



**CASE STUDY
OF
RAILWAY RESERVATION
MANAGEMENT SYSTEM
IN
RELATIONAL DATABASE DESIGN**

Submitted To:- Dept. of Computer Science & Engineering

CASE STUDY IN RELATIONAL DATABASE DESIGN

TITLE: RAILWAY RESERVATION MANAGEMENT
SYSTEM

STUDENT'S NAME: TANISH

PARNEET SINGH

PRATIBHA BHANDARI

REWA BHARDWAJ

MAITRI VYAS

GUIDE: DR. VIKAS SOLANKI

Abstract

The objective of this case study is to represent an idea on the management of the railways reservation system. This project is helpful in managing the data for a large number of users. Input for this case study is taken from its informal specification to a relational schema using entity-relationship modeling and its translation to relational model, to database schema, to implementation of the database, to interactive SQL querying of the installed database (SQL/Oracle).

The Railway Reservation System facilitates the passengers to enquire about the trains available on the basis of source and destination, Booking and Cancellation of tickets, enquire about the status of the booked ticket, etc. The aim of case study is to design and develop a database maintaining the records of different trains, train status, and passengers.

This project contains Introduction to the Railways reservation system. It is the computerized system of reserving the seats of train seats in advanced. It is mainly used for long route. Online reservation has made the process for the reservation of seats very much easier than ever before.

In our country India, there are number of counters for the reservation of the seats and one can easily make reservations and get tickets. Then this project contains entity relationship model diagram based on railway reservation system and introduction to relation model. There is also design of the database of the railway reservation system based on relation model. Example of some SQL queries to retrieves data from rail management database.

Acknowledgments

I would like to express my gratitude to all of those who made it possible to complete this project, in particular to Dr. Vikas Solanki. I would also like to thank my family for their understanding and continuous support.

I record it as my privilege to deeply thank Dr. Meenu Khurana, Dean of CSE Dept. for providing us the efficient faculty and facilities to make our ideas into reality.

I express my sincere thanks to Dr. Vikas Solanki for his novel association of ideas, encouragement, appreciation and intellectual zeal which motivated us to venture this project successfully.

Finally, it is pleased to acknowledge the indebtedness to all those who devoted themselves directly or indirectly to make this project report success.

Contents

Abstract.....	i
Acknowledgements	ii
Chapter 1: Introduction	6
1.1 Database Management System	6
1.2 Relational Database Management System	6
1.3 Feasibility Study.....	6
1.4 ER Diagram.....	7
1.5 Brief Introduction	9
1.6 Objectives.....	9
Chapter 2: Database Management System for Railways Reservation..	10
2.1 INFORMAL DESCRIPTION.....	10
2.2 Need Analysis.....	10
2.3 ER Diagram.....	11
2.4 Physical Schema	12
2.5 LOGICAL MODEL	14
2.6 DDL Commands	15
2.7 INTERACTIVE QUERIES	24
Chapter 3 : Conclusion and Future Work	28
Bibliography	iii
Appendix.....	iv

Chapter 1: Introduction

1.1 Database Management System

Database Management system is a software which is used to manage the database. For example: MySQL, Oracle, etc. are a very popular commercial database which are used in different applications. It is a collection of inter-related data which helps in efficient retrieval, insertion and deletion of data from database and organizes the data in the form of tables, views, schemas, reports etc. It provides protection and security to the database. In the case of multiple users, it also maintains data consistency.

1.2 Relational Database Management System

A relational database management system (RDBMS) is a collection of programs and capabilities that enable IT teams and others to create, update, administer and otherwise interact with a relational database, examples of the most popular RDBMS are MYSQL, Oracle, IBM DB2, and Microsoft SQL Server database. It is a database system that stores and retrieves data in a tabular format organized in the form of rows and columns. Interdependencies among the tables are expressed by data values rather than by pointers. This allows a high degree of data independence.

1.3 Feasibility Study

1.3.1 Economic Feasibility:

Economically, this Railway DBMS requires at least 30k – 40k of laptop for the smooth working of the system. The software being used i.e., Oracle 11g express edition and Chrome browser are free of cost, so no additional cost for the software.

1.3.2 Technical Feasibility:

Technically also, this project is possible and not very difficult to implement, anyone with knowledge of computers can easily handle this management system.

1.3.3 Operational Feasibility:

For Operational feasibility, the system can do all the inserting, updating, deleting tasks; it can also fetch the required data efficiently and with accuracy. Replacing the old paper-based management system with DBMS can greatly improve the efficiency.

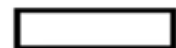
1.4 ER diagram

ER Diagram stands for Entity Relationship Diagram, also known as ERD is a diagram that displays the relationship of entity sets stored in a database. In other words, ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes and relationships.

ER Diagrams contain different symbols that use rectangles to represent entities, ovals to define attributes and diamond shapes to represent relationships.

An entity is representation of a real-life object that can be living one or non-living.

Above is represented by the following diagram



1.4.1 Strong Entity:

An entity that is not dependent on any other entity and can be identified on its own and by its own members. For example: Restaurant is a strong entity.

Above is represented by the following diagram



1.4.2 Weak Entity:

An entity that is dependent on other entities and need references from them to be identified. Taking above example, Bill is a weak

entity as its existence and identification depends on the restaurant.

Above is represented by the following diagram



1.4.3 Simple Attribute:

Attributes that cannot be divided or split further into more attributes. Like a phone number or gender.

Above is represented by the following diagram



1.4.4 Composite Attribute:

Attributes that can be divided or split further into more attributes. Like Student name can be further divided into First Name, Middle Name and Last Name.

Above is represented by the following diagram



1.4.5 Key Attribute:

Attribute that is used in DBMS as a reference/identification attribute for the entity is a Key Attribute. Roll No of a Student is a Key Attribute.

Above is represented by the following diagram



1.4.6 Derived Attribute:

Attribute that doesn't exist physically in database, but we derive it on the base of another attribute. Age can be derived from date of birth.

Above is represented by the following diagram



1.4.7 Multi-Valued Attribute:

Attribute that may or may not have more than value for an entity. Address of a student can be more than one.

Above is represented by the following diagram



1.4.8 Relationship:

Relationships are associations between or among entities.

Above is represented by the following diagram



1.4.9 Weak relationship:

Weak Relationships are connections between a weak entity and its owner.



1.5 Brief introduction

DBMS is an organized collection of data. In this system, the data is consistent and the processes like create, update, delete, retrieve becomes easier.

The main purpose of having a Database for Railways is to ensure that the data is consistent, more reliable and to minimize the chances of errors that occurs due to manual entry of the data. The speed of obtaining the data will be fast. This system can be integrated with web, so that people can have direct access to the information they want like getting their ticket info, etc., and will save a lot of time. This project will keep the information regarding the trains from source to destination, passengers, seat availability, booking & checking status tickets and if the passenger cancels the ticket, then the deleted information will also be recorded.

The management system will contain various relations, entities and entity have attributes as per the requirements. Various SQL Queries will used to fetch the information from the tables.

1.6 Objectives

1. To fetch the information about the trains available for the given route and destination.
2. To handle the booking and cancelation of tickets.
3. To book tickets online & get information about their booked trains.
4. To Improve the efficiency of searching data and reduce the redundancy in management system.
5. To enquire about the seats available & to know their status
6. To save the time people spend at booking counter and hassle of going to railway station, waiting in line for so long and get it done. Same is the scenario for the cancellation of ticket.

Chapter 2: A Database Management System for Railways Reservation

2.1 Informal Description

This project is about creating the database about Railway Reservation System. The railway reservation system facilitates the passengers to enquire about the trains available on the basis of source and destination, booking and cancellation of tickets, enquire about the status of the booked ticket, etc. The aim of case study is to design and develop a database maintaining the records of different trains, train status and passengers.

2.2 Need Analysis

This Project maintains a database for railway reservation and it should be capable of fetching information about trains, their seat availability, handling booking and cancellation of tickets, checking their reservation status, authorization of users. This is being made to simplify and ease the process of reservation of tickets and to improve the efficiency.

Hence to implement railway reservation system we need User table for authorization of user, Train table for storing information corresponding to each train, one tables to store the number of seats for their respective classes of each train, a table to fetch the available seats information, booked tickets table to store the information about the tickets and their status. If user cancels the ticket, the user's id and ticket information will be stored in a different table.

To Store the data efficiently, triggers will be used; to capitalize the user's data and then store it. On booking a ticket, a trigger will check the availability and if there are seats vacant, set status to be confirmed and decrease the seat availability by number of seats booked.

2.3 ER Diagram

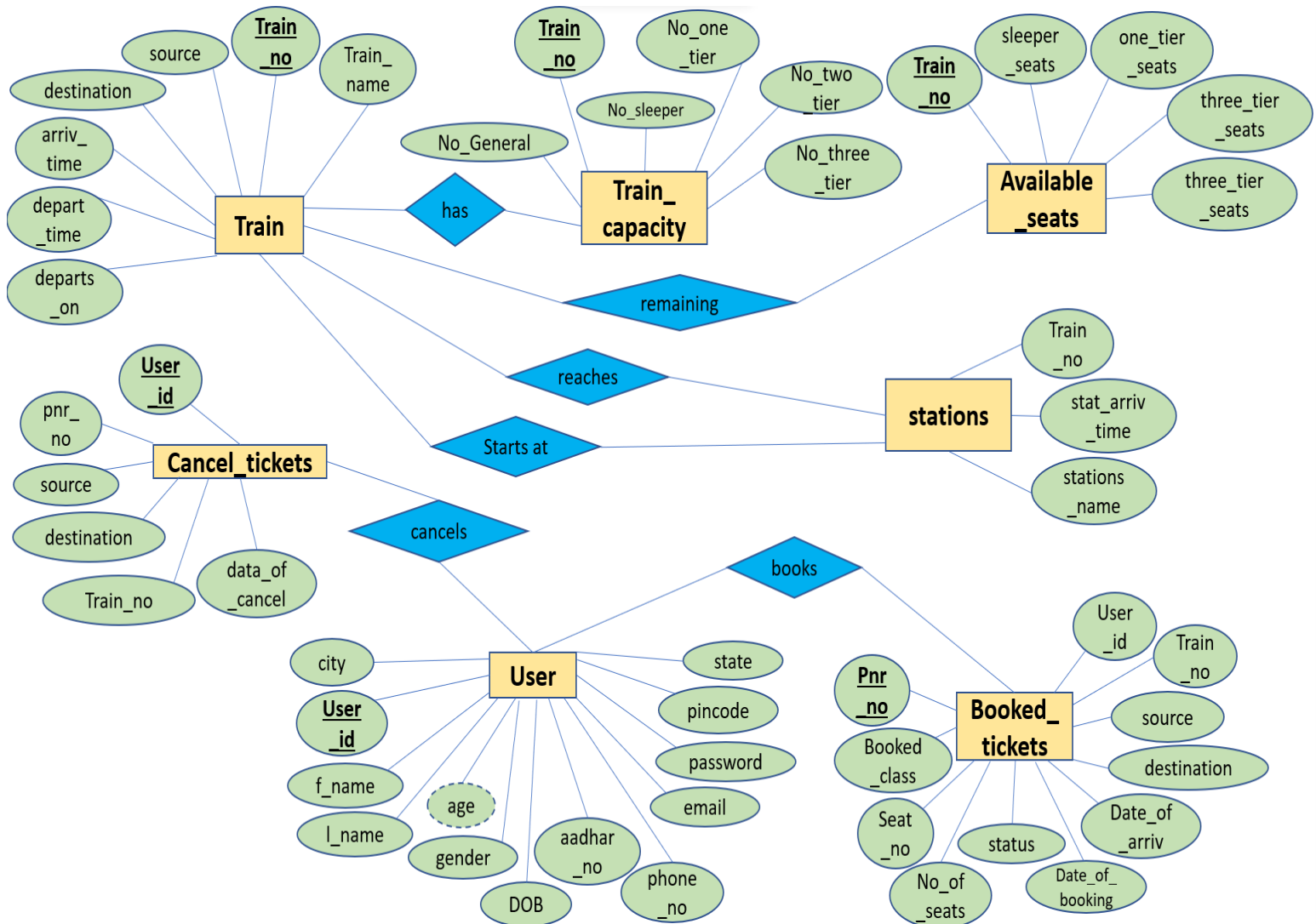


Fig 2.3.1: ER DIAGRAM

As presented through this ER Diagram, **Train** table has been created to store the information about trains. **Train** has some **Train capacity** which is a relation between these two tables which will be more like a constant valued table; Train also has some **Available Seats** signifying the seats remaining in particular train. Train has some source and destination and that data will be stored in **Stations** table where it reaches and departs from. A **User** table is created to store all the user's data; the person can book ticket and the ticket's data will be stored in **Booked Tickets** table and the number of seats will be deducted from **Available Seats** table. If user wishes to cancel the ticket, then the cancelled ticket info will be stored in **Cancel Ticket** table.

2.4 Physical Schema

2.4.1 Physical Schema Of Train:

Attribute	Data-Type	Size	Constraints
TRAIN_NO	NUMBER	-	Primary Key [PK] Foreign Key [FK]
TRAIN_NAME	VARCHAR2	50	-
SOURCE	VARCHAR2	50	-
DESTINATION	VARCHAR2	50	-
ARRIV_TIME	TIMESTAMP	-	-
DEPART_TIME	TIMESTAMP	-	-
DEPARTS_ON	VARCHAR2	100	-

2.4.2 Physical Schema Of Train Capacity:

Attribute	Data-Type	Size	Constraints
TRAIN_NO	NUMBER	-	Primary Key [PK] Foreign Key [FK]
NO_GENERAL	NUMBER	-	-
NO_SLEEPER	NUMBER	-	-
NO_ONE_TIER	NUMBER	-	-
NO_TWO_TIER	NUMBER	-	-
NO_THREE_TIER	NUMBER	-	-

2.4.3 Physical Schema Of Available Seats:

Attribute	Data-Type	Size	Constraints
TRAIN_NO	NUMBER	-	Primary Key [PK] Foreign Key [FK]
NO_GENERAL	NUMBER	-	-
NO_SLEEPER	NUMBER	-	-
NO_ONE_TIER	NUMBER	-	-
NO_TWO_TIER	NUMBER	-	-
NO_THREE_TIER	NUMBER	-	-

2.4.4 Physical Schema Of Users:

Attribute	Data-Type	Size	Constraints
USER_ID	NUMBER	-	Primary Key [PK]
F_NAME	VARCHAR2	50	-
L_NAME	VARCHAR2	50	-
AGE	NUMBER	-	-
GENDER	VARCHAR2	30	-
DOB	DATE	-	-
AADHAR_NO	NUMBER	-	-
PHONE_NO	NUMBER	-	-
EMAIL	VARCHAR2	50	-
PASSWORD	VARCHAR2	50	-
CITY	VARCHAR2	30	-
STATE	VARCHAR2	30	-
PINCODE	NUMBER	-	-

2.4.5 Physical Schema Of Booked Tickets:

Attribute	Data-Type	Size	Constraints
PNR_NO	NUMBER	-	Primary Key [PK]
USER_ID	NUMBER	-	Foreign Key [FK]
TRAIN_NO	NUMBER	-	Foreign Key [FK]
SOURCE	VARCHAR2	50	-
DESTINATION	VARCHAR2	50	-
DATE_OF_ARRIV	DATE	-	-
DATE_OF_BOOKING	DATE	-	-
STATUS	VARCHAR2	50	-
NO_OF_SEATS	NUMBER	-	-
BOOKED_CLASS	VARCHAR2	50	-
SEAT_NO	NUMBER	-	-

2.4.6 Physical Schema Of Cancel Ticket:

Attribute	Data-Type	Size	Constraints
PNR_NO	NUMBER	-	Primary Key [PK]
USER_ID	NUMBER	-	-
TRAIN_NO	NUMBER	-	-
SOURCE	VARCHAR2	50	-
DESTINATION	VARCHAR2	50	-
DATE_OF_CANCEL	DATE	-	-

2.4.7 Physical Schema Of Stations:

Attribute	Data-Type	Size	Constraints
TRAIN_NO	NUMBER	-	Primary Key [PK] Foreign Key [FK]
STATIONS_NAME	VARCHAR2	255	-
STAT_ARRIV_TIME	VARCHAR2	255	-

2.5 Logical Model

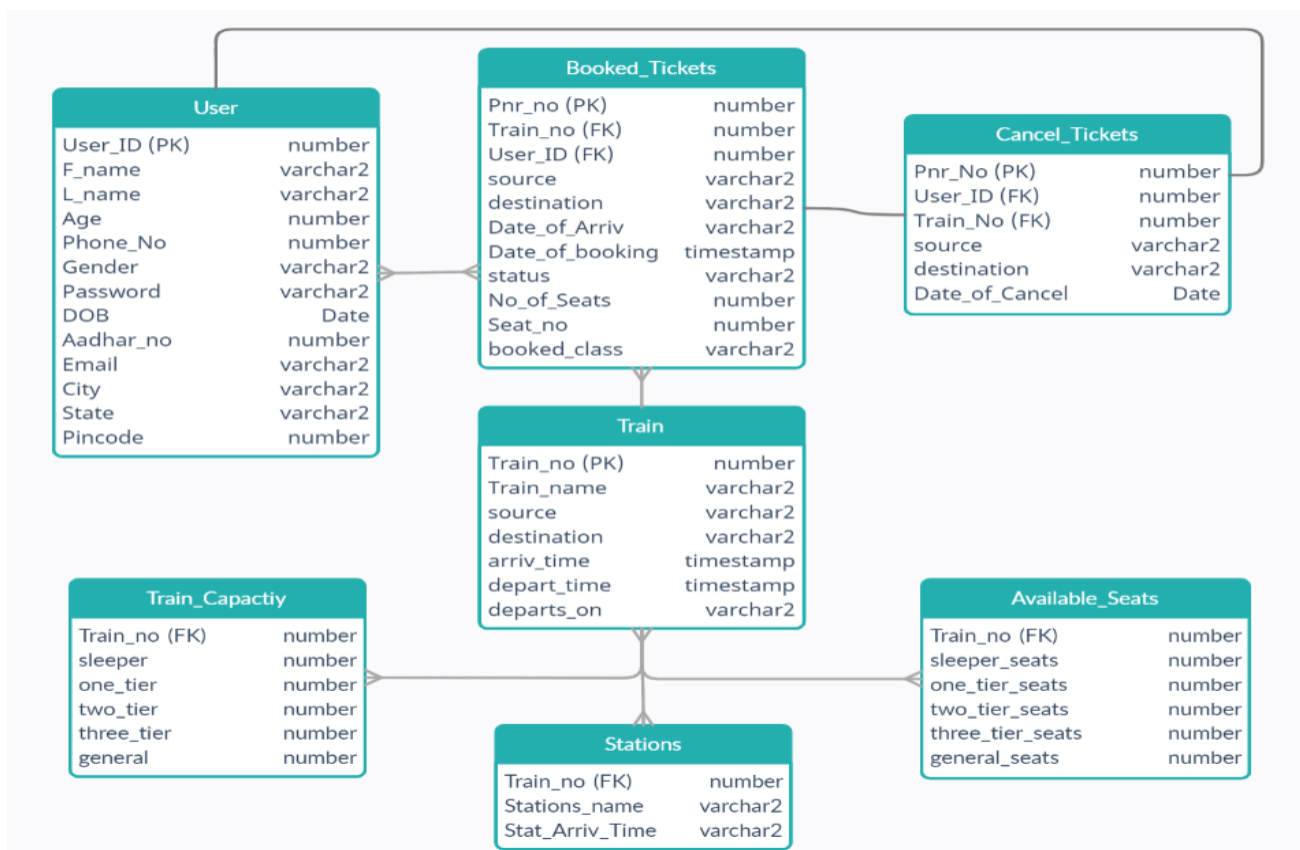


Fig 2.5.1: LOGICAL MODEL

2.6 DDL Commands

2.6.1 TABLES

2.6.1.1 TRAIN:

```
CREATE TABLE TRAIN(  
  TRAIN_NO NUMBER PRIMARY KEY,  
  TRAIN_NAME VARCHAR2(50),  
  SOURCE VARCHAR2(50),  
  DESTINATION VARCHAR2(50),  
  ARRIV_TIME TIMESTAMP,  
  DEPART_TIME TIMESTAMP,  
  DEPARTS_ON VARCHAR2(100));
```

2.6.1.2 TRAIN_CAPACITY:

```
CREATE TABLE TRAIN_CAPACITY(  
  TRAIN_NO NUMBER PRIMARY KEY REFERENCES  
  TRAIN(TRAIN_NO),  
  SLEEPER NUMBER,  
  ONE_TIER NUMBER,  
  TWO_TIER NUMBER,  
  THREE_TIER NUMBER,  
  GENERAL NUMBER);
```

2.6.1.3 AVAILABLE_SEATS:

```
CREATE TABLE AVAILABLE_SEATS(  
  TRAIN_NO NUMBER PRIMARY KEY REFERENCES  
  TRAIN(TRAIN_NO),  
  SLEEPER_SEATS NUMBER,  
  ONE_TIER_SEATS NUMBER,  
  TWO_TIER_SEATS NUMBER,  
  THREE_TIER_SEATS NUMBER,  
  GENERAL_SEATS NUMBER);
```

2.6.1.4 USERS:

```
CREATE TABLE USERS(  
  USER_ID NUMBER PRIMARY KEY,  
  F_NAME VARCHAR2(50),  
  L_NAME VARCHAR2(50),  
  AGE NUMBER,  
  GENDER VARCHAR2(30),  
  DOB DATE,  
  AADHAR_NO NUMBER,  
  PHONE_NO NUMBER,  
  EMAIL VARCHAR2(50),  
  PASSWORD VARCHAR2(50),  
  CITY VARCHAR2(30),  
  STATE VARCHAR2(30),  
  PINCODE NUMBER);
```

2.6.1.5 BOOKED_TICKETS:

```
CREATE TABLE BOOKED_TICKETS(  
  PNR_NO NUMBER PRIMARY KEY,  
  USER_ID NUMBER REFERENCES USERS(USER_ID),  
  TRAIN_NO NUMBER REFERENCES TRAIN(TRAIN_NO),  
  SOURCE VARCHAR2(50),  
  DESTINATION VARCHAR2(50),  
  DATE_OF_ARRIV DATE,  
  DATE_OF_BOOKING DATE,  
  STATUS VARCHAR2(50),  
  NO_OF_SEATS NUMBER,  
  BOOKED_CLASS VARCHAR2(50),  
  SEAT_NO NUMBER);
```

2.6.1.6 CANCEL_TICKET:

```
CREATE TABLE CANCEL_TICKETS(  
  PNR_NO NUMBER PRIMARY KEY,  
  USER_ID NUMBER REFERENCES USERS(USER_ID),  
  TRAIN_NO NUMBER REFERENCES TRAIN(TRAIN_NO),  
  SOURCE VARCHAR2(50),  
  DESTINATION VARCHAR2(50),  
  DATE_OF_CANCEL DATE);
```


2.6.1.7 STATIONS:

```
CREATE TABLE STATIONS(  
  TRAIN_NO NUMBER PRIMARY KEY,  
  STATIONS_NAME VARCHAR2(255),  
  STAT_ARRIV_TIME VARCHAR2(255));
```

2.6.2 SEQUENCES

2.6.2.1 For User_id

```
CREATE SEQUENCE USER_SEQ  
START WITH 1000  
INCREMENT BY 1  
NOCACHE  
NOCYCLE;
```

2.6.2.2 For ticket number

```
CREATE SEQUENCE TICKET_NUM_SEQ  
START WITH 2354688880  
INCREMENT BY 1  
NOCACHE  
NOCYCLE;
```

2.6.2.3 For seat number

```
CREATE SEQUENCE SEAT_NO_SEQ  
START WITH 1  
INCREMENT BY 1  
NOCACHE  
NOCYCLE;
```

2.6.3 TRIGGERS

2.6.3.1 User_capitalize_tgr: (user)

Used to capitalize each entry in database

```
create trigger User_capitalize_tgr
Before insert or update on users
For each row
Begin
:new.f_name := upper(:new.f_name);
:new.l_name := upper(:new.l_name);
:new.gender := upper(:new.gender);
:new.email:= upper(:new.email);
:new.city:= upper(:new.city);
:new.state:= upper(:new.state);
End;
/
```

2.6.3.2 Booked_tkt_capitalize_tgr: (on booked_ticket)

Used to capitalize each entry in database

```
create trigger tkt_capitalize_tgr
Before insert or update on booked_tickets
For each row
Begin
:new.source := upper(:new.source);
:new.destination := upper(:new.destination);
:new.booked_class:= upper(:new.booked_class);
:new.status := upper(:new.status);
End;
/
```

2.6.3.3 Status_tgr:

If seats available, then status = confirmed

otherwise status = waiting

If status is confirmed, seats = seats - no_of_seats;

If seats not avail , then do nothing

```

create or replace trigger status_tgr
before insert or update on booked_tickets
for each row
follows tkt_capitalize_tgr

declare
avail_seats AVAILABLE_SEATS.sleeper_seats%TYPE;

begin
if(lower(:new.booked_class) = 'sleeper') then

select sleeper_seats into avail_seats from
available_seats where train_no =
(:new.train_no);

if(avail_seats < :new.no_of_seats) then

:new.status := 'WAITING';
dbms_output.put_line('Your Ticket is in
WAITING');

end if;

update AVAILABLE_SEATS set sleeper_seats =
avail_seats - (:new.no_of_seats) where train_no
= (:new.train_no);

end if;

if(lower(:new.booked_class) = 'one_tier') then
select one_tier_seats into avail_seats from
available_seats where train_no =
(:new.train_no);

if(avail_seats < :new.no_of_seats) then

:new.status := 'WAITING';
dbms_output.put_line('Your Ticket is in
WAITING');

```

```

end if;

update AVAILABLE_SEATS set one_tier_seats =
avail_seats - (:new.no_of_seats) where train_no
= (:new.train_no);

end if;

if(lower(:new.booked_class) = 'two_tier') then

select two_tier_seats into avail_seats from
available_seats where train_no =
(:new.train_no);

if(avail_seats < :new.no_of_seats) then

:new.status := 'WAITING';
dbms_output.put_line('Your Ticket is in
WAITING');

end if;

update AVAILABLE_SEATS set two_tier_seats =
avail_seats - (:new.no_of_seats) where train_no
= (:new.train_no);

end if;

if(lower(:new.booked_class) = 'three_tier')
then

select three_tier_seats into avail_seats from
available_seats where train_no =
(:new.train_no);

if(avail_seats < :new.no_of_seats) then

:new.status := 'WAITING';
dbms_output.put_line('Your Ticket is in
WAITING');

```

```

end if;

update AVAILABLE_SEATS set three_tier_seats =
avail_seats - (:new.no_of_seats) where train_no
= (:new.train_no);

end if;

if(lower(:new.booked_class) = 'general') then

select general_seats into avail_seats from
available_seats where train_no =
(:new.train_no);

if(avail_seats < :new.no_of_seats) then

:new.status := 'WAITING';
dbms_output.put_line('Your Ticket is in
WAITING');

end if;

update AVAILABLE_SEATS set general_seats =
avail_seats - (:new.no_of_seats) where train_no
= (:new.train_no);

end if;

if(:new.status = 'CONFIRMED') then
:new.seat_no := seat_no_seq.nextval;
dbms_output.put_line('Your Ticket has been
BOOKED');
end if;

end;
/

```

2.6.3.4 Cancel_tgr:

If user cancels ticket

Seats = seats + no_of_seats for particular class,
then insert into cancel_ticket table

```
create or replace trigger tkt_cancel_tgr
after delete on booked_tickets
for each row

declare
avail_seats AVAILABLE_SEATS.sleeper_seats%TYPE;

begin

insert into cancel_tickets values (:old.pnr_no,
:old.user_id, :old.train_no, :old.source,
:old.destination, SYSDATE);

if(lower(:old.booked_class) = 'sleeper' and
(:old.status) = 'CONFIRMED') then

select sleeper_seats into avail_seats from
available_seats where train_no = (:old.train_no);
update AVAILABLE_SEATS set sleeper_seats =
avail_seats + (:old.no_of_seats) where train_no =
(:old.train_no);
end if;

if(lower(:old.booked_class) = 'one_tier' and
(:old.status) = 'CONFIRMED') then

select one_tier_seats into avail_seats from
available_seats where train_no = (:old.train_no);
update AVAILABLE_SEATS set one_tier_seats =
avail_seats + (:old.no_of_seats) where train_no =
(:old.train_no);

end if;

if(lower(:old.booked_class) = 'two_tier' and
(:old.status) = 'CONFIRMED') then

select two_tier_seats into avail_seats from
available_seats where train_no = (:old.train_no);
```

```

update AVAILABLE_SEATS set two_tier_seats =
avail_seats + (:old.no_of_seats) where train_no =
(:old.train_no);

end if;

if(lower(:old.booked_class) = 'three_tier' and
(:old.status) = 'CONFIRMED') then

select three_tier_seats into avail_seats from
available_seats where train_no = (:old.train_no);
update AVAILABLE_SEATS set three_tier_seats =
avail_seats + (:old.no_of_seats) where train_no =
(:old.train_no);

end if;

if(lower(:old.booked_class) = 'general' and
(:old.status) = 'CONFIRMED') then

select general_seats into avail_seats from
available_seats where train_no = (:old.train_no);
update AVAILABLE_SEATS set general_seats =
avail_seats + (:old.no_of_seats) where train_no =
(:old.train_no);

end if;



end;
/

```

2.7 Interactive Queries

1. To get the user's info

```
SELECT USER_ID, CONCAT(F_NAME, CONCAT(' ', L_NAME)) AS NAME,  
DOB, PHONE_NO, EMAIL, CITY, PINCODE FROM USERS WHERE USER_ID =  
1005;
```

<input checked="" type="checkbox"/> Autocommit	Rows	10			Save	Run
SELECT USER_ID, CONCAT(F_NAME, CONCAT(' ', L_NAME)) AS NAME, DOB, PHONE_NO, EMAIL, CITY, PINCODE FROM USERS WHERE USER_ID = 1005;						
Results Explain Describe Saved SQL History						
USER_ID	NAME	DOB	PHONE_NO	EMAIL	CITY	PINCODE
1005	KISHORI MHASALKAR	09/05/1996	7485964152	KISHORI@GMAIL.COM	VADODARA	390001
1 rows returned in 0.00 seconds Download						


2. To check the seats availability


```
SELECT TRAIN_NO, SLEEPER_SEATS, ARRIV_TIME, DEPART_TIME,  
DEPARTS_ON FROM AVAILABLE_SEATS NATURAL JOIN TRAIN;
```

☒ Autocommit

Rows

10





Save

Run

SELECT TRAIN_NO, SLEEPER_SEATS, ARRIV_TIME, DEPART_TIME,
DEPARTS_ON FROM AVAILABLE_SEATS NATURAL JOIN TRAIN WHERE TRAIN.SOURCE = 'NEW DELHI' AND TRAIN.DESTINATION = 'RANCHI';

Results

Explain

Describe

Saved SQL

History



TRAIN_NO	SLEEPER_SEATS	ARRIV_TIME	DEPART_TIME	DEPARTS_ON
2454	20	10:25	16:10	MON,TUE,WED,THU,FRI,SAT,SUN

1 rows returned in 0.01 seconds

[Download](#)

3. To book ticket

```
INSERT INTO BOOKED_TICKETS VALUES(TICKET_NUM_SEQ.NEXTVAL, 1001,
02454, 'NEW DELHI', 'RANCHI', DATE '2021-11-20',DATE '2021-11-
19', 'CONFIRMED', 4, 'TWO_TIER', NULL);
```

☒ Autocommit Rows   Save Run

INSERT INTO BOOKED_TICKETS VALUES(
TICKET_NUM_SEQ.NEXTVAL, 1004, 02454, 'NEW DELHI', 'RANCHI', DATE '2021-11-20',DATE '2021-11-19', 'CONFIRMED', 4, 'TWO_TIER', NULL
);

Results [Explain](#) [Describe](#) [Saved SQL](#) [History](#)



Your Ticket has been BOOKED
Your Ticket has been BOOKED
Your Ticket has been BOOKED
Your Ticket has been BOOKED

1 row(s) inserted.

0.05 seconds

4. To get ticket info

```
SELECT PNR_NO, TRAIN_NO, DATE_OF_ARRIV, BOOKED_CLASS, SEAT_NO  
FROM BOOKED_TICKETS WHERE USER_ID = 1004;
```

☒ Autocommit Rows   Save Run

SELECT PNR_NO, TRAIN_NO, DATE_OF_ARRIV, BOOKED_CLASS, SEAT_NO FROM BOOKED_TICKETS
WHERE USER_ID = 1004;



Results [Explain](#) [Describe](#) [Saved SQL](#) [History](#)

PNR_NO	TRAIN_NO	DATE_OF_ARRIV	BOOKED_CLASS	SEAT_NO
2354688930	2454	11/20/2021	TWO_TIER	28
2354688931	2454	11/20/2021	TWO_TIER	29
2354688932	2454	11/20/2021	TWO_TIER	30
2354688929	2454	11/20/2021	TWO_TIER	27

4 rows returned in 0.00 seconds [Download](#)

5. To get the passenger's data travelling from new delhi to ranchi

```
SELECT PNR_NO, USER_ID, CONCAT(F_NAME, CONCAT(' ', L_NAME)) AS NAME, PHONE_NO, CITY FROM USERS NATURAL JOIN BOOKED_TICKETS WHERE SOURCE='NEW DELHI' AND DESTINATION = 'RANCHI' AND STATUS = 'CONFIRMED';
```

☒ Autocommit Rows   Save Run

```
SELECT PNR_NO, USER_ID, CONCAT(F_NAME, CONCAT(' ', L_NAME)) AS NAME, PHONE_NO, CITY FROM USERS NATURAL JOIN BOOKED_TICKETS WHERE SOURCE='NEW DELHI' AND DESTINATION = 'RANCHI' AND STATUS = 'CONFIRMED';
```



Results Explain Describe Saved SQL History

PNR_NO	USER_ID	NAME	PHONE_NO	CITY
2354688930	1004	NITIKA MISRA	9458761236	DELHI
2354688929	1004	NITIKA MISRA	9458761236	DELHI
2354688932	1004	NITIKA MISRA	9458761236	DELHI
2354688931	1004	NITIKA MISRA	9458761236	DELHI

4 rows returned in 0.01 seconds [Download](#)

6. To get the train's info

```
SELECT TRAIN_NAME, SOURCE, DESTINATION, ARRIV_TIME, DEPART_TIME, DEPARTS_ON FROM TRAIN WHERE TRAIN_NO = 02454;
```

☒ Autocommit Rows   Save Run

```
SELECT TRAIN_NAME, SOURCE, DESTINATION, ARRIV_TIME, DEPART_TIME, DEPARTS_ON FROM TRAIN WHERE TRAIN_NO = 02454;
```



Results Explain Describe Saved SQL History

TRAIN_NAME	SOURCE	DESTINATION	ARRIV_TIME	DEPART_TIME	DEPARTS_ON
NEW-DELHI-RANCHI RAJDHANI SPECIAL	NEW DELHI	RANCHI	10:25	16:10	MON,TUE,WED,THU,FRI,SAT,SUN

1 rows returned in 0.01 seconds [Download](#)

7. If user cancels his ticket

7.1 DELETE FROM BOOKED_TICKETS WHERE PNR_NO = 2354688926;



☒ Autocommit Rows   Save Run

delete from booked_tickets where pnr_no = 2354688926;

Results Explain Describe Saved SQL History

1 row(s) deleted.

7.2 SELECT * FROM CANCEL_TICKETS;

☒ Autocommit Rows   Save Run

select * from cancel_tickets

Results Explain Describe Saved SQL History

PNR_NO	USER_ID	TRAIN_NO	SOURCE	DESTINATION	DATE_OF_CANCEL
2354688926	1001	2454	NEW DELHI	RANCHI	11/20/2021

1 rows returned in 0.00 seconds [Download](#)

Chapter 3: Conclusion and Future Work

In this Project Railway reservation system, we have stored all the information about the Trains scheduled and the users' booking tickets and even status of trains, seats etc. This database is helpful for the applications which facilitate passengers to book the train tickets and check the details of trains and their status from their place itself it avoids inconveniences of going to railway station for each and every query they get.

We had considered the most important requirements only; many more features and details can be added to our project in order to obtain even more user-friendly applications. These applications are already in progress and in future they can be upgraded and may become part of amazing technology.

Bibliography

1. Database System Concepts', Abraham Silberschatz, Henry F. Korth, Sudharsan, McGraw- Hill, Seventh Edition.
2. Simplified Approach to DBMS, Parteek Bhatia, Gurvinder Singh.
3. CodeQuotient (online platform- <https://codequotient.com/>)
4. 'Database Systems', Ramez.Z.Elmasri, Shamkant B.Navathe, Pearson Education, Seventh Edition.
5. Introduction to PL/SQL by Ivan Bayross, BPB Publications, Fourth Edition

Appendix

<u>Name of Member</u>	<u>Role</u>
1. Maitri Vyas	Data Entry
2. Parneet Singh	Database Designer
3. Pratibha Bhandari	Implementing design
4. Rewa Bhardwaj	Implement design
5. Tanish	Database Designer