SISTEMAS DE INFORMAÇÃO E
BASES DE DADOS

Assignment 2

Implementing the Database

Authors:

Niewiński Łukasz 95018
Lewandowska Gabriela 94898
Gualdi Philipp 95059

Group:

47

Lisboa, March 3, 2020

1 Database Schema

The developed code during the project can be found under the link:

`https://github.com/LukaszNiewinski/Database_model_dental_clinic`

IMPORTANT: Insert queries are avaible under the github link repo, its because of its huge volume.

```
CREATE TABLE employee
(
    VAT        char(15),
    name       varchar(150) NOT NULL,
    birth_date date,
    street     varchar(150),
    city       varchar(150),
    zip        varchar(15),
    IBAN       varchar(20)  NOT NULL,
    salary     varchar(15),
    PRIMARY KEY (VAT),
    UNIQUE (IBAN),
    CHECK ( salary >= 0 )
);

CREATE TABLE nurse
(
    VAT char(15),
    PRIMARY KEY (VAT),
    Foreign key (VAT) references employee (VAT)
);
CREATE TABLE receptionist
(
    VAT char(15),
    PRIMARY KEY (VAT),
    Foreign key (VAT) references employee (VAT)
);

CREATE TABLE doctor
(
    VAT            char(15),
    specialization char(30),
    biography      TEXT,
    email          char(30) NOT NULL,
    PRIMARY KEY (VAT),
    FOREIGN KEY (VAT) references employee (VAT) ON DELETE CASCADE,
    UNIQUE (email)
);
```

```
CREATE TABLE permanent_doctor
(
    years TINYINT  NOT NULL,
    VAT   char(15) NOT NULL,
    primary key (VAT),
    foreign key (VAT) references doctor (VAT) ON DELETE CASCADE
);

CREATE TABLE trainee_doctor
(
    VAT            char(30),
    VAT_supervisor char(30) NOT NULL,
    primary key (VAT),
    foreign key (VAT) references doctor (VAT) ON DELETE CASCADE,
    foreign key (VAT_supervisor) references permanent_doctor (VAT) ON DELETE CASCADE
);

CREATE TABLE supervision_report
(
    VAT            char(15),
    date_timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    description    TEXT,
    evaluation     ENUM ('1','2','3','4','5'),
    primary key (VAT, date_timestamp),
    foreign key (VAT) references trainee_doctor (VAT) ON DELETE CASCADE
);

CREATE TABLE phone_number_employee
(
    VAT   char(15),
    phone char(15),
    primary key (VAT, phone),
    foreign key (VAT) references employee (VAT) ON DELETE CASCADE
);

CREATE TABLE client
(
    VAT        char(15),
    name       char(30),
    birth_date date NOT NULL,
    street     char(30),
    city       char(30),
    zip        char(15),
    gender     ENUM ('man', 'woman', ''),
    primary key (VAT),
    age        int
);
```

```
CREATE TABLE phone_number_client
(
    VAT   char(15),
    phone char(15),
    primary key (VAT, phone),
    foreign key (VAT) references client (VAT)
);

CREATE TABLE appointment
(
    VAT_doctor     char(15),
    date_timestamp timestamp,
    description    TEXT,
    VAT_client     char(15),
    primary key (VAT_doctor, date_timestamp),
    foreign key (VAT_doctor) references doctor (VAT) ON DELETE CASCADE,
    foreign key (VAT_client) references client (VAT)
);

CREATE TABLE consultation
(
    VAT_doctor     char(15),
    date_timestamp timestamp,
    SOAP_S         MEDIUMTEXT,
    SOAP_O         MEDIUMTEXT,
    SOAP_A         MEDIUMTEXT,
    SOAP_P         MEDIUMTEXT,
    primary key (VAT_doctor, date_timestamp),
    foreign key (VAT_doctor, date_timestamp)
        references appointment (VAT_doctor, date_timestamp)
);

CREATE TABLE consultation_assistant
(
    VAT_doctor     char(15),
    date_timestamp timestamp,
    VAT_nurse      char(15),
    primary key (VAT_doctor, date_timestamp, VAT_nurse),
    foreign key (VAT_nurse) references nurse (VAT),
    foreign key (VAT_doctor, date_timestamp)
        references consultation (VAT_doctor, date_timestamp)
);

CREATE TABLE diagnostic_code
(
    ID          char(15),
    description MEDIUMTEXT,
    primary key (ID)
);

CREATE TABLE diagnostic_code_relation
(
    ID1  char(15),
    ID2  char(15),
    type char(30),
    primary key (ID1, ID2),
    foreign key (ID1) references diagnostic_code (ID),
    foreign key (ID2) references diagnostic_code (ID)
);

CREATE TABLE consultation_diagnostic
(
    VAT_doctor     char(15),
    date_timestamp timestamp,
    ID             char(15),
    primary key (VAT_doctor, date_timestamp, ID),
    foreign key (VAT_doctor, date_timestamp)
        references consultation (VAT_doctor, date_timestamp),
    foreign key (ID) references diagnostic_code (id)
);

CREATE TABLE medication
(
    name char(30),
    lab  char(30),
    primary key (name, lab)
);

CREATE TABLE prescription
(
    name           char(30),
    lab            char(30),
    VAT_doctor     char(15),
    ID             char(15),
    date_timestamp timestamp,
    description    LONGTEXT,
    primary key (name, lab, VAT_doctor, date_timestamp, ID),
    foreign key (VAT_doctor, date_timestamp, ID)
        references consultation_diagnostic (vat_doctor, date_timestamp, id),
    foreign key (name, lab) references medication (name, lab)
);

CREATE TABLE procedure_clinic
```

```
(
    name char(30),
    type char(30),
    primary key (name)
);

CREATE TABLE procedure_in_consultation
(
    name           char(30),
    VAT_doctor     char(15),
    date_timestamp timestamp,
    description    LONGTEXT,
    primary key (name, VAT_doctor, date_timestamp),
    foreign key (VAT_doctor, date_timestamp)
        references consultation (VAT_doctor, date_timestamp),
    foreign key (name) references procedure_clinic (name)
);

CREATE TABLE procedure_radiology
(
    name           char(15),
    file           char(100),
    VAT_doctor     char(15),
    date_timestamp timestamp,
    primary key (name, file, VAT_doctor, date_timestamp),
    foreign key (name, VAT_doctor, date_timestamp)
        references procedure_in_consultation (name, VAT_doctor, date_timestamp)
);

CREATE TABLE teeth
(
    quadrant ENUM ('1','2','3','4'),
    number   int,
    name     char(30),
    primary key (quadrant, number)
);

CREATE TABLE procedure_charting
(
    name           char(30),
    VAT            char(15),
    date_timestamp timestamp,
    quadrant       ENUM ('1', '2', '3', '4'),
    number         int,
    description    TEXT,
    measure        INT,
    primary key (name, VAT, date_timestamp, quadrant, number),
    foreign key (name, VAT, date_timestamp)
        references procedure_in_consultation (NAME, VAT_DOCTOR, DATE_TIMESTAMP),
    foreign key (quadrant, number) references teeth (quadrant, number)
);
```

2 SQL queries

1. Query

List the VAT, name, and phone number(s) for all clients that had consultations with the doctor named Jane Sweettooth. The list should be presented according to the alphabetical order for the names.

```
SELECT c.VAT as client_vat,
       c.name as client_name,
       GROUP_CONCAT(DISTINCT p_c.phone) as client_phone_numbers
FROM client c
         join appointment a on c.VAT = a.VAT_client
         join doctor d on a.VAT_doctor = d.VAT
         join employee e on d.VAT = e.VAT,
     phone_number_client p_c
where e.name
    like 'Jane Sweettoth'
  and a.date_timestamp <= NOW()
  and c.VAT = p_c.VAT
GROUP BY c.VAT, c.name
ORDER BY c.name;
```

2. Query

List the name of all trainee doctors with reports associated to an evaluation score below the value of three, or with a description that contains the term insufficient. The name should be presented together with the VAT of the trainee, the name for the doctor that made the evaluation, the evaluation score, and the textual description for the evaluation report. Results should be sorted according to the evaluation score, in descending order.

```
SELECT DISTINCT e.name         as traine_name,
                e.VAT          as traine_VAT,
                e_d.name       as doctor_name,
                sr.evaluation,
                sr.description as evaluation_description
FROM employee e
         join trainee_doctor td on e.VAT = td.VAT
         join supervision_report sr on td.VAT = sr.VAT
         join employee e_d on e_d.VAT = td.VAT_supervisor
WHERE sr.evaluation < 3
   or sr.description like '%insufficient%'
ORDER BY sr.evaluation DESC;
```

3. Query

List the name, city, and VAT for all clients where the most recent consultation has the objective part of the SOAP note mentioning the terms gingivitis or periodontitis.

```
SELECT DISTINCT c.name as client_name,
                c.city as client_city,
                c.VAT as client_VAT
FROM client c
         join appointment a on c.VAT = a.VAT_client
         join consultation con on a.VAT_doctor = con.VAT_doctor
    and a.date_timestamp = con.date_timestamp
         JOIN (
    SELECT cl.VAT as client_vat, MAX(con.date_timestamp) as recent_consultation
    FROM consultation con
             join appointment ap on con.VAT_doctor = ap.VAT_doctor
        and con.date_timestamp = ap.date_timestamp
             join client cl on ap.VAT_client = cl.VAT
    GROUP BY cl.VAT) as r_c on c.VAT = r_c.client_vat
    and con.date_timestamp = r_c.recent_consultation
WHERE con.SOAP_O like '%gingivitis%'
   or con.SOAP_O like '%periodontitis%';
```

4. Query

List the name, VAT and address (i.e., street, city and zip) of all clients of the clinic that have had appointments but that never had a consultation (i.e., clients that never showed to an appointment)

```
SELECT DISTINCT c.name as client_name,
                c.VAT as VAT_client,
                c.street, c.city,
                c.zip
from client c
         join appointment a on c.VAT = a.VAT_client
where a.date_timestamp < NOW()
  and (a.VAT_doctor, a.date_timestamp) not in
      (SELECT con.VAT_doctor, con.date_timestamp
        FROM consultation con);
```

5. Query

For each possible diagnosis, presenting the code together with the description, list the number of distinct medication names that have been prescribed to treat that condition. Sort the results according to the number of distinct medication names, in ascending order

```
SELECT d_c.ID                 as diagnosis_id,
       d_c.description        as diagnosis_description,
       COUNT(DISTINCT p.name) as number_medicaments_prescribed
from diagnostic_code d_c
         join consultation_diagnostic cd on d_c.ID = cd.ID
         join prescription p on cd.VAT_doctor = p.VAT_doctor
            and cd.date_timestamp = p.date_timestamp
            and cd.ID = p.ID
group by d_c.ID, d_c.description
order by COUNT(DISTINCT p.name) ASC;
```

6. Query

Present the average number of nurses/assistants, procedures, diagnostic codes, and prescriptions involved in consultations from the year 2019, respectively for clients belonging to two age groups: less or equal to 18 years old, and more than 18 years old.

```
SELECT DISTINCT 'Equal or Above 18'       as patients_age,
                AVG(ca1.count_nurse)      as avg_num_nurses,
                AVG(pic1.count_procedure) as avg_num_procedures,
                AVG(cd1.count_diagnosis)  as avg_num_diagnosis
from (SELECT ca.date_timestamp, ca.VAT_doctor, COUNT(ca.VAT_nurse) as count_nurse
      from consultation_assistant ca
               join appointment a on ca.date_timestamp = a.date_timestamp
               join client c on a.VAT_client = c.VAT
      where ca.date_timestamp >= DATE('2019-01-01')
        and (YEAR(CURDATE()) - YEAR(c.birth_date)) >= 18
      group by ca.date_timestamp, ca.VAT_doctor) as ca1,
     (SELECT pic.date_timestamp, pic.VAT_doctor, COUNT(pic.name) as count_procedure
      from procedure_in_consultation pic
               join appointment a on pic.date_timestamp = a.date_timestamp
               join client c on a.VAT_client = c.VAT
      where pic.date_timestamp >= DATE('2019-01-01')
        and (YEAR(CURDATE()) - YEAR(c.birth_date)) >= 18
      group by pic.date_timestamp, pic.VAT_doctor) as pic1,
     (SELECT cd.date_timestamp, cd.VAT_doctor, COUNT(cd.ID) as count_diagnosis
      from consultation_diagnostic cd
               join appointment a on cd.date_timestamp = a.date_timestamp
               join client c on a.VAT_client = c.VAT
      where cd.date_timestamp >= DATE('2019-01-01')
        and (YEAR(CURDATE()) - YEAR(c.birth_date)) >= 18
      group by cd.date_timestamp, cd.VAT_doctor) as cd1
UNION
SELECT 'Below 18' as Patient_age,
       AVG(ca2.count_nurse),
       AVG(pic2.count_procedure),
       AVG(cd2.count_diagnosis)
FROM (SELECT ca.date_timestamp, ca.VAT_doctor,
             COUNT(ca.VAT_nurse) as count_nurse
      FROM consultation_assistant ca
               join appointment a on ca.date_timestamp = a.date_timestamp
               join client c on a.VAT_client = c.VAT
      WHERE ca.date_timestamp >= DATE('2019-01-01')
        and (YEAR(CURDATE()) - YEAR(c.birth_date)) < 18
      GROUP BY ca.date_timestamp, ca.VAT_doctor) as ca2,
     (SELECT pic.date_timestamp,
             pic.VAT_doctor,
             COUNT(pic.name) as count_procedure
      FROM procedure_in_consultation pic
               join appointment a on pic.date_timestamp = a.date_timestamp
               join client c on a.VAT_client = c.VAT
      WHERE pic.date_timestamp >= DATE('2019-01-01')
        and (YEAR(CURDATE()) - YEAR(c.birth_date)) < 18
      group by pic.date_timestamp, pic.VAT_doctor) as pic2,
     (SELECT cd.date_timestamp,
             cd.VAT_doctor,
             COUNT(cd.ID) as count_diagnosis
      FROM consultation_diagnostic cd
               join appointment a on cd.date_timestamp = a.date_timestamp
               join client c on a.VAT_client = c.VAT
      WHERE cd.date_timestamp >= DATE('2019-01-01')
        and (YEAR(CURDATE()) - YEAR(c.birth_date)) < 18
      GROUP BY cd.date_timestamp, cd.VAT_doctor) as cd2;
```

7. Query

For each diagnostic code, present the name of the most common medication used to treat that condition (i.e., the medication name that more often appears associated to prescriptions for that diagnosis).

```
SELECT dc.ID as diagnostic_code, pre.name as most_freq_medicaments
FROM diagnostic_code dc
         join prescription pre on pre.ID = dc.ID
GROUP BY dc.ID, pre.name
HAVING COUNT(pre.name) >= ALL (SELECT COUNT(pre1.name)
                               FROM diagnostic_code dc1
                                        JOIN prescription pre1 on pre1.ID = dc1.ID
                               WHERE dc1.ID = dc.ID
                               GROUP BY dc1.ID, pre1.name);
```

8. Query

List, alphabetically, the names and labs for the medications that, in the year 2019, have been used to treat dental cavities", but have not been used to treat any infectious disease". You can use the aforementioned names for searching diagnostic codes in the dataset, without considering relations (e.g., part-of relations) between diagnostic codes.

```
SELECT med.name, med.lab
from medication med
where (med.name, med.lab) in
      (SELECT med.name, med.lab
       from medication med
                join prescription p on med.name = p.name
```

```
                    and med.lab = p.lab
                join consultation_diagnostic cd
                     on p.VAT_doctor = cd.VAT_doctor
                    and p.date_timestamp = cd.date_timestamp
                    and p.ID = cd.ID
                join diagnostic_code dc on cd.ID = dc.ID
       where YEAR(cd.date_timestamp) = 2019
         and dc.description like 'dental cavities')
  and (med.name, med.lab) not in
      (SELECT med.name, med.lab
       from medication med
                join prescription p on med.name = p.name and med.lab = p.lab
                join consultation_diagnostic cd
                     on p.VAT_doctor = cd.VAT_doctor
                    and p.date_timestamp = cd.date_timestamp
                    and p.ID = cd.ID
                join diagnostic_code dc on cd.ID = dc.ID
       where YEAR(cd.date_timestamp) = 2019
         and dc.description like 'infectious disease')
order by med.name, med.lab;
```

9. Query

List the names and addresses of clients that have never missed an appointment in 2019 (i.e., the clients that, in the year 2019, have always appeared in all the consultations scheduled for them).

```
SELECT DISTINCT c.name as client_name, c.city, c.street, c.zip
from client c
         join appointment a on c.VAT = a.VAT_client
         join consultation c1 on a.VAT_doctor = c1.VAT_doctor
            and a.date_timestamp = c1.date_timestamp
where a.VAT_client not in (
    SELECT DISTINCT ap.VAT_client
    FROM appointment ap
         left outer join consultation con on ap.VAT_doctor = con.VAT_doctor
         and  ap.date_timestamp = con.date_timestamp
    WHERE con.VAT_doctor IS NULL
        AND YEAR(ap.date_timestamp) = 2019)
    AND YEAR(a.date_timestamp) = 2019;
```

3 Index

For the first query, we propose an index between the employee name and his/her ID, to increase the search time.

```
CREATE INDEX employee_name_idx
ON employee (name);
```

For the second query, we propose an index over the evaluation score to increase the search time. We choose the evaluation score instead, for example, the description. To create an index over a text it would require a large amount of storage and will not be effective as an index over the grades.

```
CREATE INDEX supervision_report_evaluation_idx
ON supervision_report (evaluation);
```

4 Update Queries

1. Query

Change the address of the doctor named Jane Sweettooth, to a different city and street of your choice.

```
UPDATE employee e
SET city = 'Plock', street = 'Grabowa'
WHERE name = 'Jane Sweettooth';
```

2. Query

Change the salary of all doctors that had more than 100 appointments in 2019. The new salaries should correspond to an increase in 5% from the old values.

```
UPDATE doctor d, employee e
SET e.salary = 1.05 * e.salary
WHERE e.vat = d.vat AND (
    SELECT COUNT(a.date_timestamp)
    FROM appointment a
     WHERE d.VAT = a.VAT_doctor) > 4;
```

3. Query

Delete the doctor named Jane Sweettooth from the database, removing also all the appointments and all the consultations (including the associated procedures, diagnosis and prescriptions) in which she was involved. Notice that if there are procedures/diagnosis that were only performed/assigned by this doctor, you should remove them also from the database.

```
SET @e_vat = (SELECT d.VAT
    FROM doctor d, employee e
    WHERE d.VAT = e.VAT AND
         e.name = 'Jane Sweettoth');

DELETE
FROM prescription
WHERE VAT_doctor = @e_vat;

DELETE
FROM consultation_assistant
WHERE VAT_doctor = @e_vat;

DELETE
FROM consultation_diagnostic
WHERE VAT_doctor = @e_vat;

DELETE
```

```
FROM procedure_charting
WHERE VAT= @e_vat;

DELETE
FROM procedure_in_consultation
WHERE VAT_doctor = @e_vat;

DELETE
FROM consultation
WHERE VAT_doctor = @e_vat;

DELETE
FROM employee
WHERE VAT = @e_vat;
```

4. Query

Find the diagnosis code corresponding to gingivitis. Create also a new diagnosis code corresponding to periodontitis. Change the diagnosis from gingivitis to periodontitis for all clients where, for the same consultation/diagnosis, a dental charting procedure shows a value above 4 in terms of the average gap between the teeth and the gums.

```
UPDATE consultation_diagnostic c_d
    join procedure_in_consultation pic
        on c_d.date_timestamp = pic.date_timestamp
    join procedure_charting pc
        on pic.name = pc.name
        AND pic.VAT_doctor = pc.VAT
        AND pic.date_timestamp = pc.date_timestamp
    join consultation c on c_d.VAT_doctor = c.VAT_doctor
        ANDc_d.date_timestamp = c.date_timestamp
    set c_d.ID = (
    SELECT dc.ID
    FROM diagnostic_code dc
    WHERE dc.description like '%periodontitis%')
WHERE c_d.ID = (
    SELECT dc.ID
    FROM diagnostic_code dc
    WHERE dc.description like '%gingivitis%')
    AND pc.measure >= 4;
```

5 Views

1. Query

```
dim_date(date_timestamp,day,month,year)
IC: date_timestamp corresponds to a date existing in consultations


CREATE VIEW dim_date AS
    SELECT c.date_timestamp as date_stamp,
           EXTRACT(DAY FROM c.date_timestamp) AS day,
           EXTRACT(MONTH FROM c.date_timestamp) AS month,
           EXTRACT(YEAR FROM c.date_timestamp) AS year
FROM consultation c;
```

2. Query

```
dim_client(VAT,gender,age)
VAT: FK(client)


CREATE VIEW dim_client AS
SELECT c.VAT                 as VAT,
       c.gender              as gender,
       YEAR(CURDATE()) - YEAR(c.birth_date) - IF
           (STR_TO_DATE(CONCAT(YEAR(CURDATE()),
               '-', MONTH(c.birth_date), '-',
               DAY(c.birth_date)), '%Y-%c-%e') >
            CURDATE(), 1, 0) AS age
FROM client c;
```

3. Query

```
dim_location_client(zip,city)
IC: zip corresponds to a zip code existing in clients


CREATE VIEW dim_location_client AS
    SELECT DISTINCT c.zip as zip,
                    c.city as city
    FROM client c;
```

4. Query

```
facts_consults(VAT,date,zip,num_procedures,num_medications,num_diagnostic_codes)
VAT: FK(dim_client)
date: FK(dim_date)
zip: FK(dim_location_client)
```

```
CREATE VIEW facts_consults AS
    SELECT c.VAT    as VAT_client,
           dat.day,
           dat.month,
           dat.year as date,
           loc.zip  as zip
    FROM dim_client c,
         dim_date dat,
         dim_location_client loc;
```