RAILWAY MANAGEMENT SYSTEM

CSE-3110 : Database Systems Laboratory

FARHAN MIRAZ SHIHAB

ROLL : 2007064

Abstract

A brief overview of the project, highlighting the objective of designing a database for a railway management system to manage passenger details, train information, ticket bookings, and transactions.

Introduction

The railway management system is designed to automate and streamline the operations associated with managing railway transportation involving scheduling trains, managing passenger information, handling ticket bookings, and overseeing financial transactions related to these activities. The aim of this project is to develop a database that is not only functional but also scalable, flexible and secure, allowing railway operators to efficiently manage their operations . The database schema described in this report is optimized for performance .This report outlines the design of the database, including detailed descriptions of the tables and their relationships.

Project Objectives

The primary objective of this project is to create a flexible system so that further addition to this project does not create any complications in the future. Other key objectives are :

- **Improving data management :** The relational schema gives a way to handle the vast detailed data with high reliability and accuracy.

- **Enhancing working convenience :** The database stores data of all the passengers and the train scheduling also the transactions of the passengers in an organized manner so that these data can be accessed easily. Which will also help the passengers to know about their data and their preferred train's information.

- **Automate Scheduling and Ticketing**: Develop a system that automates the scheduling of trains and the booking of tickets, reducing manual input and minimizing errors.

- **Ensure Scalability**: Design the system to be scalable so that it can handle increasing amounts of data and user queries as the railway network expands.

Project Design

For designing this project I have used Oracle Database 11g Express Edition.

Passenger Table

- **Purpose**: Stores all relevant information about passengers.
- **Fields**:
  - passenger_ID: Unique identifier for each passenger.
  - first_name: Passenger's first name.
  - last_name: Passenger's last name.
  - gender: Passenger's gender.
  - age: Passenger's age.
  - contact: Unique contact information for each passenger.

Train_info table

- **Purpose**: Contains information about train schedules and destinations.
- **Fields**:
  - train_name: Unique name of the train.
  - arrival_time: Time when the train arrives.
  - departure_time: Time when the train departs.
  - destination: Final destination of the train.

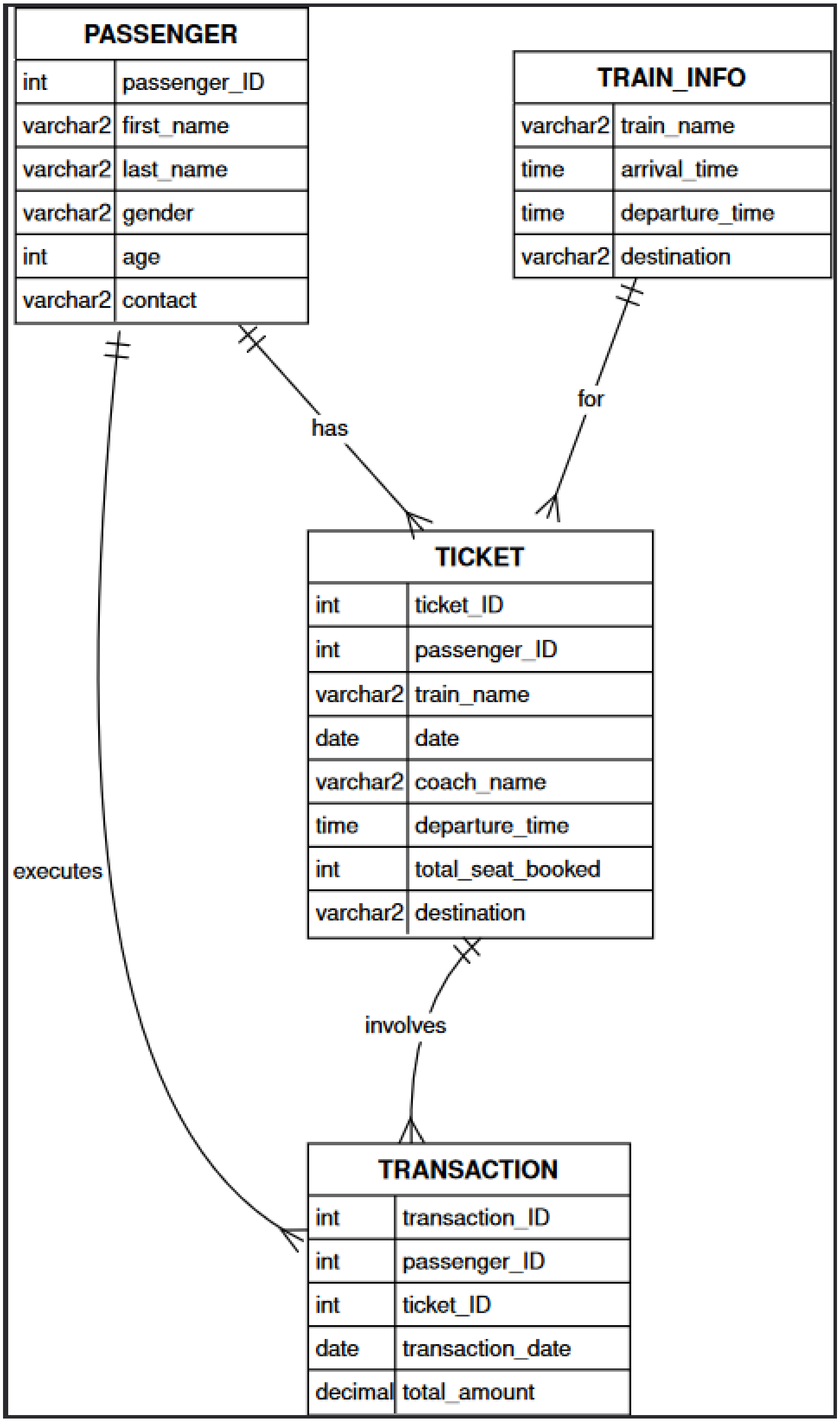
Ticket table

- **Purpose**: Manages ticket bookings for passengers.
- **Fields**:
  - ticket_ID: Unique identifier for each ticket.
  - passenger_ID: References Passenger.
  - train_name: References Train_info.
  - date: Date of travel.
  - coach_name: Type of the coach.
  - departure_time: Time of departure.
  - total_seat_booked: Number of seats booked.
  - destination: Ticket's destination.

Transaction

- **Purpose**: Records transactions related to ticket purchases.
- **Fields**:
  - transaction_ID: Unique identifier for each transaction.
  - passenger_ID: References Passenger.
  - ticket_ID: References Ticket.
  - transaction_date: Date of the transaction.
  - total_amount: Total amount of the transaction.

- ER Diagram



PASSENGER	
int	passenger_ID
varchar2	first_name
varchar2	last_name
varchar2	gender
int	age
varchar2	contact

TRAIN_INFO	
varchar2	train_name
time	arrival_time
time	departure_time
varchar2	destination

TICKET	
int	ticket_ID
int	passenger_ID
varchar2	train_name
date	date
varchar2	coach_name
time	departure_time
int	total_seat_booked
varchar2	destination

TRANSACTION	
int	transaction_ID
int	passenger_ID
int	ticket_ID
date	transaction_date
decimal	total_amount

Table creations :

```sql
create table Passenger (
    passenger_ID int primary key,
    first_name varchar2(100),
    last_name varchar2 (100),
    gender varchar2(20),
    age int,
    contact varchar2 (100) unique not null
);

create table Train_info (
    train_name varchar2(100) primary key,
    arrival_time TIME,
    departure_time TIME,
    destination varchar2 (100)
);

create table Ticket (
    ticket_ID int primary key,
    passenger_ID int,
    train_name varchar2 (100),
    date DATE,
    coach_name varchar2 (50),
    departure_time TIME,
    total_seat_booked INT,
    destination varchar2 (100),
    foreign key (passenger_ID) references Passenger(passenger_ID),
    foreign key (train_name) references Train_info(train_name)
);

create table Transaction (
    transaction_ID int primary key,
    passenger_ID int,
    ticket_ID int,
    transaction_date DATE,
    total_amount deciamal(10, 2),
    foreign key (passenger_ID) references Passenger(passenger_ID),
    foreign key (ticket_ID) references Ticket(ticket_ID)
);
```

SQL Queries

- ## Data retrieval

1. show details of trains headed to Chittagong

```
SELECT * FROM Train_info WHERE destination = 'Chittong';
```

2. get a list of all tickets where the coach is "First Class"

```
SELECT * FROM Ticket WHERE coach_name = 'First Class';
```

3. How do we find all male passengers?

```
SELECT * FROM Passenger WHERE gender = 'M';
```

4. What are the details of passengers who are under 20 years old?

```
SELECT * FROM Passenger WHERE age < 20;
```

- ## Data Manipulation/CRUD

1. change the age column in the Passenger table to allow for larger values

```
ALTER TABLE Passenger MODIFY (age NUMBER(3));
```

2. delete a record from the Passenger table where the passenger_ID is 10

```
DELETE FROM Passenger WHERE passenger_ID = 10;
```

3. combine records from two queries with distinct results using UNION and filter common records using INTERSECT

```
   SELECT first_name FROM Passenger WHERE age < 25 UNION SELECT
first_name FROM Passenger WHERE gender = 'F';
```

4. total number of tickets sold?

```
SELECT COUNT(*) FROM Ticket;
```

- PL/SQL

1. How can you declare a variable to store the highest number of seats booked on a ticket and print that value?

```
DECLARE
max_seats INT;
BEGIN
  SELECT MAX(total_seat_booked) INTO max_seats FROM Ticket;
 DBMS_OUTPUT.PUT_LINE('Maximum seats booked on a single ticket: ' ||
max_seats);
END;
```

2. Trigger a warning if a ticket booking exceeds 5 seats.

```
CREATE OR REPLACE TRIGGER Check_Seat_Booking BEFORE INSERT ON Ticket
FOR EACH ROW BEGIN IF :NEW.total_seat_booked > 5 THEN
RAISE_APPLICATION_ERROR(-20001, 'Cannot book more than 5 seats.');
END IF; END;
```

3. Loops through each record and prints train details.

```
BEGIN
  FOR train_rec IN (SELECT train_name, arrival_time, departure_time,
destination FROM Train_info)
  LOOP
    DBMS_OUTPUT.PUT_LINE('Train Name: ' || train_rec.train_name ||
                         ', Arrival Time: ' ||
TO_CHAR(train_rec.arrival_time, 'HH24:MI:SS') ||
                         ', Departure Time: ' ||
TO_CHAR(train_rec.departure_time, 'HH24:MI:SS') ||
                         ', Destination: ' || train_rec.destination);
  END LOOP;
END;
```

4. Print the value of a variable containing the number of passengers.

```
DECLARE num_passengers INT;
BEGIN SELECT COUNT(*) INTO num_passengers FROM Passenger;
DBMS_OUTPUT.PUT_LINE ('Number of passengers: || num_passengers');
END;
```

Discussion

After completing the project I have learned thoroughly about the SQL and the manipulation of data using different aspects of it. I have tried to complete the objectives of my projects by creating a database that is well-defined and flexible by giving relational aspects to each of the tables of the database for the easy accessibility of data. The transaction table is separated from other tables which only gets data in a one-way relation, so that passenger's data integrity is not compromised. The current database schema effectively supports the fundamental operations required by the railway management system. However, the need for more sophisticated designs and features might increase with the increase of usage of this system. For increased functionality it can be expanded to include more detailed data such as multiple transactions, tracking of trains, prebooking system, online booking, passenger friendly UI system etc.

Conclusion

By maintaining a flexible and scalable database design, the railway management system is well-defined to adapt to future upgrades and continue providing an essential service to its users with efficiency and reliability.

Github repo : https://github.com/Blackbird-M23/Railway_Management_System_DB