Database Airlines

Final Project CS195

Prepared by: Corbin Hastings & Nate Rengo

Date: 12/04/2023

Contents

Related Documents

<insert name and description of the scripts that are provided for the project>

Document Title	Quick Description
Create_statements1.sql	Statements used to create all tables and establish Check constraints.
Final_statements.sql	Holds all queries and procedures.
Procedures.sql	Holds a secondary doc that holds procedures.
Final(4).pdf	Holds ERD.
Database_airlines (folder)	Holds exported database.
Database_info .mov	Video for class
Aircraft.sql	Insert statements for the aircraft table
Airport.sql	Insert data for insert
Employee.sql	Insert for employee
Ticket.sql	Insert statements for ticket table
Customer.sql	Insert statements for customer
Employee flight.sql	Insert for employee flight table
Flight and flight pattern.sql	Insert statements for flight and flight_pattern
Ticket_flight.sql	Insert statements for ticket_flight table.

1. Project Definition, Scope Case

1.1. Problem statement

This database provides a system to store and track important data to help run our airline. It creates a reliable and complete collection of data to support our employees.

1.2. Users

The users will be employees and analysts for the company.

1.3. Data Requirement

We store data from employees, customers, aircraft, flights, tickets, and airports.

1.4. Constraints

Our database refers to only one airline and a limited number of destination airports. For customer the membership it can only be “gold”, “silver”, “bronze”, “elite”.

1.5. Real World Requirement

This data would be used on a regular basis to track, schedule, and support the running of the company. We have the ability to see when/where employees are working, what equipment we have available (aircraft), and can query information about transactions with customers.

1.6. Information Requirement

Retrieve information about customers, tickets, employees, aircraft, airports, flights, and all the relationships between these.

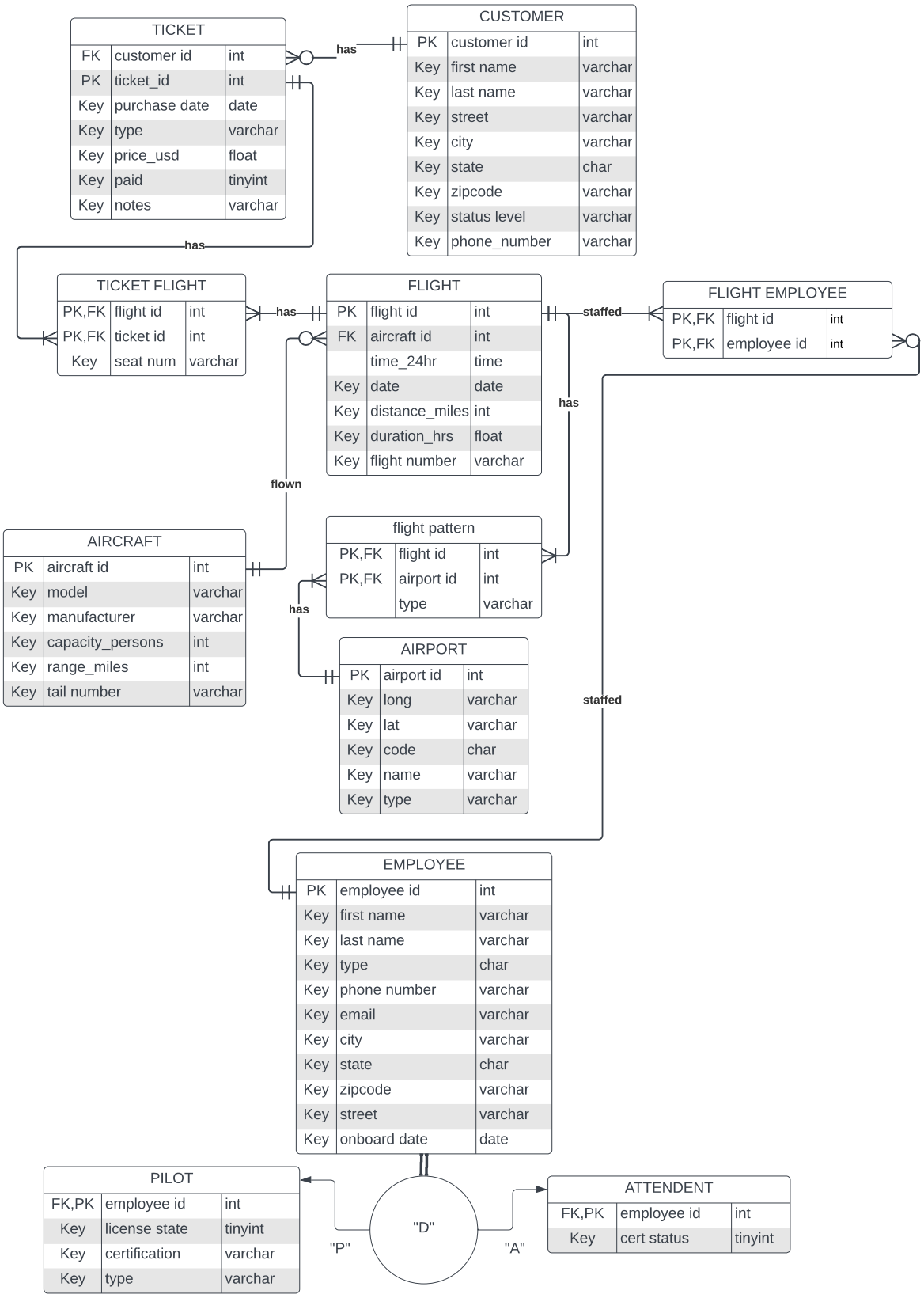
1.7. Competitive Advantage

Having a complete database will give our company the ability to collect data and find holes in coverage faster than a competing company. We would have the ability to quickly find relations and information faster. This would result in a increase work flow and improve efficiency throughout the company.

2. Conceptional Design

2.1. Entity Relationship Diagram

The following information is the high-level model and design for the scope of this project.


TICKET
FK customer id int
PK ticket_id int
Key purchase date date
Key type varchar
Key price_usd float
Key paid tinyint
Key notes varchar
has
CUSTOMER
PK customer id int
Key first name varchar
Key last name varchar
Key street varchar
Key city varchar
Key state char
Key zipcode varchar
Key status level varchar
Key phone_number varchar
has
TICKET FLIGHT
PK,FK flight id int
PK,FK ticket id int
Key seat num varchar
has
FLIGHT
PK flight id int
FK aircraft id int
time_24hr time
Key date date
Key distance_miles int
Key duration_hrs float
Key flight number varchar
staffed
FLIGHT EMPLOYEE
PK,FK flight id int
PK,FK employee id int
has
flown
AIRCRAFT
PK aircraft id int
Key model varchar
Key manufacturer varchar
Key capacity_persons int
Key range_miles int
Key tail number varchar
flight pattern
PK,FK flight id int
PK,FK airport id int
type varchar
has
AIRPORT
PK airport id int
Key long varchar
Key lat varchar
Key code char
Key name varchar
Key type varchar
staffed
EMPLOYEE
PK employee id int
Key first name varchar
Key last name varchar
Key type char
Key phone number varchar
Key email varchar
Key city varchar
Key state char
Key zipcode varchar
Key street varchar
Key onboard date date
PILOT
FK,PK employee id int
Key license state tinyint
Key certification varchar
Key type varchar
"P"
"D"
"A"
ATTENDENT
FK,PK employee id int
Key cert status tinyint

2.2. Normalization

The following information is the high-level construction and optimization of the database for the scope of this project.

- 1st normal form
  - All columns are one data type. No fixes needed.
  - All values are singular. Example: ticket notes to note. Allowing only one ‘note’
  - All tables have a primary key
- 2nd normal form
  - 1NF done
  - All non key columns depend on key. Customer information moved to its own table from ticket. Seat number moved to ticket_flight. Seat does not only apply to the ticket.breaking out pilot and attendant from employee due to different columns needed.
- 3rd normal form
  - The only way to relate a given column is by the Primary key of that table. AKA 3rd Normal form

3. Logical Design

3.1. Single Query

1. Gives us how many tickets have been sold. Total sales so far, how is our company doing?
   I. select count(ticket_id) as "Tickets Sold"
   II. from ticket;

2. Gives all pilots from Colorado. Demographics of employees, checking how many local pilots we have.
   I. select first_name, last_name
   II. from employee
   III. where state= "CO" and employee_type= "P";

3. Gives all planes that hold more than x number of people. What planes and their types do we have on call to carry x number of people.
   I. select aircraft_id, model, capacity_persons
   II. from aircraft
   III. where capacity_persons > 150;

4. All customers from Colorado. Checking the demographics of who our airline serves. This could be used to see if there is a hole in our coverage.
   I. select customer_id, first_name, last_name
   II. from customer
   III. where state= "CO";

5. Find attendants that aren't certified. To check if our staff is trained up to date and allowed to work.
   I. select employee_id
   II. from attendent
   III. where cert_status is false;

3.2. Multi-Query

1. Finds customers who have not paid for their ticket. WE do not want someone getting on a flight they have not paid for.
   I. select first_name, last_name, phone_number

II. from ticket
III. join ticket_flight
IV. on ticket.ticket_Id=ticket_flight.ticket_id
V. join customer
VI. on ticket.customer_id= customer.customer_id
VII. where pay_status = false;

2. Gets the info for all our pilots. Maybe we need to find a pilot to cover a flight or need to contact all pilots.
   I. select employee.employee_id, first_name,last_name, phone_number, email
   II. from employee
   III. join pilot
   IV. on pilot.employee_id= employee.employee_id;

3. This gives us what flight a specific passenger is on and what their destination is. Tracking where somebody is going, maybe their bag was found and needs to be sent to a certain location.
   I. select first_name, last_name, flight.flight_id, airport.airport_id as "Destination_id", airport_name as "Destination"
   II. from customer
   III. join ticket
   IV. on ticket.customer_id=customer.customer_id
   V. join ticket_flight
   VI. on ticket.ticket_id=ticket_flight.ticket_id
   VII. join flight
   VIII. on ticket_flight.flight_id= flight.flight_id
   IX. join flight_pattern
   X. on flight.flight_id= flight_pattern.flight_id
   XI. join airport
   XII. on flight_pattern.airport_id= airport.airport_id
   XIII. where first_name= "David" and last_name = "Jones"and pattern_type = "destination";

4. Gives us all the flights a specific employee is working on. Thi helps an employee see their schedule.
   I. select first_name, last_name, group_concat(flight.flight_id) as "Flights"
   II. from flight
   III. join flight_employee
   IV. on flight.flight_id= flight_employee.flight_id
   V. join employee
   VI. on flight_employee.employee_id= employee.employee_id
   VII. where first_name= "Liam" and last_name= "Anderson"
   VIII. group by employee.employee_id;

5. Give all information about both airports given a flight_id
    I. #this is to quiclky give the destination and origin of a flight for use by mangement or pilots so they can plan
    II. #givin a flight id give airpot info
    III. select airport_name, airport_code,airport_type,longitude,latitude
    IV. from airport
    V. where airport_id in (select airport_id from flight_pattern
    VI. where flight_id = 4)

3.3. Stored Procedures / Functions

1. call getPlanes(2000, 150);
    a. This allows us to find the planes that are capable of traveling provided distance and carrying more than x number of passengers. We are planning a flight and need to carry some number of people some distance.
        i. DELIMITER //
        ii. create procedure getplanes(
        iii. in distance int, people int)
        iv. begin
        v. select aircraft_id, tail_number
        vi. from aircraft
        vii. where range_miles > distance and capacity_persons > people;
        viii. END //
        ix. DELIMITER ;
        x.
2. call getNames(2);
    a. This allows us to get the names of every passenger on a flight given the ID. If something happens on a flight, or someone loses something, and we need to find a passenger this function would be used.
        i. DELIMITER //
        ii. create procedure getNames(
        iii. in flight int)
        iv. begin
        v. select flight_id as flight, first_name, last_name, phone_number
        vi. from customer
        vii. join ticket
        viii. on customer.customer_id= ticket.customer_id
        ix. join ticket_flight
        x. on ticket_flight.ticket_id= ticket.ticket_id
        xi. where flight_id= flight;
        xii. END //

xiii. DELIMITER ;

3.4. Miscellaneous

4. Design Overview

Aircraft: The aircraft table stores an id, model, manufacturer, capacity, range, and tail number. The table is associated to the flight table. We have an aircraft table to track the planes in our fleet.

Customer: The customer table stores an id, first and last name, full address, a phone number, and membership. The customer table is associated with ticket table. We track customer info to allow us to see who is on what flight, and how full a flight is.

Employee: The employee table has two sub tables of pilot and attendant. In employee we store id, first and last name, type (pilot or attendant), phone number, email, and full address. In the sub tables pilot also holds certification, type, and license state. The sub table attendant just holds certification status. The employee table is associated with the flight employee table. We have the sub tables of attendant and pilot since they are two very different jobs and store information specific to one job and not the other.

Airport: The airport table has an id, longitude and latitude, an airport code, name, and type. It is associated with the flight pattern table. We are tracking airports so that we can see where flights are going, this also relates to planning and locating passengers' destination.

Ticket: The ticket table holds customer id, ticket id, purchase date, type, price, payment status, and a notes section. It is associated with customer and ticket flight. The ticket table allows us to relate passengers to flights and keep a connection between the two.

Flight: The flight table holds, flight id, aircraft id, time of the flight, date, distance, duration, and flight number. It is associated with ticket flight, flight pattern, and flight employee. Having a flight table allows us to keep track of each individual flight and find origin, destination, and what passengers are on the flight.

Ticket Flight: Has flight id, ticket id, and seat number. It is associated with ticket and flight. Establishes a connection between our tickets and flights, which in turn relates customers to flights.

Flight pattern: Flight pattern holds flight id, airport id, and type. This table holds destination and origin airports for flight. It is associated with airport and flight. Flight pattern gives us an established destination and origin airport.

Flight employee: Holds flight id, and employee id. It is related to flight and employee tables. The flight employee table provides a connection between our employees and flights allowing us to tracking who is working when and where.