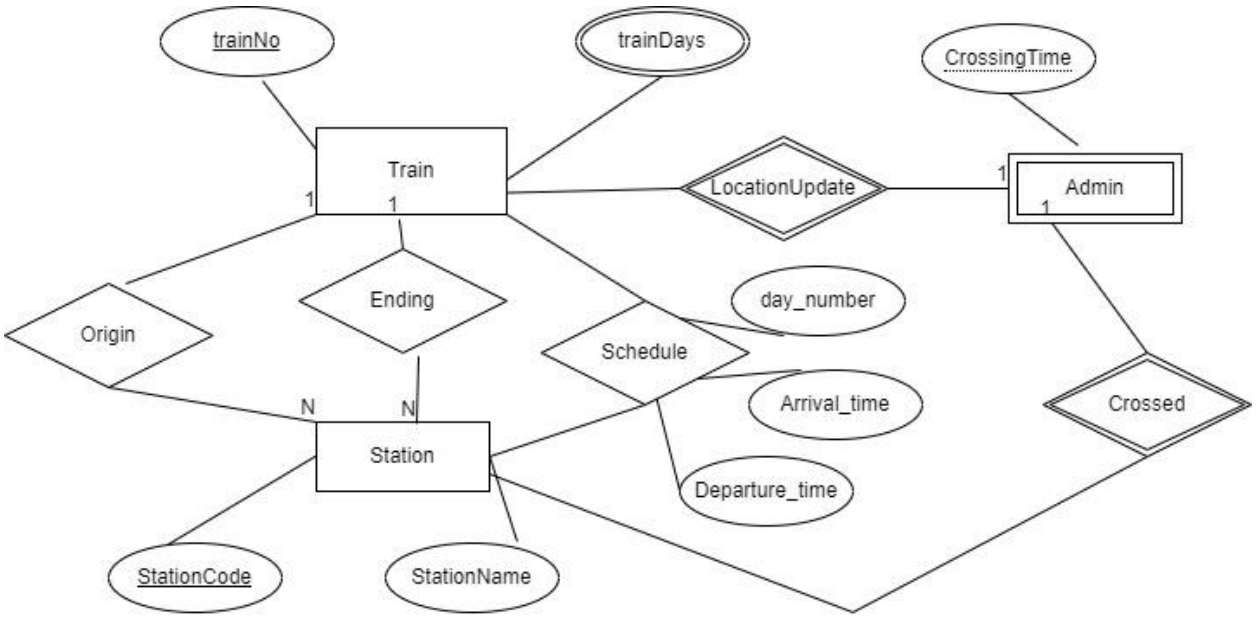
LIVE TRAIN RUNNING STATUS

MRIDUL BHATIA-2019B3A70410P
AVIRAL OMAR-2019B3A70411P

SYSTEM REQUIREMENTS

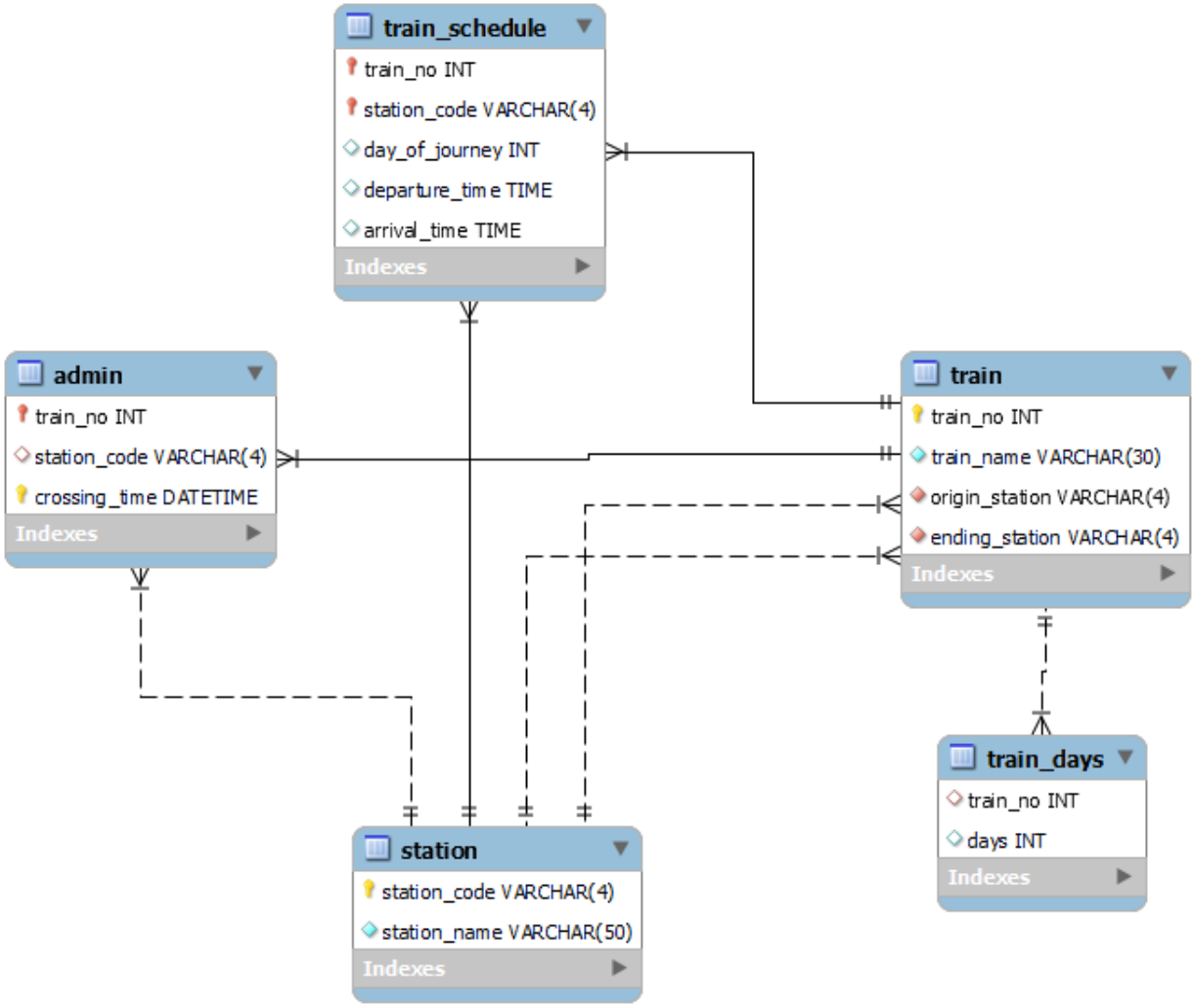
The MySQL version should be at least 8.0. This project is made entirely on MySQL.

ER DIAGRAM



SCHEMA

The following schema is constructed after constructing the required tables from the above ER diagram and then merging/dividing a certain number of tables for normalization purposes to avoid the redundancy. So in total 5 tables are constructed as shown below.



LIST OF TABLES REQUIRED WITH DESCRIPTION

1. **Train(train_no,train_name,origin_station,ending_station)**--This table is used to store the details of the train.
    a. **Train_no** is the train number which serves as the **primary key** for this table because for a given train and a route, the train number is unique.It has an integer data type. A **check constraint** is used to ensure that the train number is a 5-digit number.
    b. **Origin_station** is the station code from where the station starts.It has a varchar data type There can be one originating station for multiple trains. This column references the station code of the Station table so it is a foreign key.**On delete cascade** constraint is used on this column since if a station is removed, the train

will not be originating from that particular station.It also has a **not null** constraint because if a train number exists then the train is bound to have a starting station.

c. **Ending_station** is the station at which the station ends it's journey.It has a varchar data type.There can be multiple trains having the same ending station.This column also references the station code of the Station table so it is a foreign key.**On delete cascade** constraint is used for this column also.It also has a **not null** constraint because if a train number exists then the train is bound to have a destination station.

d. The **train_name** contains the name of the corresponding train. It has a varchar data type. It cannot be a primary key because for a given train originating station and destination station can be different. For example, if a train X having originating station A and ending station B will travel from A to B on a given day. But on some other day, when it travels from station B to station A, the originating station and destination station will be different, but the train name remains the same. It also has a **not null** constraint because if a train number exists then the train is bound to have a name.

e. The table is in **3NF** because trainNo->trainName, origin_station and ending_station. As these are the only functional dependencies possible and in each of them the left hand side is the superkey , the table is in 3NF. It was already in 2NF since every non prime attribute is dependent on the primary key.

2. **Station(stationCode, stationName)--** This table is used to store the details of a particular station.

   a. The **stationCode** column stores the particular code of the station. For example, NDLS for New Delhi station.It has a varchar data type. As for every station the code is unique and determines the name of the station as well, it is the **primary key** for this table. It also has a **check constraint** on it which ensures that the station code is at least 2 characters long.

   b. The **stationName** column stores the name of that particular station.It has a varchar data type A maximum character length of 50 is taken.

   c. The table is in **3NF.** stationCode->stationName and stationName->stationCode. Both of these dependencies have superkeys on the left side. Hence in 3NF.

3. **Train_Days(trainNo, days)**-- It is used to store the days of the week on which the train travels.

a. The **trainNo** is the **foreign key** which refers to the trainNo of the train table. It is also a **primary key** of the table. **On delete cascade** constraint is used on it because if the trainNo is removed from the train table, the corresponding record in this table should also be deleted.
b. The **days** attribute stores the day number on which the train travels. It has int data type. So if a train travels on Sunday, 1 is stored in the column. For Monday, 2 is written and likewise for every other day of the week. Sunday-1, Monday-2, Tuesday-3…….Saturday-7. So a **check constraint** is there on the column in order to ensure that inputted value lies in the range of 1-7.
c. Since each train can have multiple values of days, it is in **1NF.** For each train number there can be multiple values of days. Hence there is no functional dependency but the values are atomic. Hence 1NF.

4. **Train_schedule(train_no, station_code, day_of_journey, departure_time, arrival_time)**-- This table is used to store the route of a particular train. **Primary key of this table is (train_no, station_code)**
    a. The **train_no** is the **foreign key** referencing the train table.
    b. The **station_code** is the **foreign key** referencing the station table.
    c. The **day_of_journey** stores the label of the day (00:00-24:00) on which the train is traveling.It is of int data type. For example, if a train crosses station X on Monday and the train has started on Monday itself and crosses station Y on Tuesday, then the corresponding entry for the day_of_journey would be 1 for station X and 2 for station Y. It has a **check constraint** which ensures that the value is not greater than 3.
    d. The **departure_time** stores the time at which the station departs from the station. In case the station is the destination station of that train, the departure_time holds **null** value. It has the time data type.
    e. The **arrival_time** stores the time at which the station departs from the station. In case the station is the originating station of that train, the arrival_time holds **null** value. It has the time data type.
    f. The table is in **3NF.**
    (train_no,station_code)->departure_time,(train_no,station_code)->arrival_time. These are the only functional dependencies. As the left hand side of each functional dependency is super key and the table is already in 2NF, it is in 3NF.

5. **Admin(train_no, station_code, crossing_time)**-- This table is used by the administrator to feed the real time data. **(train_no,crossing_time) is the primary key**
    a. The **train_no** is the **foreign key** referencing the train table.
    b. The **station_code** is the **foreign key** referencing the station table.
    c. The **crossing_time** stores the time at which a train crosses a particular station. It's data type is **Datetime**.
    d. The table is in **3NF.** (train_no, crossing_time)->station_code. Only one functional dependency hence in 3NF.

PROCEDURES

1. **extractDays(in trainNo)**-- This procedure is used to extract the day numbers from the train_days table by taking the train number as the input. In order to do so, a temporary table is created called temp_train_days. This table contains the day numbers on which the inputted train travels.
2. **adminUpdate(trainNo, stationCode, crossTime)**-- This function is used by the admin user to update the admin table. It is used to update the table in real time
3. **trainBetweenStations(startStation,endStation)-**- This function is used to output the trains which are going from startStation to endStation. The logic used here is that a cross product of train_schedule table is done with itself and those values of trains are fetched in which for startStation , the departure time < the arrival time for endingStation or the train is passing from startStation on the day previous to the day on which the train is crossing endingStation.
4. **trainsBetweenStationsOnDate(startStation,endStation,inputDate)**--This function enables us to return the trains which are there between startStation and endStation on the given input date.In this function, the checkDayofJourney function is called.

FUNCTIONS

1. **getArrivalTime(trainNo, stationCode)**-- This function is created so as to get the arrival time for a given train number and station code. This function comes handy in other procedures and functions.
2. **getDepartureTime(trainNo, stationCode)**--This function provides the departure_time for a given train number and station code. This function comes handy in other procedures and functions.

3. **getCrossingTime(trainNo, stationCode)**-- This function provides the crossing time for a given train number and station code. This function is also used in other procedures and functions.
4. **getJourneyStartTime(trainNo)**-- This function returns the time at which the journey starts i.e. the time at which the train with the inputted train number leaves the originating station.
5. **getOriginStation(trainNo)**-- This function returns the originating station for the given train number. This function can be used instead of writing queries at multiple places. This saves time.
6. **getDestinationStation(trainNo)**--This function returns the destination station for the given train number.
7. **getLastStationCrossed(trainNo)**-- This function makes use of **SELECT TOP** . For the inputted trainNo we get the list of crossing times from the **admin table**. Then we order it by the crossing times in the descending order and then by using SELECT TOP we extract the top row from which the last crossed station is extracted.
8. **getTimeDelay(crossing_time,departure_time)**-- This is used to calculate the time by which the train is delayed if it is delayed. The simple logic used here is that if the crossing_time>departure_time, then report delay= crossing_time-departure_time else the train is on time.
9. **checkTrainExists(trainNo)**-- It is used to check if the train number inputted really exists or not.
10. **getTrainStatus(trainNo)**-- It is used to display the status of the inputted train. The general methodology followed here is that for the inputted train number it is checked whether the train exists or not using the checkTrainExists() function. Then it is checked whether it travels on the given day or not using the extractDays() function. Then using getLastStationCrossed(), getDestinationStation(), getDepartureTime(),getJourneyStartTime() etc. functions, it is checked if the journey has been completed or not. If the journey is not completed then it is checked for the delay using the getTimeDelay()function. The appropriate messages and details are displayed accordingly.
11. **checkDayofJourney(inputDate,train_no,startStation)**-- This function is used to check if the train with the train_no is travelling on the input date or not. Here, the extractDays() function is used.