НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«Київський політехнічний інститут імені Ігоря Сікорського»Факультет інформатики та обчислювальної техніки Кафедра інформатики та програмної інженерії

Пояснювальна записка до Курсової роботи з дисципліни

«Бази даних»

«База даних лікувального закладу»

Прийняв:
Викладач кафедри ІПІ
Марченко О. І.
21 листопада 2021 року

Виконав
Студент групи ІТ-01
Бардін В. Д.

Київ – 2021

ЗАВДАННЯ

Завданням курсової роботи є розробка бази даних і її використання для вирішення практичних задач.

При розробці бази даних необхідно враховувати:

- вимоги до функціональності (наявність усіх функцій, які необхідні для реалізації поставленої задачі);
- вимоги до цілісності даних;
- вимоги до мінімізації об’єму даних, що зберігаються;
- наявність багатокористувальницького режиму;
- вимоги до швидкодії.

В процесі роботи над курсовою роботою повинні бути виконані наступні завдання:

- побудувати ER-модель, для чого необхідно:
  - детально проаналізувати предметне середовище;
  - сформулювати бізнес-правила, які будуть основою завдання обмежень при проектуванні та реалізації бази даних;
  - виявити необхідний набір сутностей;
  - визначити необхідний набір атрибутів для кожної сутності;
  - визначити зв'язки між об'єктами;
  - описати отриману ER-модель в одній з відомих нотацій;
  - розробити модель користувачів бази даних з описом їх прав;
- побудувати реляційну схему з ER-моделі, для чого необхідно:
  - побудувати набір необхідних відношень бази даних;
  - виділити первинні і зовнішні ключі у кожному з відношень;
  - привести отримані відношення до третьої нормальної формі;

    - визначити обмеження цілісності для спроектованих відношень;
- створити базу даних, що була спроектована, у форматі обраної системи управління базою даних (СУБД);
- створити користувачів бази даних, реалізувавши розроблену багатокористувальницьку модель;
- імпортувати дані з використанням засобів СУБД в створену базу даних;
- мовою SQL написати запити для визначених на етапі аналізу предметного середовища потреб користувачів;
- оптимізувати роботу запитів (продемонструвати роботу до і після оптимізації).

Обов’язковими вимогами є:

- кількість таблиць бази даних не менше 10;
- реалізація створених бізнес-правил;
- забезпечення цілісності бази даних;
- використання збережених процедур/функцій;
- використання тригерів;
- використання генераторів;
- використання представлень;
- створення не менше 20 DML-запитів типу SELECT (не включаючи insert, delete, update);
- кількість таблиць, атрибути яких використовуються у запиті не менше двох.

ЗМІСТ

1. ВСТУП

Будь-яка БД є частиною інформаційної системи і призначена для збору, зберігання і роботи з інформацією.

У сучасному світі жодна велика система не обходиться без бази даних. База даних являє собою спосіб управління інформації, що зберігається, і використовується у всіх сферах людського життя. Вона систематизує такі матеріали як інформацію про співробітників, про клієнтів, про постачальників, розрахунки, звіти і так далі, і обробляє цю інформацію за допомогою програми в комп'ютері.

Створюючи базу даних, ми прагнемо впорядкувати інформацію за різними ознаками і швидко отримувати її вибірку з поєднанням ознак за необхідності.

Проектування бази даних являє собою складний трудомісткий процес відображення предметної області. Процес проектування бази даних полягає в створенні схеми бази даних і визначення обмежень інформації.

База даних може зберігати величезну кількість інформації, і швидко видаватиїї запитом користувача. Ось чому бази даних актуальні в сучасному світі.

База даних - це організована структура, призначена для зберігання, зміни і обробки взаємозалежної інформації.

Реляційна база даних - це набір даних з умовленими зв'язками між ними. Цідані організовані у вигляді набору таблиць, що складаються із стовпців і рядків. Утаблицях зберігається інформація про об'єкти, представлених в базі даних. У кожному стовпчику таблиці зберігається певний тип даних, в кожному осередку - значення атрибута. Кожен рядок таблиці являє собою набір пов'язаних значень, щовідносяться до одного об'єкту або сутності. Кожен рядок в таблиці може бути позначений унікальним ідентифікатором, званим первинним ключем, а рядки з декількох таблиць можуть бути пов'язані з допомогою зовнішніх ключів.

СУБД - це програмна прошарок між користувачем і сервером бази даних.

Вона дозволяє абстрагувати користувача від системного бачення БД, а системі надає спосіб взаємодіяти з користувачем.

PostgreSQL - система керування базами даних. Основними завданнями цієї системи є:

- маніпуляція інформацією, яка знаходиться у базі даних,
- організація одночасного доступу до даних великої кількості користувачів
- PostgreSQL підтримує реляційну модель даних і створює об’єкти БД (таблиць, індексів, представлень), здійснює перевірку цілісності БД і відповідає за безпеку даних.

В даній курсовій роботі була розроблена реляційна база даних «База даних лікарняного закладу» за допомогою PostgreSQL, в якій необхідно було вирішити такі завдання:

- побудувати ER-модель, для чого необхідно:
  - детально проаналізувати предметне середовище;
  - сформулювати бізнес-правила, які будуть основою завдання обмежень при проектуванні та реалізації бази даних;
  - виявити необхідний набір сутностей;
  - визначити необхідний набір атрибутів для кожної сутності;
  - визначити зв'язки між об'єктами;
  - описати отриману ER-модель в одній з відомих нотацій;
  - розробити модель користувачів бази даних з описом їх прав;
- побудувати реляційну схему з ER-моделі, для чого необхідно:
  - побудувати набір необхідних відношень бази даних;
  - виділити первинні і зовнішні ключі у кожному з відношень;
  - привести отримані відношення до третьої нормальної формі;

    - визначити обмеження цілісності для спроектованихвідношень;
- створити базу даних, що була спроектована, у форматі обраної системи управління базою даних (СУБД);
- створити користувачів бази даних, реалізувавши розробленубагатокористувальницьку модель;
- імпортувати дані з використанням засобів СУБД в створену базуданих;
- мовою SQL написати запити для визначених на етапі аналізу предметного середовища потреб користувачів;
- оптимізувати роботу запитів (продемонструвати роботу до і післяоптимізації).

2. РОЗДІЛИ ОСНОВНОЇ ЧАСТИНИ РОБОТИ

2.1 Опис предметного середовища

2.1.1 Загальне поняття предметної області

Інформаційна система відповідає за забезпечення інформаційних процесів, забезпечення створення, поширення, використання, збереження і видалення інформації. Сама інформаційна база складається з однієї або декількох баз даних.

Предметна область - частина реального світу, що описує інформаційна система в залежності від її призначення.

В даний час усі організації повинні мати доступ до інформації. Це дуже важливо, тому бази даних, які задовольняють потреби зі зберіганням й управлінням даних, мають неабияку цінність, бо допомагають людям в цій справі.

При розробці реляційної бази даних «База даних лікарняного закладу» було проведено дослідження предметної області.

Отже, в є лікарняний заклад, який обслуговують людей. У інформаційній системі лікарняного закладу має зберігатися: список лікарів(співробітників), клієнтів та їх медичних карток, а також повна історія пацієнта. Також у БД має бути інформація по послуги, які надає заклад, а також інша допоміжна інформація.

2.1.2 Опис вхідних даних

При розробці реляційної БД були виділені наступні вхідні дані:

- Персонал(лікарі)
- Клієнти
- Послуги лікарського закладу
- Історія хвороби пацієнтів

Важливою умовою є те, що ці дані повинні зберігатися в БД, яка буде приведена до 3-ї нормальної форми.

2.1.3 Опис вихідних даних

Вихідні дані - повідомлення і результати, які видаються самою системою. Беруться з постійних даних. Опис запитів до БД буде наведено далі.

2.2 Постановка завдання

Реалізувати базу даних для лікарняного закладу, яка буде реалізовувати багатокористувацьку модель доступу до даних. БД має бути приведена до 3 нормальної форми.

2.3 Концептуальна модель бази даних

Як було зазначено вище було виділено 4 основні напрямки даних, які потрібно було зберігати. Персонал(лікарі), клієнти, послуги лікарського закладу, історія хвороби пацієнтів, але зберігати дані у такому вигляді не можна, бо тоді і мови йти не буде про 3 нормальну форму. Тож, для того, щоб привести дані до 3-НФ потрібно зробити додаткові, проміжні, таблиці. Допоміжні таблиці будуть представлені дані, у розділі про реалізацію БД.

На концептуальному рівні нам потрібно зберігати такі дані: ПІБ, ІПН, пол для пацієнтів та лікарів. Для лікарів також потрібно зберігати їх спеціалізацію, а також дату працевлаштування. Для історії хвороби потрібно: дата візиту, посилання на лікаря, до якого приходив пацієнт, а також опис того, що лікар сказав користувача. Для послуг закладу: її назву і вартість.

2.3.1 Наївна модель бази даних

– Користувачі та лікар

Номер атрибуту	**Назва атрибуту**	**Тип атрибуту**
1	Id	UUID
2	TIN	LONG
3	First Name	TEXT
4	Last Name	TEXT
5	Sex	PersonSex
6	Birth Date	DATE

Табл. 2.3.1.1 — Дані для користувачів та лікарів

– Лікарі

Номер атрибуту	**Назва атрибуту**	**Тип атрибуту**
7	Hiring Date	DATE
8	Specialization	SPECIALIZATION

Табл. 2.3.1.2 — Додаткові дані для лікарів

– Історія хвороби

Номер атрибуту	**Назва атрибуту**	**Тип атрибуту**
1	Id	UUID
2	Visit Date	DATE
3	Doctor	Doctor Id
4	Description	TEXT

Табл. 2.3.3 — Дані для історії хвороби

– Послуги закладу

Номер атрибуту	Назва атрибуту	Тип атрибуту
1	Id	UUID
2	Name	TEXT
3	Price	DECIMAL

Табл. 2.3.1.4 — Дані для послуг закладу

2.3.2 Деталізація бази даних. Встановлення відношень

Так як в курсовій роботу необхідно розробити реляційну модель потрібно встановити взаємо зв’язки між таблицями та деталізувати їх. Немає сенсу наводити повний перелік кроків, які робилися для нормалізації даних, тому наведу остаточний результат у вигляді ER-діаграми.

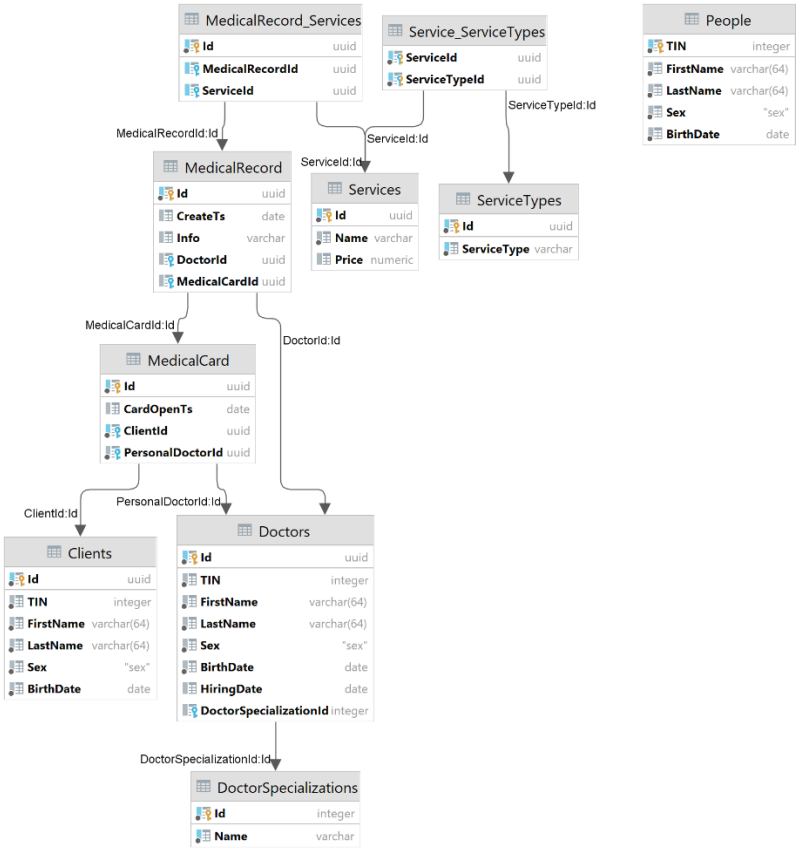


Рис. 2.3.2.1 — ER-діаграма спроектованої бази даних

2.4 Логічна модель бази даних

Тож, у цьому розділі наведено пояснення до ER-моделі наведеної в пункті 2.3.2.

Лікарі та їх спеціалізації

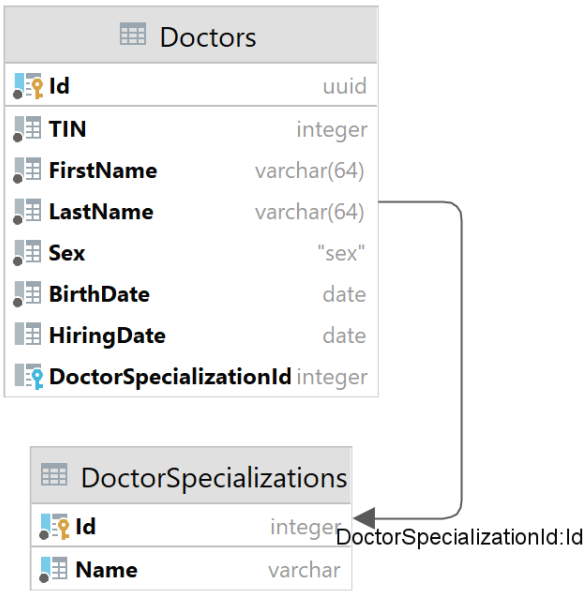


Рис. 2.4.1 — Реалізація відношення лікарів та спеціалізації

Так як спеціалізацій може бути дуже багато, і ми не можемо передбачити всі їх було вирішено створити додаткову таблицю, яка буде відповідати за збереження всіх наявних на даний момент спеціалізацій. Так як кожен лікар може мати лише одну спеціалізацію, жодних проміжних таблиць не створювалось, а зв’язок було встановлено за допомогою Foreign Key.

Сервіси закладу

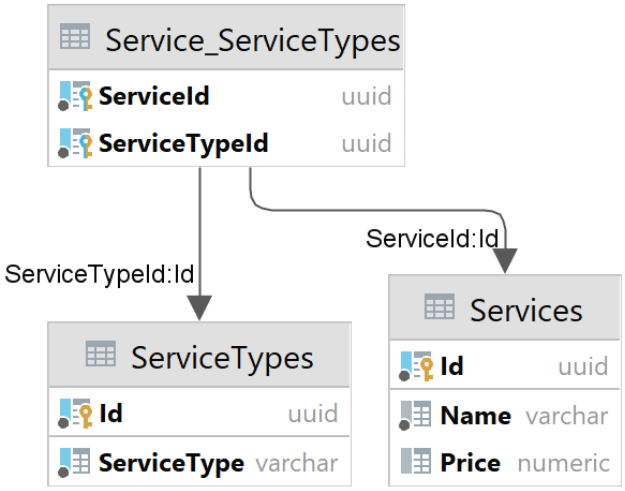


Рис 2.4.2 — Реалізація збереження сервісів закладу

Для збереження закладів використовується таблиця «Services», але так як у кожного сервісу є певний тип, а може бути і декілька було створено проміжну таблицю «Service_ServiceTypes», яка дозволяє створити зв’язок 1 до багатьох.

Історія хвороби

Історія хвороби є занадто складною сутністю, яка має багато посилань, так фактично є агрегатом. Тому її слід розглядати лише повністю. Для цього звернемося до рис. 2.3.2.1.

Коренем цієї сутності є медична картка, для їх збереження використовується табл. «MedicalCard», у цієї під сутності є посилання на клієнта, якому вона належить та на персонального(сімейного) лікаря.

Для збереження записів про відвідування клієнтом лікарів створено таблицю «MedicalRecord». Вона містить посилання на лікаря, якого

відвідував клієнт, а також на сервіси, що були йому надані.

2.5 Реалізація бази даних

2.5.1 Створення БД та користувачів

SQL скрипт

```
-- Create DB owner user, DB and grant him all privileges
CREATE USER "HospitalOwner" WITH PASSWORD 'hospital_owner';

CREATE DATABASE "HospitalDb" OWNER "HospitalOwner";
ALTER DATABASE "HospitalDb" SET TIMEZONE = 'UTC';

GRANT ALL PRIVILEGES ON DATABASE "HospitalDb" TO "HospitalOwner";

-- Create user with ReadOnly access to public schema
CREATE USER "HospitalReader" WITH PASSWORD 'hospital_reader';
GRANT CONNECT ON DATABASE "HospitalDb" TO "HospitalReader";
GRANT USAGE ON SCHEMA public TO "HospitalReader";
GRANT SELECT ON ALL TABLES IN SCHEMA public TO "HospitalReader";
```

Коментарі до скрипта

Для того, щоб забезпечити безпечний доступ до БД буду створено додаткового користувача, який має можливість лише читати дані. До того ж такий підхід корисний при використані CQRS архітектури для додатку, бо дозволяє використовувати read-only підключення до БД, коли немає необхідності писати щось в БД.

2.5.2 Створення таблиць

```
-- Create types

CREATE TYPE "Sex" AS ENUM ('Male', 'Female', 'X');

-- Create tables

-- Common/Base tables
CREATE TABLE IF NOT EXISTS "People"
(
    "TIN"       INTEGER PRIMARY KEY,
    "FirstName" VARCHAR(64)                          NOT NULL,
    "LastName"  VARCHAR(64)                          NOT NULL,
    "Sex"       "Sex"                                NOT NULL,
    "BirthDate" DATE CHECK ( "BirthDate" <= NOW() ) NOT NULL
);

CREATE UNIQUE INDEX "People_UIndex" ON "People" (
    "TIN" ASC
    );
```

```
-- Doctors tables

CREATE TABLE IF NOT EXISTS "DoctorSpecializations"
(
    "Id"   INTEGER PRIMARY KEY,
    "Name" VARCHAR NOT NULL UNIQUE
);

CREATE UNIQUE INDEX "DoctorSpecializations_UIndex" ON "DoctorSpecializations"
(
    "Id" ASC
    );

CREATE TABLE IF NOT EXISTS "Doctors"
(
    "Id"                     UUID PRIMARY KEY,
    "HiringDate"             DATE CHECK ( "HiringDate" <= now() ),
    "DoctorSpecializationId" INTEGER,
    CONSTRAINT "Doctors_DoctorsSpecializations__fk"
        FOREIGN KEY ("DoctorSpecializationId")
            REFERENCES "DoctorSpecializations" ("Id")
) INHERITS ("People");

CREATE UNIQUE INDEX "Doctors_Uindex" ON "Doctors" (
    "Id" ASC
    );

CREATE TABLE IF NOT EXISTS "ServiceTypes"
(
    "Id"          UUID PRIMARY KEY,
    "ServiceType" VARCHAR NOT NULL
);

CREATE UNIQUE INDEX "ServiceTypes_Uindex" ON "ServiceTypes" (
    "Id" ASC,
    "ServiceType" ASC
    );


CREATE TABLE IF NOT EXISTS "Services"
(
    "Id"    UUID PRIMARY KEY,
    "Name"  VARCHAR NOT NULL,
    "Price" DECIMAL CHECK ( "Price" >= 0 )
);

CREATE UNIQUE INDEX "Services_Uindex" ON "Services" (
    "Id" ASC
    );
```

```
CREATE TABLE IF NOT EXISTS "Service_ServiceTypes"
(
    "ServiceId"     UUID,
    "ServiceTypeId" UUID,
    CONSTRAINT "Service_ServiceTypes_pk"
        PRIMARY KEY ("ServiceId", "ServiceTypeId"),
    CONSTRAINT "Service_ServiceTypes_Service__fk"
        FOREIGN KEY ("ServiceId") REFERENCES "Services" ("Id"),
    CONSTRAINT "Service_ServiceTypes_ServiceType__fk"
        FOREIGN KEY ("ServiceTypeId") REFERENCES "ServiceTypes" ("Id")
);

-- Clients tables

CREATE TABLE IF NOT EXISTS "Clients"
(
    "Id" UUID PRIMARY KEY
) INHERITS ("People");

CREATE UNIQUE INDEX "Customers_Uindex" ON "Clients" (
    "Id" ASC
    );

-- General tables

CREATE TABLE IF NOT EXISTS "MedicalCard"
(
    "Id"               UUID PRIMARY KEY,
    "CardOpenTs"       DATE DEFAULT NOW(),
    "ClientId"         UUID UNIQUE NOT NULL,
    "PersonalDoctorId" UUID        NOT NULL,
    CONSTRAINT "MedicalCard_Clients__fk" FOREIGN KEY ("ClientId") REFERENCES
"Clients" ("Id"),
    CONSTRAINT "MedicalCard_Doctors__fk" FOREIGN KEY ("PersonalDoctorId")
REFERENCES "Doctors" ("Id")
);

CREATE UNIQUE INDEX "MedicalCard_Uindex" ON "MedicalCard" (
                                                           "Id" ASC
    );

CREATE TABLE IF NOT EXISTS "MedicalRecord"
(
    "Id"            UUID PRIMARY KEY,
    "CreateTs"      DATE DEFAULT NOW(),
    "Info"          VARCHAR,
    "DoctorId"      UUID,
    "MedicalCardId" UUID,
    "ServiceId"     UUID,
    CONSTRAINT "MedicalRecord_Doctors__fk"
        FOREIGN KEY ("DoctorId") REFERENCES "Doctors" ("Id"),
    CONSTRAINT "MedicalRecord_MedicalCard__fk"
        FOREIGN KEY ("MedicalCardId") REFERENCES "MedicalCard" ("Id"),
    CONSTRAINT "MedicalRecord_Service__fk"
        FOREIGN KEY ("ServiceId") REFERENCES "Services" ("Id")
);
```

```
CREATE UNIQUE INDEX “MedicalRecord_Uindex” ON “MedicalRecord” (
    “Id” ASC
    );

CREATE TABLE IF NOT EXISTS “MedicalRecord_Services”
(
    “Id”              UUID PRIMARY KEY,
    “MedicalRecordId” UUID,
    “ServiceId”       UUID,
    CONSTRAINT “MedicalRecord_Services_MedicalRecord__fk”
        FOREIGN KEY (“MedicalRecordId”) REFERENCES “MedicalRecord” (“Id”),
    CONSTRAINT “MedicalRecord_Services_Services__fk”
        FOREIGN KEY (“ServiceId”) REFERENCES “Services” (“Id”)
);

CREATE INDEX “MedicalRecord_Services_Index” ON “MedicalRecord_Services” (
    “Id” ASC,
    “ServiceId” ASC,
    “MedicalRecordId” ASC
    );
```

2.5.3 Заповнення таблиць тестовими даними

Скрипти заповнення наводити в пояснювальній записці немає сенсу - вони однотипні і займають дуже багато місця. Ознайомитись з ними можна тут: https://github.com/Bardin08/KPI-Third-Term/tree/master/DB/cw/Dateseeds/.

2.5.4 Інші об’єкти бази даних

```
CREATE OR REPLACE FUNCTION get_most_popular_service_at_timespan(
    start_date DATE, end_date DATE)
    RETURNS RECORD
AS
$$
DECLARE
    result "Services";
BEGIN
    SELECT s."Id", count(s."Id")
    INTO result
    FROM "Services" AS S
        JOIN "MedicalRecord_Services" MRS on S."Id" = MRS."ServiceId"
        JOIN "MedicalRecord" MR on MRS."MedicalRecordId" = MR."Id"
    WHERE MR."CreateTs" >= start_date AND MR."CreateTs" <= end_date
    GROUP BY S."Id"
    ORDER BY count(s."Id") DESC
    LIMIT 1;

    RETURN result;
END;
$$
    LANGUAGE plpgsql;

SELECT get_most_popular_service_at_timespan('12.01.2020', '12.08.2021');
```

```sql
CREATE OR REPLACE FUNCTION get_most_popular_service_type_at_timespan(
    start_date DATE, end_date DATE)
    RETURNS RECORD
AS
$$
DECLARE
    result "ServiceTypes";
BEGIN
    SELECT ST."Id", ST."ServiceType"
    INTO result
    FROM "Services" AS S
        JOIN "MedicalRecord_Services" MRS on S."Id" = MRS."ServiceId"
        JOIN "MedicalRecord" MR on MRS."MedicalRecordId" = MR."Id"
        JOIN "Service_ServiceTypes" SST on S."Id" = SST."ServiceId"
        JOIN "ServiceTypes" ST on SST."ServiceTypeId" = ST."Id"
    WHERE MR."CreateTs" >= start_date AND MR."CreateTs" <= end_date
    GROUP BY ST."Id", ST."ServiceType"
    ORDER BY count(ST."ServiceType") DESC
    LIMIT 1;

    RETURN result;
END;
$$
    LANGUAGE plpgsql;

SELECT get_most_popular_service_type_at_timespan('12.01.2020', '12.08.2021');

CREATE OR REPLACE FUNCTION get_profit_by_client(client_id UUID)
    RETURNS DECIMAL
AS
$$
DECLARE
    profit DECIMAL;
BEGIN
    CREATE TEMP TABLE IF NOT EXISTS "Client's Services" AS
    SELECT S."Price"
    FROM "MedicalCard" AS MC
             JOIN "MedicalRecord" MR on MC."Id" = MR."MedicalCardId"
             JOIN "MedicalRecord_Services" MRS on MR."Id" =
MRS."MedicalRecordId"
             JOIN "Services" S on MRS."ServiceId" = S."Id"
    WHERE "ClientId" = client_id;

    SELECT sum(CS."Price")
    INTO profit
    FROM "Client's Services" AS CS;

    DROP TABLE "Client's Services" CASCADE;
    RETURN profit;
END;
$$ LANGUAGE plpgsql;

SELECT get_profit_by_client('63d066b9-a92c-466f-a2fd-1cc6fd48f0ce');
```

```sql
CREATE OR REPLACE FUNCTION get_profit_by_doctor(doctor_id UUID)
    RETURNS DECIMAL
AS
$$
DECLARE
    profit DECIMAL;
BEGIN
    CREATE TEMP TABLE IF NOT EXISTS "Doctor's Services" AS
    SELECT S."Price"
    FROM "MedicalRecord" MR
             JOIN "MedicalRecord_Services" MRS on MR."Id" =
MRS."MedicalRecordId"
             JOIN "Services" S on MRS."ServiceId" = S."Id"
    WHERE MR."DoctorId" = doctor_id;

    SELECT sum(CS."Price")
    INTO profit
    FROM "Doctor's Services" AS CS;

    DROP TABLE "Doctor's Services" CASCADE;
    RETURN profit;
END;
$$ LANGUAGE plpgsql;

SELECT get_profit_by_doctor('37abe20e-fb67-448d-a929-11048de547c7');

CREATE OR REPLACE FUNCTION get_doctors_by_specialization(specialization_id
INTEGER)
    RETURNS TABLE
            (
                "DoctorId" UUID
            )
AS
$$
DECLARE
BEGIN
    RETURN QUERY SELECT D."Id"
    FROM "DoctorSpecializations" AS SP
        JOIN "Doctors" D on SP."Id" = D."DoctorSpecializationId"
    WHERE SP."Id" = specialization_id;
END;
$$ LANGUAGE plpgsql;

SELECT get_doctors_by_specialization(1);
```

```sql
CREATE OR REPLACE FUNCTION
get_doctors_with_experience_greater_than_years(years_exp INTEGER)
    RETURNS SETOF RECORD
AS
$$
SELECT D."Id", SP."Name"
FROM "DoctorSpecializations" AS SP
         JOIN "Doctors" D on SP."Id" = D."DoctorSpecializationId"
WHERE extract(YEAR FROM age(D."HiringDate")) >= years_exp;
$$ LANGUAGE sql;

SELECT get_doctors_with_experience_greater_than_years(0);

CREATE OR REPLACE FUNCTION get_all_doctors_who_worked_with_client(client_id
UUID)
    RETURNS SETOF RECORD
AS
$$
SELECT D."Id", D."FirstName", D."LastName"
FROM "Doctors" AS D
         JOIN "MedicalRecord" MR on D."Id" = MR."DoctorId"
         JOIN "MedicalCard" MC on MC."Id" = MR."MedicalCardId"
         JOIN "Clients" C on MC."ClientId" = C."Id"
WHERE C."Id" = client_id;
$$ LANGUAGE sql;

SELECT get_all_doctors_who_worked_with_client('fd45d03b-c870-4e32-a70a-
44df891f4113');

CREATE OR REPLACE FUNCTION sp_get_all_receptions_during_timespan(
    start_date DATE, end_date date)
    RETURNS SETOF RECORD
AS
$$
SELECT C."Id", C."FirstName", C."LastName", MR."Info"
FROM "MedicalRecord" AS MR
    JOIN "MedicalCard" MC on MR."MedicalCardId" = MC."Id"
    JOIN "Clients" C on MC."ClientId" = C."Id"
WHERE MR."CreateTs" >= start_date AND MR."CreateTs" <= end_date;
$$ LANGUAGE sql;

SELECT sp_get_all_receptions_during_timespan('12.01.2021', '12.08.2021');

CREATE OR REPLACE FUNCTION get_doctors_personal_patients(doctor_id UUID)
    RETURNS SETOF RECORD
AS
$$
SELECT C.*
FROM "Clients" AS C
         JOIN "MedicalCard" MC on C."Id" = MC."ClientId"
         JOIN "Doctors" D on D."Id" = MC."PersonalDoctorId"
WHERE D."Id" = doctor_id;
$$
    LANGUAGE sql;

SELECT get_doctors_personal_patients('491d8628-3f96-4e98-874e-754cf4526713');
```

```
CREATE OR REPLACE PROCEDURE sp_doctor_patients_at_timespan(
    doctor_id UUID,
    start_date DATE,
    end_date DATE)
AS
$$
    CREATE TEMP TABLE IF NOT EXISTS "DocsPatientsTempTable" AS
    SELECT concat(C."FirstName", ' ', C."LastName") AS "Client name"
    FROM "MedicalRecord" MR
             JOIN "MedicalCard" MC ON MR."MedicalCardId" = MC."Id"
             JOIN "Clients" C ON MC."ClientId" = C."Id"
    WHERE MR."CreateTs" >= start_date
      AND MR."CreateTs" <= end_date
      AND MR."DoctorId" = doctor_id;
    $$ LANGUAGE sql;


CREATE OR REPLACE PROCEDURE sp_create_view_for_doctors_patients(
    doctor_id UUID,
    start_date DATE,
    end_date DATE)
AS
$$
BEGIN
    CALL sp_doctor_patients_at_timespan(
            doctor_id,
            start_date,
            end_date);

    CREATE OR REPLACE VIEW "Doctor's () Patients for 01/12/2020 to
12/08/2021" AS
    SELECT *
    FROM "DocsPatientsTempTable";
END;
$$
    LANGUAGE plpgsql;

CALL sp_create_view_for_doctors_patients('08713628-1646-490b-98cd-
0c96a4b7cbf4', '12.01.2020', '12.08.2021');

SELECT *
FROM "Doctor's () Patients for 01/12/2020 to 12/08/2021";

DROP VIEW IF EXISTS "Doctor's () Patients for 01/12/2020 to 12/08/2021";
DROP TABLE IF EXISTS "DocsPatientsTempTable";

CREATE OR REPLACE PROCEDURE sp_illness_history(client_id UUID)
AS
$$
BEGIN
    CREATE TEMP TABLE IF NOT EXISTS "ClientIllnesses" AS
    SELECT MR.*
    FROM "MedicalCard" AS MC
        JOIN "MedicalRecord" MR on MC."Id" = MR."MedicalCardId"
    WHERE MC."ClientId" = client_id;

    CREATE OR REPLACE VIEW "Client's illnesses history" AS
```

```
    SELECT * FROM "ClientIllnesses";

END;
$$
    LANGUAGE plpgsql;

CALL sp_illness_history('789ea97b-a0e1-4e17-b1ef-d11bf82ef1c3');

SELECT *
FROM "Client's illnesses history";
DROP TABLE IF EXISTS "ClientIllnesses" CASCADE;

CREATE OR REPLACE PROCEDURE sp_get_reception_invoice(med_record_id UUID)
AS
$$
BEGIN
    CREATE TEMP TABLE IF NOT EXISTS "Invoice" AS
    SELECT MR."Id"         AS "Visit Id",
           SUM(S2."Price") AS "Visit Price"
    FROM "MedicalRecord" AS MR
             JOIN "MedicalRecord_Services" MRS on MR."Id" =
MRS."MedicalRecordId"
             JOIN "Services" S2 on MRS."ServiceId" = S2."Id"
    WHERE MR."Id" = med_record_id
    GROUP BY MR."Id";

    CREATE OR REPLACE VIEW "Visit total Price" AS
    SELECT * FROM "Invoice";
END;
$$ LANGUAGE plpgsql;

CALL sp_get_reception_invoice('d1c04734-05cb-4f01-ba99-c018bac7e99c');
SELECT *
FROM "Visit total Price";
DROP TABLE IF EXISTS "Invoice" CASCADE;
```

Коментарі

Всі вибори даних винесено або в збережені процедури, або в функцій для більш зручного їх використання. А також для того, щоб у разі необхідності використання ORM або Micro ORM можна було отримати максимальну швидкість взаємодії з БД за рахунок використання об’єктів бази даних, що зберігаються на сервері СУБД та не потребують передачі коду запиту на сервер. Детальніший аналіз усіх наведених об’єктів доступний за посиланням https://github.com/Bardin08/KPI-Third-Term/tree/master/DB/cw/docs/db-objects.

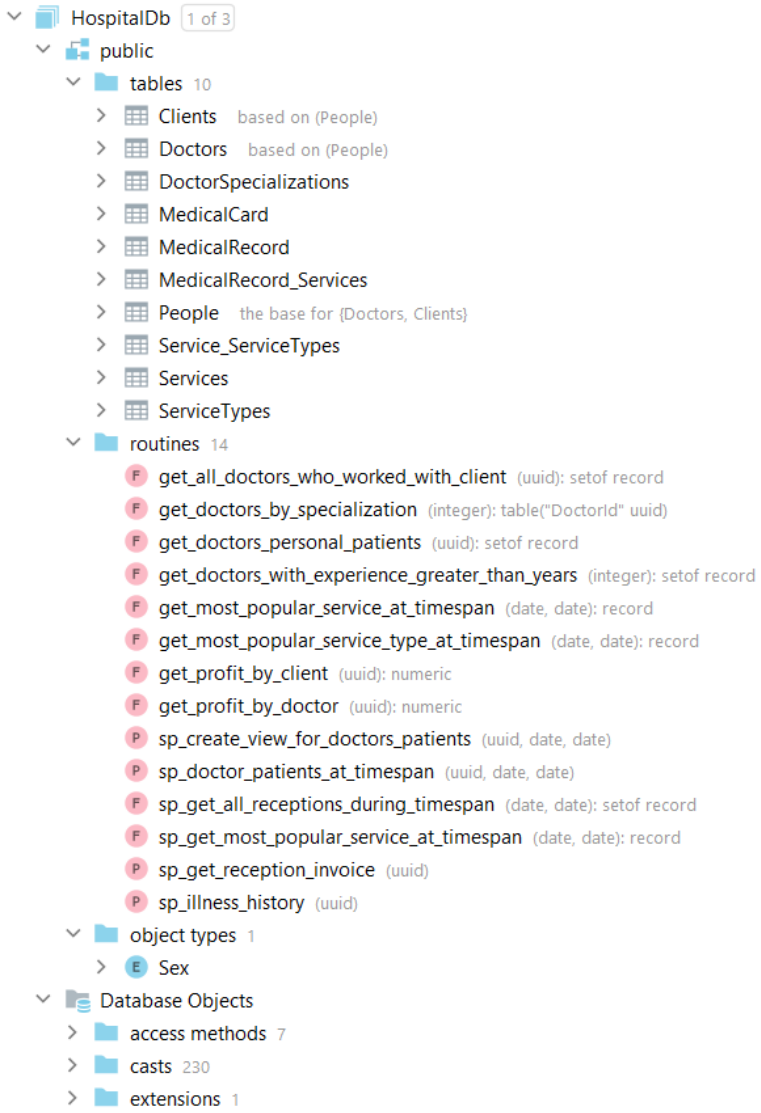


Рис. 2.5.4.1 — Схема БД після виконання всіх скриптів побудови БД

3 ВИСНОВКИ

Метою курсової роботи було проектування бази даних лікарського закладу. Для виконання курсової роботи були проведені всі необхідні дослідження, , що стосуються розробки стратегії автоматизації та оптимізації обробки даних у системі закладу, в результаті яких була отримана відповідь на запитання, щодо збереження та взаємодії з даними. Спочатку ми провели повний аналіз нашої предметної області і тоді вже почали її описувати

Після цього була побудована концептуальна модель за допомогою мови ER-опису предметної області, яка базується на концепції, що інформаційна модель предметної області може бути описана із застосування таких понять, як сутність, атрибут, зв’язок. Після побудови ER діаграми, було проведено нормалізацію таблиць бази даних. Створено даталогічну модель бази даних. Потім було створено саму базу даних, її таблиці, та інші об’єкти. Далі було заповнено таблиці бази даних, створнено збережені процедури, тригери, представлення для створення вибірок інформації.

В даній курсовій роботі була розроблена база даних «Ювелірна майстерня» в системі управління базами даних Microsoft SQL Server 18.

Створення бази даних «Ювелірна майстерня» є досить актуальним і корисним, бо вона полегшує життя і управління майстернею і слідкування за всіма замовленнями.

4 СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Офіційний сайт PostgreSQL, розділ документації — https://www.postgresql.org/docs/

5 ДОДАТКИ

Додаток А — Результати виконання запитів для вибору даних

sp_create_view_for_doctors_patients

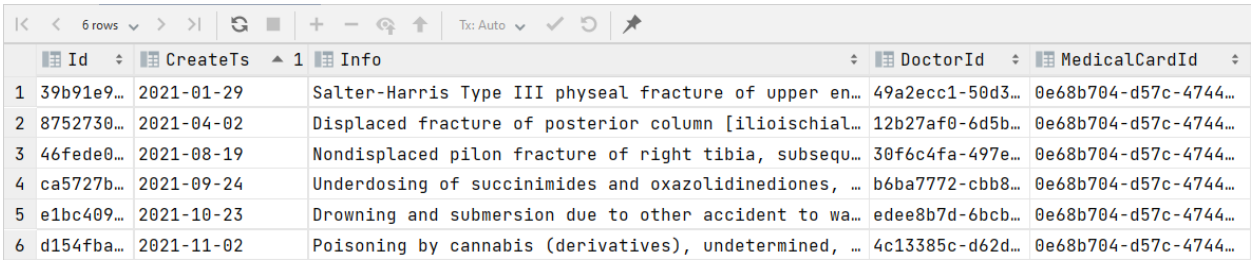
17 rows

	Client name
1	Mendel Winstanley
2	Fanchon Najafian
3	Caresse Kenewell
4	Wilburt Biggerstaff
5	Ariela Casely
6	Tanny Dabnor
7	Gillian Zmitruk
8	Willetta Drezzer
9	Deloria Kilalea
10	Mano Klarzynski
11	Gerty Johnsey
12	Beck Muckleston
13	Allissa Wooderson
14	Oswald Hambribe
15	Conney Emblem
16	Edeline Threlkeld
17	Gabriello Scruton

Рис. 5.А.1 — Результат виконання *sp_create_view_for_doctors_patients*

```
CALL sp_create_view_for_doctors_patients('08713628-1646-490b-98cd-
0c96a4b7cbf4', '12.01.2020', '12.08.2021');
SELECT *
FROM "Doctor's () Patients for 01/12/2020 to 12/08/2021";
```

sp_illness_history

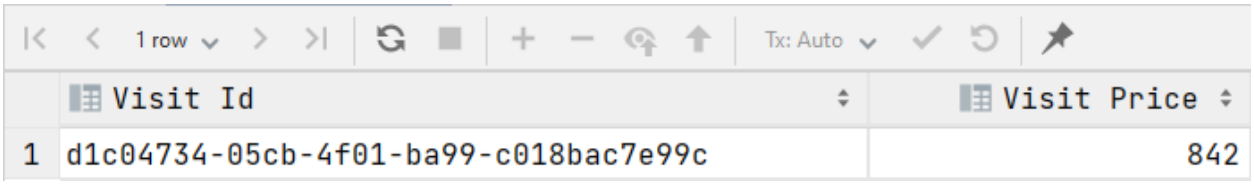


	Id	CreateTs	Info	DoctorId	MedicalCardId
1	39b91e9…	2021-01-29	Salter-Harris Type III physeal fracture of upper en…	49a2ecc1-50d3…	0e68b704-d57c-4744…
2	8752730…	2021-04-02	Displaced fracture of posterior column [ilioischial…	12b27af0-6d5b…	0e68b704-d57c-4744…
3	46fede0…	2021-08-19	Nondisplaced pilon fracture of right tibia, subsequ…	30f6c4fa-497e…	0e68b704-d57c-4744…
4	ca5727b…	2021-09-24	Underdosing of succinimides and oxazolidinediones, …	b6ba7772-cbb8…	0e68b704-d57c-4744…
5	e1bc409…	2021-10-23	Drowning and submersion due to other accident to wa…	edee8b7d-6bcb…	0e68b704-d57c-4744…
6	d154fba…	2021-11-02	Poisoning by cannabis (derivatives), undetermined, …	4c13385c-d62d…	0e68b704-d57c-4744…

Рис. 5.A.2 — Результат виконання *sp_illness_history*

```
CALL sp_illness_history('789ea97b-a0e1-4e17-b1ef-d11bf82ef1c3');
SELECT *
FROM "Client's illnesses history";
```

sp_get_reception_invoice

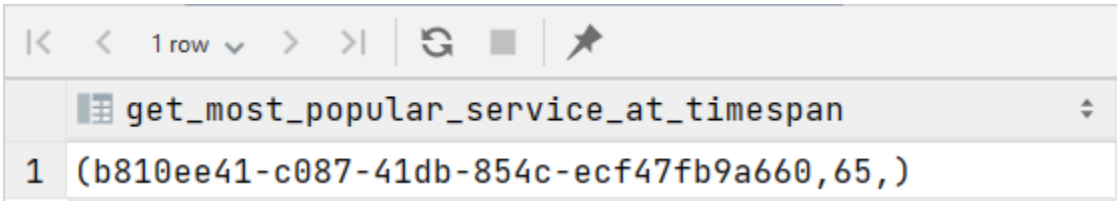


	Visit Id	Visit Price
1	d1c04734-05cb-4f01-ba99-c018bac7e99c	842

Рис. 5.A.3 — Результат виконання *sp_get_reception_invoice*

```
CALL sp_get_reception_invoice('d1c04734-05cb-4f01-ba99-c018bac7e99c');
SELECT *
FROM "Visit total Price";
```

get_most_popular_service_at_timespan

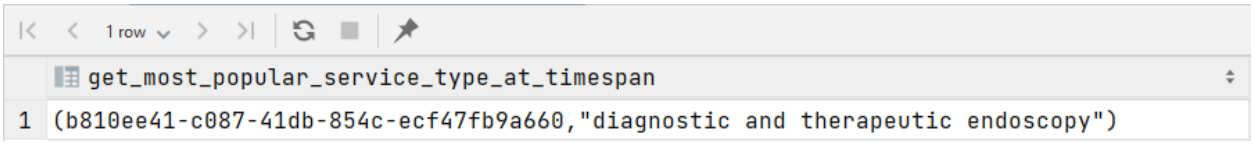


	get_most_popular_service_at_timespan
1	(b810ee41-c087-41db-854c-ecf47fb9a660,65,)

Рис. 5.A.4 — Результат виконання *get_most_popular_service_at_timespan*

```
SELECT get_most_popular_service_at_timespan('12.01.2020', '12.08.2021');
```

get_most_popular_service_type_at_timespan



	get_most_popular_service_type_at_timespan
1	(b810ee41-c087-41db-854c-ecf47fb9a660,"diagnostic and therapeutic endoscopy")

Рис. 5.A.5 — Результат виконання *get_most_popular_service_type_at_timespan*

```
SELECT get_most_popular_service_type_at_timespan('12.01.2020', '12.08.2021');
```

get_profit_by_client

	get_profit_by_client
1	7367

Рис. 5.A.6 — Результат виконання *get_profit_by_client*

```
SELECT get_profit_by_client('63d066b9-a92c-466f-a2fd-1cc6fd48f0ce');
```

get_profit_by_doctor

	get_profit_by_doctor
1	18187

Рис. 5.A.7 — Результат виконання *get_profit_by_doctor*

```
SELECT get_profit_by_doctor('37abe20e-fb67-448d-a929-11048de547c7');
```

get_doctors_by_specialization

	get_doctors_by_specialization
1	6622b791-9643-493d-9954-8bcc3c992a79
2	a8d3bd43-6863-4973-a647-ccacb3e19bec

Рис. 5.A.7 — Результат виконання *get_doctors_by_specialization*

```
SELECT get_doctors_by_specialization(1);
```

get_doctors_with_experience_greater_than_years

	get_doctors_with_experience_greater_than_years
1	(a8d3bd43-6863-4973-a647-ccacb3e19bec,"Allergy and immunology")
2	(6622b791-9643-493d-9954-8bcc3c992a79,"Allergy and immunology")
3	(9f3e9df8-dbda-4b0f-af7c-162c6d7314a2,Anesthesiology)
4	(9005e546-2859-4f31-8633-083a6b44604b,Anesthesiology)
5	(7d8e6be3-89c6-4c5b-a0bf-2a6b95818ea4,Dermatology)
6	(12b27af0-6d5b-4c66-8fd9-90d6188bcc12,Dermatology)
7	(30f6c4fa-497e-49f1-bb96-a26bc96c8d84,"Diagnostic radiology")
8	(37abe20e-fb67-448d-a929-11048de547c7,"Diagnostic radiology")
9	(f24e0a25-0a54-4ea3-9b07-1acac4336d88,"Emergency medicine")
10	(ce548058-24ae-4ecf-9a68-e8098b017530,"Emergency medicine")
11	(99e1ab3d-09eb-4aac-bee5-4ebce9aee4e6,"Family medicine")
12	(4c13385c-d62d-4926-bce3-8eb2e5c617f7,"Family medicine")
13	(b6ba7772-cbb8-401b-9ed8-b49e91433336,"Internal medicine")
14	(9c513789-ddff-4068-92ba-cdab13538cc7,"Internal medicine")
15	(eaba3a85-d83b-4b24-9da1-4d6e0f41c09c,"Medical genetics")
16	(b05388c3-d60e-4c44-9fd0-878406b3c44b,"Medical genetics")
17	(edee8b7d-6bcb-4781-8ba7-61d6e4efc45d,Neurology)
18	(49a2ecc1-50d3-4b42-a0fd-84945fc815cd,Neurology)

Рис. 5.A.8 — Результат виконання *get_doctors_with_experience_greater_than_years*

```
SELECT get_doctors_with_experience_greater_than_years(0);
```

get_all_doctors_who_worked_with_client

	get_all_doctors_who_worked_with_client
1	(f405aad8-97a9-4dad-a458-64442f8b8abd,Bobbye,Kornyakov)
2	(08713628-1646-490b-98cd-0c96a4b7cbf4,Cordelia,Medina)
3	(9f3e9df8-dbda-4b0f-af7c-162c6d7314a2,Zena,Farnan)
4	(f24e0a25-0a54-4ea3-9b07-1acac4336d88,Nichols,Prandoni)
5	(eaba3a85-d83b-4b24-9da1-4d6e0f41c09c,Barnabe,"Van Der Hoog")

Рис. 5.A.9 — Результат виконання *get_all_doctors_who_worked_with_client*

```
SELECT get_all_doctors_who_worked_with_client('fd45d03b-c870-4e32-a70a-
44df891f4113');
```

sp_get_all_receptions_during_timespan

	sp_get_all_receptions_during_timespan
1	(f638aca7-5fda-4fb5-8022-5dd6df8af5d8,Wilburt,Bigge…
2	(51f43995-43af-4791-91cd-d67893d66a88,Danie,Downage…
3	(a4f2140f-3fd9-4f37-bb98-e74b8104f8ca,Deloria,Kilal…
4	(8acb3079-9b46-4795-ab43-c56af24091df,Friedrick,Bag…
5	(f615e578-f917-431a-87de-30eb76e558c6,Gale,Curnock,…
6	(7e925a3b-e17e-48d3-8256-76a4e3d7f338,Derwin,Echali…
7	(33b974d5-eb64-44a9-8ec8-dc16f87dd8f7,Thaine,O'Dyvo…

Рис. 5.A.10 — Результат виконання *sp_get_all_receptions_during_timespan*

```
SELECT sp_get_all_receptions_during_timespan('12.01.2021', '12.08.2021');
```

get_doctors_personal_patients

	get_doctors_personal_patients
1	(8,Willetta,Drezzer,Female,1985-06-03,5348418b-7ef3-4488-bbb8-ad01b9cd8792)
2	(8578,Violante,Maycock,Male,2010-10-14,9831f4ba-0c0a-4216-b1b0-315f8d3a59cf)
3	(646,Thaine,O'Dyvoie,Female,1996-01-27,33b974d5-eb64-44a9-8ec8-dc16f87dd8f7)

Рис. 5.A.10 — Результат виконання *get_doctors_personal_patients*

```
SELECT get_doctors_personal_patients('491d8628-3f96-4e98-874e-754cf4526713');
```