

Railway Management System

Youssef Mohamed, Mustafa Marzouk, Steven Hany

211777 213009 210991

Faculty of Computer Science

Group A

Dr. Amr El Maghraby

May 8, 2022

TABLE OF CONTENTS

1.0 INTRODUCTION…………………………………..………………………………………...2

1.1 Background……………………………….………….2

1.2 Problem Statement…………………………………...2

1.3 Objectives……………………………...……………..3

2.0 ER Diagrams & Conversions ………………………..………………………………………...3

2.1 Entity Relationship Diagram……...………………….4

2.2 Mapped Relational Model…………….……………...5

3.0 DDL, DML, & DQL ………………………..………………………….……………………...6

3.1 Data Definition Language (DDL)……...…………….6

3.2 Data Manipulation Language (DML) …….………...13

3.2 Data Query Language (DQL) …………….………...13

4.0 Implementation… ………………………..………………………….……………………...16

**1.0 Introduction**

**1.1 Background**

Rail transport is nowadays being widely used in many countries across the world. It is the type of transport in which people and goods are carried over long distances. Many people prefer railways over the road transport because of the increase in the number of accidents and huge traffic congestions. They are also less expensive compared to the other modes of transport. This is the best mode of transport for the public in the developing countries as they are very cheap. A railway management system must contain certain contributing actors to function properly. Those actors involve: passengers, employees, trains, stations, and tickets generated after transactions. All those actors play an important role in managing the time sensitive system to insure proper functions. A simple demonstration of the system would be: A passenger asks the employee to book a ticket with the desired timing, seats, and destination. The passenger then gives his/her information to be recorded and verified. The employee looks for an availability accordingly, by cross checking data in a backend server. If the available timing and seat is up to the passenger’s needs, then the transaction is made and the ticket is booked.

**1.2 Problem Statement**

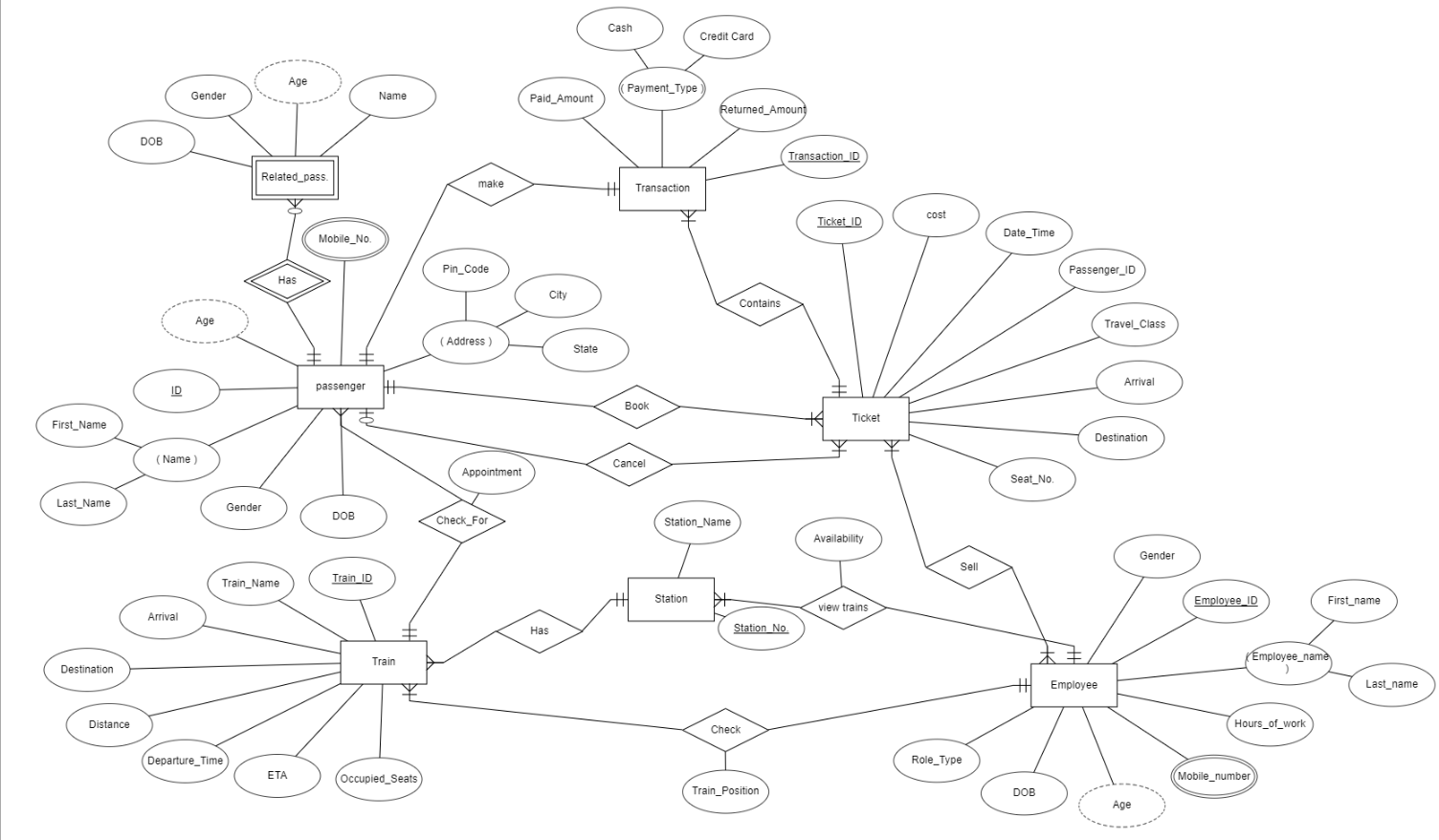
Railway management systems should apply fast and proper measures to provide transport services and prevent delay propagation to the rest of the network. Currently, predefined solutions called contingency plans are used to assist such time sensitive systems in dealing with disrupted traffic and overbooking problems, due to the use of outdated methods such as binder records. A more resilient automated system makes it less prone to system errors. Evidently, this makes it a less chance of resorting to the applications of such contingency plans that may or may not fulfil the system’s needs. Open Shared booking sessions must respond fast in terms of queries to the servers, to insure the are no simultaneous contradictions of data stored. Manipulating data carefully in such systems are highly important as it’s critical in nature to apply such practicality in a consistent bases with thousands of passengers commuting every day to their desired destinations.

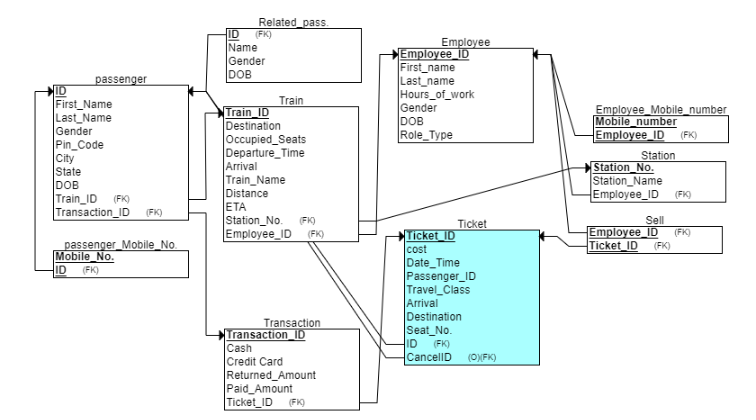
**1.3 Objectives**

Our main objective is to create a simple but effective railway management system. The main focus of our project is to integrate a hassle-free UI with a reliable and secure backend database for both employees operating on it and the customer/passengers involved. There are several notable core functions that are of paramount importance to include so the system may function as intended. Starting up, the backend database, A passenger must have their information stored securely on it as a reference for current and future bookings. This could aid in keeping track and identifying the same passengers traveling again in the future by earning their trust. A database must also contain information about the current employees working and their respective roles. Coming to the most important aspect, the database must contain the tickets assigned to each passenger alongside their transactions. Our goal here is to maximize the data flow of our system to keep up with every new change. Another function that should be integrated is the view of available trips sorted by set departure date, seat number, and destinations for both passengers and employees. This ensures faster data accessing times for booking purposes. All in all, the system should be capable of handling multiple processes at once in processing and archiving data.

**2.0 ER Diagram & Conversion**

A blueprint must be made to further clarify the entities, Attributes, and relationships the system consists of. The ER diagram and Relational model below shows the connections of each available entity with their respective attributes, and how their relationships coexist to make up a railway management system. Core Entities includes: Passengers, Employees, Trains, Stations, Tickets, and transactions made.

**2.1 Entity Relationship Diagram [ERDPlus]**

**2.2 Mapped Relational Mode [ERDPlus]**

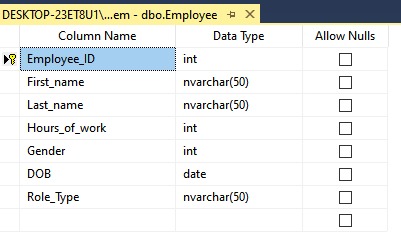
**3.0 ER Diagram & Conversion**

Below is the SQL code compiled by SQL Management Studio, the graphs are also displayed below for reference.

**3.1 Data Definition Language (DDL)**

CREATE TABLE Employee

(

 Employee\_ID INT NOT NULL,

First\_name INT NOT NULL,

Last\_name INT NOT NULL,

Hours\_of\_work INT NOT NULL,

Gender INT NOT NULL,

DOB INT NOT NULL,

Role\_Type INT NOT NULL,

PRIMARY KEY (Employee\_ID)

);

CREATE TABLE Employee\_Mobile\_number

(

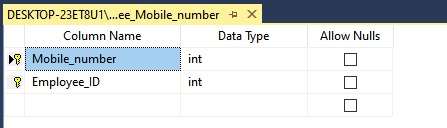
Mobile\_number INT NOT NULL,

Employee\_ID INT NOT NULL,

PRIMARY KEY (Mobile\_number, Employee\_ID),

FOREIGN KEY (Employee\_ID) REFERENCES Employee(Employee\_ID)

);



CREATE TABLE Station

(

Station\_Name INT NOT NULL,

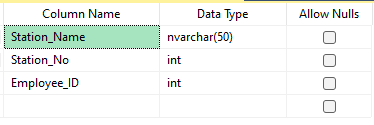
Station\_No INT NOT NULL,

Employee\_ID INT NOT NULL,

PRIMARY KEY (Station\_No),

FOREIGN KEY (Employee\_ID) REFERENCES Employee(Employee\_ID)

);



CREATE TABLE Train

(

Train\_ID INT NOT NULL,

Destination INT NOT NULL,

Occupied\_Seats INT NOT NULL,

Departure\_Time INT NOT NULL,

Arrival INT NOT NULL,

Train\_Name INT NOT NULL,

Distance INT NOT NULL,

ETA INT NOT NULL,

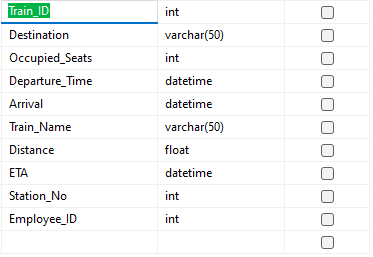
Station\_No INT NOT NULL,

Employee\_ID INT NOT NULL,

PRIMARY KEY (Train\_ID),

FOREIGN KEY (Station\_No) REFERENCES Station(Station\_No),

FOREIGN KEY (Employee\_ID) REFERENCES Employee(Employee\_ID) );



CREATE TABLE passenger

(

ID INT NOT NULL,

First\_Name INT NOT NULL,

Last\_Name INT NOT NULL,

Gender INT NOT NULL,

Pin\_Code INT NOT NULL,

City INT NOT NULL,

State INT NOT NULL,

DOB INT NOT NULL,

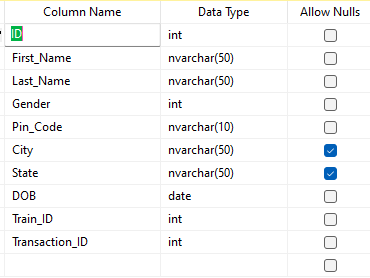
Train\_ID INT NOT NULL,

Transaction\_ID INT NOT NULL,

PRIMARY KEY (ID),

FOREIGN KEY (Train\_ID) REFERENCES Train(Train\_ID),

);



CREATE TABLE Ticket

(

Ticket\_ID INT NOT NULL,

cost INT NOT NULL,

Date\_Time INT NOT NULL,

Passenger\_ID INT NOT NULL,

Travel\_Class INT NOT NULL,

Arrival INT NOT NULL,

Destination INT NOT NULL,

Seat\_No INT NOT NULL,

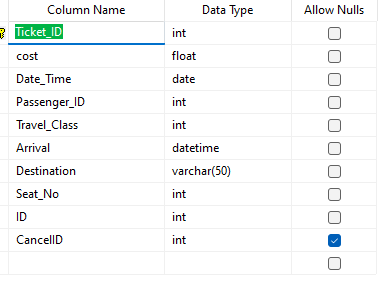
ID INT NOT NULL,

CancelID INT,

PRIMARY KEY (Ticket\_ID),

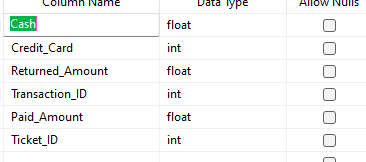
FOREIGN KEY (ID) REFERENCES passenger(ID),

FOREIGN KEY (CancelID) REFERENCES passenger(ID) );



CREATE TABLE Transact

(

 Cash INT NOT NULL,

Credit\_Card INT NOT NULL,

Returned\_Amount INT NOT NULL,

Transaction\_ID INT NOT NULL,

Paid\_Amount INT NOT NULL,

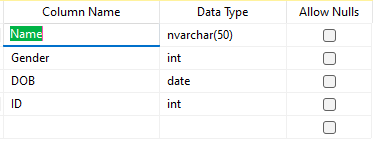
Ticket\_ID INT NOT NULL,

PRIMARY KEY (Transaction\_ID),

FOREIGN KEY (Ticket\_ID) REFERENCES Ticket(Ticket\_ID)

);

CREATE TABLE Related\_pass

(

Name INT NOT NULL,

Gender INT NOT NULL,

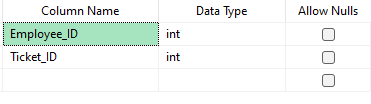
DOB INT NOT NULL,

ID INT NOT NULL,

PRIMARY KEY (ID),

FOREIGN KEY (ID) REFERENCES passenger(ID)

);

CREATE TABLE Sell

(

Employee\_ID INT NOT NULL,

Ticket\_ID INT NOT NULL,

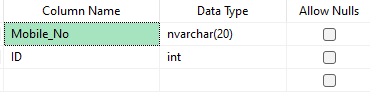
PRIMARY KEY (Employee\_ID, Ticket\_ID),

FOREIGN KEY (Employee\_ID) REFERENCES Employee(Employee\_ID),

FOREIGN KEY (Ticket\_ID) REFERENCES Ticket(Ticket\_ID)

);

CREATE TABLE passenger\_Mobile\_No

(

Mobile\_No INT NOT NULL,

ID INT NOT NULL,

PRIMARY KEY (Mobile\_No, ID),

FOREIGN KEY (ID) REFERENCES passenger(ID)

);

ALTER TABLE Passenger

Table

Description automatically generatedADD FOREIGN KEY (transact\_id) REFERENCES Transact(id);

**3.2 Data Query Language (DQL)**

Graphical user interface, text, application, Word

Description automatically generated

**3.3 Data Manipulation Language (DML)**

INSERT INTO Employee (

first\_name,

last\_name,

gender,

date\_of\_birth,

hours\_of\_work,

role\_type

)

VALUES

(

'steven',

'hany',

1,

'2008-11-11',

5,

'conductor'

),

(

'mustafa',

'marzouk',

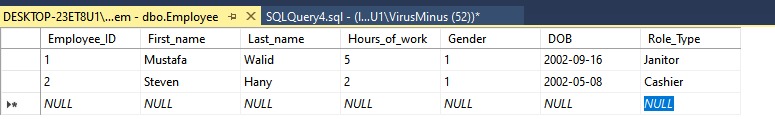
5,

'2008-11-11',

5,

'cashier'

);



**4.0 Implementation**

