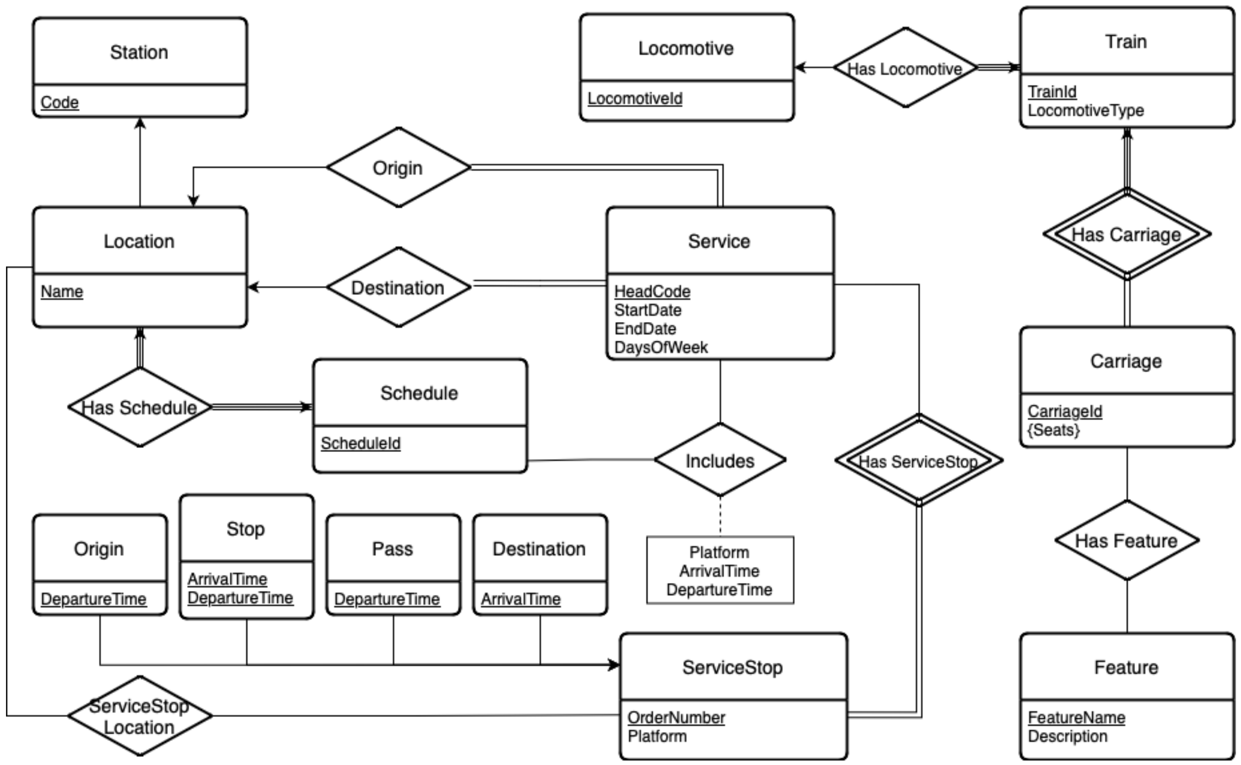
1. Entity Relation Model

1.1 ER Diagram



1.2 Justification of Design

My provided ER diagram effectively captures the core elements of the British Rail database. It represents the entities, attributes, and relationships necessary to model the railway network. Within the diagram, there are ten entities stored for use: Station, Location, Train, Locomotive, Carriage, Feature, Service, Schedule and ServiceStop, Origin, Pass, Stop, and Destination. A Station represents physical train stations, identified by a unique code. The Code attribute ensures uniqueness and serves as the primary identifier. The Location entity represents any geographical point, including stations, stops, and intermediate locations. The Name attribute stores the location name. A Train entity shows a unit with a locomotive and one or more carriages. Locomotives are a type of engine component of a train, linked to a specific train entity. Carriages represent a section of the train, with features and seat information. A Feature defines amenities or characteristics of a carriage, such as Wi-Fi or first-class seating. A Service is a scheduled journey with origin, destination, and a timetable. The Schedule records the service details for a specific location. ServiceStop while it sounds ambiguous represents intermediate stops, with arrival and departure times. Origin, Stop, Pass, and Destination indicate whether a service stops at or passes through a location. These four also show the hierarchy of the database as each one are technique attribute of a service.

1.3 Assumptions and Ambiguities

During implementation a few assumptions were made regarding the scenario provided. Firstly, I have assumed that every service has both an origin and a destination location. I made this assumption to simplify implementation and because there was nothing in the specification that led me to assume otherwise regardless of not being explicitly stated. Secondly, some service routes might run multiple days of the week on repeat. I create a DaysOfWeek attribute to store that information. This attribute is stored as a string for simplicity and shows the first (and sometimes second) letter for the day of the week. (e.g., Monday – M, Tuesday – T, Wednesday – W, Thursday – Th, Friday – F, Saturday – S, Sunday - S). Furthermore, Platform numbers are stored as strings to accommodate possible alphanumeric values. I also assumed that ServiceStops allow for both stopping and passing locations. Another assumption I made was that CarriageFeature is used for the many to many relationships between a Carriage and Feature. My final assumption was that ScheduleLocation in Schedule refers to a location where services are recorded. As for any ambiguity, the idea of a train stopping at a specific location was replaced with the ServiceStop entity to properly capture location order and stop details.

2. Relational Model

2.1 Tables and Attributes

Station(Code: CHAR(3), PRIMARY KEY (Code), FOREIGN KEY (Name) REFERENCES Location(Name))

Location(Name: VARCHAR(100), PRIMARY KEY (Name))

Locomotive(LocomotiveId: INT, PRIMARY KEY (LocomotiveId))

Train(TrainID: INT, LocomotiveId: INT, LocomotiveType: VARCHAR(50), PRIMARY KEY (TrainID), FOREIGN KEY (LocomotiveId) REFERENCES Locomotive(LocomotiveId))

Carriage(CarriageID: INT, TrainID: INT, Seats: INT, PRIMARY KEY (CarriageID), FOREIGN KEY (TrainID) REFERENCES Train(TrainID))

Feature(FeatureName: VARCHAR(50), Description: TEXT, PRIMARY KEY (FeatureName))

CarriageFeature(CarriageID: INT, FeatureName: VARCHAR(50), FOREIGN KEY (CarriageID) REFERENCES Carriage(CarriageID), FOREIGN KEY (FeatureName) REFERENCES Feature(FeatureName))

Service(HeadCode: VARCHAR(10), StartDate: DATE, EndDate: DATE, DaysOfWeek: VARCHAR(20), Origin: VARCHAR(100), Destination: VARCHAR(100), PRIMARY KEY (HeadCode), FOREIGN KEY (Origin) REFERENCES Location(Name), FOREIGN KEY (Destination) REFERENCES Location(Name))

Schedule(ScheduleId: INT, HeadCode: VARCHAR(10), Location: VARCHAR(100), PRIMARY KEY (ScheduleId), FOREIGN KEY (HeadCode) REFERENCES Service(HeadCode), FOREIGN KEY (Location) REFERENCES Location(Name))

ServiceStop(StopId: INT, HeadCode: VARCHAR(10), Location: VARCHAR(100), OrderNumber: INT, Platform: VARCHAR(10), ArrivalTime: TIME, DepartureTime: TIME, PRIMARY KEY (StopId), FOREIGN KEY (HeadCode) REFERENCES Service(HeadCode), FOREIGN KEY (Location) REFERENCES Location(Name))

Stop(StopId: INT, ArrivalTime: TIME, DepartureTime: Time, FOREIGN KEY (StopId) REFERENCES ServiceStop(StopId))

Pass(PassId: INT, ServiceId: INT, LocationId: INT, DepartureTime: TIME, PRIMARY KEY (PassId), FOREIGN KEY (ServiceId) REFERENCES Service(ServiceId), FOREIGN KEY (LocationId) REFERENCES Location(LocationId))

Origin(OriginId: INT, DepartureTime: TIME, FOREIGN KEY (OriginId) REFERENCES Location(LocationId))

Destination(DestinationId: INT, ArrivalTime: TIME, FOREIGN KEY (DestinationId) REFERENCES Location(LocationId))

2.2 Functional Dependencies

1. Station: Code -> Name

2. Location: LocationId -> Name

3. Locomotive: LocomotiveId -> {TrainId, LocomotiveType}

4. Train: TrainId -> {LocomotiveId, LocomotiveType}

5. Carriage: CarriageId -> {TrainId, Seats}

6. Service: ServiceId -> {HeadCode, StartDate, EndDate, DaysOfWeek, Origin, Destination}

7. ServiceStop: StopId -> {ServiceId, LocationId, OrderNumber, Platform, ArrivalTime, DepartureTime}

2.3 Normal Forms

My model is in Boyce-Codd Normal Form. All functional dependencies have determinants that are super keys. There are no partial dependencies (as all non-key attributes fully depend on the primary key). There are no transitive dependencies.

3. Database Representation

3.1 MariaDB DDL

The SQL DDL script creates all the necessary tables and enforces primary and foreign key constraints. It uses appropriate data types and ensures data integrity by referencing related tables with foreign keys.

3.2 Implementation Considerations

To ensure integrity constraints primary keys are created with unique IDs to ensure uniqueness and identify each record. A few examples of this are Station(Code), and Carriage(CarriageId). Foreign keys enforce referential integrity between related tables. For example, Train(LocomotiveId) references Locomotive(LocomotiveId). NOT NULL constraints prevent missing critical data when inserting information into a table. For example, Service(HeadCode) cannot be NULL as it is critical information. The main data types used are CHAR(3) for station codes to accommodation fixed-length codes. VARCHAR is used for various-length textual data. TIME, DATE, and INT are used appropriately for time date, and numeric values as needed.

3.3 Differences from Relational Model

The main key difference from the relational model is that each entity now has its own specific identification as a primary key. These are called the entity followed by Id, and each one is assigned a unique integer value. This change instead of using Name, HeadCode, etc attributes was to ensure integrity of the database. There is the possibility that a location, stations, trains etc could have the same name or identifying attribute which would provide an issue. Other than that, there are no signification differences between the ER model and the implementation. The relational model was nearly directly translated into SQL.

4. Conclusion

The British Rail database design effectively models the railway network's services, rolling stock, and locations. It adheres to best practices in normalization and ensures data integrity through appropriate constraints. The SQL implementation is designed to be clear, efficient, and easily maintainable.