AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH
(AIUB)
Faculty of Science and Technology
Department of Computer Science and Engineering

MID TERM REPORT

SECTION: B & C

Advance Database Management System

Submitted to: Rezwan Ahmed Assistant Professor Computer Science Department of CSE	Semester: Summer 2021-22 Date of Submission: 25/08/2022

Project Name: Air-Ticket Online Booking System

Serial no	Student's Name	ID	Dept.
1	SHADRIL HASSAN SHIFAT(C)	20-42451-1	CSE
2	ANIKA SABA IBTE SUM(C)	20-43242-1	CSE
3	PRITOM DEBNATH(B)	20-42414-1	CSE
4	MAHBUBA SHARMIN RUBA(B)	19-41565-3	CSE

Table of Contents

Searching and Advance Searching ……..……..…3
Sequence ……..……..……3
View…….……4
Procedure……...4-6
Trigger…….....7

✓ Searching and Advance Searching:

1. Flight with Minimum Cost: select * from flight where flight_cost=(select min(flight_cost) from flight)

2. Manager Who Approved Maximum Flights: select mgr_id,mgr_name from manager where mgr_id in (select m.mgr_id from manager m, flight f where m.mgr_id=f.mgr_id group by m.mgr_id having count(m.mgr_id) in (select max(count(m.mgr_id)) from manager m, flight f where m.mgr_id=f.mgr_id group by m.mgr_id))

3. Customer With Maximum Booked Flight: select * from customer where customer_id in (select c.customer_id from customer c, flight f, booking b where c.customer_id=b.customer_id and b.flight_id=f.flight_id group by c.customer_id having count(c.customer_id) in (select max(count(c.customer_id)) from customer c, flight f, booking b where c.customer_id=b.customer_id and b.flight_id=f.flight_id group by c.customer_id))

4. Flight-wise total booked tickets: select f.flight_id, sum(t.total_ticket) as booked_tickets from flight f, ticket t, order_ticket ot where t.ticket_id=ot.ticket_id and ot.flight_id=f.flight_id group by f.flight_id

5. Business class flight lowest cost: select flight_id,departure,destination, departure_time from flight where flight_class='BUSINESS' and flight_cost in (select min(flight_cost) from flight where flight_class='BUSINESS')

6. Customers Who Booked Flights: select distinct c.customer_name, c.customer_email from customer c, ticket t where c.customer_id=t.customer_id and ticket_status='booked'

✓ Sequence:

1. create sequence seq_flight_id increment by 1 start with 1 nocache nocycle;
    a. INSERT INTO FLIGHT(flight_id, departure, destination, departure_time, arrival_time, flight_cost, flight_class, mgr_id) VALUES (seq_flight_id.NEXTVAL, 'dhaka','cox bazar', to_date ('5-aug-22 10:00 a.m.','dd-mon-yy hh:mi a.m.'), to_date ('5-aug-22 11:30 a.m.','dd-mon-yy hh:mi a.m.'), 7000, 'Economy',2);
2. create sequence seq_ticket_id increment by 1 start with 1 nocache nocycle;
    a. insert into TICKET values (seq_ticket_id.NEXTVAL, 1, 'booked',2);
3. create sequence seq_customer_id increment by 1 start with 1 nocache nocycle;
    a. insert into CUSTOMER(customer_id, customer_name, customer_pass, customer_email, customer_phn, mgr_id) VALUES (seq_customer_id.NEXTVAL, 'Abir', 'coolman', 'abir@yahoo.com','01754402481',1);
4. create sequence seq_mgr_id increment by 1 start with 1 nocache nocycle;
    a. INSERT INTO MANAGER(mgr_id, mgr_pass, mgr_name, mgr_email) VALUES(seq_mgr_id.NEXTVAL, 'tiger', 'shadril', 'shadrilhassan@outlook.com');

✓ View:

1. create or replace view pending_tickets as select distinct c.customer_id, c.customer_name, c.customer_email, t.ticket_id, t.total_ticket, t.ticket_status from customer c, ticket t where c.customer_id=t.customer_id and t.ticket_status='pending';

2. create or replace view customer_flight_details as select distinct c.customer_id, c.customer_name, f.flight_id, f.departure, f.destination, f.departure_time, t.ticket_id, t.total_ticket , t.total_ticket*f.flight_cost as TOTAL_FLIGHT_COST from flight f, ticket t, booking b, customer c where c.customer_id=t.customer_id and b.customer_id=c.customer_id and b.flight_id=f.flight_id;

3. create or replace view customer_flight_booking as select distinct c.customer_id, c.customer_name, c.customer_email, c.customer_phn, b.booking_id,b.flight_id from customer c, booking b where c.customer_id=b.customer_id;

4. create or replace view tickets_of_flights as select f.flight_id, sum(t.total_ticket) as total_tickets from flight f, ticket t, order_ticket ot where t.ticket_id=ot.ticket_id and ot.flight_id=f.flight_id group by f.flight_id;

✓ Procedure & Functions (including package and exception handling):

1. Package (1 procedure & 1 private function)

```
create or replace package p_customer_ticket as
procedure book_customer_ticket(c_id customer.customer_id%type, f_id flight.flight_id%type,
no_tickets ticket.total_ticket%type);
end p_customer_ticket;

create or replace package body p_customer_ticket
as
function valid_flight(f_id in flight.flight_id%type)
return boolean
is
cnt_f number;
begin
select count(*) into cnt_f from flight where flight_id=f_id;
if cnt_f>0 then
return true;
else
return false;
end if;
end valid_flight;
function valid_customer(c_id customer.customer_id%type)
return boolean
```

```
is
cnt_c number;
begin
select count(*) into cnt_c from customer where customer_id=c_id;
if cnt_c>0 then
return true;
else
return false;
end if;
end valid_customer;

procedure book_customer_ticket(c_id customer.customer_id%type, f_id flight.flight_id%type,
no_tickets ticket.total_ticket%type)
as
inval_flight exception;
inval_customer exception;
inval_ticket exception;
begin
if c_id<1 then
raise inval_customer;
elsif f_id<1 then
raise inval_flight;
elsif no_tickets<1 then
raise inval_ticket;
elsif valid_customer(c_id) AND valid_flight(f_id) then
insert into TICKET values (seq_ticket_id.NEXTVAL, no_tickets, 'pending',c_id);
insert into BOOKING values (seq_booking_id.NEXTVAL,c_id,f_id);
end if;
exception
when inval_flight then
raise_application_error(-20227,'Flight Not Exist');
when inval_customer then
raise_application_error(-20228,'Customer Not Exist');
when inval_ticket then
raise_application_error(-20229,'Invalid Ticket');
when others then
raise_application_error(-20230,'Something Went Wrong!');
end book_customer_ticket;

end p_customer_ticket;

begin
p_customer_ticket.book_customer_ticket(7,21,5);
end;
```

2. Package(2 procedures & 2 private functions)

```
create or replace package p_flight
as
procedure flight_cost_up(f_id flight.flight_id%type, f_cost flight.flight_cost%type);
procedure flight_departure_up(f_id flight.flight_id%type,f_dep flight.departure%type);
end p_flight;
```

```
create or replace package body p_flight
as
function valid_flight(f_id flight.flight_id%type)
return boolean
is
cnt_f number;
begin
select count(*) into cnt_f from flight where flight_id=f_id;
if cnt_f>0 then
return true;
else
return false;
end if;
end valid_flight;

procedure flight_cost_up(f_id flight.flight_id%type, f_cost flight.flight_cost%type)
as
inval_flight exception;
begin
if f_id<1 then
raise inval_flight;
elsif valid_flight(f_id) then
update flight set flight_cost=f_cost where flight_id=f_id;
end if;
exception
when inval_flight then
raise_application_error(-20225,'Flight Not Exist');
when others then
raise_application_error(-20226,'Something Went Wrong!');
end flight_cost_up;

procedure flight_departure_up(f_id flight.flight_id%type,f_dep flight.departure%type)
as
inval_flight exception;
begin
if f_id<1 then
raise inval_flight;
elsif valid_flight(f_id) then
update flight set departure=f_dep where flight_id=f_id;
end if;
exception
when inval_flight then
raise_application_error(-20225,'Flight Not Exist');
when others then
raise_application_error(-20226,'Something Went Wrong!');
end flight_departure_up;
end p_flight;
begin
p_flight.flight_cost_up(:f_id,:cost);
end;

begin
p_flight.flight_departure_up(21,'sylhet');
```

```
end;
```

3. (1 function & 1 procedure)

```
create or replace function check_fid(f_id flight.flight_id%type)
return boolean
as
cntx number;
begin
select count(flight_id) into cntx from flight where flight_id=f_id;
if cntx>0 then
return true;
else
return false;
end if;
end;

create or replace procedure delete_flight(f_id in flight.flight_id%type)
as
cnt boolean;
begin
cnt:=check_fid(f_id);
if cnt then
delete from booking where flight_id=f_id;
delete from flight where flight_id=f_id;
dbms_output.put_line('Flight Deleted');
else
dbms_output.put_line('Error in Deleting Flight');
end if;
end;

begin
delete_flight(2);
end;
```

✓ Trigger

1.

```
create or replace trigger flight_security before insert or update or delete on flight
begin
if to_char(sysdate,'HH24') not between '8' and '22' or to_char(sysdate,'DY') in ('FRI','SAT') then
raise_application_error(-20200,'Operation Failed');
end if;
end;
```

2.

```
create sequence seq_flight_dml increment by 1 start with 1 nocycle nocache;

create table flight_dml_log(fdml_id number(10) primary key,user_name varchar2(20), opt_name
varchar2(10), opt_date date);

create or replace trigger flight_log after insert or delete or update on flight
declare
opt varchar2(10);
begin
if inserting then
opt:='insert operation';
elsif updating then
opt:='update operation';
else
opt:='delete operation';
end if;
insert into flight_dml_log values(seq_flight_dml.NEXTVAL, user, opt,sysdate);
end;
```

3.

```
create table flight_price_uplog(flight_no number not null, old_flight_price number(10),
new_flight_price number(10), opt_date date);

create or replace trigger flight_price_uplog after update of flight_cost on flight
for each row
begin
insert into flight_price_uplog values(:old.flight_id,:old.flight_cost, :new.flight_cost, sysdate);
end;
```

4.

```
create or replace trigger ticket_trigger before insert or update of total_ticket on ticket
for each row
begin
if :new.total_ticket<1 or :new.total_ticket>9 then
raise_application_error(-20500,'No. Of ticket must be from 1 to 9');
end if;
end;
```