**Intent:**

The intent of this Entity-Relationship (ER) diagram is to design a structured Bus Transportation Management System that efficiently manages bus riders, routes, fares, bus stops, schedules, and drivers. The system aims to handle the relationship between these entities by organizing key information like route types, stop locations, fare details, and timing schedules. This ER model will help in optimizing route planning, scheduling, and service management for both passengers and drivers across different service days and times.

**Objectives**

To identify and define key entities involved in a bus transportation system such as Bus Rider, Bus Driver, Route, Bus Stop, Fare, and Schedule.

To establish relationships between these entities to manage operational aspects like fare calculation, driver scheduling, route planning, and stop management.

To design a system that efficiently handles route schedules based on different service days (weekdays, weekends, holidays) and timing requirements.

To manage driver assignments and schedules in relation to routes, dates, and times, ensuring smooth service operation.

To track fare structures based on rider types and route types for better fare management and passenger categorization.

To provide a clear data structure that supports operational decision-making, such as assigning drivers to routes and scheduling trips based on demand patterns.

To improve passenger service and operational efficiency by organizing key system data in a way that makes it easier to retrieve, analyze, and update information.

To serve as the foundation for developing a database system or application that automates and streamlines bus transportation management tasks.

import graphviz

# Create a new ER diagram using Graphviz

er = graphviz.Digraph('Enhanced\_ER\_Diagram', format='png')

er.attr(rankdir='LR', size='12')

# Entities and their attributes

entities = {

'Rider': ['rider\_id (PK)', 'name', 'age', 'gender', 'email', 'type\_id (FK)'],

'RiderType': ['type\_id (PK)', 'type\_name', 'description'],

'Preference': ['preference\_id (PK)', 'preference\_name'],

'RiderPreference': ['rider\_id (FK)', 'preference\_id (FK)'],

'Fare': ['fare\_id (PK)', 'rider\_type\_id (FK)', 'route\_type\_id (FK)', 'price'],

'Route': ['route\_id (PK)', 'route\_name', 'route\_type\_id (FK)'],

'RouteType': ['route\_type\_id (PK)', 'type\_name'],

'BusStop': ['stop\_id (PK)', 'location\_name', 'latitude', 'longitude'],

'RouteStop': ['route\_id (FK)', 'stop\_id (FK)', 'stop\_order'],

'Schedule': ['schedule\_id (PK)', 'route\_id (FK)', 'schedule\_type\_id (FK)', 'start\_time', 'key\_stop\_times'],

'ScheduleType': ['schedule\_type\_id (PK)', 'type\_name'],

'BusDriver': ['driver\_id (PK)', 'name', 'employee\_number', 'contact\_info'],

'DriverSchedule': ['schedule\_id (PK)', 'driver\_id (FK)', 'date', 'route\_id (FK)', 'time\_slot']

}

# Add entities as nodes

for entity, attrs in entities. items():

label = f"{entity}|" + "\\l”. join(attrs) + "\\l"

er.node (entity, label=f"{{{label}}}", shape='record')

# Define relationships

relationships = [

('Rider', 'RiderType', 'has', 'N', '1'),

('Rider', 'RiderPreference', '', '1', 'N'),

('Preference', 'RiderPreference', '', '1', 'N'),

('Fare', 'RiderType', '', 'N', '1'),

('Fare', 'RouteType', '', 'N', '1'),

('Route', 'RouteType', '', 'N', '1'),

('RouteStop', 'Route', '', 'N', '1'),

('RouteStop', 'BusStop', '', 'N', '1'),

('Schedule', 'Route', '', 'N', '1'),

('Schedule', 'ScheduleType', '', 'N', '1'),

('DriverSchedule', 'BusDriver', '', 'N', '1'),

('DriverSchedule', 'Route', '', 'N', '1')

]

# Add relationships as edges

for src, dst, label, src\_card, dst\_card in relationships:

er.edge(src, dst, label=label + f" ({src\_card}:{dst\_card})" if label else f"({src\_card}:{dst\_card})")

# Render the diagram

er.render(directory='/tmp', cleanup=True)

'/tmp/Enhanced\_ER\_Diagram.png'