CSU 44D01  
Information Management

Trinity College Dublin

Submitted by

Apnatva Singh Rawat

21354929

Contents

[CSU 44D01 Information Management 3](#_Toc119405710)

[Entity Relationship Diagram todo 4](#_Toc119405711)

[Mapping to Relational Schema todo 5](#_Toc119405712)

[Functional Dependency Diagrams todo 6](#_Toc119405713)

[Explanations and SQL Codes 8](#_Toc119405714)

[Creating and Using the Database 8](#_Toc119405715)

[Creating the Database Tables 8](#_Toc119405716)

[Altering Tables 12](#_Toc119405717)

[Triggering Operations 13](#_Toc119405718)

[Creating Views 16](#_Toc119405719)

[Populating the Tables 18](#_Toc119405720)

[Retrieving Information from the Database todo 21](#_Toc119405721)

[Security (Roles and Permissions) 22](#_Toc119405722)

[Creating Roles 22](#_Toc119405723)

[Creating Users 22](#_Toc119405724)

[Granting/Revoking Privileges 22](#_Toc119405725)

[Assigning/Revoking Roles 23](#_Toc119405726)

[Additional SQL Features todo 24](#_Toc119405727)

# CSU 44D01 Information Management

This project aims to model the database of an airport. The tables hold information regarding Airlines, their Flight Plans, Number of Planes that can be intersected with the data on Customers to provide them with personalized data which is easy to retrieve. It also contains information about the staff employed by the airport. To limit the size of the database, I chose to ignore the presence of shops, and restaurants, but I did include lounges. The entries in each table are also limited, which is unrealistic, but such huge data can only be operated by some level of automation which I cannot achieve and is out of the scope of this project. Apart from these, the database has been designed to be as realistic as possible. Another additional note is that the database only takes into account outbound flights.

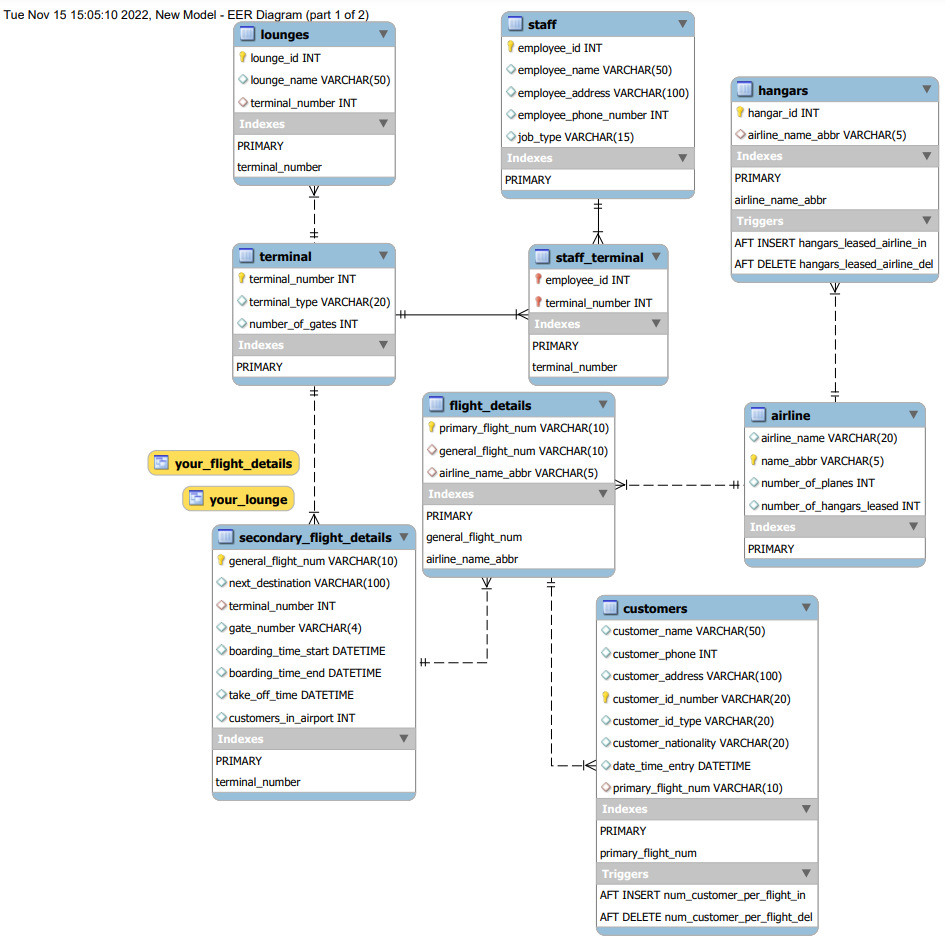
The tables in the database include a terminal, airline, flight\_details, customers, lounges, secondary\_flight\_details, hangars, staff, and staff\_terminal. Table names are very generic and model as their name suggests. Some of the tables are present only to model one-to-many or many-to-many relationships in the database.

The tables were normalized with BCN leading to extra tables to represent the many-to-one relationships. All attributes in the table are directly and only dependent on the full primary key of the table.

# Entity Relationship Diagram

# Mapping to Relational Schema

# Functional Dependency Diagrams

From MySQL Workbench

# Explanations and SQL Codes

## Creating and Using the Database

-- create database

create database flight;

-- use this database

use flight;

## Creating the Database Tables

1. Creating a table to model terminals. Each terminal is assigned a unique number which acts as an identifier. It describes the number of gates in the terminal and the type, Domestic, International, or Cargo.

create table terminal(

    terminal\_number int,

    terminal\_type varchar(20) not null,

    number\_of\_gates int,

    primary key (terminal\_number)

    );

1. Creating a table to model the airlines. Each airline is assumed to have a unique name. An abbreviation is also associated with the airline which is used as the primary key since that would always be of a constant length which makes it easier when using it as a foreign key. It contains information regarding the number of planes the airline currently has in the airport and the number of hangars they have leased from the airport authorities.

create table airline(

    airline\_name varchar(20) not null unique,

    name\_abbr varchar(5),

    number\_of\_planes int,

    number\_of\_hangars\_leased int default 0,

    primary key (name\_abbr)

    );

1. A randomly generated alphanumeric code is assigned to each flight. This is stored as the general\_flight\_num which is used to uniquely identify each identity. The information regarding a flight, the destination, where to board it from, what time to board it, and the take-off time is stored in this table.

create table secondary\_flight\_details(

    general\_flight\_num varchar(10),

    next\_destination varchar(100),

    terminal\_number int,

    gate\_number varchar(4),

    boarding\_time\_start datetime,

    boarding\_time\_end datetime,

    take\_off\_time datetime,

    customers\_in\_airport int default 0,

    primary key (general\_flight\_num),

    foreign key (terminal\_number) references terminal(terminal\_number)

    );

1. This table maps the flight numbers (primary\_flight\_num) to the internal flight numbers general\_flight\_num (reference table secondary\_flight\_details). Multiple airlines can run the same flight making this table necessary. It also references the name\_abbr (table airline) to identify which airline has scheduled the flight. The on-delete cascade allows the database to take the information of the flight once it has taken off. According to the rules of mapping, the general flight number should have been stored in the column for customers as it is a many-to-one relationship. But this number is for internal use by the airport, and hence should not be available to a customer. This was why a new table was constructed to accurately map all the foreign keys.

create table flight\_details(

    primary\_flight\_num varchar(10),

    general\_flight\_num varchar(10),

    airline\_name\_abbr varchar(5),

    primary key (primary\_flight\_num),

    foreign key (general\_flight\_num) references secondary\_flight\_details(general\_flight\_num) on delete cascade,

    foreign key (airline\_name\_abbr) references airline(name\_abbr)

    );

1. This table stores information about all the customers who have entered the airport. Since customers can enter the airport with various valid IDs (Passports, National-ID) so a customer\_id\_type had to be included where the customer\_id\_num is the primary key for the table. Other information like name, phone number, address, time of entry, and flight number is stored in the table. The ON DELETE CASCADE constraint allows all customers to be removed from the database for the flight that has taken off.

create table customers(

    customer\_name varchar(50),

    customer\_phone int,

    customer\_address varchar(100),

    customer\_id\_number varchar(20),

    customer\_id\_type varchar(20),

    customer\_nationality varchar(20),

    date\_time\_entry datetime,

    primary\_flight\_num varchar(10),

    primary key (customer\_id\_number),

    foreign key (primary\_flight\_num) references flight\_details(primary\_flight\_num) on delete cascade

    );

1. Contains information about lounges which can be uniquely identified by a lounge\_id (primary key). A lounge is available to every customer in that terminal and hence references the terminal\_number from the table terminal. The secondary identification of a lounge is its name, which is less generic than just calling the lounge by a number.

create table lounges(

    lounge\_id int,

    lounge\_name varchar(50) default 'Lounge',

    terminal\_number int,

    primary key (lounge\_id),

    foreign key (terminal\_number) references terminal(terminal\_number)

    );

1. Planes are parked in the hangar, where each hangar has a unique ID assigned to it as hangar\_id (primary key). It references the airline table as airline\_name\_abbr so that we can associate an airline with it. This is helpful for airlines to identify if they need to lease more hangars or less by presenting a count of hangars for their airline.

create table hangars(

    hangar\_id int,

    airline\_name\_abbr varchar(5),

    primary key (hangar\_id),

    foreign key (airline\_name\_abbr) references airline(name\_abbr)

    );

1. This table stores information about the staff that is employed by the airport only. It does not include information about pilots, co-pilots, stewards, etc. who work for an airline. It focuses on staff which is concerned with the cleanliness, and security, of the airport, managing flight and landing schedules, and runway. Some information to identify the staff is stored, employee\_id (primary key) their name, address, phone number, and job\_type which describes cleaning, security, etc.

create table staff(

    employee\_id int,

    employee\_name varchar(50),

    employee\_address varchar(100),

    employee\_phone\_number int,

    job\_type varchar(15),

    primary key (employee\_id)

    );

1. A staff can work at multiple terminals, this table stores information about how many and which terminals a staff member works in, referencing two tables staff and terminal. The two attributes are joined to form the primary key of the table.

create table staff\_terminal(

    employee\_id int,

    terminal\_number int,

    foreign key (employee\_id) references staff(employee\_id),

    foreign key (terminal\_number) references terminal(terminal\_number),

    primary key (employee\_id, terminal\_number)

    );

## Altering Tables

The following changes to the table are not intended in the final database. These queries were constructed to provide examples of how tables, columns, names, datatypes can be manipulated. The actual final database where the other queries are performed require that these changes (if made) are reversed.

1. Adding a new column(s)

alter table hangars

    add hangar\_name varchar (20);

alter table staff

    add national\_id\_number varchar (10),

    add national\_id\_type varchar (10);

1. Changing the datatype of the column

alter table terminal

    modify column terminal\_type int;

1. Renaming a column

alter table secondary\_flight\_info

    rename to more\_flight\_info;

## Triggering Operations

1. Trigger to count the number of customers in each flight and update it in the secondary\_flight\_details table. Updates when new customers enter.

delimiter //

create trigger num\_customer\_per\_flight\_in

after insert on customers

for each row

begin

    declare new\_count int;

    declare new\_gen\_flight\_num varchar(10);

    set new\_gen\_flight\_num = ( select flight\_details.general\_flight\_num from customers

                            left join flight\_details

                            on  customers.primary\_flight\_num = flight\_details.primary\_flight\_num

                            where flight\_details.primary\_flight\_num = new.primary\_flight\_num

                            limit 1

                            );

    set new\_count = ( select count(\*) from customers

                    left join flight\_details

                    on  customers.primary\_flight\_num = flight\_details.primary\_flight\_num

                    where flight\_details.general\_flight\_num = new\_gen\_flight\_num

                    );

    update secondary\_flight\_details

        set customers\_in\_airport = new\_count

        where new\_gen\_flight\_num = secondary\_flight\_details.general\_flight\_num;

end //

delimiter ;

1. Trigger to count the number of customers in each flight and update it in the secondary\_flight\_details table. Updates when a customer leaves.

delimiter //

create trigger num\_customer\_per\_flight\_del

after delete on customers

for each row

begin

    declare new\_count int;

    declare old\_gen\_flight\_num varchar(10);

    set old\_gen\_flight\_num = ( select flight\_details.general\_flight\_num from customers

                            left join flight\_details

                            on  customers.primary\_flight\_num = flight\_details.primary\_flight\_num

                            where flight\_details.primary\_flight\_num = old.primary\_flight\_num

                            limit 1

                            );

    set new\_count = ( select count(\*) from customers

                    left join flight\_details

                    on  customers.primary\_flight\_num = flight\_details.primary\_flight\_num

                    where flight\_details.general\_flight\_num = old\_gen\_flight\_num

                    );

    update secondary\_flight\_details

        set customers\_in\_airport = new\_count

        where old\_gen\_flight\_num = secondary\_flight\_details.general\_flight\_num;

end //

delimiter ;

1. Counts the number of hangars leased by each airline when new entries are added into the hangars table.

delimiter //

create trigger hangars\_leased\_airline\_in

after insert on hangars

for each row

begin

    declare new\_count int;

    set new\_count = ( select count(\*) from hangars

                left join airline

                on hangars.airline\_name\_abbr = airline.name\_abbr

                where airline.name\_abbr = new.airline\_name\_abbr

                limit 1

                );

    update airline

        set number\_of\_hangars\_leased = new\_count

        where name\_abbr = new.airline\_name\_abbr;

end //

delimiter ;

1. Counts the number of hangars leased by each airline when entries are removed from the hangars table.

delimiter //

create trigger hangars\_leased\_airline\_del

after delete on hangars

for each row

begin

    declare new\_count int;

    set new\_count = ( select count(\*) from hangars

                left join airline

                on hangars.airline\_name\_abbr = airline.name\_abbr

                where airline.name\_abbr = old.airline\_name\_abbr

                limit 1

                );

    update airline

        set number\_of\_hangars\_leased = new\_count

        where name\_abbr = old.airline\_name\_abbr;

end //

delimiter ;

## Creating Views

1. This will display all the details that a customer might require about their flight. It joins 4 tables, customers, airline, flight\_details, and secondary\_flight\_details to get the desired result. This information constantly keeps updating because there might be delays on the flight.

create or replace view your\_flight\_details as (

    select customers.customer\_name,

    customers.primary\_flight\_num,

    customers.customer\_id\_number,

    airline.airline\_name,

    secondary\_flight\_details.terminal\_number,

    secondary\_flight\_details.gate\_number,

    secondary\_flight\_details.boarding\_time\_start,

    secondary\_flight\_details.boarding\_time\_end,

    secondary\_flight\_details.take\_off\_time

    from customers left join flight\_details

    on customers.primary\_flight\_num = flight\_details.primary\_flight\_num

    left join secondary\_flight\_details on

    flight\_details.general\_flight\_num = secondary\_flight\_details.general\_flight\_num

    left join airline on

    airline.name\_abbr = flight\_details.airline\_name\_abbr

);

1. Similar to the earlier view, this joins multiple tables to tell the customer the name of the lounge for the terminal they can go to. This is intended to be information that updates once for the customer and stays the same until the flight time. Due to the difference in the frequency required to access this information, they were kept in separate views.

create or replace view your\_lounge as (

    select customers.customer\_name,

    customers.primary\_flight\_num,

    customers.customer\_id\_number,

    secondary\_flight\_details.terminal\_number,

    lounges.lounge\_name

    from customers left join flight\_details

    on customers.primary\_flight\_num = flight\_details.primary\_flight\_num

    left join secondary\_flight\_details on

    flight\_details.general\_flight\_num = secondary\_flight\_details.general\_flight\_num

    left join lounges on

    secondary\_flight\_details.terminal\_number = lounges.terminal\_number

    );

## Populating the Tables

1. Inserting values into the table terminal. 5 Entries.

insert into terminal values

    (1, "domestic", 10),

    (2, "domestic", 40),

    (3, "cargo", 20),

    (4, "international", 20),

    (5, "international", 30);

1. Inserting values into table airline. 5 Entries.

insert into airline (airline\_name, name\_abbr, number\_of\_planes)

    values ("Qatar Airways", "QAR", 19),

    ("Air India", "AIR", 23),

    ("Aer Lingus", "AEL", 45),

    ("Lufthansa", "LUF", 23),

    ("British Airways", "BRA", 38);

1. Inserting values into table secondary\_flight\_details. Default value for 0 customers is assigned automatically. 5 Entries.

insert into secondary\_flight\_details (

    general\_flight\_num,

    next\_destination,

    terminal\_number,

    gate\_number,

    boarding\_time\_start,

    boarding\_time\_end,

    take\_off\_time) values

    ('random1', 'Bellavista', '1', 'H45', '2022-07-22 15:02:55', '2022-07-08 15:58:28', '2022-07-22 16:33:41'),

    ('random2', 'Pontal', '4', 'J19', '2022-07-22 09:27:28', '2022-07-22 10:25:03', '2022-07-22 11:10:03'),

    ('random3', 'Huatajata', '1', 'L44', '2022-07-22 23:57:10', '2022-07-23 00:50:29', '2022-07-23 01:22:06'),

    ('random4', 'Agualote', '2', 'S32', '2022-07-22 11:11:29', '2022-07-22 12:01:43', '2022-07-22 13:01:30'),

    ('random5', 'Ambatofinandrahana', '5', 'B29', '2022-07-22 19:39:31', '2022-09-08 20:24:28', '2022-07-22 21:05:48');

1. Inserting values into table flight\_details. 6 Entries.

insert into flight\_details values

    ( 'QAR769', 'random1', 'QAR'), ( 'AIR802', 'random1', 'AIR'),

    ( 'AIR994', 'random2', 'AIR'), ( 'LUF676', 'random4', 'LUF'),

    ( 'AEL785', 'random3', 'AEL'), ( 'BRA426', 'random5', 'BRA');

1. Inserting values into table customers. 20 Entries.

insert into customers values

    ('Ned Nacey', '56850944', '58893 Bultman Way', '3CY2EWT8P', 'Passport', 'Poland', '2022-01-06 03:46:05', 'QAR769'),

    ('Granville Beaston', '11726708', '63 Stephen Plaza', 'VH5YSYL6E', 'Passport', 'Poland', '2022-10-25 11:17:44', 'AIR802'),

    ('Annice Klasen', '33619098', '955 Johnson Drive', 'YJJSBV0QH', 'Passport', 'Indonesia', '2022-08-24 20:18:54', 'AIR994'),

    ('Aldric Guite', '77370950', '63 Sheridan Trail', 'H3L25C1EJ', 'Passport', 'Madagascar', '2022-07-26 16:48:48', 'QAR769'),

    ('Helli Cardenas', '36375814', '58 Scoville Hill', '8JA4FDUEU', 'Passport', 'Argentina', '2022-04-23 20:06:00', 'AIR802'),

    ('Tory Dimbylow', '60507239', '37 Redwing Park', '0ZZODVWMJ', 'Passport', 'China', '2021-12-23 04:20:20', 'LUF676'),

    ('Briant McGookin', '28296325', '589 Armistice Parkway', 'Q88XXJ8TN', 'Passport', 'Czech Republic', '2022-04-25 21:32:46', 'AEL785'),

    ('Tiena Villaret', '84109979', '94740 Pearson Crossing', 'US5Y2DNTU', 'Passport', 'Vietnam', '2022-01-11 00:14:36', 'QAR769'),

    ('Carlye Fraine', '47873101', '97 Waywood Pass', '0272SXZT7', 'National-ID', 'South Africa', '2022-04-14 22:52:59', 'AEL785'),

    ('Emera Pomeroy', '82108194', '4 North Avenue', 'C6I7G9AJT', 'National-ID', 'Portugal', '2022-05-22 21:13:15', 'LUF676'),

    ('Roxane West-Frimley', '53083462', '87861 Boyd Center', 'GE8B6H2WL', "Driver's Licence", 'Indonesia', '2022-04-07 13:10:35', 'AEL785'),

    ('Ora Wantling', '85293408', '09 Elka Center', 'NEV61CVOB', 'National-ID', 'Brazil', '2021-12-29 11:21:50', 'QAR769'),

    ('Devin Caley', '94142481', '4 Hermina Hill', 'CC3J9Z086', 'National-ID', 'Zimbabwe', '2022-06-07 22:07:55', 'LUF676'),

    ('Larina Deane', '87912508', '630 Pepper Wood Drive', 'IEPN7SEZF', 'Passport', 'Poland', '2022-06-02 08:02:22', 'LUF676'),

    ('Maurie Weins', '73989544', '75 Sauthoff Alley', 'MWFF74EXV', 'Passport', 'Yemen', '2022-10-14 22:19:25', 'QAR769'),

    ('Agnes Bogue', '80004987', '5 Talmadge Place', '5XISDRPTU', 'Passport', 'Argentina', '2021-11-23 16:44:12', 'BRA426'),

    ('Jaimie Rabat', '19943769', '32989 Dwight Alley', 'TCS3P85S0', 'Passport', 'Brazil', '2022-10-10 08:34:18', 'AEL785'),

    ('Bert Gerger', '89764303', '75 Oak Point', '5V5XTDSED', 'Passport', 'Indonesia', '2022-04-14 02:42:15', 'BRA426'),

    ('Debbie Baskeyfied', '50486105', '20 Village Green Hill', 'CSOAWY137', 'Passport', 'Brazil', '2022-02-20 04:18:08', 'BRA426'),

    ('Bryant Cramond', '93811915', '032 Graedel Circle', '1KNB6J7KJ', 'Passport', 'Indonesia', '2022-05-26 14:35:53', 'AIR802');

1. Inserting values into table lounges. 5 Entries.

insert into lounges values

    ( 1, "Main Lounge", 5),

    ( 2, "Side Lounge", 4),

    ( 3, "Side Lounge", 3),

    ( 4, "Small Lounge", 2),

    ( 5, "Crystal Lounge", 1),

    ( 6, "Mini Crystal Lounge", 1);

1. Inserting values into table hangars. 9 Entries.

insert into hangars values

    ( 1, "QAR"), ( 2, "QAR"), ( 3, "AIR"),

    ( 4, "BRA"), ( 5, "BRA"), ( 6, "BRA"),

    ( 7, "LUF"), ( 8, "AIR"), ( 9, "QAR");

1. Inserting values into table staff. 9 Entries.

insert into staff values

    (23, "Neil", "USA", 12345678, "Cleaning"),

    (24, "Patrick", "Ireland", 87654321, "Cleaning"),

    (12, "Harris", "UK", 345678934, "Security"),

    (56, "Chris", "France", 987643456, "Security"),

    (78, "Robert", "Spain", 456654219, "Control Tower"),

    (34, "Scarlett", "Germany", 987465768, "Cleaning"),

    (55, "Benedict", "221 B Baker Street", 876545678, "Help Desk"),

    (99, "Brie", "Canada", 345654321, "Control Tower"),

    (67, "Groot", "Dehradun", 789876545, "Security");

1. Inserting values into table staff\_terminal. 15 Entries.

insert into staff\_terminal values

    (23,1), (24,2), (12,3), (56,2), (78,2),

    (34,3), (55,4), (99,5), (67,3), (67,4),

    (56,5), (23,5), (23,4), (23,2), (34,1);

## Retrieving Information from the Database

1. Selecting all columns from various tables in the database

select \* from terminal;

select \* from lounges;

select \* from hangars;

select \* from airline;

select \* from staff;

select \* from staff\_terminal;

select \* from secondary\_flight\_details;

1. Selecting all columns from various views in the database

select \* from your\_flight\_details;

select \* from your\_lounge;

1. Returns the total number of gates in the airport.

select sum(number\_of\_gates) as total\_gates\_in\_airpot from terminal;

1. Find the number of people in each job category working at the airport.

select job\_type, count(\*) as people\_in\_department from staff group by job\_type;

1. Find the number of terminals where each staff member works

select employee\_id, count(\*) as num\_of\_terminals\_worked from staff\_terminal group by employee\_id;

1. Number of people currently in the airport.

select count(\*) from customers;

1. Count the number of hangars leased by each airline

select airline\_name\_abbr, count(\*) as number\_of\_hangars\_leased from hangars group by airline\_name\_abbr;

1. Count the number of airlines running the same flight

select general\_flight\_num, count(primary\_flight\_num) from flight\_details group by general\_flight\_num;

1. Find the list of all the customers who are flying on the defined primary flight number.

select customer\_name, customer\_phone from customers where primary\_flight\_num = "QAR769";

1. Find the list of all customers who are flying on the defined general flight number.

select customers.\*, primary\_flight from customers

    left join flight\_details on customers.primary\_flight\_num = flight\_details.primary\_flight\_num

    left join secondary\_flight\_details on flight\_details.general\_flight\_num = secondary\_flight\_details.general\_flight\_num

    where secondary\_flight\_details.general\_flight\_num = 'random1';

1. Find the next two flights to take off.

select \* from secondary\_flight\_details order by take\_off\_time limit 2;

## Security (Roles and Permissions)

### Creating Roles

Three roles are assigned. Flight manager can add/remove flights from the schedule. Hangar Authorizer handles the leasing of hangars and updates the table with information about which hangars are leased. Customer is a role assigned to customers.

create role 'flight\_manager';

create role 'hangar\_authoriser';

create role 'customer';

### Creating Users

create user 'read\_only' identified by 'pa55';

create user 'allow\_all' identified by 'skills';

create user 'customer\_1' identified by 'hehe';

create user 'hangar\_auth\_1' identified by 'daboy';

create user 'flight\_man\_1' identified by 'daman';

### Granting/Revoking Privileges

1. Allow a user to only read all values in the database

grant select on flight.terminal to 'read\_only';

grant select on flight.lounges to 'read\_only';

grant select on flight.hangars to 'read\_only';

grant select on flight.flight\_details to 'read\_only';

grant select on flight.secondary\_flight\_details to 'read\_only';

grant select on flight.staff\_terminal to 'read\_only';

grant select on flight.customers to 'read\_only';

grant select on flight.staff to 'read\_only';

1. Revoking read information on Customers and Staff because this may contain sensitive personal information which should not be accessible.

revoke select on flight.customers from 'read\_only';

1. Granting all privileges on all tables in the flight database to a user
2. grant all on flight.\* to 'allow\_all';
3. Allowing customer (role) to run the procedure to retrieve flight information

grant execute on procedure flight.get\_customer\_info\_from\_view to 'customer';

1. Allowing hangar authority to manipulate the tables

grant update on flight.hangars to 'hangar\_authoriser';

1. Allowing flight managers to add/remove entries to the relevant tables

grant update on flight.secondary\_flight\_details to 'flight\_manager';

grant update on flight.flight\_details to 'flight\_manager';

### Assigning/Revoking Roles

Each role created is assigned to a user which can be revoked later.

grant 'hangar\_authoriser' to 'hangar\_auth\_1';

grant 'flight\_manager' to 'flight\_man\_1';

grant 'customer' to 'customer\_1';

revoke 'customer' from 'customer\_1';

## Additional SQL Features

1. A procedure is created to update the take-off time for the flight. If the flight is not found the error is handled and tells the user that the query did not work. Similar procedures can be constructed to update boarding start/end time.

delimiter //

create procedure update\_take\_off\_time(in p\_flight\_num varchar(5), in new\_take\_off\_time datetime)

begin

    declare g\_flight\_num varchar(10);

    declare exit handler for sqlexception

    begin

        rollback;

        select 'Error Occurred. Could Not Update Take Off Time';

    end;

    set g\_flight\_num = (select general\_flight\_num from flight\_details

                        where primary\_flight\_num = p\_flight\_num);

    update secondary\_flight\_details

        set take\_off\_time = new\_take\_off\_time

        where general\_flight\_number = g\_flight\_num;

end //

delimiter ;

The first query will update the table. The second one will cause an error

call update\_take\_off\_time('QAR769', '2023-05-26 14:35:53');

call update\_take\_off\_time('pp', '2022-12-26 14:35:53');

1. This procedure can be used by a customer to log in and receive all of the information regarding their flights. Since most of the information is already compiled in the view it makes it simpler to retrieve the information.

delimiter //

create procedure get\_customer\_info\_login(in customer\_id varchar(20))

begin

    select your\_flight\_details.\*, your\_lounge.lounge\_name from your\_flight\_details

    left join your\_lounge

    on your\_flight\_details.customer\_id\_number = your\_lounge.customer\_id\_number

    where your\_flight\_details.customer\_id\_number = customer\_id;

end //

delimiter ;

1. This creates an event that is scheduled to go off every minute. Once the take\_off\_time for the flight is past, it automatically deletes the record from the table. Since delete cascades, the relevant records from the flight\_details and customers table are also deleted. This information can also be passed on to more archival storage where it can be stored there.

set global event\_scheduler = on;

create event if not exists test\_event

on schedule every 1 minute

starts now()

do

    delete from secondary\_flight\_details

        where datediff(take\_off\_time, now()) < 0.5;