Online-airlines-reservation-system

The Airline Reservation system is developed to fulfill the client's requirements which are given below:

1. This application must be user-friendly and easy to use.

2. Customer can check the timings of flights which travel between two places on a specified date and their availability through online at anytime and anywhere, by using this application.

3. A customer should get a chance of take a trip of continuous travel in which flights are changed frequently by using this web application.

4. The basic concepts of airline ticket reservation should be maintained by this web application.

5. When a specific flight gets delayed due to technical problems this application should send mobile alerts to the travelers, which helps the travelers in reservation of tickets.

Modules

- Flight Searching
- Flight Booking
- Flight Cancellation
- Trip Deals

CREATING FARE SCHEMA

Step 1: Connect to database
C:\>sqlplus system/manager@xe

Step2: Create tablespace

```
CREATE TABLESPACE tbs_fareuser DATAFILE 'tbs_fareuser.dat' SIZE
10M AUTOEXTEND ON;
```

Note: alter session set "_ORACLE_SCRIPT"=true; This is required in Oracle 12c

Step3: Create a new user in Oracle

```
CREATE USER fareuser IDENTIFIED BY aspire123 DEFAULT TABLESPACE
tbs_fareuser QUOTA unlimited on tbs_fareuser;
```

Note: In oracle, a schema is automatically created when a user is created.

Step4: Grant permissions

```
GRANT create session TO fareuser;
GRANT create table TO fareuser;
GRANT create sequence TO fareuser;
```

Step5: Disconnect from system account and connect to fareuser

```
Sql>exit
C:\>sqlplus fareuser/aspire123@xe
```

Step6: Create tables and sequences

```
drop table fare cascade constraints;
drop sequence fare_seq;

create table fare (id number(19) primary key, fare
varchar2(255), flight_date varchar2(255), flight_number
varchar2(255));

create sequence fare_seq start with 1 increment by 1;
```

Step7: Insert records

```
insert into fare(id, fare, flight_date, flight_number) values
(fare_seq.nextVal, '100', '22-JAN-16', 'BF100');
insert into fare(id, fare, flight_date, flight_number) values
(fare_seq.nextVal, '101', '22-JAN-16', 'BF101');
insert into fare(id, fare, flight_date, flight_number) values
(fare_seq.nextVal, '102', '22-JAN-16', 'BF102');
insert into fare(id, fare, flight_date, flight_number) values
(fare_seq.nextVal, '103', '22-JAN-16', 'BF103');
insert into fare(id, fare, flight_date, flight_number) values
(fare_seq.nextVal, '104', '22-JAN-16', 'BF104');
insert into fare(id, fare, flight_date, flight_number) values
(fare_seq.nextVal, '105', '22-JAN-16', 'BF105');
insert into fare values (fare_seq.nextVal, '106', '22-JAN-16',
'BF106');

commit;
```

Step8: Read data from FAREUSER schema

```
SELECT * FROM "FAREUSER"."FARE";
```

ID	FLIGHT_NUMBER	FLIGHT_DATE	FARE
1	BF100	22-JAN-16	100
2	BF101	22-JAN-16	101
3	BF102	22-JAN-16	102
4	BF103	22-JAN-16	103
5	BF104	22-JAN-16	104
6	BF105	22-JAN-16	105
7	BF106	22-JAN-16	106

CREATING SEARCH SCHEMA

Step 1: Connect to database (ignore if already connected)
C:\>sqlplus system/manager@xe

Step2: Create tablespace

```
CREATE TABLESPACE tbs_searchuser DATAFILE 'tbs_searchuser.dat'
SIZE 10M AUTOEXTEND ON;
```

Note: alter session set "_ORACLE_SCRIPT"=true; This is required in Oracle 12c

Step3: Create a new user in Oracle

```
CREATE USER searchuser IDENTIFIED BY aspire123 DEFAULT
TABLESPACE tbs_searchuser QUOTA unlimited on tbs_searchuser;
```

Note: In oracle, a schema is automatically created when a user is created.

Step4: Grant permissions

```
GRANT create session TO searchuser;
GRANT create table TO searchuser;
GRANT create sequence TO searchuser;
```

Step5: Disconnect from system account and connect to searchuser

```
Sql>exit
C:\>sqlplus searchuser/aspire123@xe
```

Step6: Create tables and sequences

```
drop table fare cascade constraints;
drop table flight cascade constraints;
drop table inventory cascade constraints;

drop sequence fare_seq;
drop sequence flight_seq;
drop sequence inventory_seq;

create sequence fare_seq start with 1 increment by 1;
create sequence flight_seq start with 1 increment by 1;
create sequence inventory_seq start with 1 increment by 1;

create table fare1 (fare_id number(19) primary key, currency
varchar2(255), price varchar2(255));

create table inventory (inv_id number(19) primary key, count
number(10) not null);

create table flight (id number(19) primary key, destination
varchar2(255), flight_date varchar2(255),
```

```
flight_number varchar2(255), origin varchar2(255), fare_id
number(19) references fare1(fare_id), inv_id number(19)
references inventory(inv_id));

Step7: Insert records
insert into fare1 values (fare_seq.nextVal, 'USD', 100);
insert into fare1 values (fare_seq.nextVal, 'USD', 101);
insert into fare1 values (fare_seq.nextVal, 'USD', 102);
insert into fare1 values (fare_seq.nextVal, 'USD', 103);
insert into fare1 values (fare_seq.nextVal, 'USD', 104);
insert into fare1 values (fare_seq.nextVal, 'USD', 105);
insert into fare1 values (fare_seq.nextVal, 'USD', 106);

insert into inventory (count, inv_id) values (100,
inventory_seq.nextVal);
insert into inventory (count, inv_id) values (100,
inventory_seq.nextVal);
insert into inventory (count, inv_id) values (100,
inventory_seq.nextVal);
insert into inventory (count, inv_id) values (100,
inventory_seq.nextVal);
insert into inventory (count, inv_id) values (100,
inventory_seq.nextVal);
insert into inventory (count, inv_id) values (100,
inventory_seq.nextVal);
insert into inventory (count, inv_id) values (100,
inventory_seq.nextVal);

insert into flight (id, flight_number, origin, destination,
flight_date, fare_id, inv_id) values (flight_seq.nextVal,
'BF100', 'SEA', 'SFO', '22-JAN-16', 1, 1);
insert into flight (id, flight_number, origin, destination,
flight_date, fare_id, inv_id) values (flight_seq.nextVal,
'BF101', 'NYC', 'SFO', '22-JAN-16', 2, 2);
insert into flight (id, flight_number, origin, destination,
flight_date, fare_id, inv_id) values (flight_seq.nextVal,
'BF102', 'CHI', 'SFO', '22-JAN-16', 3, 3);
insert into flight (id, flight_number, origin, destination,
flight_date, fare_id, inv_id) values (flight_seq.nextVal,
'BF103', 'HOU', 'SFO', '22-JAN-16', 4, 4);
insert into flight (id, flight_number, origin, destination,
flight_date, fare_id, inv_id) values (flight_seq.nextVal,
'BF104', 'LAX', 'SFO', '22-JAN-16', 5, 5);
insert into flight (id, flight_number, origin, destination,
flight_date, fare_id, inv_id) values (flight_seq.nextVal,
'BF105', 'NYC', 'SFO', '22-JAN-16', 6, 6);
```

```
insert into flight (id, flight_number, origin, destination,
flight_date, fare_id, inv_id) values (flight_seq.nextVal,
'BF106', 'NYC', 'SFO', '22-JAN-16', 7, 7);
commit;
```

Step8: Read data from SEARCHUSER schema

```
SELECT * FROM "SEARCHUSER"."FARE";
```

FARE_ID	FARE	CURRENCY
1	100	USD
2	101	USD
3	102	USD
4	103	USD
5	104	USD
6	105	USD
7	106	USD

```
SELECT * FROM "SEARCHUSER"."INVENTORY";
```

INV_ID	COUNT
1	100
2	100
3	100
4	100
5	100
6	100
7	100

```
SELECT * FROM "SEARCHUSER"."FLIGHT";
```

ID	FLIGHT_NUMBER	FLIGHT_DATE	ORIGIN	DESTINATION	FARE_ID	INV_ID
1	BF100	22-JAN-16	SEA	SFO	1	1
2	BF101	22-JAN-16	NYC	SFO	2	2
3	BF102	22-JAN-16	CHI	SFO	3	3
4	BF103	22-JAN-16	HOU	SFO	4	4
5	BF104	22-JAN-16	LAX	SFO	5	5
6	BF105	22-JAN-16	NYC	SFO	6	6
7	BF106	22-JAN-16	NYC	SFO	7	7

CREATING BOOKING SCHEMA

Step 1: Connect to database (ignore if already connected)
C:\>sqlplus system/manager@xe

Step2: Create tablespace

```
CREATE TABLESPACE tbs_bookinguser DATAFILE 'tbs_bookinguser.dat'
SIZE 10M AUTOEXTEND ON;
```

Note: alter session set "_ORACLE_SCRIPT"=true; This is required in Oracle 12c

Step3: Create a new user in Oracle

```
CREATE USER bookinguser IDENTIFIED BY aspire123 DEFAULT
TABLESPACE tbs_bookinguser QUOTA unlimited on tbs_bookinguser;
```

Note: In oracle, a schema is automatically created when a user is created.

Step4: Grant permissions

```
GRANT create session TO bookinguser;
GRANT create table TO bookinguser;
GRANT create sequence TO bookinguser;
```

Step5: Disconnect from system account and connect to bookinguser

```
Sql>exit
C:\>sqlplus bookinguser/aspire123@xe
```

Step6: Create tables and sequences

```
drop table inventory cascade constraints;
drop table booking_record cascade constraints;
drop table passenger cascade constraints;

drop sequence booking_seq;
drop sequence inventory_seq;
drop sequence passenger_seq;

create sequence booking_seq start with 1 increment by 1;
create sequence inventory_seq1 start with 1 increment by 1;
create sequence passenger_seq start with 1 increment by 1;

create table booking_record (id number(19) primary key,
booking_date timestamp, destination varchar2(255), price
varchar2(255), flight_date varchar2(255), flight_number
varchar2(255), origin varchar2(255), status varchar2(255));

create table inventory (id number(19) primary key, available
number(10) not null, flight_date varchar2(255), flight_number
varchar2(255));

create table passenger (id number(19) primary key, first_name
varchar2(255), gender varchar2(255), last_name varchar2(255),
booking_id number(19) references booking_record(id));
```

Step7: Insert records

```
insert into inventory1 (flight_number, flight_date, available, id)
values ('BF100', '22-JAN-16', 100, inventory_seq1.nextVal);
insert into inventory1 (flight_number, flight_date, available, id)
values ('BF101', '22-JAN-16', 100, inventory_seq1.nextVal);
```

```
insert into inventory1 (flight_number, flight_date, available, id)
values ('BF102', '22-JAN-16', 100, inventory_seq1.nextVal);
insert into inventory1 (flight_number, flight_date, available, id)
values ('BF103', '22-JAN-16', 100, inventory_seq1.nextVal);
insert into inventory1 (flight_number, flight_date, available, id)
values ('BF104', '22-JAN-16', 100, inventory_seq1.nextVal);
insert into inventory1 (flight_number, flight_date, available, id)
values ('BF105', '22-JAN-16', 100, inventory_seq1.nextVal);
insert into inventory1 (flight_number, flight_date, available, id)
values ('BF106', '22-JAN-16', 100, inventory_seq1.nextVal);

commit;
```

Step8: Read data from BOOKINGUSER schema

```
SELECT * FROM "INVENTORY1";
```

ID	FLIGHT_NUMBER	FLIGHT_DATE	AVAILABLE
1	BF100	22-JAN-16	100
2	BF101	22-JAN-16	**99**
3	BF102	22-JAN-16	100
4	BF103	22-JAN-16	100
5	BF104	22-JAN-16	100
6	BF105	22-JAN-16	100
7	BF106	22-JAN-16	100

```
SELECT * FROM "BOOKINGUSER"."BOOKING_RECORD";
```

ID	BOOKING_DATE	ORIGIN	DESTINATION	PRICE	FLIGHT_DATE	FLIGHT_NUMBER	STATUS
1	2017-06-06 20:46:01	NYC	SFO	101	22-JAN-16	BF101	**BOOKING_CONFIRMED**

```
SELECT * FROM "BOOKINGUSER"."PASSENGER";
```

ID	FIRST_NAME	LAST_NAME	GENDER	BOOKING_ID
1	Gean	Franc	Male	1

CREATING CHECKIN SCHEMA

Step 1: Connect to database (ignore if already connected)
C:\>sqlplus system/manager@xe

Step2: Create tablespace

```
CREATE TABLESPACE tbs_checkinuser DATAFILE 'tbs_checkinuser.dat'
SIZE 10M AUTOEXTEND ON;
```

Note: alter session set "_ORACLE_SCRIPT"=true; This is required in Oracle 12c

Step3: Create a new user in Oracle

```
CREATE USER checkinuser IDENTIFIED BY aspire123 DEFAULT
TABLESPACE tbs_checkinuser QUOTA unlimited on tbs_checkinuser;
```
Note: In oracle a schema is created when a user is created.

Step4: Grant permissions
```
GRANT create session TO checkinuser;
GRANT create table TO checkinuser;
GRANT create sequence TO checkinuser;
```

Step5: Disconnect from system account and connect to checkinuser
```
Sql>exit
C:\>sqlplus checkinuser/aspire123@xe
```

Step6: Create tables and sequences
```
drop table check_in_record cascade constraints;
drop sequence checkin_seq;

create sequence checkin_seq start with 1 increment by 1;

create table check_in_record (id number(19)primary key,
booking_id number(19) not null, check_in_time timestamp,
first_name varchar2(255), flight_date varchar2(255),
flight_number varchar2(255), last_name varchar2(255),
seat_number varchar2(255));
```

Step7: Insert records
No need to insert data manually

Step8: Read data from CHECKINUSER schema
```
SELECT * FROM "CHECKINUSER"."CHECK_IN_RECORD";
```

ID	BOOKING_ID	CHECK_IN_TIME	FIRST_ NAME	LAST_NAME	FLIGHT_DATE	FLIGHT_NUMBER	SEAT_NUMBER
1	1	2017-06-06 21:18:46	Gean	Franc	22-JAN-16	BF101	**28A**

Other useful commands

DROP TABLESPACE tbs_testuser INCLUDING CONTENTS AND DATAFILES;

DROP USER testuser;