**IS 420**

**Group Project**

*Team Members:*

*Derek Luong*

*Ian Twombly*

*Josh Flitter*

*Thomas Zhang*

*Saccit Gupta*

*Zhimin Zhou*

**Team Charter**

**1. Goals and objectives of the team**

Use knowledge gained from class to create a management system that organizes customer booking information. The customer will be able to reserve and cancel reservations for rooms and halls in specific hotels for events. Payment calculations need to be made for the type of hall reserved by the customer along with the number of people attending the event along with additional services. There are a total of 21 tasks that must be completed and will be split up between the group members.

**2.How each member will equally communicate their views and ideas**

Group members will communicate their ideas and views through GroupMe, email or at the meetings.

**3.When and where to meet on a regular basis along with contact information for each member**

The team will meet both in person before class and electronically via the GroupMe app. This will prevent any conflicts of schedule that may occur from school and work responsibilities.

**Team members information:**

Derek Luong

Email:[dluong2@umbc.edu](mailto:dluong2@umbc.edu)

Ian Twombly

Email: [twombly1@umbc.edu](mailto:twombly1@umbc.edu)

Josh Flitter

Email:[flitter1@umbc.edu](mailto:flitter1@umbc.edu)

Thomas Zhang

Email: [thomasz1@umbc.edu](mailto:thomasz1@umbc.edu)

Saccit Gupta

Email: [gupta2@umbc.edu](mailto:gupta2@umbc.edu)

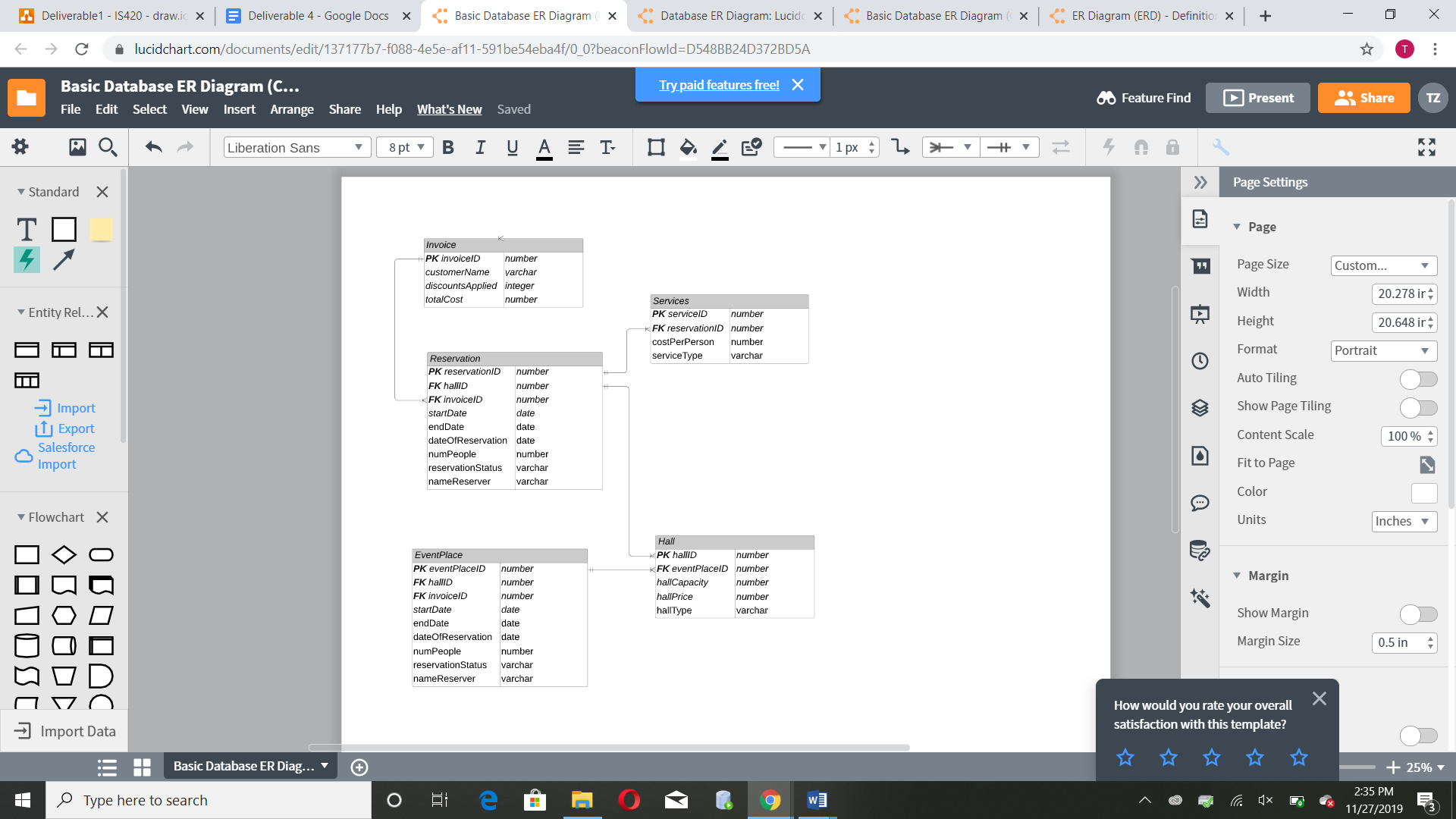
Zhimin Zhou

Email: [zzhou1@umbc.edu](mailto:zzhou1@umbc.edu)

**4.“kick-out:” what will happen if members do not show up for the meeting, or do not do their work, how to resolve conflicts should they arise**

* 1. All conflicts will be decided on using a group vote. The majority decision will be the group’s decision and this is what will allow us to move forward with a project decision
  2. Should any member(s) disagree with the process or outcome the member(s) will have the option to communicate with the professor or TA for a new conflict resolution protocol.
  3. Should any conflicts arise with a team member they are expected to notify the other team members so that the portion of work or tasks gets accomplished.

**Updated ER Diagram**

****

**Procedures**

**1. Add a new event place: Create a new event place with appropriate information about it as input parameters. Minimum parameters are Address, Phone, Building Type, etc. Building Types are: hotel, college, university. Have in mind that you may need more parameters when you call this procedure.**

Create the tables and store the data:

Create table event\_place

Value

(sequence\_id,

Name varchar2(20),

Address varchar2(40),,

City varchar2(20),

State varchar2(2),

Zip\_code number,

Phone varchar2(20),,

Type varchar2(1),,

Event\_id number,

Hall\_id number);

Create sequence event\_place

Start with 1

increment by 1;

Insert into event\_place

Value

(sequence\_id, address,city,state,zip\_code,phone,type,event\_id, hall\_id);

**2. Find an event place: Provide as input the address of the event place and return its event ID**

Store different place data, so client don’t need to input too much:

Output: a event\_place, b event\_place, c event\_place

Input: a, b, c

Output: event\_id

**3. Event place info: Given an event place ID, display all information about that event place, including its building type.**

Show information:

Select \* from event\_place where event\_id =’&ev\_id’;

* 1. **Add Event Hall: Given an event place ID, add a new hall for events to be held in that event place. The hall types are: Small Hall, Medium hall, Large hall.**

Show information:

Select hall\_type from hall where hall.hall\_id = event\_place.hall\_id and event\_id =’&ev\_id’;

* 1. **Report Event Places and Event Halls In State: Given a state, display event hall information of all event places in that particular state. Include total capacity per event hall type per event place. Make sure to display the building type in this report.**

Show information after input:

Select \* from event\_place where state=’&ev\_state’;

**6. Make an event reservation: Input parameters: Event place ID, requestor’s name, start date, end date, event type, date of reservation, number of people attending, etc. Output: event reservation ID. NOTE: Only one person (requestor) can make an event reservation. However, the same person can make multiple reservations. Event types: Conference, Workshop, Hackathon. Also, make sure that the reserved hall has a capacity that can hold the number of people attending. For example, for a conference of 500 people, a customer must reserve 2 medium halls and a large hall for each day of the conference, usually 3 consecutive days.**

Create table reservation(

EventplaceId integer

requestorName varchar2(100)

numberPeople integer

startDate date

endDate date

eventType varchar2(10)

)

Insert into reservation

values( --------)

**7. Find an event reservation: Input is a requestor’s name, event date, and event place ID. Output is event reservation ID**

Create or replace function findReservation(requestName in varchar2, eventDate in date, placeNum in integer) return integer

* Parameter that takes in requester’s name and finds it in the reservation list
* Parameter that takes in a date to find the event date requested
* Parameter that takes in an integer that is used to find a matching eventplace id

Finds event reservation using name, date and event place id. Returns 1 if found and 0 if not found

**8. Cancel an event: Input the event reservation ID and mark the reservation as cancelled (do NOT delete it)**

Create or replace procedure cancelReservation(reserveID integer) - Takes in parameter to look for matching eventReservation ID

Finds reservation id that matches the id passed through parameter and cancels it.

**9. ShowCancelations: Print all cancelled events in the event management system. Show event reservation ID, event place name, building type, location, event type, room type, dates.**

Create or replace procedure showCancelations

(

event\_ID reservation.eventReservationId%Type

placeN reservation.placeName%Type

buildingT reservation.buildingType%Type

Loca reservation.address%Type

roomT reservation.roomType%Type

eventD reservation.eventDate%Type

)

·

Print all cancelled reservations found using cursor/loop/procedure

**10. change\_eventDate: Input the event reservation ID, and change event start and end date. Check if there is availability in the same or larger room type for the new date interval**

change\_eventDate--Takes as inputs reservation\_id, start\_date, and end\_date, and trys to change the startdate and enddate of the reservation corresponding with that id to new\_start\_date and new\_end\_date. Throws an exception if there is a reservation overlap.

Create or replace procedure change\_eventDate(reservation\_id IN number, new\_start\_date IN date, new\_end\_date IN date)

**11. change\_eventHall: Input the reservation ID and change the Hall type if there is availability for the new hall during the reservation’s date interval. For example, a Hackathon was scheduled in a large hall, it now needs to move to a medium hall (if the participants can fit in the medium hall), and vice-versa, an event could be moved from a small hall to a medium or large hall**

change\_eventHall--Takes reservation\_id and hall\_size as inputs, and tries to change the hall\_id associated with the eventplaceid to one where needed\_hall\_capacity is equal to hall\_size

Create or replace procedure change\_eventHall(reservation\_id IN number, needed\_hall\_capacity IN number)

**12. eventTypeList: Given an event type (Hackathon, Conference, etc.) display all events of that type in all event places along with the address of the place and the date of the event.**

eventTypeList--Takes event\_category and uses it to select all events where event\_category = eventType in a select command

Create or replace procedure eventTypeList(event\_category IN varchar(20))

**13. Show events by person: Given a person’s name, find all events under that name (the person is the requestor of the event)**

Create or replace procedure All\_Events(requestorsName in char -- input param nameReserver to identify which persons’ events to be displayed)

Procedure: All\_Events - this procedure takes in person name as a parameter and displays all events that are reserved under that name.

**14. Total Monthly Income Report: Calculate and display income from all sources of all event places. Totals must be printed by month, and for each month by event and service type. Include discounts. At the end of this report print the overall total.**

Create or replace procedure Total\_Income is

Procedure: Total\_Income -- procedure with no parameters that print out the total profits from each event that took place each month by event and service type including discounts and then print the overall total at the end of the table as well.

·

**15. Add a service to an event: Input: Event reservationID, and a specific service. Add the service to the event for a particular date. Multiple services are allowed on a reservation for the same date. For meals make sure to multiply the amount by the number of people attending the event.**

Create or replace procedure addService(resersationID in number -- input param reservation ID to identity an event to add service to, service\_type in varchar(20)-- input service name that will be added using procedure)

Procedure: addService - this procedure will add the service to the event that correspond with the input reservation id.

**Ian Twombly**

**16. Reservation Services Report: Input the event reservation ID and display all services on this reservation. Also print the number of attendees of the event. Print “no services for this reservation” if none exists.**

Create or replace procedure reservationService(reservation\_id in number -- input param reservation id to identify the service that corresponds with the reservation id.)

Procedure: reservationService - this procedure will display all service related to this reservation input, it will also print number of attendees.

**17. Show Specific Service Report: Input the service name, and display information on all reservations that have this service in all event places**

Create or replace procedure showSpecificService(service\_type in varchar(20) -- input param service\_type to identify the name of service to look for in reservations )  
Procedure showSpecificService - this procedure will display all reservations that have the input service.

**18. Services Income Report: Given an event place ID, calculate and display income from all services in all reservations in that event place.**

Create or replace procedure displayIncome(eventPlace\_id in number input param eventPlace id to look for service along with cost in the eventPlace)

Procedure displayIncome- this procedure will calculate and display income from all services in the input event place.

·

**19. Show available halls by type: Given an event place ID, display the count of all available halls by hall type.**

Create or replace procedure showHalls(eventPlace\_id in number - input parameter eventPlace\_id to direct what place should be displayed)

showHalls - generates a visible table for the user, organized by hall type (small, medium, large) based on the input of the ID for the event location

**20. Event Invoice: Input: Event reservationID. Output:**

* + - Name of person that reserved the event
    - Event place name, type, and address
    - Event hall number(s), rate per day and possibly multiple halls (if someone reserved several rooms)
    - Services rendered per date, type, and amount
    - Discounts applied (if any)
    - Total amount to be paid

Create or replace procedure invoiceInfo(reservation\_id in number - input parameter reservation\_id to identify which reservation is to be accessed and shown)

invoiceInfo - displays all of the information necessary for the user to have a basic understanding of the event through the input of the event reservation ID

**21. Income by State Report: Input is a specific state. Print total income from all events as follows: Each output line should contain information of a specific event ID (income from halls, income from services, total income of this event). At the end of the report, print the total income of all events and services. Include discounts.**

Create or replace procedure incomeByState(event\_state in char(2) - input parameter event\_state to specify the state that is categorizing total income)

incomeByState - prints total income for the company from a particular input state, and details the specificities of where the income came from and what calculations were made upon the total

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Event place table :**

Create table event\_place

(ep\_id number,

ep\_name varchar2(20),

Address varchar2(40),

City varchar2(20),

ep\_State varchar2(2),

Zip\_code number,

Phone varchar2(20),

ep\_type varchar2(3),

Event\_id number,

Hall\_id number,

CONSTRAINT ep\_id\_pk PRIMARY KEY (ep\_id));

create sequence seq\_event\_place

minvalue 1

MAXVALUE 999999

start with 2

increment by 1;

**Hall table:**

create table hall

(hall\_id NUMBER,

hall\_type VARCHAR2(20),

hall\_capacity NUMBER,

price NUMBER,

CONSTRAINT hall\_id\_pk PRIMARY KEY(hall\_id)

);

**Invoice table:**

create table Invoice(

invoiceID number,

customerName varchar(20),

discountsApplied number,

totalCost number,

constraint invoice\_pk primary key(invoiceID)

);

**Services table:**

create table Services(

serviceID number,

eventPlaceID number,

costPerPerson number,

serviceType varchar(20),

constraint service\_pk primary key(serviceID)

constraint service\_fk foreign key(eventPlaceID) references eventPlace(eventPlaceID)

);

**Reservation Table:**

create table reservation(

reservationID integer,

eventPlaceID integer, //Fk

invoiceID integer, //FK

startDate date,

endDate date,

dateOfreservation date,

numOfPeople integer,

reservationStatus varchar2(15),

Constraint reservationID\_pk Primary Key (reservationID)

);

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**UPDATED TABLE SCRIPT**

alter table Hall drop constraint fk\_eventPlaceID\_hall;

alter table Services drop constraint fk\_eventPlaceID\_services;

alter table Reservation drop constraint fk\_invoiceID\_reservation;

alter table Reservation drop constraint fk\_hallID\_reservation;

drop table hall;

drop table services;

drop table invoice;

drop table reservation;

drop table eventplace;

Create table Invoice(

invoiceID number,

customerName varchar(20),

discountsApplied integer,

totalCost integer,

constraint pk\_invoiceID Primary Key (invoiceID)

);

Create table EventPlace(

eventPlaceID number,

address varchar(20),

city varchar(20),

state\_name varchar(2),

zipCode integer,

buildingType varchar(20),

phoneNumber varchar(20),

constraint pk\_eventPlaceID primary key (eventPlaceID)

);

Create table Reservation(

reservationID number,

hallID number,

invoiceID number,

startDate date,

endDate date,

eventType varchar(20),

dateOfReservation date,

numPeople number,

reservationStatus varchar(20),

nameReserver varchar(20),

constraint pk\_reservationID Primary Key (reservationID)

);

Create table Hall(

hallID number,

eventPlaceID number,

hallCapacity number,

hallPrice number,

hallType varchar(20),

constraint pk\_hallID primary key (hallID)

);

Create table Services(

serviceID number,

eventPlaceID number,

costPerperson number,

serviceType varchar(20),

constraint pk\_serviceID Primary Key (serviceID)

);

alter table Hall

add constraint fk\_eventPlaceID\_hall foreign key (eventPlaceID) references EventPlace(eventPlaceID);

alter table Services

add constraint fk\_eventPlaceID\_services foreign key (eventPlaceID) references EventPlace(eventPlaceID);

alter table Reservation

add constraint fk\_hallID\_reservation foreign key (hallID) references hall(hallID);

alter table Reservation

add constraint fk\_invoiceID\_reservation foreign key (invoiceID) references Invoice(invoiceID);

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**BASIC TESTING ENTRIES**

insert into invoice values(1,'John',0,500);

insert into invoice values(2,'Joe',0,200);

insert into invoice values(3,'Bob',0,300);

insert into invoice values(4,'Kim',0,500);

insert into invoice values(5,'Vancy',0,100);

insert into invoice values(6,'Larry',0,500);

insert into eventPlace values(1,'123 Test Rd','Testville','MD',20835,'TES','123-456-7890');

insert into eventPlace values(2,'124 Test Rd','Testville','MD',20835,'TES','123-456-1234');

insert into eventPlace values(3,'125 Test Rd','Testville','MD',20835,'TES','123-456-4567');

insert into hall values(1,1,10,500,'Small');

insert into hall values(2,1,200,5000,'Medium');

insert into hall values(3,1,500,50000,'Large');

insert into reservation values(1,1,1,date '2005-12-2',date '2005-12-5','Type',date '2005-12-1',6,'Cancelled','John');

insert into reservation values(2,1,2,date '2006-12-2',date '2006-12-5','Type',date '2006-12-1',53,'Active','Joe');

insert into reservation values(3,2,3,date '2007-12-2',date '2007-12-5','Type',date '2007-12-1',27,'Active','Bob');

insert into reservation values(4,3,4,date '2008-12-2',date '2008-12-5','Type',date '2008-12-1',35,'Cancelled','Kim');

insert into services values(1,1,50,'Massage');

insert into services values(2,1,500,'Spa');

insert into services values(3,1,5000,'MicroTransactions');

**NEW CODE**

Create table Invoice(

invoiceID varchar(20),

customerName varchar(20),

discountsApplied integer,

totalCost integer,

Primary Key (invoiceID)

);

Create table EventPlace(

eventPlaceID varchar(20),

reservationID varchar(20),

address varchar(20),

buildingType char,

phoneNumber number,

Primary Key (eventPlaceID)

);

Create table Reservation(

reservationID varchar(20),

eventPlaceID varchar(20),

invoiceID varchar(20),

startDate date,

eventType date,

dateOfReservation date,

numPeople number,

reservationStatus boolean,

nameReserver varchar(20),

Primary Key (reservationID)

);

Create table Hall(

hallID varchar(20) primary key,

eventPlaceID varchar(20),

hallCapacity number,

hallPrice integer,

hallType char

);

Create table Services(

serviceID varchar(20),

eventPlaceID varchar(20),

costPerperson number,

serviceType char,

Primary Key (serviceID)

);

alter table Hall drop constraint fk\_eventPlaceID\_hall;

alter table Services drop constraint fk\_eventPlaceID\_services;

alter table EventPlace drop constraint fk\_reservationID\_eventPlace;

alter table Reservation drop constraint fk\_invoiceID\_reservation;

alter table Reservation drop constraint fk\_eventPlaceID\_reservation;

drop table hall;

drop table services;

drop table invoice;

drop table reservation;

drop table eventplace;

Create table Invoice(

invoiceID number,

customerName varchar(20),

discountsApplied integer,

totalCost integer,

constraint pk\_invoiceID Primary Key (invoiceID)

);

Create table EventPlace(

eventPlaceID number,

reservationID number,

address varchar(20),

city char,

state\_name char,

zipCode number,

buildingType char,

phoneNumber number,

constraint pk\_eventPlaceID primary key (eventPlaceID)

);

Create table Reservation(

reservationID number,

eventPlaceID number,

invoiceID number,

startDate date,

eventType varchar(20),

dateOfReservation date,

numPeople number,

reservationStatus varchar(20),

nameReserver varchar(20),

constraint pk\_reservationID Primary Key (reservationID)

);

Create table Hall(

hallID number,

eventPlaceID number,

hallCapacity number,

hallPrice number,

hallType char,

constraint pk\_hallID primary key (hallID)

);

Create table Services(

serviceID number,

eventPlaceID number,

costPerperson number,

serviceType char,

constraint pk\_serviceID Primary Key (serviceID)

);

alter table Hall

add constraint fk\_eventPlaceID\_hall foreign key (eventPlaceID) references EventPlace(eventPlaceID);

alter table Services

add constraint fk\_eventPlaceID\_services foreign key (eventPlaceID) references EventPlace(eventPlaceID);

alter table EventPlace

add constraint fk\_reservationID\_eventPlace foreign key (reservationID) references Reservation(reservationID);

alter table Reservation

add constraint fk\_eventPlaceID\_reservation foreign key (eventPlaceID) references EventPlace(eventPlaceID);

alter table Reservation

add constraint fk\_invoiceID\_reservation foreign key (invoiceID) references Invoice(invoiceID);

Insert into invoice values(1,’Brown’, 0, 1000);

insert into reservation

values (1,2, 1, TO\_DATE('2019-03-10', 'YYYY-MM-DD'),

TO\_DATE('2019-03-10', 'YYYY-MM-DD'), 'Conference',

TO\_DATE('2019-03-10', 'YYYY-MM-DD'),

100, 'Confirmed', 'Brown');

**PROCEDURE 16**

create or replace procedure reservationService(res\_id IN number) as

resattend integer;

occupiedTimeSlot exception;

cursor service\_cursor is select s.\* from services s, reservation r where s.reservationID = r.reservationID and r.reservationID = res\_ID;

begin

select numPeople into resattend from reservation where reservationID = res\_id;

dbms\_output.put\_line('Number Attending: ' || resattend);

for item in service\_cursor

loop

dbms\_output.put\_line(item.serviceType);

end loop;

Exception

when no\_data\_found

then

dbms\_output.put\_line('Reservations not found. Try again?');

end;

**PROCEDURE 17**

create or replace procedure showSpecificService(servicename IN varchar) as

noService exception;

errorcheck integer;

cursor service\_cursor is select s.serviceid, s.servicetype, r.\* from services s, reservation r where s.reservationID = r.reservationID and s.servicetype = servicename;

begin

select count(\*) into errorcheck from services s, reservation r where s.reservationID = r.reservationID and s.servicetype = servicename;

if errorcheck = 0

then

raise noservice;

else

for item in service\_cursor

loop

dbms\_output.put\_line(item.serviceType);

end loop;

end if;

Exception

when noService

then

dbms\_output.put\_line('Services not found. Try again?');

end;

**PROCEDURE 18**

set SERVEROUTPUT ON;

create or replace procedure displayIncome(eventID in number) as

noService exception;

errorcheck integer;

cursor service\_cursor is select s.serviceid, s.servicetype, s.costperperson, r.numpeople, r.reservationID

from services s, reservation r, eventplace e, hall h

where s.reservationID = r.reservationID and r.hallid = h.hallid and h.eventplaceID = e.eventplaceid and e.eventplaceid = eventid;

totalpeople number;

unitcost number;

total number;

begin

select count(\*) into errorcheck from services s, reservation r, eventplace e, hall h

where s.reservationID = r.reservationID and r.hallid = h.hallid and h.eventplaceID = e.eventplaceid and e.eventplaceid = eventid;

totalpeople := 0;

if errorcheck = 0

then

raise noservice;

else

total := 0;

for item in service\_cursor

loop

dbms\_output.put\_line('Reservation: ' || item.reservationID || ' Number of People: ' || item.numpeople || ' Service ID: ' || item.serviceID || ' Service Type: ' || item.servicetype || ' Cost: ' || item.costperperson);

total := total + (item.numpeople \* item.costperperson);

end loop;

dbms\_output.put\_line('Total: ' || total);

end if;

Exception

when noService

then

dbms\_output.put\_line('Services not found. Try again?');

end;

**PROCEDURE 19**

create or replace procedure Typehall(ep\_id\_h in number)

as

hallcountS number;

hallcountM number;

hallcountL number;

begin

select count(hallType) into hallcountS

from Hall WHERE hallType = 'Small' and eventplaceid = ep\_id\_h;

select count(hallType) into hallcountM

from Hall WHERE hallType = 'Medium' and eventplaceid = ep\_id\_h;

select count(hallType) into hallcountL

from Hall WHERE hallType = 'Large' and eventplaceid = ep\_id\_h;

dbms\_output.put\_line('Small :'|| hallcountS);

dbms\_output.put\_line('Medium :'|| hallcountM);

dbms\_output.put\_line('Large :'|| hallcountL);

end;

**PROCEDURE 20**

create or replace procedure eventInvoice(reservID in number) as

cname varchar(50);

epname varchar(50);

btype varchar(50);

adres varchar(50);

hid number(5);

hprice number(10);

numserv number(5);

stype varchar(50);

costpp number(10);

dcount integer(5);

tcost number(10);

begin

select invoice.customername, eventplace.eventplacename, eventplace.buildingType, eventplace.address, hall.hallID,

hall.hallprice, count(services.serviceId), services.serviceType, services.costperperson, invoice.discountsapplied,

invoice.totalcost into cname, epname, btype, adres, hid, hprice, numserv, stype, costpp, dcount, tcost

from invoice, eventplace, hall, services, reservation

where hall.eventplaceid = eventplace.eventplaceid and services.reservationid = reservation.reservationid and

hall.hallid = reservation.hallid and reservation.invoiceid = invoice.invoiceid

and reservation.reservationid = reservID

group by invoice.customername, eventplace.eventplaceid, eventplace.buildingType, eventplace.address, hall.hallID,

hall.hallprice, services.serviceType, services.costperperson, invoice.discountsapplied, invoice.totalcost,

eventplace.eventplacename;

dbms\_output.put\_line('Customer: ' || cname);

dbms\_output.put\_line('Event place name: ' || epname);

dbms\_output.put\_line('Building type: ' || btype);

dbms\_output.put\_line('Address: '|| adres);

dbms\_output.put\_line('Hall ID'|| hid);

dbms\_output.put\_line('Hall price'|| hprice);

dbms\_output.put\_line('Number of services'|| numserv);

dbms\_output.put\_line('Service type'|| stype);

dbms\_output.put\_line('Cost per person'|| costpp);

dbms\_output.put\_line('Discounts'|| dcount);

dbms\_output.put\_line('Total cost'|| tcost);

exception when no\_data\_found then

dbms\_output.put\_line('Not found');

end;

**PROCEDURE 21**

create or replace procedure IncByState(event\_state in char) as

sumHall number(5);

sumServ number(5);

sumSum number(6);

E\_ID number(5);

discount number(5);

counting number(5);

cursor stateID is

select eventplaceid as ID\_event from eventplace

where event\_state = state\_name;

begin

for ID\_event in stateID

loop

select e.eventplaceid, count(e.eventplaceid), sum(h.hallprice), sum(s.costperperson), i.discountsApplied

into e\_id, counting, sumHall, sumServ, discount

from eventplace e, hall h, services s, invoice i, reservation r

where e.state\_name = event\_state and h.eventplaceid = e.eventplaceid and h.hallid = r.hallid and

i.invoiceid = r.invoiceid and s.reservationid = r.reservationid

group by e\_id, sumhall, sumserv

order by e\_id;

sumsum := sumhall + sumserv;

sumsum := sumsum \* discount;

dbms\_output.put\_line('ID: ' || e\_id || ' Total hall income: ' || sumhall || ' Total service income: ' || sumserv);

dbms\_output.put\_line('Total income: ' || sumsum);

end loop;

exception when no\_data\_found then

dbms\_output.put\_line('Not found');

end;