# Design Issues and Version Control

## Data cleansing

When it came to cleansing the data in the excel spreadsheet, I realised that there were some discrepancies and data errors. Typical data errors were duplicate NHI’s for the patient. It had to be unique and only one can exist, so I had to remove some of those patients so that there was only one unique NHI for each patient. I also removed a column that included random dates that had no connection to the system. Some of these dates were unrealistic as well so that was removed after sanity checking. I also cleared the patients that were not health target eligible as specified, which should not be there if they are not eligible. I also found there to be some missing dates in FSA Date for example, so what I did was insert a date so that the days waiting from referral date field could calculate a proper day instead of an undefined value. Another field I did not include was year-month, as this was already covered in the referral date.

## Choice of entities

When it came to choosing entities, I wanted to make it as realistic as I could for a real hospital system, with a surgeon having an appointment with the patient for example. Other entities may be connected as well, but my knowledge of hospital systems is not vast enough and could only make tables with the fields provided. The entities I chose were Patient, Appointment, Surgeon, Referral, Referree, Profession, and Department. Patient I chose because they are the client, and the main part of the system. Surgeon was the next thing which I knew would have some sort of relationship with patient, and then other bridging tables such as appointment came in. More of these entities were added over time to expand the system further with the fields provided as well as new calculated ones. I chose to include referral date in appointment so that I could use FSA Date which existed in appointment to calculate the days waiting from referral date in the excel spreadsheet. I also included DOB and added to waitlist date in referral so that I can have the patient age at referral be calculated from those two fields.

## Choice of attributes

In terms of attributes of entities, I first sought out a primary key for each one such as SurgeonID, NHI, AppointmentID, DepartmentID, ReferralID etc. From here I could establish other attributes such as firstname and lastname, which were derived from patient name, surgeon name and referree name. DOB was another choice for patient as well as gender and health target eligible. If I were to do this assignment again I would remove health target eligible as it is not needed if they are eligible. For surgeon I did the same having a surgeon ID as the primary key, then attributes related to surgeon such as the composite attribute of name becoming firstname lastname. Department also had a department ID and just a name. Appointment was to have an ID as well as well as FSA date as it seemed fitting to go there. For referral I chose to have a referral ID and the date and add other foreign key attributes as primary keys of other tables.

## Choice of keys

In the **Department** table, I chose departmentID as the primary key to uniquely identify the department.

In the **Patient** table I chose the existing field NHI as the primary key to uniquely identify the patient, instead of creating an entirely new patientID which is not needed.

In the **Profession** table, I chose to make professionID the primary key to uniquely identify the profession.

In the **Appointment** table, I chose appointmentID to be the primary key, so that each appointment is unique. I also decided to have the patient NHI in the table as a foreign key, so that we know what appointment belongs to which patient. Surgeon ID was another foreign key in this instance where you know what patient has what appointment with what surgeon.

With **Referral**, I chose referralID to be the primary key so that each referral is a unique referral. NHI I chose to be a foreign key so that we know what referral belongs to which patient. SurgeonID I chose to be foreign key as well so that we know what patient is going to be seeing which surgeon as referred by a referree. Referree ID I chose to be included as a foreign key as well in referral so we know which referree made that specific patients referral.

In **Referree**, I chose referree ID to be the primary key to uniquely identify the referree. I also chose to have profession ID as the foreign key of referree so that we know if the referree is GP or internal etc.

In the **Surgeon** table I chose surgeonID as the unique identifier for that surgeon, as well as a foreign key of departmentID so that we know what department that surgeon belongs to.

## Connectivity/Relationships

**Department-Surgeon:** A surgeon can belong to one and only one department, where as a department can have one to many surgeons.

**Surgeon-Appointment:** A surgeon can have one or many appointments, but an appointment can belong to one and only one surgeon.

**Appointment-Patient:** A patient can have one and only one appointment, and an appointment can belong to one and only one patient.

**Patient-Referral:** A patent can have one and only one referral, and a referral can belong to one and only one patient.

**Referral-Referree:** A referral can be written by one and only one referree, where a referree can write one or many referrals.

## Use of composite/bridging entities

Originally, I had a **Patient-Surgeon** relationship, but I realised in a real hospital situation, a surgeon can attend to many patients, and a patient can see many surgeons. This posed a many to many relationship, which relational database cannot handle, so I implemented an **Appointment** bridging table, which seemed to make sense.

## Extent of normalisation

When I got to the normalisation process, I found that there were many fields to handle at once. I found that some of the foreign keys were not necessary to generate a partial relationship with a field you want to find out from that particular table. So my normalisation process may have gotten lost in translation as I had an idea of what I wanted the system to look like, and the many different ways one could have gone about building it.

## Version Control

Version control has always been a fantastic tool for me in terms of keeping track of changes I have made, and maintaining a consistent work flow. This time around because of lockdown it was not a routine to commit changes as regularly, so the consistency of committing to the repository was not great. This Is a personal issue I had this time around with using version control, but would always use it anyways.