**Group 27 DB CW group 27 - Eurostar 2030**

|  |  |
| --- | --- |
| **Name** | **Student Number** |
| Safaa Sheikh | 220364432 |
| Daryna Tsvykh | 220530680 |
| Vinisha Vekaria | 220348557 |
| Ryan Boone | 230729247 |

**Requirement Analysis:**

Eurostar is an international high-speed rail service in Western Europe, connecting Belgium, France, Germany, the Netherlands and the United Kingdom and reaching an reach a top speed of 300 kilometres per hour (that’s 186 miles per hour) on high-speed lines The very first Eurostar and Thalys trains departed in 1994 and 1996 respectively. Since then, the company carried millions of passengers through the Channel Tunnel from London and between some of the most exciting European cities including Paris, Brussels, Amsterdam, Rotterdam, Antwerp, and Cologne.

A Eurostar database is required to keep track of travel routes. This will help consider new routes including planes and ferries.

Also,customer satisfaction is monitored via online survey or by interacting with staff.Customer Care Team handles complaints; between 1 st April 2022 and 31st March 2023 our Customer Care team received 4489 customer contacts that were logged as a complaint.

For trains travelling between Belgium, France, The Netherlands and Germany, ticket issuing is still handled by Thalys‘ shareholder railway undertakings (SNCF, SNCB) and partner (NS) through their distribution channels, as well as through other contractual distributors, using the information systems of Thalys' shareholders and partner.

27th June 2023, London, UK: Eurostar Group, the high-speed rail service connecting the UK with Belgium, France, Germany and the Netherlands, today announces its 2022 turnover of €1,532 million. This is 2.5 times higher than in 2021.The Group carried 14.8 million passengers in 2022: 8.3 million for Eurostar and 6.5 million for Thalys. Following a severe Covid downturn in early 2022, it took the two businesses just 3 months to recover from 30% to 80% in passenger volume.

“These results are encouraging, and as the backbone of sustainable travel in Europe, we are doing everything we can to grow our offer to meet demand,” says Eurostar Group CEO, Ms Gwendoline Cazenave. “We will continue striving to reach our target of carrying 30 million passengers by 2030”.

Therefore, database design is important for Eurostar to be able to meet their goals like increasing the number of customers up to 30 million.

***Design Assumptions:***

\*Passengers travelling on the new routes cannot use the old trains\*

\*Passengers should be able to find out when their trains are departing and arriving to its destination; keep track of their own journeys\*

\*For the train to be successfully operated, at least 11 employees are needed to be on-board.\*

\*Each trip is designated to a fixed train, with a fixed number of employees eg (at least 11 employees), each trip is allocated to a specific crew and train (each train might have an id)\*

\*Each employee is allocated to specific train journey with the unique train team\*

\*Each train journey has various types of different passengers

**A textual description of the Entity-relationship model:**

Entities include: passengers, employees, stations, trains, routes, journey, repairs, price fares

*Entities:*

Train: This entity is a representation of the Eurostar trains. To keep track of the routes that each train can service, it has features like "Train ID," "Manufacture Year," "Status" (operational, under service check, under repair)

Route: The Route object lists the various routes that Eurostar offers for travel. A few of its properties are "RouteID", "StartStation", "TerminalStation", "Distance”, "JourneyTime",

Employee: This entity represents the workers of Eurostar and to which train & position they were allocated. A few examples of attributes are "Employee ID," "FirstName", “LastName”, "RoleID", “StartTime”, “EndTime” (such as driver, conductor, member of the service team, security guard, manager, or sales), and "Department."

Passenger: Data regarding travellers on Eurostar trains is stored by the Passenger entity. Attributes may include "PassengerID," "FirstName,"LastName”, "Type" (e.g., student, adult, senior), “Age”

Journey: Specific train journey data is captured by the Journey entity to have more information about each trip It includes attributes like "JourneyID," "Date," "TrainID,"

"Route ID," "PassengerCount," ”PassengerType” "EmployeeIDs" (to link to the employees involved in the trip), “CrewID”(to which crew employees were assigned)

Ticket: “TransactionID”, “BuyerType”, “TicketPrice”

*Relationships:*

Employees can be assigned to one train - many to one relationship

Each journey is assigned with one train - one to one relationship

Each ticket corresponds to one train journey - one to one relationship

Each journey may have more than one route - one to many relationship

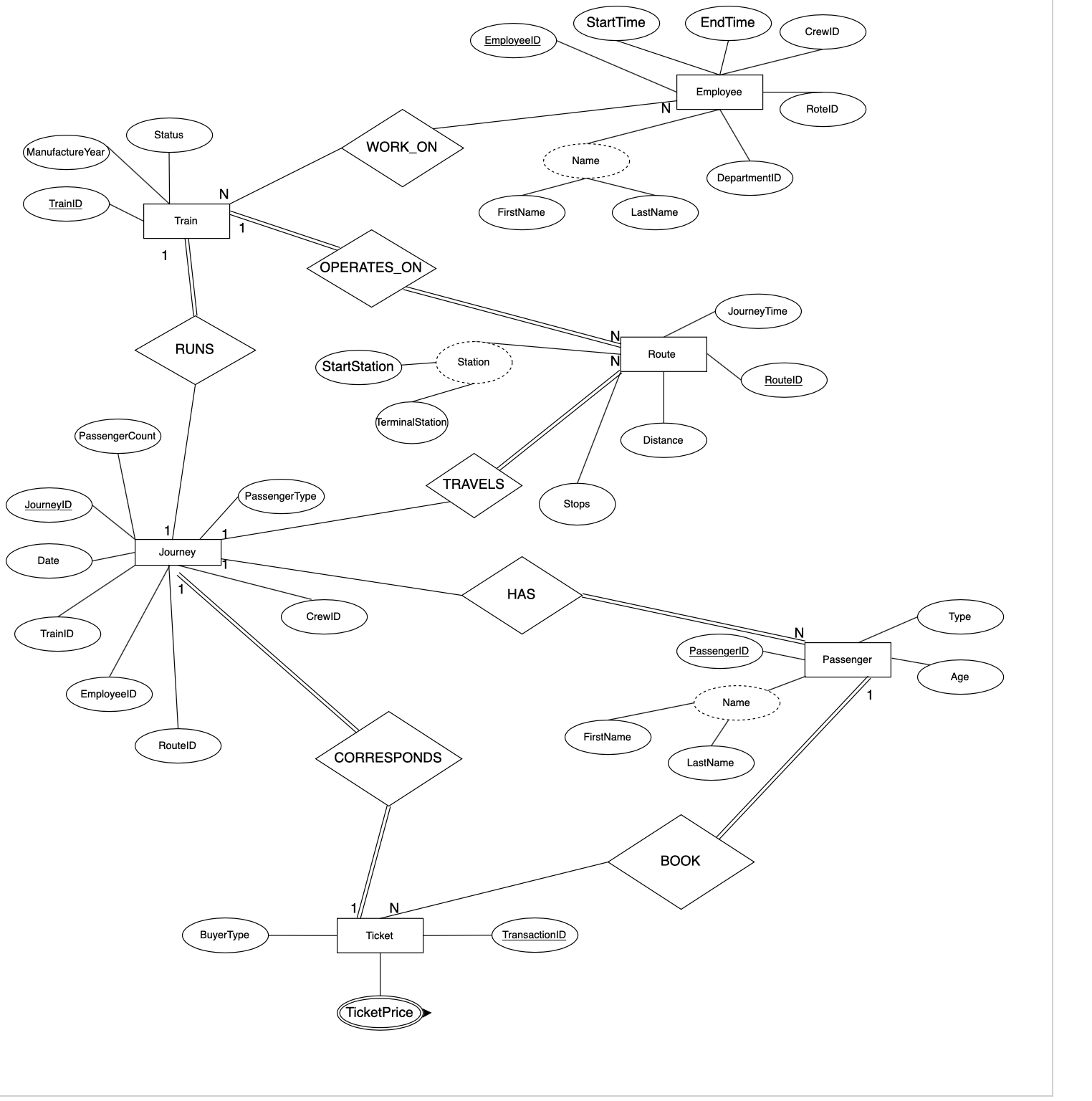
Passengers can book multiple tickets - one to many

One train can go through many routes - one to many

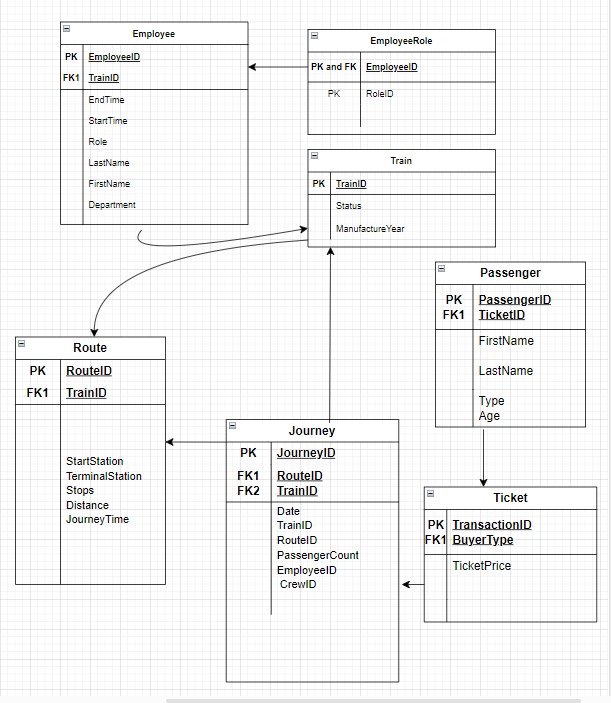
One route can have multiple journeys - one to many

|  |  |  |  |
| --- | --- | --- | --- |
| **Entities** | **Attributes** | **Relationships** | **Participation** |
| Train (Strong entity - not dependent on other entities) | TrainID(primary key),  ManufactureYear, Status | RUNS (Train, Journey) - 1:1  OPERATES\_ON (Train, Routes) - 1:N | Mandatory - Train entity,  Optional - Journey entity(journey cannot exist without a train)  Mandatory - Train entity;  Mandatory - Route entity (route is independent) |
| Passenger (Strong entity - not dependent on any other entities) | PassengerID  (primary key), FirstName, LastName, Type, Age | BOOK (Passenger, Ticket) - 1:N | Mandatory - Passenger entity  Optional - Ticket entity |
| Route (Strong entity) | RouteID(primary key), StartStation, TerminalStation, Stops, Distance, JourneyTime | TRAVELS - (Route, Journey) - 1:N | Mandatory - Route entity  Optional - Journey entity |
| Journey (weak entity - foreign key attributes) | JourneyID(primary key), Date, TrainID, RouteID, PassengerCount, EmployeeID, CrewID | HAS - (Journey, passengers) - 1:N | Mandatory- Passenger entity  Optional- Route entity |
| Employee (Strong entity - not dependent on other entities) | EmployeeID  (primary key), FirstName,  LastName, RoleID, DepartmentID, TrainID, StartTime, EndTime | WORK\_ON  (Employees, Train) - N:1 | Optional -Employee entity  Optional- Train entity |
| Ticket (weak entity - dependent on passenger entity) | TransactionID, BuyerType, TicketPrice | CORRESPONDS  (Ticket, Journey) - 1:1 | Mandatory- Ticket entity  Mandatory - Journey entity |

**A graphical description (ER diagram)**



**The mapping of the conceptual model to the logical model**



1. Start by identifying the entities and attributes from the eurostar model. Each entity becomes a table, and each attribute becomes a column in the table such as Train, Route, Station, Staff, Journey, Passenger, and Ticket. Consider attributes like TrainID, RouteID, EmployeeID, etc.
2. For each entity, figure out what the primary key could be. This could be a single attribute or a combination of attributes that uniquely identifies each record in the table. For example, TrainID could be the primary key for the Train entity.
3. Identify the relationships between entities. Decide on the type of relationship (one-to-one, one-to-many, many-to-many). Once this is done we are able to create foreign keys in tables to represent these relationships.For instance, a Train serves multiple Routes, and employees are associated with one Train for a Journey.
4. Normalising the tables to remove redundancy and improve data integrity. This makes sure that each table represents a single concept and that there are minimal data redundancies.
5. Apply constraints to enforce data integrity. This includes primary key constraints, foreign key constraints to ensure referential integrity, and any other constraints necessary for the business rules .For example, make sure that only Modern trains can be assigned to run on the new Stratford-Cologne route.
6. Assign appropriate data types and lengths to each column based on the nature of the data. For example, use VARCHAR for names, INT for numeric identifiers, DATE for date values, etc. Use appropriate data types for attributes like DepartureTime, ArrivalTime, etc.
7. Define default values for columns where appropriate. Decide whether columns should allow NULL values or if they should be constrained to be NOT NULL.
8. Identify columns that should be indexed for performance optimization. This is particularly important for columns used in JOIN operations or WHERE clauses.
9. Review the logical model to make sure that it correctly reflects the requirements of the eurostar model and document the logical model, including tables, columns, relationships, constraints, and any other relevant information. This documentation serves as a guide for database administrators, developers, and other stakeholders.
10. Finally, add the logical model in the eurostar model. Create the tables, define relationships, and apply constraints based on the finalised model.

**Normalise the schema to 3NF where required**

The passenger entity satisfies the requirements of 2NF as non-prime attributes don’t rely on the candidate key {PassenerID}.

The train entity also satisfies the requirements of being in 2NF form because non-prime attributes do not depend on candidate key: {TrainID}

**Employee\_Information Table (before normalisation):**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **EmployeeID** | FirstName | LastName | RoleID | DepartmentID | TrainID | StartTime | EndTime |

**Role\_Information Table to 3NF (after normalisation):**

|  |  |
| --- | --- |
| RoleID | RoleName |

**Department\_Information Table to 3NF (after normalisation):**

|  |  |
| --- | --- |
| DepartmentID | DepartmentName |