Flight Reservation

Introduction to Databases

Fall 2017

Team Members

Sophia Tan

David Ruvalcaba

Amir Radman

* **Database Schema**

drop database if exists flightReservation;

create database flightReservation;

use flightReservation;

drop table if exists user;

create table user

(uID INT AUTO\_INCREMENT Primary Key,

uNAME VARCHAR(30) NOT NULL,

age INT);

ALTER Table user AUTO\_INCREMENT = 100;

Drop table if exists flightList;

create table flightList

(fID INT AUTO\_INCREMENT Primary Key,

aName VARCHAR(30) NOT NULL,

numSeats INT NOT NULL);

ALTER Table flightList AUTO\_INCREMENT = 200;

Drop table if exists reservation;

create table reservation

(uID INT,

fID INT,

reservedDate Date NOT NULL Default '0000-00-00' ,

updatedAt timestamp NOT NULL on update current\_timestamp default current\_timestamp,

PRIMARY KEY (uID,fID,reservedDate),

FOREIGN KEY (uID) references user (uID) on delete cascade,

FOREIGN KEY (fID) references flightList (fID) on delete cascade);

Drop table if exists seat;

create table seat

(uID INT,

fID INT,

seatNumber INT NOT NULL auto\_increment UNIQUE,

PRIMARY KEY (uID,fID,seatNumber),

FOREIGN KEY (uID) references user (uID) on delete cascade,

FOREIGN KEY (fID) references flightList (fID) on delete cascade);

ALTER Table seat AUTO\_INCREMENT = 300;

Drop table if exists canceledReservation;

create table canceledReservation

(uID INT,

fID INT,

canceledDate Date NOT NULL Default '0000-00-00',

PRIMARY KEY (uID,fID)

);

Drop table if exists Archive;

create table Archive

(uId INT,

fId INT,

updatedAt timestamp NOT NULL);

* **Two Trigger(s)**

DROP TRIGGER IF EXISTS INSERTTORES;

delimiter //

CREATE TRIGGER INSERTTORES

AFTER INSERT ON RESERVATION

FOR EACH ROW

BEGIN

insert into seat(uID,fID) values(new.uID,new.fID);

END;//

delimiter ;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

DROP TRIGGER IF EXISTS INSERTTOCANCEL;

delimiter //

CREATE TRIGGER INSERTTOCANCEL

AFTER INSERT ON CANCELEDRESERVATION

FOR EACH ROW

BEGIN

delete from reservation where new.uID = uID and new.fID = fID;

delete from seat where new.uID = uID and new.fID = fID;

END;//

delimiter ;

* **Stored Procedure(s)**

drop PROCEDURE if exists archivedReservation;

delimiter //

create PROCEDURE archivedReservation(IN value varchar(55))

BEGIN

insert into Archive select uid,fid,updatedAt from reservation where date(updatedAt) > value ;

delete from reservation where date(updatedAt) > value;

END; //

delimiter ;

* Functional Requirements 🡨 🡪 SQL Query

|  |  |
| --- | --- |
| **Functional Requirements** | **Corresponding SQL Query** |
| Adding Users | Insert into user(uName,age) values(?,?) |
| Deleting Users | Delete from user where uID = ? |
| Adding Reservation | Insert into reservation (uID , fID,reservedDate ) values(?,?,current\_Date()) |
| Canceling Reservation | Delete from reservation where uid = ? and fid = ? |
| Users with reservation who have not canceled flights AT ALL | select distinct uid  from reservation where uid NOT IN (select uid from canceledreservation); |
| Adding Flights | Insert into flightList(aName, numSeats)  values(?,?) |
| Deleting Flights | Delete from flightList where aName = ?  Or  Delete from flightList where fid = ? |
| User’s reservation with age > \_\_ | Select uID from user JOIN reservation using(uID) where age > ? |
| User’s cancellation with age > \_\_ | Select uID from user JOIN canceledReservation using(uID) where age > ? |
| Finding flights using airline name | Select aName from flightList group by aName having (aName = ?) |
| Finding flights with \_\_ # seats | Select fid,aName from flightList where numSeats = ? |
| Flights that have not been reserved | select \* from flightList FL left Outer JOIN reservation R on FL.fid = R.fid; |
| High Season Month(s): Months with more than 3 reservations | select MonthName(str\_to\_date(Month(reservedDate),'%m')) as 'High Season Month(s)' from reservation group by reservedDate having(count(\*)>=3); |
| Users who have canceled more than \_\_ reservation | Select uid,uname from user u1 where ? < (select count(\*) from canceledreservation where u1.uid = uid group by uid); |
| Average age of users who reserved | Select avg(age) as averageAge from user JOIN reservation USING(uid) |

* Screenshots