Online Lecture Hall Booking Portal

Project, CS315: Introduction to Database Systems

Ankesh Kumar Singh (Y9090)

April 10, 2013

---

1. Application Description

The application is intended to facilitate the procedure of booking lecture halls by automating the process online.

1.1 Present Method

The current procedure of booking is a manual one. Booking of lecture hall is managed by LHC office, where they maintain a register containing the details of bookings. At the beginning of semester, time table of various courses is frozen and no bookings are possible on those time slots. However, this time table is not applicable on holidays. A booking is typically made by:

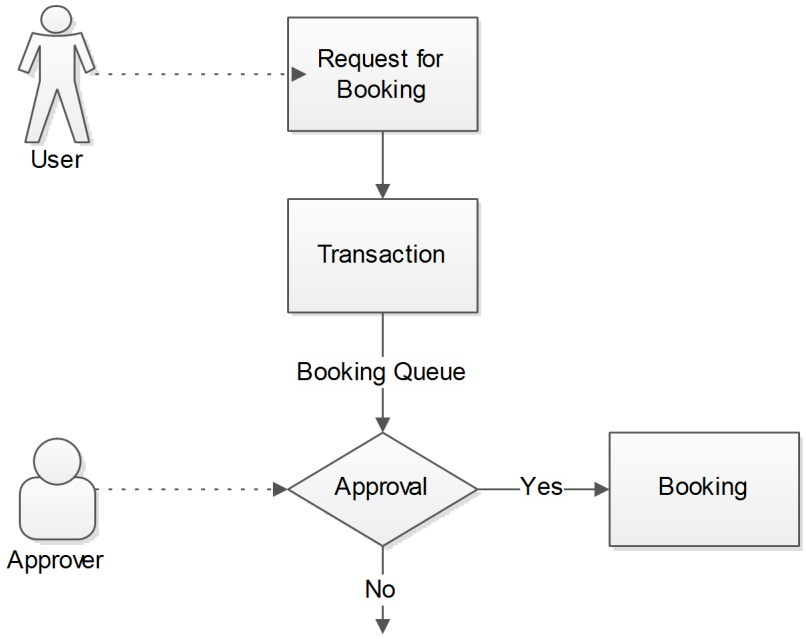
- A faculty member for an extra class/quiz
- A student with permission from a faculty member for a special lecture (includes bookings by SPO approved by SPO head)
- A student club/cell for some special lecture or workshop or competition

For a booking made by a course instructor, only confirmation from LHC office is needed. For a booking by student, it must be approved by a faculty member, followed by approval from DOAA and then confirmation from LHC office. For booking by a club, request is initiated by the coordinator, which is approved by president/general secretary of that council, followed by approval from DOSA, and then DOAA, and finally confirmation by LHC office.

In any of the above cases, if air conditioning is required, approval is also needed from Deputy Directors office.

1.2 Proposed Design

By this application, a person requesting for a booking may be able to know the availability of various lecture halls at the time of booking. Once the booking has been initiated, it will go for approval to appropriate authority in sequence. Finally, it would be with LHC office, where the booking is confirmed. Typical workflow for a booking is as follows:



2. EER Model

The database design is carried out in a manner so as to conform to these requirements. An extended ER model is utilized for initial table design. The entities are User, Booking, Lecture hall and Calendar. The relationships are Transaction, Approval, Location and Weekly booking.

Some of the typical queries to be addressed:

- Find all free lecture halls in a given time slot
- Find all requests that require approval by DOSA
- Find all requests rejected by any authority
- Find the booking corresponding to a cancellation request

- Find all bookings that have not expired corresponding to a given user

Some of the typical changes to be addressed:

- Create a new booking for a lecture hall
- Cancel the existing booking by processing cancellation request
- Approve a booking for which user has permission to do so

Additional features provided by application:

- Send email alerts for booking requests, approvals, confirmations and confirmations
- Send SMS alerts by using a SMS gateway

2.1 Data Description

The application consists of users who can request for bookings, as well as approve bookings. Users are classified into student, faculty, administrator and office staff. Students are further classified into executives and coordinators. Application is authenticated by using user ID and password for each user. Additional details are provided in order to identify a given user. Bookings are essentially of 2 types, those which are frozen by academic time table and those which are requested by different users. A lecture hall is identified by its number and also has information about capacity.
Final EER diagram for application is shown in Figure 1.

Field	**Domain type**	**Description**
User	*entity*	Any user accessing the application
userID	varchar(10)	Unique userID to authenticate the application
password	varchar(50) hash	Password to authenticate application
name	varchar (30)	Name of user
email	varchar(50)	Email ID to send alerts and contact if necessary
contactno	varchar(15)	Phone no. in case clarifications are required or to send alerts
address	text	Institute address: For students: room no. and hostel no. For faculty: office address For office staff: office address

Student	*subclass*	Users who are students
rollno	varchar(10)	Student roll no.
Executive	*subclass*	General secretaries of student councils and president
post	varchar(10)	president, sntsecy, cultsecy, sportsecy, fmcsecy
Coordinator	*subclass*	Students who are club coodinators, hobby group leaders, part of festival organizing teams and so on
club	varchar(20)	Description of the student group
Faculty	*subclass*	Users who are faculty members
facID	varchar(10)	Unique faculty ID for each faculty member
Auth	*subclass*	Users who have authority to approve bookings (other than student executives and faculty not in administrative framework)
type	varchar(10)	DOSA, DOAA, DD, SPO Head etc.
Office	*subclass*	Users who are office staff i.e. they finalize bookings and manage lecture halls
type	varchar(8)	LHC, Audi, OAT etc
Booking	*entity*	Details for a booking
bookID	int	unique ID for a booking
date	Date	date of booking
start	Time	Start time
end	Time	End time
requirement	varchar(10)	Additional equipments needed: Multimedia projector, OHP, Microphones (multivalued attribute)
ac	boolean	Whether air conditioning is needed.
Lechall	*entity*	Details for a lecture hall
hallno	varchar(8)	Hall no. eg: L7, CS101 etc
capacity	int	capacity of lecture hall
Calendar	*entity*	Days corresponding to all dates which are working days in academic calendar
date	Date	
day	varchar(2)	
Transaction	*relation*	Details for a transaction
type	char(1)	B=Book, C=Cancel, S=Shift
datetime	Timestamp	Date and time when a transaction is requested
details	text	Additional information about the booking, eg: "Takneek competition"

Approval	*relation*	Approval of a booking by required users
state	char(1)	P=Pending, A=Approved, R=Rejected
Location	*relation*	Location for a particular booking
TimeTable	*relation*	Fixed academic time table for a semester
course	varchar(10)	
day	varchar(2)	Day of the week
start	Time	Lecture start time
end	Time	Lecture end time

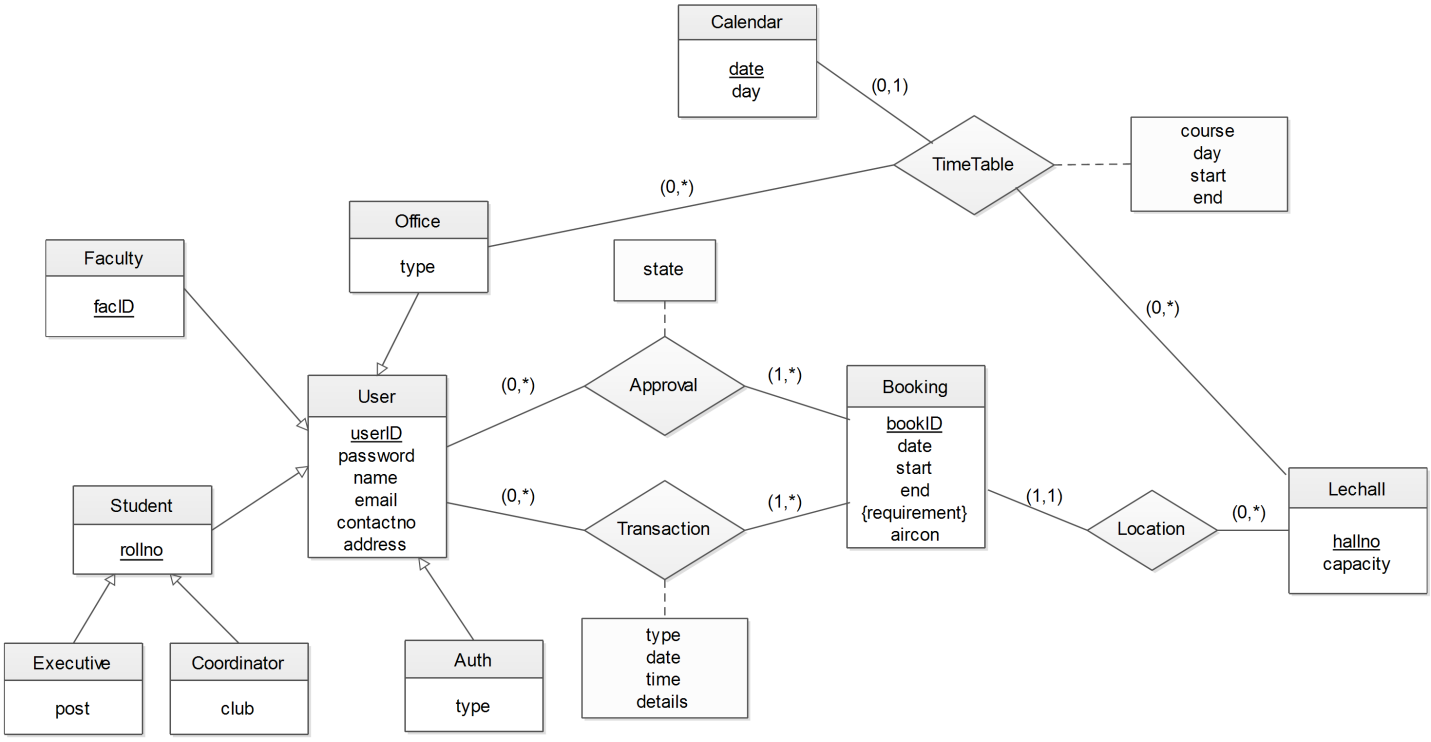


Figure 1: EER Diagram

3. Table Design

The EER model is reduced to obtain an initial table design. The referential integrity and domain constraints are shown in SQL Create statements in 3.4.

3.1 Initial Design

User (userId, password, name, email, contactno, address)
Student (userId, rollno)
Executive (rollno, post)
Coordinator (rollno, club)
Faculty (userId, facId)
Office (userId, type)
Auth (userId, type)
Transaction (userId, bookId, transNo, type, atTime, details)
Approval (userId, bookId, state, updated)
Booking (bookId, date, start, end, aircon)
Requirement (bookId, equip)
Location (bookId, hallno)
Lechall (hallno, capacity)
TimeTable (date, hallno, course, type, day, start, end)
Calendar (date, day)

3.2 Functional Dependencies

1. User:
   userId → (password, name, email, contactno, address)

2. Student:
   userId → rollno

3. Executive:
   rollno → post

4. Coordinator:
   rollno → club

5. Faculty:
   userId → facId

6. Auth and Office:
   userId → type

7. Transaction:
   transNo → (userId, bookId, type, atTime, details)
   (userId, bookId) → (type, atTime, details)

8. Approval:
   (userId, bookId) → (state, atTime)

9. Booking:
   bookId → hallno

10. Lechall:
    hallno → capacity

11. TimeTable and Calendar:
    date → day
    (course, day, type) → (hallno, start, end)

3.3 Normalized Tables

Table	**Normal Form**	
User	BCNF	userId is superkey
Student	BCNF	
Executive	BCNF	
Coordinator	BCNF	
Faculty	BCNF	
Office	BCNF	
Auth	BCNF	
Transaction	3-NF	In order to keep the relation dependency preserving, it is not decomposed.
Approval	BCNF	(userId, bookId) is superkey
Booking	BCNF	
Requirement	BCNF	
Location	BCNF	
Lechall	BCNF	
Calendar	BCNF	

Relations with just two attributes are in BCNF. TimeTable is not in BCNF as (date→day) FD exists and date is not a superkey. Hence it is decomposed. It turns out that one of the relations formed is already accounted for in Calendar. Hence the relation reduces to:

TimeTable	BCNF	(day, course, type, hallno, start, end)

3.4 SQL Create Statements

Many table names and attribute names conflict with sql syntax. Hence the terminology is adjusted accordingly.

```
create table users (
      userId varchar(10) not null,
      password varchar(50) not null,
      name varchar(30) not null,
      email  varchar(50) not null,
      contactno  varchar(15) not null,
      address  text,
      primary key (userId)
);
create table student (
      userId varchar(10) not null,
      rollno varchar(10) not null,
      primary key (rollno),
      foreign key (userId) references users
);
create table executive (
      rollno varchar(10) not null,
      post varchar(10) not null,
      primary key (rollno, post),
      foreign key (rollno) references student
);
create table coordinator (
      rollno varchar(10) not null,
      club varchar(20) not null,
      primary key (rollno, club),
      foreign key (rollno) references student
);
create table faculty (
      userId varchar(10) not null,
      facId  varchar(10) not null,
      primary key (userId, facId),
      foreign key (userId) references users
);
create table office (
      userId varchar(10) not null,
      offtype  varchar(10) not null,
      primary key (userId, offtype),
      foreign key (userId) references users
);
create table auth (
      userId varchar(10) not null,
      authtype varchar(10) not null,
      primary key (userId, authtype),
      foreign key (userId) references users
```

```
45 );
   create table booking(
47       bookId serial primary key,
         bookDate date not null,
49       startTime  time not null,
         endTime  time,
51       aircon boolean default false
   );
53 create table transaction(
         transNo  serial primary key,
55       userId varchar(10) not null,
         bookId int not null,
57       transtype  char(1) not null,
         atTime timestamp default 'now',
59       details  text,
         foreign key (userId) references users,
61       foreign key (bookId) references booking
   );
63 create table approval(
         userId varchar(10) not null,
65       bookId int not null,
         state  char(1) default 'P',
67       updated  timestamp default 'now',
         primary key (userId, bookId),
69       foreign key (userId) references users,
         foreign key (bookId) references booking
71 );
   create table requirement(
73       bookId int not null,
         equip  varchar(10),
75       foreign key (bookId) references booking
   );
77 create table lechall(
         hallno varchar(8) not null,
79       capacity int,
         primary key (hallno)
81 );
   create table location(
83       bookId   int not null,
         hallno varchar(8) not null,
85       foreign key (bookId) references booking,
         foreign key (hallno) references lechall
87 );
   create table calendar(
89       dateOf date,
         dayOf  varchar(2),
91       primary key (dateOf)
   );
93 create table timetable(
```

```
        dayOf   varchar(2) not null,
95      course varchar(10),
        meettype char(1),
97      hallno varchar(8) not null,
        startTime  time not null,
99      endTime  time,
        foreign key (hallno) references lechall
101 );
```

4. Application

For developing the application, three additional tables were needed.

precedence: The order in which approvals are needed

instructor: Faculty members who are course instructors

alerts: Email and SMS alerts to be processed

```
1 create table instructor(
  facid varchar(10) not null,
3 userId varchar(10) not null,
  course varchar(10),
5 foreign key (userId, facid) references faculty on delete cascade);

7 create table precedence(
      approver varchar(10) not null,
9     weight int,
      primary key(approver)
11 );

13 create table alerts(
      alertno  serial primary key,
15    userId varchar(10) not null,
      alerttype  varchar(5),
17    alertto  varchar(20),
      msg  text not null,
19    alertAt  timestamp default 'now',
      delivered  boolean default false,
21    foreign key (userId) references users
  );
```

4.1 Platform and Description

The application is a web-app with client side UI supported by HTML 5, CSS 3 and javascript using jQuery library in particular. Server side scripting is done in PHP 5 and database used in PostGreSQL.

The application allows the user to login, make a booking, check the status of bookings made by that user and cancel them. A user with approving rights may approve or reject bookings. In addition to this, users can view all bookings and core academic time table.

4.2 Source Description

index.php	The main application. Entire UI is created dynamically by this script
settings.php	Required parameters like database server, username and password, cookie variables and time out
pgdb.php	Make a database connection, also has some utility functions
components.php	
– loginform()	create login form along with error message call back JS
– footer()	make page footer
– dashboard()	make a dashboard for user with profile description and useful actions
form.php	Generate booking form
avail.php	Used by form to query for available bookings in a timeslot
addbook.php	Process request for a new booking
viewbook.php	Display all bookings requested by a user
cancel.php	Process cancellation request
viewappr.php	Display all bookings requiring approval by a user
approve.php	Process approval or rejection request
timetable.php	View academic core time table
users.php	
–refreshCookie()	extend session if user is active
–nextApprovalAlert()	called to send alerts to next required approver

4.3 UI Features

1. New Booking : Make a new booking as per requirement
2. Bookings>Live : Bookings that are either pending or approved and not past their date. Also shows which one is approved and gives an option to cancel.
3. Booings>Completed : Bookings past their date and not rejected.

4. Booings>Completed : Bookings that had been rejected.

5. Approvals>Required : Bookings requiring approval from a user. Gives an option to approve and shows which one is pending approval.

6. Shows user profiles and allows edit using Ajax.