Student No: 14320715

Description Of Project

The database I designed and implemented describes some of the main operations of a train company such as Irish Rail. In such a company there's Trains, Staff, Stations, Tickets which are all in place for a Service to run on a Route. Information about all of these necessary for the successful operation of a Service is found in the database. The database has a total of 10 tables each of which relate to one another, their attributes are noted below.

The Staff table has the employee number as the primary key. Each employee has a first name, last name, the station id in which they work out of, the pps, the id of the role they have. As the pps is also unique to every employee it is a secondary key.

The StaffRole table has the role id as the primary key. Each role then has a name, a base salary for that role and the number of employees in that role.

The Station table has the station id as the primary key. Each station has a name, an address and the number of employees that are based in that station.

The Train table has the train id as the primary key. Every train with a train id has a train type.

The TrainType table has the train type as the primary key. Each train type has a capacity number and the number of carriages attached to the train.

The Route table has the route id as the primary key. Each route has the id of both the start and end stations and the duration (in minutes).

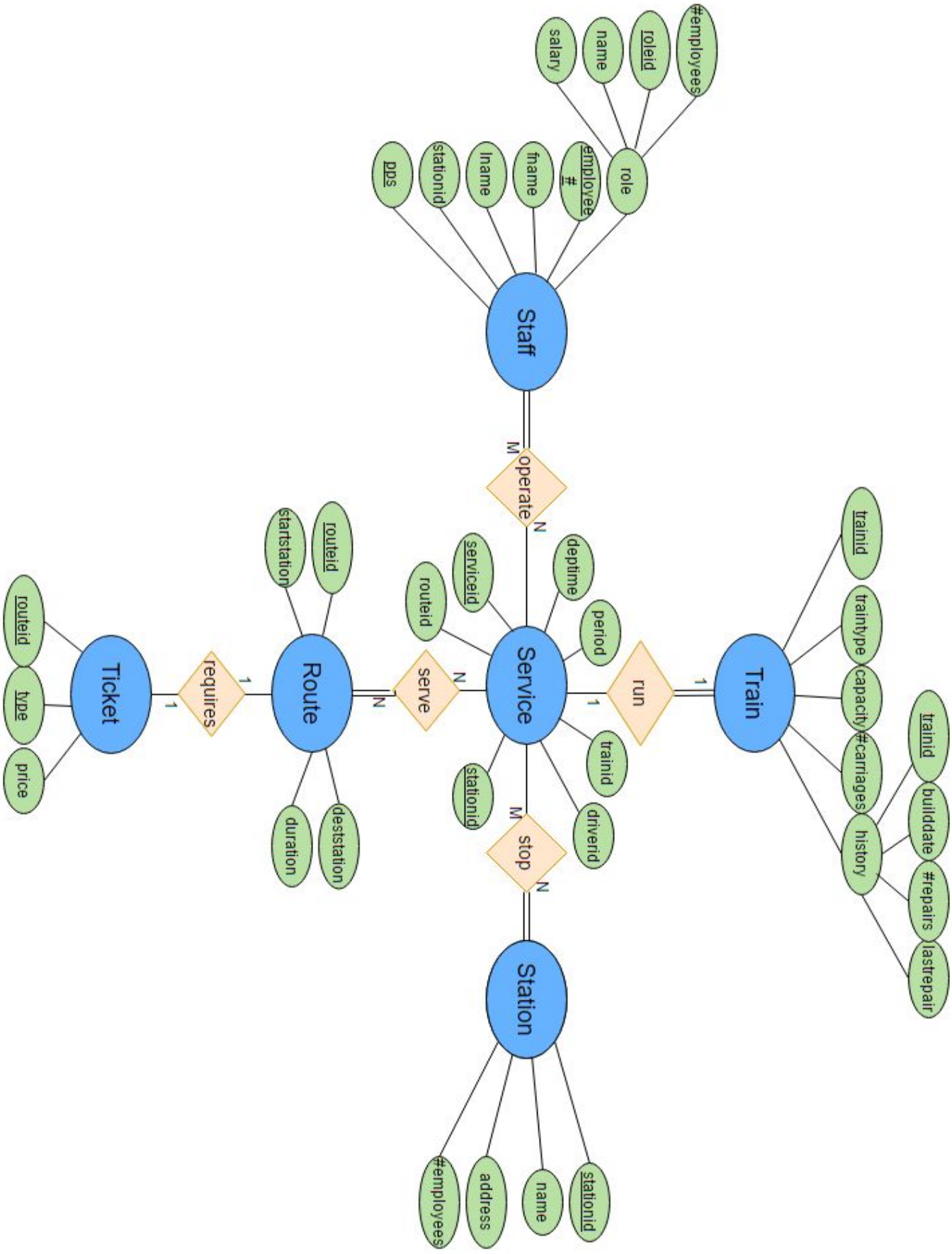
The Service table has the service id and the station id together as the primary key. Each service has a station id of the station it stops at and the time it depts that station. The service represents all the stations the train stops at on a particular route and the time it departs that station at. Eg the service number 4 stops at Pearse, Grand Canal Dock, Blackrock ,Greystones. On the route 44 which starts at Pearse and ends at Greystones.

The ServiceDetail table has the service id as the primary key. Each service also has some detail about it such as the route it services, the period in which it runs (eg MF is Monday-Friday), the id of the train and driver running the service. This table was made separate to Service to avoid unnecessary duplication.

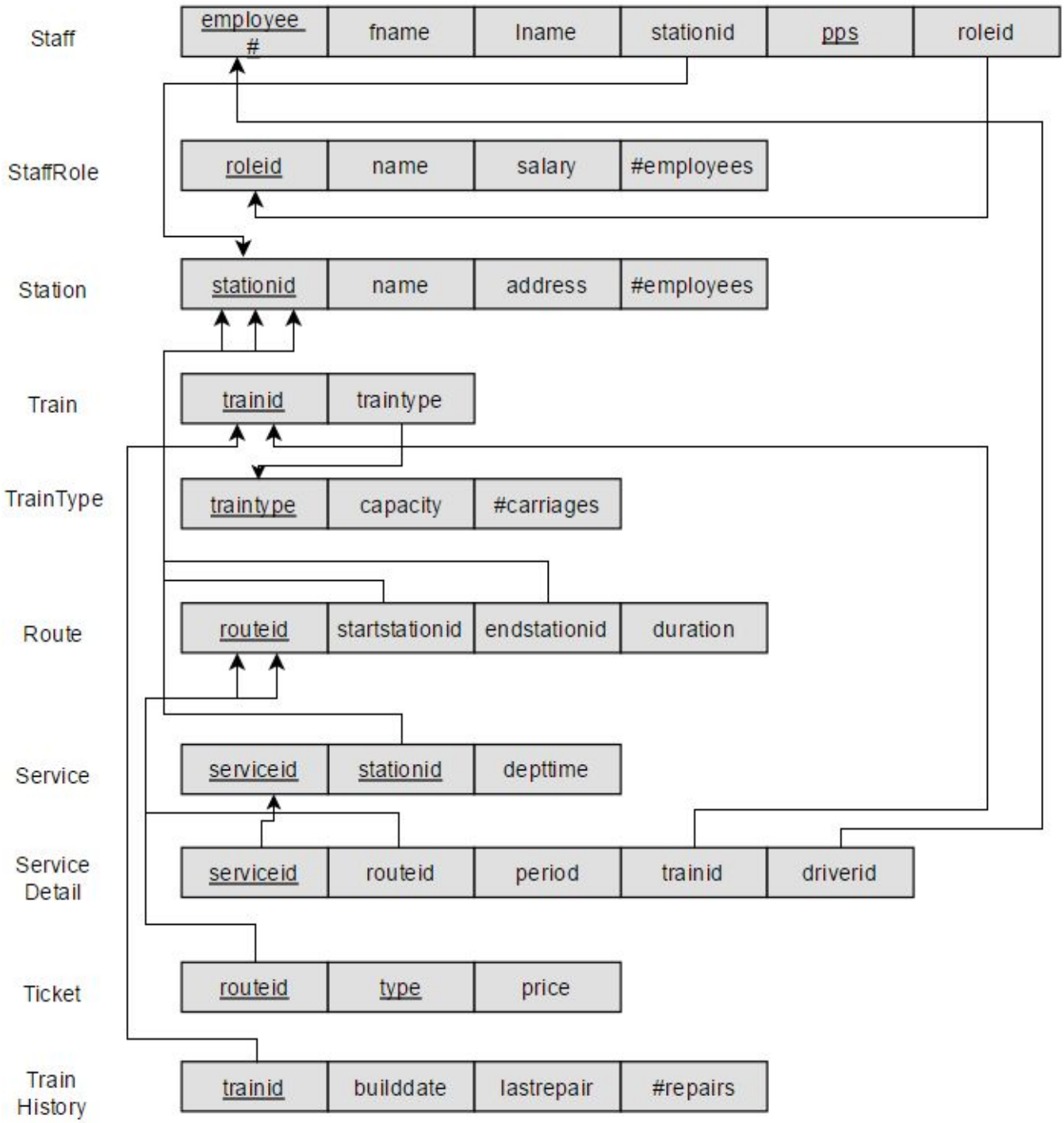
The Ticket table has the route id as the primary key. Each ticket has a type (eg S is for Student) and a price as a decimal.

The TrainHistory table has the train id as the primary key. Each train has a history relating to it which includes the dates of when it was built and of its last repair. It also contains the number of repairs the train has been through.

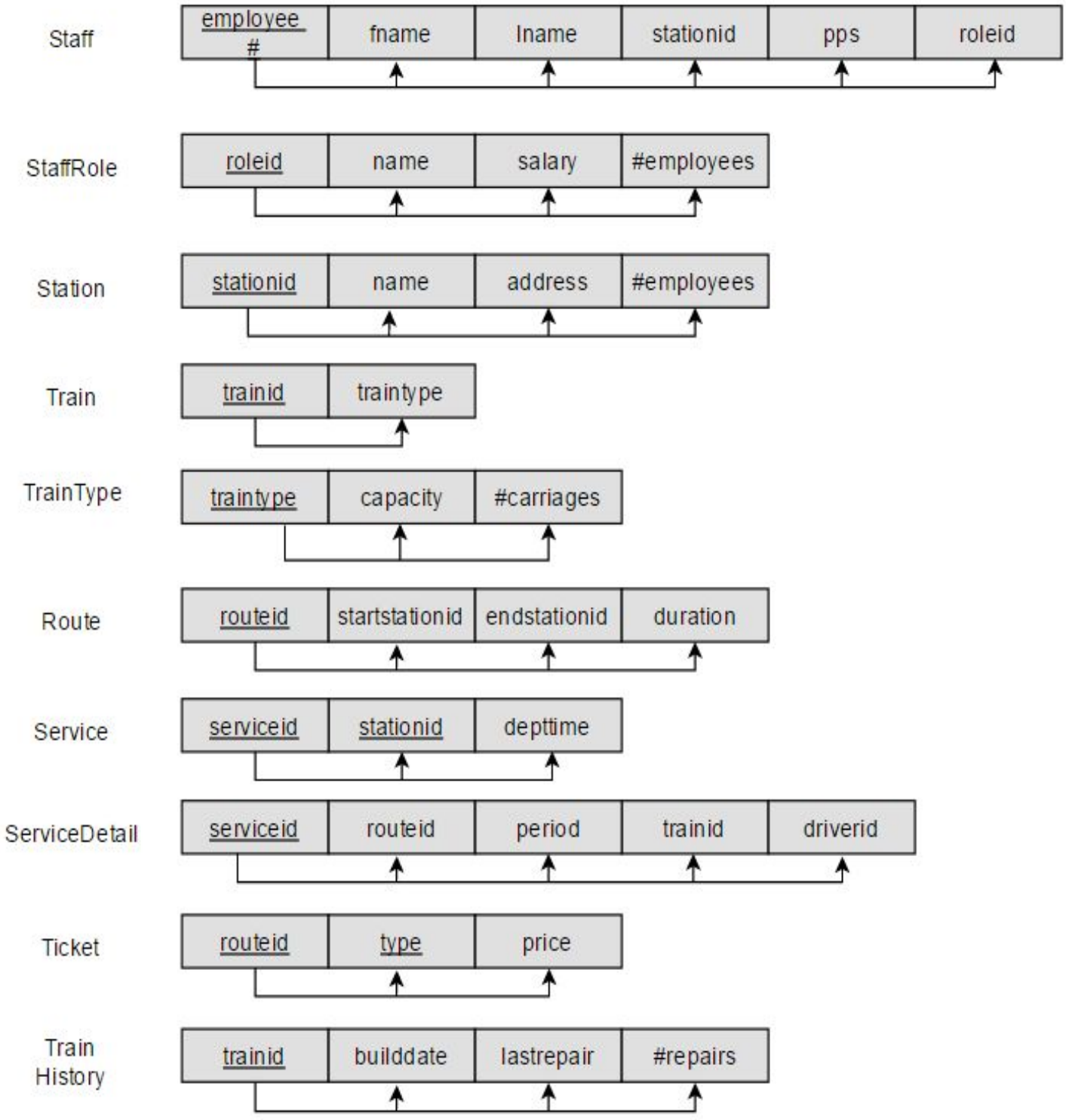
Entity Relationship Diagram



Mapping to Relational Schema



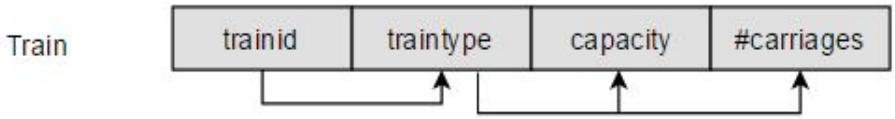
Functional Dependencies



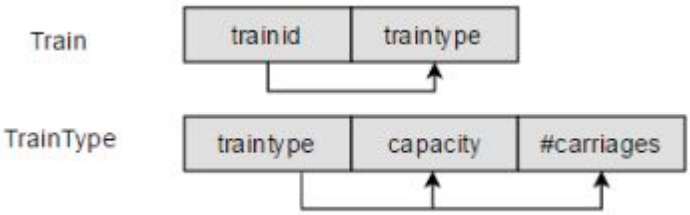
Normalisation

1st Normalised Form: In 1NF all attribute values must be atomic. In this case all attributes are atomic. I.e For every row-column value there exists only one value and not a list or multiple values. The database is valid under 1st Normalised Form.

2nd Normalised Form: In 2NF it is 1NF compliant and every non-key value is fully functionally dependent on the primary key. As seen in the previous Functional Dependencies diagram this is not achieved. To correct this I will split the Train table in 2. Creating a Train and TrainType tables. The original structure was:



The normalised structure is:



In both these tables all attributes are now fully functionally dependent on their primary key (trainid and traintype respectively). Therefore the database is now 2NF compliant too.

3rd Normalised Form: In 3NF it is 2NF compliant and no non-key attributes are transitively dependent upon the primary key. The database fulfills this and is therefore compliant with 3NF.

All these changes made to the Database using the normalisation forms reduced redundancy and data duplication and in turn increasing data integrity.

Semantic Constraints

Semantic constraints were added to the database to ensure its integrity and avoid the corruption of data in its tables.

Entity Integrity Constraints:
All primary key attributes are specified so they can't be null, this is to guarantee that each tuple in the relevant relation can be uniquely identified.

Referential Integrity Constraints:
All foreign key attributes are also specified so they can't be null, this ensures consistency across relative tuples in two relations.

Table Constraints:
The pps attribute in the Staff Table is a secondary key, as every member of staff has a unique employee number but also a unique pps. Therefore if inserting a new employee into the table with a pps already present then that employee is already in the table. There was also a check constraint added to the pps to validate the inputted length is correct. A pps number must be 8 characters long.

The Period attribute in ServiceDetail was given a check constraint to validate it was the correct period value ['MF'(Mon-Fri), 'SA'(Sat), 'SU'(Sun),'BH'(Bank Hol)]. This is to ensure consistency throughout the tuples in this table.

The Type attribute in Ticket was also given a check constraint to validate its value as one of the following: ['A'(Adult), 'C'(Child), 'S'(Student)]. This was also to ensure consistency in the data, as these are the only types of tickets available .

Constraint checks were also put on multiple attributes that acted as counts to ensure they were not <= 0 such as No_Employees in Roles and Stations and not <0 for Capacity and No_Carriages in TrainType and No_Repairs in TrainHistory.

Database Security

Security for this particular database would be important because it supports large operations that require accurate information, but also to protect important private data such as Staff's pps or salary.

Setting up some form of access control was the first step of adding security. By restricting unauthorised access to the database we can also limit potential deliberate corruption. The database will limit the access available to a user based on their user account and privileges.

Database administrators (DBA) will be employed by the company to update, maintain and control the database. They're the only users with the privilege of creating, updating and dropping tables. They will also be able to grant and revoke privileges to other users, this is done using the Grant command. Generally users of the database will be assigned a role and their privileges are determined by that role, this is achieved using the Create Role command. The code for the Grant and Create Role shown below:

```
Create Role Director;
Create Role Manager;
Create Role Employee;
Grant Manager to PierreGignac;
Grant Director to CassandraHella;
```

In this scheme the Directors have access to all of the database. The Managers have access to all of the database barring the information relating to the Directors pps and salary. The Employees have access to all the database the managers do besides all staff pps and salary. These roles create a hierarchy of authorisation to view sensitive information.

Then other users not assigned a role have access to all information Service, ServiceDetail, Route, Station and Ticket.

If a user is found to be misusing or abusing their privileges then the DBA has the right to revoke permission as they see fit. They can also completely banned from the database in special circumstances, also when they leave their position. This is done using the Revoke command. For example if Pierre is found to be sharing information about other employees salaries without good reason he can be revoke the ability to access that information.

```
Revoke Select on StaffRole From PierreGignac;
```

Views & Selects

I implemented two Views both of which include multiple tables. The first one is to list which train driver is on which service by their name. This would look something like a roster, so drivers could easily see what services they were driving:

```
Create View drivers_roster As
Select Staff.Fname, Staff.Lname, ServiceDetail.Service_ID, ServiceDetail.Route_ID
From Staff, ServiceDetail
Where Staff.Employee_No=ServiceDetail.Driver_ID And Staff.Role_ID=1;
```

My Second view was given to give a view of the capacity of the trains running on a particular service:

```
Create View service_capacity As
Select  ServiceDetail.Service_ID,TrainType.Capacity
From TrainType, ServiceDetail, Train
Where ServiceDetail.Train_ID=Train.Train_ID And Train.Train_Type=TrainType.Train_Type;
```

I also implemented two selects which used joins across multiple tables. The first one was to get the price of tickets for a particular given a service's station and dept time. The current select gives the ticket prices for the train leaving station with ID 2 (Pearse) at 10:00am.

```
Select Ticket.Ticket_Type ,Ticket.Price
From Ticket,ServiceDetail,Service
Where Service.Station_ID=2 And Service.Dept_Time='10:00' And ServiceDetail.Service_ID=Service.Service_ID
And Ticket.Route_ID=ServiceDetail.Route_ID;
```

The second select I implemented was to get the Train ID's, type and number of repairs if it currently has over 20 repairs recorded. This could be used to identify trains that may need to be replaced or a potential hazard.

```
Select Train.Train_ID, TrainType.Train_Type, TrainHistory.No_Repairs
From Train, TrainHistory, TrainType
Where TrainHistory.No_Repairs >20 And TrainHistory.Train_ID=Train.Train_ID And Train.Train_Type=
TrainType.Train_Type;
```

Updates

I've shown three updates below, one where the salary of all staff in a role is updated, one where the last name of an employee is changed and another where the price of a child's ticket on a certain route is increased:

```
Update StaffRole Set Salary =42500 Where Role_ID=1;
Update Staff Set Lname='Kelly' Where Employee_No=1023;
Update Ticket Set Price=1.40 Where Route_ID=2 And Ticket_Type='C';
```

Triggers

I implemented two triggers in the database. Both triggers are used to update the number of employees in a particular role when a new employee is added/deleted by incrementing/decrementing the current count. Then I tested the trigger by inserting and deleting a new train driver entry in Staff, I checked the count before and after both commands to ensure the trigger was functioning correctly.

```
Create or Replace Trigger update_noEmployees_Insert
Before Insert on Staff
For each Row
Declare
rId Number;
Begin
 rID := :NEW.Role_ID;
 Update StaffRole
 Set No_Employees = No_Employees +1
 Where Role_ID=rID;
End;
.
Run;

Create or Replace Trigger update_noEmployees_Delete
Before Delete on Staff
For each Row
Declare
rId Number;
Begin
 rID := :OLD.Role_ID;
 Update StaffRole
 Set No_Employees = No_Employees -1
 Where Role_ID=rID;
End;
.
Run;

Select StaffRole.No_Employees
From StaffRole
Where StaffRole.Role_ID=1;

Insert Into Staff Values(1074,'Keith','Hooper',1,'8741557O',1);

Select StaffRole.No_Employees
From StaffRole
Where StaffRole.Role_ID=1;

Delete from Staff
Where Employee_No=1074;

Select StaffRole.No_Employees
From StaffRole
Where StaffRole.Role_ID=1;
```

Appendix

```
--Reset All tables
drop table Staff cascade constraints;
drop table StaffRole cascade constraints;
drop table Station cascade constraints;
drop table Train cascade constraints;
drop table TrainType cascade constraints;
drop table Route cascade constraints;
drop table Service cascade constraints;
drop table ServiceDetail cascade constraints;
drop table Ticket cascade constraints;
drop table TrainHistory cascade constraints;
drop View drivers_roster;
drop View service_capacity;

--Create tables
Create Table StaffRole
(
Role_ID Integer NOT NULL,
Name Varchar(255) NOT NULL,
Salary Integer,
No_Employees Integer,
Primary Key(Role_ID),
Constraint check_no_employees Check(No_Employees>0)
);
Create Table Staff
(
Employee_No Integer NOT NULL,
Fname Varchar(255) NOT NULL,
Lname Varchar(255) NOT NULL,
Station_ID Integer  NOT NULL,
PPS Varchar(8) NOT NULL,
Role_ID Integer NOT NULL,
Primary Key(Employee_No),
Foreign Key(Role_ID) References StaffRole(Role_ID),
Constraint check_pps Check(Length(PPS)=8)
);
Create Table Station
(
Station_ID Integer NOT NULL,
Name Varchar(255) NOT NULL,
Address Varchar(255),
No_Employees Integer,
Primary Key(Station_ID),
Constraint check_no_employees2 Check(No_Employees>0)
);
Create Table TrainType
(
Train_Type Varchar(255) NOT NULL,
Capacity Integer NOT NULL,
```

```
No_Carriages Integer,
Primary Key(Train_Type),
Constraint check_capacity Check(Capacity>=0),
Constraint check_no_carriages Check(No_Carriages>=0)
);
Create Table Train
(
Train_ID Integer NOT NULL,
Train_Type Varchar(255) NOT NULL,
Primary Key(Train_ID),
Foreign Key(Train_Type) References TrainType(Train_Type)
);
Create Table TrainHistory
(
Train_ID Integer NOT NULL,
Build_Date Date NOT NULL,
Last_Repair Date,
No_Repairs Integer,
Primary Key(Train_ID),
Foreign Key(Train_ID) References Train(Train_ID),
Constraint check_no_repairs Check(No_Repairs>=0)
);
Create Table Route
(
Route_ID Integer NOT NULL,
Start_Station Integer NOT NULL,
End_Station Integer NOT NULL,
Duration Integer,
Primary Key(Route_ID),
Foreign Key(Start_Station) References Station(Station_ID),
Foreign Key(End_Station) References Station(Station_ID)
);
Create Table Service
(
Service_ID Integer NOT NULL,
Station_ID Integer NOT NULL,
Dept_Time Varchar(255) NOT NULL,
Primary Key(Service_ID,Station_ID),
Foreign Key(Station_ID) References Station(Station_ID)
);
Create Table ServiceDetail
(
Service_ID Integer NOT NULL,
Route_ID Integer NOT NULL,
Period Varchar(2),
Train_ID Integer NOT NULL,
Driver_ID Integer NOT NULL,
Primary Key(Service_ID),
Foreign Key(Route_ID) References Route(Route_ID),
Foreign Key(Train_ID) References Train(Train_ID),
Foreign Key(Driver_ID) References Staff(Employee_No),
```

```
--Foreign Key(Service_ID) References Service(Service_ID),
Constraint period_check Check (Period in ('MF', 'SA','SU','BH'))
);
Create Table Ticket
(
Route_ID Integer NOT NULL,
Ticket_Type char NOT NULL,
Price Decimal(3,2) NOT NULL,
Primary Key(Route_ID,Ticket_Type),
Foreign Key(Route_ID) References Route(Route_ID) ,
Constraint type_check Check(Ticket_Type in('A', 'C', 'S'))
);

--Alters
Alter Table Staff Add unique(pps);
Alter Table Staff
Add Foreign Key(Station_ID) References Station(Station_ID);

--Inserts
Insert Into StaffRole Values(1,'Train Driver', 40000,24);
Insert Into StaffRole Values(2,'Station Controller', 60000,7);
Insert Into StaffRole Values(3,'Ticket Inspector', 38000,4);
Insert Into StaffRole Values(4,'Manager', NULL,2);
Insert Into StaffRole Values(5,'Director', NULL,2);

Insert Into Station Values(4,'Tara St', 'Tara St, Dublin 2',2);
Insert Into Station Values(1,'Heuston', 'Ushers, Dublin',18);
Insert Into Station Values(2,'Pearse', 'Westland Row, Dublin 2',3);
Insert Into Station Values(12,'Malahide', 'Malahide, Co.Dublin',2);
Insert Into Station Values(31,'Howth Junction', NULL,5);
Insert Into Station Values(3,'Connolly', 'North Dock, Dublin 1',34);
Insert Into Station Values(21,'Bray', NULL,1);

Insert Into Staff Values(1001,'Joesph','Miller',3,'2321344R',1);
Insert Into Staff Values(1023,'Jane','Mac',1,'5634441S',3);
Insert Into Staff Values(1004,'Adam','Fitzgerald',1,'8741344P',1);
Insert Into Staff Values(1014,'Thomas','Morgan',2,'2349044Q',2);
Insert Into Staff Values(1092,'Lea','Mirou',4,'0120044C',3);
Insert Into Staff Values(1002,'Cassandra','Hella',3,'7840534Q',5);
Insert Into Staff Values(1049,'Pierre','Gignac',12,'2420887C',4);

--Insert Into Values();
Insert Into TrainType Values('P22340A',760,6);
Insert Into TrainType Values('P22341B',500,4);
Insert Into TrainType Values('F660',0,0);
Insert Into TrainType Values('P22350A',900,8);
Insert Into TrainType Values('P892Q',232,4);

Insert Into Train Values(44, 'P22340A');
Insert Into Train Values(38, 'P22340A');
Insert Into Train Values(5, 'F660');
```

```
Insert Into Train Values(50, 'P22341B');
Insert Into Train Values(58, 'P22350A');


Insert Into TrainHistory Values(5,'22-march-2009', '11-november-2016', 24);
Insert Into TrainHistory Values(58,'05-dec-2015', NULL, NULL);
Insert Into TrainHistory Values(44,'22-jun-2012', '04-oct-2016', 5);
Insert Into TrainHistory Values(50,'17-jan-2015', NULL, 0);
Insert Into TrainHistory Values(38,'17-may-2010', '29-may-2016',12);

Insert Into Route Values(1,3,21,50);
Insert Into Route Values(2,2,12,28);
Insert Into Route Values(3,3,1,12);
Insert Into Route Values(4,21,3,50);
Insert Into Route Values(8,12,2,28);
Insert Into Route Values(10,1,3,12);

Insert Into Service values(21, 2, '10:00');
Insert Into Service values(21, 4, '10:04');
Insert Into Service values(21, 3, '10:09');
Insert Into Service values(21, 31, '10:20');
Insert Into Service values(21, 12, '10:28');
Insert Into Service values(45, 1, '08:18');
Insert Into Service values(45, 3, '08:31');
Insert Into Service values(87, 3, '17:40');
Insert Into Service values(87, 4, '17:43');
Insert Into Service values(87, 21, '18:30');
Insert Into Service values(101, 1, '20:00');
Insert Into Service values(101, 3, '20:12');
Insert Into Service values(12, 21, '13:40');
Insert Into Service values(12, 4, '13:43');
Insert Into Service values(12, 3, '14:30');

Insert Into ServiceDetail Values(21,2,'MF',38,1001);
Insert Into ServiceDetail Values(45,10,'SA',38,1004);
Insert Into ServiceDetail Values(87,1,'MF',58,1004);
Insert Into ServiceDetail Values(101,10,'SU',44,1001);
Insert Into ServiceDetail Values(12,4,'BH',50,1004);

Insert Into Ticket Values(2,'A',3.60);
Insert Into Ticket Values(2,'C',1.60);
Insert Into Ticket Values(2,'S',2.20);
Insert Into Ticket Values(4,'A',4.90);
Insert Into Ticket Values(4,'C',3.00);
Insert Into Ticket Values(4,'S',4.10);

--Security Measures
Create Role Director;
Create Role Manager;
Create Role Employee;
```

```
Grant Manager to PierreGignac;
Grant Director to CassandraHella;

Revoke Select on StaffRole From PierreGignac;
--Views
Create View drivers_roster As
Select Staff.Fname, Staff.Lname, ServiceDetail.Service_ID, ServiceDetail.Route_ID
From Staff, ServiceDetail
Where Staff.Employee_No=ServiceDetail.Driver_ID And Staff.Role_ID=1;

Create View service_capacity As
Select  ServiceDetail.Service_ID,TrainType.Capacity
From TrainType, ServiceDetail, Train
Where ServiceDetail.Train_ID=Train.Train_ID And Train.Train_Type=TrainType.Train_Type;

--Selects
Select Ticket.Ticket_Type ,Ticket.Price
From Ticket,ServiceDetail,Service
Where Service.Station_ID=2 And Service.Dept_Time='10:00' And ServiceDetail.Service_ID=Service.Service_ID
And Ticket.Route_ID=ServiceDetail.Route_ID;

Select Train.Train_ID, TrainType.Train_Type, TrainHistory.No_Repairs
From Train, TrainHistory, TrainType
Where TrainHistory.No_Repairs >20 And TrainHistory.Train_ID=Train.Train_ID And Train.Train_Type=
TrainType.Train_Type;

--Updates
Update StaffRole Set Salary =42500 Where Role_ID=1;

Update Staff Set Lname='Kelly' Where Employee_No=1023;

Update Ticket Set Price=1.40 Where Route_ID=2 And Ticket_Type='C';

--Triggers
Create or Replace Trigger update_noEmployees_Insert
Before Insert on Staff
For each Row
Declare
rId Number;
Begin
 rID := :NEW.Role_ID;
 Update StaffRole
 Set No_Employees = No_Employees +1
 Where Role_ID=rID;
End;
.
Run;
```

```
Create or Replace Trigger update_noEmployees_Delete
Before Delete on Staff
For each Row
Declare
rId Number;
Begin
 rID := :OLD.Role_ID;
 Update StaffRole
 Set No_Employees = No_Employees -1
 Where Role_ID=rID;
End;
.
Run;

--Test Triggers
Select StaffRole.No_Employees
From StaffRole
Where StaffRole.Role_ID=1;

Insert Into Staff Values(1074,'Keith','Hooper',1,'8741557O',1);

Select StaffRole.No_Employees
From StaffRole
Where StaffRole.Role_ID=1;

Delete from Staff
Where Employee_No=1074;

Select StaffRole.No_Employees
From StaffRole
Where StaffRole.Role_ID=1;
```