# Databases Autumn 2025

# Hand-In Exercise 1

October 3, 2025

Aiysha Frutiger
Jannick Seper
Luis Tritschler

| **Total Points** | |
|---|---|

| **Task** | **Points** |
|---|---|
| | |

Aiysha Frutiger
Jannick Seper
Luis Tritschler



# Task 1

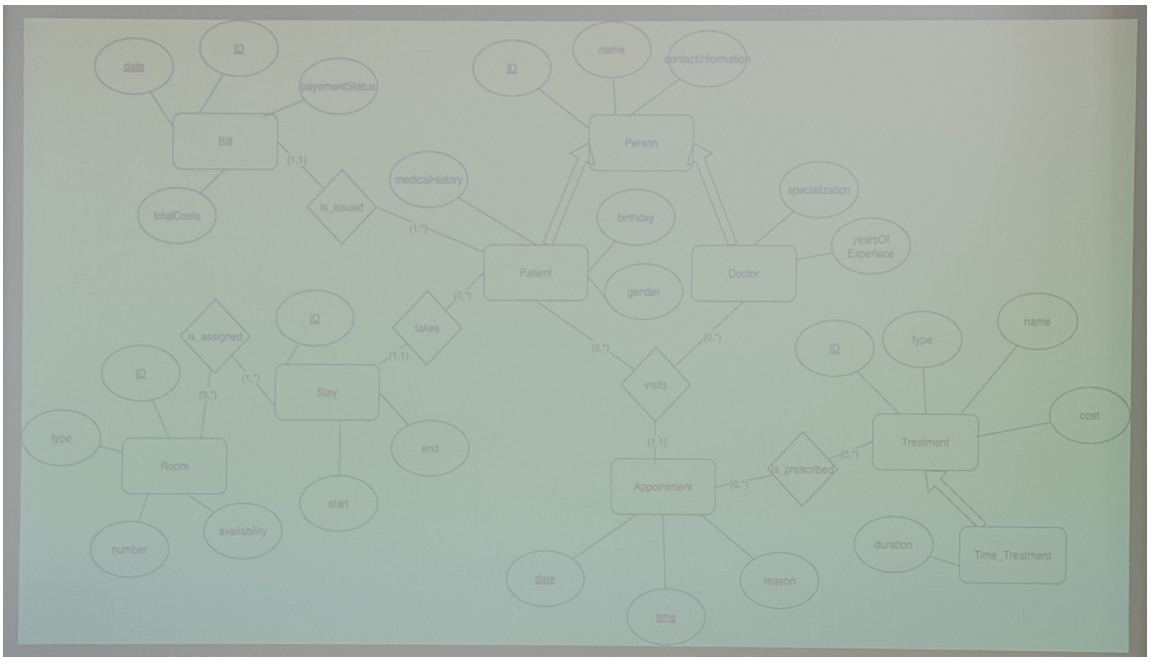


Figure 1: EER

# Task 2

Out of the Picture:

| | |
|---|---|
| Person | (ID, Name, ContactInformation) |
| Doctor | (ID, Name, ContactInformation, Specialization, YearsOfExperience) |
| Patient | (ID, Name, ContactInformation, Birthday, MedicalHistory, Gender) |
| Appointment | (Date, Time, Reason) |
| Stay | (ID, Start, End) |
| Treatment | (ID, type, name, cost) |
| Time_Treatment | (ID, type, name, cost, duration) |
| Bill | (ID, Date, TotalCosts, PaymentStatus) |
| Room | (ID, Type, Number, Availability) |
| | |
| visits | (Patient_ID, Doctor_ID, Date, Time) |
| takes | (Stay_ID, Patient_ID) |
| is_prescribed | (Treatment_ID, Date, Time) |
| is_assigned | (Room_ID, Stay_ID) |
| is_issued | (Patient_ID, Bill_ID, Date) |

Merged:

| | |
|---|---|
| Person | (ID, Name, ContactInformation, Birthday) |
| Doctor | (ID, Name, ContactInformation, Birthday, Specialization, YearsOfExperience) |
| Patient | (ID, Name, ContactInformation, Birthday, MedicalHistory) |
| Appointment | (Date, Time, Reason, Patient_ID, Doctor_ID) |
| Stay | (ID, Start, End, Patient_ID) |
| Treatment | (ID, type, name, cost) |
| Time_Treatment | (ID, type, name, cost, duration) |
| Bill | (ID, Date, TotalCosts, PaymentStatus, Patient_ID) |
| Room | (ID, Type, Number, Availability) |
| | |
| is_prescribed | (Treatment_ID, Date, Time) |
| is_assigned | (Room_ID, Stay_ID) |

# Task 3

Included SQL snippets to the questions:

```sql
-- Exercise 3a
SELECT Name, Specialization, Years_of_Experience
FROM Doctor
WHERE Years_of_Experience > 5;
```

```sql
-- Exercise 3b
SELECT D.Name, COUNT(A.Patient_ID) AS Total_Patients
FROM Doctor D
JOIN Appointment A ON D.Doctor_ID = A.Doctor_ID
GROUP BY D.Name;
```

```sql
-- Exercise 3c
SELECT D.Name, COUNT(DISTINCT A.Patient_ID) AS Total_Patients
FROM Doctor D
JOIN Appointment A ON D.Doctor_ID = A.Doctor_ID
GROUP BY D.Name
HAVING COUNT(DISTINCT A.Patient_ID) > 10;
```

# Task 4

Included SQL snippets to create DB (100% ChatGPT):

```sql
BEGIN;

DROP TABLE IF EXISTS is_assigned        CASCADE;
DROP TABLE IF EXISTS is_prescribed      CASCADE;
DROP TABLE IF EXISTS appointment        CASCADE;
DROP TABLE IF EXISTS bill               CASCADE;
DROP TABLE IF EXISTS stay               CASCADE;
DROP TABLE IF EXISTS room               CASCADE;
DROP TABLE IF EXISTS time_treatment     CASCADE;
DROP TABLE IF EXISTS treatment          CASCADE;
DROP TABLE IF EXISTS patient            CASCADE;
DROP TABLE IF EXISTS doctor             CASCADE;
DROP TABLE IF EXISTS person             CASCADE;

CREATE TABLE person (
    id                  INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    name                TEXT        NOT NULL,
    contact_information TEXT        NOT NULL,
    birthday            DATE        NOT NULL
);

CREATE TABLE doctor (
    id                  INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    name                TEXT        NOT NULL,
    contact_information TEXT        NOT NULL,
    birthday            DATE        NOT NULL,
    specialization      TEXT        NOT NULL,
    years_of_experience INTEGER     NOT NULL CHECK (years_of_experience >= 0)
);

CREATE TABLE patient (
    id                  INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    name                TEXT        NOT NULL,
    contact_information TEXT        NOT NULL,
    birthday            DATE        NOT NULL,
    medical_history     TEXT
);

CREATE TABLE treatment (
    id      INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    type    TEXT            NOT NULL,
    name    TEXT            NOT NULL,
```

```
43      cost      NUMERIC(12,2)    NOT NULL CHECK (cost >= 0)
44  );
45
46  CREATE TABLE time_treatment (
47      id          INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
48      type        TEXT             NOT NULL,
49      name        TEXT             NOT NULL,
50      cost        NUMERIC(12,2)    NOT NULL CHECK (cost >= 0),
51      duration    INTERVAL         NOT NULL
52  );
53
54  CREATE TABLE room (
55      id              INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
56      type            TEXT         NOT NULL,
57      number          TEXT         NOT NULL,
58      availability    BOOLEAN      NOT NULL DEFAULT TRUE
59  );
60
61  CREATE TABLE appointment (
62      appointment_date  DATE     NOT NULL,
63      appointment_time  TIME     NOT NULL,
64      reason            TEXT,
65      patient_id        INTEGER NOT NULL,
66      doctor_id         INTEGER NOT NULL,
67      CONSTRAINT pk_appointment PRIMARY KEY (appointment_date, appointment_time),
68      CONSTRAINT fk_appointment_patient
69          FOREIGN KEY (patient_id) REFERENCES patient(id),
70      CONSTRAINT fk_appointment_doctor
71          FOREIGN KEY (doctor_id)  REFERENCES doctor(id)
72  );
73
74  CREATE TABLE stay (
75      id          INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
76      start_date  TIMESTAMP   NOT NULL,
77      end_date    TIMESTAMP   NOT NULL,
78      patient_id  INTEGER     NOT NULL,
79      CONSTRAINT fk_stay_patient
80          FOREIGN KEY (patient_id) REFERENCES patient(id),
81      CONSTRAINT chk_stay_interval CHECK (end_date > start_date)
82  );
83
84  CREATE TABLE bill (
85      id              INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
86      date_issued     DATE           NOT NULL,
87      total_costs     NUMERIC(12,2) NOT NULL CHECK (total_costs >= 0),
88      payment_status  TEXT           NOT NULL,
```

```
    patient_id      INTEGER         NOT NULL,
    CONSTRAINT fk_bill_patient
        FOREIGN KEY (patient_id) REFERENCES patient(id)
);

CREATE TABLE is_assigned (
    room_id  INTEGER NOT NULL,
    stay_id  INTEGER NOT NULL,
    CONSTRAINT pk_is_assigned PRIMARY KEY (room_id, stay_id),
    CONSTRAINT fk_is_assigned_room
        FOREIGN KEY (room_id) REFERENCES room(id),
    CONSTRAINT fk_is_assigned_stay
        FOREIGN KEY (stay_id) REFERENCES stay(id)
);

CREATE TABLE is_prescribed (
    treatment_id       INTEGER NOT NULL,
    appointment_date   DATE    NOT NULL,
    appointment_time   TIME    NOT NULL,
    CONSTRAINT pk_is_prescribed PRIMARY KEY (treatment_id, appointment_date, appointmen
    CONSTRAINT fk_is_prescribed_treatment
        FOREIGN KEY (treatment_id) REFERENCES treatment(id),
    CONSTRAINT fk_is_prescribed_appointment
        FOREIGN KEY (appointment_date, appointment_time)
        REFERENCES appointment(appointment_date, appointment_time)
);

CREATE INDEX idx_appointment_patient ON appointment(patient_id);
CREATE INDEX idx_appointment_doctor  ON appointment(doctor_id);
CREATE INDEX idx_stay_patient        ON stay(patient_id);
CREATE INDEX idx_bill_patient        ON bill(patient_id);

COMMIT;
```

# Task 5

Um das Schema auf einem MacBook mit PostgreSQL auszuführen, sind folgende Schritte nötig:

1. **PostgreSQL installieren** (falls noch nicht vorhanden):

2. **Neue Datenbank anlegen**:

   ```
   psql postgres
   ```

   und im Prompt:

   ```
   CREATE DATABASE hospital;          #creates DB
   \q                                 #quits psql
   ```

3. **Schema-Datei ausführen**:

   ```
   cd /Pfad/zum/Ordner
   psql -d hospital -f schema.sql     #imports schema
   ```

4. **Tabellen testen**:

   ```
   psql hospital
   ```

   und im Prompt:

   ```
   \dt              -- zeigt alle Tabellen
   \d appointment   -- zeigt Struktur von appointment
   ```