flight number is associated with a specific departure time and an arrival time. The same flight number is used any day of the week that an airlines flies between the same airports at the same time.

Each flight is assigned a specific crew. Each crew is composed of one pilot, one co-pilot, one navigator, and anywhere from two to five flight attendants according to the number of passengers an airplane holds and the length of the flight.   
Due to the tightened security restrictions, each crew member under goes a background check by the FAA and is assigned an FAA number before being allowed on an airplane.

With the rising cost of gasoline, some airlines are now charging for water, pillows and blankets, and checking bags on a per flight basis on local or domestic flight.

You will also need to support generation of an Incident report - this allows a flight crew employee to file a report related to any type of incident that occurred on a flight. It could be a problem, a concern, an emergency, or recognition of good customer service. This report involves a particular flight, the type of incident, a description of the incident, the crew member reporting the incident, and the crew member involved in the incident.

You may assume that all flights are non-stop. You don't need to worry about changing planes or detailing portions of a flight.

Specific instances of a flight are the actual flights from one airport to another on a specific date. Although they are set to arrive and depart at specific times, the actual times can be different.

Model this enterprise using only the information supplied here. Do not model any processes not mentioned here such as payments, travel agents, customers, etc.

You should use your knowledge of airlines and flying in general, and you may want to pay new, closer attention to details of the operation of any airline or airport sites you may visit on the web as you are working on this assignment.

## **Additional Business Rules**

Each team member will add two more business rules (requirements) to the above business rules. You will have to provide some means to enforce these business rules in your database, either by means database constraint such as referential integrity, uniqueness constraint, not null constraint or the like. The business rule needs to be something that will show up in the model of your design. None of your additional business rules can contradict any of the business rules provide in the project definition.

Each team member will also have to create two more queries related to those business rules.

### **Output**

Output of the database must support the following products. You do not need to develop “pretty” printed or on-screen reports. You will run the views/queries in MySQL Query Browser in the lab.

* + The list of all airlines for a given airport.
  + The list of all flights for a given airline. You must be able to sort this list by starting location, destination, longest flight or shortest flight.
  + Flights that charge for extras (water, etc.)
  + The crew roster for each flight for each airline
  + The trips that are available if you do make one stop over
  + Management reports of mifly information, including arriving flights per city, departing flights per city, list of airlines in each service category, crews that fly multiple flights in a single day.
  + A list of all incident reports by flight
  + Flights that are scheduled to depart in three days
  + All flights that arrived in the busiest airport in the last week
  + Flights that departed more than 30 minutes late.
  + Two additional queries per team member.

# **Denormalization**

Denormalization is a conscious, deliberate change of a design from 3rd normal form to some lower normal form in order to meet some particular objective. You will describe the denormalization used in part one of the project. If you did any denormalization in your design, please include a separate paragraph(s) stating what you did and why. If you did not do any denormalization, state that and why.

For the purposes of this project, a denormalization may introduce redundancy of some sort into the physical structure of the dataset. For instance, merging a child table and its parent together and creating a subkey in the resulting table, or creating a multi-valued attribute while maintaining a junction table to represent those values as well would also introduce redundancy into the structure.

I will also consider any deliberate breaking of the design rules to enforce the model. For example, you may need to model a subclass even though it doesn’t have any attributes of its own, but it will enforce a constraint if you do.

### **Output**

This project does **not** require any graphical user interface front end. All of the interaction between your database and the user will be through the database client (NetBeans).

### **Deliverables**

Your work will be done in three parts with one submission of each part for each team. See the class schedule for due dates. See the breakdown of the Phases below for details.

All parts will be submitted through dropbox.

#### (25 points) The first part, design (conceptual), will consist of:

* Your additional business rules
* Class diagram.
* English description of all classes.
* English description of all associations.

#### (10 points) The second part (logical), will include:

* A revised design, based on feedback from the first part. This includes class diagrams and English descriptions of classes and associations.
* The relation scheme, based on the design.

#### (45 points) The third part(physical), will include:

* A revised design, based on feedback from the second part. This includes class diagrams and English descriptions of classes and associations.
* The relation scheme, based on the design and feedback from the second part.
* English description of all attributes.
* DDL used to create all the tables and the DML used to insert the data.
* Sample Output from the Queries – **Be sure to include sample output** in with your queries. The sample output could just be text, or screen captures. Call this file query\_output.docx or .pdf. Please include the SQL for the query immediately before the query output. This makes it immensely easier for me to review the query output.
* If you did any denormalization in your design, please include a separate paragraph(s) stating what you did and why. If you did not do any denormalization, state that and why.

## **Project Phases**

I have found that students package their deliverables in an apparently endless variety of possible configurations. This makes the grading of these projects much more difficult, particularly if the team makes more than one submission for either of the phases. By giving you the outline of what I expect to see in your drop boxes, I hope that the teams will each structure their deliverables in the same fashion and it will be easy for me to find what I need when it comes time to do the grading.

### **Phase 1 Draft** – Preliminary conceptual design

Think of this as a “mock” turn in. I will grade your work and provide you feedback as though you received a grade for your work, but it will not contribute to your final grade. Not until you get to Phase 1 Final will I award your team a “real” grade for phase 1. For Phase 1 Preliminary, turn in:

1. The description of your additional business rules. Please call this BusinessRules.docx, txt, …
2. The normalized UML class diagram – as an image.
3. English description of all classes and associations. Please call this ClassAndAssociationDefinitions.txt, or docx, or …

### **Phase 1 – Final conceptual design (25 points)**

Turn in everything that you turned in for Phase 1 Preliminary. This time it counts toward your final grade.

### **Phase 2 – Logical Design (10 points)**

1. Everything that you turned in for Phase 1.
2. The relation scheme diagram. Please pay attention to the layout of your relation scheme diagram. Please use the same layout in the relation scheme diagram that you did in the UML diagram. If a given class is in the upper left-hand corner of your UML diagram, put the corresponding relation scheme in the same place in the relation scheme diagram. This makes my job enormously easier.

### **Phase 3 – Physical Design (45 points)**

1. Everything that you turned in for Phases 1 and 2.
2. The DDL – This is all of the create table statements. Please put the primary key constraint and the foreign key constraint into the create table statement. Please call this file create\_table.sql.
3. The DML – The insert statements used to populate the tables. Please remember to put the column names into your insert statements and use just one insert statement for each table. Order your insert statements so that the execution of the inserts meet all of the referential integrity constraints.
4. The queries – In all cases, remember:

* Each query is a **single** SQL statement.
* **Never** return just the ID of a given thing in your queries, always do any necessary joins so that you can display a proper name.
* I will dock points for using literals in your queries. For instance, use the now() function to get the current date when asked to find visits within the past year, do not use a literal and put in the due date of the assignment for the current date.
* Be sure that the sample data that you insert into your tables is adequate to return **some** data from each of these queries.

Sample Output from the Queries – **Be sure to include sample output** in with your queries. The sample output could just be text, or screen captures. Call this file query\_output.docx or .pdf. Please include the SQL for the query immediately before the query output. This makes it immensely easier for me to review the query output.

1. Description of the attributes – Each attribute in the UML model must have a description for it in Phase 3. Please call this AttributeDescription.docx, .txt … Do not bother defining your surrogate keys, those are obvious. On the other hand, please **do** define the derived attributes. Also, where possible, include sample values to illustrate what could go in as a valid value for your attributes. Organize your attribute definitions by class, and alphabetize the classes. This makes it much easier for me to find the proper definitions when I need them. When you define attributes quantities, please supply the units in the definition.
2. The explanation of your denormalization and how you are going to enforce data integrity in spite of the redundancy of the denormalized structure. Please call this document Denormalization.docx, .txt, …

# **Lessons Learned by Your Predecessors**

* Unless you plan your time carefully, and stay on top of the deliverables, you **will** run out of time.
* Make sure that your relation scheme diagram is neat and easy to read early on. This will make everything much easier throughout the project.
* The data manipulation language (both the inserts and the selects) will take an immense amount of time if you are not careful. To keep that under control:
  + Read the queries in this rubric first, and plan your UML model around how you will answer the queries.
  + Split the sample data up between team members. This takes a long time to do because of all of the referential integrity constraints.
* Get the DDL done as soon as you are reasonably sure that the UML is solid. Start this at least two weeks ahead of the due date.
* Have someone in the team who has not done a given task, such as the queries, do a QA check before you move forward.
* Make use of a tool such as Google Drive to manage all of the files for the project.
* Use a group-messaging tool to keep in touch with each other.
* Within the team, be very clear who has what responsibility. Do not change things without going through the person who is responsible for that part of the project.
* Write out your create table statements in the proper order so that the referential integrity constraints are a part of the create table statement or immediately afterwards.
* When you get feedback from me (your friendly professor) be sure to share it with the whole team.
* Get feedback early. Face to face is always best, but I take E-mail, as you doubtless know by now.
* Make a “to do” list of the project deliverables, and track % completion on all of them from the very beginning.
* Read this rubric several times. There are no “throw away” statements here!

Above all, have fun!

## **Configuration Management**

In the business world, there is nothing more deadly to a project than sloppy or spotty configuration management. I will leave it up to the team how they manage the configuration of the various deliverables for this project, but please be certain that the team as a whole is 100% certain that what is going into the project drop box is what the team wants to turn in. I am not going to be sympathetic to the team that tells me the day after a given phase is due that the wrong version of something got turned in by mistake, or that one of the team members turned something in before the rest of the team was entirely ready.