CS411 T47 Stage2 UML, 3NF, Relational Schema

Entity:

User: We think each user has an unique email, username, and favorites_id, and each can have the same password as another user.

Favorite: We think that each user's favorite items include one unique favorite_id, while favorite_stops and favorite_routes can be repeated

Planner: We think each planner has a unique planner_id, while email and plan_stops can be repeated.

Trips: We think each trip has an unique trip_id and shape_id, and other attributes can be repeated.

Routes: We think that each route has an unique route_id, but it can have repeated route_long_name, route_color, route_text_color, fare_id.

Fares: We think each fare has a unique fare_id, while price, currency_type, payment_method, transfers, and transfer_duration can be the same.

Bus_stops: We think each stop has a unique stop_id, but it can have repeated stop_name, stop_desc, stop_lat, stop_lon.

Frequencies: We think each trip_id has many start_time, end_time, and headway_secs, while the start_time, end_time, and headway_secs could be the same. In addition to that, we think the combination of (trip_id, start_time) is unique.

Calendar: We think each calendar has a unique service_id, but it can have repeated values for the rest of attributes.

Shapes: Each shape represents a coordinate/point of the routes. We think the combination of (shape_id, shape_pt_sequence) is unique, but shape_pt_lat and shape_pt_lon can be repeated.

Relationship:

Favor: We think each user can create(favor) up to one favorite entity, and each favorite corresponds to a unique user by its favorites_id. (one-to-one)

Create: We think each user can create many planners but not required, and each planner corresponds to a unique user by email. (user to planner: one-to-many)

Comment: We think each user can leave as many comments for any route but is not required. Each route can have reviews commented by users but also can have no comments at all. Each comment has four aspects of crowdedness, safety, temperature, and accessibility. (many-to-many)

Consist: We think each route consists of at least one trip since it might have two opposite directions. For each trip, it must only apply to one route. (route to trip: one-to-many)

Cost: We think routes can have only one fare, but each fare can apply to multiple routes. (route to cost: many-to-one)

Arrive: We think that each trip arrives at many stops, and for each arrival, there is arrival_time and stop_sequence. (many-to-many)

Frequent: We think each frequency corresponds to unique trips, but each trip has many frequencies. (frequency to trip: many-to-one)

Map: We think each trip has many points/coordinates (shapes). (trip to shapes: one-to-many)

Date: We think each trip has only one kind of calendar, but there are many trips having the same kind of calendar. (trip to calendar: many-to-one)

Functional Dependencies

User: username -> password, favorite_id, email | email -> username, password, favorite_id

Planner: planner_id -> plan_stops, email

Favorites: favorite_id -> favorite_stops, favorite_routes

Bus_stops: stop_id -> stop_name, stop_desc, stop_lat, stop_lon | stop_lat, stop_ lon -> stop_id

Routes: route_id -> route_long_name, route_color, route_text_color, fare_id

Fares: fare_id -> price, currency_type, payment_method, transfers, transfer_duration

Shapes: shape_id, shape_pt_sequence -> shape_pt_lat, shape_pt_lon, shape_dist_traveled

Trips: trip_id -> route_id, service_id, trip_headsign, direction_id, shape_id | route_id, direction_id -> trip_id

Frequencies: trip_id, start_time -> headway_secs, stop_time | trip_id, stop_time -> headway_secs, start_time

Calendar: service_id -> monday, tuesday, wednesday, thursday, friday, saturday, sunday, start_date, end_date

Comment: email, route_id -> crowdedness, temperature, accessibility, safety

Arrival: trip_id, stop_id -> arrival_time, stop_sequence | trip_id, stop_sequence -> stop_id

3NF VS BCNF

We chose 3NF as our normalization method since we care about the steps taken place in functional dependencies, so we cannot use BCNF to minimize the redundancy. For example, we do need to use functional dependencies where the key is on the right side (using stop_lat and stop_long to get stop_id).

Relational Schema

User(username: VARCHAR(15), password: VARCHAR(15), email: VARCHAR(40) [PK], favorite_id: INT [FK to Favorites.favorites_id])

Planner(planner_id: INT [PK], plan_stops: LIST, email: VARCHAR(40) [FK to User.email])

Favorites(favorites_id: INT [PK], favorite_stops: LIST, favorite_routes: LIST)

Comment(email: VARCHAR(40)[PK][FK to User.email], route_id: VARCHAR(7) [PK][FK to Routes.route_id])

Trips(trip_id: VARCHAR(9) [PK], route_id: VARCHAR(7) [FK to Routes.route_id], service_id: VARCHAR(3) [FK to Calendar.service_id], trip_headsign: VARCHAR(256), direction_id: INT, shape_id: INT [FK to Shapes.shape_id])

Routes(route_id: VARCHAR(7) [PK], route_long_name: VARCHAR(256), route_color: VARCHAR(6), route_text_color: VARCHAR(6), fare_id: VARCHAR(21) [FK to Fares.fare_id])

Fares(fare_id: VARCHAR(21) [PK], price: REAL, currency_type: VARCHAR(3), payment_method: INT, transfers: VARCHAR(1), transfer_duration: INT)

Bus_stops(stop_id: INT [PK], stop_name: VARCHAR(256), stop_desc: VARCHARE(256), stop_lat: REAL, stop_lon: REAL)

Frequencies(trip_id: VARCHAR(9) [PK] [FK to Trips.trip_id], headway_secs: INT, start_time: VARCHAR(8) [PK], stop_time: VARCHAR(8))

Calendar(service_id: VARCHAR(3) [PK], monday: INT, tuesday: INT, wednesday: INT, thursday: INT, friday: INT, saturday: INT, sunday: INT, start_date: INT, end_date: INT)

Shapes(shape_id: INT [PK], shape_pt_sequence: INT [PK], shape_pt_lat: REAL, shape_pt_lon: REAL, shape_dist_traveled: REAL)

Arrival(trip_id: VARCHAR(9) [PK][Fk Trips.trip_id], stop_id: INT [PK][FK to Bus_stops.stop_id], arrival_time: VARCHAR(8), stop_sequence: INT)