# Railroad Database

**Sourav Dhar, Atil Goker, Omar Muhammad, Walter Pompa**

**Group Name: Invincible ϖ**

**CSC 4710: Database Systems**

**Rafal Angryk**

# 0. Table of Contents

**0. Table of Contents** **2**

**1. System Requirements** **3**

**2. Contextual Data Flow Diagram** **6**

**3. ER Model** **7**

**4. Normalized DB Model** **8**

**5. Comparison Table** **10**

**6. DB Model** **12**

**7. Data Dictionary** **13**

**8. SQL** **18**

**9. Time Logs** **27**

**10. Appendix** **28**

# 1. System Requirements

## 1.1. Client Requirements

1. TRAIN
    1. Train MUST have a unique ID.
    2. Train MUST have a model.
    3. Train MUST have a name.
    4. Train MAY have zero or more TRIPs.
    5. Train MAY have zero or more ROUTEs.
    6. Train MUST BE either a PASSENGER_TRAIN or a FREIGHT_TRAIN.

2. PASSENGER_TRAIN
    1. Passenger Train MUST have a TRAIN ID.
    2. Passenger Train MUST have a number of seats.
    3. Passenger Train MAY have zero or more TICKETs.

3. PASSENGER
    1. Passenger MUST have a unique ID.
    2. Passenger MUST have a first name.
    3. Passenger MUST have a last name.
    4. Passenger MUST have a birthdate.
    5. Passenger MUST have an age. [derived attribute]
    6. Passenger MAY have no more than one USER_ACCOUNT.
    7. Passenger MAY have zero or more TICKETs.
    8. Passenger MAY BE a SPECIAL_REQUIREMENT_PASSENGER.

4. TICKET
    1. Ticket MUST have a unique ID.
    2. Ticket MUST have a PASSENGER.
    3. Ticket MUST have a PASSENGER_TRAIN.
    4. Ticket MUST have a TRIP.
    5. Ticket MUST have a seat number.
    6. Ticket MUST have a ticket class type [derived attribute].

5. SPECIAL_REQUIREMENT_PASSENGER
    1. Special Requirement Passenger MUST have a PASSENGER ID.
    2. Special Requirement Passenger MUST have a boolean disability flag.
    3. Special Requirement Passenger MUST have a boolean accompanied travel flag.

6. USER_ACCOUNT
    1. User Account MUST have a unique ID.
    2. User Account MUST have one or more PASSENGERs.
    3. User Account MUST have a password.
    4. User Account MUST have a first name.
    5. User Account MUST have a last name.
    6. User Account MUST have a birthdate.
    7. User Account MUST have a gender.
    8. User Account MUST have an age [derived attribute].
    9. User Account MUST have an e-mail address.
    10. User Account MUST have an address street name.
    11. User Account MUST have an address building number.
    12. User Account MAY have an address apartment number.
    13. User Account MUST have an address zip code.
    14. User Account MAY have zero or more SECURITY_INFO.

7. ADDRESS
    1. Address MUST have a zip code.
    2. Address MUST have a city.
    3. Address MUST have a state.
    4. Address MUST have a country.

8. SECURITY_INFO
    1. Security Info MUST have a unique ID.
    2. Security Info MUST have a USER_ACCOUNT.
    3. Security Info MUST have a security question.
    4. Security Info MUST have an answer.

9. FREIGHT_TRAIN
    1. Freight Train MUST have a TRAIN ID.
    2. Freight Train MAY have zero or more CARGO.

10. CARGO
    1. Cargo MUST have a unique ID.
    2. Cargo MUST have a FREIGHT_TRAIN.
    3. Cargo MUST have a CARGO_OWNER.
    4. Cargo MUST have a TRIP.
    5. Cargo MUST have a weight.
    6. Cargo MUST have a special handling requirement boolean flag.

11. CARGO_OWNER
    1. Cargo Owner MUST have a unique ID.
    2. Cargo Owner MUST have a company name.
    3. Cargo Owner MAY have zero or more CARGO.

12. TRIP
    1. Trip MUST have an ID.
    2. Trip MUST have a TRAIN.
    3. Trip MUST have an origin STATION.
    4. Trip MUST have a destination STATION.

    5. Trip MUST have a departure time.
    6. Trip MUST have an arrival time.
    7. Trip MUST have a CREW.

13. CREW
    1. Crew MUST have a TRIP ID.
    2. Crew MUST have a train CONDUCTOR.
    3. Crew MUST have a train ENGINEER.
    4. Crew MAY have zero or more passenger ATTENDANTs.
    5. Crew MAY have a kitchen COOK.
    6. Crew MUST have a SECURITY_OFFICER.

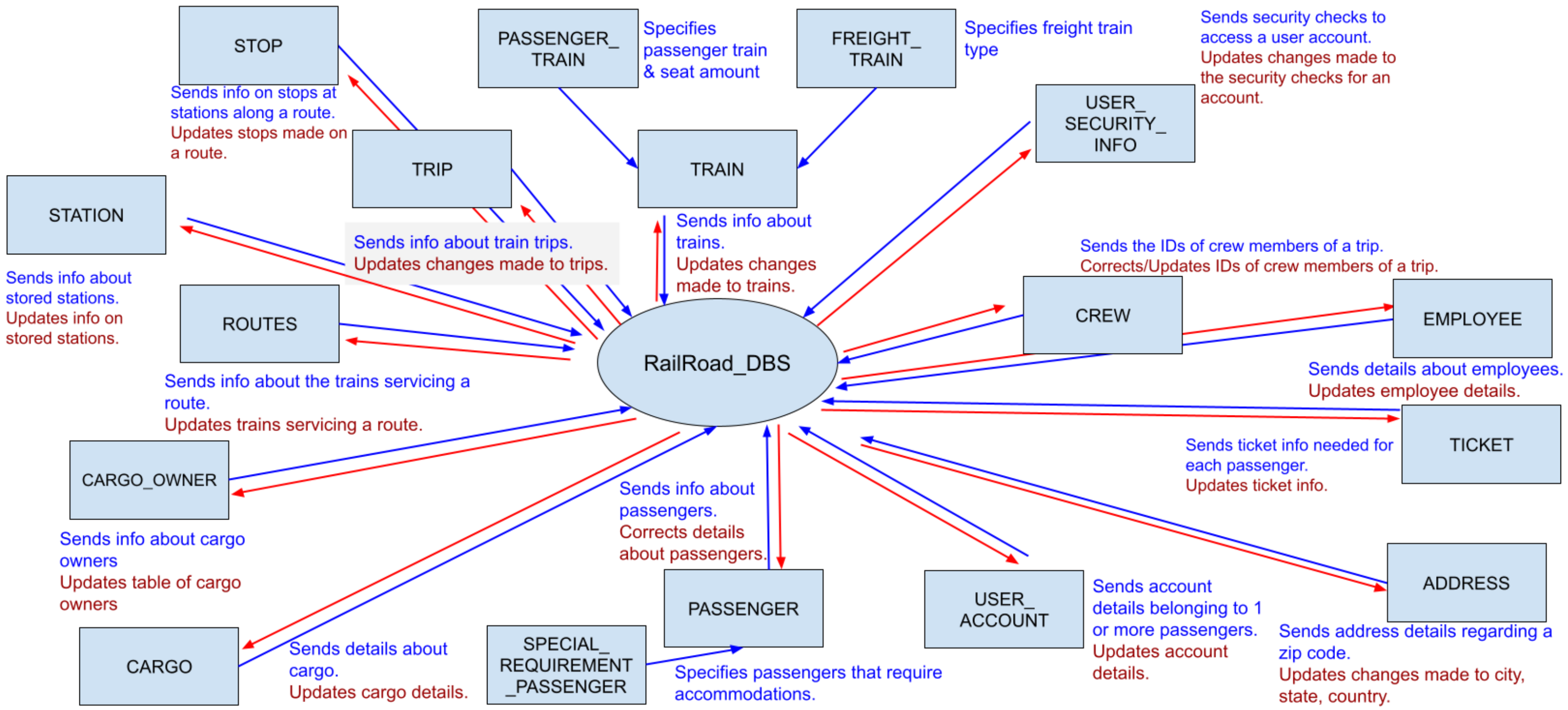
14. EMPLOYEE
    1. Employee MUST have a unique ID.
    2. Employee MUST have a first name.
    3. Employee MUST have a last name.
    4. Employee MUST have a social security number.
    5. Employee MAY have zero or more CREWs.
    6. Employee MUST be one of CONDUCTOR, ENGINEER, ATTENDANT, COOK, or SECURITY_OFFICER.

15. STATION
    1. Station MUST have a unique ID.
    2. Station MUST have a name.
    3. Station MUST have a zip code.
    4. Station MAY have zero or more connected stations.
    5. Station MAY have zero or more TICKETs.
    6. Station MAY have zero or more TRIPs.

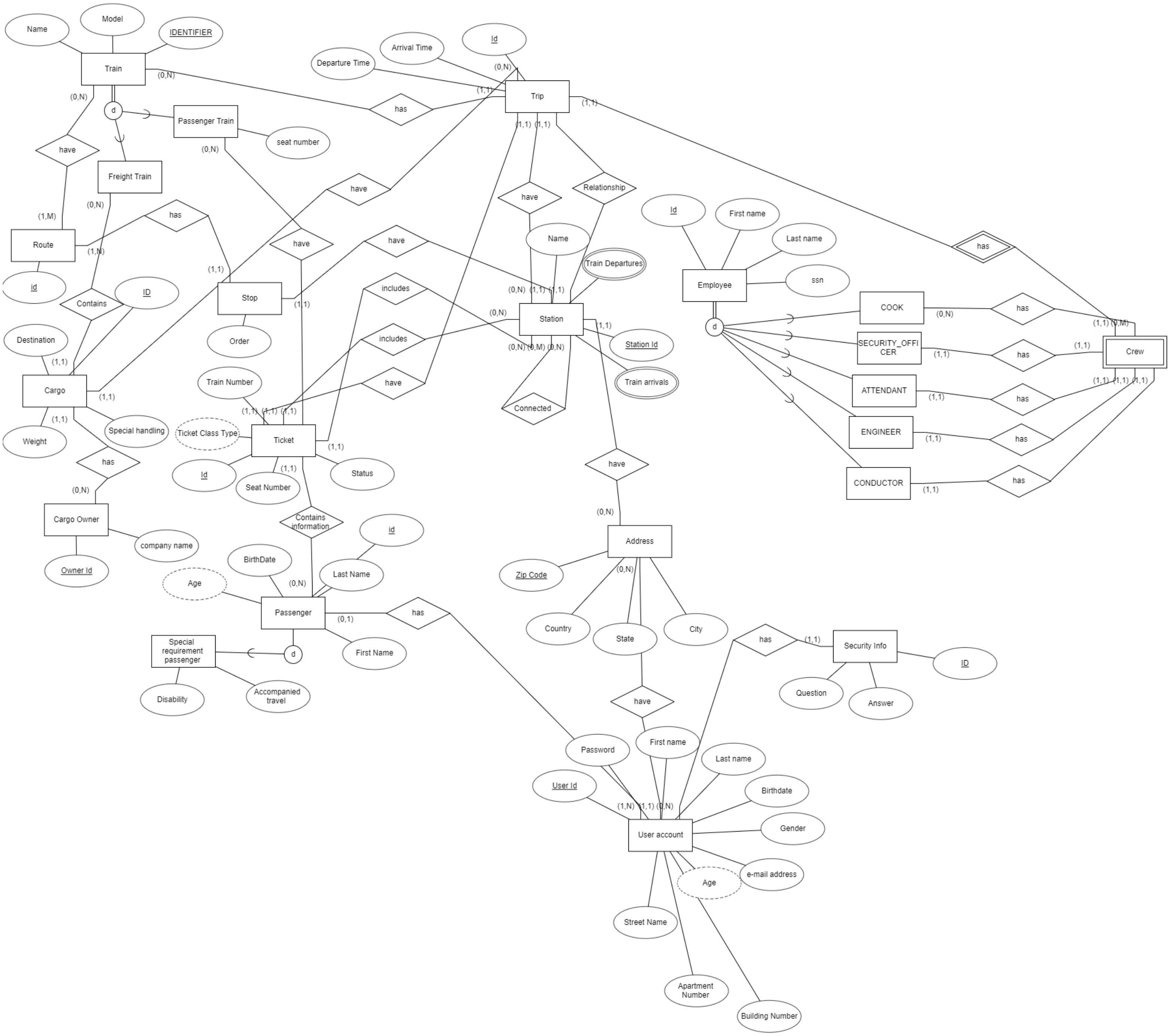
16. ROUTE
    1. Route MUST have a unique ID.
    2. Route MUST have one or more STOPs (station).
    3. Route MUST have one or more TRAINs.

17. STOP
    1. Stop MUST have a ROUTE ID.
    2. Stop MUST have a STATION ID.
    3. Stop MUST have an order.

# 2. Contextual Data Flow Diagram


STOP
PASSENGER_ TRAIN
Specifies passenger train & seat amount
FREIGHT_ TRAIN
Specifies freight train type
Sends security checks to access a user account.
Updates changes made to the security checks for an account.
Sends info on stops at stations along a route.
Updates stops made on a route.
USER_ SECURITY_ INFO
TRIP
TRAIN
STATION
Sends info about train trips.
Updates changes made to trips.
Sends info about trains.
Updates changes made to trains.
Sends the IDs of crew members of a trip.
Corrects/Updates IDs of crew members of a trip.
Sends info about stored stations.
Updates info on stored stations.
ROUTES
CREW
EMPLOYEE
RailRoad_DBS
Sends details about employees.
Updates employee details.
Sends info about the trains servicing a route.
Updates trains servicing a route.
TICKET
CARGO_OWNER
Sends ticket info needed for each passenger.
Updates ticket info.
Sends info about passengers.
Corrects details about passengers.
Sends info about cargo owners
Updates table of cargo owners
ADDRESS
PASSENGER
USER_ ACCOUNT
Sends account details belonging to 1 or more passengers.
Updates account details.
Sends address details regarding a zip code.
Updates changes made to city, state, country.
Sends details about cargo.
Updates cargo details.
SPECIAL_ REQUIREMENT _PASSENGER
CARGO
Specifies passengers that require accommodations.

# 3. ER Model


Name
Model
IDENTIFIER
Train
(0,N)
(0,N)
d
Passenger Train
seat number
have
Freight Train
(0,N)
(0,N)
(1,M)
Route
(1,N)
has
(1,1)
id
Contains
ID
Stop
(1,1)
Order
Destination
(1,1)
Cargo
(1,1)
(1,1)
Weight
Special handling
has
(0,N)
Cargo Owner
company name
Owner Id
Arrival Time
Departure Time
Id
(0,N)
(1,1)
Trip
has
(1,1)
(1,1)
(1,1)
have
have
have
Relationship
have
Name
Train Departures
includes
includes
(0,N)
(1,1)
(1,1)
(0,N)
Station
(1,1)
Station Id
(0,N)
(0,M)
(0,N)
Train arrivals
Connected
have
Train Number
(1,1)
(1,1)
(1,1)
Ticket Class Type
Ticket
(1,1)
Id
Seat Number
(1,1)
Status
Contains information
id
BirthDate
Age
(0,N)
Last Name
Passenger
(0,1)
has
First Name
Special requirement passenger
d
Disability
Accompanied travel
have
(0,N)
Address
Zip Code
(0,N)
Country
State
City
have
has
(1,1)
Security Info
ID
Question
Answer
Password
First name
Last name
User Id
Birthdate
(1,N)
(1,1)
(0,N)
User account
Gender
e-mail address
Age
Street Name
Apartment Number
Building Number
Id
First name
Last name
Employee
ssn
has
(1,1)
COOK
(0,N)
has
(1,1)
(0,M)
d
SECURITY_OFFICER
(1,1)
has
(1,1)
Crew
ATTENDANT
(1,1)
has
(1,1)
(1,1)
(1,1)
ENGINEER
(1,1)
has
CONDUCTOR
(1,1)
has

## 4. Normalized DB Model

Our DB model was normalized to 3NF only because most of it was already in 3NF or 2NF. We removed all transitivity we detected.

TRAIN

| Train_ID | Model | Name |
|---|---|---|

ROUTES

| Route_ID | Train_ID |
|---|---|

TRIP

| Trip_ID | Departure_Time | Arrival_Time | Train_ID | Origin_ID | Destination_ID |
|---|---|---|---|---|---|

STATION

| Station_ID | Name | Zip_Code |
|---|---|---|

STATION_CONNECTIONS

| Station_ID | Connected_ID |
|---|---|

STOP

| Route_ID | Station_ID | Order |
|---|---|---|

EMPLOYEE

| Employee_ID | First_Name | Last_Name | SSN | Emp_Type |
|---|---|---|---|---|

CREW

| Trip_ID | Conductor_ID | Engineer_ID | Cook_ID | Security_Officer_ID |
|---|---|---|---|---|

TRIP_ATTENDANTS

| Trip_ID | Attendant_ID |
|---|---|

CARGO

| ID | Weight | Special_Handling_Requirement | Trip_ID | Owner_ID | Train_ID |
|---|---|---|---|---|---|

FREIGHT_TRAIN

| Train_ID |
|---|

PASSENGER_TRAIN

| Train_ID | Seat_Amount |
|---|---|

CARGO_OWNER

| ID | Company_Name |
|---|---|

TICKET

| Ticket_ID | Seat_Number | Passenger_ID | Trip_ID | Train_ID |
|---|---|---|---|---|

PASSENGER

| Passenger_ID | First_Name | Last_Name | Birthdate | User_Account_ID |
|---|---|---|---|---|

SPECIAL_REQUIREMENT_PASSENGER

| Passenger_ID | Disability | Accompanied_Travel |
|---|---|---|

USER_ACCOUNT

| User_ID | Email | Password | First_Name | Last_Name | Birthdate | Gender | Street | Building | Apartment_Number | Zip_Code |
|---|---|---|---|---|---|---|---|---|---|---|

ADDRESS

| Zip_Code | City | State | Country |
|---|---|---|---|

USER_SECURITY_INFO

| ID | User_ID | Question | Answer |
|---|---|---|---|

# 5. Comparison Table

**MySql:**

| Pros | Cons |
|---|---|
| More flexible for data types | Expensive costs associated if you require specific features and plugins. |
| Multiple user access | Requires client/server architecture |
| Strong authentication and security features. | Larger amount of secondary memory required |
| Scalable to a very large database. | Does not implement the full SQL standard |
| Multi-user accessibility | Development is limited and controlled by Oracle |
| Very popular, much easier to find people who are familiar with MySQL. | |
| One of the fastest database solutions | |

**SQLite:**

| Pros | Cons |
|---|---|
| Serverless database | Owned by Oracle / Not Public Domain |
| Open Source | Only supports Blob, integer, null, text, real data types |
| Suitable for small database | No specific user management functionality |
| Portable as information is stored on single file | No built in authentication mechanism files can be accessed by anyone |
| Serverless as SQLite is self-contained | Not very scalable as volume of data is increased performance degrades |
| Very lightweight requiring very little secondary memory | Users not able to concurrently make changes at same time |
| Only cost is optional for support from developers | |

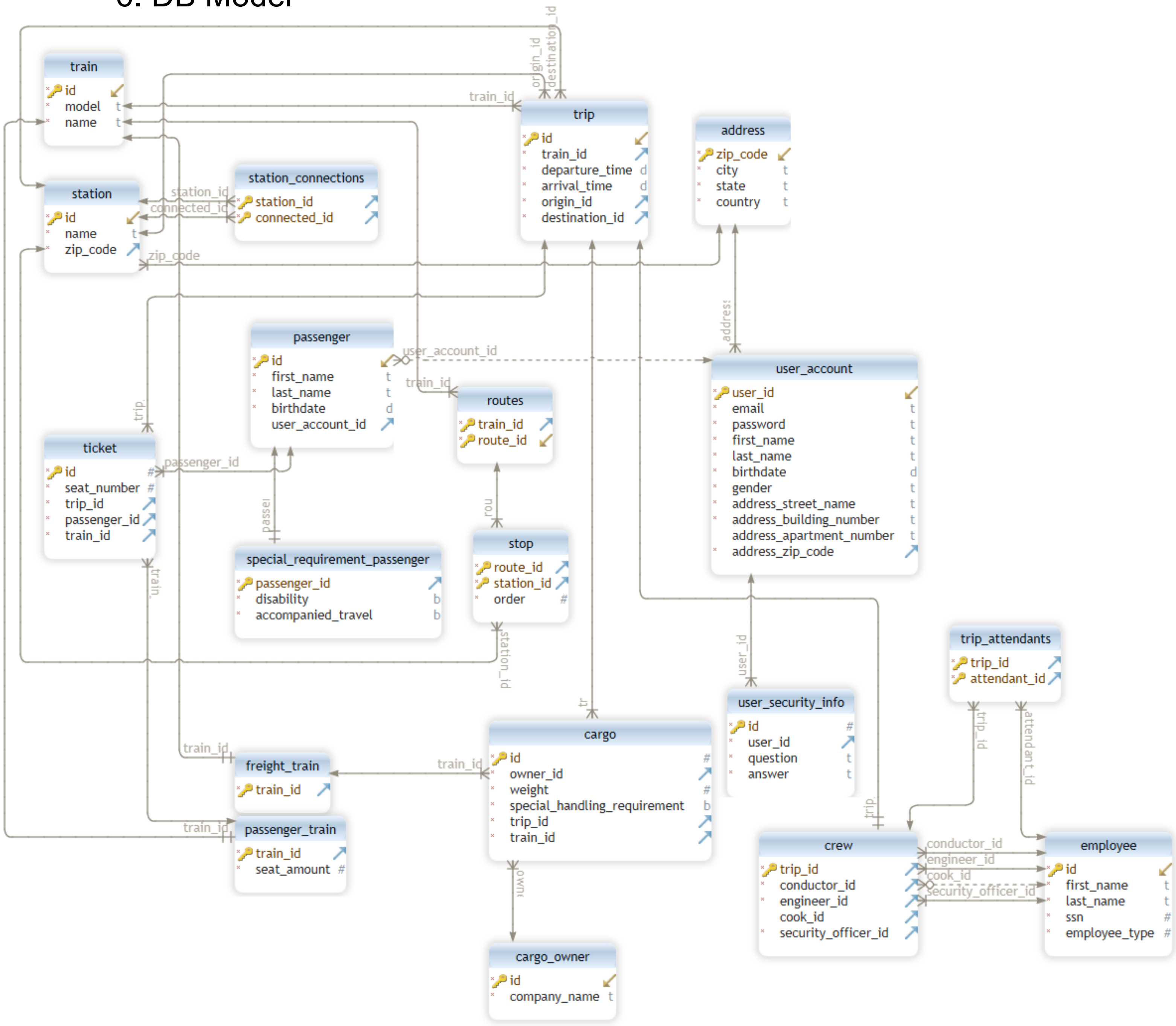
**MongoDB:**

| Pros | Cons |
|---|---|
| Dynamic Schematic Architecture | Implementation for indexing must be correct otherwise the user receives slow performance |
| Sharding is utilized for scalability | Acquiring data from multiple connections may require multiple queries. |
| Higher Speed compared to other database technologies | Duplication of data |
| Pricing starts very as low as $57 a month for smaller servers | High memory usage |
| Simple query syntax | Monthly costs significantly rise if the database becomes very large |
| Technical support with quick response times | Limited data sizes |

Our database management system of choice would be Oracle's MySql. MySql has multiple benefits to our railroad database model in regards to pricing, ease of use and security features when compared to other services. In terms of pricing MySql's support does have a few costs associated with it, but includes benefits such as 24 hour support, access to consultative support and consistent updates made by Oracle to ensure safety and efficiency. A major benefit in terms of cost is the ease of finding developers who are familiar with MySql at a high level compared to other database management system alternatives. One other major cost saving factor is the scalability of MySql. Over time a business may need to adjust their need to acclimate to their demands so the scalability of the database will also need to shrink or grow with the needs of the business. If the business requires a very large scale system, we would not have to allocate further time and resources to migrate a system or completely redesign aspects of our model.

MySql also provides various authentication and security features for a reliable and secure database. One of the needs of our client is to have various levels of authentication. We would not want a situation where a user can change train departure or arrival times or give everyone in the company the same access to all information. Using MySql's authentication system we will be able to limit what each user can view and modify. MySql also provides multiple user access which would be needed by the client. We would require multiple users creating transactions at the same time making modifications to their seats or requesting accommodations. Our system also would require a way to access and view each passenger's tickets simultaneously throughout the company's multiple stations and with real time updates to the information in the DBMS constantly changing as tickets are scanned and booked.

## 6. DB Model



The reason why we chose our tables to be in 3NF is because some of our schema was in 2NF and it had a lot of transitive dependencies. So in order to get rid of the transitive dependencies we had to choose 3NF. Every value in this schema is atomic and the attributes are fully functionally dependent.

# 7. Data Dictionary

| Name | Description | Role | Type | Format | Null allowed | Default value |
|---|---|---|---|---|---|---|
| **address** | Address of a passenger | E | — | — | — | — |
| zip_code | Address zip code (primary key) | A | varchar(10) | — | NO | — |
| city | Address city | A | varchar(255) | — | NO | — |
| state | Address state | A | varchar(255) | — | NO | — |
| country | Address country | A | varchar(255) | — | NO | — |
| **cargo_owner** | Information about a company which owns freight train cargo | E | — | — | — | — |
| id | ID of a cargo owner (primary key) | A | int | — | NO | — |
| company_name | Name of the company which owns the cargo | A | varchar(255) | — | NO | — |
| **employee** | Train employee | E | — | — | — | — |
| id | Employee ID (primary key) | A | int | — | NO | — |
| first_name | — | A | varchar(255) | — | NO | — |
| last_name | — | A | varchar(255) | — | NO | — |
| ssn | Employee social security number | A | int | — | NO | — |
| employee_type | Employee type, identifying attribute | A | tinyint | 1=CONDUCTOR \| 2=ENGINEER \| 3= ATTENDANT \| 4=COOK \| 5=SECURITY OFFICER | NO | — |
| **station** | Train station | E | — | — | — | — |
| id | Station ID (primary key) | A | int | — | NO | — |
| name | Station name | A | varchar(255) | — | NO | — |
| zip_code | Station zip code | A<br>R N:1 | varchar(10) | — | NO | — |

| | | | | | | |
|---|---|---|---|---|---|---|
| **station_connections** | Stations which are connected to a station | E | — | — | — | — |
| station_id | Station ID (primary key) | A<br>R 1:N | int | — | NO | — |
| connected_id | Connected station's ID (primary key) | A<br>R 1:N | int | — | NO | — |
| **train** | Train information | E | — | — | — | — |
| id | Train ID (primary key) | A | int | — | NO | — |
| model | Train model name | A | varchar(255) | — | NO | — |
| name | Train name | A | varchar(255) | — | NO | — |
| **trip** | Information about a train's trip | E | — | — | — | — |
| id | Trip ID (primary key) | A | int | — | NO | — |
| train_id | ID of the train which made the trip | A | int | — | NO | — |
| departure_time | The date and time the train departed | A | date | — | NO | — |
| arrival_time | The date and time the train arrived | A | date | — | NO | — |
| **user_account** | Information about a user account | E | — | — | — | — |
| user_id | User ID (primary key) | A | int | — | NO | — |
| email | Email associated with the account | A | varchar(255) | — | NO | — |
| password | User's password | A | varchar(255) | — | NO | — |
| first_name | First name of a user | A | varchar(255) | — | NO | — |
| last_name | Last name of a user | A | varchar(255) | — | NO | — |
| birthdate | Birth date of a user | A | date | — | NO | — |
| gender | Gender of a user | A | varchar(255) | — | NO | — |
| address_street_name | Street name of a user's address | A | varchar(255) | — | NO | — |
| address_building_number | Building number of a user's address | A | varchar(255) | — | NO | — |
| address_apartment_number | Building number of  a user's | A | varchar(255) | — | — | — |

| | | | | | | |
|---|---|---|---|---|---|---|
| | address | | | | | |
| address_zip_code | Zip code of the primary passenger’s address | A<br>R N:1 | varchar(10) | — | NO | — |
| **user_security_info** | Security information associated with a user account | E | — | — | — | — |
| id | ID of the security info (primary key) | A | int | — | NO | — |
| user_id | User account ID | A<br>R N:1 | int | — | NO | — |
| question | Security question | A | varchar(255) | — | NO | — |
| answer | Security answer | A | varchar(255) | — | NO | — |
| **crew** | Crew for a single train trip | E | — | — | — | — |
| trip_id | Trip ID associated with the crew (primary key) | A<br>R 1:1 | int | — | NO | — |
| conductor_id | Employee ID of the conductor | A<br>R 1:1 | int | — | NO | — |
| engineer_id | Employee ID of the engineer | A<br>R 1:1 | int | — | NO | — |
| cook_id | Employee ID of the cook | A<br>R 1:1 | int | — | — | — |
| security_officer_id | Employee ID of the security officer | A<br>R 1:1 | int | — | NO | — |
| **freight_train** | Information about a freight train | E | — | — | — | — |
| train_id | ID of the train (primary key) | A<br>R 1:1 | int | — | NO | — |
| **passenger** | Information about a passenger on a single passenger train trip | E | — | — | — | — |
| id | The ID of the passenger (primary key) | A | int | — | NO | — |
| first_name | Passenger’s first name | A | varchar(255) | — | NO | — |
| last_name | Passenger’s last name | A | varchar(255) | — | NO | — |
| birthdate | Passenger’s birth date | A | date | — | NO | — |
| user_account_id | A user account associated with a passenger | A<br>R N:1 | int | — | — | — |

| | | | | | | |
|---|---|---|---|---|---|---|
| **passenger_train** | Information about a passenger train | E | — | — | — | — |
| train_id | ID of the train (primary key) | A<br>R 1:1 | int | — | NO | — |
| seat_amount | Amount of seats available on the train | A | int | — | NO | — |
| **routes** | Information about a route a train takes | E | — | — | — | — |
| train_id | ID of the train (primary key) | A<br>R N:1 | int | — | NO | — |
| route_id | ID of the route (primary key) | A | int | — | NO | — |
| **special_requirement_passenger** | Passenger which has special requirements | E | — | — | — | — |
| passenger_id | ID of the passenger (primary key) | A<br>R 1:1 | int | — | NO | — |
| disability | Whether the passenger requires disability accommodations | A | bit | 0=no \| 1=yes | NO | — |
| accompanied_travel | Whether the passenger requires accompaniment | A | bit | 0=no \| 1=yes | NO | — |
| **stop** | An individual stop on a train route | E | — | — | — | — |
| route_id | ID of the route (primary key) | A<br>R N:1 | int | — | NO | — |
| station_id | ID of the station (primary key) | A<br>R N:1 | int | — | NO | — |
| order | Order of the stop in the route | A | int | — | NO | — |
| **ticket** | Information about a passenger's ticket | E | — | — | — | — |
| id | ID of the ticket (primary key) | A | int | — | NO | — |
| train_id | ID of a passenger train | A | int | — | NO | — |
| seat_number | Train seat number | A | int | — | NO | — |
| trip_id | ID of the train trip | A | int | — | NO | — |
| passenger_id | ID of the passenger | A | int | — | NO | — |

| | | | | | | |
|---|---|---|---|---|---|---|
| **trip_attendants** | Information about attendant employees on trips | E | — | — | — | — |
| trip_id | ID of the train trip (primary key) | A<br>R N:1 | int | — | NO | — |
| attendant_id | ID of the employee (primary key) | A<br>R 1:1 | int | — | NO | — |
| **cargo** | Information about freight train cargo | E | — | — | — | — |
| id | ID of the cargo (primary key) | A | int | — | NO | — |
| train_id | ID of a freight train | A<br>R 1:1 | int | — | NO | — |
| owner_id | ID of the cargo owner | A<br>R N:1 | int | — | NO | — |
| weight | Weight of the cargo | A | int | pounds | NO | — |
| trip_id | ID of the train trip | A<br>R N:1 | int | — | NO | — |
| special_handling_requirement | Whether the cargo has a special handling requirements | A | bit | 0=no \| 1=yes | NO | — |

# 8. SQL

## 8.1. Schema

```
CREATE SCHEMA traindb;

CREATE TABLE traindb.address (
    zip_code             varchar(10)  NOT NULL    PRIMARY KEY,
    city                 varchar(255)  NOT NULL,
    state                varchar(255)  NOT NULL,
    country              varchar(255)  NOT NULL
 );

CREATE TABLE traindb.cargo_owner (
    id                   int  NOT NULL    PRIMARY KEY,
    company_name         varchar(255)  NOT NULL
 );

CREATE TABLE traindb.employee (
    id                   int  NOT NULL    PRIMARY KEY,
    first_name           varchar(255)  NOT NULL,
    last_name            varchar(255)  NOT NULL,
    ssn                  int  NOT NULL,
    employee_type        tinyint  NOT NULL
 );

ALTER TABLE traindb.employee ADD CONSTRAINT cns_employee CHECK (
employee_type in (1,2,3,4,5) );

CREATE TABLE traindb.station (
    id                   int  NOT NULL    PRIMARY KEY,
    name                 varchar(255)  NOT NULL,
    zip_code             varchar(10)  NOT NULL
 );

CREATE TABLE traindb.station_connections (
    station_id           int  NOT NULL,
    connected_id         int  NOT NULL,
    CONSTRAINT pk_station_connections PRIMARY KEY ( station_id,
connected_id )
 );
```

```
CREATE TABLE traindb.train (
    id                   int  NOT NULL    PRIMARY KEY,
    model                varchar(255)  NOT NULL,
    name                 varchar(255)  NOT NULL
 );

CREATE TABLE traindb.trip (
    id                   int  NOT NULL    PRIMARY KEY,
    train_id             int  NOT NULL,
    departure_time       date  NOT NULL,
    arrival_time         date  NOT NULL,
    origin_id            int  NOT NULL,
    destination_id       int  NOT NULL
 );

CREATE TABLE traindb.user_account (
    user_id              int  NOT NULL    PRIMARY KEY,
    email                varchar(255)  NOT NULL,
    password             varchar(255)  NOT NULL,
    first_name           varchar(255)  NOT NULL,
    last_name            varchar(255)  NOT NULL,
    birthdate            date  NOT NULL,
    gender               varchar(255)  NOT NULL,
    address_street_name  varchar(255)  NOT NULL,
    address_building_number varchar(255)  NOT NULL,
    address_apartment_number varchar(255)  ,
    address_zip_code     varchar(10)  NOT NULL
 );

CREATE TABLE traindb.user_security_info (
    id                   int  NOT NULL    PRIMARY KEY,
    user_id              int  NOT NULL,
    question             varchar(255)  NOT NULL,
    answer               varchar(255)  NOT NULL
 );

CREATE TABLE traindb.crew (
    trip_id              int  NOT NULL    PRIMARY KEY,
    conductor_id         int  NOT NULL,
    engineer_id          int  NOT NULL,
    cook_id              int  ,
    security_officer_id  int  NOT NULL
 );

ALTER TABLE traindb.crew ADD CONSTRAINT cns_crew CHECK ( NOT EXISTS (SELECT
id FROM traindb.crew JOIN traindb.employee ON
traindb.crew.conductor_id=employee.id WHERE employee_type != 1) );

ALTER TABLE traindb.crew ADD CONSTRAINT cns_crew_0 CHECK ( NOT EXISTS
(SELECT id FROM traindb.crew JOIN traindb.employee ON
traindb.crew.engineer_id=employee.id WHERE employee_type != 2) );

ALTER TABLE traindb.crew ADD CONSTRAINT cns_crew_1 CHECK ( NOT EXISTS
(SELECT id FROM traindb.crew JOIN traindb.employee ON
traindb.crew.security_officer_id=employee.id WHERE employee_type != 3) );

ALTER TABLE traindb.crew ADD CONSTRAINT cns_crew_2 CHECK ( NOT EXISTS
(SELECT id FROM traindb.crew JOIN traindb.employee ON
traindb.crew.cook_id=employee.id WHERE employee_type != 5) );

CREATE TABLE traindb.freight_train (
    train_id             int  NOT NULL    PRIMARY KEY
 );

CREATE TABLE traindb.passenger (
    id                   int  NOT NULL    PRIMARY KEY,
    first_name           varchar(255)  NOT NULL,
    last_name            varchar(255)  NOT NULL,
    birthdate            date  NOT NULL,
    user_account_id      int
 );

CREATE TABLE traindb.passenger_train (
    train_id             int  NOT NULL    PRIMARY KEY,
    seat_amount          int  NOT NULL
 );

CREATE TABLE traindb.routes (
    train_id             int  NOT NULL,
    route_id             int  NOT NULL,
    CONSTRAINT pk_train_routes PRIMARY KEY ( train_id, route_id ),
    CONSTRAINT unq_train_routes_route_id UNIQUE ( route_id )
 );

CREATE TABLE traindb.special_requirement_passenger (
    passenger_id         int  NOT NULL    PRIMARY KEY,
    disability           bit  NOT NULL,
    accompanied_travel   bit  NOT NULL
 );

CREATE TABLE traindb.stop (
    route_id             int  NOT NULL,
    station_id           int  NOT NULL,
    `order`              int  NOT NULL,
    CONSTRAINT pk_stop PRIMARY KEY ( route_id, station_id )
 );

CREATE TABLE traindb.ticket (
    id                   int  NOT NULL    PRIMARY KEY,
    seat_number          int  NOT NULL,
    trip_id              int  NOT NULL,
    passenger_id         int  NOT NULL,
    train_id             int  NOT NULL
 );

CREATE TABLE traindb.trip_attendants (
    trip_id              int  NOT NULL,
    attendant_id         int  NOT NULL,
    CONSTRAINT pk_crew_attendants PRIMARY KEY ( trip_id, attendant_id )
 );

ALTER TABLE traindb.trip_attendants ADD CONSTRAINT cns_trip_attendants
CHECK ( NOT EXISTS (SELECT id FROM traindb.trip_attendants JOIN
traindb.employee ON traindb.trip_attendants.attendant_id=employee.id WHERE
employee_type != 4) );

CREATE TABLE traindb.cargo (
    id                   int  NOT NULL    PRIMARY KEY,
    owner_id             int  NOT NULL,
    weight               int  NOT NULL,
    special_handling_requirement bit  NOT NULL,
    trip_id              int  NOT NULL,
    train_id             int  NOT NULL
 );

ALTER TABLE traindb.cargo ADD CONSTRAINT fk_cargo_cargo_owner FOREIGN KEY (
owner_id ) REFERENCES traindb.cargo_owner( id ) ON DELETE NO ACTION ON
UPDATE NO ACTION;
```

```
ALTER TABLE traindb.cargo ADD CONSTRAINT fk_cargo_trip FOREIGN KEY (
trip_id ) REFERENCES traindb.trip( id ) ON DELETE NO ACTION ON UPDATE NO
ACTION;

ALTER TABLE traindb.cargo ADD CONSTRAINT fk_cargo_freight_train FOREIGN KEY
( train_id ) REFERENCES traindb.freight_train( train_id ) ON DELETE NO
ACTION ON UPDATE NO ACTION;

ALTER TABLE traindb.crew ADD CONSTRAINT fk_crew_trip FOREIGN KEY ( trip_id
) REFERENCES traindb.trip( id ) ON DELETE NO ACTION ON UPDATE NO ACTION;

ALTER TABLE traindb.crew ADD CONSTRAINT fk_crew_employee FOREIGN KEY (
conductor_id ) REFERENCES traindb.employee( id ) ON DELETE NO ACTION ON
UPDATE NO ACTION;

ALTER TABLE traindb.crew ADD CONSTRAINT fk_crew_employee_0 FOREIGN KEY (
engineer_id ) REFERENCES traindb.employee( id ) ON DELETE NO ACTION ON
UPDATE NO ACTION;

ALTER TABLE traindb.crew ADD CONSTRAINT fk_crew_employee_1 FOREIGN KEY (
cook_id ) REFERENCES traindb.employee( id ) ON DELETE NO ACTION ON UPDATE
NO ACTION;

ALTER TABLE traindb.crew ADD CONSTRAINT fk_crew_employee_2 FOREIGN KEY (
security_officer_id ) REFERENCES traindb.employee( id ) ON DELETE NO ACTION
ON UPDATE NO ACTION;

ALTER TABLE traindb.freight_train ADD CONSTRAINT fk_freight_train_train
FOREIGN KEY ( train_id ) REFERENCES traindb.train( id ) ON DELETE NO ACTION
ON UPDATE NO ACTION;

ALTER TABLE traindb.passenger ADD CONSTRAINT fk_passenger_user_account
FOREIGN KEY ( user_account_id ) REFERENCES traindb.user_account( user_id )
ON DELETE NO ACTION ON UPDATE NO ACTION;

ALTER TABLE traindb.passenger_train ADD CONSTRAINT fk_passenger_train_train
FOREIGN KEY ( train_id ) REFERENCES traindb.train( id ) ON DELETE NO ACTION
ON UPDATE NO ACTION;

ALTER TABLE traindb.routes ADD CONSTRAINT fk_train_routes_train FOREIGN KEY
( train_id ) REFERENCES traindb.train( id ) ON DELETE NO ACTION ON UPDATE
NO ACTION;
```

```
ALTER TABLE traindb.special_requirement_passenger ADD CONSTRAINT
fk_special_requirement_passenger_passenger FOREIGN KEY ( passenger_id )
REFERENCES traindb.passenger( id ) ON DELETE NO ACTION ON UPDATE NO ACTION;

ALTER TABLE traindb.station ADD CONSTRAINT fk_station_address FOREIGN KEY (
zip_code ) REFERENCES traindb.address( zip_code ) ON DELETE NO ACTION ON
UPDATE NO ACTION;

ALTER TABLE traindb.station_connections ADD CONSTRAINT
fk_station_connections_station FOREIGN KEY ( station_id ) REFERENCES
traindb.station( id ) ON DELETE NO ACTION ON UPDATE NO ACTION;

ALTER TABLE traindb.station_connections ADD CONSTRAINT
fk_station_connections_station_0 FOREIGN KEY ( connected_id ) REFERENCES
traindb.station( id ) ON DELETE NO ACTION ON UPDATE NO ACTION;

ALTER TABLE traindb.stop ADD CONSTRAINT fk_stop_station FOREIGN KEY (
station_id ) REFERENCES traindb.station( id ) ON DELETE NO ACTION ON UPDATE
NO ACTION;

ALTER TABLE traindb.stop ADD CONSTRAINT fk_stop_train_routes FOREIGN KEY (
route_id ) REFERENCES traindb.routes( route_id ) ON DELETE NO ACTION ON
UPDATE NO ACTION;

ALTER TABLE traindb.ticket ADD CONSTRAINT fk_ticket_trip FOREIGN KEY (
trip_id ) REFERENCES traindb.trip( id ) ON DELETE NO ACTION ON UPDATE NO
ACTION;

ALTER TABLE traindb.ticket ADD CONSTRAINT fk_ticket_passenger FOREIGN KEY (
passenger_id ) REFERENCES traindb.passenger( id ) ON DELETE NO ACTION ON
UPDATE NO ACTION;

ALTER TABLE traindb.ticket ADD CONSTRAINT fk_ticket_passenger_train FOREIGN
KEY ( train_id ) REFERENCES traindb.passenger_train( train_id ) ON DELETE
NO ACTION ON UPDATE NO ACTION;

ALTER TABLE traindb.trip ADD CONSTRAINT fk_trip_train FOREIGN KEY (
train_id ) REFERENCES traindb.train( id ) ON DELETE NO ACTION ON UPDATE NO
ACTION;

ALTER TABLE traindb.trip ADD CONSTRAINT fk_trip_station FOREIGN KEY (
origin_id ) REFERENCES traindb.station( id ) ON DELETE NO ACTION ON UPDATE
```

```
NO ACTION;

ALTER TABLE traindb.trip ADD CONSTRAINT fk_trip_station_0 FOREIGN KEY (
destination_id ) REFERENCES traindb.station( id ) ON DELETE NO ACTION ON
UPDATE NO ACTION;

ALTER TABLE traindb.trip_attendants ADD CONSTRAINT fk_crew_attendants_crew
FOREIGN KEY ( trip_id ) REFERENCES traindb.crew( trip_id ) ON DELETE NO
ACTION ON UPDATE NO ACTION;

ALTER TABLE traindb.trip_attendants ADD CONSTRAINT
fk_crew_attendants_employee FOREIGN KEY ( attendant_id ) REFERENCES
traindb.employee( id ) ON DELETE NO ACTION ON UPDATE NO ACTION;

ALTER TABLE traindb.user_account ADD CONSTRAINT fk_user_account_address
FOREIGN KEY ( address_zip_code ) REFERENCES traindb.address( zip_code ) ON
DELETE NO ACTION ON UPDATE NO ACTION;

ALTER TABLE traindb.user_security_info ADD CONSTRAINT
fk_user_security_info_user_account FOREIGN KEY ( user_id ) REFERENCES
traindb.user_account( user_id ) ON DELETE NO ACTION ON UPDATE NO ACTION;
```

## Useful Queries

Displays the ticket ID, trip ID, the train number that the passenger will be on, the passenger’s seat number (where they would be sitting) along with the passenger’s name and if the passenger requires special accommodations.

```
SELECT t.id, t.train_id, t.seat_number, t.trip_id, p.first_name,
p.last_name, COALESCE(srp.disability, 0) AS disablity_travel,
COALESCE(srp.accompanied_travel, 0) AS accompanied_travel
FROM traindb.ticket t
      INNER JOIN traindb.passenger p ON ( t.passenger_id = p.id  )
      LEFT JOIN traindb.special_requirement_passenger srp ON ( p.id =
srp.passenger_id  )
```

Displays the train information such as the train’s ID, the model of the train, the name of the train, and counts how many trips were taken by each train

```
SELECT t.id, t.model, t.name, COUNT(t1.id) AS TRIP_COUNT
FROM traindb.train t
      INNER JOIN traindb.trip t1 ON ( t.id = t1.train_id  )
      GROUP BY t.id;
```

Displays each employee’s name, employment type, SSN, and employee ID.

```
SELECT e.id, e.first_name, e.last_name, e.ssn, e.employee_type
FROM traindb.employee e
```

Pulls each user’s address and personal information.

```
SELECT ua.user_id, ua.email, ua.password, ua.first_name, ua.last_name,
ua.birthdate, ua.gender, ua.address_street_name,
ua.address_building_number, ua.address_apartment_number,
ua.address_zip_code
FROM traindb.user_account ua
```

Pulls the station’s ID, name, city, state and zip-code.

```
SELECT s.id, s.name, a.city, a.state,s.zip_code
FROM traindb.station s
      INNER JOIN traindb.address a ON ( s.zip_code = a.zip_code  )
```

Selects the trip ID, train ID and counts the amount of passengers with special accommodations on each trip.

```
SELECT t.trip_id, t.train_id, COUNT(spr.passenger_id) FROM
traindb.special_requirement_passenger spr
    JOIN traindb.ticket t ON (spr.passenger_id = t.passenger_id) GROUP BY
t.trip_ID;
```

Selects trip ID, train ID, the amount of seats on the train and checks to see how many seats are booked.

```
SELECT t.id, t.train_id, pt.seat_amount, COUNT(t2.id) AS tickets_booked
FROM traindb.trip t
      INNER JOIN traindb.train t1 ON ( t.train_id = t1.id  )
      INNER JOIN traindb.passenger_train pt ON ( t1.id = pt.train_id  )
      INNER JOIN traindb.ticket t2 ON ( pt.train_id = t2.train_id  ) group
by train_id
```

Displays the total number of tickets booked by station displaying the busiest to the least busy stations in descending order along with the station’s ID, name, zip_code, city and state.

```
SELECT s.id, s.name, s.zip_code, a.city, a.state, count(t.id) AS trip_count
FROM traindb.station s
      INNER JOIN traindb.address a ON ( s.zip_code = a.zip_code  )
      INNER JOIN traindb.trip t ON ( s.id = t.destination_id  ) group by
s.id order by trip_count DESC;
```

Displays the employees’ total amount of taken trips in descending order.

```
SELECT e.id, e.first_name, e.last_name, e.employee_type, COUNT(ta.trip_ID) +
COUNT(c.trip_ID) AS trips_taken
FROM traindb.employee e
      LEFT OUTER JOIN traindb.crew c ON ( e.id = c.conductor_id OR e.id =
c.engineer_id OR e.id = c.cook_id OR e.id = c.security_officer_id  )
      LEFT OUTER JOIN traindb.trip_attendants ta ON ( e.id = ta.attendant_id
) GROUP BY e.id ORDER BY trips_taken DESC
```

## 9. Time Logs

| TASK | Sourav Dhar | Atil Goker | Omar Muhammad | Walter Pompa |
|---|---|---|---|---|
| System Requirements | 1 | 3 | 3 | 1.5 |
| Contextual Data Flow | .5 | .5 | 0 | 3.5 |
| ER/EER Model | 4 | 1.5 | 0.5 | 0 |
| Normalized DB Model | .5 | .5 | 0.5 | 3.5 |
| DBMS And Rationale | .5 | 1 | 0 | 0 |
| Implementation DB Model | 0 | 0 | 2 | 0 |
| Data Dictionary | 5 | 0 | 2.5 | 0 |
| SQL (Creation & Queries) | 1 | 3 | 0 | 0.5 |
| Time Log | .5 | .5 | 0.5 | 0.5 |
| Appendix | .5 | .5 | 0.5 | 0.5 |
| Meetings | 12 | 15 | 15 | 15 |
| **Total (hours)** | 25.5 Hours | 25.5 Hours | 25.5 hr | 25 hr |

Walter Pompa Omar Muhammad Sourav Dhar

Atil Goker Walter Pompa Omar Muhammad Sourav Dhar

# 10. Appendix

System Requirements (pg 3-5)

Freight trains can have zero cargo to cover the case of freight trains traveling with no cargo. Stations may have zero connected stations to cover the case of newly built stations or stations for which construction is in progress.

Contextual Data Flow (pg 6)

There is so much information flowing in and out that we decided to color code the diagram for readability.

ER/EER Model (pg 7)

For this ER diagram model we chose the ERD plus software [ https://erdplus.com/ ]. This software has all these attributes, entity, relationship shapes.

Normalized DB Model (pg 8-9)

Our model was discussed and done in such a way that we practically already had it in 3NF.

DBMS And Rationale (pg 10-11)

We compared SQLite, MongoDB and MySQL as we wanted to contrast a solution that was implemented into the end program, a NoSQL solution and a client-server SQL solution.

Implementation DB Model (pg 12)

We used Software[ https://dbschema.com/ ] to implement the DB Model.

Data Dictionary (pg 13-17)

The left value in the relationship role denotes the amount of the attribute it is under; that is, N:1 means that N of the attribute in the containing relation corresponds to 1 in the foreign relation.

SQL (pg 18-26)

N/A

# Feedback

Hi,

Please proceed with Railroad Database.

Please follow guidelines given by professor for Phase-1.

Feedback Date
Nov 8, 2021 4:26 PM

---

You guys can proceed forward with next phases.

Feedback Date
11/16/2021 12:55 PM