REWARD POINT BASED AIRLINE MANAGEMENT SYSTEM

Description

- Sometimes there arises problems due to booking of seats in a particular flight twice which ultimately causes problems for the customers as well as depreciates the quality of the airline system.
- Regular customers like common business fliers as well as other local customers do not get a valid discount upon flying with that airliner on a frequent basis.
- This database enables to store information about the key entities like customers, flights, ticket and airliner as well depicts a relationship among them to give a customer his best deal. Reward points will be assigned to a customer and they will be automatically redeemed during next booking.
- Techniques like stored procedures as well triggers will be used to capture special events which are handled with best results.

Key Tables

- Customer is an entity in this database which books a particular flight having source and destination. The table will include customerId, First name, Last name, Age, Reward points as attributes.
- Flight is an entity which stores information about the flight status as well as scheduling of flight. The table will include flightId, Flight status, Source, Destination as attributes.
- Airliner is an entity which stores information about its various flights. The table will include AirlinerId, Name, model as attributes.
- Ticket stores information about the customer, allocated seat, and flight. The table includes ticketId, SeatId, CustomerId, and FlightId, price as attributes.

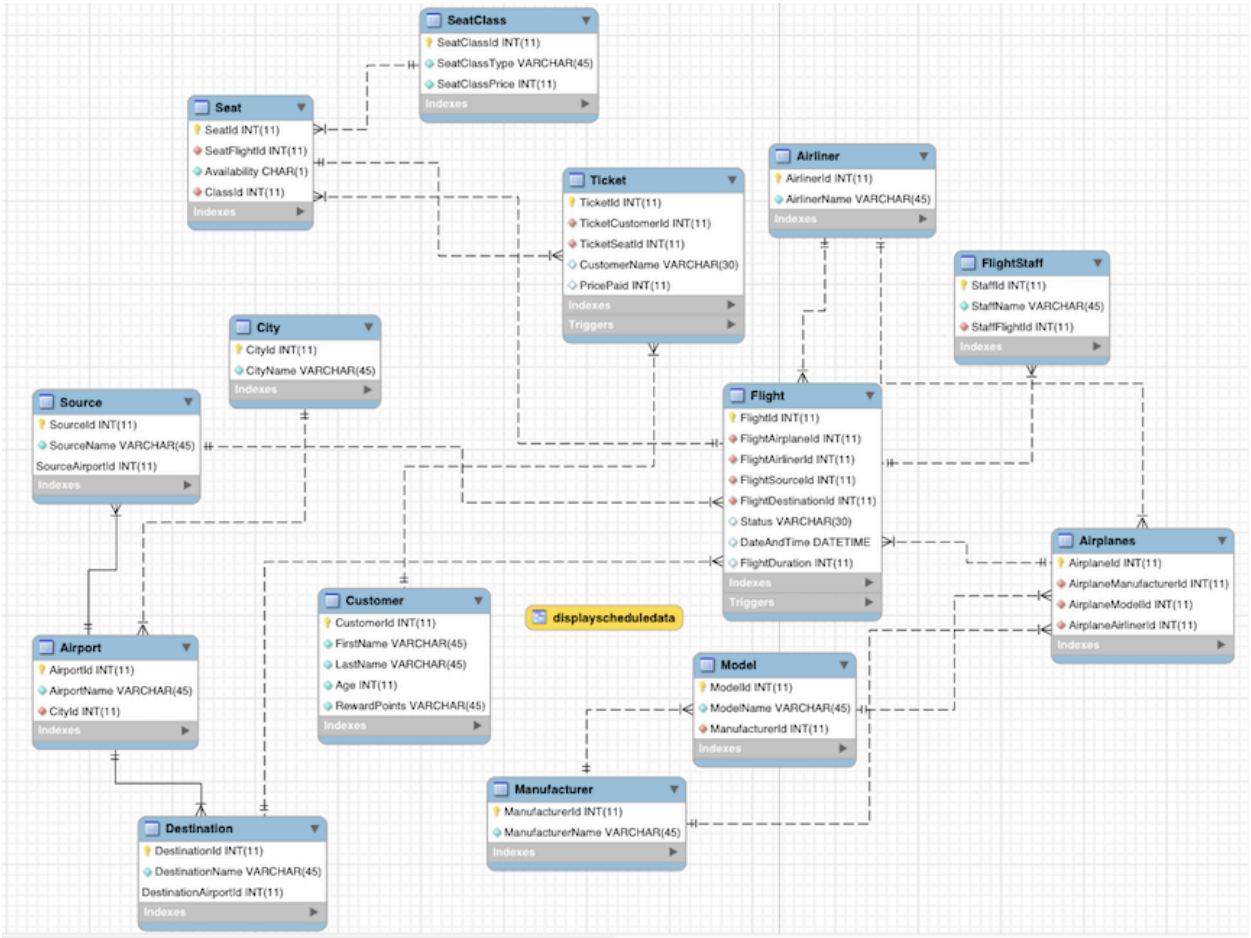
Relationships

- There will be one-to-many relationship present between customer and ticket.
- There will be one-to-many relationship present between Airliner and flight.
- There will be one-to-many relationship present between Flight and seat.
- There will be one-to-one relationship present between Ticket and seat.

UNIQUE FEATURE

The Unique feature about this approach is that, most of the times during booking of a seat in a flight a customer is not aware about the reward points that he has availed after traveling and hence either he forgets about the points and pays full amount or the points gets expired. This system deals with this problem and ensures that during booking of a seat the customer reward points are used by default (If he has any) and the price paid is reduced, which makes this system reliable than other systems.

E-R DIAGRAM



Users

1)System Admin(Root)

- This is the user who will be responsible for the whole database management system. He will have rights and privileges to do anything to the database for ex: Inserting data, Updating data, Deleting data, Creating new procedures, functions, triggers to make functionality more smoother and efficient.

2)Customer Role

- This is the user who will be able to just view the **Customer** Table and he has access to view his itinerary through a created stored procedure for the same.
- The syntax for creating this particular user as well as granting the required access to him is as follows:

create user 'Customer'@'localhost' identified by 'customer';
grant select on testproject.customer to 'Customer'@'localhost';
grant execute on procedure testproject.getitinerary to 'Customer'@'localhost';

3)Booking Agent Role

- This user will be responsible for helping the customers for with providing a service for booking a ticket for a particular flight.
- He will be able to search flights given source and destination and he will help a particular customer with booking a seat in a given flight.
- This user has access just to **Flight**, **Customer** and **Ticket** table and has access to some procedures used during searching and booking a seat in a flight.
- The syntax for creating this user is as follows:

create user 'BookingAgent'@'localhost' identified by 'agent';
grant select on testproject.flight to 'BookingAgent'@'localhost';
grant select on testproject.customer to 'BookingAgent'@'localhost';
grant select, update, delete on testproject.ticket to 'BookingAgent'@'localhost';
grant execute on procedure testproject.searchFlights to 'BookingAgent'@'localhost';
grant execute on procedure testproject.checkAvailableSeats to 'BookingAgent'@'localhost';
grant execute on procedure testproject.BookTicket to 'BookingAgent'@'localhost';

4)Air Traffic Controller Role

- This user has access to the gain air traffic on a particular city's airport.
- The user has also access to a view which is responsible for displaying the schedule of all the flights.
- The syntax for creating this user is as follows:

create user 'AirTrafficController'@'localhost' identified by 'air';
grant execute on function testproject.getTraffic to 'AirTrafficController'@'localhost';
grant select on testproject.displayScheduleData to 'AirTrafficController'@'localhost';

SCREENSHOTS OF SOME OF THE PROCEDURES, FUNCTIONS, VIEWS AND TRIGGERS USED FOR ANALYTICS

1) This Procedure is used to search flights given the source and destination.

```
delimiter //
create procedure searchFlights(IN CustomerSource varchar(20), IN Customerdestination varchar(20))
begin
select f.flightId, a.AirlinerName, f.status, f.DateAndTime, S.SourceName, d.DestinationName, f.flightDuration
 from flight f
 inner join Source s on f.FlightSourceId = s.SourceId
 inner join Destination d on f.FlightDestinationId= d.DestinationId
 inner join Airliner a on a.AirlinerId = f. flightAirlinerId where s.SourceName = CustomerSource && d.DestinationName = CustomerDestination;
end //
```

2) This procedure is used to check availability of the seats given a particular flight ID.

```
delimiter //
create procedure checkAvailableSeats(IN BookingFlightId int)
begin
select seatId,seatFlightId, Availability, SeatClassType,SeatClassPrice
from seat
inner join seatclass on seat.classId = seatclass.seatClassId
where seatflightId = (select flightid from flight where flightId = BookingFlightId) order by seatId;
end //
```

---

3) This procedure is used to book a ticket for a particular customer given Customer ID and SeatId which is to be booked. The transaction is included in the procedure and the isolation level is kept serializable so that no two customer can book the same seat at single time.

```
delimiter //
create procedure BookTicket(IN bookingSeatId int, IN passengerId int)
begin
start transaction;
set transaction isolation level serializable;

insert into ticket(TicketCustomerId ,TicketSeatId) values (passengerId, BookingSeatId);

update ticket
set CustomerName = (Select concat(FirstName,' ' ,LastName) as Name from Customer where customerId = passengerId)
where TicketSeatId = bookingSeatId;

update ticket
set pricePaid = (select seatclassPrice from seatclass where SeatClassId = (Select classId from seat where seatId = bookingSeatId))
                                                                    - ( select rewardpoints from customer where customerId = passengerId)

where ticketSeatId = bookingSeatid;

update customer
set rewardpoints = 0
where customerId = passengerId;

commit;
end //
```

---

4) After a particular customer has booked a seat then this trigger ensures that the Seat availability of the booked seat is kept ‘N’ so that it depicts that it is booked.

```
Delimiter //
create trigger DeleteBookedSeats
after insert
on Ticket
for each row
begin
update seat
set availability = 'N'
where seatId = new.ticketSeatId;
end //
```

---

5) This view helps air traffic controller to gather information about all the flights which are being scheduled as well as landed.

```
create view displayScheduleData
as
select flightId, AirlinerName,SourceName, DestinationName,status from Flight f inner join Airliner a
 on f.FlightAirlinerId = a.airlinerId
 inner join Source s
 on f.flightSourceId = s.SourceId
 inner join Destination d
 on f.FlightDestinationId= d.DestinationId;
```

---

6) This procedure forwards the time for the given flight No and check what will be the status of this particular flight after provided hours.

```
delimiter //
create procedure timelapse(IN flightNo int, IN hour int)
begin
declare var datetime;
declare var1 datetime;
declare var3 int;
declare var4 datetime;
declare var5 datetime;
set var5= (select dateAndtime from flight where flightId = flightNo);
set var3 = (select flightDuration from flight where flightId = flightNo);
set var4 = (select addtime(@var5, concat(var3, ':00:00')));
set var1 =( select addtime(@var5, concat(hour,':00:00')));
select var5, var4, var3, var1;
update flight
set status = 'landed'
where var4- var1 = 0 && flightId = flightNo;
end//
```

---

7) This trigger ensures that whenever a customer travels with any flight he is awarded 100 points.

```
delimiter //
create trigger GiveRewardPoints
after
update on
flight
for each row
begin
if new.status = 'landed'
then
update customer set rewardpoints = rewardpoints+100
where customerid in(select t.ticketCustomerId from seat s inner join ticket t on s.seatid = t.ticketSeatId where s.seatflightid=new.flightid);
end if;
end //
```

---

8) This function is used by the air traffic controller to get the count of traffic on a particular airport provided its location, which can help him in making the appropriate decisions.

```
delimiter //
create function getTraffic (s varchar(20))
returns int deterministic
begin
declare value int;
set value = (select count(*) from flight where FlightSourceId =
                                                    (Select SourceId from Source where SourceName = s) ll FlightDestinationId
                                                      = (Select DestinationId from Destination where DestinationName = s));
return value;

end //
```

---

9) This procedure helps the customer to view his itinerary so that he is aware about his flight timings and the airliner which he is traveling with.

```
delimiter //
create procedure getItinerary(IN CustomerNo int)
begin
select f.flightId, ticketId,s.seatId, CustomerName ,  SourceName, DestinationName,PricePaid, f.status, f.DateAndTime,f.flightduration,a.airlinerName
from ticket t
inner join seat s on t.ticketSeatId = s.SeatId
inner join flight f on s.seatFlightId = f.flightId
inner join Airliner a on f.FlightAirlinerId = a.AirlinerId
inner join Source sou on f.FlightSourceId = sou.SourceId
inner join Destination d on f.flightDestinationId = d.DestinationId
inner join customer c on t.TicketCustomerId = c.CustomerId
where c.customerId = customerNo;
end //
```
