02 E1 Project

Information Systems and Databases

Instituto Superior Técnico

December 2, 2023

Group 2:

- 66325 Tomás Marques Videira Fonseca (100%) tomas.mvf@gmail.com
- 96135 Afonso Brito Caiado Correia Alemão (100%) afonso.alemao@tecnico.ulisboa.pt
- 96317 Rui Pedro Canário Daniel (100%) ruipcdaniel@tecnico.ulisboa.pt

Teachers:

- Flávio Martins
- Alessandro Gianola
- Francisco Regateiro

Lab Shift number: PB03

Project E1

PART I – E-R Model

1. Proposed database design

This E-R model describes our proposed database design. Every design decision that can be captured in the E-R model is represented in the diagram. We also include additional integrity constraints not captured in the E-R model.

We define Appointment as a weak entity, because its partial key {date} is not sufficient to uniquely discriminate each of the instances, so Appointment is associated to the Strong entity Doctor which has the primary key {vat}. That way, Appointment's primary key is the set of attributes {date, Doctor(vat)}.

Using a similar approach, we define Tooth as a weak entity, because its partial key {mouth_quad, number} is not sufficient to uniquely discriminate each of the instances, so Tooth is associated to the Strong entity Consultation which has the primary key {date, doctor_vat}. That way, Tooth's primary key is the set of attributes {mouth_quad, number, Consultation(date, doctor_vat)}.

In the same way, we define Report as a weak entity, because its partial key {date} is not sufficient to uniquely discriminate each of the instances, so Report is associated to the Strong entity Trainee which has the primary key {vat}. That way, Report's primary key is the set of attributes {date, Trainee(vat)}.

In order to standardize the Diagnosis code we chose to use the restricted controlled vocabulary SNODENT.

According to the Indicative Field Sizes tables given in the Course Slides, we have established that a Doctor's email must have length greater than or equal to 6, a Person's phone number must have length greater than or equal to 3, and a Person's postal code (zip) must have length greater than or equal to 2.

Furthermore, since the each prescription 'dosage' can vary in scale (e.g., 'mg', 'g') we chose to store it in text based format.

In the Relational Model, instead of creating a table for the ‘Address’ entity, we extend the table 'Person' with the attributes of the Address ({street, zip, city}). This is feasible because both every record of Person must be connected to one Address, and every Address must be associated with at least one person. That way, a table for the ‘Address’ entity would be redundant.

E-R Model

IC-1: a Person phone must be unique.

IC-2: an Employee iban must be unique.

IC-3: a Doctor email must be unique.

IC-4: a Supervisor can only write a Report which evaluates a Trainee he/she supervises.

IC-5: Reports regarding the same Supervisor and Trainee must have dates at least 2 months apart.

IC-6: a Report score must be in the range from 1 to 5.

IC-7: a Diagnosis code must be restricted to the controlled vocabulary SNODENT.

IC-8: Diagnosis are not allowed to relate to themselves.

IC-9: Tooth's mouth quadrant and number must be according to the Palmer Notation Numbering System.

IC-10: Tooth's gap must be in millimeters.

IC-11: Tooth's gap must be greater than or equal to 0.

IC-12: a Consultation can only include a Tooth if that same Consultation involves a Procedure of type 'Dental Charting'.

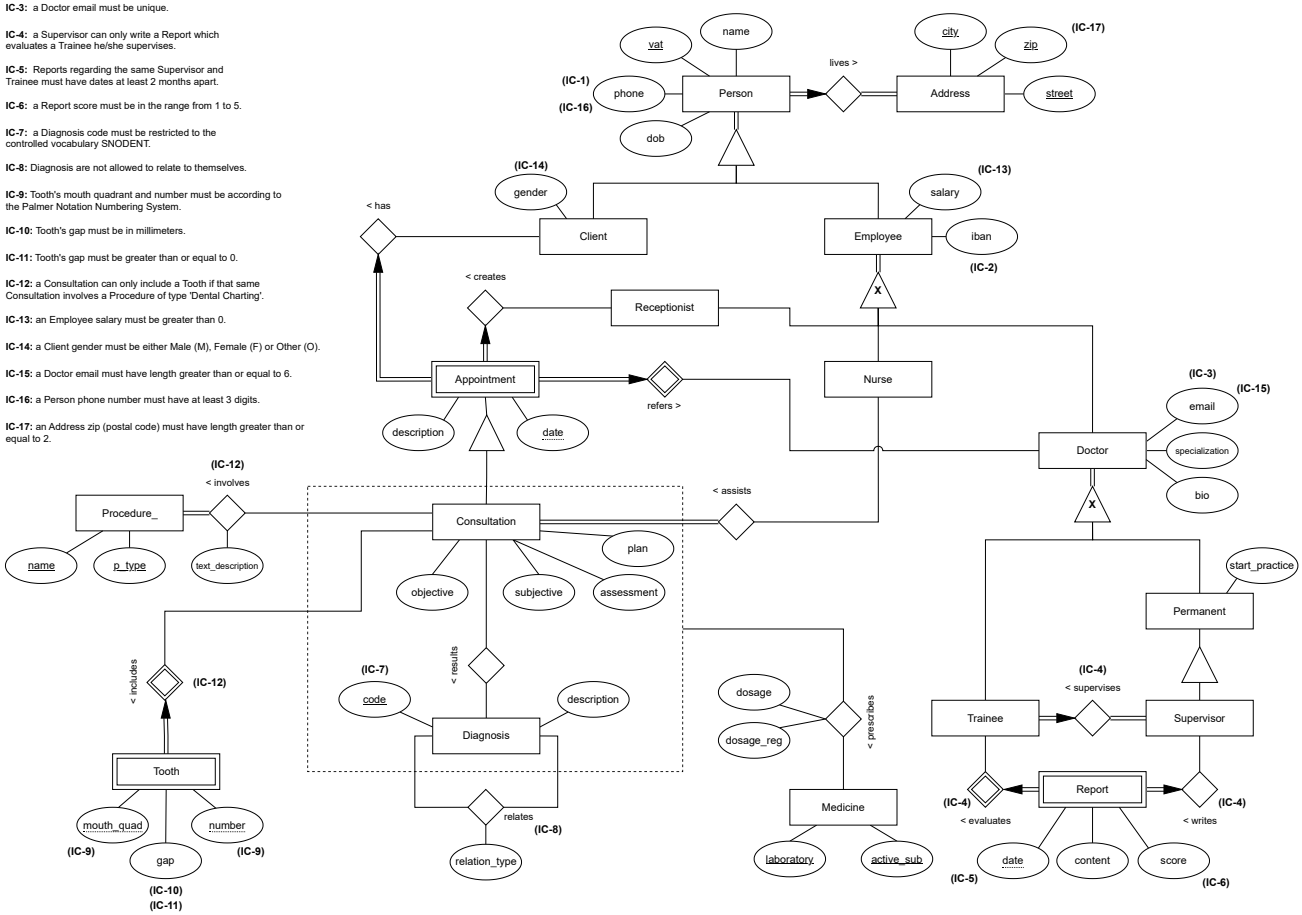
IC-13: an Employee salary must be greater than 0.

IC-14: a Client gender must be either Male (M), Female (F) or Other (O).

IC-15: a Doctor email must have length greater than or equal to 6.

IC-16: a Person phone number must have at least 3 digits.

IC-17: an Address zip (postal code) must have length greater than or equal to 2.



PART II – Relational Model

Database Schema

1. Create the tables and integrity constraints corresponding to the relational database schema obtained.

In [ ]:
```
%load_ext sql
%sql postgresql+psycopg://db:db@postgres/db
```

In [ ]:
```
%%sql

DROP TABLE IF EXISTS prescribes;
DROP TABLE IF EXISTS Medicine;
DROP TABLE IF EXISTS Tooth;
DROP TABLE IF EXISTS involves;
DROP TABLE IF EXISTS Procedure_;
DROP TABLE IF EXISTS relates;
DROP TABLE IF EXISTS results;
DROP TABLE IF EXISTS Diagnosis;
DROP TABLE IF EXISTS assists;
DROP TABLE IF EXISTS Consultation;
DROP TABLE IF EXISTS Report;
DROP TABLE IF EXISTS Trainee;
DROP TABLE IF EXISTS Supervisor;
DROP TABLE IF EXISTS Permanent;
DROP TABLE IF EXISTS Appointment;
DROP TABLE IF EXISTS Doctor;
DROP TABLE IF EXISTS Nurse;
DROP TABLE IF EXISTS Receptionist;
DROP TABLE IF EXISTS Client;
DROP TABLE IF EXISTS Employee;
DROP TABLE IF EXISTS Person;


CREATE TABLE Person (
    vat VARCHAR(20),
    name VARCHAR(80) NOT NULL,
    dob DATE NOT NULL,
    phone VARCHAR(15) NOT NULL,
    street VARCHAR(255) NOT NULL,
    zip VARCHAR(12) NOT NULL,
    city VARCHAR(30) NOT NULL,
    PRIMARY KEY(vat),
    UNIQUE (phone),
    CHECK (LENGTH(phone) >= 3),
    CHECK (LENGTH(zip) >= 2)
    /* -- Every Person must exist either in the table 'Client' or in the table 'Employee' */
);

CREATE TABLE Employee (
    vat VARCHAR(20),
    salary NUMERIC (16,4) NOT NULL,
    iban VARCHAR(30) NOT NULL,
    PRIMARY KEY(vat),
    FOREIGN KEY(vat) REFERENCES Person(vat),
    UNIQUE (iban),
    CHECK (salary > 0)
    /* -- No Employee can exist at the same time in both the table 'Nurse'
    and in the table 'Doctor' */
    /* -- No Employee can exist at the same time in both the table 'Receptionist'
    and in the table 'Doctor' */
```

```
    /* -- No Employee can exist at the same time in both the table 'Nurse' and
    in the table 'Receptionist' */
    /* -- Every Employee must exist either in the table 'Nurse' or in the table 'Doctor'
    or in the table 'Receptionist' */
);

CREATE TABLE Client (
    vat VARCHAR(20),
    gender CHAR(1) NOT NULL,
    PRIMARY KEY(vat),
    FOREIGN KEY(vat) REFERENCES Person(vat),
    CHECK (gender in ('M', 'F', 'O') )
);

CREATE TABLE Receptionist (
    vat VARCHAR(20),
    PRIMARY KEY(vat),
    FOREIGN KEY(vat) REFERENCES Employee(vat)
);

CREATE TABLE Nurse (
    vat VARCHAR(20),
    PRIMARY KEY(vat),
    FOREIGN KEY(vat) REFERENCES Employee(vat)
);

CREATE TABLE Doctor (
    vat VARCHAR(20),
    email VARCHAR(254) NOT NULL,
    specialization VARCHAR(200) NOT NULL,
    bio TEXT NOT NULL,
    PRIMARY KEY(vat),
    FOREIGN KEY(vat) REFERENCES Employee(vat),
    UNIQUE(email),
    CHECK (LENGTH(email) >= 6)
    /* -- No Doctor can exist at the same time in both the table 'Permanent'
    and in the table 'Trainee' */
    /* -- Every Doctor must exist either in the table 'Permanent' or
    in the table 'Trainee' */
);

CREATE TABLE Permanent (
    vat VARCHAR(20),
    start_practice DATE NOT NULL,
    PRIMARY KEY(vat),
    FOREIGN KEY(vat) REFERENCES Doctor(vat)
);

CREATE TABLE Supervisor (
    vat VARCHAR(20),
    PRIMARY KEY(vat),
    FOREIGN KEY(vat) REFERENCES Permanent(vat)
    /* -- Every Supervisor must exist in the table 'Trainee' */
);

CREATE TABLE Trainee (
    vat VARCHAR(20),
    supervisor_vat VARCHAR(20) NOT NULL,
    PRIMARY KEY(vat),
    FOREIGN KEY(vat) REFERENCES Doctor(vat),
```

```
    FOREIGN KEY(supervisor_vat) REFERENCES Supervisor(vat)
);

CREATE TABLE Report (
    content TEXT NOT NULL,
    date DATE,
    score NUMERIC (3,2) NOT NULL,
    trainee_vat VARCHAR(20),
    supervisor_vat VARCHAR(20) NOT NULL,
    PRIMARY KEY(date, trainee_vat),
    FOREIGN KEY(trainee_vat) REFERENCES Trainee(vat),
    FOREIGN KEY(supervisor_vat) REFERENCES Supervisor(vat),
    CHECK (score >= 1 and score <= 5)
    /* -- Every Report regarding the same Supervisor and Trainee must have dates
    at least 2 months apart */
    /* -- Every Report must have a Supervisor and a Trainee, in which that Trainee
    has that same Supervisor in the table 'Trainee' */
);

CREATE TABLE Appointment (
    description TEXT NOT NULL,
    date TIMESTAMP,
    doctor_vat VARCHAR(20),
    client_vat VARCHAR(20) NOT NULL,
    receptionist_vat VARCHAR(20) NOT NULL,
    PRIMARY KEY(date, doctor_vat),
    FOREIGN KEY(doctor_vat) REFERENCES Doctor(vat),
    FOREIGN KEY(client_vat) REFERENCES Client(vat),
    FOREIGN KEY(receptionist_vat) REFERENCES Receptionist(vat)
);

CREATE TABLE Consultation (
    subjective TEXT NOT NULL,
    objective TEXT NOT NULL,
    assessment TEXT NOT NULL,
    plan TEXT NOT NULL,
    doctor_vat VARCHAR(20),
    date TIMESTAMP,
    PRIMARY KEY(date, doctor_vat),
    FOREIGN KEY(date, doctor_vat) REFERENCES Appointment(date, doctor_vat)
    /* -- Every Consultation must exist in the table 'assists' */
);

CREATE TABLE assists (
    nurse_vat VARCHAR(20),
    date TIMESTAMP,
    doctor_vat VARCHAR(20),
    PRIMARY KEY(nurse_vat, date, doctor_vat),
    FOREIGN KEY(nurse_vat) REFERENCES Nurse(vat),
    FOREIGN KEY(date, doctor_vat) REFERENCES Consultation(date, doctor_vat)
);

CREATE TABLE Diagnosis (
    description TEXT NOT NULL,
    code VARCHAR(7),
    PRIMARY KEY(code)
    /* -- Every Diagnosis code must be restricted to the controlled vocabulary SNODENT */
);

CREATE TABLE results (
```

```
    doctor_vat VARCHAR(20),
    date TIMESTAMP,
    code VARCHAR(7),
    PRIMARY KEY(date, doctor_vat, code),
    FOREIGN KEY(code) REFERENCES Diagnosis(code),
    FOREIGN KEY(date, doctor_vat) REFERENCES Consultation(date, doctor_vat)
);

CREATE TABLE relates (
    code_1 VARCHAR(7),
    code_2 VARCHAR(7),
    relation_type VARCHAR(200) NOT NULL,
    PRIMARY KEY(code_1, code_2),
    FOREIGN KEY(code_1) REFERENCES Diagnosis(code),
    FOREIGN KEY(code_2) REFERENCES Diagnosis(code),
    CHECK(code_1 <> code_2)
);

CREATE TABLE Procedure_ (
    name VARCHAR(200),
    p_type VARCHAR(150),
    PRIMARY KEY(name, p_type)
    /* -- Every Procedure_ must exist in the table 'involves' */
);

CREATE TABLE involves (
    doctor_vat VARCHAR(20),
    date TIMESTAMP,
    name VARCHAR(200),
    p_type VARCHAR(150),
    text_description TEXT NOT NULL,
    PRIMARY KEY(date, doctor_vat, name, p_type),
    FOREIGN KEY(date, doctor_vat) REFERENCES Consultation(date, doctor_vat),
    FOREIGN KEY(name, p_type) REFERENCES Procedure_(name, p_type)
);

CREATE TABLE Tooth (
    mouth_quad CHAR(1),
    gap NUMERIC(6,2) NOT NULL,
    number CHAR(1),
    doctor_vat VARCHAR(20),
    date TIMESTAMP,
    PRIMARY KEY(mouth_quad, number, date, doctor_vat),
    FOREIGN KEY(date, doctor_vat) REFERENCES Consultation(date, doctor_vat),
    CHECK (gap >= 0),
    CHECK (mouth_quad IN ('┐','┘','┌','└')),
    CHECK (number IN ('A','B','C','D','E','1','2','3','4','5','6','7','8'))
    /* -- Every Tooth gap must be in milimeters */
    /* -- Every Tooth can only be associated with a Consultation, if that same
    Consultation has a Procedure of p_type 'Dental Charting' in the table 'Consultation' */
);

CREATE TABLE Medicine (
    laboratory VARCHAR(255),
    active_sub VARCHAR(200),
    PRIMARY KEY(laboratory, active_sub)
);

CREATE TABLE prescribes (
    laboratory VARCHAR(255),
```

```
    active_sub VARCHAR(200),
    doctor_vat VARCHAR(20),
    date TIMESTAMP,
    code VARCHAR(7),
    dosage VARCHAR(80) NOT NULL,
    dosage_regime TEXT NOT NULL,
    PRIMARY KEY(laboratory, active_sub, date, doctor_vat, code),
    FOREIGN KEY(laboratory, active_sub) REFERENCES Medicine(laboratory, active_sub),
    FOREIGN KEY(date, doctor_vat, code) REFERENCES results(date, doctor_vat, code)
);
```