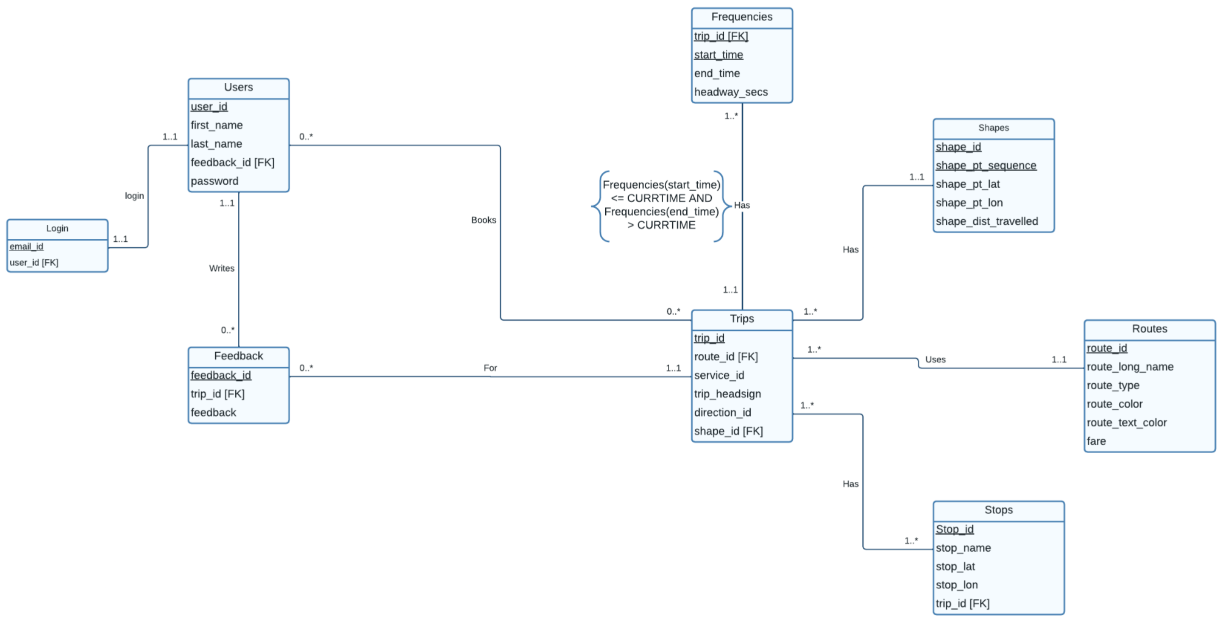
STAGE 2

UML DIAGRAM:



SCHEMA NORMALIZATION:

From the UML diagram mentioned above, we can see that the ‘user’ entity attribute only depends on the Primary key- ‘user_id’ and has no transitive dependencies.

In the entities ‘frequencies’ and ‘shapes’, we have candidate keys as (trip_id, start_time) and (shape_id, shape_pt_sequence) respectively. These two entities are non-transitive in nature as well.

Similar to user entity, all other entity attributes(Trips,Fairs,Routes,Stops,Feedback, Login) all depend on the Primary of the respective entities and have no partial/transitive dependency.

So, we chose BCNF in comparison to 3NF since:

- So, all of them follow BCNF- all entities depend on the PK only
- The schema cannot be further normalized.
- The redundancy is lower in BCNF
- Since there are no transitive/partial dependencies, this is in BCNF. If X -> Y is an FD, then X should be a superkey. And this is satisfied in the FD below.

Functional Dependencies:

Users(user_id, first_name, last_name, feedback_id, password)

- FD: user_id ->first_name,last_name,feedback_id, password
- No other attribute can uniquely identify any other attribute, except for the primary key.

Login(email_id, user_id)

- FD: Email -> UserId
- We removed email_id from user table since there a a transitive dependency between email_id and User_id, since both are unique and can identify each other, to reduce it to a BCNF we had to normalise the original User table and create a Login table.
- The login table as evident satisfies BCNF rules.

Feedback(feedback_id, trip_id, feedback)

- FD: feedback_id -> trip_id, feedback
- No other attribute can uniquely identify any other attribute, except for the primary key.

Frequencies(trip_id, start_time, end_time, headway_secs)

- FD: trip_id, start_time ->end_time, headway_secs
- No other attribute can uniquely identify any other attribute, except for the primary key.
- Trip_id alone cannot uniquely identify other attributes because trip_id is a reference to the name of the route, whereas when we take the start time into consideration, we get unique values because each vehicle will have a set trip and time of origination. For example, if 5Green is the trip_id, then there will be multiple buses running on this route, but 5Green at 8 a.m. is a unique bus, and there will only be that one bus running at that time on that route.

Shapes (shape_id, shape_pt_sequence, shape_pt_lat, shape_pt_long, shape_dist_travelled)

- FD: shape_id, shape_pt_sequence-> shape_pt_lat, shape_pt_long, shape_dist_travelled
- No other attribute can uniquely identify any other attribute, except for the primary key.
- 2NF is when a relation is in 1NF and it has no partial dependencies, meaning there are no predicates (columns) that depend on only part of a multi-part key,

our primary key (shape_id,shape_pt_sequence) is hence in 2NF, and since there is no transitive dependency, it is also in 3NF.

Routes(route_id, route_long_name, route_type, route_color, route_text_color, fare)

- FD: route_id-> route_long_name, route_type, route_color, route_text_color, fare
- No other attribute can uniquely identify any other attribute, except for the primary key.

Trips(trip_id, route_id,service_id, trip_headsign, direction_id, shape_id)

- FD: trip_id -> route_id,service_id, trip_headsign, direction_id, shape_id
- No other attribute can uniquely identify any other attribute, except for the primary key.

Stops(stop_id, stop_name,stop_lat, stop_lat)

- FD: stop_id -> stop_name,stop_lat, stop_lat
- No other attribute can uniquely identify any other attribute, except for the primary key.

RELATIONAL SCHEMA:

Users(user_id: INT[PK], first_name: VARCHAR(100), last_name: VARCHAR(100), feedback_id: INT[FK to Feedback.feedback_id], password: VARCHAR(100))

Login(email_id: VARCHAR(100)[PK], user_id: INT[FK to Users.user_id])

Feedback(feedback_id: INT[PK], trip_id: VARCHAR(100)[FK to Trips.trip_id], feedback: VARCHAR(200)

Trips(trip_id: VARCHAR(100)[PK], route_id: VARCHAR(100)[FK to Routes.route_id], service_id: INT, trip_headsign: VARCHAR(100), direction_id: INT, shape_id: INT[FK to Shapes.shape_id])

Frequencies(trip_id: VARCHAR(100)[FK to Trips.trip_id], start_time: DATETIME[PK], end_time: DATETIME, headway_secs: INT)
(trip_id and start_time act as primary key)

Routes(route_id: VARCHAR(100)[PK], route_long_name: VARCHAR(100), route_type: INT, route_color: VARCHAR(100), route_text_color: VARCHAR(100), fare: INT)

Shapes(shape_id: INT[PK], shape_pt_sequence: INT[PK], shape_pt_lat: REAL, shape_pt_lon: REAL, shape_dist_traveled: REAL)
(shape_id and shape_pt_sequence act as primary keys.)

Stops(stop_id: INT[PK], stop_name: VARCHAR(100), stop_lat: REAL, stop_lon: REAL)

DESCRIPTIONS, CARDINALITY & ASSUMPTIONS OF RELATIONSHIPS:

BASIC ASSUMPTIONS:

- For joining Frequencies with Trips table, we will use the following conditions:
    - Frequencies(start_time) <= CURRTIME AND Frequencies(end_time) > CURRTIME.
- We assume that all fares are only 1$.
- We will be having sessions for the login process for the persistence of userId to be a foreign key in the Login.

We will mention the assumptions pertaining to each of the relations below with descriptions and cardinality.

1. **Users:**
    a. User books Trips. Cardinality: Users table has 0..* relationship with Trips
        Each user can have zero or more trips.

    b. User writes Feedback. Cardinality: Users table has 0..* relationship with Feedback
        Each user can give no feedback or many feedbacks for each trip. Therefore, the relationship is zero to many.

    c. User has a Login. Cardinality: Users table has 1.1 relationship with Login.
        Each user account can only have one unique email, therefore the relationship is of the kind one to one.

2. **Feedback:**
    a. Feedback is written by Users. Cardinality: Feedback table has 1..1 relationship with Users
        For the selected feedback, it can be written by one user. Therefore, the relationship is one to one.

    b. Feedback is provided for the Trips. Cardinality: Feedback table has 1..1 relationship with Trips
        The feedback is mapped to a single trip. Therefore, the relationship is one to one.

3. **Trips:**
    a. Trips have Feedbacks. Cardinality: Trips table has 0..* relationship with Feedbacks

        Each trip can have no feedback or any number of feedback. Therefore, the relationship is 0 to many.

    b. Trips are booked for Users. Cardinality: Trips table has 0..* relationship with Users

        Either no trip or any number of trips can be booked for each user. Therefore, the relationship is zero to many.

    c. Trips have Frequencies. Cardinality: Trips table has 1..* relationship with Frequencies

        Each trip can have one or more frequencies. Therefore, the relationship is one to many.

    d. Trips have Shapes. Cardinality: Trips table has 1..1 relationship with Shapes

        Each trip can have one shape. Therefore the relationship is one to one.

    e. Trips use Routes. Cardinality: Trips table has 1..1 relationship with Routes

        Each trip can have one type of route. Therefore, the relationship is one to one.

    f. Trips have Stops. Cardinality: Trips table has 1..* relationship with Stops

        Each trip can have many stops. Therefore, the relationship is one to many.

4. **Stops**
    a. Stops are there(has) for Trips. Cardinality: Stops table has 1..* relationship with Trips

        Each stop can be reached by many buses(trips). Therefore, the relationship is one to many.

5. **Shapes**
    a. Shapes are there(has) for Trips. Cardinality: Shapes table has 1..* relationship with trips *

        A shape can have many trips. Therefore, the relationship is one to many.

6. **Frequencies**
    a. Frequencies are there(has) for Trips. Cardinality: Frequencies table has 1..1 relationship with Trips

        A single frequency can have a trip. Therefore, the relationship is one to one.

7. **Routes**
    a. Routes are there(has) for Trips. Cardinality: Routes table has 1..* relationship with Trips

        Each route can have multiple Trips. Therefore, the relationship is one to many.

8. **Login**
    a. Login is there for Users. Cardinality: Login table has 1..1 Relationship with Users

For each user, there will be one login. Therefore, the relationship is one to one.

Public Transportation Dataset Tables Information:

This is the modified version of the proposed datasets provided that we will be using for the project.

Table: Frequencies

- trip_id
    - Description: Unique identifier for the trip.
- start_time
    - Description: Start time of the trip.
- end_time
    - Description: End time of the trip
- headway_secs
    - Description: The time between successive trips on the same route.

Table: Routes

- route_id
    - Description: Unique identifier for the route.
- route_long_name
    - Description: Full name of the route.
- route_type
    - Description: Type of the transportation route.
- route_color
    - Description: Color associated with the route.
- route_text_color
    - Description: Color of the text or labels associated with route.
- fare
    - Description: Price of the fare.

Table: Shapes

- shape_id
    - Description: Unique identifier for the shape.
- shape_pt_lat
    - Description: Latitude of a point on the shape.
- shape_pt_lon
    - Description: Longitude of a point on the shape.
- shape_pt_sequence
    - Description: The sequence order of the point on the shape.
- shape_dist_traveled
    - Description: The distance traveled along the shape up to this point.

Table: Stops

- stop_id
    - Description: Unique identifier for the stop.
- stop_name
    - Description: Name of the stop.
- stop_lat
    - Description: Latitude of the stop location.
- stop_lon
    - Description: Longitude of the stop location.
- trip_id
    - Description: Unique identifier for the trip.

Table: Trips

- route_id
    - Description: Unique identifier for the route.
- service_id
    - Description: Identifier for the service associated with the trip.
- trip_id
    - Description: Unique identifier for the trip.
- trip_headsign
    - Description: Headsign of the trip.
- direction_id
    - Description: Direction of the trip. (E.g., 0 for outbound, 1 for inbound)
- shape_id
    - Description: Identifier for the shape associated with the trip.

Created Extra Tables:

Table: Users

- user_id
    - Description: Unique identifier for the user.
- first_name
    - Description: User's first name.
- last_name
    - Description: User's last name.
- feedback_id
    - Description: User's feedback statement.
- password
    - Description: User's password.

Table: Login

- email
    - Description: Unique email for the login.
- user_id
    - Description: Login user_id relation to Users.user_id.

Table: Feedback

- feedback_id
    - Description: Unique identifier for the feedback.
- trip_id
    - Description: Identifier for the trip associated with the feedback.
- feedback
    - Description: User’s feedback or comments on the trip.

STAGE 1 IMPROVEMENTS:

We added **Realness** as instructed:

The dataset titled "Public Transportation Service" hosted on Kaggle contains information about the public transportation system of São Paulo, a major Brazilian city. The data is provided in a format called GTFS (General Transit Feed Specification), which is specifically designed to model transportation schedules and geographic data. The link: https://www.kaggle.com/datasets/mateuscco/sao-paulo-transportation-service

FUNCTIONS:

When implementing the functionalities, we will use the update and delete functions in the context of triggers and cursors.

Example:

ON DELETE SET NULL (in trigger)

We will use the update function in the user table to update the password and change the personal info when needed.

We will also provide Database Manager access to delete and update entries in the database.

We also have possible updations/deletions of the shape the transport takes and could switch the routes on specific dates.

USEFULNESS:

[Usefulness is vague about similar applications. You want to present examples of similar ones clearly.]: (your comments)

There are similar ideas to ours, like https://tripplanner.airtreks.com/ or Travelocity, but they only offer some of our proposed functionalities. For example, they offer planning for trips but offer no statistics. On the other hand, our website provides insights into popular trips, routes and busy stations based on our data.