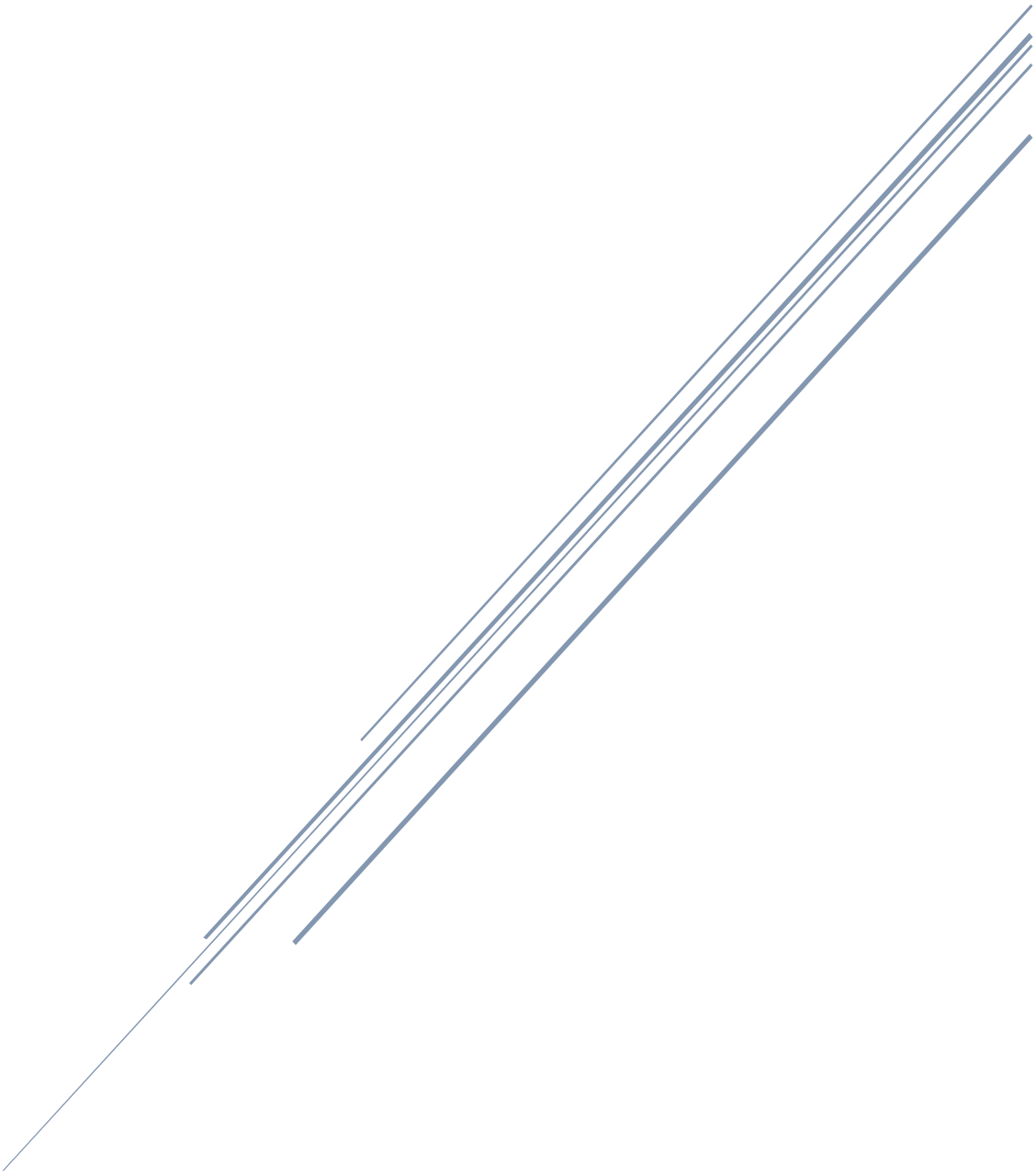
HSE

Database Systems (Lec. Anne Dunphy) - 2023



SSD2 - SETU (WIT)
Piotr Płaczek (20097618)

1 CONTENTS

INTRODUCTION

This report is designed to provide an overview of the database design and development I am undertaking as part of my CA for the Database Systems module.

For this project, I have decided to use the HSE as my model and create a database that implements a vague representation of how I imagine the HSE manages its data.

2 ORGANISATION DESCRIPTION

The organisation in question is the Health Service Executive (HSE) in Ireland. The HSE is a government body responsible for providing healthcare services to the public of Ireland (HSE, 2023). The HSE provides a wide range of services in the areas of public health, primary care, mental health, disability services, community care, and acute hospital care. The HSE is also responsible for the regulation and management of healthcare services, and the development and implementation of health policies.

The major functions of the HSE include providing healthcare services to the Irish public, the management of health services and the development of health policy (HSE, 2022). The benefits of implementing a database for the HSE include improved data collection and management, improved communication between different departments and increased efficiency in the delivery of services. The database should also be able to provide timely and accurate information to support decision-making, and to improve the quality and safety of healthcare services.

Data was collected for this database through a combination of interviews, surveys and website research, you can find them in the references section at the end of this report.

The database should include data on the following items:

1. **Patient information**: This includes basic information such as name, contact details, medical history, and insurance details.

2. Services information: This includes information about the services that the HSE provides, such as primary care, mental health, and disability services.

3. **Staff information**: This includes information about the staff employed by the HSE, such as name, contact details, qualifications, and job roles.

4. Financial information: This includes information about the financial cost of providing services, and the revenue generated by the HSE.

5. Health policies: This includes information about the health policies developed and implemented by the HSE.

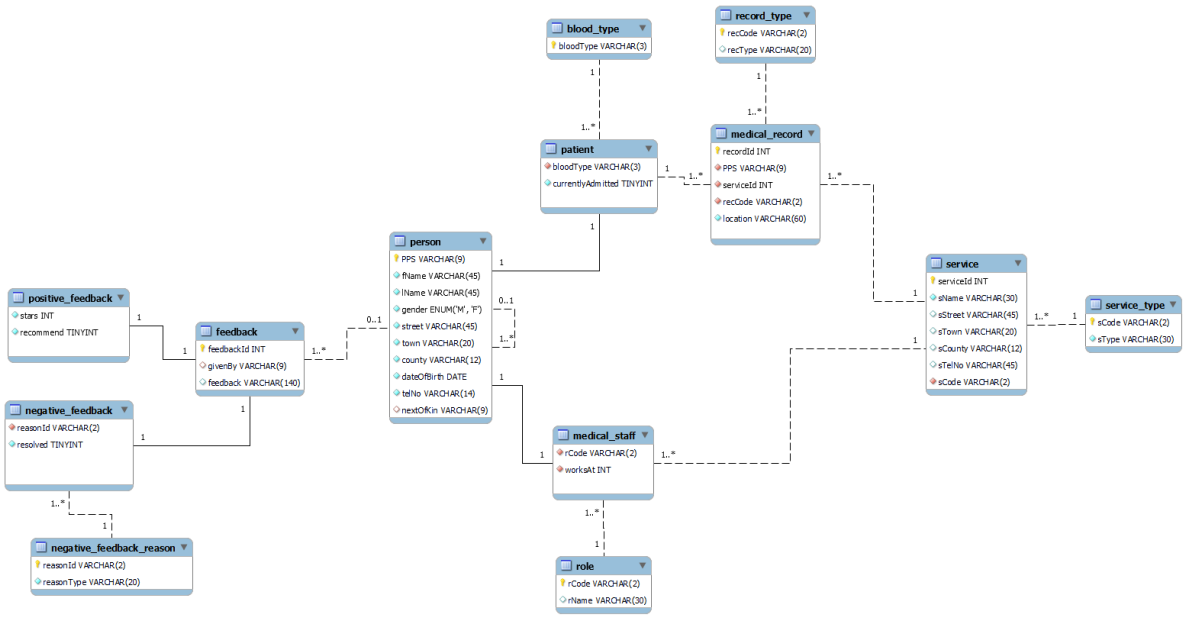
6. Outcomes and performance: This includes information about the outcomes of healthcare services and their performance. This will help the HSE to identify areas for improvement and to ensure that services are effective and efficient. Medical Records

7. Quality and safety: This includes information about the quality and safety of services provided by the HSE. This will help to ensure that the services provided are of the highest quality and that patients are safe.

8. **Patient feedback**: This includes information on the feedback from patients and other stakeholders about the services provided by the HSE. This will help the HSE to identify areas for improvement and to ensure that services are of the highest quality.

9. The database should also include the ability to store and analyse data, so that the HSE can easily access and analyse data in order to make informed decisions. The database should also be **secure and reliable**, so that all data is kept safe and secure.

3 EER MODEL



4 TABLES

Different super/sub relationships represented below in storage section

```
({MANDATORY, AND} person -> patient, medical_staff)

person {PPS, fName, lName, gender, street, town, county,
dateOfBirth, telNo, nextOfKin}
Primary key: PPS
Foreign key: nextOfKin references person (PPS) ON DELETE RESTRICT ON
UPDATE CASCADE
```

```
role {rCode, rName}
Primary key: rCode
```

```
service_type {sCode, sType}
Primary key: sCode
```

```
service {serviceId, sName, sStreet, sTown, sCounty, sTelNo, sCode}
Primary key: serviceId
Foreign key: sCode references service_type (sCode)
```

```
medical_staff {PPS, rCode, worksAt}
Primary key: PPS
Foreign key: PPS references medical_staff (PPS)
Foreign key: rCode references role (rCode)
Foreign key: worksAt references service (serviceId)
```

```
({MANDATORY, OR} feedback -> positive_feedback, negative_feedback)

feedback {feedbackId, givenBy, feedback}
Primary key: feedbackId
Foreign key: givenBy references person (PPS)
```

```
blood_type {bloodType}
Primary key: bloodType
```

```
patient {PPS, bloodType, currentlyAdmitted}
Primary key: PPS
Foreign key: PPS references person (PPS)
Foreign key: bloodType references blood_type (bloodType)
```

```
record_type {recCode, recType}
Primary key: recCode
```

```
medical_record (recordId, PPS, serviceId, recCode, location)
Primary key: recordId
Foreign key: serviceId references service (serviceId)
Foreign key: recCode references record_type (recCode)
Foreign key: PPS references patient (PPS)
```

```
positive_feedback (feedbackId, stars, recommend)
Primary key: feedbackId
Foreign key: feedbackId references feedback (feedbackId)
```

```
negative_feedback_reason (reasonId, reasonType)
Primary key: reasonId
```

```
negative_feedback (feedbackId, reasonId, resolved)
Primary key: feedbackId
Foreign key: feedbackId references feedback (feedbackId)
Foreign key: reasonId references negative_feedback_reason (reasonId)
```

5 STORAGE REPRESENTATION

The size of the fields I have decided on for each table can be seen in the ER diagram.

Here is an example of how I decided on these:

service {serviceId, sName, sStreet, sTown, sCounty, sTelNo, sCode}

- serviceId – This is just an ID that auto increments so it can be an INT, assuming a generous 100 services (dentist offices, doctors offices, etc.) per county that's 100x26 = 2600. Number (4) should be plenty
- sName – Name of the service, 30 chars should be plenty for even the longest name
- sStreet – Street name, 45 should be plenty haven't seen one longer
- sTown – Town name, 20 should be enough
- sCounty – County, no county longer than 12 chars[1]
- sCode – 2 chars is enough, there aren't that many e.g. dentist, doctor

Apologies if some of these seem vague, but I can assure you I tried several random address generators to determine a suitable value that should accommodate each of the fields[2].

Adding these up you get:

- **person** = 9 + 45 + 45 + 1 + 45 + 20 + 12 + 3 + 14 + 9 = 203 bytes x 100 = 20300 bytes
- **patient** = 9 + 3 + 1 = 13 bytes x 95 = 1235 bytes
- **medical_staff** = 9 + 2 + 4 = 15 bytes x 10 = 150 bytes

- **feedback** = 4 + 9 + 140 = 153 bytes x 20 = 3060 bytes
- **positive_feedback** = 4 + 4 + 1 = 9 bytes x 10 = 90 bytes
- **negative_feedback** = 4 + 2 + 1 = 7 bytes x 10 = 70 bytes

- **role** = 2 + 30 = 32 bytes x 10 = 320 bytes
- **service_type** = 2 + 40 = 42 bytes x 10 = 420 bytes
- **service** = 4 + 30 + 45 + 20 + 12 + 45 + 2 = 158 bytes x 10 = 1580 bytes
- **blood_type** = 3 bytes x 8 = 24 bytes
- **record_type** = 2 + 20 = 22 bytes x 10 = 220 bytes
- **medical_record** = 4 + 9 + 4 + 2 + 60 = 79 bytes x 200 = 15800 bytes
- **negative_feedback_reason** = 2 + 20 = 22 bytes x 5 = 110 bytes

[1] (Irish Genealogy Toolkit, 2023)
[2] (BestRandoms, 2023) (GeneratorMix, 2023)

I decided on the super and associated sub entity model because I think it is the clearest solution in both cases, here is an example of the 'person' sup/sub relationship:

Assuming 100 people, 90 of which are patients and 5 are medical staff, and 5 are both

Super table with sub tables

- person = [9 + 45 + 45 + 1 + 45 + 20 + 12 + 3 + 14 + 9 = 203 bytes] x 100 people = 20300 bytes
- patient = 9 + 3 + 1 = 13 bytes x 95 people = 1235 bytes
- medical_staff = 9 + 2 + 4 = 15 bytes x 10 people = 150 bytes
- **TOTAL = 21685**

Only sub entities

- patient = [9 + 45 + 45 + 1 + 45 + 20 + 12 + 3 + 14 + 9 = 203 bytes] + [3 + 1] = 207 bytes x 95 people = 19665 bytes
- medical_staff = [9 + 45 + 45 + 1 + 45 + 20 + 12 + 3 + 14 + 9 = 203 bytes] +[2 + 4] = 209 bytes x 10 people = 2090 bytes
- **TOTAL = 21755**

All-in-one entity

- person = [9 + 45 + 45 + 1 + 45 + 20 + 12 + 3 + 14 + 9 = 203 bytes] + [3 + 1] + [2 + 4] = 213 bytes x 100 people = 19665 bytes
- **TOTAL = 21300**

While putting everything into one entity saves the most space, the difference between the largest and smallest options is 455 bytes, in the best case scenario this is only a 2% difference. Therefore, I have decided on the super/sub model because that ensures no unnecessary null values.

Assuming 100 people in the system and the other estimations above, the total size for the database is: **43379 bytes or 43.38KB**

6 TABLE DESIGN

Person

FIELD	TYPE	SIZE	NULL/not NULL	DEFAULT	Constraints	Description
PPS	varchar	9	Not null		Unique Primary key	PPS of the person
fName	varchar	45	Not null			First name
lName	varchar	45	Not null			Last name
gender	enum		Not null			M' for male, 'F' for female
street	varchar	45	Not null			Street address
town	varchar	20	Not null			Town name
county	varchar	12	Not null			County name
dateOfBirth	date		Not null			Date of birth
telNo	varchar	14	Not null			Telephone number
nextOfKin	varchar	9			Foreign key	Foreign key pointing to a person who is the next of kin of the current

```
CREATE TABLE IF NOT EXISTS person (
  PPS VARCHAR(9) NOT NULL,
  fName VARCHAR(45) NOT NULL,
  lName VARCHAR(45) NOT NULL,
  gender ENUM('M', 'F') NOT NULL,
  street VARCHAR(45) NOT NULL,
  town VARCHAR(20) NOT NULL,
  county VARCHAR(12) NOT NULL,
  dateOfBirth DATE NOT NULL,
  telNo VARCHAR(14) NOT NULL,
  nextOfKin VARCHAR(9) NULL,
  PRIMARY KEY (PPS),
  CONSTRAINT FK_PERSON_PERSON
    FOREIGN KEY (nextOfKin)
    REFERENCES person (PPS)
    ON DELETE RESTRICT
    ON UPDATE CASCADE
);
```

Role

FIELD	TYPE	SIZE	NULL/not NULL	DEFAULT	Constraints	Description
rCode	varchar	2	Not null		Unique Primary key	Unique 2 char code to identify each role
rName	varchar	30	Not null			Name of the role e.g. Dentist

```
CREATE TABLE IF NOT EXISTS role (
  rCode VARCHAR(2) NOT NULL,
  rName VARCHAR(30) NOT NULL,
  PRIMARY KEY (rCode)
);
```

Service_Type

FIELD	TYPE	SIZE	NULL/not NULL	DEFAULT	Constraints	Description
sCode	varchar	2	Not null		Primary key	Unique ID for each type of service e.g. Hospital
sType	varchar	40	Not null			Short title/description of the service type e.g. Hospital

```
CREATE TABLE IF NOT EXISTS service_type (
  sCode VARCHAR(2) NOT NULL,
  sType VARCHAR(40) NOT NULL,
  PRIMARY KEY (sCode)
);
```

Service

FIELD	TYPE	SIZE	NULL/not NULL	DEFAULT	Constraints	Description
serviceId	int		Not null	AUTO_INCREMENT	Primary key	Unique ID for service
sName	varchar	30	Not null			Service name
sStreet	varchar	45	Not null			Service street address
sTown	varchar	20	Not null			Service town name
sCounty	varchar	12	Not null			Service county name
sTelNo	varchar	45	Not null			Service telephone number
sCode	varchar	2	Not null		Foreign key	ID indicating the type of service e.g. Hospital

```
CREATE TABLE IF NOT EXISTS service (
  serviceId INT NOT NULL AUTO_INCREMENT,
  sName VARCHAR(30) NOT NULL,
  sStreet VARCHAR(45) NOT NULL,
  sTown VARCHAR(20) NOT NULL,
  sCounty VARCHAR(12) NOT NULL,
  sTelNo VARCHAR(45) NOT NULL,
  sCode VARCHAR(2) NOT NULL,
  PRIMARY KEY (serviceId),
  CONSTRAINT FK_SERVICE_SERVICE_TYPE
    FOREIGN KEY (sCode)
    REFERENCES service_type (sCode)
    ON DELETE RESTRICT
    ON UPDATE CASCADE
);
```

Medical_Staff

Field	Type	Size	NULL/not null	DEFAULT	Constraints	Description
PPS	varchar	9	not null		Primary key	PPS of the person
rCode	varchar	2	not null		Foreign key	The id of the role of the medical staff e.g. DS=Dentist
worksAt	int					id of the service the medical staff works at

```
CREATE TABLE IF NOT EXISTS medical_staff (
  PPS VARCHAR(9) NOT NULL,
  rCode VARCHAR(2) NOT NULL,
  worksAt INT NOT NULL,
  PRIMARY KEY (PPS),
  CONSTRAINT FK_MEDICAL_STAFF_PERSON
    FOREIGN KEY (PPS)
    REFERENCES person (PPS)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT FK_MEDICAL_STAFF_ROLE
    FOREIGN KEY (rCode)
    REFERENCES role (rCode)
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT FK_MEDICAL_STAFF_SERVICE
    FOREIGN KEY (worksAt)
    REFERENCES service (serviceId)
    ON DELETE RESTRICT
    ON UPDATE CASCADE
);
```

Feedback

FIELD	TYPE	SIZE	NULL/not NULL	DEFAULT	Constraints	Description
feedbackId	INT		Not null	AUTO_INCREMENT	Primary key	Unique identifier for each feedback
givenBy	VARCHAR	9	Not null		Foreign key	Identifier of the person who gave the feedback
feedback	VARCHAR	140	Not null			The actual feedback text

```
CREATE TABLE IF NOT EXISTS feedback (
  feedbackId INT NOT NULL AUTO_INCREMENT,
  givenBy VARCHAR(9) NOT NULL,
  feedback VARCHAR(140) NOT NULL,
  PRIMARY KEY (feedbackId),
  CONSTRAINT FK_FEEDBACK_PERSON
    FOREIGN KEY (givenBy)
    REFERENCES person (PPS)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

Blood_Type

FIELD	TYPE	SIZE	NULL/not NULL	DEFAULT	Constraints	Description
bloodType	VARCHAR	3	Not null		Primary key	Unique identifier for each blood type

```
CREATE TABLE IF NOT EXISTS blood_type (
  bloodType VARCHAR(3) NOT NULL,
  PRIMARY KEY (bloodType)
);
```

Patient

FIELD	TYPE	SIZE	NULL/not NULL	DEFAULT	Constraints	Description
PPS	VARCHAR	9	Not null		Primary key	Personal Public Service (PPS) number of the patient
bloodType	VARCHAR	3	Not null		Foreign key	Blood type of the patient
currentlyAdmitted	TINYINT		Not null			Whether the patient is currently admitted to a hospital or not

```
CREATE TABLE IF NOT EXISTS patient (
  PPS VARCHAR(9) NOT NULL,
  bloodType VARCHAR(3) NOT NULL,
  currentlyAdmitted BOOLEAN NOT NULL,
  PRIMARY KEY (PPS),
  CONSTRAINT FK_PATIENT_PERSON
    FOREIGN KEY (PPS)
    REFERENCES person (PPS)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT FK_PATIENT_BLOOD_TYPE
    FOREIGN KEY (bloodType)
    REFERENCES blood_type (bloodType)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
);
```

Record_Type

FIELD	TYPE	SIZE	NULL/not NULL	DEFAULT	Constraints	Description
recCode	VARCHAR	2	Not null		Primary key	Unique identifier for each type of medical record
recType	VARCHAR	20	Not null			Short title/description for the record type

```
CREATE TABLE IF NOT EXISTS record_type (
  recCode VARCHAR(2) NOT NULL,
  recType VARCHAR(20) NOT NULL,
  PRIMARY KEY (recCode)
);
```

Medical_Record

FIELD	TYPE	SIZE	NULL/not NULL	DEFAULT	Constraints	Description
recordId	INT		Not null		Primary key	Unique identifier for each medical record
PPS	VARCHAR	9	Not null		Foreign key	Personal Public Service (PPS) number of the patient the medical record belongs to
serviceId	INT		Not null		Foreign key	Identifier of the service the medical record belongs to
recCode	VARCHAR	2	Not null		Foreign key	Unique identifier for each type of medical record
location	VARCHAR	60	Not null			Short description or URL to describe where the record is stored.

```
CREATE TABLE IF NOT EXISTS medical_record (
  recordId INT NOT NULL,
  PPS VARCHAR(9) NOT NULL,
  serviceId INT NOT NULL,
  recCode VARCHAR(2) NOT NULL,
  location VARCHAR(60) NOT NULL,
  PRIMARY KEY (recordId),
  CONSTRAINT FK_MEDICAL_RECORD_PATIENT
    FOREIGN KEY (PPS)
    REFERENCES patient (PPS)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT FK_MEDICAL_RECORD_SERVICE
    FOREIGN KEY (serviceId)
    REFERENCES service (serviceId)
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT FK_MEDICAL_RECORD_RECORD_TYPE
    FOREIGN KEY (recCode)
    REFERENCES record_type (recCode)
    ON DELETE RESTRICT
    ON UPDATE CASCADE
);
```

Positive_Feedback

FIELD	TYPE	SIZE	NULL/not NULL	DEFAULT	Constraints	Description
feedbackId	INT		Not null		Foreign key	Unique identifier for each feedback in positive_feedback table
stars	INT		Not null			Rating for the feedback on a scale of 1 to 5 stars
recommend	TINYINT		Not null			Whether the feedback recommends the service or not

```
CREATE TABLE IF NOT EXISTS positive_feedback (
  feedbackId INT NOT NULL,
  stars INT NOT NULL,
  recommend BOOLEAN NOT NULL,
  PRIMARY KEY (feedbackId),
  CONSTRAINT FK_POSITIVE_FEEDBACK_FEEDBACK
    FOREIGN KEY (feedbackId)
    REFERENCES feedback (feedbackId)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

Negative_Feedback_Reason

FIELD	TYPE	SIZE	NULL/not NULL	DEFAULT	Constraints	Description
reasonId	varchar	2	Not null		Unique Primary key	Unique 2 char ID for each complaint reason
reasonType	varchar	20	Not null			Short description of complaint reason

```
CREATE TABLE IF NOT EXISTS negative_feedback_reason (
  reasonId VARCHAR(2) NOT NULL,
  reasonType VARCHAR(40) NOT NULL,
  PRIMARY KEY (reasonId)
);
```

Negative_Feedback

FIELD	TYPE	SIZE	NULL/not NULL	DEFAULT	Constraints	Description
feedbackId	int		Not null		Unique Primary key	Unique ID for each feedback
reasonId	varchar	2	Not null		Foreign key	ID of the reason for complaint (they are predetermined)
resolved	tinyint		Not null			Whether the complaint was resolved e.g. true/false. 1 = true

```
CREATE TABLE IF NOT EXISTS negative_feedback (
  feedbackId INT NOT NULL,
  reasonId VARCHAR(2) NOT NULL,
  resolved BOOLEAN NOT NULL,
  PRIMARY KEY (feedbackId),
  CONSTRAINT FK_NEGATIVE_FEEDBACK_FEEDBACK
    FOREIGN KEY (feedbackId)
    REFERENCES feedback (feedbackId)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT FK_NEGATIVE_FEEDBACK_REASON
    FOREIGN KEY (reasonId)
    REFERENCES negative_feedback_reason (reasonId)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
);
```

7 QUERIES

Some possible useful queries:

1. List the first and last names of all the people who are next of kin.

```
SELECT * FROM Next_Of_Kin;
```

2. List all the medical staff who do not work in pharmacies.

```
SELECT *
FROM person
WHERE PPS IN ( SELECT PPS
                FROM medical_staff
                WHERE worksAt NOT IN ( SELECT serviceId
                                       FROM service
                                         WHERE sCode = 'PH' ) );
```

3. List the stars and feedback that has between 3 and 5 stars.

```
SELECT *FROM Feedback_Stats;
```

4. List the people born in May (to wish them happy birthday).

```
SELECT *
FROM person
WHERE dateOfBirth LIKE '%-05-%';
```

5. List patients along with their date of birth in a nice format, sorted alphabetically using their first name.

```
SELECT concat(fName, " ", lName) as Name,
DATE_FORMAT(dateOfBirth,'%d %M %Y') as "Year of Birth"
FROM patient NATURAL JOIN person ORDER BY fName ASC;
```

6. List all the medical records associated with a particular patient, along with their blood type.

```
SELECT PPS, location as Location, recType as "Record Type",
bloodType as "Blood Type"
FROM medical_record NATURAL JOIN record_type NATURAL JOIN patient
WHERE PPS = '7835628BH';
```

7. Count how many people are currently admitted.

```
SELECT * FROM Number_Of_Admitted;
```

8. Get the highest and lowest feedback rating given, along with the average.

```
SELECT MAX(stars) as "Highest rating given", AVG(stars) as "Average
rating", MIN(stars) as "Lowest rating given"
FROM positive_feedback;
```

9. List how many medical records each person in the system has (excluding people with one or less), from highest to lowest.

```
SELECT PPS, COUNT(PPS) as "Number of Medical Records"
FROM medical_record GROUP BY PPS HAVING COUNT(PPS) > 1 ORDER BY
COUNT(PPS) DESC;
```

10. Update some patients to be shown as currently admitted.

```
UPDATE patient
SET currentlyAdmitted=1
WHERE PPS IN ("3277285NS", "6996954HC");
```

11. Add new medical staff (you need to change sql workbench settings for this to work).

```
START TRANSACTION;
INSERT INTO medical_staff VALUES ('8870945GO', 'AN', 2);
INSERT INTO medical_staff VALUES ('6996954HC', 'DS', 5);
INSERT INTO medical_staff VALUES ('0586878XT', 'FD', 3);
COMMIT;
```

8 SECURITY

Based on the organization and tables, I have come up with the following database users:

NOTE: THE FOLLOWING CODE IS ABBREVIATED AS SOME OF THE ENTRIES ARE PRETTY LONG, FULL CODE IS IN THE SQL FILE.

1. Trainees: Read-only access to simple views This access is given as trainees need to be able to read some basic details, however they should not yet be given access to medical records as these are important and should only be shared to suitable employees.

```
CREATE USER 'Trainee';
GRANT SELECT ON Number_Of_Admitted TO 'Trainee';
GRANT SELECT ON Feedback_Stats TO 'Trainee';
```

2. Desk Staff: Update access on the person and patient tables and read access on medical_staff. This access is given as desk staff need to be able update information in these tables.

```
CREATE USER 'Desk_Staff';
GRANT SELECT, INSERT, UPDATE, DELETE, TRIGGER ON person TO
'Desk_Staff';
GRANT SELECT, TRIGGER, UPDATE ON medical_staff TO 'Desk_Staff';
```

3. Medical Staff: Read-only access on the patient and write access on medical record tables. This access is given as medical staff (e.g. doctors) don't need to update a patients personal details, but they might need to update their medical records.

```
CREATE USER 'Medical_Staff';
GRANT SELECT, UPDATE, INSERT, TRIGGER, DELETE ON medical_record TO
'Medical_Staff';
GRANT SELECT ON record_type TO 'Medical_Staff';
GRANT SELECT, UPDATE, TRIGGER ON patient TO 'Medical_Staff';
```

4. Service Manager: Update access on the patient, medical record, person, patient, feedback and medical staff tables. Is able to grant these to other users. This user/role is important and is used to create other users.

```
CREATE USER 'Service_Manager';
GRANT GRANT OPTION, TRIGGER, UPDATE, SELECT, INSERT, DELETE ON
person TO 'Service_Manager';
GRANT GRANT OPTION, DELETE, INSERT, SELECT, UPDATE, TRIGGER ON
medical_staff TO 'Service_Manager';
```

5. HSE Admins: Update access to service table. This access is given as the HSE Admin needs to update services e.g. hospitals, dentists, etc. but does not need details on employees or medical records.

```
CREATE USER 'HSE_Admin';
GRANT SELECT ON medical_staff TO 'HSE_Admin';
GRANT SELECT ON role TO 'HSE_Admin';
GRANT DELETE, INSERT, SELECT, UPDATE, TRIGGER ON service TO
'HSE_Admin';
```

9 VIEW

Here are three views that I decided to implement:

1. Count how many people are currently admitted. This is a simple view that a trainee would be given access read-only to, as it doesn't reveal personal information. It would be useful for general hospital operations.

```
CREATE VIEW Number_Of_Admitted AS SELECT COUNT(PPS) as "Number of
people currently admitted" FROM patient WHERE currentlyAdmitted="1";
```

2. Get the highest and lowest feedback rating given, along with the average. This is another simple view that a trainee would have read-only access to, but this time desk staff would also have full read-write access so they can add new feedback. This would be useful in improving hospital services.

```
CREATE OR REPLACE VIEW Feedback_Stats AS SELECT MAX(stars) as
"Highest rating given", AVG(stars) as "Average rating", MIN(stars)
as "Lowest rating given" FROM positive_feedback;
```

3. List the first and last names of all the people who are next of kin. This is a view that contains personal information, which is why trainee's would not have access to it. In this case, desk staff has full read-write access since they might need to contact next of kin and update them, while medical staff has read-only access in case they need to know.

```
CREATE OR REPLACE VIEW Next_Of_Kin AS SELECT concat(fName, " ",
lName) as Name FROM person WHERE PPS IN ( SELECT DISTINCT nextOfKin
FROM person WHERE nextOfKin IS NOT NULL );
```

10 CONCLUSION

I thought designing the database was fun. I like working with databases (for the most part) and I think it is always nice to get some experience working with them.

If I had to improve something, I would work a bit more on the security aspect of the project. However, I think the current level is satisfactory for a small-scale example.

11 REFERENCES

BestRandoms, 2023. *Random address in Ireland.* [Online]
Available at: https://www.bestrandoms.com/random-address-in-ie
[Accessed 27 February 2023].

GeneratorMix, 2023. *Random Address In Ireland.* [Online]
Available at: https://www.generatormix.com/random-address-in-ireland
[Accessed 27 February 2023].

HSE, 2022. *Executive Management Team and Operational Services Structure.* [Online]
Available at: https://www.hse.ie/eng/about/who/executive-management-team-and-operational-services-structure.pdf
[Accessed 2 February 2023].

HSE, 2023. *HSE Structure.* [Online]
Available at: https://www.hse.ie/eng/about/who/

Irish Genealogy Toolkit, 2023. *County Map of Ireland.* [Online]
Available at: https://www.irish-genealogy-toolkit.com/County-map-of-Ireland.html
[Accessed 27 February 2023].