

KHULNA UNIVERSITY OF ENGINEERING & TECHNOLOGY

Course No: CSE 3110

Course Name: Database Systems Laboratory

Project Name:**Bus** **Travel Database Management System**

Submitted to:

**Jarin Firose Moon Md. Milon Islam**

Lecturer Lecturer

Department of Computer Science & Engineering Department of Computer Science & Engineering

Khulna University of Engineering & Technology Khulna University of Engineering & Technology

Submitted by:

**Md.Somir Khan**

Roll: 1507049

Section: A

Department of Computer Science & Engineering

**Project Overview**

**Brief Description:**

* The main objective of this project is to maintain an Bus Travel Agency's daily transactions, transportation and its Passengers information using Database Management System.
* Daily Transportation Tracking
* Daily Transaction Management
* Maintaining Bus Agency’s Different Counters
* Passenger's Information Management
* Managing Bus information's
* Most of the Functionalities of Bus Travel Management Systems are credited in this project

**Database Structure Description:**

**Bus Transportation Management system** is Commonly used in Bus travel Agencies. There are many Bus Agencies in Bangladesh.People travel everyday so the demand of transportation is high.Keeping track of transportation and tansactions Tavel Agencies needs a Database Management System. This project is based on this needs. It includes daily transport informations ,daily transactions, Passenger information's maintenance system and information handling of other branches.

There are five tables as follows:

* **Passenger,**
* **Bus\_details,**
* **Ticket,**
* **Route,**
* **Station.**

**Passenger table** consists of all the information about Passenger's. ID(ps\_id),Name(name), contact(contact\_no) and Address (ps\_address) are the attributes of Passenger table. **ps\_id** is the primary key of Passenger table.

**Bus\_details** describe Bus related informations.Availabilty of seats ,total number of seats and price are managed in this table. The attributes of the Bus\_details tables are Bus\_id, Bus\_agency, ticket\_price,total\_capacity, available\_seat. The **bus\_id** is the primary key of the Bus\_details table.

**Ticket table** contains all the information of transaction's. , PS\_ID, BUS\_ID , Quantity and Cost are the attributes of the Ticket table entity. ps\_ID and Bus\_id are the foreign key of Ticket Table the passengers transaction information are stored in Ticket table.It calculates the cost from ticket price and update Cost using trigger.

**Station table** contains the informations of bus counter(Bus station).ID,Name and Location are the attributes of station table.

**Route table** describes the Outgoing and Incoming Information Bus on that route.ID, Bus\_ ID, Source\_ID, Destination\_id and Route are the attributes of Route table.Bus\_ID is the foreign keys of Route table.

**Functionality:**

The database allows to complete the following functions:

* Passenger and Bus informations are stored in database.
* The Customers can buy ticket to various destinations.
* Total transactions are done from this table.
* The availability of seat in a particular bus are also known.
* Sell on a particular route can be checked.
* Number of passengers traveling in same route can be checked.

**Customers / Audience:**

The main customers are the Travellers who want to travel by bus from the different Bus station's of Country. The Travel Agencies tracks the activities which are completed by the bus transportation management system.

**Database Design Process:**

Our goal was to create a practical based database management design system. It was developed in Oracle. It was designed in Toad. Some planning, design, and review of the existing prototype were researched to build the project.

Five tables were designed to maintain the Bus Transportation system. Our table designprovides future flexibility for growth and changes to the database tables.

We learned several important lessons through the design process.

These include:

* Designing the tables is the most important step and must be done early in the project.
* Building a database from scratch is often easier than revising an

existing database.

* Initial design is so important.
* Being able to design a database well for a client requires a lot of understanding about the business process and needs behind the applications.

**Future of the Database:**

The database is currently functional as a practical application of Bus Transportation System. It is completed with Oracle 11g and Toad.

It is anticipated that the

following tasks will need to be accomplished in order to achieve the goals:

* Create the database tables in Oracle and Toad.
* Update code to incorporate feedback.
* Develop information for needed forms and database tables to support the form.

In future this database can be added in web application in the Bangladesh portal so that all the stations can get connected and insert, update the perfect data to increase efficiency and working speed in Bus transportation systems.

**Summary:**

This project was learning experience as this was a pretty small project as we have learned database very first time. Thus the project has helped to better understand user level application of databases how the different tasks and operation can be done in the database. Thus we have also about some programming which can be done to get certain output from the database.

|  |
| --- |
| Passenger |
| Ps\_id |
| Name |
| Ps\_address |
| Contact Number |

|  |
| --- |
| Bus\_Details |
| Bus\_id |
| Bus\_name |
| Ticket\_price |
| Total\_capacity |
| Available \_seat |

|  |
| --- |
| Ticket(transaction) |
| PS\_ID |
| Bus\_ID |
| quantity |
| Cost |

|  |
| --- |
| Station |
| ID |
| Name |
| Location |

|  |
| --- |
| Route |
| ID |
| Bus\_id |
| Source\_id |
| Destination\_id |
| Route |

**Code:**

Table:

drop table ticket;

drop table route;

drop table station;

drop table bus\_details;

drop table passenger;

create table passenger(

ps\_id number,

name varchar(40),

ps\_address varchar(40),

contact\_no number unique,

primary key (ps\_id)

);

create table bus\_details(

bus\_id number,

bus\_name varchar(40),

ticket\_price number,

total\_capacity number,

available\_seat number,

primary key(bus\_id)

);

create table station(

Id number,

name varchar(40),

location varchar(40),

primary key (id)

);

create table route(

Id number,

bus\_id number,

source\_id number,

destination\_id number,

*/\*time number(4,2),\*/*

route varchar(40),

primary key (id),

foreign key (source\_id) references station(id) on delete cascade,

foreign key (source\_id) references station(id) on delete cascade,

foreign key (bus\_id) references bus\_details(bus\_id) on delete cascade

);

create table ticket(

ps\_id number,

bus\_id number,

quantity number,

cost number,

foreign key (bus\_id) references bus\_details(bus\_id) on delete cascade,

foreign key (ps\_id) references passenger(ps\_id) on delete cascade

);

**Trigger:**

**1.UP\_INSERT:** This trigger updates the available\_seats attribute in bus\_details table when values are inserted in ticket table.

*--insert*

CREATE OR REPLACE TRIGGER up\_insert

AFTER INSERT ON ticket

FOR EACH ROW

BEGIN

UPDATE bus\_details

SET available\_seat = (:new.quantity-available\_seat)\*-1

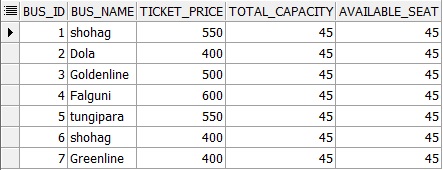
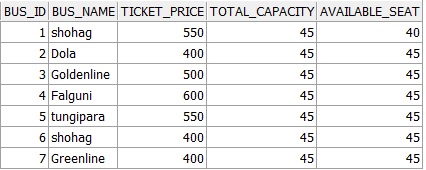
WHERE bus\_id = :NEW.bus\_id;

END;

**Query:**

****

**Result:**

** **

**Initial After Insert**

**2.UP\_UPDATE:** This trigger update the available\_seat attribute in bus\_details table when values are updated in ticket table.

*--update*

CREATE OR REPLACE TRIGGER up\_update

AFTER update ON ticket

FOR EACH ROW

declare

a bus\_details.available\_seat%type;

b bus\_details.available\_seat%type;

BEGIN

select available\_seat into b from bus\_details where bus\_id=:new.bus\_id;

a:=b+:old.quantity;

UPDATE bus\_details

SET available\_seat = (:new.quantity-a)\*-1

WHERE bus\_id = :NEW.bus\_id;

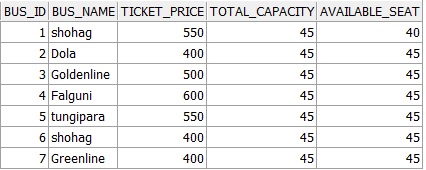
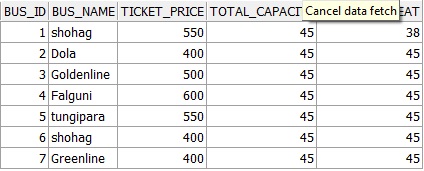
END;

**Query:**

****

**Result:**

**after insert if update query is pefromed the result will be like as follows:**

** **

**After Insert After Update**

**3.UP\_DELETE:** This trigger update the available\_seat attribute in bus\_details table when values are deleted in ticket table.

*--ticket delete*

CREATE OR REPLACE TRIGGER up\_delete

after delete ON ticket

FOR EACH ROW

Declare

a bus\_details.total\_capacity%type;

b number;

c BUS\_DETAILS.AVAILABLE\_SEAT%type;

BEGIN

select available\_seat into c from bus\_details where bus\_id=:old.bus\_id;

c:=c+:old.quantity;

UPDATE bus\_details

SET available\_seat = c

WHERE bus\_id = :old.bus\_id;

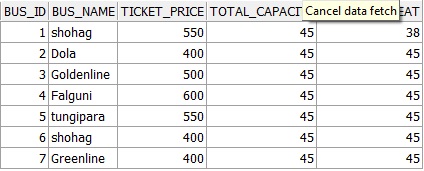
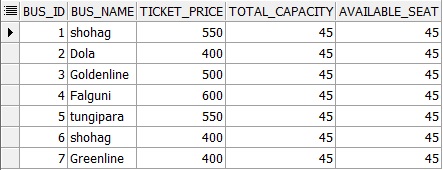
END;

Query:



Result:

after insert if **Delete** query is pefromed the result will be like as follows:

**** 

**After Update After Delete**

**4.UPDATE\_COST:**

This trigger updates the cost in ticket table according to the quantity in ticket table using ticket\_price from bus\_details table;

CREATE OR REPLACE TRIGGER update\_cost

before INSERT ON ticket

FOR EACH ROW

declare m bus\_details.ticket\_price%type;

BEGIN

select ticket\_price into m from bus\_details where bus\_id=:new.bus\_id;

:new.cost:= :new.quantity\*m;

END;

**Query:**

insert into ticket(ps\_id,bus\_id,quantity) values(1,1,6);

**Result:**

****

**After inserting values in to ticket table the cost is updated by trigger.**

**Data Insert:**

insert into station values(1 , 'gabtoli' ,'dhaka');

insert into station values(2 , 'zaflong' ,'sylhet');

insert into station values(3 , 'tigerpass' ,'chittagong');

insert into station values(4 , 'rangpur' ,'rangpur');

insert into station values(5 , 'fulbarigate' ,'khulna');

insert into station values(6 , 'dinajpur' ,'dinajpur');

insert into bus\_details values(1 , 'shohag',550,45,45);

insert into bus\_details values(2 , 'Dola',400,45,45);

insert into bus\_details values(3 , 'Goldenline',500,45,45);

insert into bus\_details values(4 , 'Falguni',600,45,45);

insert into bus\_details values(5 , 'tungipara',550,45,45);

insert into bus\_details values(6 , 'shohag',400,45,45);

insert into bus\_details values(7 , 'Greenline',400,45,45);

insert into route values(1,1,1,5,'dha-khu');

insert into route values(11,7,1,5,'dha-khu');

insert into route values(2,1,5,1,'khu-dha');

insert into route values(3,3,1,2,'dha-syl');

insert into route values(4,3,2,1,'syl-dha');

insert into route values(5,2,1,3,'dha-chi');

insert into route values(6,2,3,1,'chi-dha');

insert into route values(7,4,1,3,'dha-din');

insert into route values(8,4,3,1,'din-dha');

insert into route values(9,5,1,4,'dha-rang');

insert into route values(10,5,4,1,'rang-dha');

commit;

**File Read:**

create or replace directory MYDIR as 'C:\Users\Somir\Desktop';

grant read,write on directory MYDIR to public;

declare

f ***utl\_file.file\_type***;

line varchar(10000);

id passenger.ps\_id%type;

name passenger.name%type;

address PASSENGER.PS\_ADDRESS%type;

contact\_no PASSENGER.CONTACT\_NO%type;

begin

f:= ***utl\_file.fopen***('MYDIR','data.csv','R');

if ***utl\_file.is\_open***(f) then

***utl\_file.get\_line***(f,line,10000);

loop

begin

***utl\_file.get\_line***(f,line,10000);

if line is null then exit;

end if;

id:=regexp\_substr(line,'[^,]+',1,1);

name:=regexp\_substr(line,'[^,]+',1,2);

address:=regexp\_substr(line,'[^,]+',1,3);

contact\_no:=regexp\_substr(line,'[^,]+',1,4);

insert into passenger values(id,name,address,contact\_no);

commit;

exception

when no\_data\_found then exit;

end;

end loop;

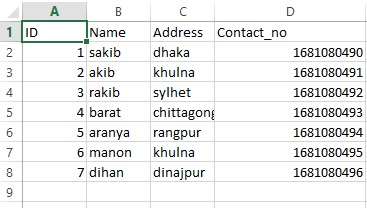
end if;

***utl\_file.fclose***(f);

end;

/

**File read Input excel file:**

****

**Function:**

**Total\_Sell:**

This function returns the value of total Sell’s in a specified route.

CREATE OR REPLACE function total\_sell(p\_route in route.route%type)

return number is

total number := 0;

c\_id route.bus\_id%type;

c\_num ticket.quantity%type;

cursor c\_busid is

select bus\_id from route where route=p\_route;

cursor c\_cost is

select cost from ticket where bus\_id=c\_id;

begin

open c\_busid;

loop

fetch c\_busid into c\_id;

EXIT WHEN c\_busid%notfound;

open c\_cost;

loop

fetch c\_cost into c\_num;

EXIT WHEN c\_cost%notfound;

total:=total+c\_num;

end loop;

close c\_cost;

end loop;

close c\_busid;

return total;

end;

**Function Call:**

DECLARE

c ticket.cost%type;

BEGIN

c := total\_sell('dha-khu');

***dbms\_output.put\_line***('Total Sell in this route =' || c);

END;

/

**Result:**

****

Procedure:

Number\_of\_passenger\_in\_bus:

This procedure shows the total number of Passengers in a specified route.The bus\_id from route table is used in bus\_details table to find the number of passenger in that route.

CREATE OR REPLACE PROCEDURE Number\_of\_passenger\_in\_bus (r in route.route%type)

AS

t BUS\_DETAILS.TOTAL\_CAPACITY%type;

a BUS\_DETAILS.AVAILABLE\_SEAT%type;

b bus\_details.bus\_id%type;

c number :=0;

cursor c\_busid is

select bus\_id from route where route=r;

BEGIN

open c\_busid;

loop

fetch c\_busid into b;

EXIT WHEN c\_busid%notfound;

select total\_capacity into t from bus\_details where bus\_id=b;

select available\_seat into a from bus\_details where bus\_id=b;

c:=c+t-a;

end loop;

close c\_busid;

***dbms\_output.put\_line***('number of passenger on this route'||'\_'||r ||'='|| c);

END;

/

**Procedure Call:**

set serveroutput on;

BEGIN

Number\_of\_passenger\_in\_bus ('dha-khu');

END;

/

**Result:**

****

**View:**

This view shows the bus\_id,ticket\_price and available seats of that bus where ticket price is greater than the average ticket price.

CREATE OR Replace VIEW Available\_Seats AS

SELECT Bus\_id, ticket\_price,available\_seat

FROM bus\_details

WHERE ticket\_price > (SELECT AVG(ticket\_price) FROM bus\_details);

**Query:**

**SELECT \* FROM Available\_Seats;**

**Result:**

****

Join: This join operation shows bus name,available seat and route name.

SELECT Bus\_details.bus\_name, BUS\_DETAILS.TICKET\_PRICE,BUS\_DETAILS.AVAILABLE\_SEAT,Route.route

FROM Bus\_details

INNER JOIN route ON bus\_details.bus\_id = route.id;

**Result:**

****

**File Write:**

declare

f ***utl\_file.file\_type***;

cursor c is select \* from ticket;

begin

f:= ***utl\_file.fopen***('MYDIR','write.csv','W');

***utl\_file.put***(f,'PS\_ID' || ',' || 'BUS\_ID' || ',' || 'Quantity' || ',' || 'Cost');

***utl\_file.new\_line***(f);

for c\_record in c

loop

***utl\_file.put***(f,c\_record.ps\_id || ',' || c\_record.BUS\_ID|| ',' || c\_record.quantity ||','|| c\_record.cost);

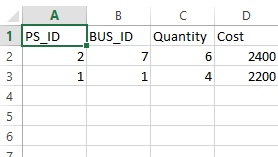
***utl\_file.new\_line***(f);

end loop;

***utl\_file.fclose***(f);

end;

***File Write Output File:***

******