Airport Database System by Group 12
Database Systems - 02170

Oliver G. B. Hansen - s194591
Sabina Bartule - s217175
Patrick H. Clemmensen - s194607
Grigorios Papaspyropoulos - s203223
Daniel J. Schütt - s194592

April 2022

Contents

1 Statement of Requirements

List of tables:

- Table of *cities* identified by a **city name** and a **country**
- Table of several different *airports* associated by **location**
- Table of commercial *shops* associated with an airport.
- Table of *airlines* identified by an **IATA code (airline code)**
- Table of *terminals* which would be associated with both an *airport* and an *airline*
- Table of *airplanes* identified by a **Model Nr.** and/or **model name** and associated with an *airline*
- Table of *flights* identified by an associated ***airport***, a ***time slot*** (and *terminal* ?)
- Table of *time slots* for flights to occupy identified by **time**.
- Table of *pilots* identified by a **pilot id** and **name**, and would be associated with an *airlines* as well as zero or more *flights*
- Table of *passengers* identified by a **passport number** as well as some personal information such as name, sex etc - passengers are only stored when they have one or more *tickets*
- Table of *tickets* identified by a **ticket number**

The airport management system is meant for organizing data across multiple airports. An *airport* has a certain capacity, and a certain amount of *stores*, which each have their own type, such as restaurant, liquor store etc. Each airport has associated *terminals*, comprised of a terminal number and the airport name.

There is a list of *airlines*, comprised of a name representing a corporation such as Ryanair. each airline have associated *airplanes*, which are defined by their modelnumber. The airline has a list of hired *Pilots*, who's got a name and a salary.

Each *Flight* from the airport is defined by a date, timeslots for arrival and departure, and lastly whether there are stops underway. A flight has an associated pilot, airport, airplane, airline, as well as associated tickets. *Timeslots* is the entire day divided into 15 minute intervals, so we can assure departure and arrival of flights. Examples are 6:15 pm, 6:30 pm, etc.

A *ticket* has a certain price and a specific seat. It is otherwise associated with a passenger and a flight. A *passenger* is defined by their Passportnumber, age, sex and lastly their email.

2 Conceptual Design

Here we introduce the entity-relationship diagram developed based on the statement of requirements listed earlier.

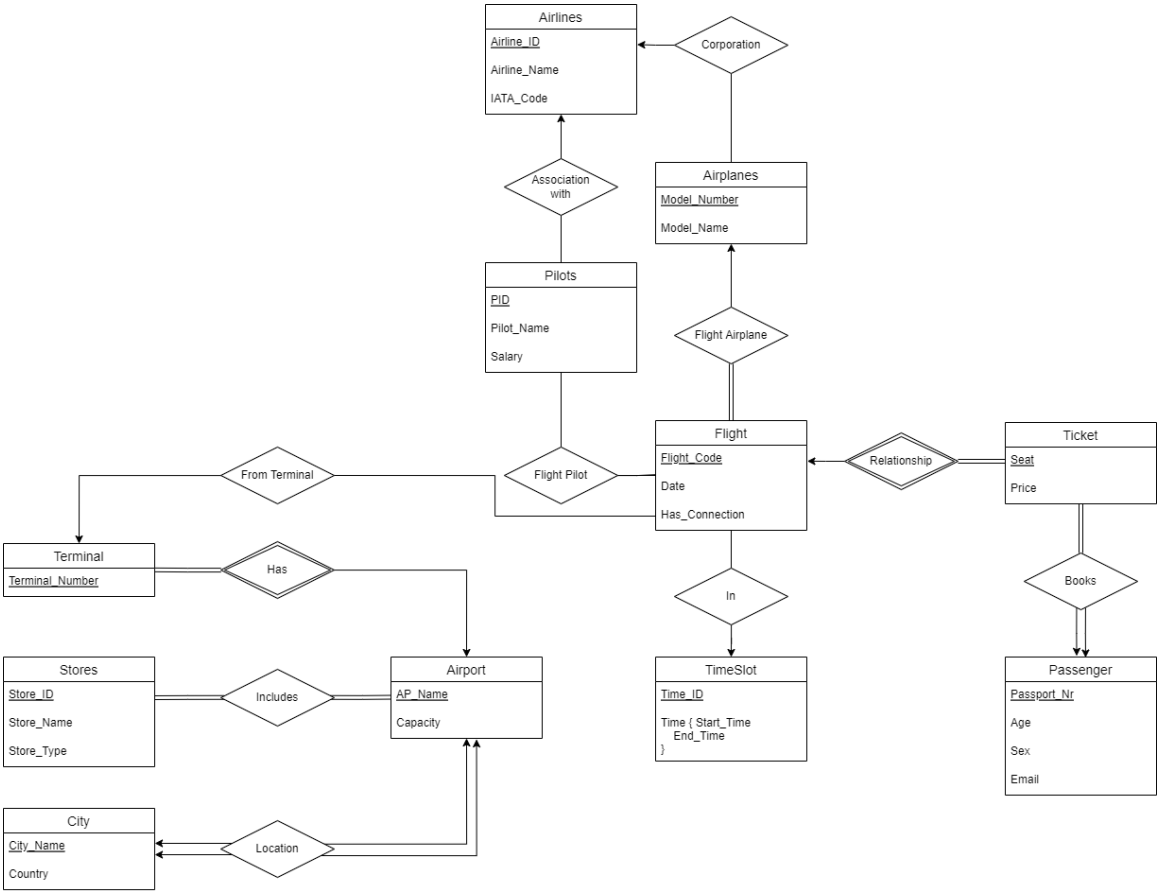

Figure 1: Entity-Relation Diagram for the Airport Database System. Weak entity sets primary key attributes were unable to be underlined by dashed lines, and will instead be informally described.

In figure 2, you can see how the different entities are related to each other.

As an *airport* consists of *stores* and *terminals*, and resides in a *city*, all those three entities will be in **full participation** with regards to an *airport*. *Stores* and *cities* are strong entities, however, *terminal* is a weak entity. The terminal is not only identified by its own number, but also the airport. As the database system only stores relevant data regarding external information, such as cities and passengers, we expect **full participation** from cities w.r.t., airports.

Passengers are uniquely identified by their passport number[1]

The same argument holds for *passengers* and *tickets*. Only persons who has purchased a ticket should be stored in the database, and all tickets should have an associated owner. Tickets are furthermore regarded as a weak entity set. This is because that a ticket is identified not only by the seat for which the ticket is for, but also which flight it is for. The price for a seat might change dependent on the flight.

Airlines are entities with relations to both an entity set of *pilots* and one of *airplanes*. Pilots can be unemployed, but only associated with one airline if they are employed. Airplane models are likewise only associated with one airline at a time. Airlines can have many airplane models and pilots, but might also have none or either.

Stores table, refers to the retail services of many products inside the airport for providing enhanced convenience to the people travelling. Stores might be different based on which airport they reside in, so we expect the store to require attributes from airports to form its primary key.

Tickets are identified by both the ticket ID and the seat of the passenger, so they use both of those information as primary keys.

Flights are at the center of it all, associated with many of the other entity sets. Each flight is uniquely identified by a single flight code, but refers to many other entities as well. The departure and arrival times from `time slots` and `airports`. Flights are associated with an airplane model, airline and some pilots. A flight might not necessarily at all times have an pilot assigned, but we require that it is associated with an airplane model. It also contains a relation to a terminal number from which the flight will depart from

[1]Passport numbers issued within a country are unique, however, they can possibly be identical between countries. We assume for this project that passport numbers can uniquely be assigned to a single passenger/person.

3 Logical Design

Here we introduce the relationship schema which design is based off of the conceptual design shown earlier.

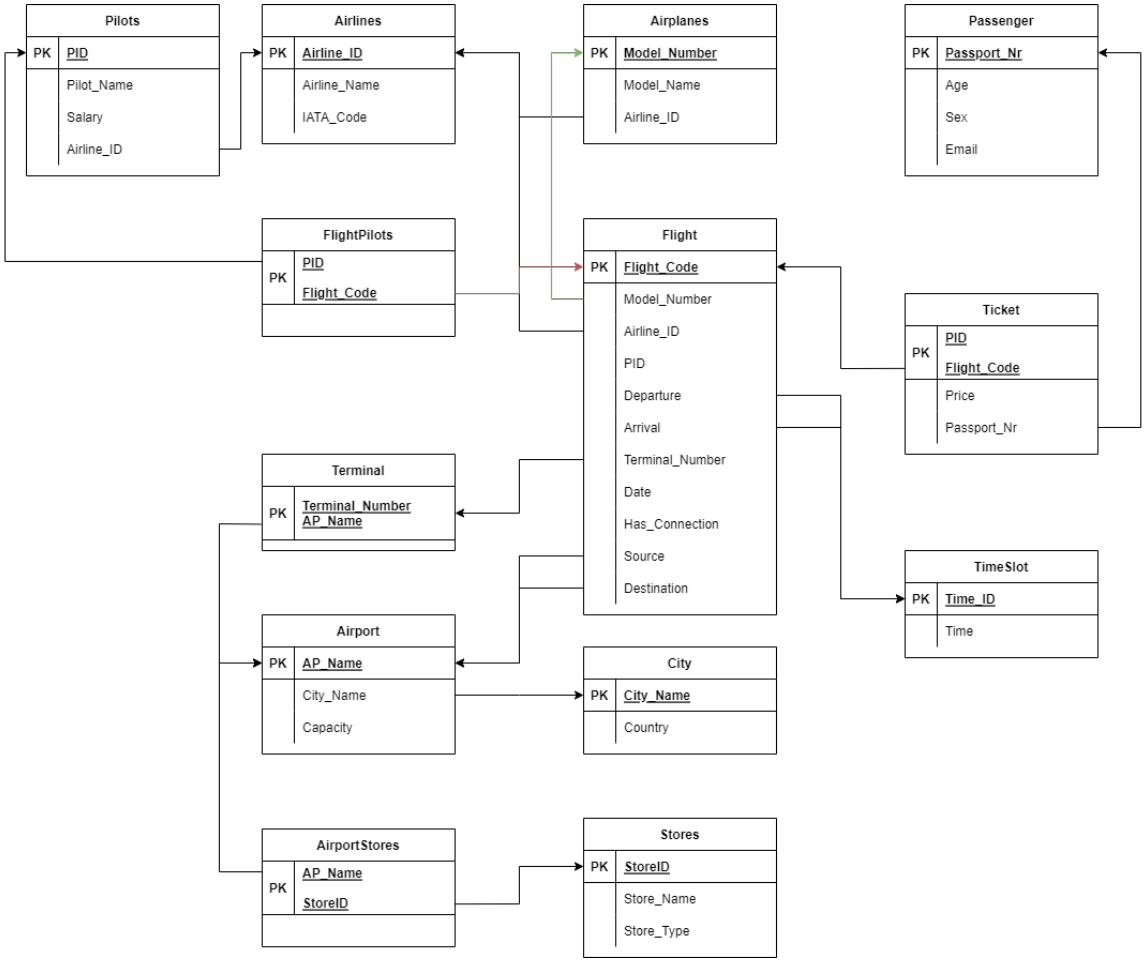

Figure 2: Relationship Schema for the Airport Database System. Color coded arrows are present as to minimize confusion for source and destinations of the arrows.

Starting with the one-to-one relation between cities and airports, the key from city is added as a foreign key attribute at the terminal relation. Either way would have been fine, as both has full participation, but it made sense to keep the reference in the airport for this database system.

All the many-to-many relationships have been converted such that a new relation table was added referencing both of the tables which had a many-to-many relation. Thus *AirportStores* was formed as a way to keep track of the many airport who could contain many stores - and vice versa. The same is true for pilots - there can be many pilots on a flight, and pilots can be assigned to several

flights which is captured by the relation *FlightPilots*.

The rest are many-to-one (or one-to-many) relationships. These are promptly converted such that the primary key from the one-side is added as an foreign key reference at the many-side. This can particularly be seen in the *flight* relation with source, destination, arrival, departure etc., as well as with *Ticket* which has a passport number associated with it.

4 Normalization

First Normal Form (1NF)

Since none of the tables in the relationship schema has composite attributes, all tables are automatically in 1NF.

Second Normal Form (2NF)

For a table to be in 2NF, all non-primary key attributes must depend on the entire primary key, not a subset of it. To determine this, we only have to regard tables which have a primary key that consists of two or more elements; *Ticket* and *Terminal*s

Terminal trivially satisfies the requirements of being in 2NF as it does not contain any non-primary key attributes.

Ticket also satisfies the requirements of the being in 2NF. The two attributes are related to both of the primary key attributes, as it depends on both the flight itself, but also which seat in the flight - (business, economy etc.).

All other tables are trivially considered to be in 2NF because their primary key consists of only one attribute or has no non-primary key attributes.

Third Normal Form (3NF)

A table is considered to be in 3NF when it is in 2NF and that all non-primary key attributes depend non-transitively on the whole primary key i.e., '$K \rightarrow B \rightarrow A$' would violate the requirement for 3NF.

Terminal, *AirportStores*, *FlightPilots*, *City* and *TimeSlot* can be considered in 3NF, as they are already in 2NF, and only relations whose number of attributes are larger than or equal to 3 has to be analyzed - otherwise there can be no transitive dependency.

Ticket The only attributes in the *Ticket* table that aren't primary keys are "Price" and "Passport_Nr". Price is strictly dependant on both the flight, and the seat. The price varies based on the flight, as destination/departure locations change prices. Seating can further change the price, in case it's a first class seat. Passport_number is dependent on both the seat and flight code as well, as it identifies which person has that seat. The passport_number isn't dependent on the price of the ticket, neither is the ticket price based on passport_number, so there are no transitive reliance.

Flights contains quite a lot of non-primary key attributes, but none of them transitively depend on the primary key through another non-primary key attribute. The "departure", "arrival", "source", "destination" and "date" attributes trivially depends solely on the "flight_code" itself. "Pilot" and

"Model_Number" also depend on the flight itself, and not through for example the "airline_id". The "airline_id" also is determined by the flight only. Thus none of the attributes depend on the primary key by transitive dependency.

The *Passenger* Consists of a passport number, age, sex and email address. Age isn't reliant on gender, nor email address, it is solely reliant on the individual, who is represented by their passport. Gender isn't based on age or email address, only on the owner of the passport. Email address isn't reliant on age or gender, and is linked to only a specific passport. Since none of these attributes depend on each other, they are not transitively dependent on one another. This makes *Passenger* a table in 3NF

The *Stores* relationship has two non-primary attributes to be considered. "Store_Name" and "Store_Type" are directly related to the ID of the store itself. As a store also can only exists as long as there is an airport.

Airlines only has two attributes to consider for it to be determined to be in 3NF; the name of the airline as well as its "IATA_Code". The name has nothing to do with the IATA code. The IATA code has to be dependent also on just the ID of the airline, and not the airline's name.

Pilots have 3 attributes, Pilot_Name, Salary, and Airline_ID. the name of the pilot is only based on the pilot itself. Similarly the airline the pilot works for isn't dependent on either salary, or the name of the pilot. Lastly salary isn't dependent on the name of the pilot. In the real world, salary is very likely dependent on the airline who's hired the pilot. However in our database pilots are salaried solely based on their abilities, and not on the specific airline. This makes the table be in 3NF

Airplane has two non-primary key attributes. The Model_Name and the Airline_ID. The owner of the airplane, which is the airline, has nothing to do with the name of the model. Only the plane itself has anything to do with the name of the plane. Thus the table must be in 3NF

airport has two attributes which aren't primary keys. The City name and the Capacity of the airport. The capacity isn't dependant on the name of the city, but of the airport itself. The name of the city which the airport lies in is only dependent of the location of the airport, and has nothing to do with capacity. Thus there is no transitive dependency, and the table is in 3NF.

5 Implementation

The database instance is pretty straight forward from the logical design of the system. To make the system more cohesive, we allow foreign key cascades in regards to updates, and deletions also sometimes does cascades, in case that it makes sense - for example with FlightPilots, if a flight no longer exists, we do not need to store pilots for an now non-existent flight. However, if a pilot needs removal, we require that the issue be resolved w.r.t., the flights that pilot might be on.

Below the whole database instance code can be seen:

```
CREATE TABLE airlines (
    Airline_ID int PRIMARY KEY,
    Airline_Name VARCHAR(50) UNIQUE NOT NULL,
    IATA_Code VARCHAR(255) UNIQUE NOT NULL,
    Created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    Updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);

CREATE TABLE stores (
    Store_ID INT UNIQUE NOT NULL PRIMARY KEY,
    Store_Name VARCHAR(250) NOT NULL,
    Store_Type VARCHAR(250) NOT NULL,
    Created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    Updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);

CREATE TABLE city (
    City_Name VARCHAR(250) PRIMARY KEY,
    Country VARCHAR(250) UNIQUE NOT NULL,
    Created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    Updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);

CREATE TABLE airplane (
    Model_number int PRIMARY KEY,
    Model_Name VARCHAR(50) NOT NULL,
    Airline_ID int,
    Created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    Updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    FOREIGN KEY (Airline_ID) REFERENCES airlines(Airline_ID) ON DELETE SET NULL ON UPDATE
        CASCADE
);

CREATE TABLE pilots (
    PID int PRIMARY KEY,
    Pilot_Name VARCHAR(50) UNIQUE NOT NULL,
    Salary FLOAT(9, 2) NOT NULL,
    Airline_ID int,
    Created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    Updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    FOREIGN KEY (Airline_ID) REFERENCES airlines(Airline_ID) ON DELETE SET NULL ON UPDATE
```

```
        CASCADE
);

CREATE TABLE airports (
    AP_Name VARCHAR(250) UNIQUE NOT NULL PRIMARY KEY,
    City_Name VARCHAR(250) UNIQUE NOT NULL,
    Capacity VARCHAR(250) NOT NULL,
    Created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    Updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    FOREIGN KEY (City_Name) REFERENCES city(City_Name) ON UPDATE CASCADE
);

CREATE TABLE terminal (
    Terminal_Number VARCHAR(50),
    AP_Name VARCHAR(250),
    Created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    Updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    PRIMARY KEY (Terminal_Number, AP_Name),
    FOREIGN KEY (AP_Name) REFERENCES airports(AP_Name) ON UPDATE CASCADE
);

CREATE TABLE timeslot (
    Time_ID int PRIMARY KEY,
    Time_Slot time(0) UNIQUE NOT NULL,
    Created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    Updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);

CREATE TABLE flights (
    Flight_code VARCHAR(255) PRIMARY KEY,
    Model_Number int NOT NULL,
    Airline_ID int NOT NULL,
    Departure INT NOT NULL,
    Arrival INT NOT NULL,
    Terminal_Number VARCHAR(50),
    Date_of_departure DATE NOT NULL,
    Has_Connection BIT(1),
    Source_location VARCHAR(255) NOT NULL,
    Destination_location VARCHAR(255) NOT NULL,
    Created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    Updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    FOREIGN KEY (Model_Number) REFERENCES airplane(Model_Number) ON UPDATE CASCADE,
    FOREIGN KEY (Terminal_Number) REFERENCES terminal(Terminal_Number) ON UPDATE CASCADE,
    FOREIGN KEY (Airline_ID) REFERENCES airlines(Airline_ID) ON UPDATE CASCADE,
    FOREIGN KEY (Source_location) REFERENCES airports(AP_Name) ON UPDATE CASCADE,
    FOREIGN KEY (Destination_location) REFERENCES airports(AP_Name) ON UPDATE CASCADE,
    FOREIGN KEY (Departure) REFERENCES timeslot(Time_ID),
    FOREIGN KEY (Arrival) REFERENCES timeslot(Time_ID)
);

CREATE TABLE passenger (
    Passport_Nr VARCHAR(50) PRIMARY KEY,
```

```
    Age int UNIQUE NOT NULL,
    Sex VARCHAR(50) NOT NULL,
    Email VARCHAR(50) UNIQUE NOT NULL,
    Created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    Updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);

CREATE TABLE ticket (
    Seat VARCHAR(50) NOT NULL UNIQUE,
    Flight_Code VARCHAR(255) NOT NULL,
    Price FLOAT(9, 2) NOT NULL,
    Passport_Nr VARCHAR(50) NOT NULL,
    Created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    Updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    primary key (Seat, Flight_code),
    FOREIGN KEY (Passport_Nr) REFERENCES passenger(Passport_Nr) ON UPDATE CASCADE,
    FOREIGN KEY (Flight_Code) REFERENCES flights(Flight_Code) ON DELETE CASCADE
);

CREATE TABLE flight_pilots (
    PID int NOT NULL,
    Flight_Code VARCHAR(255) NOT NULL,
    Created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    Updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    primary key (PID, Flight_Code),
    FOREIGN KEY (PID) REFERENCES pilots(PID),
    FOREIGN KEY (Flight_Code) REFERENCES flights(Flight_Code) ON DELETE CASCADE
);

CREATE TABLE airport_stores (
   Store_ID int NOT NULL,
    AP_Name VARCHAR(255) NOT NULL,
    Created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    Updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    primary key (Store_ID, AP_Name),
    FOREIGN KEY (Store_ID) REFERENCES stores(Store_ID) ON DELETE CASCADE,
    FOREIGN KEY (AP_Name) REFERENCES airports(AP_Name) ON UPDATE CASCADE ON DELETE CASCADE
);
```

6 Database Instance

Our Airport Database System we populated with the following data. Note, two extra columns were added: 'Created_at' and 'Updated_at' as these are good practice to include in databases.

AP_Name	City_Name	Capacity	Created_at	Updated_at
Charles de gaulle airport	Paris	9000	2022-04-01 16:16:31	2022-04-01 16:16:31
Copenhagen airport	Copenhagen	50000	2022-04-01 16:16:31	2022-04-01 16:16:31
Ferenc Liszt	Budapest	500000	2022-04-01 16:16:31	2022-04-01 16:16:31
Hamad International airport	Doha	12500	2022-04-01 16:16:31	2022-04-01 16:16:31
Haneda Airport	Tokyo	30000	2022-04-01 16:16:31	2022-04-01 16:16:31
Jorge Chavez international airport	Lima	10000	2022-04-01 16:16:31	2022-04-01 16:16:31
Luxembourg Findel airport	Luxembourg	10000	2022-04-01 16:16:31	2022-04-01 16:16:31
Oslo airport	Oslo	10000	2022-04-01 16:16:31	2022-04-01 16:16:31

Figure 3: The table "Airports"

Airline_ID	Airline_Name	IATA_Code	Created_at	Updated_at
0	SAS	SK	2022-04-01 16:16:44	2022-04-01 16:16:44
1	Ryanair	FR	2022-04-01 16:16:44	2022-04-01 16:16:44
2	Lufthansa Group	LH	2022-04-01 16:16:44	2022-04-01 16:16:44
3	Delta Air Lines	DL	2022-04-01 16:16:44	2022-04-01 16:16:44
4	China Southern Airlines	CZ	2022-04-01 16:16:44	2022-04-01 16:16:44

Figure 4: The table "Airlines"

Model_number	Model_Name	Airline_ID	Created_at	Updated_at
10120	Sukhoi Superjet 100	0	2022-04-01 16:16:48	2022-04-01 16:16:48
10121	Sukhoi Superjet 100	2	2022-04-01 16:16:48	2022-04-01 16:16:48
10320	Boeing 737-800	1	2022-04-01 16:16:48	2022-04-01 16:16:48
11230	Airbus A330-300	3	2022-04-01 16:16:48	2022-04-01 16:16:48
11231	Airbus A330-300	3	2022-04-01 16:16:48	2022-04-01 16:16:48
12310	Embraer 170/175	4	2022-04-01 16:16:48	2022-04-01 16:16:48
12311	Embraer 170/175	4	2022-04-01 16:16:48	2022-04-01 16:16:48

Figure 5: The table "Airplanes"

PID	Pilot_Name	Salary	Airline_ID	Created_at	Updated_at
0	Adams Armstrong	45000.00	0	2022-04-01 16:16:46	2022-04-01 16:16:46
1	James Hansen	40500.00	0	2022-04-01 16:16:46	2022-04-01 16:16:46
2	Robert Neil	43200.00	1	2022-04-01 16:16:46	2022-04-01 16:16:46
3	John Depp	45100.00	1	2022-04-01 16:16:46	2022-04-01 16:16:46
4	Michael Jackson	38800.00	2	2022-04-01 16:16:46	2022-04-01 16:16:46
5	William Macbeth	39600.00	2	2022-04-01 16:16:46	2022-04-01 16:16:46
6	David Anderson	36300.00	3	2022-04-01 16:16:46	2022-04-01 16:16:46
7	Kenneth Pløger	55000.00	3	2022-04-01 16:16:46	2022-04-01 16:16:46
8	Nadia Antonio	46200.00	3	2022-04-01 16:16:46	2022-04-01 16:16:46
9	Rabija Gandhi	41500.00	4	2022-04-01 16:16:46	2022-04-01 16:16:46
10	Sun Tzu	39500.00	4	2022-04-01 16:16:46	2022-04-01 16:16:46
11	Wi Tu Lo	38000.00	4	2022-04-01 16:16:46	2022-04-01 16:16:46

Figure 6: The table "Pilots"

Flight_code	Model_Number	Airline_ID	Departure	Arrival	Terminal_Number	Date_of_departure
100	10120	0	74	78	1	2022-04-01
101	10320	1	14	22	1	2022-04-01
102	10121	2	12	18	2	2022-04-01
103	11230	3	31	40	1	2022-04-01
104	11231	3	31	35	2	2022-04-01
105	12310	4	12	22	3	2022-04-01
106	10121	2	14	24	1	2022-04-01
107	12311	4	2	22	1	2022-04-01
108	12311	4	83	90	2	2022-04-01
109	10320	1	74	84	1	2022-04-01

Figure 7: The table "Flight"

Date_of_departure	Has_Connection	Source_location	Destination_location	Created_at	Updated_at
2022-04-01	0	Copenhagen airport	Ferenc Liszt	2022-04-01 16:16:51	2022-04-01 16:16:51
2022-04-01	0	Ferenc Liszt	Oslo airport	2022-04-01 16:16:51	2022-04-01 16:16:51
2022-04-01	0	Copenhagen airport	Jorge Chavez international airport	2022-04-01 16:16:51	2022-04-01 16:16:51
2022-04-01	1	Haneda Airport	Copenhagen airport	2022-04-01 16:16:51	2022-04-01 16:16:51
2022-04-01	0	Jorge Chavez international airport	Copenhagen airport	2022-04-01 16:16:51	2022-04-01 16:16:51
2022-04-01	1	Luxembourg Findel airport	Ferenc Liszt	2022-04-01 16:16:51	2022-04-01 16:16:51
2022-04-01	0	Oslo airport	Haneda Airport	2022-04-01 16:16:51	2022-04-01 16:16:51
2022-04-01	0	Charles de gaulle airport	Haneda Airport	2022-04-01 16:16:51	2022-04-01 16:16:51
2022-04-01	0	Ferenc Liszt	Oslo airport	2022-04-01 16:16:51	2022-04-01 16:16:51
2022-04-01	1	Charles de gaulle airport	Ferenc Liszt	2022-04-01 16:16:51	2022-04-01 16:16:51

Figure 8: The table "Flight" continuation

PID	Flight_Code	Created_at	Updated_at
0	100	2022-04-03 09:28:36	2022-04-03 09:28:36
1	100	2022-04-03 09:28:36	2022-04-03 09:28:36
1	106	2022-04-03 09:28:36	2022-04-03 09:28:36
2	101	2022-04-03 09:28:36	2022-04-03 09:28:36
2	106	2022-04-03 09:28:36	2022-04-03 09:28:36
3	101	2022-04-03 09:28:36	2022-04-03 09:28:36
3	102	2022-04-03 09:28:36	2022-04-03 09:28:36
4	101	2022-04-03 09:28:36	2022-04-03 09:28:36
5	103	2022-04-03 09:28:36	2022-04-03 09:28:36
5	104	2022-04-03 09:28:36	2022-04-03 09:28:36
5	107	2022-04-03 09:28:36	2022-04-03 09:28:36
6	103	2022-04-03 09:28:36	2022-04-03 09:28:36
7	104	2022-04-03 09:28:36	2022-04-03 09:28:36
8	105	2022-04-03 09:28:36	2022-04-03 09:28:36
9	105	2022-04-03 09:28:36	2022-04-03 09:28:36
10	108	2022-04-03 09:28:36	2022-04-03 09:28:36
11	109	2022-04-03 09:28:36	2022-04-03 09:28:36

Figure 9: The table "FlightPilots"

Passport_Nr	Age	Sex	Email	Created_at	Updated_at
BU35265464	26	MALE	passenger_bu@dtu.com	2022-04-01 16:16:42	2022-04-01 16:16:42
DA54416575	32	FEMALE	passenger_da@dtu.com	2022-04-01 16:16:42	2022-04-01 16:16:42
DE54546422	40	FEMALE	passenger_de@dtu.com	2022-04-01 16:16:42	2022-04-01 16:16:42
FR54416198	70	FEMALE	passenger_fr@dtu.com	2022-04-01 16:16:42	2022-04-01 16:16:42
GR23232311	25	MALE	passenger_gr@dtu.com	2022-04-01 16:16:42	2022-04-01 16:16:42
IT15413214	18	MALE	passenger_it@dtu.com	2022-04-01 16:16:42	2022-04-01 16:16:42
UK54613146	21	FEMALE	passenger_uk@dtu.com	2022-04-01 16:16:42	2022-04-01 16:16:42
US15444644	50	FEMALE	passenger_us@dtu.com	2022-04-01 16:16:42	2022-04-01 16:16:42

Figure 10: The table "Passenger"

Terminal_Number	AP_Name	Created_at	Updated_at
1	Charles de gaulle airport	2022-04-01 16:16:35	2022-04-01 16:16:35
1	Copenhagen airport	2022-04-01 16:16:35	2022-04-01 16:16:35
1	Ferenc Liszt	2022-04-01 16:16:35	2022-04-01 16:16:35
1	Hamad International airport	2022-04-01 16:16:35	2022-04-01 16:16:35
1	Haneda Airport	2022-04-01 16:16:35	2022-04-01 16:16:35
1	Jorge Chavez international airport	2022-04-01 16:16:35	2022-04-01 16:16:35
1	Luxembourg Findel airport	2022-04-01 16:16:35	2022-04-01 16:16:35
1	Oslo airport	2022-04-01 16:16:35	2022-04-01 16:16:35
2	Copenhagen airport	2022-04-01 16:16:35	2022-04-01 16:16:35
2	Ferenc Liszt	2022-04-01 16:16:35	2022-04-01 16:16:35
2	Hamad International airport	2022-04-01 16:16:35	2022-04-01 16:16:35
2	Jorge Chavez international airport	2022-04-01 16:16:35	2022-04-01 16:16:35
2	Luxembourg Findel airport	2022-04-01 16:16:35	2022-04-01 16:16:35
3	Ferenc Liszt	2022-04-01 16:16:35	2022-04-01 16:16:35
3	Luxembourg Findel airport	2022-04-01 16:16:35	2022-04-01 16:16:35

Figure 11: The table "Terminal"

City_Name	Country	Created_at	Updated_at
Budapest	Hungary	2022-04-01 16:16:27	2022-04-01 16:16:27
Copenhagen	Denmark	2022-04-01 16:16:27	2022-04-01 16:16:27
Doha	Qatar	2022-04-01 16:16:27	2022-04-01 16:16:27
Lima	Peru	2022-04-01 16:16:27	2022-04-01 16:16:27
Luxembourg	Luxembourg	2022-04-01 16:16:27	2022-04-01 16:16:27
Oslo	Norway	2022-04-01 16:16:27	2022-04-01 16:16:27
Paris	France	2022-04-01 16:16:27	2022-04-01 16:16:27
Tokyo	Japan	2022-04-01 16:16:27	2022-04-01 16:16:27

Figure 12: The table "City"

Time_ID	Time_Slot	Created_at	Updated_at
1	00:00:00	2022-04-01 16:16:37	2022-04-01 16:16:37
2	00:15:00	2022-04-01 16:16:37	2022-04-01 16:16:37
3	00:30:00	2022-04-01 16:16:37	2022-04-01 16:16:37
4	00:45:00	2022-04-01 16:16:37	2022-04-01 16:16:37
5	01:00:00	2022-04-01 16:16:37	2022-04-01 16:16:37
6	01:15:00	2022-04-01 16:16:37	2022-04-01 16:16:37
7	01:30:00	2022-04-01 16:16:37	2022-04-01 16:16:37
8	01:45:00	2022-04-01 16:16:37	2022-04-01 16:16:37
9	02:00:00	2022-04-01 16:16:37	2022-04-01 16:16:37
10	02:15:00	2022-04-01 16:16:37	2022-04-01 16:16:37
11	02:30:00	2022-04-01 16:16:37	2022-04-01 16:16:37
12	02:45:00	2022-04-01 16:16:37	2022-04-01 16:16:37
13	03:00:00	2022-04-01 16:16:37	2022-04-01 16:16:37
14	03:15:00	2022-04-01 16:16:37	2022-04-01 16:16:37
15	03:30:00	2022-04-01 16:16:37	2022-04-01 16:16:37
16	03:45:00	2022-04-01 16:16:37	2022-04-01 16:16:37
17	04:00:00	2022-04-01 16:16:37	2022-04-01 16:16:37
18	04:15:00	2022-04-01 16:16:37	2022-04-01 16:16:37
19	04:30:00	2022-04-01 16:16:37	2022-04-01 16:16:37
20	04:45:00	2022-04-01 16:16:37	2022-04-01 16:16:37
21	05:00:00	2022-04-01 16:16:37	2022-04-01 16:16:37
22	05:15:00	2022-04-01 16:16:37	2022-04-01 16:16:37
23	05:30:00	2022-04-01 16:16:37	2022-04-01 16:16:37
24	05:45:00	2022-04-01 16:16:37	2022-04-01 16:16:37
25	06:00:00	2022-04-01 16:16:37	2022-04-01 16:16:37

Figure 13: The table "TimeSlot"

#	Seat	Flight_Code	Price	Passport_Nr	Created_at	Updated_at
1	10a	102	350.00	DA54416575	2022-04-04 14:47:23	2022-04-04 14:47:23
2	10b	103	120.00	DE54546422	2022-04-04 14:47:23	2022-04-04 14:47:23
3	12b	101	275.00	GR23232311	2022-04-04 14:47:23	2022-04-04 14:47:23
4	15b	105	1250.00	IT15413214	2022-04-04 14:47:23	2022-04-04 14:47:23
5	15c	104	750.00	US15444644	2022-04-04 14:47:23	2022-04-04 14:47:23
6	1a	100	300.00	GR23232311	2022-04-04 14:47:23	2022-04-04 14:47:23
7	20a	106	900.00	UK54613146	2022-04-04 14:47:23	2022-04-04 14:47:23
8	20b	108	850.00	FR54416198	2022-04-04 14:47:23	2022-04-04 14:47:23
9	2c	101	400.00	BU35265464	2022-04-04 14:47:23	2022-04-04 14:47:23

Figure 14: The table "Tickets"

Store_ID	Store_Name	Store_Type	Created_at	Updated_at
0	StarBucks	Food & Drinks	2022-04-03 09:19:05	2022-04-03 09:19:05
1	WHSmith	Books & Magazines	2022-04-03 09:19:05	2022-04-03 09:19:05
2	Airport Pharmacy	Medicine	2022-04-03 09:19:05	2022-04-03 09:19:05
3	ECCO	Fashion	2022-04-03 09:19:05	2022-04-03 09:19:05
4	Global Exchange	Currency	2022-04-03 09:19:05	2022-04-03 09:19:05
5	LEGO	Toys	2022-04-03 09:19:05	2022-04-03 09:19:05
6	Gorm's	Food & Drinks	2022-04-03 09:19:05	2022-04-03 09:19:05

Figure 15: The table "Stores"

Store_ID	AP_Name	Created_at	Updated_at
0	Ferenc Liszt	2022-04-03 09:23:18	2022-04-03 09:23:18
0	Jorge Chavez international airport	2022-04-03 09:23:18	2022-04-03 09:23:18
0	Luxembourg Findel airport	2022-04-03 09:23:18	2022-04-03 09:23:18
0	Oslo airport	2022-04-03 09:23:18	2022-04-03 09:23:18
1	Copenhagen airport	2022-04-03 09:23:18	2022-04-03 09:23:18
1	Ferenc Liszt	2022-04-03 09:23:18	2022-04-03 09:23:18
1	Hamad International airport	2022-04-03 09:23:18	2022-04-03 09:23:18
1	Oslo airport	2022-04-03 09:23:18	2022-04-03 09:23:18
2	Copenhagen airport	2022-04-03 09:23:18	2022-04-03 09:23:18
2	Hamad International airport	2022-04-03 09:23:18	2022-04-03 09:23:18
2	Haneda Airport	2022-04-03 09:23:18	2022-04-03 09:23:18
2	Jorge Chavez international airport	2022-04-03 09:23:18	2022-04-03 09:23:18
3	Charles de gaulle airport	2022-04-03 09:23:18	2022-04-03 09:23:18
3	Copenhagen airport	2022-04-03 09:23:18	2022-04-03 09:23:18
3	Hamad International airport	2022-04-03 09:23:18	2022-04-03 09:23:18
3	Oslo airport	2022-04-03 09:23:18	2022-04-03 09:23:18
4	Charles de gaulle airport	2022-04-03 09:23:18	2022-04-03 09:23:18
4	Ferenc Liszt	2022-04-03 09:23:18	2022-04-03 09:23:18
4	Hamad International airport	2022-04-03 09:23:18	2022-04-03 09:23:18
4	Haneda Airport	2022-04-03 09:23:18	2022-04-03 09:23:18
5	Charles de gaulle airport	2022-04-03 09:23:18	2022-04-03 09:23:18
5	Copenhagen airport	2022-04-03 09:23:18	2022-04-03 09:23:18
5	Haneda Airport	2022-04-03 09:23:18	2022-04-03 09:23:18
5	Jorge Chavez international airport	2022-04-03 09:23:18	2022-04-03 09:23:18
6	Ferenc Liszt	2022-04-03 09:23:18	2022-04-03 09:23:18
6	Hamad International airport	2022-04-03 09:23:18	2022-04-03 09:23:18
6	Haneda Airport	2022-04-03 09:23:18	2022-04-03 09:23:18
6	Luxembourg Findel airport	2022-04-03 09:23:18	2022-04-03 09:23:18

Figure 16: The table "AirportStores"

7 SQL Data Queries

The following is a list and description of example SQL queries on the airport database.

The following statement is meant to aggregate what model of planes each airline owns, and how many of them they own.

```
SELECT airL.airline_id, airline_name, planes.Model_Name, COUNT(planes.Model_Name) owned
    FROM airlines airL
LEFT JOIN airplane planes
 ON airL.airline_id = planes.airline_id
 GROUP BY airL.airline_id;
```

	airline_id	airline_name	Model_Name	owned
▶	0	SAS	Sukhoi Superjet 100	1
	1	Ryanair	Boeing 737-800	1
	2	Lufthansa Group	Sukhoi Superjet 100	1
	3	Delta Air Lines	Airbus A330-300	2
	4	China Southern Airlines	Embraer 170/175	2

Figure 17: Resulting query

Below is an example of listing the amount of flights all passengers in the database have currently. It works by doing two joins, one of which is with a sub-query. It is grouped by their passport number.

```
SELECT T.Passport_Nr, count(F.Flight_Code) AS NumFlights, P.Email FROM ticket T
JOIN flights F JOIN (
   SELECT Passport_Nr, Email from passenger) P
ON T.Flight_Code = F.Flight_Code AND T.Passport_Nr = P.Passport_Nr
GROUP BY Passport_Nr;
```

#	Passport_Nr	NumFlights	Email
1	BU35265464	1	passenger_bu@dtu.com
2	DA54416575	1	passenger_da@dtu.com
3	DE54546422	1	passenger_de@dtu.com
4	FR54416198	1	passenger_fr@dtu.com
5	GR23232311	2	passenger_gr@dtu.com
6	IT15413214	1	passenger_it@dtu.com
7	UK54613146	1	passenger_uk@dtu.com
8	US15444644	1	passenger_us@dtu.com

Figure 18: Resulting query - as only passenger with GR in their passport_number has more than 1 flight in the system, this is expected.

The next statement creates a list of all passengers, who has already bought a ticket. For this query we used two tables "Passenger" and "Ticket" and joined them using the column "Passport Id".

```
SELECT Passport_Nr, Email, Flight_Code, Seat
FROM passenger JOIN ticket USING (Passport_Nr)
ORDER BY Passport_Nr ASC;
```

Passport_Nr	Email	Flight_Code	Seat
BU35265464	passenger_bu@dtu.com	101	2c
DA54416575	passenger_da@dtu.com	102	10a
DE54546422	passenger_de@dtu.com	103	10b
FR54416198	passenger_fr@dtu.com	108	20b
GR23232311	passenger_gr@dtu.com	100	1a
IT15413214	passenger_it@dtu.com	105	15b
UK54613146	passenger_uk@dtu.com	106	20a
US15444644	passenger_us@dtu.com	104	15c

Figure 19: Resulting query

The next statement gets relevant flight information for all flights with a duration of 1 hour or less. This is done by using 4 INNER JOINs.

```
SELECT flights.flight_code, Pilot_Name, Pilots.PID, Airline_Name, Departure, Arrival,
    Source_location, Destination_location, Model_Name
FROM airlines
INNER JOIN flights ON airlines.Airline_ID = flights.Airline_ID
INNER JOIN flight_pilots ON flights.flight_code = flight_pilots.flight_code
INNER JOIN pilots ON flight_pilots.PID = pilots.PID
INNER JOIN airplane ON flights.Airline_ID = airplane.Airline_ID
WHERE flights.arrival - flights.Departure <= 4
GROUP by pilots.PID;
```

Passport_Nr	Email	Flight_Code	Seat
BU35265464	passenger_bu@dtu.com	101	2c
DA54416575	passenger_da@dtu.com	102	10a
DE54546422	passenger_de@dtu.com	103	10b
FR54416198	passenger_fr@dtu.com	108	20b
GR23232311	passenger_gr@dtu.com	100	1a
IT15413214	passenger_it@dtu.com	105	15b
UK54613146	passenger_uk@dtu.com	106	20a
US15444644	passenger_us@dtu.com	104	15c

Figure 20: Resulting query

8 SQL Table Modification

Below is a list of table modification examples on the airport management database.

This is an example of an airport needing a name change. The reason why it is interesting, is that it triggers cascades from other tables, as we're modifying a primary key from the airports table, which is used in other tables.

```
UPDATE airports set AP_Name =' Danish international airport'
where city_Name = 'Copenhagen';
```

AP_Name	City_Name	Capacity	Created_at	Updated_at
Danish international airport	Copenhagen	50000	2022-04-03 08:52:10	2022-04-03 08:52:18
Charles de gaulle airport	Paris	9000	2022-04-03 08:52:10	2022-04-03 08:52:10
Ferenc Liszt	Budapest	500000	2022-04-03 08:52:10	2022-04-03 08:52:10
Hamad International airport	Doha	12500	2022-04-03 08:52:10	2022-04-03 08:52:10
Haneda Airport	Tokyo	30000	2022-04-03 08:52:10	2022-04-03 08:52:10
Jorge Chavez international airport	Lima	10000	2022-04-03 08:52:10	2022-04-03 08:52:10
Luxembourg Findel airport	Luxembourg	10000	2022-04-03 08:52:10	2022-04-03 08:52:10
Oslo airport	Oslo	10000	2022-04-03 08:52:10	2022-04-03 08:52:10
NULL	NULL	NULL	NULL	NULL

Figure 21: resulting change after changing the name

This however also changes the tables *Flight*, as well as *airport stores* and *terminal*. Here is an example of the change shown in the *terminal* table.

	Terminal_Number	AP_Name	Created_at	Updated_at
▶	1	Danish international airport	2022-04-03 08:52:10	2022-04-03 08:52:10
	1	Charles de gaulle airport	2022-04-03 08:52:10	2022-04-03 08:52:10
	1	Ferenc Liszt	2022-04-03 08:52:10	2022-04-03 08:52:10
	1	Hamad International airport	2022-04-03 08:52:10	2022-04-03 08:52:10
	1	Haneda Airport	2022-04-03 08:52:10	2022-04-03 08:52:10
	1	Jorge Chavez international airport	2022-04-03 08:52:10	2022-04-03 08:52:10
	1	Luxembourg Findel airport	2022-04-03 08:52:10	2022-04-03 08:52:10
	1	Oslo airport	2022-04-03 08:52:10	2022-04-03 08:52:10
	2	Danish international airport	2022-04-03 08:52:10	2022-04-03 08:52:10
	2	Ferenc Liszt	2022-04-03 08:52:10	2022-04-03 08:52:10
	2	Hamad International airport	2022-04-03 08:52:10	2022-04-03 08:52:10
	2	Jorge Chavez international airport	2022-04-03 08:52:10	2022-04-03 08:52:10
	2	Luxembourg Findel airport	2022-04-03 08:52:10	2022-04-03 08:52:10
	3	Ferenc Liszt	2022-04-03 08:52:10	2022-04-03 08:52:10
	3	Luxembourg Findel airport	2022-04-03 08:52:10	2022-04-03 08:52:10
*	NULL	NULL	NULL	NULL

Figure 22: terminal being updated by the on update cascade trigger.

The following example shows, how pilot salary have changed, when the company "Delta Air Lines"(ID=3) increased the salary of its employees by 10

```
UPDATE pilots SET Salary = Salary * 1.10 WHERE Airline_ID = 3;
```

PID	Pilot_Name	Salary	Airline_ID	Created_at	Updated_at
6	David Anderson	43923.00	3	2022-04-01 16:16:46	2022-04-03 10:18:30
7	Kenneth Pløger	66550.00	3	2022-04-01 16:16:46	2022-04-03 10:18:30
8	Nadia Antonio	55902.00	3	2022-04-01 16:16:46	2022-04-03 10:18:30

Figure 23: The pilot salary after update

9 Programming

In this section we introduce an example of a `function`, `procedure`, `trigger` and an `event`.

9.0.1 Transaction

This is both a combination of a *procedure* with *transaction*. The trigger-procedure's name is **ChangeFlightDeparture**. The procedure expects the flight_code, the new terminal name and the new time of departure. Its function is to verify that the update will not create any new conflicts w.r.t, existing flights.

The full trigger can be seen below:

```
DELIMITER \\
CREATE PROCEDURE ChangeFlightDeparture(
   IN vFlight_Code VARCHAR(255), vNewTimeSlotID int, vNewTerminalNum VARCHAR(50), OUT
       vStatus VARCHAR(255))
BEGIN
   DECLARE AnyFlightsAtNewTime INT DEFAULT 0;
    DECLARE AirportName VARCHAR(255) DEFAULT "";
    DECLARE AirportMaxTerminals INT DEFAULT 0;
    DECLARE DOD DATE DEFAULT curdate();
    START TRANSACTION;
    SET AirportName = (SELECT Source_location FROM flights WHERE vFlight_Code =
        Flight_Code);
    SET AirportMaxTerminals = (SELECT count(*) FROM terminal WHERE AP_Name = AirportName);
    IF AirportMaxTerminals < vNewTerminalNum
      THEN SET vStatus = 'Terminal Number does not exists for the current airport,
          rollback!'; ROLLBACK;
        ELSE UPDATE flights SET Departure = vNewTimeSlotID, Terminal_Number =
            vNewTerminalNum WHERE Flight_Code = vFlight_Code;
          SET DOD = (SELECT Date_of_departure FROM flights WHERE Flight_Code =
              vFlight_Code);
          SET AnyFlightsAtNewTime = (SELECT count(*) FROM flights F WHERE vNewTimeSlotID
              = F.Departure AND DOD = F.Date_of_departure AND F.Terminal_Number =
              vNewTerminalNum AND F.Source_location = AirportName);
          IF AnyFlightsAtNewTime > 1
            THEN SET vStatus = 'A flight departure update would conflict with an existing
                one, rollback!'; ROLLBACK;
            ELSE SET vStatus = 'Flight has been updated, committed!'; COMMIT;
          END IF;
    END IF;
END \\
DELIMITER ;
```

Three test cases were made. The first one was a simple: '`CALL ChangeFlightDeparture(100, 74, 11, @tStatus);`'. As the flight with code 100 resides in Copenhagen Airport, which in our database only has 2 terminals, the expected output is that it fails due to incorrect terminal number. And it

does!

The second test is a valid one, where we '`CALL ChangeFlightDeparture(100, 74, 1, @tStatus);`'. This one gives a status of "Flight has been updated, committed!", and the flight now has the updated values in the tables.

Lastly, a test was made to try and change the departure time and terminal to one that already exists at the same date. The call is the following: '`CALL ChangeFlightDeparture(100, 12, 2, @tStatus);`'. This one does fail, with the reason that "A flight departure update would conflict with an existing one, rollback"!, which is the correct reason.

9.0.2 Trigger

The trigger-procedure's name is **upd_salary_check**. It's purpose is to validate a new salary of pilot and in case the salary value is negative the trigger returns zero.

The Trigger can be seen below:

```
USE airport_db;
DROP TRIGGER IF EXISTS upd_salary_check;

delimiter //

CREATE TRIGGER upd_salary_check BEFORE INSERT ON pilots FOR EACH ROW IF NEW.Salary < 0
THEN SET NEW.Salary = 0;
END IF; //

# Test
INSERT INTO
    pilots(PID, Pilot_Name, Salary, Airline_ID)
VALUES
    (7000, "Test Trigger", -45000, 40);
```

Finally, we are testing our trigger with a faulty insertion on pilots with negative salary. The trigger prevents the update correctly and instead zero value is inserted on the new pilot.

9.0.3 Function

Here is an example of a function for our database. The function flightDuration takes two timeID's, which represents a departure and an arrival. It then computes the duration between those two timeslots in the *time* SQL format.

```
DELIMITER //
CREATE FUNCTION flightDuration(Departure int,Arrival int) RETURNS time
BEGIN
   DECLARE difference time;
    DECLARE arrivaltime datetime;
```

```
    DECLARE departuretime datetime;
    SELECT CAST(time_slot AS datetime) from timeslot where arrival = time_id INTO
        arrivaltime;
    SELECT CAST(time_slot AS datetime) from timeslot where departure= time_id INTO
        departuretime;
    SELECT SEC_TO_TIME(TIMESTAMPDIFF(SECOND, arrivaltime,departuretime)) into difference;
    RETURN difference ;
END//
DELIMITER ;
```

To come with an example of the function being used, here is a query which computes the length of all flights in our database, as well as the result.

```
select flight_code, flightDuration(arrival,departure) duration from flights;
```

flight_code	duration
100	01:00:00
101	02:00:00
102	01:30:00
103	02:15:00
104	01:00:00
105	02:30:00
106	02:30:00
107	05:00:00
108	01:45:00
109	02:30:00

Figure 24: Duration of all flights in the database

9.0.4 Event

Here is an example of an event for our database. The event deleteDepartedFlights is scheduled to run every day. It then goes through the entries of the flights table, making a *datediff()* check to see a flight has departed and deleting it if true.

```
DROP EVENT IF EXISTS ‘deleteDepartedFlights‘;
DELIMITER $$
CREATE EVENT ‘deleteDepartedFlights‘
  ON SCHEDULE EVERY 1 DAY STARTS ’2022-01-01 00:00:00’
  ON COMPLETION PRESERVE
DO BEGIN
   DELETE FROM Flights
   WHERE datediff(curdate(), Date_of_departure)>=1;

END$$
DELIMITER ;
```

An alternative way to handle the departed flights, could be to create another table called flightHistory. We could then store the departed flights in there, instead of deleting it.