Project Report

Databse Management System

Group_7

April 21, 2015

Introduction

We all know that embedded sql is the way to connect our database with the pgAdmin, and we can manually insert, delete and update in the database using the user interface.

Here is an embedded sql code for the insertion, deletion and update for three table of our database:
1. Food
2. Facility &
3. Room

We wanted to mention the attributes in our tables defined above.

1. Food - {Product_Id{PK}, rate, type, name}
2. Facility - {Facility_Id{PK},rate, no_of_hours, facility_type}
3. Room - {Room_No{PK}, rate, type, status_occupied }

Code:

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include "sqlca.h"

EXEC SQL INCLUDE sqlca;

EXEC SQL BEGIN DECLARE SECTION;

// Keeping Facility Record
typedef struct
{
varchar f_id[50];
int f_rate;
int f_time;
varchar f_type[50];
} facility_record;

// Keeping Uses Record

typedef struct
{
varchar fa_id[50];
varchar guest_id[50];
int quantity;
} uses_record;
```

```
// Keeping Food Record
typedef struct
{
varchar food_id[50];
int food_rate;
varchar food_type[50];
varchar food_name[50];
} food_record;


// Keeping Room Record
typedef struct
{
varchar room_id[50];
varchar room_type[50];
int room_rate;
varchar room_status[50];
} room_record;

EXEC SQL END DECLARE SECTION;

/////////////////////////////////////////////////////////////////////////////////////////


void prompt(char s[], char t[]) {
char c;
int i = 0;
printf("%s",s);
while ((c = getchar()) != '\n') {
t[i] = c;
i++;
}
t[i] = '\0';
}

// Insert Item on Food

void insert_food()
{

printf("\n Enter details:\n");

EXEC SQL BEGIN DECLARE SECTION;
food_record fo;
EXEC SQL END DECLARE SECTION;

printf("Enter Product ID\n");
scanf("%s",fo.food_id.arr);
fo.food_id.len = strlen(fo.food_id.arr);

printf("Enter the food rate\n");
scanf("%d",&fo.food_rate);
getchar();
```

```
printf("Type of the Food\n ");
scanf("%s",fo.food_type.arr);
fo.food_type.len = strlen(fo.food_type.arr);

printf("Enter the Name of the Food");
scanf("%s",fo.food_name.arr);
fo.food_name.len = strlen(fo.food_name.arr);

EXEC SQL set transaction read write;
EXEC SQL insert into food
    values(:fo.food_id,:fo.food_rate,:fo.food_type,:fo.food_name);

if (sqlca.sqlcode < 0) {
printf("\n\nPRODUCT (%s) already exists\n",fo.food_id.arr);
EXEC SQL rollback work;
EXEC SQL commit;
return;
}
else
{
printf("\n New Food Item Added Succesfully\n");
EXEC SQL commit;
return;
}
}

////////////////////////////////////////////////////////////////////////////////////////

//  Insert Item in Room

void insert_room()
{
printf("Enter the details of the new room\n");
EXEC SQL BEGIN DECLARE SECTION;
room_record ro;
EXEC SQL END DECLARE SECTION;

printf("\nEnter the Room_Number ");
scanf("%s",ro.room_id.arr);
ro.room_id.len=strlen(ro.room_id.arr);

printf("\nEnter the type of the room");
scanf("%s",ro.room_type.arr);
ro.room_type.len=strlen(ro.room_type.arr);

printf("\nEnter the rate of the room");
scanf("%d", &ro.room_rate);
getchar();

printf("\nEnter the Status");
scanf("%s",ro.room_status.arr);
ro.room_status.len=strlen(ro.room_status.arr);
```

```
EXEC SQL set transaction read write;
EXEC SQL insert into room
    values(:ro.room_id,:ro.room_type,:ro.room_rate,:ro.room_status);

if (sqlca.sqlcode < 0) {
printf("\n\nRoom (%s) is already added\n",ro.room_id.arr);
EXEC SQL rollback work;
return;
}
else
{
printf("\n New Room Added Succesfully\n");
EXEC SQL commit;
return;
}
}
////////////////////////////////////////////////////////////////////////////////////////////


void insert_facility()
{
printf("Enter the details of the new facility\n");
EXEC SQL BEGIN DECLARE SECTION;
facility_record fo;
EXEC SQL END DECLARE SECTION;

printf("\nEnter Facility ID ");
scanf("%s",fo.f_id.arr);
fo.f_id.len=strlen(fo.f_id.arr);

printf("\nEnter the rate of the facility");
scanf("%d", &fo.f_rate);
getchar();

printf("\nEnter the no_of_hours of the facility");
scanf("%d", &fo.f_time);
getchar();

printf("\nEnter the type of the facility");
scanf("%s",fo.f_type.arr);
fo.f_type.len=strlen(fo.f_type.arr);

EXEC SQL set transaction read write;
EXEC SQL insert into facility
    values(:fo.f_id,:fo.f_rate,:fo.f_time,:fo.f_type);

if (sqlca.sqlcode < 0) {
printf("\n\nFaclity (%s) is already added\n",fo.f_id.arr);
EXEC SQL rollback work;
return;
}
else
{
```

```
printf("\n New Faclity Added Succesfully\n");
EXEC SQL commit;
return;
}
}
////////////////////////////////////////////////////////////////////////////////////////////////

void remove_food()
{
EXEC SQL begin declare section;
food_record frec;
varchar food_n[50];
int onum;
varchar pid[20];
varchar pname[20];
varchar ptype[20];
int prate;
EXEC SQL end declare section;

printf("Enter Product_ID to be deleted: ");
scanf("%s",food_n.arr);
food_n.len = strlen(food_n.arr);


EXEC SQL select *
into :frec
from food
where product_id = :food_n;

if (sqlca.sqlcode > 0) {
printf("Product ID does not exist\n");
return;
}

EXEC SQL declare del_cur cursor for
select product_id from food where product_id = :food_n;

EXEC SQL set transaction read only;
EXEC SQL open del_cur;
EXEC SQL fetch del_cur into :onum;
if (sqlca.sqlcode == 0) {
printf("food_item is in ordered so delete order first\n");
EXEC SQL commit;
return;
}

EXEC SQL commit;
EXEC SQL set transaction read write;
EXEC SQL delete from food where product_id = :food_n;
EXEC SQL DECLARE deleteS CURSOR FOR
           SELECT * FROM food WHERE product_id = :food_n;

   EXEC SQL OPEN deleteS;
```

```
   do {
      EXEC SQL FETCH deleteS INTO :pid, :prate, :ptype, :pname;
      if (sqlca.sqlcode != 0)
       break;
      printf( "\nDeleting the product...\n");
      EXEC SQL DELETE FROM food WHERE CURRENT OF deleteS;
      } while ( 1 );

   EXEC SQL CLOSE deleteS;
printf("\nFood DELETED\n");
EXEC SQL commit;
}

////////////////////////////////////////////////////////////////////////////////////////////

void remove_facility()
{
EXEC SQL begin declare section;
facility_record farec;
uses_record urec;
varchar faci_n[50];
int onum;
varchar fid[20];
varchar ftype[20];
int ftime[20];
int frate;
EXEC SQL end declare section;

printf("Enter Facility_ID to be deleted: ");
scanf("%s",faci_n.arr);
faci_n.len = strlen(faci_n.arr);


EXEC SQL select *
into :farec
from facility
where facility_id = :faci_n;

if (sqlca.sqlcode > 0) {
printf("Facility ID does not exist\n");
   return;

}


EXEC SQL declare del_cur1 cursor for
select facility_id from facility where facility_id = :faci_n;

EXEC SQL set transaction read only;
EXEC SQL open del_cur1;
EXEC SQL fetch del_cur1 into :onum;
if (sqlca.sqlcode == 0) {
printf("facility_id is in use so delete order first\n");
EXEC SQL commit;
```

```
return;
}

EXEC SQL commit;
EXEC SQL set transaction read write;
EXEC SQL delete from facility where facility_id = :faci_n;
EXEC SQL DECLARE deleteP CURSOR FOR
          SELECT * FROM facility WHERE facility_id = :faci_n;

   EXEC SQL OPEN deleteP;

   do {
      EXEC SQL FETCH deleteP INTO :fid, :frate, :ftime, :ftype;
      if (sqlca.sqlcode != 0) break;
      printf( "\nDeleting the facility id...\n");
      EXEC SQL DELETE FROM facility WHERE CURRENT OF deleteS;
      } while ( 1 );

   EXEC SQL CLOSE deleteP;
printf("\nFacility DELETED\n");
EXEC SQL commit;
}
////////////////////////////////////////////////////////////////////////////////////////////

void remove_room()
{
EXEC SQL begin declare section;
room_record rrec;
varchar room_n[50];
int onum;
varchar rid[20];
varchar rtype[20];
varchar rstatus[20];
int rrate;
EXEC SQL end declare section;

printf("Enter Room_Number to be deleted: ");
scanf("%s",room_n.arr);
room_n.len = strlen(room_n.arr);


EXEC SQL select *
into :rrec
from room
where room_no = :room_n;

if (sqlca.sqlcode > 0) {
printf("Room does not exist\n");
return;
}


EXEC SQL declare del_cur2 cursor for
select room_no from room where room_no = :room_n;
```

```
EXEC SQL set transaction read only;
EXEC SQL open del_cur2;
EXEC SQL fetch del_cur2 into :onum;
if (sqlca.sqlcode == 0) {
printf("room_no is alloted first delete the allot\n");
EXEC SQL commit;
return;
}

EXEC SQL commit;
EXEC SQL set transaction read write;
EXEC SQL delete from room where room_no = :room_n;
EXEC SQL DECLARE deleteQ CURSOR FOR
          SELECT * FROM room facility WHERE room_no= :room_n;

   EXEC SQL OPEN deleteQ;

   do {
      EXEC SQL FETCH deleteQ INTO :rid, :rtype, :rrate, :rstatus;
      if (sqlca.sqlcode != 0) break;
      printf( "\nDeleting the given Room...\n");
      EXEC SQL DELETE FROM room WHERE CURRENT OF deleteS;
      } while ( 1 );

   EXEC SQL CLOSE deleteQ;
printf("\nRoom DELETED\n");
EXEC SQL commit;
}

////////////////////////////////////////////////////////////////////////////////////////////


void update_food() {

EXEC SQL SET SEARCH_PATH TO hotel;
EXEC SQL begin declare section;
food_record food1;
varchar fo_id[40];
varchar foo_id[40];
varchar n[30];
varchar t[30];
int r;
int temp;
EXEC SQL end declare section;

printf("Enter the product_id you want to update\n ");
scanf("%s",fo_id.arr);
fo_id.len = strlen(fo_id.arr);
EXEC SQL select *
into
:food1
from
food
```

```
where product_id = :fo_id;
if (sqlca.sqlcode > 0) {
printf("food_item(%s) does not exist\n",fo_id.arr);
return;
}

printf("Current rate: %d\n",food1.food_rate);
printf("Enter the rate :\n ");
scanf("%d", &r);
food1.food_rate = r;

printf("Current Tpye%s\n",food1.food_type.arr);
printf("New Type of the food : \n");
scanf("%s",t.arr);
if (strlen(t.arr) > 1) {
strcpy(food1.food_type.arr,t.arr);
food1.food_type.len = strlen(food1.food_type.arr);
}

printf("Current Name%s\n",food1.food_name.arr);
printf("New Name of the food :\n ");
scanf("%s", n.arr);
if (strlen(n.arr) > 1) {
strcpy(food1.food_name.arr,n.arr);
food1.food_name.len = strlen(food1.food_name.arr);
}



EXEC SQL set transaction read write;
EXEC SQL UPDATE food
set product_id = :food1.food_id,
rate = :food1.food_rate,
type = :food1.food_type,
name = :food1.food_name
where product_id = :food1.food_id;
if (sqlca.sqlcode < 0) {
printf("\n\nError on Update\n");
EXEC SQL rollback work;
return;
}
EXEC SQL commit;
printf("\nFood Item (%s) updated.\n",food1.food_id.arr);
}

////////////////////////////////////////////////////////////////////////////////////////////////

void update_room() {

EXEC SQL SET SEARCH_PATH TO hotel;
EXEC SQL begin declare section;
room_record room1;
varchar ro_id[40];
varchar roo_id[40];
```

```
varchar rs[30];
varchar t[30];
int r;
EXEC SQL end declare section;

printf("Enter the room_id you want to update \n");
scanf("%s",ro_id.arr);
ro_id.len = strlen(ro_id.arr);
EXEC SQL select *
into
:room1
from
room
where room_no = :ro_id;
if (sqlca.sqlcode > 0) {
printf("room(%s) does not exist\n",ro_id.arr);
return;
}

printf("Current rate: %d\n",room1.room_rate);
printf("Enter the rate:\n ");
scanf("%d", &r);
room1.room_rate = r;

printf("Current Tpye%s\n",room1.room_type.arr);
printf("New Type of the room:\n");
scanf("%s",t.arr);
if (strlen(t.arr) > 1) {
strcpy(room1.room_type.arr,t.arr);
room1.room_type.len = strlen(room1.room_type.arr);
}

printf("Current Status%s\n",room1.room_status.arr);
printf("New Status of the room: \n");
scanf("%s", rs.arr);
if (strlen(rs.arr) > 1) {
strcpy(room1.room_status.arr,rs.arr);
room1.room_status.len = strlen(room1.room_status.arr);
}



EXEC SQL set transaction read write;
EXEC SQL UPDATE room
set room_no = :room1.room_id,
type = :room1.room_type,
rate = :room1.room_rate,
status_occupied = :room1.room_status
where room_no = :room1.room_id;
if (sqlca.sqlcode < 0) {
printf("\n\nError on Update\n");
EXEC SQL rollback work;
return;
}
```

```
EXEC SQL commit;
printf("\nRoom is (%s) updated.\n",room1.room_id.arr);
}

////////////////////////////////////////////////////////////////////////////////////////////

void update_facility() {

EXEC SQL SET SEARCH_PATH TO hotel;
EXEC SQL begin declare section;
facility_record fac1;
varchar fa_id[40];
varchar faa_id[40];
varchar ft[30];
int fh;
int r;
EXEC SQL end declare section;

printf("Enter the facility_id you want to update \n");
scanf("%s",fa_id.arr);
fa_id.len = strlen(fa_id.arr);
EXEC SQL select *
into
:fac1
from
facility
where facility_id = :fa_id;
if (sqlca.sqlcode > 0) {
printf("facility(%s) does not exist\n",fa_id.arr);
return;
}

printf("Current rate: of the facility %d\n",fac1.f_rate);
printf("Enter the rate:\n ");
scanf("%d", &r);
fac1.f_rate = r;

printf("Current Time for the facility%d\n",fac1.f_time);
printf("New Time for the facility\n");
scanf("%d",&fh);
fac1.f_time = fh;


printf("Current Type%s\n",fac1.f_type.arr);
printf("New Type of the facility: \n");
scanf("%s", ft.arr);
if (strlen(ft.arr) > 1) {
strcpy(fac1.f_type.arr,ft.arr);
fac1.f_type.len = strlen(fac1.f_type.arr);
}


EXEC SQL set transaction read write;
EXEC SQL UPDATE facility
```

```
set facility_id = :fac1.f_id,
rate = :fac1.f_rate,
no_of_hours = :fac1.f_time,
facility_type = :fac1.f_type
where facility_id = :fac1.f_id;
if (sqlca.sqlcode < 0) {
printf("\n\nError on Update\n");
EXEC SQL rollback work;
return;
}
EXEC SQL commit;
printf("\nfacility is (%s) updated.\n",fac1.f_id.arr);
}



////////////////////////////////////////////////////////////////////////////////////////////////////

void main_menu()
{
printf("\n\n");
printf("\t\t\t\t\t\tWelcome to GUI For Our Project\n\n");
printf("\t\t\t* *   * ***** *    **** ****  *     * *****   *** ****   *
    * **** *** **** * \n");
printf("\t\t\t * * * * ***  *    *    * *   * * * * ***      *   * *
    **** * *  *  *** *   \n");
printf("\t\t\t *  *   ***** **** **** ****  *   * *  *****    *   ****   *
    * **** *  **** *** \n\n");


printf("\t\t\t\t\t--------------------Hotel Management
    System-----------------\n\n");
printf("\n\t\t\t\t\t\tPlease select an option:\n");
printf("\t\t\t\t\t1. Insert a new Entry\n");
printf("\t\t\t\t\t2. Update an existing entry\n");
printf("\t\t\t\t\t3. Remove an existing Entry\n");
printf("\t\t\t\t\t4. Exit\n\n");
printf("\t\t\t\t\t*********************************************************\n");
}

////////////////////////////////////////////////////////////////////////////////////////////////////

void main_insert()
{
int insert_choice;
printf("Enter the type of a new Entry\n");
printf("1. Food\n");
printf("2. Room\n");
printf("3. Facility\n");
printf("4. Exit\n");
printf("Enter your choice: ");
scanf("%d",&insert_choice);
switch(insert_choice)
{
```

```
case 1:
insert_food();
break;
case 2:
insert_room();
break;
case 3:
insert_facility();
break;
case 4:
EXEC SQL COMMIT;
EXEC SQL DISCONNECT;
exit(0);
}
}

////////////////////////////////////////////////////////////////////////////////////////////////////


void main_remove()
{
int remove_choice;
printf("Enter the type of entry you want to delete\n");
printf("1. Food\n");
printf("2. Facility\n");
printf("3. Room\n");
printf("4. Exit\n");
printf("Enter your choice: ");
scanf("%d",&remove_choice);
switch(remove_choice)
{
case 1:
remove_food();
break;
case 2:
remove_facility();
break;
case 3:
remove_room();
break;
case 4:
EXEC SQL COMMIT;
EXEC SQL DISCONNECT;
exit(0);
}
}


////////////////////////////////////////////////////////////////////////////////////////////////////

void main_update()
{
int remove_choice;
printf("Enter the type of entry you want to update\n");
```

```
printf("1. Food\n");
printf("2. Facility\n");
printf("3. Room\n");
printf("4. Exit\n");
printf("Enter your choice: ");
scanf("%d",&remove_choice);
switch(remove_choice)
{
case 1:
update_food();
break;
case 2:
update_facility();
break;
case 3:
update_room();
break;
case 4:
EXEC SQL COMMIT;
EXEC SQL DISCONNECT;
exit(0);
}
}

////////////////////////////////////////////////////////////////////////////////////////////

int main()
{
EXEC SQL CONNECT TO try_project@localhost:5432 USER postgres;
EXEC SQL set search_path to hotel;;
int main_choice;
X:
main_menu();
printf("\nEnter your choice (eg. 1 for 1st choice): ");
scanf("%d",&main_choice);
switch(main_choice)
{
case 1:
main_insert();
goto X;
break;
case 2:
main_update();
goto X;
break;
case 3:
main_remove();
goto X;
break;
case 4:
EXEC SQL COMMIT;
EXEC SQL DISCONNECT;
exit(0);
default:
```

```
printf("Re-enter your choice");
goto X;
}
EXEC SQL DISCONNECT try_project;
return 0;
}
```

Output

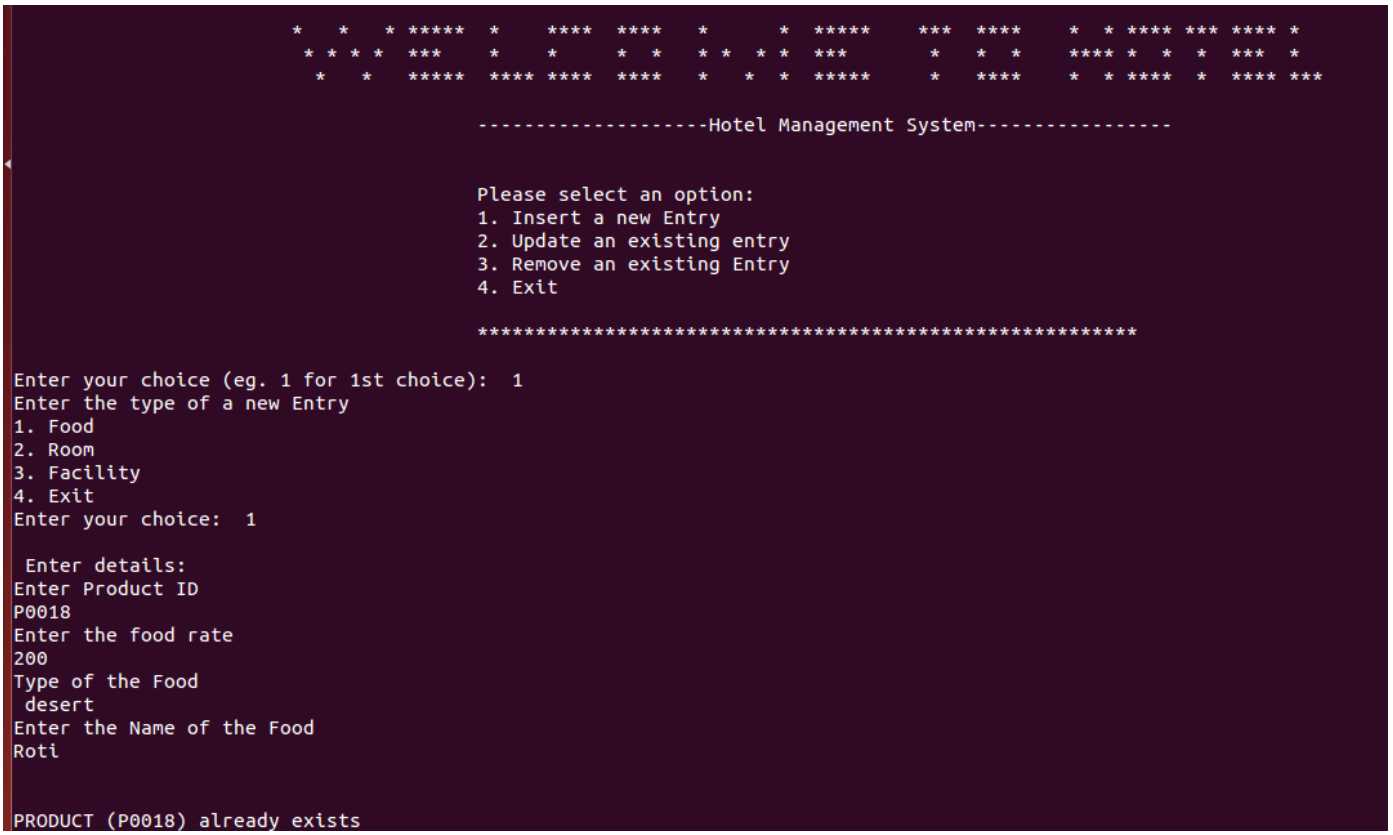

Figure 1: **Sample output of Embedded sql Query**