Lee Mathew

CS-324

Thierry Petit

29 April 2021

**Final Report**

**Introduction**

In this project, the task was to see create a functioning UML and Relational model, and eventually a database, for a company with specific requests. This was done by taking in the information given by the company and seeing how it all connects. The database created is the “final form” of both the UML and relational model. In the database there are eight main classes: Employee, Aircrew, Aircraft, Flight, Departure, Route, Passenger, and Ticket. The majority of the classes have stayed the same The first database class, Employee, has stayed consistent with all rewrites of the models, as this is a titular class. From here there has been one major change to the entire model. This change is the presence the Ticket class. In the original model, the class was nonexistent. Originally, all of the attributes of the class Ticket was in the class Passenger. It was believed at first that these attributes belonged to Passenger. This is because a passenger buys a ticket, and things like ticket price, date of purchase, etc., are all intrinsically linked to the passenger who the ticket belongs to. The problem with this, was the question “Is it appropriate to keep all these attributes linked to passenger?”. A ticket holds valuable information, and while this is linked to the passenger who bought it, it ultimately because its own class. This change was done because while it could have worked, having the class Ticket helps to keep the database less cluttered and more uniform. Other than this major change, the model has stayed consistent for the majority of the project. Some minor changes had been on things like what attributes belong to what class, and if they belong in more than one, what purpose do they serve there.

**Queries**

With the database, queries can be used to understand different aspects of it, and to present the data in a more upfront matter. It can also help in grabbing information in a quicker way, rather than sifting through the database itself. One such query is a projection query, which is also follows: Select Distinct Aircraft from Flight;. This query brings forth the different aircrafts that are being used for flights. This is a helpful query as one can make an inventory of the aircrafts being used flights at that current time. The next query is Select Distinct Employee\_t from Employee;. This is another projection query, and works in the same manner as the previous query. In this one, the types of employees working can be brought up and observed. This could be useful for company so that they can keep track of the different types of employees that are hired. If there is ever a need for another type, or more of one type, this query can help to show that. The next query would work in par with this one. The third query is SELECT Skill, count(\*) from Aircrew GROUP BY Skill;. This query is slightly different from the last two. While the last two were just projecting the different types asked for by the query, this third query will not only project the Skill, it will also give count of how many times said skill shows up in the database. This is helpful especially for aircrew. The company specifically asks for a good mixture of all the different skill sets in one flight. Using this query can help to see how many of each skill is in a crew, and if there needs to be changes to the crew itself. The fourth query is SELECT First\_Name, Last\_Name from Passenger where First\_Name, Last\_Name in (SELECT First\_Name, Last\_Name from Ticket);. This particular query is unique in its own way. With this, two tables (or classes) are joined and referenced together to project certain information. In this particular query, the Ticket and Passenger class are joined and referenced together. From both, the query asks for the First and Last names of people. This is an especially helpful query as the company can use it to cross-reference which passengers are on flights with those who have purchased tickets. A more in-depth query might even show when the passengers are flying, and the ticket would help to confirm that information. The last query is about similar to the previous one: SELECT Departure\_date from Ticket where Departure\_date in (SELECT Departure\_date from Departure);. This query joins the classes Departure and Ticket, and asks to cross-reference the departure date between the two. This can be used as a simple confirmation to make sure that a passenger is flying out at the right time.

**Conclusion**

In conclusion, there are many uses for the database itself. While there are many different ways it could have been written, a strength of this particular database is that each of the company’s demands were considered and were made into the classes as seen in the database. Eight classes in total to make sure that each specification was seen to. A limitation could be that there are a lot of similar attributes between each class, and so it might be confusing when looking at it to understand what each one means in regard to its respective class. In the end, the overall database does its job of presenting the information given by the client.