CONTENTS

INTRODUCTION

Database is an organized collection of data. The data is typically organized to model aspects of reality in a way that supports processes requiring information. A DBMS makes it possible for end users to create, read, update and delete data in a database. The DBMS essentially serves as an interface between the database and end users or application programs, ensuring that data is consistently organized and remains easily accessible. The DBMS manages three important things: the data, the database engine that allows data to be accessed, locked and modified and the database schema, which defines the database's logical structure. These three foundational elements help provide concurrency, security, data integrity and uniform administration procedures. The DBMS can offer both logical and physical data independence. That means it can protect users and applications from needing to know where data is stored or having to be concerned about changes to the physical structure of data.

The main purpose of maintaining database for Railway Management System is to reduce the manual errors involved in the booking and cancelling of tickets and make it convenient for the customers and providers to maintain the data about their customers and also about the seats available at them.Due to automation many loopholes that exist in the manual maintenance of the records can be removed. The speed of obtaining and processing the data will be fast. For future expansion the proposed system can be web enabled so that clients can make various enquiries about trains between stations. Due to this, sometimes a lot of problems occur and they are facing many disputes with customers. To solve the above problem, we design a data base which includes customer details, availability of seats in trains, no of trains and their details.

Technique Used :

SQL - Structured Query Language or SQL is a standard Database language which is used to create, maintain and retrieve the relational database. It is particularly used to work with structured data where there is relations associated within the data itself.

PL/SQL - PL/SQL is a block structured language that enables developers to combine the power of SQL with procedural statements.All the statements of a block are passed to oracle engine all at once which increases processing speed and decreases the traffic.

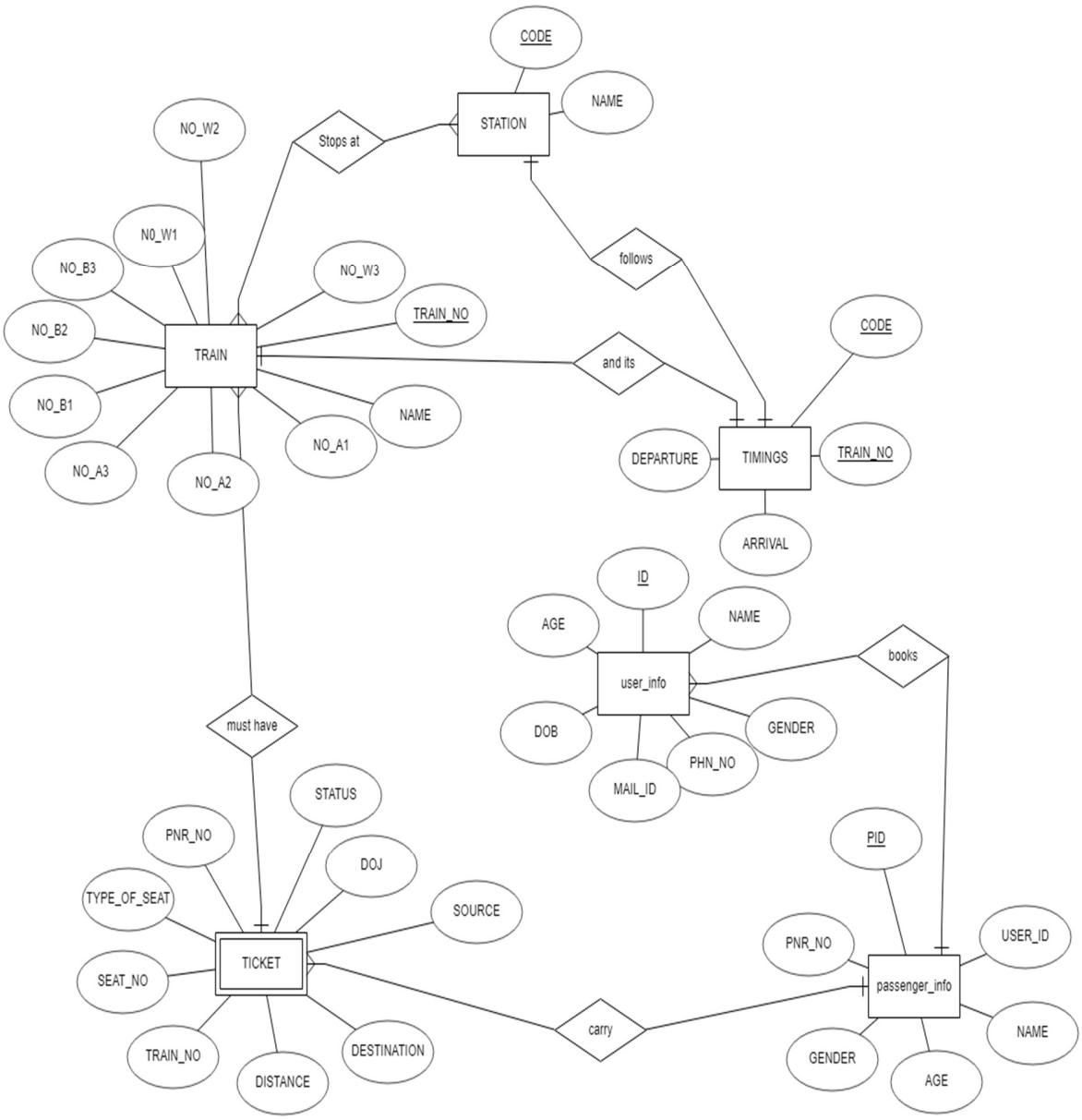
Disadvantages of SQL:

- SQL doesn't provide the programmers with a technique of condition checking, looping and branching.
- SQL statements are passed to Oracle engine one at a time which increases traffic and decreases speed.

Features of PL/SQL:

- PL/SQL is basically a procedural language, which provides the functionality of decision making, iteration and many more features of procedural programming languages.
- PL/SQL can execute a number of queries in one block using single command.
- One can create a PL/SQL unit such as procedures, functions, packages, triggers, and types, which are stored in the database for reuse by applications.
- PL/SQL provides a feature to handle the exception which occurs in PL/SQL block known as exception handling block.
- Applications written in PL/SQL are portable to computer hardware or operating system where Oracle is operational.

ER DIAGRAM


CODE
NAME
STATION
Stops at
NO_W2
N0_W1
NO_B3
NO_W3
NO_B2
TRAIN_NO
TRAIN
follows
and its
CODE
NO_B1
NAME
NO_A1
NO_A3
NO_A2
DEPARTURE
TIMINGS
TRAIN_NO
ARRIVAL
ID
AGE
NAME
user_info
books
DOB
GENDER
PHN_NO
MAIL_ID
must have
STATUS
PNR_NO
DOJ
TYPE_OF_SEAT
SOURCE
TICKET
SEAT_NO
PID
PNR_NO
USER_ID
passenger_info
carry
NAME
TRAIN_NO
DISTANCE
DESTINATION
GENDER
AGE

LIST OF ENTITIES AND ATTRIBUTES

ENTITES	ATTRIBUTES
User_info	**ID** Name Gender Age DOB Mail_ID Phn_no
passenger	**PID** User_ID Name Gender Age Pnr_no

Train	**Train_no** Name NO_A1 NO_A2 NO_A3 NO_B1 NO_B2 NO_B3 NO_W1 NO_W2 NO_W3
Station	**CODE** Name
Ticket	Train_no DOJ Source Destination Distance Seat_No Type_of_Seat PNR_NO Status
Timings	**CODE**

	Train_NO Arrival Departure

NORMALIZATION

1. 1 NF - Every table has atmost one values at the intersection of rows and columns and therefore every table is in first normal form.

2. 2 NF - Only table 'TIMINGS' has a composite key and other columns are fully dependent on this composite key, therefore it is in second normal form.

3. 3 NF – Every non-key column in every table is directly dependent on the key and there is no dependency between non-key columns. So, there are no transitive functional dependencies. Hence, the tables are in third normal form.

4. BCNF – As there are no cases of choices between overlapping composite keys at any stage of developing the tables, so we can say that every table is in BCNF also.

Hence, all the tables are normalized and redundancy has been removed from the database.

PL/SQL Code

Table Creation

Table-1 user_info

create table user_info(id number primary key,name varchar2(15),gender varchar2(1) check(gender in('f','F','m','M','o','O')),phn_no number,mail_id varchar2(30),dob date,age number default null);

Table-2 passenger_info

create table passenger_info(pid number primary key,user_id number references user_info(id),name varchar2(15),age number,gender varchar2(1) check(gender in('M','m','F','f','o','O')),pnr_no number unique);

Table-3 train

create table train(train_no number primary key,name varchar2(20),no_A1 number,no_A2 number,no_A3 number,no_B1 number,no_B2 number,no_B3 number,no_W1 number,no_W2 number,no_W3 number);

Table-4 Ticket

create table ticket(pnr_no number references passenger_info(pnr_no),doj date,source varchar2(10),destination varchar2(10),distance number,train_no number references train(train_no),seat_no number,type_of_seat number,status varchar2(10));

Table-5 Station

create table station(code varchar2(5) primary key,name varchar2(20));

Table-6 Timings

create table timings(code varchar2(5) references station(code),train_no number references train(train_no),arrival date,departure date,primary key(code,train_no));

INSERTION

INSERTION IN TABLE USER_INFO:

```
declare
rec user_info%rowtype;
begin
rec.name:=&Name;
rec.gender:=&Gender;
rec.phn_no:=&Phone_NO;
rec.mail_id:=&Mail_ID;
rec.dob:=&Date_Of_Birth;
insert into user_info
values(user_user_id.nextval,rec.name,rec.gender,rec.phn_no,rec.mail_id,rec.dob,null);
end;
```

```
SQL> select * from user_info;

        ID NAME            G     PHN_NO MAIL_ID                      DOB           AGE
---------- --------------- - ---------- ---------------------------- --------- ----------
       103 arsh            m  764392231 arsh123@hotmail.com          26-APR-99         20
       104 devi            f 9816627152 devil@gmail.com              08-JAN-98         21
       105 tayal           m 9457930987 tayal722@gmail.com           16-JUL-80         38
       106 mehak           f 8273728923 mehk@gmail.com               05-AUG-96         22
       107 mehak           f 8392632117 hello536@hotmail.com         27-SEP-88         30
```

INSERTION IN TABLE PASSENGER_INFO:

```
declare
rec passenger_info%rowtype;
begin
rec.user_id:=&User_ID;
rec.name:=&Name;
rec.age:=&Age;
rec.gender:=&Gender;
insert into passenger_info
values(passenger_id.nextval,rec.user_id,rec.name,rec.age,rec.gender,pnr.nextval);
end;
```

```
SQL> select * from passenger_info;

       PID    USER_ID NAME                 AGE G     PNR_NO
---------- ---------- -------------- ---------- - ----------
        27        104 manish                 21 m       2380
        29        107 manav                  19 m       2390
        31        104 mehak                  20 f       2400
        33        106 arsh                   22 m       2410
        35        105 devi                   16 f       2420
```

INSERTION IN TABLE TRAIN:

```
declare
rec train%rowtype;
begin
rec.train_no:=&Train_NO;
rec.name:=&Name;
rec.no_A1:=&MAX_Type1seats;
rec.no_A2:=&MAX_Type2seats;
rec.no_A3:=&MAX_Type3seats;
rec.no_B1:=&booked_seats_type1;
rec.no_B2:=&booked_seats_type2;
rec.no_B3:=&booked_seats_type3;
rec.no_W1:=&waiting_seats_type1;
rec.no_W2:=&waiting_seats_type2;
rec.no_W3:=&waiting_seats_type3;
insert into train
values(rec.train_no,rec.name,rec.no_A1,rec.no_A2,rec.no_A3,rec.no_B1,rec.no_B2,rec.no_B3,rec.no_W1,rec.no_W2,rec.no_W3);
end;
```

```
SQL> select * from train;

 TRAIN_NO NAME                 NO_A1      NO_A2      NO_A3      NO_B1      NO_B2      NO_B3      NO_W1      NO_W2      NO_W3
--------- -------------------- ---------- ---------- ---------- ---------- ---------- ---------- ---------- ---------- ----------
    12342 kalawati exp                  4          2          4         14         24         54          0          0          0
    25372 gayatri exp                   0          0          0         25         60         40         13         34         10
    25712 mumbai-delhi exp             10         20         29          4          9          7          0          0          0
    25122 ratnasagar exp                0          0          0         50         60         70         10         11         10
```

INSERTION IN TABLE STATION:

```
declare
rec station%rowtype;
begin
rec.code:=&Code_of_station;
rec.name:=&Name;
insert into station values(rec.code,rec.name);
end;
```

```
SQL> select * from station;

CODE  NAME
----- --------------------
PTA   patiala
AGC   Agra Cantt.
BNC   Bangalore cantt.
DLI   Delhi
BCT   Mumbai Central
```

INSERTION IN TABLE TIMINGS:

```
declare
rec timings%rowtype;
begin
rec.code:=&Code_of_station;
rec.train_no:=&Train_NO;
rec.arrival:=to_date(&Arrival_Time,'hh24:mi:ss');
rec.departure:=to_date(&Departure_Time,'hh24:mi:ss');
insert into timings values(rec.code,rec.train_no,rec.arrival,rec.departure);
end;
```

```
SQL> select code,train_no,to_char(arrival,'hh24:mi:ss'),to_char(departure,'hh24:mi:ss') from timings;

CODE    TRAIN_NO TO_CHAR( TO_CHAR(
----- ---------- -------- --------
BCT        12342 15:40:00 16:00:00
DLI        25712 09:10:00 09:30:00
AGC        25122 17:30:00 17:35:00
BNC        25712 12:00:00 23:10:00
```

INSERTION IN TABLE TICKET:

```
declare
rec ticket%rowtype;
begin
rec.pnr_no:=&PNR_NO;
rec.doj:=to_date(&Date_of_Journey,'dd-mm-yy');
rec.source:=&Source;
rec.destination:=&Destination;
rec.distance:=&Distance;
rec.train_no:=&Train_NO;
rec.seat_no:=&Seat_NO;
rec.type_of_seat:=&Type_of_seat;
insert into ticket
values(rec.pnr_no,rec.doj,rec.source,rec.destination,rec.distance,rec.train_no,rec.seat_no,rec.type_of_seat,rec.status);
end;
```

```
SQL> select * from ticket;

   PNR_NO DOJ       SOURCE     DESTINATIO   DISTANCE   TRAIN_NO    SEAT_NO TYPE_OF_SEAT STATUS
--------- --------- ---------- ---------- ---------- ---------- ---------- ----------
     2380 09-MAY-19 BNC        DLI             520      12342         23          1 B
     2390 23-JUN-19 AGC        BCT             230      25712         45          3 B
     2420 07-AUG-19 BNC        AGC             400      25122         78          2 W
     2400 30-MAY-19 PTA        BCT             450      25372         50          1 W
```

TRIGGERS

Trigger-1 For calculating age of user from date of birth:

create trigger calc_age_user

before insert or update of dob,age on user_info

for each row

begin

:new.age:=trunc(months_between(sysdate,:new.dob)/12);

end;

```
SQL> create trigger calc_age_user
  2  before insert or update of dob,age on user_info
  3  for each row
  4  begin
  5  :new.age:=trunc(months_between(sysdate,:new.dob)/12);
  6  end;
  7  /

Trigger created.
```

Trigger-2 To update the number of seats when booking a seat:

```
create trigger update_seats
before insert or update of type_of_seat on ticket
for each row
begin
declare
n number;
begin
if :new.type_of_seat=1 then
select no_A1 into n from train where :new.train_no=train_no;
if n>0 then
```

```
update train
set no_A1=no_A1-1,no_B1=no_B1+1
where train_no=:new.train_no;
:new.status:='B';
elsif n=0 then
update train
set no_W1=no_W1+1
where train_no=:new.train_no;
:new.status:='W';
end if;
elsif :new.type_of_seat=2 then
select no_A2 into n from train where :new.train_no=train_no;
if n>0 then
update train
set no_A2=no_A2-1,no_B2=no_B2+1
where train_no=:new.train_no;
:new.status:='B';
elsif n=0 then
update train
set no_W2=no_W2+1
where train_no=:new.train_no;
:new.status:='W';
end if;
elsif :new.type_of_seat=3 then
select no_A3 into n from train where :new.train_no=train_no;
if n>0 then
update train
set no_A3=no_A3-1,no_B3=no_B3+1
where train_no=:new.train_no;
:new.status:='B';
elsif n=0 then
update train
set no_W3=no_W3+1
where train_no=:new.train_no;
:new.status:='W';
end if;
end if;
end;
exception
when no_data_found then
```

null;
end;

```
SQL> create trigger update_seats
  2  before insert or update of type_of_seat on ticket
  3  for each row
  4  begin
  5  declare
  6  n number;
  7  begin
  8  if :new.type_of_seat=1 then
  9  select no_A1 into n from train where :new.train_no=train_no;
 10  if n>0 then
 11  update train
 12  set no_A1=no_A1-1,no_B1=no_B1+1
 13  where train_no=:new.train_no;
 14  :new.status:='B';
 15  elsif n=0 then
 16  update train
 17  set no_W1=no_W1+1
 18  where train_no=:new.train_no;
 19  :new.status:='W';
 20  end if;
 21  elsif :new.type_of_seat=2 then
 22  select no_A2 into n from train where :new.train_no=train_no;
 23  if n>0 then
 24  update train
 25  set no_A2=no_A2-1,no_B2=no_B2+1
 26  where train_no=:new.train_no;
 27  :new.status:='B';
 28  elsif n=0 then
 29  update train
 30  set no_W2=no_W2+1
 31  where train_no=:new.train_no;
 32  :new.status:='W';
 33  end if;
 34  elsif :new.type_of_seat=3 then
 35  select no_A3 into n from train where :new.train_no=train_no;
 36  if n>0 then
 37  update train
 38  set no_A3=no_A3-1,no_B3=no_B3+1
 39  where train_no=:new.train_no;
 40  :new.status:='B';
 41  elsif n=0 then
 42  update train
 43  set no_W3=no_W3+1
 44  where train_no=:new.train_no;
 45  :new.status:='W';
 46  end if;
 47  end if;
 48  end;
 49  exception
 50  when no_data_found then
 51  null;
 52  end;
 53  /

Trigger created.
```

Trigger-3 To update number of seats when cancelling a ticket:

```
create trigger del_seat
before delete on ticket
for each row
begin
declare
n number;
begin
if :old.type_of_seat=1 then
if upper(:old.status)='B' then
update train
set no_A1=no_A1+1,no_B1=no_B1-1
where train_no=:old.train_no;
elsif upper(:old.status)='W' then
update train
set no_W1=no_W1-1
where train_no=:old.train_no;
select min(pnr_no) into n from ticket where upper(status)='W';
update ticket
set status='B'
where pnr_no=n;
end if;
elsif :old.type_of_seat=2 then
if upper(:old.status)='B' then
update train
set no_A2=no_A2+1,no_B2=no_B2-1
where train_no=:old.train_no;
```

```
elsif upper(:old.status)='W' then
update train
set no_W2=no_W2-1
where train_no=:old.train_no;
select min(pnr_no) into n from ticket where upper(status)='W';
update ticket
set status='B'
elsif :old.type_of_seat=3 then
if upper(:old.status)='B' then
update train
set no_A3=no_A3+1,no_B3=no_B3-1
where train_no=:old.train_no;
elsif upper(:old.status)='W' then
update train
set no_W3=no_W3-1
where train_no=:old.train_no;
select min(pnr_no) into n from ticket where upper(status)='W';
update ticket
set status='B'
where pnr_no=n;
end if;end if;
end;
end;
```

```
SQL> create trigger del_seat
  2  before delete on ticket
  3  for each row
  4  begin
  5  declare
  6  n number;
  7  begin
  8  if :old.type_of_seat=1 then
  9  if upper(:old.status)='B' then
 10  update train
 11  set no_A1=no_A1+1,no_B1=no_B1-1
 12  where train_no=:old.train_no;
 13  elsif upper(:old.status)='W' then
 14  update train
 15  set no_W1=no_W1-1
 16  where train_no=:old.train_no;
 17  select min(pnr_no) into n from ticket where upper(status)='W';
 18  update ticket
 19  set status='B'
 20  where pnr_no=n;
 21  end if;
 22  elsif :old.type_of_seat=2 then
 23  if upper(:old.status)='B' then
 24  update train
 25  set no_A2=no_A2+1,no_B2=no_B2-1
 26  where train_no=:old.train_no;
 27  elsif upper(:old.status)='W' then
 28  update train
 29  set no_W2=no_W2-1
 30  where train_no=:old.train_no;
 31  select min(pnr_no) into n from ticket where upper(status)='W';
 32  update ticket
 33  set status='B'
 34  where pnr_no=n;
 35  end if;
 36  elsif :old.type_of_seat=3 then
 37  if upper(:old.status)='B' then
 38  update train
 39  set no_A3=no_A3+1,no_B3=no_B3-1
 40  where train_no=:old.train_no;
 41  elsif upper(:old.status)='W' then
```

```
 41  elsif upper(:old.status)='W' then
 42  update train
 43  set no_W3=no_W3-1
 44  where train_no=:old.train_no;
 45  select min(pnr_no) into n from ticket where upper(status)='W';
 46  update ticket
 47  set status='B'
 48  where pnr_no=n;
 49  end if;
 50  end if;
 51  end;
 52  end;
 53  /

Trigger created.
```

Trigger-4 To validate the mail id of user:

```
create trigger validate_mail_id
  before insert or update on user_info
   for each row
   begin
   if :new.mail_id not like '%@%.___' then
   raise_application_error(-20005,'Invalid Mail Id');
  end if;
  end;
```

```
SQL> create trigger validate_mail_id
  2  before insert or update on user_info
  3  for each row
  4  begin
  5  if :new.mail_id not like '%@%.___' then
  6  raise_application_error(-20005,'Invalid Mail Id');
  7  end if;
  8  end;
  9  /

Trigger created.
```

SEQUENCES

Sequence For User_ID:

```
create sequence user_user_id
increment by 1
start with 100
maxvalue 10000
nocycle;
```

```
SQL> create sequence user_user_id
  2  increment by 1
  3  start with 100
  4  maxvalue 10000
  5  nocycle;

Sequence created.
```

Sequence For Passenger_ID:

```
create sequence passenger_id
increment by 2
start with 23
maxvalue 6000
nocycle;
```

```
SQL> create sequence passenger_id
  2  increment by 2
  3  start with 23
  4  maxvalue 6000
  5  nocycle;

Sequence created.
```

Sequence For PNR:

```
create sequence pnr
increment by 10
start with 2350
maxvalue 56000
nocycle;
```

```
SQL> create sequence pnr
  2  increment by 10
  3  start with 2350
  4  maxvalue 56000
  5  nocycle;

Sequence created.
```

References

We took help from some sources while making this project. Reference to those sources:

1. https://www.geeksforgeeks.org/sql-tutorial/
2. https://www.geeksforgeeks.org/plsql-introduction/
3. https://www.tutorialspoint.com/sql/
4. https://stackoverflow.com/questions/19779483/pl-sql-ora-01422-exact-fetch-returns-more-than-requested-number-of-rows
5. Slides given on our course site.