André Almeida, Nº 88960, P3G10

BD - Projeto

Introduction 1

Requirement Analysis 1

Entity-Relationship Diagram 2

Relational Schema 2

Normalization 3

Views 3

SQL Programming 3

Stored Procedures 3

User Defined Functions 3

Scalar 3

Inline Table Valued 3

Multi-Statement Table Valued 3

Triggers 4

Final Notes 4

Introduction

The database created in this project stores data in service to a hypothetical public transport system, where its details are covered more thoroughly in the next section (requirement analysis).

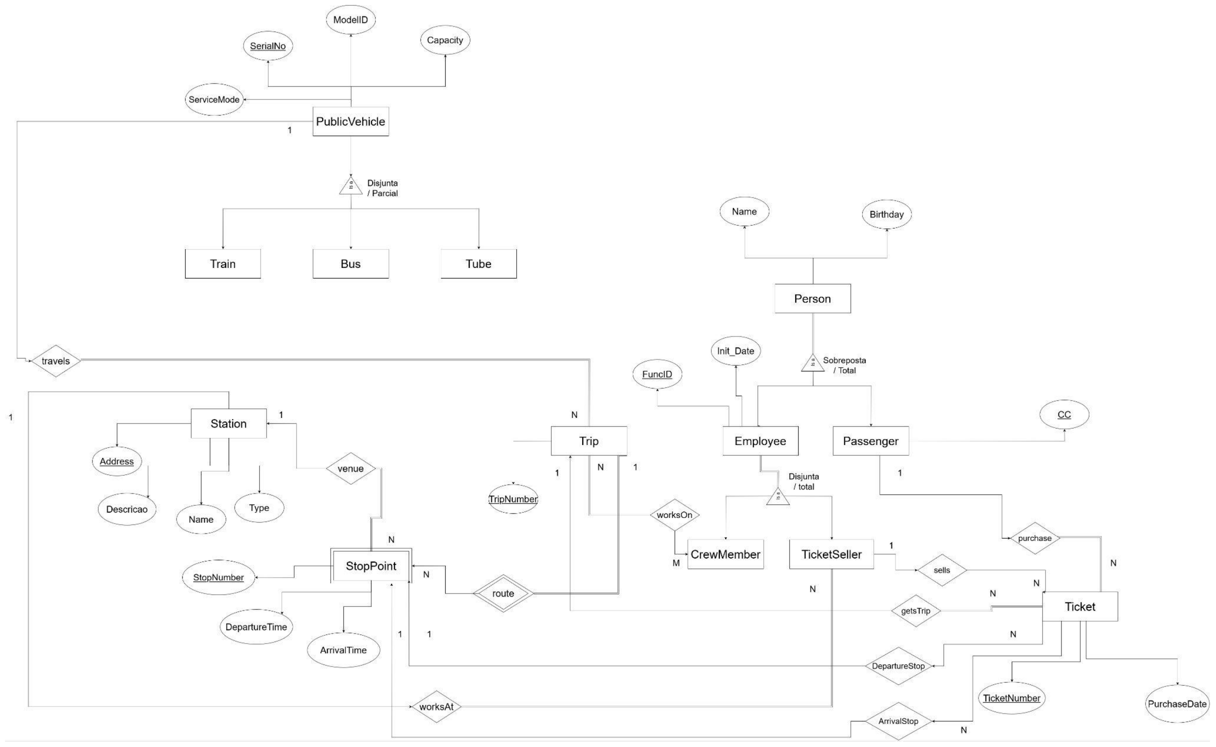
Requirement Analysis

Para tal, é necessário armazenar informação sobre:

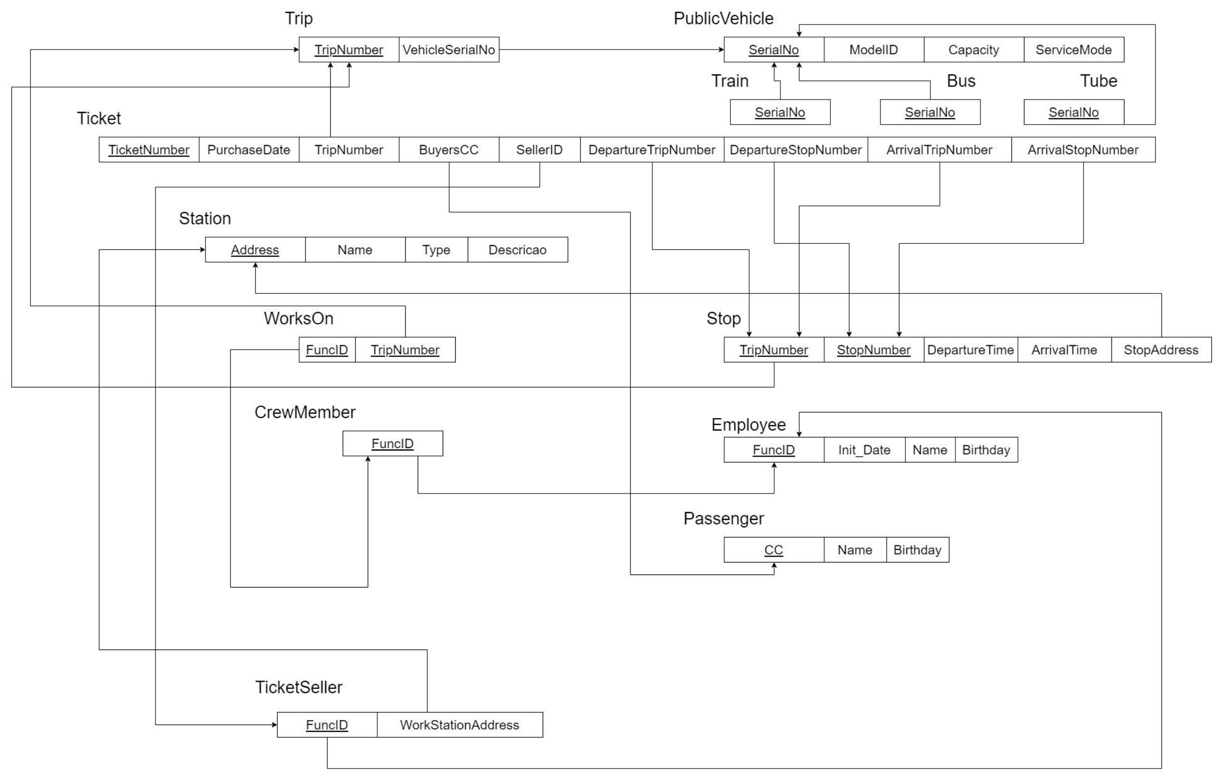
- Transport Vehicles
  - Serial number, vehicle model, capacity, service mode
- Employees
  - Who take part in trips (e.g. driver, ticket validator)
  - Who sell the tickets
  - Any employee should have a unique id
- Passengers
  - Identifier (Citizen Card Number)
  - Basic info like name and birthday
- Stations
  - Address
  - Type (primary, secondary, …)

- Trips
  - Stop points
  - Vehicle which is driving
- Tickets
  - Purchase date
  - Seller / purchaser
  - Associated trip as well as the departure and arrival stop locations and date(time)

Entity-Relationship Diagram



Relational Schema


Trip
TripNumber | VehicleSerialNo
PublicVehicle
SerialNo | ModelID | Capacity | ServiceMode
Train
SerialNo
Bus
SerialNo
Tube
SerialNo
Ticket
TicketNumber | PurchaseDate | TripNumber | BuyersCC | SellerID | DepartureTripNumber | DepartureStopNumber | ArrivalTripNumber | ArrivalStopNumber
Station
Address | Name | Type | Descricao
WorksOn
FuncID | TripNumber
Stop
TripNumber | StopNumber | DepartureTime | ArrivalTime | StopAddress
CrewMember
FuncID
Employee
FuncID | Init_Date | Name | Birthday
Passenger
CC | Name | Birthday
TicketSeller
FuncID | WorkStationAddress

Normalization

All entities are normalized according to the Boyce-Codd Normal Form.

Views

In this project, three views were written (and tested).

- TripQuant
    - All vehicles, their info and their respective number of trips performed
- NumberOfStops
    - All trips, their info and their respective number of stops
- AbstractVehicle
    - Vehicles that were not specialized to existing sub-types

SQL Programming

All batches / queries were properly tested at the end of their respective files. If no error is mentioned then the code blocks are functional.

Stored Procedures

There was implemented one stored procedure that serves as an utility “function”, named getVehicleType.

As the name suggests, the function tells the caller which type of vehicle it is, whether it is a bus, a train or a subway.
The procedure uses basic batch variables and if else statements.

User Defined Functions

Functions were more thoroughly exemplified in this project.
There were three features implemented, one for each function sub-type, as later explained.

Scalar

There were 2 functions, carried out to accomplish the following feature: average employee total worked hours, whether it being for the ticket sellers or crew members.
This feature is extended by the second scalar function, where it calculates the same, however in this case the aggregation is made around the stations (e.g. average total work hours for ticket sellers in each station).
As a last note on this section, the functions assume an average of 40 hours per week of every employee, as well as 50 weeks per week.

Inline Table Valued

There is one function of this type: getTrips, which features trips, given two stop points and two *smalldatetimes* serving as time limits, resulting in a timestamp, which the trip must be enclosed to, regarding the departure and arrival dates.

Multi-Statement Table Valued

The function getRoutes conceptually complements the "getTrips" where it provides all possible transit (two trips) options between point A and B given as arguments, and of course, within a given timestamp.
This function uses a cursor, but only for illustration purposes, because it does not necessarily provide more optimized queries.
Note: For the moment, the query doesn't work, because of one unresolved error of casting string to smalldatetime.

Triggers

Three triggers, where each one asserts that when adding one specialized vehicle (to table bus, for example) its SerialNumber doesn't exist on other tables. Conceptually, this would mean registering a vehicle as a bus when it is already listed as a train. Such an attempt should be rejected, or in SQL terms, the transaction would be rollbacked.

Final Notes

Unfortunately, some topics that were lectured weren't illustrated here, like:
- Indexes
- Forms
- SQL Injection Protection
- Stored procedure as a way to encapsulate the database

These subjects can, however, be implemented in future work.