NHS Hospitals and Waiting Lists

Salman Hussain

Requirement Analysis

Background

The UK Government has requested our team to design a new database to address the issues within the NHS which relates to the patient data management and waiting lists. Around 10,000 records were lost or inconsistent which caused severe delays in patient and safety risks.

It is also currently known that the current NHS system's data sharing is a major issue: many hospitals, GPs and consultants are not receiving up to date appointments or information on time.

The aim of the new database system is to improve the efficiency and the accuracy of waiting list monitoring and where possible assign resources and reduce waiting times across the UK. The new database system must also represent the NHS's regional hierarchy: 4 countries, 48 counties and 317 districts, where each have their own population and health records.

The database system must also support how many patients are seeking help, and these must all be categorised into groups sorted by age and employment status.

Some queries that the database system must be able to use are:

- To identify areas with high health demand
- To be able to monitor waiting lists for each county

Across the entire database system, data security and confidentiality must be held. Hence, secure access control is needed to ensure only verified members can access the database.

Assumptions

- The database system will store the data of patients, appointments, admissions and waiting lists accurately and consistently.
- Data must be shared securely between only authorised members of the NHS. This includes GPs, hospitals and consultants.
- Waiting list information should be updated all the time to maintain current healthcare demand and consistency.
- The design should be flexible to allow requested by the government.
- There should be a way in which the government can allocate resources dynamically to areas with higher waiting lists to reduce delays.
- The database should cover all NHS hospitals, GPs and clinics.

The ER Model

Entities and Relationships

The UK government requires a database system to keep track of NHS patient records. The Government has specified for the database to represent a regional structure of the UK

- The UK is to be grouped by its four countries where each country has a unique ID and a name.
- Every country is divided into several counties, and every county has a unique ID, name. Every county is divided into districts where each district has a unique ID, name, population.
- Every District contains one or more Hospitals. Each Hospital has a unique ID, name, type (e.g Hospital, GP, Clinic) and address located within a specific District.
- This database also keeps track of patients. Each patient will have a unique ID, name, employment status, date of birth and an address which links the patient to a specific District.
- Patients can also be admitted for treatment. Each admission will contain a unique ID, start date, discharge date and is linked to the patient and Hospital where the admission occurred.
- A Patient can have several appointments. Each appointment has a unique ID, type of appointment and the date it took place.
- The database contains waiting list entries. A procedure has a unique ID, name and category. The waiting list links the patient to the procedure they are waiting for at a certain Hospital. The waiting list contains a unique ID, date added to the waiting list and patient's priority level.
- A First Minister is linked to one country and contains a unique ID, name, start and end dates of their term. A First Minister may issue several Minister Statements about healthcare or waiting times. Each statement contains a unique ID, topic, content of the statement and the date it was made.

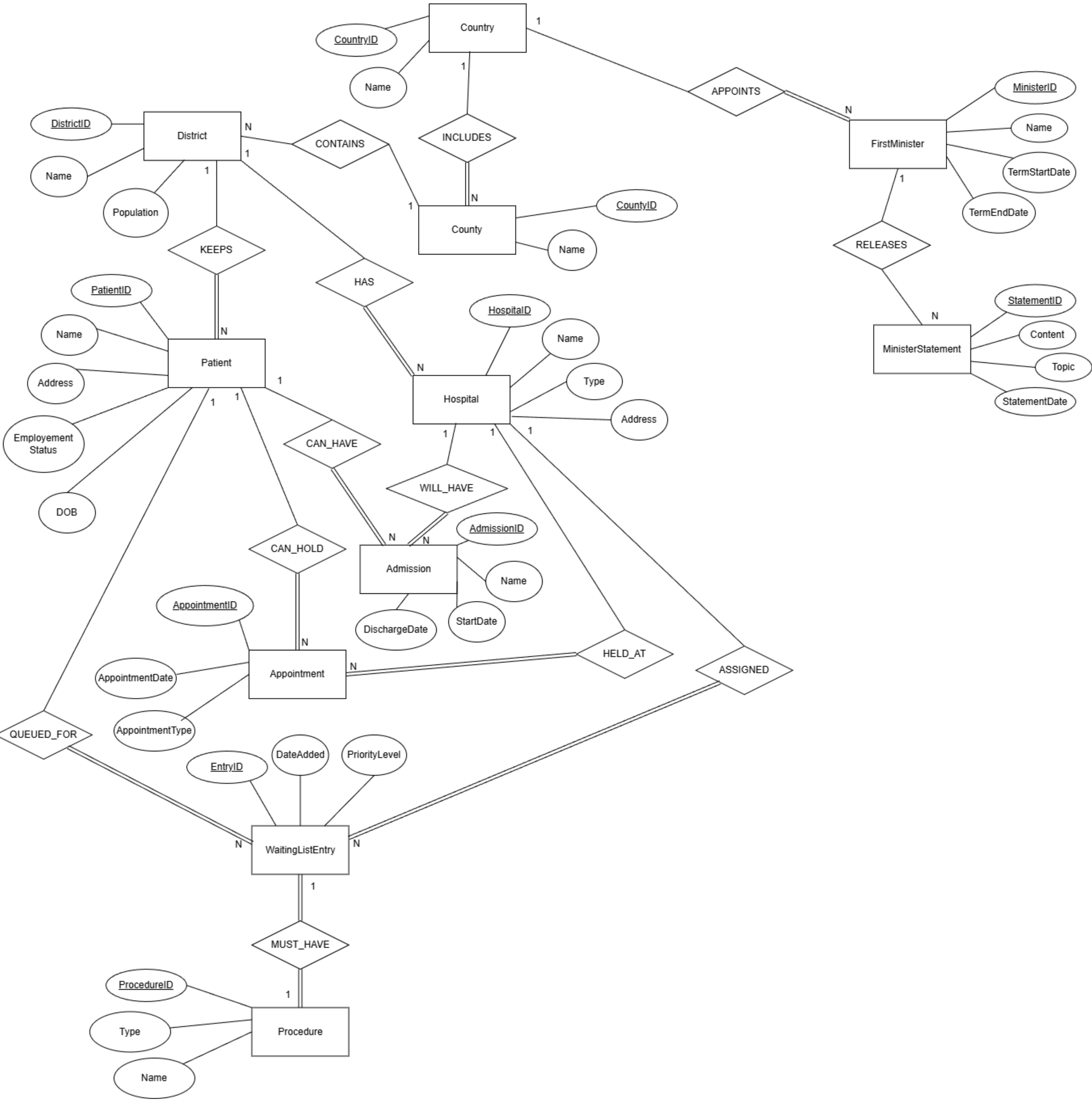
Initial Design: Entity Types

- **Country** *(CountryID, Name)*
- **County** *(CountyID, Name)*
- **District** *(DistrictID, Name, Population)*
- **Hospital** *(HospitalID, Name, Type, Address)*
- **Patient** *(PatientID, Name, EmploymentStatus, DOB, Address)*
- **Admission** *(AdmissionID, StartDate, DischargeDate)*
- **Appointment** *(AppointmentID, AppointmentType, AppointmentDate)*
- **Procedure** *(ProcedureID, Name, Type)*
- **WaitingList** *(EntryID, DateAdded, PriorityLevel)*
- **FirstMinister** *(MinisterID, Name, TermStartDate, TermEndDate)*
- **MinisterStatement** *(StatementID, Topic, Content, StatementDate)*

Refining Design: Relationships

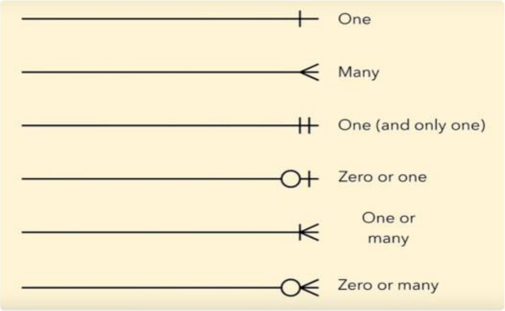
INCLUDES 1:N (between Country, County)
CONTAINS 1:N (between County, District)
KEEPS 1:N (between District, Patient)
HAS 1:N (between District, Hospital)
CAN_HAVE 1:N (between Patient, Admission)
WILL_HAVE 1:N (between Hospital, Admission)
CAN_HOLD 1:N (between Patient, Appointment)
MUST_HAVE 1:1 (between WaitingList, Procedure)
HOLDS 1:N (between Hospital, Appointment)
ASSIGNED 1:N (between Hospital, WaitingList)
QUEUED_FOR 1:N (between Patient, WaitingList)
APPOINTS 1:N (between Country, FirstMinister)
RELEASES 1:N (between FirstMinister, MinisterStatement)

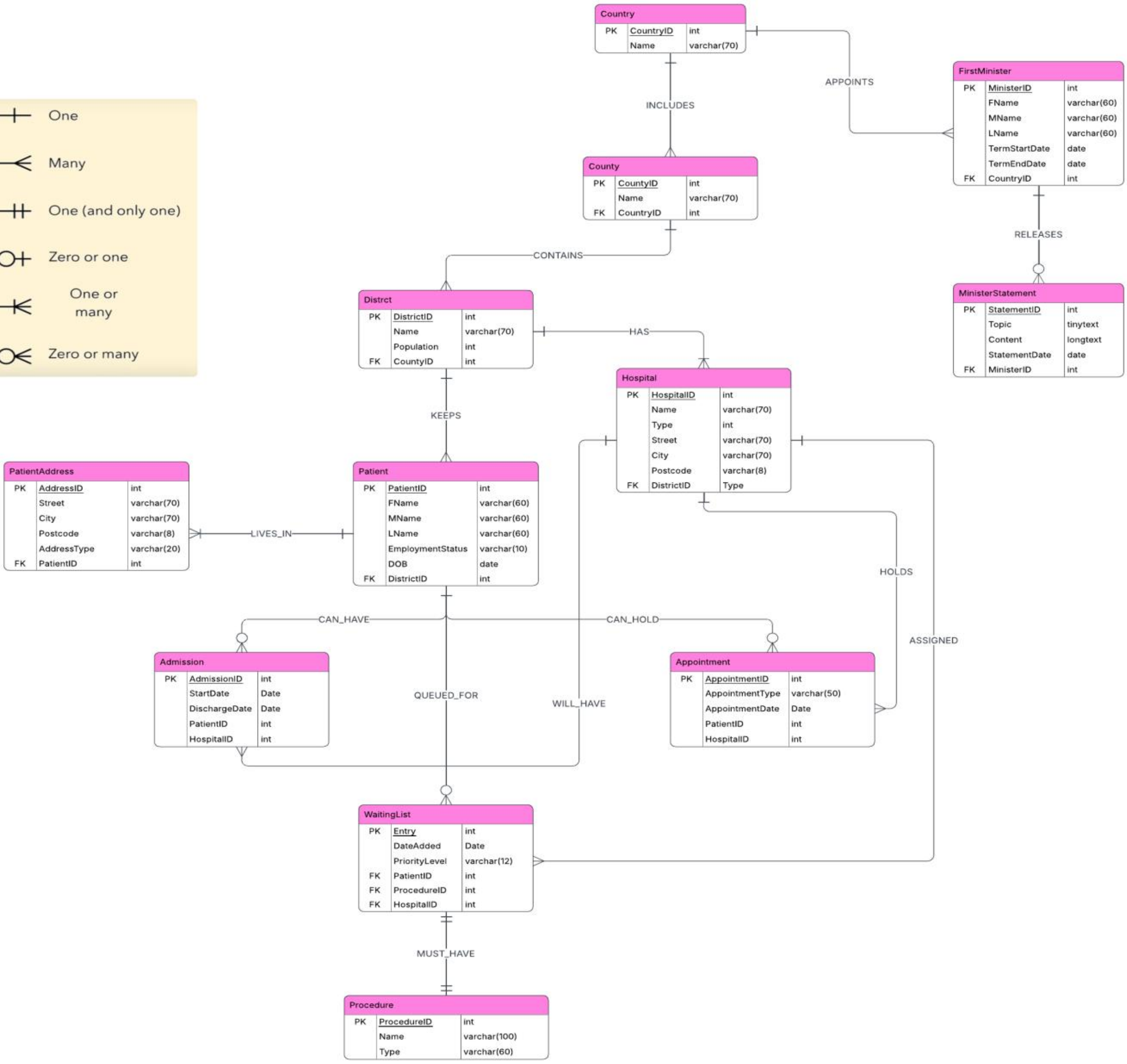
The ER Diagram



Mapping Steps & Logical Model

Logical Model





Country		
PK	CountryID	int
	Name	varchar(70)

FirstMinister		
PK	MinisterID	int
	FName	varchar(60)
	MName	varchar(60)
	LName	varchar(60)
	TermStartDate	date
	TermEndDate	date
FK	CountryID	int

County		
PK	CountyID	int
	Name	varchar(70)
FK	CountryID	int

MinisterStatement		
PK	StatementID	int
	Topic	tinytext
	Content	longtext
	StatementDate	date
FK	MinisterID	int

Distrct		
PK	DistrictID	int
	Name	varchar(70)
	Population	int
FK	CountyID	int

Hospital		
PK	HospitalID	int
	Name	varchar(70)
	Type	int
	Street	varchar(70)
	City	varchar(70)
	Postcode	varchar(8)
FK	DistrictID	Type

PatientAddress		
PK	AddressID	int
	Street	varchar(70)
	City	varchar(70)
	Postcode	varchar(8)
	AddressType	varchar(20)
FK	PatientID	int

Patient		
PK	PatientID	int
	FName	varchar(60)
	MName	varchar(60)
	LName	varchar(60)
	EmploymentStatus	varchar(10)
	DOB	date
FK	DistrictID	int

Admission		
PK	AdmissionID	int
	StartDate	Date
	DischargeDate	Date
	PatientID	int
	HospitalID	int

Appointment		
PK	AppointmentID	int
	AppointmentType	varchar(50)
	AppointmentDate	Date
	PatientID	int
	HospitalID	int

WaitingList		
PK	Entry	int
	DateAdded	Date
	PriorityLevel	varchar(12)
FK	PatientID	int
FK	ProcedureID	int
FK	HospitalID	int

Procedure		
PK	ProcedureID	int
	Name	varchar(100)
	Type	varchar(60)

Mapping Steps

The mapping process began by converting each strong entity within the ER model into a table. For each entity (Country, County, District, FirstMinister, MinisterStatement, Hospital, Patient, PatientAddress, Admission, WaitingList, Procedure, Appointment) a separate table was created in the relational model**.**

We then identified possible key attributes and began to find attributes that can be uniquely identified by the entity. After identifying these primary keys to represent its records (for example, CountryID is the primary key for the entity Country), all the remaining attributes were added as non-key attributes within those tables.

One-to-one relationships were implemented by inserting a primary key of one entity as a foreign key in the other entity. This can also be done on either side depending on the total participation.

For example:

- Each waiting list must be associated with a single Procedure. This ensures that either we can have the primary key EntryID included as a foreign key within the Procedure entity or have the primary key ProcedureID inserted into the WaitingList entity to link each waiting list record to its procedure.

One-to-many relationships were implemented by placing the primary key of the '1' side of the entity as a foreign key to the 'N' side of the entity.

For example:

- Many Districts belong to one County, thus the primary key CountyID will be included as a foreign key within the District entity.
- One district also contains many Hospitals, which means that the primary key DistrictID will be a foreign key within the Hospital entity.

Normalisation

To achieve 1NF, all attributes were made atomic. For example, the Name attribute in **Patient** and **FirstMinister** was split into FName, MName and LName.

Is our table in 1NF?

- The above database is in first normal form (1NF) because all attributes are atomic, which means no attribute is composite.
- All entities have a primary key, noted by the underline.
- Every column name is unique.
- All multivalued fields have been moved to a separate table.

Is our table in 2NF?

- Every non-key attribute is functionally dependant on the primary key of its entity.
    - For example, In WaitingList, the field PriorityLevel is fully functionally dependent on the primary key EntryID.

Is our table in 3NF?

- There are no transitive dependencies within our entities. No non key attributes will functionally depend on another non-key attribute.
    - For example, if we were to place the CountyName within the Patient entity, this means that CountyName will depend on the DistrictID, rather than the PatientID, and so will violate 3NF.

The Process of 2NF to 3NF – Example Snippet

Before 3rd Normal Form

Suppose we had:
Hospital (HospitalID, Name, Type, Street, City, Postcode, DistrictName, DistrictPopulation)

Here, DistrictPopulation depends on DistrictName, creating a transitive dependency.

The Solution

We can split the Hospital entity into a separate entity for the District information (population and name). A reference can then be held within the Hospital entity.

- **District** *(<u>DistrictID</u>, DistrictName, DistrictPopulation, CountryID)*

Mapping:

- DistrictID is a foreign key in Hospital where there is a total participation of Hospital in District.

Hospital *(<u>HospitalID</u>, Name, Type, Address, DistrictID)*
District *(<u>DistrictID</u>, DistrictName, DistrictPopulation, CountryID)*

Why it is 3NF?

Hospital

- Primary Key: HospitalID
- All non-key attributes (Name, Type, Address, DistrictID) now all depend on HospitalID
- No non-key attributes determine another non-key attribute in Hospital therefore this table is in third normal form.

District

- Primary Key: DistrictID
- All non-key attributes (DistrictName, DistrictPopulation) all depend on DistrictID
- No non-key attributes determine another non-key attribute in District therefore this table is also in third normal form

Final Result: No transitive dependencies remain in these entities.