Database Systems - CW2 Eurostar 2030 – Group 28

Student ID	First Name	Last Name
200310837	Nicolae	Savin
200203573	Sameer	Alam
200382845	Anis	Zainudin
200181949	Asad	Khan
200856690	Constantinos	Caryotis

---

1. Assumptions – interpretation of the scenario + requirements analysis

A database (DB) for the Eurostar company is required. The DB should keep track of the trains operated by the company on specific routes. Details about crews, trips or passengers are also contained within the database.

- Eurostar company operates on several routes. We design a route as an entity type. Each route has a start station and a terminal station. A route also includes its distance, a journey time and information about whether the route is considered new or old. Any route is identified by its uniquely assigned ID. We consider routes with the same start or terminal station.
- If a route is new, then a simple single-valued attribute stores the details of its status (whether it is operational, it will be operational soon, or it is just planned to be constructed in the future).
- Each route contains 0 to many middle stations which are designed as a stop - entity type. There is a partial participation between a route and a stop as a route can have no stops - the terminal station of a route is not considered a stop. In this case, a direct train operates on that route.
- For each stop the database stores its name which acts as the primary key because the names of the stops (stations) are unique and not null. We consider that each stop must belong to at least one route (total participation). Between route and stop we define a many-to-many relationship as a route may also have many stops.
- The database keeps track of the trains which operate on specific routes. We design a train as an entity type. Each train can operate on many routes and each route can have many trains assigned. This forms a M:N relationship. There is a total participation between a route and a train, but a train which is not operable is not assigned to a route – partial participation.
- For a train, the DB stores its start and end points. We store a primary key for every train: its ID. Moreover, the DS stores an attribute which defines whether a train is operational and the price of the journey on that train. If a train is not operational, then the database cannot assign it to a trip.
- The train's fabrication date is also stored to state whether the train is considered modern or not. In order to assign only modern trains (fabricated starting with 2016) to only new routes, we need to check whether the fabrication date of a train is higher than '01-01-2016' and whether the route is classified as

a new one. The database does not permit old trains to operate on new routes and vice versa by checking these constraints.

- A relationship between a train and a stop needs to be defined. The stops of a route are stored in the database the same as the stops of a train, but the stops assigned to a train and a route can be different. A train may not have a stop (direct train) - partial participation, but in other cases a train has multiple stops. This forms a 1:N relationship between train and stop. The ending point of a train does not represent a stop for that train.
- The database stores information about the employees of the company. Each employee has a unique single-valued ID (primary key), a first name, a last name and a role. Roles include driver, conductor, service team member, security guard, management, sales or mechanics (train repair).
- The DB keeps track of crews represented as an entity type. The crew is made up of eleven employees: two drivers, two conductors, five service team members and two security guards, but we will design this in our system as N employees assigned to a crew. In our design, a crew is fixed as crew members are not interchangeable through other crews. Each employee may be assigned to only one crew and one crew has N employees. This forms a 1:N relationship. For each crew, the database stores the ID of the head of the crew as a primary key.
- The database is designed such that each crew operates on a single train at a time and every train has a single crew assigned, forming a 1:1 relationship with a total participation for both sides.
- The database should also keep track of the information about passengers. As an entity type, a passenger is identified uniquely by their National Insurance Number (NIN), which acts as a primary key. The database stores other useful information for every passenger: their first and last name and the type for each passenger which includes: pupil, student, adult or senior.
- Each passenger takes at least one train for their journey. The database stores the day, month and year every passenger travel on a train. Each train has at least one passenger assigned for the trip. This forms a M:N relationship between a train and a passenger with total participation from both sides. A passenger that takes only one train on a route for their journey uses a direct train and the price of the journey is higher as it is more convenient for the passenger.

---

2. Entity-Relationship Model (ERM) – textual form

Summary of all entities:

- Route
- Stop
- Train
- Passenger
- Crew
- Employee

Details about the entities and their attributes:

- Route (<u>ID</u>, StartStation, TerminalStation, Distance, JourneyTime, IsOld, Status)
  - Domain for ‘IsOld’: Yes/No
  - Domain for ‘Status’: Operational, Soon, Planning
- Stop (<u>Name</u>)

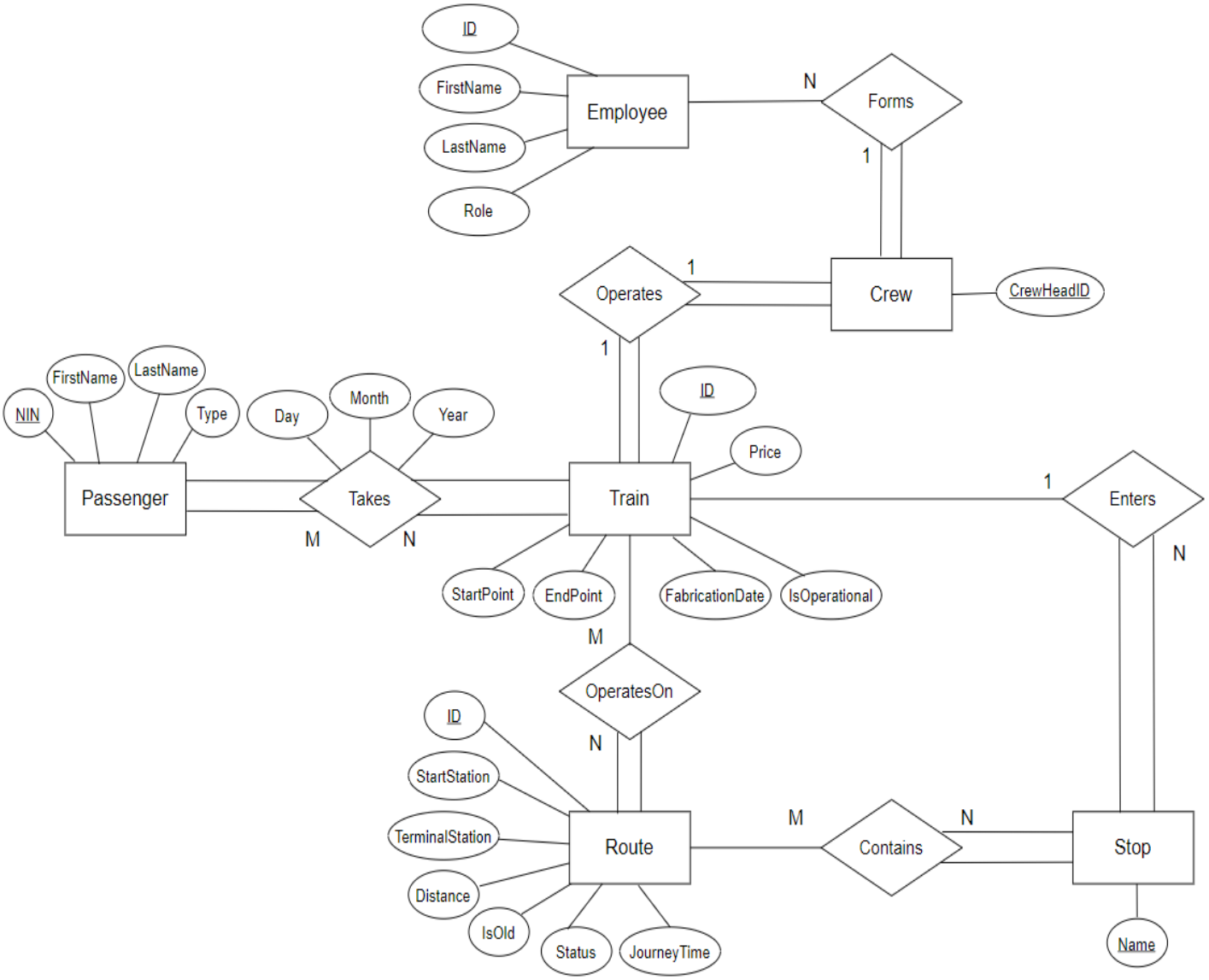
- Train (ID, StartPoint, EndPoint, FabricationDate, IsOperational)
    - Domain for ‘IsOperational’: Yes/No
    - Domain for ‘FabricationDate’: date value
- Passenger (NIN, FirstName, LastName, Type)
    - Domain for ‘Type’: Pupil, Student, Adult, Senior
- Crew (CrewHeadID)
- Employee (ID, FirstName, LastName, Role)
    - Domain for ‘Role’: Driver, Conductor, Member, Security, Management, Sales, Mechanic

Relationships and their multiplicity:

- Contains (Route, Stop) - M:N
    - Partial participation between Route and Stop
    - Total participation between Stop and Route
- OperatesOn (Train, Route) - M:N
    - Partial participation between Train and Route
    - Total participation between Route and Train
- Enters (Train, Stop) - 1:N
    - Partial participation between Train and Stop
    - Total participation between Stop and Train
- Operates (Crew, Train) - 1:1
    - Total participation between both ends
- Forms (Crew, Employee) - 1:N
    - Partial participation between Employee and Crew
    - Total participation between Crew and Employee
- Takes (Passenger, Train) - M:N
    - Total participation between both ends

---

3. Entity-Relationship Diagram (ERD)



4.1. Relational Database Schema

We need to design the key migrations between relationships with multiplicity 1:1 and 1:N.

Initial entity tables – including their attributes and primary key:

- Route (ID, StartStation, TerminalStation, Distance, JourneyTime, IsOld, Status)
- Stop (Name)
- Train (ID, StartPoint, EndPoint, FabricationDate, IsOperational, Price)
- Passenger (NIN, FirstName, LastName, Type)
- Crew (CrewHeadID)
- Employee (ID, FirstName, LastName, Role)

For each relationship, we apply the key migration if required. The schemas for relationships in the DB are:

- Contains (RouteID, StopName)
    - RouteID refers to Route.ID
    - StopName refers to Stop.Name
    - No key migration required
- OperatesOn (RouteID, TrainID)
    - RouteID refers to Route.ID
    - TrainID refers to Train.ID
    - No key migration required
- Enters (TrainID, StationName)
    - TrainID refers to Train.ID
    - StationName refers to Stop.Name
    - There is a key migration in Stop. We include the attribute TrainID which refers to Train.ID as a foreign key to Train – Stop (…, TrainID)
- Operates (TrainID, CrewHeadID)
    - TrainID refers to Train.ID
    - CrewHeadID refers to Crew.CrewHeadID
    - There is a key migration in Train. We include a CrewID which refers to Crew.ID as a foreign key to Crew – Train (…, CrewID)
    - There is a key migration in Crew. We include a TrainID which refers to Train.ID as a foreign key to Train – Crew (…, TrainID)
- Forms (EmployeeID, CrewHeadID)
    - EmployeeID refers to Employee.ID
    - CrewHeadID refers to Crew.CrewHeadID
    - There is not a key migration in Employee since not all employees form a crew.
- Takes (PassengerNIN, TrainID, Day, Month, Year)
    - PassengerNIN refers to Passenger.NIN
    - TrainID refers to Train.ID
    - We want to store the day, month and year of a passenger's journey on a train
    - There is no key migration required – M:N relationship

Final configuration of the database schema after key migration:

- Route (ID, StartStation, TerminalStation, Distance, JourneyTime, IsOld, Status)
- Stop (Name, TrainID)
- Train (ID, StartPoint, EndPoint, FabricationDate, IsOperational, Price, CrewID)
- Passenger (NIN, FirstName, LastName, Type)
- Crew (CrewHeadID, TrainID)
- Employee (ID, FirstName, LastName, Role)
- Contains (RouteID, StopName)
- OperatesOn (RouteID, TrainID)
- Enters (TrainID, StationName)

- Operates (TrainID, CrewHeadID)
- Forms (EmployeeID, CrewHeadID)
- Takes (PassengerNIN, TrainID, Day, Month, Year)

4.2. Normalised Schema

- 1st Normal Form
  - As the relationships of the database contain only atomic attributes, all of them respect the first normal form.
- 2nd Normal Form – Analysing whether there are partial dependencies within relationships
  - All the tables are in the first normal form
  - Possible partial dependency - Takes (PassengerNIN, TrainID, Day, Month, Year)
  - A passenger that travels on a specific route can take multiple trains on the same route on the same date. That means that given any attribute - day, month or year, we cannot identify either the passenger NIN or the ID of the train a passenger travels on since a passenger can take many trains in the same day/month/year and a train makes multiple journeys on the same day.
  - The relationships are indeed in second normal form
- 3rd Normal Form – Analysing the existence of transitive dependencies in relationships
  - All the relationships are in second normal form
  - Route (ID, StartStation, TerminalStation, Distance, JourneyTime, IsOld, Status) - possible transitive dependency
  - One can argue that the Distance, JourneyTime or IsOld attributes are dependent on the StartStation or TerminalStation. In our database, they are not because there are many different routes with the same start or terminal stations. There are many routes with the same JourneyTime or Distance. At the same time, the value of IsOld may change if the route suffers an upgrade.
  - In this case, all the relationships are indeed in third normal form.