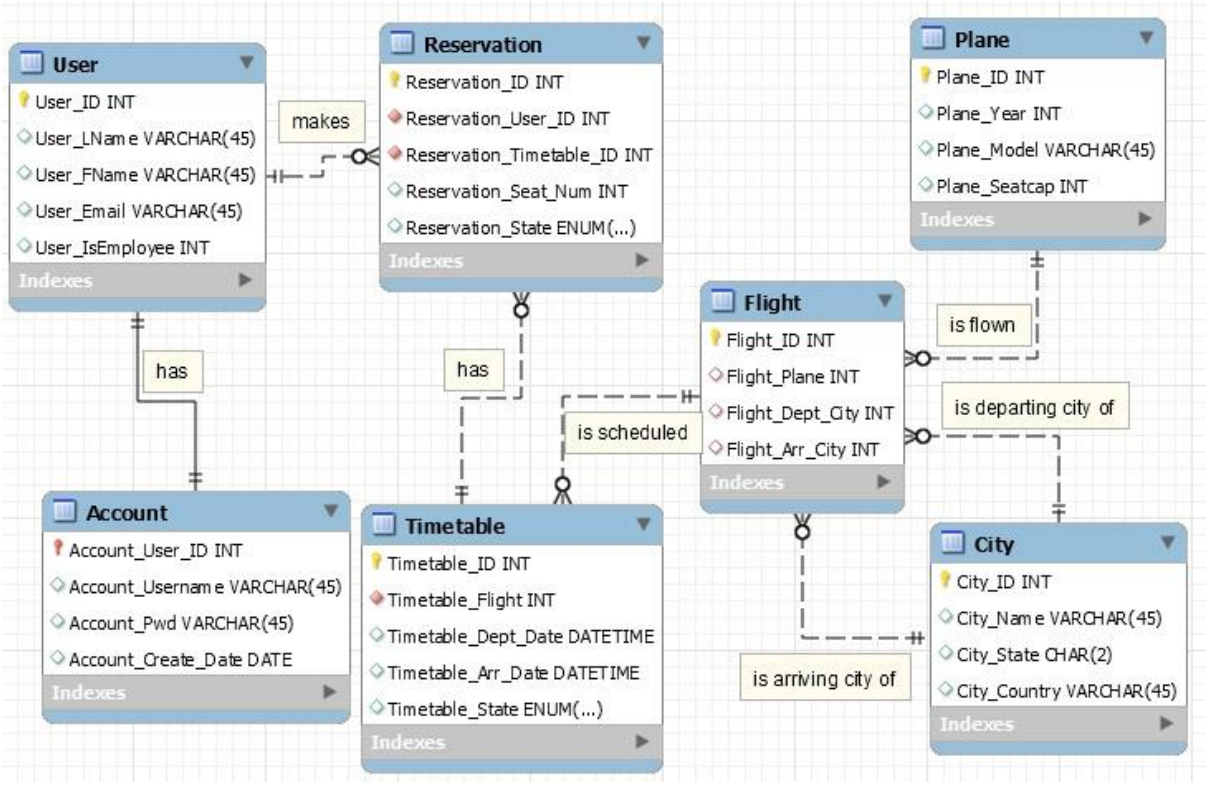
CS61 Homework #5
Min Hyung (Daniel) Kang

Database : Almost There Airlines

I. Overview

This database is implemented as a reservation system for an airline company *Almost There Airlines*. It will be used to manage information about the customers as well as allow the customers to reserve flights. An employee will be able to view the flight schedules, make reservations for customers (or themselves if necessary) and cancel/make reservations for customers. A customer will be able to view flight schedules, view his/her reservations, and make or cancel reservations for oneself. Refer to the following ERD for the design of the database

II. ERD



Note that every table is using IDs as primary keys. True, there are some tables which can use composite primary keys. For example, each row of Timetable can be identified with flight and the departure time. However, such composite keys make the query process more complex and time-consuming. Hence, I used ID numbers for each relation for easier and straightforward reference. Since every table uses one attribute as their primary key, there is obviously no partial dependency. As mentioned before with the case of Timetable, there may exist transitive dependencies. However, I chose to use ID despite that for computational and readability reasons. Moreover, it is sensible for a flight to have reference id in real world. For more information regarding the relations, please refer to the rest of this document.

III. Data Dictionary (Tabular Form)

Table Name	Attribute Name	Contents	Type	Format	Range	Required	Unique	PK or FK	FK Table	Default
Account	Account_User_ID	User ID	INT	###...	Greater than 0	Y	Y	PK,FK	User	
	Account_Username	User Username	Varchar(45)	xxxxxxxx…	NA	Y	Y			
	Account_Pwd	User Password	Varchar(45)	xxxxxxxx…	NA	Y	N			
	Account_Create_Date	Date created	Date	YYYY-MM-DD	NA	Y	N			
City	City_ID	ID of city	INT	####	1000-9999	Y	Y	PK		
	City_Name	Name of City	Varchar(45)	xxxxxx…	NA	Y	N			
	City_State	State of City	Char(2)	XX	NA	N	N			
	City_Country	Country of City	Varchar(45)	xxxxxxxxx…	NA	Y	N			
Flight	Flight_ID	ID of flight	INT	#####	10000-99999	Y	Y	PK		
	Flight_Plane_ID	ID of plane	INT	###	100-999	Y	N	FK	Plane	
	Flight_Dept_City	ID of departure city	INT	####	1000-9999	Y	N	FK	City	
	Flight_Arr_City	ID of arrival city	INT	####	1000-9999	Y	N	FK	City	
Plane	Plane_ID	ID of plane	INT	###	100-999	Y	Y	PK		
	Plane_Year	Year plane acquired	INT	####	1000-9999	Y	N			
	Plane_Model	Model of plane	Varchar(45)	xxxxx…	NA	Y	N			
	Plane_SeatCap	Seat capacity of plane	INT	###	1-999	Y	N			
Reservation	Reservation_ID	ID of reservation	INT	####...	Greater than 0	Y	Y	PK		
	Reservation_User_ID	ID of User	INT	####….	Greater than 0	Y	N	FK	User	
	Reservation_Timetable_ID	ID of timetable	INT	####...	Greater than 0	Y	N	FK	Timetable	
	Reservation_Seat_Num	Reserved Seat Num	INT	###	1-999	Y	N			
	Reservation_State	State of reservation	ENUM	Reserved/ Cancelled/ Past/In Flight		Y	N			
Timetable	Timetable_ID	ID of schedule	INT	####...	Greater than 0	Y	Y	PK		
	Timetable_Flight	ID of flight	INT	#####	10000-99999	Y	N	FK	Flight	
	Timetable_Dept_Date	Departure date/time	Datetime	Yyyy-mm-dd hh:mm:ss	NA	Y	N			
	Timetable_Arr_Date	Arrival date/time	Datetime		NA	Y	N			
	Timetable_State	State of schedule	ENUM	Scheduled/Cancelled/Past/In Flight		Y	N			
User	User_ID	ID of User	INT	###....	Greater than 0	Y	Y	PK		
	User_LName	Last name of user	Varchar(45)	Xxxxx…	NA	Y	N			
	User_FName	First name of user	Varchar(45)	Xxxxx…	NA	Y	N			
	User_Email	Email of User	Varchar(45)	Xxxx…	NA	N	N			
	User_IsEmployee	User is employee	INT	#	0 or 1	Y	N			-1

IV. Definitions, assumptions and notes (In order of appearance)

I. Tables

A. User

i. Description : Contains information about users and whether they are employees or not.

ii. Note : used data from Sakila Database. Also, the default value of User_IsEmployee is -1. This is so that the trigger can be used to determine if the user is employee or customer if it had not been specified. (Just for testing purpose). I am using INT instead of Boolean so that if necessary, I can specify different INTs to represent different types of employees.

B. Account

i. Description : Contains account information (username, password, join date) for each user.

ii. Assumption : Since we are simply testing the data, I made username the last name of the corresponding user and the password as username with 0 in the end. Hence, I am assuming (just for test purpose) that the last names of users are unique.

C. City

i. Description : Contains information about cities where airports are at.

ii. Note : The states can be null, for international cities often do not have states. I did not include the airport names for I did not think it to be that important and it was not required. I do not make any assumptions with respect to uniqueness of city names.

D. Plane

i. Description : Keeps track of the actual, physical planes that company possesses.

ii. Note : Plane models and capacities were mostly made up.

E. Flight

i. Description : Represents a predefined journey of a designated plane from a city to a city.

F. Timetable

i. Description : Represents a single planned journeys of each date. A flight has various corresponding timetables if the flight is flown on numerous days. Each timetable has departing city and time, as well as arriving city and time.

ii. Note : It may sound awkward, but throughout the code and document, such an entity as described above will be called a timetable. We do not consider timezones or possible differences that may occur from different timezones.

G. Reservation

i. Description : Shows the customers' reservations. Includes information about the timetable and chosen seat.

ii. Note : One may make a new reservation or cancel preexisting one. Also, one can make numerous reservations from a single timetable.

II. Trigger

A. ChooseEmployee

i. Description : Before inserting values on the User table, if isEmployee field is -1, randomly set him/her as an employee or a regular customer. (Just for inputting data for test purpose)

III. Procedure

A. updateTimetable

i. Updates the timetable so that all the timetables which passed its arrival time is corrected as “past”, those which passed its departure time but not arrival time is “on flight”, and those which did not pass its departure time is “scheduled.” All the cancelled ones remain cancelled.

B. updateReservation

i. Updates the reservation states to match the timetable states, except the case when the user’s reservation was already cancelled. In that case, leave the user’s reservation as cancelled.

C. setupDates

i. Input : beginDate, endDate

ii. Given inputs, set the global variables BeginDate and EndDate as those values or default values if NULL is given as parameters. This sets up variables for other functions.

D. viewReservationsDateProcedure

i. Input : username, pwd, beginDate, endDate

ii. Procedure version to show the reservations between the two specified dates. Only employees may use this

E. viewReservationPerCustomer

i. Input : username, pwd (password), targeted (ID (Not username) of customer to view reservations of), beginDate, endDate

ii. Shows all the reservations made by the specific customer only if this was requested by an employee or the customer oneself. If date is input, only look for values between those dates

F. viewReservationPerFlight

i. Input : username, pwd(password), flightID (ID of the flight to view reservations for)

ii. Shows all the reservations made for a particular flight only if this was requested by an employee

G. viewReservationPerTimetable

i. Input : username, pwd(password), timetableID (ID of the timetable to view reservations for)

ii. Shows all the reservations made for a particular timetable only if this was requested by an employee

H. viewSeatsForTimetable

i. Input : timetableID(ID of the timetable to view available seats of)

ii. Shows all the available seats for the specified timetable

I. reserveSeat

i. Input : username(of person making reservation), pwd(of person making the reservation) ,timetableID (the desired ID of timetable), seat (the desired seat of reservation), userID (customer for whom the reservation will be)

ii. Output : Tells if the reservation was made or not

iii. The user can reserve a seat of a specific timetable, and will be told if the reservation works or not. Employee can make reservation for anyone, while user can only make reservation for oneself.

J. cancelSeat

i. Input : username, pwd (password), reservationID (reservation to cancel)

ii. Output : Tells if the cancellation was made or not

iii. The user can cancel one's own reservation, and the employees can cancel any reservation.

K. Login

i. Input : username, pwd

ii. Output : result (whether user successfully logged in or not)

iii. Logs a user in

L. Logout

i. Deletes the credentials and logs out

IV. View

A. viewSchedule

i. Show all the planned schedules. (This should be accessible to anyone)

B. viewReservations

i. Show all the reservations (This should be only accessible to the employees)

C. viewReservationsDate

i. Show all the view within specified dates. (Dates should be specified for variable @StartDate and @EndDate beforehand)

V. Function

A. fn_getStartDate

i. Output : @StartDate (The starting date of the query)

ii. Simply returns the beginning date to be used in other procedures, views, etc

B. fn_getEndDate

i. Output :@EndDate (The ending date of the query)

ii. Simply returns the ending date to be used in other procedures, views, etc

C. isEmployee

i. Input : username (username of user), passwd (password of user)

ii. Returns : Boolean (Whether the user is employee or not)

iii. Given user id and password, tells if the user is an employee or not

D. isAvailable

i. Input : timetableID(the timetable one wants to check), seat (the seat to check availability of)

ii. Returns : Boolean (whether the chosen seat is available or not)

iii. Given a timetable and seat, tells if that seat is available

E. fn_getCurrentUser

i. Output : @CurrentUser (the username of the current user)

ii. Simply returns who is currently logged on to the system

F. fn_getCurrentPwd

i. Output : @CurrentPwd (the password of the current user)

ii. Simply returns password of who is currently logged on to the system