WORKING WITH DATA

Marwan Elashry, ID 202201360

Topic: Railway Database Management System
Lecturer: Dr. Abeer Ali

Introduction

A railway system is a complex and difficult database to manage. A railway station has several trains departing and arriving throughout the day. Each train has a unique number, manufacturing date, number of seats, class, departure and arrival times will also be recorded. The drivers of each train are also recorded to keep track of them. In the database, passengers and their tickets will also be recorded along with the relationship between the tickets and the train.

Requirements Specification and Analysis

In a railway transportation database system, it is used to keep track of trains, stations, passengers, train drivers, and ticketing.

There are train companies that manufacture the trains operating. Each company could be manufacturing several trains.

TRAIN_COMPANY: Company_ID, Company_Name

All the stations that the trains depart and arrive from are also recorded in the database.

STATION: Station_ID, Station_Name, Location

A train driver could be controlling multiple trains and routes. We store the driver's first name, last name, ID number, and the trains that they control.

TRAIN DRIVER: First name, Last name, ID_Number, Trains

TRAIN: Train number, Company_ID, MDate, Class, Number_of_Seats, Station_ID_Depart, Station_ID_Arrive

The information of the passengers will also be recorded in the database, including their first name, last name, age, and ticket number.

PASSENGER: Passenger_ID, First_Name, Last_Name

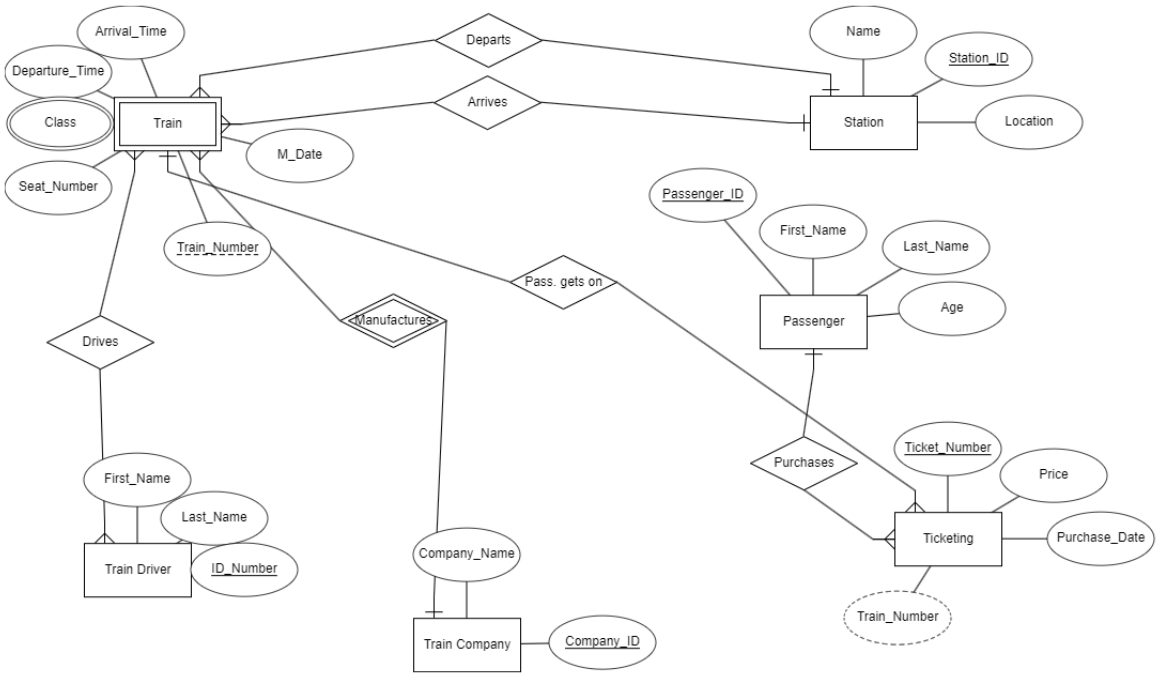
A ticket is created in the database when the passenger purchases a ticket, and all the details for their train is also recorded.

TICKETING: Ticket_Number, Purchase_Date, Price, Train_Number, Company_ID, Passenger_ID

Since there can be multiple drivers that operate multiple trains, there needs to be recorded data for the drivers and their trains as well as assigning new drivers new trains.

DRIVES: Train_Number, Company_ID, Driver_ID

ER Diagram



Complex Operations

1. Search for all the trains departing a specific station
2. Displays each train number and its manufacturing company in the column next to it
3. Deletes a ticket for a passenger using their ID and inserting a new ticket for the same ID.

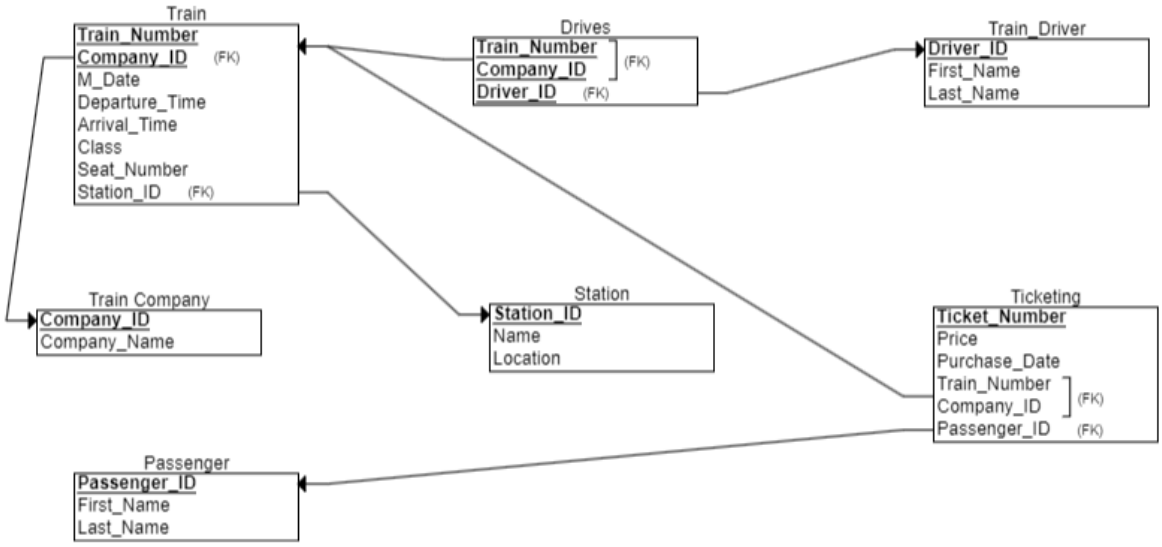
Aggregate Functions:

1. Displays each train number as well as the total number of passengers on each train.
2. Displays the maximum and minimum price of a train after being given the train number.
3. Displays the company id with the total amount of tickets sold as well as the total revenue.

NOT EXISTS Query:

Displays the stations where no train is departing from them.

Relational Data Model Schema



Normalization

1. **First Normal Form (1NF)**:

    - The database ensures atomicity by storing atomic values in each column.
    - There are no repeating groups of columns in any of the tables.
    - Column names are unique within each table.
    - Foreign key constraints are used to reference atomic values from other tables, ensuring adherence to atomicity.

2. **Second Normal Form (2NF)**:

    - All tables are already in 1NF.

- Non-key attributes in each table are fully functionally dependent on the entire primary key.
- There are no partial dependencies in any table, as each non-key attribute depends on the entire primary key.

3. **Third Normal Form (3NF)**:
   - All tables are already in 2NF.
   - No transitive dependencies exist in any table, ensuring that each non-key attribute depends only on the primary key and not on other non-key attributes.
4. **Boyce-Codd Normal Form (BCNF)**:
   - All tables are already in 3NF.
   - For each non-trivial functional dependency within the set of dependencies, the determinant (X) is a candidate key, ensuring compliance with the Boyce-Codd Normal Form (BCNF).

SQL Implementation

CREATE Statements:

```
CREATE TABLE TRAIN_COMPANY(
COMPANY_ID INT PRIMARY KEY,
COMPANY_NAME VARCHAR (255)
);

GO

CREATE TABLE STATION(
STATION_ID VARCHAR (255) PRIMARY KEY,
STATION_NAME VARCHAR (255),
LOCATION VARCHAR (255)
);

GO
CREATE TABLE TRAIN_DRIVER(
DRIVER_ID INT PRIMARY KEY,
FIRST_NAME VARCHAR (255),
LAST_NAME VARCHAR (255)
);
GO

CREATE TABLE TRAIN(
 TRAIN_NUMBER INT PRIMARY KEY,
 COMPANY_ID INT FOREIGN KEY REFERENCES TRAIN_COMPANY(COMPANY_ID),
```

```
 M_DATE DATE,
 CLASS INT,
 NUMBER_OF_SEATS INT,
 STATION_ID_DEPART VARCHAR (255) FOREIGN KEY REFERENCES STATION(STATION_ID),
 STATION_ID_ARRIVE VARCHAR (255) FOREIGN KEY REFERENCES STATION(STATION_ID),
);

GO

CREATE TABLE PASSENGER(
PASSENGER_ID INT PRIMARY KEY,
FIRST_NAME VARCHAR (255),
LAST_NAME VARCHAR (255)
);

GO

CREATE TABLE TICKETING(
TICKET_NUMBER INT PRIMARY KEY,
PURCHASE_DATE DATE,
PRICE INT,
TRAIN_NUMBER INT FOREIGN KEY REFERENCES TRAIN(TRAIN_NUMBER),
COMPANY_ID INT FOREIGN KEY REFERENCES TRAIN_COMPANY(COMPANY_ID),
PASSENGER_ID INT FOREIGN KEY REFERENCES PASSENGER(PASSENGER_ID)
);

GO


CREATE TABLE DRIVES (
    TRAIN_NUMBER INT,
    COMPANY_ID INT,
    DRIVER_ID INT,
    PRIMARY KEY (TRAIN_NUMBER, COMPANY_ID, DRIVER_ID),
    FOREIGN KEY (TRAIN_NUMBER) REFERENCES TRAIN(TRAIN_NUMBER),
    FOREIGN KEY (COMPANY_ID) REFERENCES TRAIN_COMPANY(COMPANY_ID),
    FOREIGN KEY (DRIVER_ID) REFERENCES TRAIN_DRIVER(DRIVER_ID)

);
```

INSERT Statements

```
INSERT INTO TRAIN_COMPANY VALUES
(1, 'ExpressRail'),
(2, 'TransitTrax'),
(3, 'MetroLink Logistics'),
(4, 'Global Transit Solutions'),
(5, 'SwiftTrans'),
(6, 'Railway Dynamics'),
(7, 'Raillink Express'),
(8, 'MetroTransit Solutions'),
(9, 'Speedy Rails'),
(10, 'ExpressLink Transport');



INSERT INTO STATION (STATION_ID, STATION_NAME, LOCATION) VALUES
    ('S001', 'King''s Cross Station', 'London, UK'),
```

```
    ('S002', 'Gare du Nord', 'Paris, France'),
    ('S003', 'Hauptbahnhof', 'Berlin, Germany'),
    ('S004', 'Termini Station', 'Rome, Italy'),
    ('S005', 'Estação do Oriente', 'Lisbon, Portugal'),
    ('S006', 'Sants Station', 'Barcelona, Spain'),
    ('S007', 'Central Station', 'Amsterdam, Netherlands'),
    ('S008', 'Keleti Station', 'Budapest, Hungary'),
    ('S009', 'Wien Hauptbahnhof', 'Vienna, Austria'),
    ('S010', 'Zürich Hauptbahnhof', 'Zurich, Switzerland'),
       ('S011', 'Grand Central Terminal', 'New York City, USA'),
       ('S012', 'Shinjuku Station', 'Tokyo, Japan'),
       ('S013', 'Waterloo Station', 'London, UK');

INSERT INTO TRAIN_DRIVER (DRIVER_ID, FIRST_NAME, LAST_NAME)
VALUES
    (42, 'John', 'Smith'),
    (32, 'Emily', 'Johnson'),
    (56, 'Michael', 'Williams'),
    (23, 'Daniel', 'Jones'),
    (93, 'Olivia', 'Miller'),
    (74, 'William', 'Davis'),
       (81, 'Sophia', 'Patel'),
       (38, 'Ava', 'Robinson');

INSERT INTO TRAIN (TRAIN_NUMBER, COMPANY_ID, M_DATE, CLASS, NUMBER_OF_SEATS,
STATION_ID_DEPART, STATION_ID_ARRIVE)
VALUES
    (1001, 1, '2024-04-08', 1, 120, 'S001', 'S003'),
    (1002, 2, '2024-04-08', 2, 150, 'S001', 'S004'),
    (1003, 3, '2024-04-08', 1, 100, 'S002', 'S005'),
    (1004, 4, '2024-04-08', 2, 130, 'S003', 'S006'),
    (1005, 5, '2024-04-08', 1, 110, 'S004', 'S007'),
    (1006, 6, '2024-04-08', 2, 140, 'S005', 'S008'),
    (1007, 7, '2024-04-08', 1, 120, 'S005', 'S009'),
    (1008, 8, '2024-04-08', 2, 150, 'S005', 'S010'),
    (1009, 9, '2024-04-08', 1, 200, 'S006', 'S009'),
    (1010, 10, '2024-04-08',2, 230,'S006', 'S007'),
    (1011, 1, '2024-04-08', 1, 185, 'S006', 'S009'),
    (1012, 2, '2024-04-09', 2, 190, 'S007', 'S010');

INSERT INTO PASSENGER (PASSENGER_ID, FIRST_NAME, LAST_NAME)
VALUES
    (101, 'John', 'Doe'),
    (102, 'Jane', 'Smith'),
    (103, 'Michael', 'Johnson'),
    (104, 'Emily', 'Williams'),
    (105, 'Daniel', 'Brown'),
    (106, 'Olivia', 'Jones'),
    (107, 'William', 'Miller'),
    (108, 'Sophia', 'Davis'),
    (109, 'Ethan', 'Garcia'),
    (110, 'Isabella', 'Martinez'),
    (111, 'James', 'Hernandez'),
    (112, 'Mia', 'Lopez'),
    (113, 'Alexander', 'Gonzalez'),
    (114, 'Charlotte', 'Wilson'),
    (115, 'Lucas', 'Anderson');
```

```sql
INSERT INTO TICKETING (TICKET_NUMBER, PURCHASE_DATE, PRICE, TRAIN_NUMBER, COMPANY_ID,
PASSENGER_ID)
VALUES
    (2001, '2023-05-10', 150, 1001, 1, 101),
    (2002, '2023-06-15', 200, 1002, 2, 101),
    (2003, '2023-07-20', 120, 1003, 3, 101),
    (2004, '2023-08-25', 180, 1004, 4, 102),
    (2005, '2023-09-30', 250, 1005, 5, 102),
    (2006, '2023-10-05', 100, 1006, 6, 103),
    (2007, '2023-11-10', 220, 1007, 7, 104),
    (2008, '2023-12-15', 180, 1008, 8, 105),
    (2009, '2024-01-20', 150, 1009, 9, 106),
    (2010, '2024-02-25', 200, 1010, 10, 107),
    (2011, '2024-03-30', 130, 1011, 1, 108),
    (2012, '2024-04-05', 170, 1012, 2, 109),
    (2013, '2024-05-10', 240, 1001, 1, 110),
    (2014, '2024-06-15', 190, 1002, 2, 110),
    (2015, '2024-07-20', 160, 1003, 3, 111),
    (2016, '2024-08-25', 220, 1004, 4, 112),
    (2017, '2024-09-30', 180, 1005, 5, 113),
    (2018, '2024-10-05', 150, 1006, 6, 114),
    (2019, '2024-11-10', 200, 1007, 7, 115),
    (2020, '2024-12-15', 130, 1008, 8, 115);


INSERT INTO DRIVES (TRAIN_NUMBER, COMPANY_ID, DRIVER_ID)
VALUES
    (1001, 1, 42),
    (1002, 2, 32),
    (1003, 3, 56),
    (1004, 4, 32),
    (1005, 5, 23),
    (1006, 6, 93),
    (1007, 7, 74),
    (1008, 8, 93),
    (1009, 9, 93),
    (1010, 10, 81),
    (1011, 1, 38),
    (1012, 2, 42);
```

SELECT Statements

```sql
SELECT * FROM TRAIN_COMPANY;
SELECT * FROM STATION;
SELECT * FROM TRAIN_DRIVER;
SELECT * FROM TRAIN;
SELECT * FROM PASSENGER;
SELECT * FROM TICKETING;
SELECT * FROM DRIVES;
```

Complex Operations

```sql
SELECT TRAIN_NUMBER
FROM TRAIN
WHERE STATION_ID_DEPART = 'S005';
```

```sql
SELECT
    T.TRAIN_NUMBER,
       TC.COMPANY_NAME

FROM
    TRAIN_COMPANY TC
INNER JOIN
    TRAIN T ON TC.COMPANY_ID = T.COMPANY_ID;


DELETE FROM TICKETING
WHERE PASSENGER_ID = 116;

INSERT INTO TICKETING (TICKET_NUMBER, PURCHASE_DATE, PRICE, TRAIN_NUMBER, COMPANY_ID,
PASSENGER_ID)

VALUES (2026, '2024-01-22', 210, 1004, 4, 116);
```

Aggregate Functions

```sql
SELECT TRAIN_NUMBER, COUNT(*) AS TOTAL_PASSENGERS
FROM TICKETING
GROUP BY TRAIN_NUMBER;


SELECT
    MAX(PRICE) AS MAX_PRICE,
    MIN(PRICE) AS MIN_PRICE
FROM
    TICKETING
WHERE
    TRAIN_NUMBER = 1001;


SELECT TC.COMPANY_NAME, COUNT(*) AS NUMBER_OF_TICKETS_SOLD, SUM(T.PRICE) AS TOTAL_REVENUE
FROM TICKETING T
INNER JOIN TRAIN_COMPANY TC ON T.COMPANY_ID = TC.COMPANY_ID
WHERE T.COMPANY_ID = 1

GROUP BY TC.COMPANY_NAME;
```

NOT EXISTS Query

```sql
SELECT STATION_ID, STATION_NAME, LOCATION
FROM STATION S
WHERE NOT EXISTS (
    SELECT *
    FROM TRAIN T
    WHERE S.STATION_ID = T.STATION_ID_DEPART
);
```

Error Handling:

The code uses a few error handling methods such as try-catch blocks, which catch exceptions that could occur during database operations. It also uses message boxes to show errors to the user in case of exceptions.

Security:

The entire code is protected against SQL injection through the use of parametrized queries. Parameterized queries prevents malicious users from injecting SQL code into input fields because the input is treated as data rather than executable SQL code.