IIT PALAKKAD

Indian Institute of Technology, Palakkad

DataBase Management Systems Laboratory(CS3120)

Project on

Railway Reservation System

Team No: 11

Group Members:

111901006 Aiswarya H

111901025 Jasir K

111901026 Joel Sam Mathew

Abstract

The procedure of reserving tickets has never been easier than it is now, thanks to online reservations. The railway reservation system is a computerized mechanism for reserving and canceling train seats ahead of time. It also facilitates the users to enquire about the availability of trains on the basis of source, destination and date. This system has also made it possible for passengers to check information, such as if their ticket is confirmed or in the waiting list.

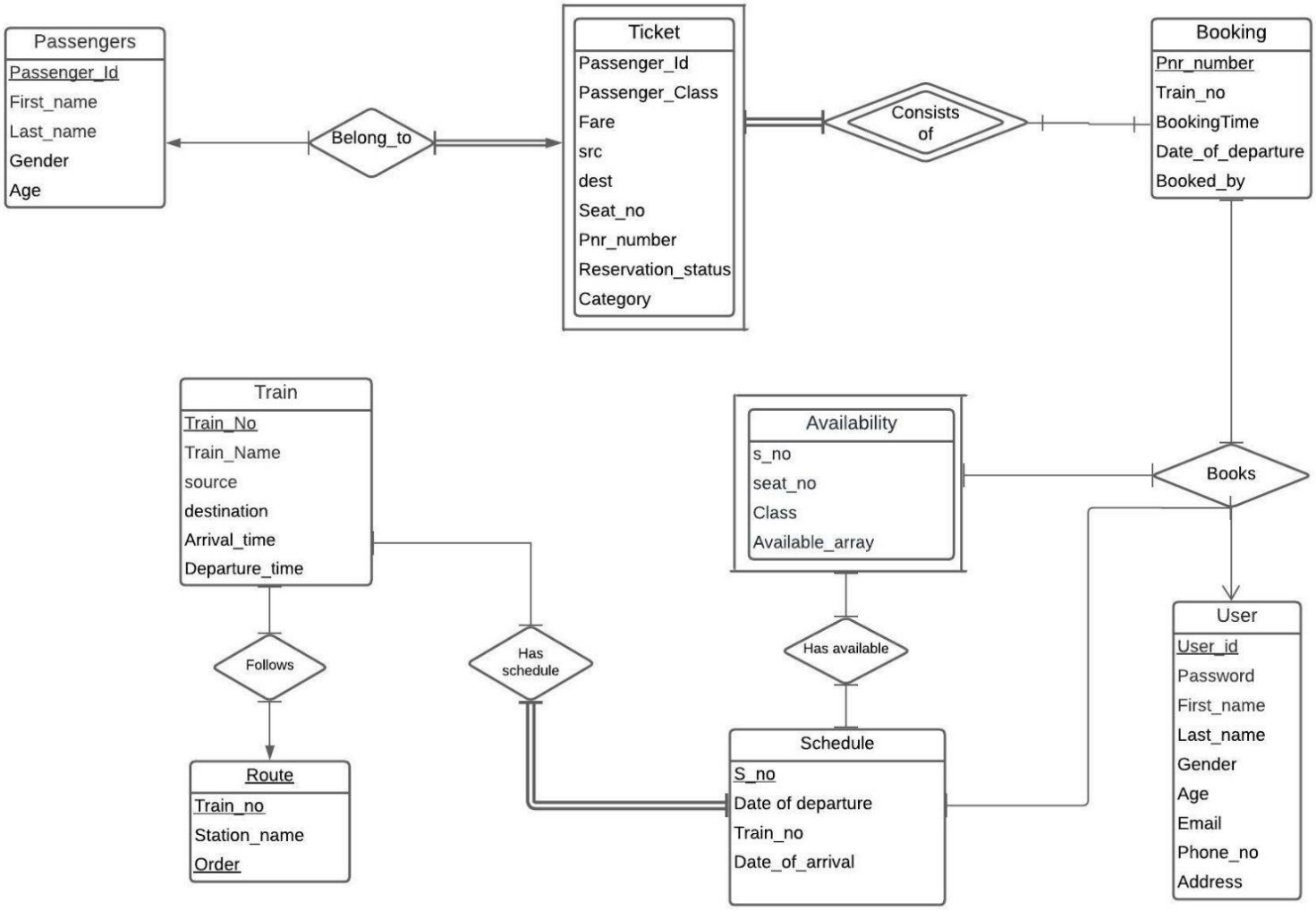
The objective of this project is to design and maintain a railway reservation database with basic functionalities.The database contains records of different trains, their status, details of passengers, and station details. The project simplifies the model by abstracting out some details and forming certain assumptions in order to reduce the complexity of the railway reservation system in reality.

Assumptions

The following assumptions have been made to simplify the implementation of this system:

1. The total number of seats in each type of `passenger_class` will be 10.
2. One booking can correspond to many tickets but a ticket is associated with a single passenger.
3. The types of `passenger_class` have been reduced to 2, namely, AC and Sleeper.
4. The booking can be cancelled only as a whole(All tickets corresponding to the entire booking will be cancelled).

ER Diagram


Passengers
Passenger_Id
First_name
Last_name
Gender
Age
Belong_to
Ticket
Passenger_Id
Passenger_Class
Fare
src
dest
Seat_no
Pnr_number
Reservation_status
Category
Consists of
Booking
Pnr_number
Train_no
BookingTime
Date_of_departure
Booked_by
Train
Train_No
Train_Name
source
destination
Arrival_time
Departure_time
Follows
Route
Train_no
Station_name
Order
Has schedule
Availability
s_no
seat_no
Class
Available_array
Has available
Schedule
S_no
Date of departure
Train_no
Date_of_arrival
Books
User
User_id
Password
First_name
Last_name
Gender
Age
Email
Phone_no
Address

Describing entities and relationships

Entities :

1. User

Attributes	data types	Constraints
user_id	integer	Not null
password	varchar(45)	Not null
first_name	varchar(45)	Not null
last_name	varchar(45)	Not null
gender	varchar(45)	Gender in (“Male”,”Female”,”Others”)
age	integer	Check age>0
email	varchar(45)	
phone_no	varchar(20)	Not null
address	text	

2. Booking

Attributes	Data types	Constraints
pnr_number	integer	Not null
booking_time	Time without time zone	Not null
booked_by	integer	Not null
train_no	integer	Not null
date_of_departure	date	Not null

Foriegn Key Constraints:

booked_by (References User)

train_no (References Train)

3. Train

Attributes	**Data types**	**Constraints**
train_no	integer	Not null
train_name	varchar(45)	Not null
source	varchar(45)	Not null
destination	varchar(45)	Not null
arrival_time	Time without time zone	Not null
departure_time	Time without time zone	Not null

4. Ticket (Weak Entity)

Attributes	**Data types**	**Constraints**
passenger_id	integer	Not null
passenger_class	varchar(45)	passenger_class in(“Sleeper”,”AC”)
fare	integer	Check fare > 0
seat_no	integer	seat_no >= 0 and seat_no <= 10
reservation_status	varchar(45)	reservation_status in (‘Confirmed’, ‘Waiting List’)
pnr_number*	integer	Not null
category	varchar(45)	Category in (“Tatkal”,”General”,”Ladies”,”Senior”)
src	varchar(45)	Not null

dest	varchar(45)	Not null

*Note: Since Ticket is a weak entity, `pnr_number` being the primary key of the identifying entity(Booking), it is taken in the table Ticket.

Foriegn Key Constraints:

Pnr_number (References Booking) ON DELETE CASCADE

Passenger_id (References Passenger)

5. Passenger

Attributes	**Data types**	**Constraints**
passenger_id	integer	Not null
first_name	varchar(45)	Not null
last_name	varchar(45)	Not null
gender	varchar(45)	Gender in (“Male”,”Female”,”Others”)
age	integer	Check age>0

6. Schedule

Attributes	**Data types**	**Constraints**
s_no	integer	Not null
date_of_departure	date	Not null
train_no	integer	Not null
date_of_arrival	date	Not null

Foriegn Key Constraints:

train_no (References Train)

7. Availability

Attributes	Data types	Constraints
s_no*	integer	Not null
seat_no	integer	seat_no >= 0 and seat_no <= 10
class	varchar	class in(“Sleeper”,”AC”)
availability_arr	boolean[]	Not null

*Note: Since Availability is a weak entity, `s_no` being the primary key of the identifying entity(Schedule), it is taken in the table Availability.

Foriegn Key Constraints:

s_no (References schedule)

8. Route

Attributes	**Data types**	**Constraints**
train_no	integer	Not null
station_name	varchar(45)	Not null
order	integer	Check order > 0

Foriegn Key Constraints:

train_no (References train)

Relationships:

- **Books** - Ternary Relationship between User, Booking,Availability and Schedule

  Describes a booking made by a user with the associated train on a particular date.

- **Belong_to** - Between Ticket and Passenger

  Represents a one-to-one correspondence between a ticket and a passenger. There is total participation from the ticket side as every ticket must be associated with a passenger.

- **Consists_of** - Between Booking and Ticket

  Gives all the tickets associated with a booking. This is the identifying relationship for the weak entity Ticket.

- **Has_schedule** - Between Schedule and Train

  Connects the schedule of the train with the train details. The train_status has total participation in the relation and each entry in schedule corresponds to a single train.

- **Follows** - Between Train and Route

  Connects the entity train with its associated route. Route has full participation as every record in the route corresponds to a train.

- **Has available** - Between Schedule and Availability

  Gives the seat availability for a particular instance of train.

General Constraints

The attribute wise constraints for each table has already been mentioned above in the list of entities. Some of the notable constraints and the reasons for them are listed below:

- The seat_no should be >= 0 and <= 10. This comes from the assumption that there are 10 seats in each passenger class of a train.The seat number 0 denotes that it is in the waiting list.
- There are only 4 possible values for a category : General, Tatkal, Ladies and Senior Citizen.
- A booking can have either of the two reservation statuses: Confirmed or Waiting List.

Integrity Constraints

Some of the key integrity constraints which are foreign key constraints are mentioned along with the entities above. The following are some additional constraints which help in preserving data integrity:

- The date of departure used in booking should be an actual date of departure for that train in the schedule. It's a referential integrity constraint.
- The order in route corresponding to the source in booking should be less than the order in route for destination in the booking.

Functionalities

1. Create a Booking
2. View Ticket,reservation status and passenger details
3. Login
4. View Train Details
5. Search Train Details for specific source and destination
6. Compute fare
7. Implementation of Waiting List
8. Implementation of Route
9. Cancel a Booking
10. Insert, Update or Delete Train details

Functions

1. Trainlist

The function `trainlist` takes the source and destination stations and outputs the trains which go through the source and destination stations(in the **route**) and the arrival and departure times of these trains along with the date of departure.

2. Passenger_details

The function `passenger_details` takes a pnr number and gives the details of all the passengers associated with the booking corresponding to that pnr number.

3. Compute_fare

The function `compute_fare` computes the fare of a ticket for a passenger. It depends on:

- Passenger_class : AC or Sleeper
- Category: General, Tatkal, Ladies, Senior Citizen
- Source and Destination

4. Login

The function `login` checks if the specified user exists in the database, and if the password entered is correct by taking the username and password. This is done for authentication purposes.

5. ***Assign_seat***

   The function `assign_seat` assigns a seat to the passenger for whom the ticket is booked, by checking availability of the seats. It will check availability based on :
   - Passenger_class
   - Desired source and destination for the passenger
   - Train (Train_no)
   - Date of departure

Procedures

1. ***Book***

   The procedure `book` allows a user to make a booking by taking user id, train no and the date of departure as arguments. A pnr number is generated(serially) and the booked time is recorded. A view named `curr_booking` is created by the same to show the booking that was just(lastly) made by the book procedure.

2. ***Book_ticket***

   The procedure `book_ticket` will enable us to enter all the passenger details one by one for a booking,which is given by the pnr_number taken as an argument. It also allows specifying the passenger_class, category, source and destination stations for individual passengers. The following things are done by the procedure:

   1) The passenger related details are entered as a new row in the passenger table. The p_id of the entered passenger is stored to be entered in the ticket table.
   2) The train number and the date of departure are obtained from the booking table with the help of pnr number.
   3) We find orders of the stations which will be used within the `compute_fare` function.
   4) A seat number is assigned to the passenger with the help of `assign_seat` function.
   5) If the previous step was successful then the passenger gets a confirmed seat, else he/she is put on the waiting list. This reservation status along with other necessary information is entered into the ticket table.

   Note that the actual booking (setting availability of seat = false) is done by a trigger on insertion into the ticket (explained in the triggers section).

3. ***Cancel***

   The procedure `cancel` provides a way to cancel a booking by just taking the pnr_number corresponding to the booking. Note that as mentioned in assumptions the

deletion of a booking is atomic, i.e., the whole booking can be canceled only as a whole. It will simply delete the booking and all the tickets corresponding to the pnr_number in the booking and ticket table respectively. The deletion of passenger entry corresponding to a ticket as well as freeing the seat is done by the triggers `delete_entry` and `free_seat` upon deletion in ticket (explained in the triggers section).

Triggers

- ***Avail_seat***
  Whenever there is a new entry in the ticket table(insertion), `avail_seat` trigger is triggered and it executes the set_avail trigger function.

  - ***Set_avail***

    It updates the seat availability in that train for the newly booked seat to unavailable for source to destination stations. The `available_arr` basically has size (no of stations - 1) and stores information about seat availability during those intervals. Given a source and destination for a particular train on a specific date it marks all intervals in the array as unavailable(false) between that source and destination.

- ***Delete_entry***

  Whenever an entry in the ticket table is deleted, `delete_entry` is triggered and it executes the `delete_passenger` trigger function.

  - ***Delete_passenger***

    It will delete the passenger corresponding to the ticket from the passenger table. This is done basically to ensure consistency in the database as ticket and passenger have a one to one correspondence.

- ***Free_seat***
  Whenever a ticket is canceled from the ticket table, then `free_seat` is triggered and the trigger function `free_avail` is executed.
  - ***Free_avail***

    It updates the seat availability in that train for the canceled seat to available for the given source to destination stations. If the ticket that was canceled was ‘Confirmed seat’, it will free(set to true) all the intervals in the available array between the two stations (source and destination) for the seat in the canceled ticket.

- ***Waiting list implementation***

  The `free_avail` also implements the allotment of seats to the waiting list upon cancellation of seats. It checks if there is someone on the waiting list and assigns that seat(which was just canceled) and changes the status to 'Confirmed'.

A short note on preserving consistency in the database

The functions `book`, `book_ticket` and `cancel` ensure that the tables booking, ticket and passenger are consistent with each other. The triggers `avail_seat` and `free_seat` help in ensuring that the seat availability is consistent with the bookings made.

Roles

1. User

   The user is a person who basically has login access to the system and can make a booking. The user can search and view the details of all trains and also see the availability of seats in a train. The user also has the privilege to cancel the booking, but only as a whole(all tickets corresponding to a ticket will be deleted in one go).The user can view the list of all passengers associated with the booking that he/she makes.

   Authentication:

   - Only select(or view) privileges on views corresponding to train or route
   - View the availability of seats in trains
   - Update and view the details corresponding to the user
   - Make or cancel a booking
   - View the details of booking and the passengers associated with the booking

2. Passenger

   A passenger is a person who is a part of a booking made by the user. The passenger, like the user, can search and view the details of all trains. But a passenger who is not a user can’t make or delete a booking. A passenger can only view the details of his/her journey and doesn’t have access to the entire booking.

   Authentication:

   - Only select(or view) privileges on views corresponding to trains
   - View the details of his/her booking with pnr number
   - View only on the details corresponding to the passenger

3. Admin

Admin is a superuser who controls the entire database. Only the admin has privileges for data manipulation(insert, update or delete) in train, route and schedule relations.

Authentication:

- Insert,delete or update on the details of trains
- Modify the schedule of trains(date of arrival or departure) or route
- View privileges on all the relations

Views

1. *Trains_view*

It lists all the trains, their source and destinations along with the arrival and departure times. It also includes the date of departure and date of arrival and lists the train in order of the date of departures.

This general view is created in order to provide an overall list of all the trains and the details of their departure and arrival to the user or passengers.

Most of the other views(like passenger details) involve some arguments (like source or destination, or pnr number) and are row specific (like passengers can view only his/her details). They are implemented using functions or authorization techniques.

Indices

1. *Hash_station*

We will use an index `hash_station` on route on the attribute station. This is because in many queries and functions we are accessing the station, especially to check if it lies on the route of some train. A query which would make use of this index will be:

```
select train_no from route where station_name = 'Kanpur Central Railway Station (CNB)';
```

Note that the index is created on **route** rather than **train** as this would be more useful since the interested source and destination need not be the actual source and destination of the train.

2. *Btree_dod*

A Btree index is being created on the departure date since it is frequently used to look for a ticket,for some given source and destination, across a range of dates. This will give

a better execution time as b tree generally works well with range queries. A query which would make use of this index will be:

```
select * from trains_view
where src = 'Kanpur Central Railway Station (CNB)'
and dest = 'Allahabad Junction Railway Station (ALD)'
and date_of_departure >= '2022-04-04'
```

3. ***Hash_pnr***

We create a hash index on the pnr number attribute of the booking table. This is because we test for equality on the pnr number in the booking table for many functions,procedures and triggers, especially the ones meant for booking and cancellation. Hence creating a hash index would be very useful for queries like

```
select * from booking where pnr_number = 10;
```

Such queries are indirectly used in the booking and cancellation procedures.