**LIVE TRAIN RUNNING STATUS (DBS\_PR\_14)**

Ayush Agarwal (2019B4A70652P)

Jayesh Singh (2018A8PS0468P)

**Table of Contents**

1. System requirement specification (SRS)
   1. Database and SQL files
   2. Scope of the Project
   3. Model Assumptions

1. System Modeling

2.1 ER Diagrams

2.2 Schema Design

2.3 Data Normalization and Design

2.4 List of tables required

* 1. Additional components (Procedures, Transactions, etc)

3. Environment Setup

**\*3.1** How to setup the running environment to code

3.2 Source of data collection from Data Scraping

3.3 Check SQL and Node versions.

3.4 Contribution of each team member

3.5 Video Link

**1.System Requirement Specifications (SRS)**

* 1. **Database and SQL Files**

The given project aims to capture the main ideas of handling a train running status system. The following documentation serves to provide sufficient guidelines to follow while testing and evaluating the project. The attached code consists of a single SQL file containing all the required queries, data definition, procedures, functions etc. The order in which the components of the SQL file has been kept is the same as mentioned in the project guidelines of the course **CS F212 Database Systems and Concepts, Fall Semester 2022**.

* 1. **Scope of the Project**

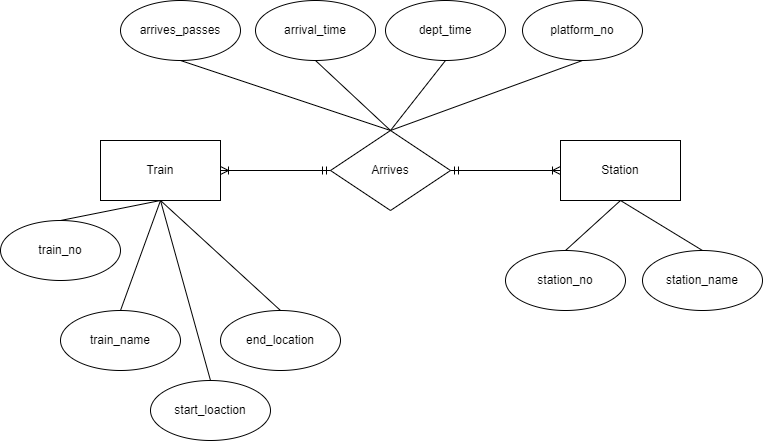
The scope of the project is to simply capture the basic requirements of a real-life simulation of an online movie ticket reservation system. The requirements that the following system fulfills is that of storing the important data in a non-redundant manner. The intricacies of the database such as normalization and schemas has been discussed ahead in the next section. Since an online live running status system is accessible to multiple users, this necessitates the use of transactions to take care of consistency of the database. Hence all the queries as per the requirements and intuition derived from real life scenarios have been handled using a suitable mode that is either concurrently handled or not. The system that we have designed is also compatible with the frontend framework and has been expanded according to the needs of the user and could be expanded further.

* 1. **Model Assumptions**

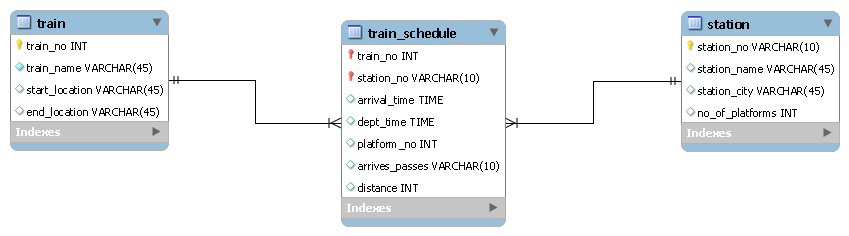
For the given model we did not add the feature where the admin or the Indian Government can add new trains and stations. It has been assumed that the sql only serves the purpose of the client who wants to query the running status of trains and the whereabouts of different stations.

**2. System Modelling**

**2.1 ER Diagram**

****

**2.2 Schema Design**



**2.3 Data Normalization and Design**

As mentioned earlier in the documentation, this project serves to fulfill the basic requirements of a train running live status database system. Coming to the data model assumptions first, we have a **train**  table consisting of the necessary details to distinguish a train entity. Followed by that we have a **station** table that stores all the train railway stations present in India , a relation between them called **train\_schedule** database that captures all the trains and their respective arrival and departure time in a station. Every **train\_schedule** has multiple **train** data as a train travels multiple stations and every station has multiple trains arriving at its stop. So the relation train\_schedule is many-many relation. Both trains and stations are strong entities and have their own primary keys respectively. Hence, even the relation train\_schedule is a strong-entity type relation. Also the data says that there is total participation of data.

**2.3 List of Tables required**

Total 3 tables are required, since we cannot reduce it further because

the relationship is M:N along with total participation.

Table1: train(train\_no, train\_name, start\_location, end\_location)

Table2: station (station\_no, station\_name,station\_city, no\_of\_platforms)

Table3: train\_schedule(train\_no, station\_no,arrival\_time,dept\_time,platform\_no,arrives\_passes, distance)

**2.4 List of Views used**

**-- -----------------------------------------------------**

**-- View `mydb`.`station\_schedule`** : Used to create a view for a particular station which contains all the trains that will arrive at that particular station with its time.

**-- -----------------------------------------------------**

**-- -----------------------------------------------------**

**-- View `mydb`.`passes\_by` :**  Used to find all the stations that a train passes through and not stops at in its journey.

**-- -----------------------------------------------------**

**-- -----------------------------------------------------**

**-- View `mydb`.`stops\_at` :** Used to find all the stations that a train stops at and not passes through in its journey.

**-- -----------------------------------------------------**

**-- -----------------------------------------------------**

**-- View `mydb`.`train\_timeline` :** Used to create a view for a particular train determining all the stations and timeline of a particular train.

**-- -----------------------------------------------------**

**2.5 Additional components**

Procedures used are

-- -----------------------------------------------------

-- procedure ETA: To find the expected arrival time of a train from a particular station.

-- -----------------------------------------------------

-----------------------------------------------------

-- procedure ETD : To find the expected departure time of a train from a particular station.

-- -----------------------------------------------------

-- -----------------------------------------------------

-- procedure available\_trains: To find all possible trains that stops between the two given stations.

-- -----------------------------------------------------

-- -----------------------------------------------------

-- procedure all\_train\_stops: Used to create a view for a particular train passed as parameter determining all the stations and timeline of a particular train.

-- -----------------------------------------------------

-- -----------------------------------------------------

-- procedure station\_schedule: Used to create a view for a particular station passed as parameter which contains all the trains that will arrive at that particular station with its time.

-- -----------------------------------------------------

-- -----------------------------------------------------

-- procedure routes : To find all possible ways of connection between station A and station B.

-- -----------------------------------------------------

-- -----------------------------------------------------

-- procedure train\_status : To fetch the current System Time and estimate where the train is actually present.

-- -----------------------------------------------------

Transactions are used for Handling concurrency with commit statements at the end of every procedures.

As mentioned earlier, there is no insertion or deletion or updation operation in this live running station project, therefore the use of triggers become obsolete.

Also the concurrency is handled on its own as there are no write operations involved in the transaction. Therefore multiple users can query the data at the same time without any delay.

**3. Environment Setup**

**\*3.1 How to setup the running environment to code**

Website Run following commands in root folder after pulling the repository from git:

npm install

node server

Then open localhost/

**SQL: Backend**

**CODE LINK:** [**https://github.com/particlesmoke/train\_database**](https://github.com/particlesmoke/train_database)

**3.2 Source of data collection from Data Scraping**

Data was collected from [trainman.in](https://www.trainman.in/).

It was scraped using a browser extension, [DataMiner](https://dataminer.io/)

**3.3 Check SQL and Node versions.**

My SQL workbench version: 8.0.28 build (Community Edition)

**3.4 Contribution of each team member**

Ayush : Formation of 3 Tables , Formation of Procedures and transactions, Documentation, Debugging.

Jayesh: Formation of Views, Collection and scraping of data, ER Model Specifications, Frontend Development, Video Editing.

**3.5 Video Link**

<https://drive.google.com/drive/folders/1YpUdIXNk3ewIuDtweyvybV1Vcywft79S?usp=sharing>