02170 Database Systems
Designing & Modeling of a Hospital Database

March 2023

Figure 1: Freepik.com icon & DTU Logo

1 Statement of Requirements

Our database is a Hospital.

A Hospital consists of different named Departments. Each Department has an ID associated with it and a name. Departments have Devices and Beds assigned to them in a way that each Bed and each Device can be assigned only to one Department. A Bed has an *ID*, *room number*, and an *Is-Specialised* attribute indicating whether the Bed has a specific design. Beds can be occupied by Patients based on a *starting date* and an *ending date*.

A Device has an *ID* and a *name*. Devices can be used by different Doctors but used in only one Service maximum. Each Doctor has an *ID*, *name*, and *specialty*. Different Doctors can provide different Services to Patients. Each Service has an *ID*, *name*, *type*, and *price*. Multiple Services can be provided to Patients on a given *date* at a certain *price*.

The Patients have a *first name* and *last name*, a *birthday*, *age*, *phone number*, and *reception date* and *discharge date*. Each Patient may receive a Diagnosis and different Drugs. The Diagnosis is given on a specific *date*, with an *ID*, *name*, and *description*. The Drugs prescribed to Patients have an *ID*, *name*, its *manufacturer's name*, *drug type*, and *price*.

2 Conceptual Design

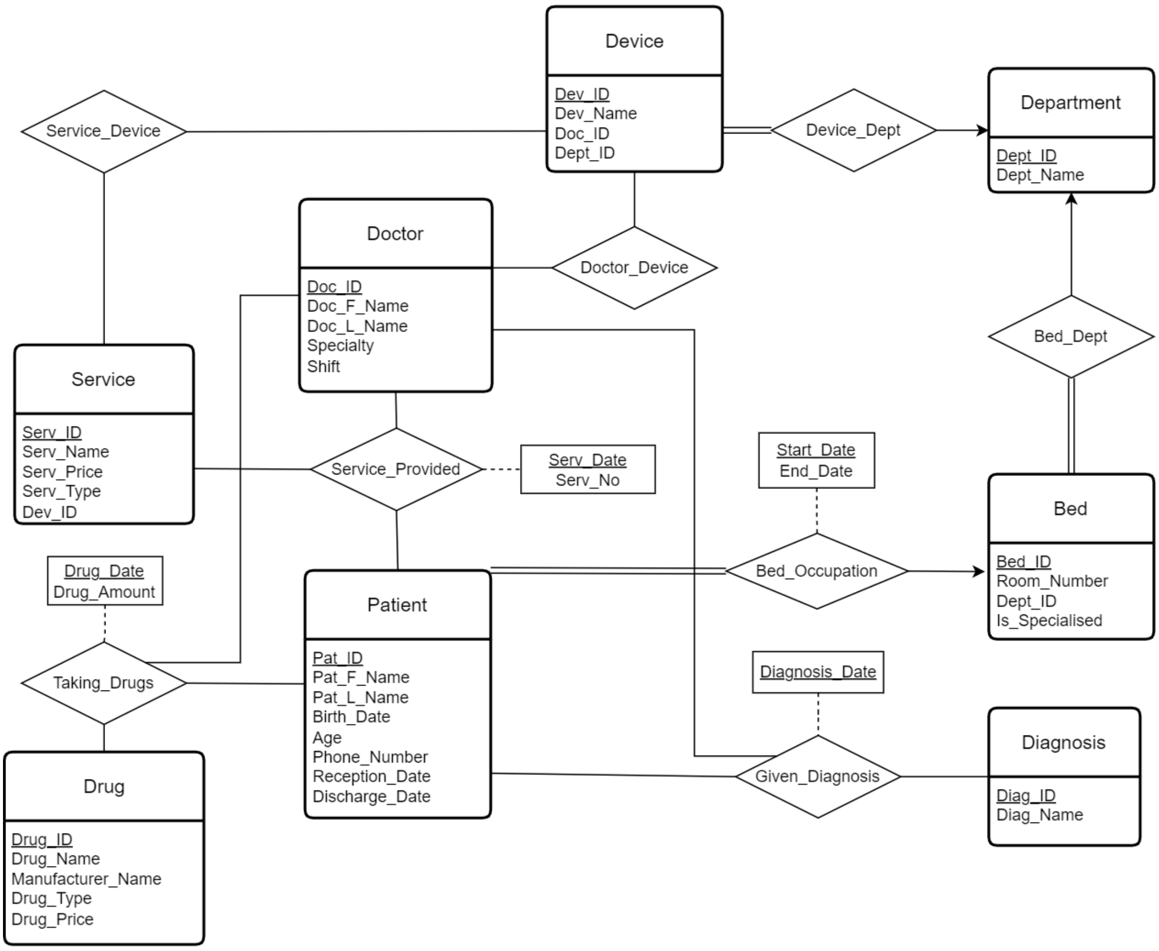

Figure 2: Entity-Relationship Diagram for the Hospital database.

Entities

Patient

The patient table includes all current and past patients who have been enrolled in the hospital. Their attributes include patient IDs, first and last names, birth date, age, phone number, reception date and discharge date.

Doctor

The Doctor tables includes all doctors who are currently employed at the hospital. They are given the attributes: Doctor ID, first and last names, speciality and shift. Some doctors are able to operate one or more devices resulting in a many-to-many cardinality for the Doctor_Device relationship.

Department
This table include the hospital departments with attributes Department ID and department name. Every bed and every device are associated with the department in which they are located.

Bed The bed table includes all beds in the hospital. These are defined by the bed's location in the hospital and whether or not it has special features. Attributes include bed ID, room number, department ID and is specialised.

Device
The Device table lists all the devices available for hospital services. The attributes are device ID, device name, the ID of the doctor who is the main responsible for the device and the department ID corresponding to the device's location.

Services
The Service table includes the services that doctor's can provide to patients at the hospital. Attributes are service ID, name, price, type (e.g. 'Imaging service' or 'Consultation') and lastly, the ID of the device needed for the service which could be NULL. Every service can be associated with many devices a device can be used for many services, yielding the many-to-many relationship Service_Device.

Drug
The Drug table includes drugs that are available at the hospital. These have attributes given by drug ID, name, manufacturer name, price and type, e.g. 'Pain reliever' or 'Hormone blocker'.

Diagnosis
The Diagnosis table include possible diagnosis the patients could be given. They are given the two attributes diagnosis ID and name.

Relationships

Bed_Occupation
Bed_Occupation represents a bed that is being used by a patient. It depends on Bed and Patient but also have Start_Date and End_Date as attributes showing the interval the bed will be occupied. The cardinality is one-to-one as a patient can be associated with at most one bed, and vice versa. Every patient is assigned to a bed, hence there is total participation of patients in the relationship set but only partial participation for Bed.

Service_Provided
Whenever a Service is provided to a Patient Service_Provided comes into play. It contains information about how many times the given Service was Provided as well as the specific time the provided Service occurred. It is a ternary relationship set with total participation of Patient. Not all doctors provide a service and not all devices are used for a service, hence these have partial participation. Every service could be given to more than one patient, include more than one doctor and multiple devices resulting in the cardinality illustrated in the ER-diagram.

Given_Diagnosis
A Patient can be diagnosed by a Doctor. Since a person can be diagnosed with the same illness several times, a Diag_Date exists to separate potential repeating diagnoses. The binary relationship has a many-to-many cardinality since every patient can be associated to more than one diagnosis, and vice versa. Participation is partial.

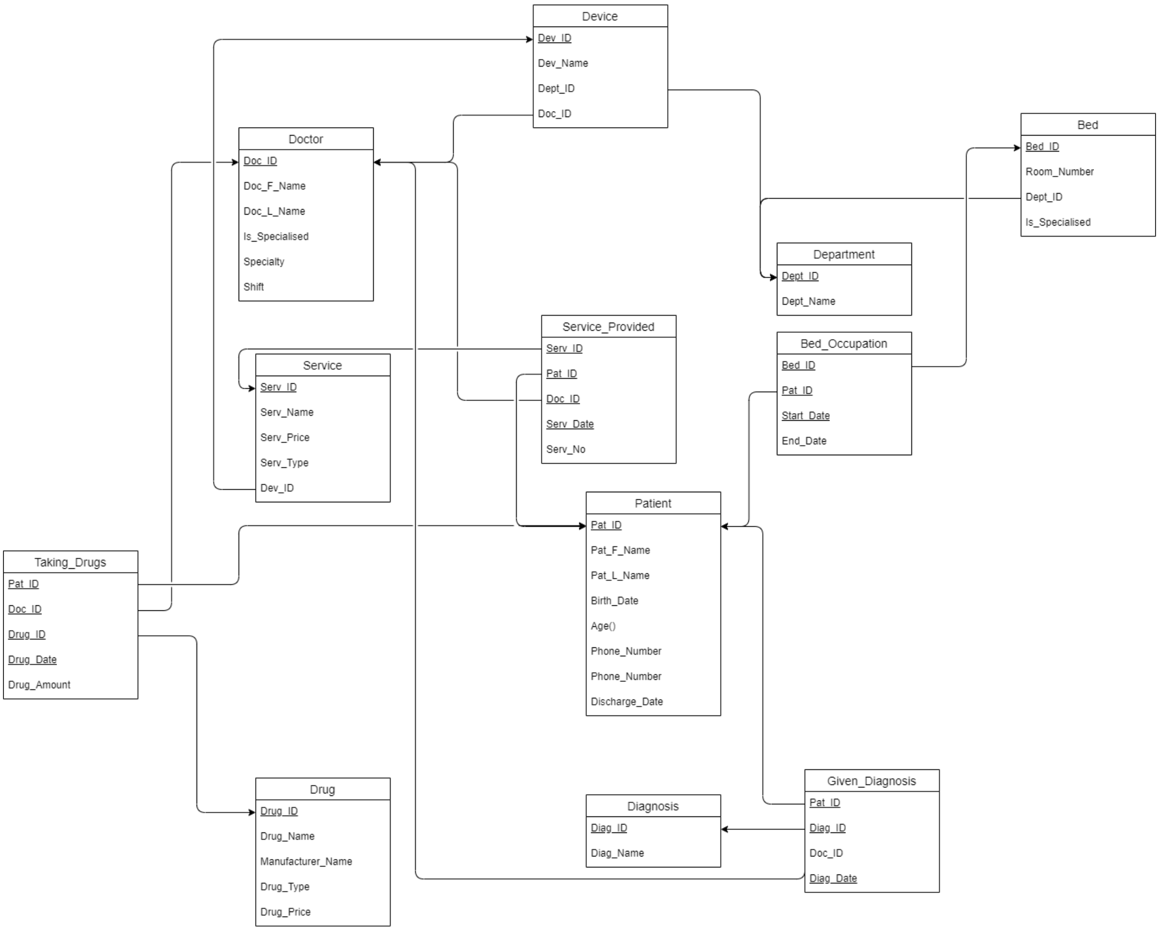

Figure 3: Logical Design diagram for the Hospital database

Taking_Drugs

A Patient can be prescribed a Drug by a Doctor. This relation also contains the amount of the given drug the patient is supposed to take and a Start_Date showing when the prescription started. This is a ternary relationship set with partial participation for all entities. Every drug can be taken by multiple patients, all doctors are able to assign drugs to patients and many drugs can be given to the same patient. Hence, the cardinality is many-to-many.

3 Logical Design

The logical design is based on the Entity-Relationship design. As a result, the tables Department, Device, Doctor, Service, Drug, Diagnosis, Patient, and Bed are in the logical design, the same as the Entity-Relationship design. Some relationships between different entities will result in new tables. Thus, these tables are needed to be designed in logical design.

Service_Provided is the result of the relationship between the tables Service, Patient, Doctor, and Device and has one foreign key from each of those tables. Plus, it has Serv_date as a primary key and another column as Serv_No. Each Service provided to each Patient at a specific date and time will be recorded in this table.

Bed_Occupation which is the result of the relationship between the tables Patient, and Bed. Moreover, it has two date columns named Start_Date and End_Date. Bed_ID and Pat_ID are the foreign keys from the aforementioned tables, and Start_Date is a primary key.

Table Taking_Drugs is the table that connects Patient, Doctor, and Drug tables. Besides, it has Drug_Date, and Drug_Amount. Foreign keys are from the mentioned tables and Drug_Date is a primary key.

Given_Diagnosis is a connection between the Patient, Diagnosis, and Doctor tables and it has one more column named Diag_Date which is also a primary key.

Besides the mentioned relationships, there are others that do not require to design of a specific table. The relation between departments will create a foreign key in Bed which is Dept_ID. This is the same with Device and Department in which, there is again Dept_ID as a foreign key in the Device table. Furthermore, for the relationship between Service and Device, Dev_ID is added into Service since it will create fewer NULLs if it is done in Service instead of Device. With the same approach, Doc_ID is added into Device in order to show the relationship between Device and Doctor.

4 Implementation

4.1 Tables

We use

```
CREATE DATABASE Hospital;
```

to create the database and name it Hospital. The tables are created following the logical design described in the previous section. The tables are then created using the following creation statements:

```
CREATE TABLE Patient
(Pat_ID BIGINT AUTO_INCREMENT,
Pat_F_Name VARCHAR(30),
Pat_L_Name VARCHAR(15),
Birth_Date DATE,
Age INT AS (TIMESTAMPDIFF(YEAR, Birth_Date, CURDATE())),
Phone_Number VARCHAR(11),
Reception_Date DATE,
Discharge_Date DATE,
PRIMARY KEY(Pat_ID));

CREATE TABLE Doctor
(Doc_ID INT(4) AUTO_INCREMENT,
Doc_F_Name VARCHAR(30) NOT NULL,
Doc_L_Name VARCHAR(15) NOT NULL,
Speciality VARCHAR(20),
Shift ENUM('Morning','Noon','Night'),
PRIMARY KEY(Doc_ID));

CREATE TABLE Department
(Dept_ID INT(4) AUTO_INCREMENT,
```

```sql
Dept_Name VARCHAR(30),
PRIMARY KEY(Dept_ID));

CREATE TABLE Bed
(Bed_ID INT(4) AUTO_INCREMENT,
Room_Number INT(4),
Dept_ID INT(4),
Is_Specialised BOOL,
PRIMARY KEY(Bed_ID),
FOREIGN KEY(Dept_ID) REFERENCES Department(Dept_ID) ON DELETE SET NULL);

CREATE TABLE Device
(Dev_ID INT(4) AUTO_INCREMENT,
Dev_name VARCHAR(20),
Doc_ID INT(4),
Dept_ID INt(4),
PRIMARY KEY(Dev_ID),
FOREIGN KEY(Dept_ID) REFERENCES Department(Dept_ID) ON DELETE SET NULL);

CREATE TABLE Service
(Serv_ID BIGINT AUTO_INCREMENT,
Serv_Name VARCHAR(25),
Serv_Price DECIMAL(10,2),
Serv_Type VARCHAR(25),
Dev_ID INT(4),
PRIMARY KEY(Serv_ID),
FOREIGN KEY(Dev_ID) REFERENCES Device(Dev_ID) ON DELETE SET NULL);


CREATE TABLE Drug
(Drug_ID BIGINT AUTO_INCREMENT,
Drug_NameVARCHAR(20),
Manufacturer_Name VARCHAR(20),
Drug_Price DECIMAL(10,2),
Drug_Type VARCHAR(20),
PRIMARY KEY(Drug_ID));

CREATE TABLE Diagnosis
(Diag_ID INT(4) AUTO_INCREMENT,
Diag_Name VARCHAR(25),
PRIMARY KEY(Diag_ID));

#--RELATIONS

CREATE TABLE Bed_Occupation
(Bed_IDINT(4) NOT NULL,
Pat_ID BIGINT NOT NULL,
Start_Date DATE,
End_Date DATE,
PRIMARY KEY(Bed_ID,Pat_ID,Start_Date),
FOREIGN KEY(Bed_ID) REFERENCES Bed(Bed_ID) ON DELETE CASCADE,
FOREIGN KEY(Pat_ID) REFERENCES Patient(Pat_ID) ON DELETE CASCADE);

CREATE TABLE Service_Provided
(Serv_ID BIGINT,
Pat_ID BIGINT,
Doc_ID INT(4),
Serv_Date DATE,
Serv_No INT(4),
PRIMARY KEY(Serv_ID, Pat_ID, Doc_ID, Serv_Date),
FOREIGN KEY(Serv_ID) REFERENCES Service(Serv_ID) ON DELETE CASCADE,
FOREIGN KEY(Pat_ID) REFERENCES Patient(Pat_ID) ON DELETE CASCADE,
FOREIGN KEY(Doc_ID) REFERENCES Doctor(Doc_ID) ON DELETE CASCADE;

CREATE TABLE Taking_Drugs
(Drug_ID BIGINT,
Pat_ID BIGINT,
Doc_ID INT(4),
Drug_Amount INT(4),
Drug_Date DATE,
PRIMARY KEY(Drug_ID,Pat_ID,Doc_ID,Drug_Date),
FOREIGN KEY(Pat_ID) REFERENCES Patient(Pat_ID) ON DELETE CASCADE,
FOREIGN KEY(Doc_ID) REFERENCES Doctor(Doc_ID) ON DELETE CASCADE,
FOREIGN KEY(Drug_ID) REFERENCES Drug(Drug_ID) ON DELETE CASCADE);

CREATE TABLE Given_Diagnosis
(Pat_ID BIGINT,
Doc_ID INT(4),
Diag_ID INT(4),
Diag_Date DATE,
PRIMARY KEY(Pat_ID, Doc_ID, Diag_ID,Diag_Date),
FOREIGN KEY(Pat_ID) REFERENCES Patient(Pat_ID) ON DELETE CASCADE,
FOREIGN KEY(Doc_ID) REFERENCES Doctor(Doc_ID) ON DELETE CASCADE,
FOREIGN KEY(Diag_ID) REFERENCES Diagnosis(Diag_ID) ON DELETE CASCADE);
```

Most of our tables use an ID as primary keys. These IDs are auto-incremented to ensure uniqueness.

4.2 Views

The Patient table contains sensitive patient information, hence we create a view called PatientView, which only allows the database user to retrieve non-sensitive information such as the patient name and age:

```
CREATE VIEW PatientView AS SELECT Pat_F_Name, Pat_L_Name, Age FROM Patient;
```

Additional views can be created for doctor's and other hospital staff to see the current and past patients including all patient information:

```
# Show current patients
CREATE VIEW Current_Patients_View AS
SELECT * FROM Patient WHERE Discharge_Date IS NULL;

# Show past patients
CREATE VIEW Past_Patients_View AS
SELECT * FROM Patient WHERE Discharge_Date IS NOT NULL;
```

5 Database Instance

The tables are populated with data using

```
INSERT INTO Table (row1, row2, ...) VALUES ((val1), (val2)...);
```

where 'Table' is replaced with the table name and 'val1', 'val2' etc. are replaced with tuples that should be inserted into the table rows.

Patient table

Pat_ID	Pat_F_Name	Pat_L_Name	Birth_Date	Age	Phone_Number	Reception_Date	Discharge_Date
1	Henry	Bertstein	1983-03-21	40	+4560878703	2022-12-21	2023-01-05
2	Adam	Hansen	1945-05-01	77	+4545672005	2022-12-21	2023-01-05
3	Sofie-Amalie	Tobiasen	1991-03-07	32	+4544802345	2022-06-21	NULL
4	Nima	Andersen	1997-05-11	25	+4547601112	2022-12-21	2023-01-05
5	Elisabeth	Mikkelsen	1965-08-28	57	+4542343404	2021-12-21	2023-01-05
6	Alberto	Milano	1955-12-12	67	NULL	2022-12-21	NULL
7	Mette Berg	Hansen	1962-03-20	61	+4589897703	2023-10-24	2023-10-25
8	Gertrud	Samuelsen	1938-09-01	84	+4544805413	2022-12-21	NULL
9	Thomas William	Helming	1980-01-30	43	+4567838376	2023-11-21	NULL
10	Niels Gram	Petersen	1959-05-04	63	+4557809070	2022-11-21	2023-01-05

Doctor table

Doc_ID	Doc_F_Name	Doc_L_Name	Speciality	Shift
1	Svend	Svendsen	Dermatology	Morning
2	John	Carlson	Pediatrics	Noon
3	Line	Gale	Dermatology	Noon
4	Eric	Milton	Psychiatry	Morning
5	Bill	Svendsen	Anesthesiology	Morning
6	Joshua	Cooper	Surgery	Night
7	Frank	Roam	Orthopedics	Morning
8	Anders Aaboe	Andersen	Surgery	Night
9	Inger	Yale	Obstetrics	Morning
10	Thomas	Nielsen	Orthopedics	Morning

Department table

Dept_ID	Dept_Name
1	Medicine
2	Surgery
3	Gynaecology
4	Obstetrics
5	Paediatrics
6	Ophthalmology
7	Dental
8	Orthopaedics
9	Neurology
10	Cardiology

Bed table

Bed_ID	Room_Number	Dept_ID	Is_Specialised
1	4	1	1
2	1	3	0
3	3	3	0
4	8	1	0
5	3	2	0
6	6	4	0
7	6	4	0
8	2	1	1
9	4	1	0
10	3	5	1

Device table

Dev_ID	Dev_name	Doc_ID	Dept_ID
1	MRI scanner	1	5
2	Ultrasound machine	3	7
3	Oxygen concentrator	2	8
4	CT scanner	7	3
5	Urine analyser	6	4
6	Hematology analyzer	4	1

Service table

Serv_ID	Serv_Name	Serv_Price	Serv_Type	Dev_ID
1	Blood test	1000.00	Blood Services	NULL
2	X-RAY	1000.00	Imaging Services	NULL
3	CT-scan	820.00	Imaging Services	4
4	Doctor consultation	NULL	Consultation	NULL
5	One day bed rest	500.00	Bed Rest	NULL
6	Emergency room	2000.00	Emergency Occupancy	NULL
7	Urine sample test	50.50	Uurine Test	5
8	Pregnancy urine test	25.00	Urine Test	5
9	Vitamin levels blood	200.00	Blood Services	NULL

Drug table

Drug_ID	Drug_Name	Manufacturer_Name	Drug_Price	Drug_Type
1	Acetaminophen	LegitCO	100.50	Pain reliever
2	Xanax	Rotciv	10.00	Antidepressant
3	Metformin	NextImproved	5.70	Insulin sensitivity
4	Brilinta	GETwl	26.10	Prevents blood clots
5	Adderall	EORC	99.90	ADHD
6	Tramadol NextGen	Novo Nordisk	91.20	Pain reliever
7	Ozempic	Enterap	1.00	Type 2 diabetes
8	Lofexidine	DrugCorp	8.10	Hormone blocker
9	Lexapro	New Initiative	3.30	Antidepressant
10	Fentanyl	Strands	89.90	Pain reliever

Diagnosis table

Diag_ID	Diag_Name
1	Physical Diseases
2	Mental Diseases
3	Infectious Diseases
4	No-infectious Diseases
5	Deficiency Diseases
6	Inherited Diseases
7	Degenerative Diseases
8	Social Diseases
9	Degenerative Diseases
10	Self-inflicted Diseases

Bed occupation (relationship) table

Bed_ID	Pat_ID	Start_Date	End_Date
2	5	2023-01-03	2023-03-08
3	3	2023-02-01	2023-03-02
4	6	2023-03-04	2023-03-07
5	1	2023-01-15	2023-03-15
5	2	2023-01-01	2023-01-14
5	9	2023-03-08	2023-03-11
6	7	2023-01-11	2023-03-12
7	4	2023-01-07	2023-03-08
8	8	2023-01-12	2023-03-12

Service provided (relationship) table

Serv_ID	Pat_ID	Doc_ID	Dev_ID	Serv_Date	Serv_No
1	2	5	1	2023-03-05	4
1	4	9	1	2023-03-13	12
1	7	1	1	2022-08-01	14
2	3	8	2	2022-12-21	10
3	1	3	3	2023-01-15	5
3	5	2	3	2023-01-30	4
3	8	6	3	2023-01-20	1
4	6	4	4	2023-03-28	7

Given diagnosis (relationship) table

Pat_ID	Doc_ID	Diag_ID	Diag_Date
1	3	1	2023-01-16
2	1	3	2023-01-01
3	7	4	2023-02-05
6	5	1	2023-03-04
8	3	6	2023-01-19
10	3	8	2023-03-14

Taking drugs (relationship) table

Drug_ID	Pat_ID	Doc_ID	Drug_Amount	Drug_Date
1	1	1	2	2023-01-18
1	6	4	1	2023-03-04
2	2	3	5	2023-01-03
2	7	4	1	2023-02-12
3	2	4	1	2023-01-05
5	2	4	1	2023-02-10
6	3	4	1	2023-02-14
6	5	4	1	2023-02-08
7	5	4	1	2023-02-09
8	8	6	1	2023-01-09
9	8	1	2	2023-03-09
10	8	8	1	2023-03-01

PatientView

PatientView shows patient data without any sensitive patient information.

Pat_F_Name	Pat_L_Name	Age
Henry	Bertstein	40
Adam	Hansen	77
Sofie-Amalie	Tobiasen	32
Nima	Andersen	25
Elisabeth	Mikkelsen	57
Alberto	Milano	67
Mette Berg	Hansen	61
Gertrud	Samuelsen	84
Thomas William	Helming	43
Niels Gram	Petersen	63

DoctorPatientView

DoctorPatientView shows every doctor with their patient if they have one.

Doc_ID	Doc_F_Name	Doc_L_Name	Pat_ID	Pat_F_Name	Pat_L_Name
1	Svend	Svendsen	7	Mette Berg	Hansen
2	John	Carlson	5	Elisabeth	Mikkelsen
3	Line	Gale	1	Henry	Bertstein
4	Eric	Milton	6	Alberto	Milano
5	Bill	Svendsen	2	Adam	Hansen
6	Joshua	Cooper	8	Gertrud	Samuelsen
7	Frank	Roam	NULL	NULL	NULL
8	Anders Aaboe	Andersen	3	Sofie-Amalie	Tobiasen
9	Inger	Yale	4	Nima	Andersen
10	Thomas	Nielsen	NULL	NULL	NULL

CurrentPatientView

Show current patients.

Pat_ID	Pat_F_Name	Pat_L_Name	Birth_Date	Age	Phone_Number	Reception_Date	Discharge_Date
3	Sofie-Amalie	Tobiasen	1991-03-07	32	+4544802345	2022-06-21	NULL
6	Alberto	Milano	1955-12-12	67	NULL	2022-12-21	NULL
8	Gertrud	Samuelsen	1938-09-01	84	+4544805413	2022-12-21	NULL
9	Thomas William	Helming	1980-01-30	43	+4567838376	2023-11-21	NULL

PastPatientView

Show past patients.

Pat_ID	Pat_F_Name	Pat_L_Name	Birth_Date	Age	Phone_Number	Reception_Date	Discharge_Date
1	Henry	Bertstein	1983-03-21	40	+4560878703	2022-12-21	2023-01-05
2	Adam	Hansen	1945-05-01	77	+4545672005	2022-12-21	2023-01-05
4	Nima	Andersen	1997-05-11	25	+4547601112	2022-12-21	2023-01-05
5	Elisabeth	Mikkelsen	1965-08-28	57	+4542343404	2021-12-21	2023-01-05
7	Mette Berg	Hansen	1962-03-20	61	+4589897703	2023-10-24	2023-10-25
10	Niels Gram	Petersen	1959-05-04	63	+4557809070	2022-11-21	2023-01-05

TotalPatientServPrice

Shows the total price for hospital services per patient.

Pat_ID	Pat_F_Name	Pat_L_Name	Serv_ID	Serv_Name	Tot_Serv_Price
1	Henry	Bertstein	3	CT-scan	4100.00
2	Adam	Hansen	1	Blood test	4000.00
3	Sofie-Amalie	Tobiasen	2	X-RAY	10000.00
4	Nima	Andersen	1	Blood test	12000.00
5	Elisabeth	Mikkelsen	3	CT-scan	3280.00
6	Alberto	Milano	4	Doctor consultation	NULL
7	Mette Berg	Hansen	1	Blood test	14000.00
8	Gertrud	Samuelsen	3	CT-scan	820.00

TotalPatientDrugPrice

Shows total price for drugs per patient.

Pat_ID	Pat_F_Name	Pat_L_Name	Drug_ID	Drug_Name	Tot_Drug_Price
1	Henry	Bertstein	1	Acetaminophen	201.00
2	Adam	Hansen	2	Xanax	50.00
3	Sofie-Amalie	Tobiasen	6	Tramadol NextGen	91.20
5	Elisabeth	Mikkelsen	6	Tramadol NextGen	91.20
6	Alberto	Milano	1	Acetaminophen	100.50
7	Mette Berg	Hansen	2	Xanax	10.00
8	Gertrud	Samuelsen	8	Lofexidine	8.10

6 SQL Data Queries

If you would like all the available drugs that relieve pain, you could do the following SELECT query:

```
SELECT Drug_name AS 'Drug name',Drug_Price AS 'Price' FROM Drug WHERE Drug_Type = 'Pain reliever';
```

"Drug name"	Price
Acetaminophen	100.50
Tramadol NextGen	91.20
Fentanyl	89.90

To get all services provided grouped by Department, this query gives you that:

```
SELECT Department.Dept_Name as Department, SUM(Service_Provided.Serv_No) as Services_provided
FROM Department
        JOIN Device ON Department.Dept_ID = Device.Dept_ID
        JOIN Service ON Device.Dev_ID = Service.Dev_ID
        JOIN Service_Provided ON Service.Serv_ID = Service_Provided.Serv_ID
GROUP BY Department.Dept_ID, Department.Dept_Name;
```

Department	Services_provided
Medicine	41
Gynaecology	10
Obstetrics	4

To order patients by the total amount of money they have spent on services, you could make the following SELECT statement:

```
SELECT  Pat_ID, Pat_F_Name, Pat_L_Name, Serv_ID, Serv_Name,
        ROUND(Serv_Price*Serv_No, 2) AS Tot_Serv_Price
FROM (Patient NATURAL JOIN Service_Provided)
        JOIN Service USING (Serv_ID)
        ORDER BY Tot_Serv_Price ASC;
```

Pat_ID	Pat_F_Name	Pat_L_Name	Serv_ID	Serv_Name	Tot_Serv_Price
6	Alberto	Milano	4	Doctor consultation	NULL
3	Sofie-Amalie	Tobiasen	10	Ultra sound	NULL
8	Gertrud	Samuelsen	3	CT-scan	820.00
3	Sofie-Amalie	Tobiasen	2	X-RAY	1000.00
5	Elisabeth	Mikkelsen	3	CT-scan	3280.00
2	Adam	Hansen	1	Blood test	4000.00
1	Henry	Bertstein	3	CT-scan	4100.00
3	Sofie-Amalie	Tobiasen	2	X-RAY	10000.00
4	Nima	Andersen	1	Blood test	12000.00
7	Mette Berg	Hansen	1	Blood test	14000.00

The following is a simple SELECT statement to get all Doctors alphabetically sorted by their *Specialty*.

```
SELECT  CONCAT(Doc_L_Name,', ',Doc_F_Name) AS 'Doctor',Speciality,
        Shift FROM DOCTOR ORDER BY Speciality;
```

Doctor	Speciality	Shift
Bill Svendsen	Anesthesiology	Morning
Svend Svendsen	Dermatology	Morning
Line Gale	Dermatology	Noon
Inger Yale	Obstetrics	Morning
Frank Roam	Orthopedics	Morning
Thomas Nielsen	Orthopedics	Morning
John Carlson	Pediatrics	Noon
Eric Milton	Psychiatry	Morning
Anders Aaboe Andersen	Surgery	Night
Joshua Cooper	Surgery	Night

To get a simple overview of all occupied Beds, use the following query:

```
SELECT CONCAT(Pat_L_Name,', ',Pat_F_Name) AS 'Patient Name',Bed_ID, End_Date AS 'Has bed until'
        FROM Patient
        JOIN Bed_Occupation ON Patient.Pat_ID = Bed_Occupation.Pat_ID;
```

"Patient Name"	Bed_ID	"Has bed until"
Elisabeth Mikkelsen	2	2023-03-08
Sofie-Amalie Tobiasen	3	2023-03-02
Alberto Milano	4	2023-03-07
Henry Bertstein	5	2023-03-15
Adam Hansen	5	2023-01-14
Thomas William Helming	5	2023-03-11
Mette Berg Hansen	6	2023-03-12
Nima Andersen	7	2023-03-08
Gertrud Samuelsen	8	2023-03-12

7 SQL Programming

7.1 Functions

When managing the Hospital's staff, it is important to be able to tell when the different shifts start and end. Therefore, we made a CurrentShift function which, when given a *Datetime* tells what the current shift is for that time.

```
    CREATE FUNCTION CurrentShift(timeOfDay DATETIME) RETURNS VARCHAR(20)
BEGIN
    IF HOUR(timeOfDay) >= 6 AND HOUR(timeOfDay) < 12 THEN
        RETURN 'Morning';
    ELSEIF HOUR(timeOfDay) >= 12 AND HOUR(timeOfDay) < 18 THEN
        RETURN 'Noon';
    ELSEIF HOUR(timeOfDay) >= 18 AND HOUR(timeOfDay) < 24 THEN
        RETURN 'Evening';
        ELSEIF HOUR(timeOfDay) >= 0 AND HOUR(timeOfDay) < 6 THEN
                RETURN 'Night';
    ELSE
        RETURN 'Undefined';
    END IF;
END
```

Some examples of running this function could be:

```
SELECT '2020-08-10 11:00:00' AS 'Current time',
            CurrentShift('2020-08-10 11:00:00') AS 'Current Shift'
        UNION ALL
        SELECT '2021-09-10 15:00:00' AS 'Current time',
            CurrentShift('2021-09-10 15:00:00') AS 'Current Shift'
        UNION ALL
        SELECT '2020-12-11 19:00:00' AS 'Current time',
            CurrentShift('2020-12-11 19:00:00') AS 'Current Shift'
        UNION ALL
        SELECT '2022-03-05 01:00:01' AS 'Current time',
            CurrentShift('2022-03-05 01:00:01') AS 'Current Shift';
```

"Current time"	"Current Shift"
2020-08-10 11:00:00	Morning
2021-09-10 15:00:00	Noon
2020-12-11 19:00:00	Evening
2022-03-05 01:00:01	Night

We also made a simple function that, when given a *Datetime* for when someone became pregnant, gives the approximate end of pregnancy.

```
    CREATE FUNCTION EndOfPregnancy(startOfPregnancy DATETIME) RETURNS DATETIME
        return ADDDATE(startOfPregnancy, INTERVAL 9 month);
```

Here are some examples of using the function:

```
SELECT DATE(now()) AS 'Start of pregnancy',
        EndOfPregnancy(DATE(now())) as 'Expected end of term'
    UNION ALL
SELECT '2020-06-10' AS 'Start of pregnancy',
        EndOfPregnancy('2020-06-10') as 'Expected end of term'
    UNION ALL
SELECT '2023-01-02' AS 'Start of pregnancy',
        EndOfPregnancy('2023-01-02') as 'Expected end of term';
```

"Start of pregnancy"	"Expected end of term"
2023-03-23	2023-12-23 00:00:00
2020-06-10	2021-03-10 00:00:00
2023-01-02	2023-10-02 00:00:00

Lastly, it is undoubtedly needed to be able to tell how many doctors are at work at any given time. Therefore, the *DoctorsWorkingShift* function was made to be able to tell just this when given a *Shift*.

```
    CREATE FUNCTION DoctorsWorkingShift(vShift VARCHAR(20)) RETURNS int
BEGIN
        DECLARE vShiftCount INT;
    SELECT COUNT(*) INTO vShiftCount FROM Doctor
    WHERE Shift = vShift;
    RETURN vShiftCount;
end;
```

So, if we want an overview of the different shifts' working doctors, we could SELECT the following:

```
 SELECT         DoctorsWorkingShift('Morning') AS 'Morning shift',
                DoctorsWorkingShift('Noon') AS 'Noon shift',
                DoctorsWorkingShift('Evening') AS 'Evening shift',
                DoctorsWorkingShift('Night') AS 'Night shift';
```

"Morning shift"	"Noon shift"	"Evening shift"	"Night shift"
6	2	0	2

7.2 Procedures

Each time a patient comes to the hospital and must be assigned a bed, the operator must insert a record in Patient and also in Bed_Occupation table. In order to make it easier, a Stored Procedure can be called to do the both inserts at once.

```
DELIMITER //

CREATE PROCEDURE Easy_Insert_Patient(

IN Par_Pat_F_Name VARCHAR(30),
IN Par_Pat_L_Name VARCHAR(15),
IN Par_Birth_Date Date,
IN Par_Reception_Date Date,
IN Par_Phone_Number VARCHAR(11),
IN Par_Bed_ID int

)
BEGIN
    INSERT INTO Patient(Pat_F_Name, Pat_L_Name, Birth_date, Phone_Number,
                  reception_date, discharge_date)
      Values(Par_Pat_F_Name, Par_Pat_L_Name, Par_Birth_date,
          Par_Phone_Number, Par_reception_date, NULL);

  Select Max(Pat_ID) into @Max_Pat_ID
  From patient;

  INSERT INTO bed_occupation(Bed_ID,Pat_ID,Start_Date,End_Date)
  Values(Par_Bed_ID,@Max_Pat_ID,Par_reception_date,NULL)
    ;
END //

DELIMITER ;
```

An example of calling this procedure is like this:

```
Call Easy_Insert_Patient("Allan","Harris","1997-01-01","2023-03-26","+45443223",1);
```

By calling this procedure, a patient named Allan Harris is added in Patient and also a record will be inserted for this patient in Bed_Occupation.

Patient

Pat_ID	Pat_F_Name	Pat_L_Name	Birth_Date	Age	Phone_Number	Reception_Date	Discharge_Date
13	Allan	Harris	1997-01-01	26	+45443223	2023-03-26	NULL

Bed_Occupation

Bed_ID	Pat_ID	Start_Date	End_Date
1	13	2023-03-26	NULL

Another procedure is for finding about the patients that are currently located in a specific room.

```
DELIMITER //
CREATE PROCEDURE Available_Patients(IN room_num INT)
BEGIN
    SELECT Patient.Pat_F_Name, Patient.Pat_L_Name, Bed_Occupation.Start_Date, Bed_Occupation.End_Da
    FROM Patient
    JOIN Bed_Occupation ON Patient.Pat_ID = Bed_Occupation.Pat_ID
    JOIN Bed ON Bed_Occupation.Bed_ID = Bed.Bed_ID
    WHERE Bed.Room_Number = room_num AND Bed_Occupation.End_Date IS NULL;
END //
DELIMITER ;
```

An example of calling Available_Patient procedure is as follows:

```
Call Available_Patients(4);
```

This will show the patients that are currently in room 4.

Pat_F_Name	Pat_L_Name	Start_Date	End_Date
Allan	Harris	2023-03-23	NULL

7.3 Triggers

It happens that an operator make mistakes with the entering the dates and insert a past date into the start date in Bed_Occupation table. This trigger will make sure that if an operator mistakenly inserts a past date in Start_Date and End_Date, it will be changed to the current date.

```
DELIMITER //
Create Trigger before_insert_bed_occupation
BEFORE INSERT ON bed_occupation FOR EACH ROW
BEGIN
IF NEW.Start_Date < CURDATE() THEN SET NEW.Start_Date = CURDATE();
END IF;
IF NEW.End_Date < CURDATE() THEN SET NEW.End_Date = CURDATE();
End IF;
END //
```

Here the operator is trying to insert a record in Bed_Occupation which its start date is 2022-01-01:

```
INSERT INTO bed_occupation(Bed_ID,Pat_ID,Start_Date,End_Date)
        VALUES(3,12,'2022-01-01','2023-05-07');
```

However, it can be seen that indeed the record has been inserted with today (2023-03-25) as the start date:

Bed_ID	Pat_ID	Start_Date	End_Date
3	12	2023-03-25	2023-05-07

Device_ID=2 is a device which is related to doctors with the specialty ”Pediatrics”. (As it is stated before, Device has a foreign key Doc_ID which connects Device to Doctor). So, whenever a new doctor with the specialty ”Pediatrics” is added to Doctor, the record Device_ID=2 in Device must be updated to be linked to this new Doctor.

```
DELIMITER //
CREATE TRIGGER after_insert_doctor
AFTER INSERT
ON doctor FOR EACH ROW
IF NEW.Speciality='Pediatrics' THEN
UPDATE Device
SET Doc_ID = new.Doc_ID
Where Dev_ID=2;
END IF;

DELIMITER //
```

8 SQL Table Modifications

Here are some table modifications performed using update and delete statements on the Hospital dataset. The result of each statement is also shown below each statement. Note that we need to use

```
SET SQL_SAFE_UPDATES=0;
```

in order to be able to update our database.

Update statements

The following statement gives a 10% discount on the price of a Drug if the amount is 10 or more:

```
UPDATE Drug
SET drug_price = drug_price - (0.1 * drug_price)
WHERE drug_amount >= 10;
```

Here is the result of the above statement on the Drug table:

Drug_ID	Drug_Name	Manufacturer_Name	Drug_Price	Drug_Amount	Drug_Type
1	Acetaminophen	LegitCO	90.45	10	Pain reliever
2	Xanax	Rotciv	10.00	5	Antidepressant
3	Metformin	NextImproved	5.70	9	Insulin sensitivity
4	Brilinta	GETwl	26.10	3	Prevents blood clots
5	Adderall	EORC	89.91	20	ADHD
6	Tramadol NextGen	Novo Nordisk	91.20	9	Pain reliever
7	Ozempic	Enterap	1.00	3	Type 2 diabetes
8	Lofexidine	DrugCorp	7.29	20	Hormone blocker
9	Lexapro	New Initiative	3.30	7	Antidepressant
10	Fentanyl	Strands	80.91	11	Pain reliever

This statement increases the price of all the Services using the ”Urine Analyser” device by 20%:

```
UPDATE Service NATURAL JOIN Device
SET serv_price = serv_price + (0.2 * serv_price)
WHERE dev_name = "Urine Analyser";
```

Serv_ID	Serv_Name	Serv_Price	Serv_Type	Dev_ID
1	blood test	1000.00	blood services	NULL
2	X-RAY	1000.00	imaging services	NULL
3	CT-scan	820.00	imaging services	4
4	doctor consultation	NULL	consultation	NULL
5	one day bed rest	500.00	bed rest	NULL
6	emergency room	2000.00	emergency occupancy	NULL
7	urine sample test	60.60	urine test	5
8	pregnancy urine test	30.00	urine test	5
9	vitamin levels blood	200.00	blood services	NULL

This statement changes shifts of all ”Orthopedics” and ”Dermatologists” to the ”night” shift:

```
UPDATE Doctor
SET shift = 'night'
WHERE speciality = 'Orthopedics' or speciality = 'Dermatology';
```

Doc_ID	Doc_F_Name	Doc_L_Name	Speciality	Shift
1	Svend	Svendsen	Dermatology	Night
2	John-John	of Arc	Pediatrics	Noon
3	Line	Gale	Dermatology	Night
4	Eric	Milton	Psychiatry	Morning
5	Bill	Svendsen	Anesthesiology	Morning
6	Joshua	Cooper	Surgery	Night
7	Frank	Roam	Orthopedics	Night
8	Anders Aaboe	Andersen	Surgery	Night
9	Inger	Yale	Obstetrics	Morning
10	Thomas	Nielsen	Orthopedics	Night

Delete statements

This statement removes the Patients with reception dates before 2022:

```
DELETE from Patient
WHERE reception_date < "2022-01-01";
```

Pat_ID	Pat_F_Name	Pat_L_Name	Birth_Date	Age	Phone_Number	Reception_Date	Discharge_Date
1	Henry	Bertstein	1983-03-21	40	+4560878703	2022-12-21	2023-01-05
2	Adam	Hansen	1945-05-01	77	+4545672005	2022-12-21	2023-01-05
4	Nima	Andersen	1997-05-11	25	+4547601112	2022-12-21	2023-01-05
6	Alberto	Milano	1955-12-12	67	NULL	2022-12-21	NULL
7	Mette	Berg Hansen	1962-03-20	61	+4589897703	2023-10-24	2023-01-05
8	Gertrud	Samuelsen	1938-09-01	84	+4544805413	2022-12-21	NULL
9	Thomas William	Helming	1980-01-30	43	+4567838376	2023-11-21	NULL
10	Niels	Gram Petersen	1959-05-04	63	+4557809070	2022-11-21	2023-01-05

This statement removes all the devices belong to ”gynaecology” Department:

```
DELETE Device
FROM device NATURAL JOIN department
WHERE dept_name = 'gynaecology';
```

Dev_ID	Dev_name	Doc_ID	Dept_ID
1	MRI scanner	1	5
2	Ultrasound machine	3	7
3	oxygen concentrator	2	8
5	Urine Analyser	6	4
6	Hematology analyzer	4	1