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

Звіт до лабораторної роботи №2 з дисципліни

«Бази даних»

Прийняв:
Викладач кафедри ІПІ
Марченко О. І.
24 жовтня 2021 року

Виконав
Студент групи ІТ-01
Бардін В. Д.

Київ – 2021

Лабораторна робота №2

Тема: Створення бази даних

Мета:

- Створення бази даних шляхом визначення схеми БД та заповнення її тестовими даними;
- Навчитися проектувати бази даних, вводити і редагувати структуру таблиць та дані в таблицях;
- Вивчити команди SQL для роботи з таблицями (створення, зміни та видалення таблиць);
- Вивчити використовувані в SQL засоби для підтримки цілісності даних та їх практичне застосування

Завдання: Програмне забезпечення «Діяльність фірми з розробки програмних продуктів». Підприємства, о розробляють ПЗ, зазвичай мають декілька відділів, а саме: дирекція, бухгалтерія, маркетинговий відділ, відділ розробки ПЗ, відділ тестування ПЗ, відділ супроводження тощо. ПЗ, котре поставляється Замовнику, має назву, список розробників (внутрішній список тестувальників, котрий Замовнику не надається), вартість, документацію, дистрибутив, правила використання. Замовниками можуть бути як фізичні так і юридичні особи. Кожний Замовник має можливість замовити декілька ПП, на кожний з яких він отримує ліцензію, в якій вказано назву продукту, дату продажу, вартість, терміни апгрейдів.

Схема спроектованої бази даних

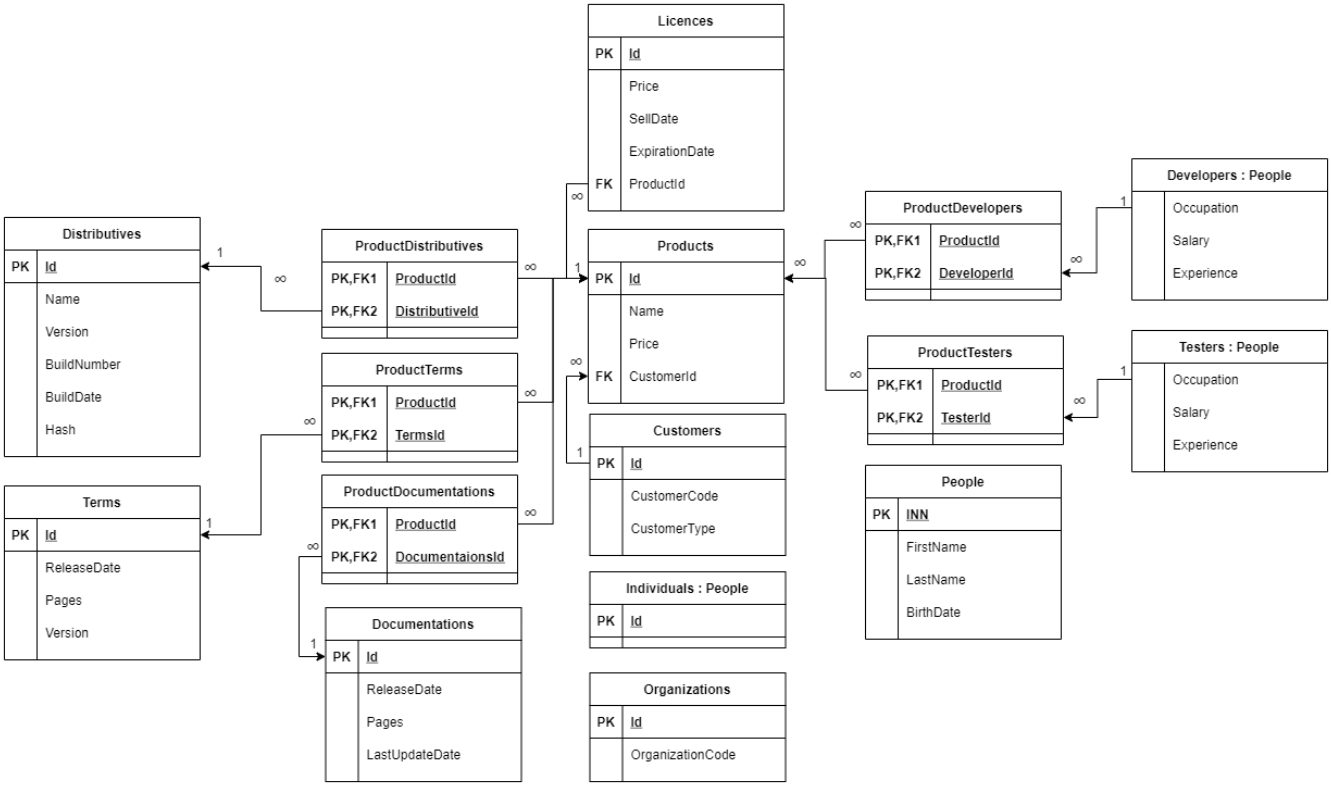


Рис. 1 — Схема БД

Після 1-ї лабораторної роботи схема зазнала змін, що дало можливість спростити її та не реалізовувати багато “зайвих” зв’язків. Зміни:

1. видалено FK-посилання на найновіші версії дистрибутиву, документації та умов використання.

Ця зміна дуже суттєво впливає на операцію оновлення, бо при випуску нової версії потрібно буде оновити менше записів. На пошук це не вплине майже ніяк, бо в нас є таблиця з усіма дистрибутивами і в ній ми завжди зможемо знайти останній відсортувавши їх за датою.

Створення бази даних за допомогою SQL

```sql
-- Create a user and a database.
-- Grant him a full access to the database.

CREATE USER SC_owner WITH PASSWORD 'SC_owner';

CREATE DATABASE "SCompany-Main-DB" owner SC_owner;
ALTER DATABASE "SCompany-Main-DB" SET TIMEZONE = 'UTC';

GRANT ALL PRIVILEGES ON DATABASE "SCompany-Main-DB" TO SC_owner;
```

Наведений SQL код створить користувача `SC_owner`, створить БД `SCompany-Main-DB` і наддасть користувачу `SC_owner` до неї доступ.

Створення таблиць, індексів, тригерів та функцій

```
-- Create custom types

CREATE TYPE "CustomerType" AS ENUM ('Individual', 'Organization');


-- Create tables and set all the required constraints.

CREATE TABLE IF NOT EXISTS "People"
(
    "INN"       INTEGER,
    "FirstName" VARCHAR(64),
    "LastName"  VARCHAR(64),
    "BirthDate" DATE CHECK ( "BirthDate" <= now() ) DEFAULT now(),
    CONSTRAINT people_pk PRIMARY KEY ("INN")
);

CREATE TABLE IF NOT EXISTS "Developers"
(
    "Occupation" VARCHAR(64),
    "Salary"     DECIMAL,
    "Experience" DATERANGE NOT NULL,
    CONSTRAINT developers_pk PRIMARY KEY ("INN")
) INHERITS ("People");

CREATE TABLE IF NOT EXISTS "Testers"
(
    "Occupation" VARCHAR(64),
    "Salary"     DECIMAL,
    "Experience" DATERANGE NOT NULL,
    CONSTRAINT testers_pk PRIMARY KEY ("INN")
) INHERITS ("People");

CREATE TABLE IF NOT EXISTS "Individuals"
(
    "Id" UUID,
    CONSTRAINT id_pk PRIMARY KEY ("Id")
) INHERITS ("People");

CREATE TABLE IF NOT EXISTS "Organizations"
(
    "Id"               UUID,
    "OrganizationCode" INTEGER,
    "CompanyName"      VARCHAR,
    CONSTRAINT organizations_pk PRIMARY KEY ("Id"),
    UNIQUE ("OrganizationCode")
);

CREATE TABLE IF NOT EXISTS "Customers"
(
    "Id"           UUID,
    "CustomerType" "CustomerType",
    "CustomerCode" INTEGER,
    CONSTRAINT customers_pk PRIMARY KEY ("Id")
```

```
);

CREATE TABLE IF NOT EXISTS "Documentations"
(
    "Id"             UUID,
    "Pages"          INTEGER,
    "ReleaseDate"    DATE NOT NULL CHECK ( "ReleaseDate" <= now() )
DEFAULT now(),
    "LastUpdateDate" DATE NOT NULL CHECK ( "LastUpdateDate" <= now() )
DEFAULT now(),
    CONSTRAINT documentations_pk PRIMARY KEY ("Id")
);

CREATE TABLE IF NOT EXISTS "Terms"
(
    "Id"          UUID,
    "ReleaseDate" DATE CHECK ( "ReleaseDate" <= now() ) DEFAULT now(),
    "Pages"       INTEGER,
    "Version"     INTEGER NOT NULL                      DEFAULT 0,
    CONSTRAINT terms_pk PRIMARY KEY ("Id")
);

CREATE TABLE IF NOT EXISTS "Distributives"
(
    "Id"          UUID,
    "Name"        VARCHAR(256),
    "BuildDate"   DATE CHECK ( "BuildDate" <= now() ) DEFAULT now(),
    -- md5 hash is 128-bit length hash that is equal to 32 characters.
    "Hash"        CHAR(32),
    "Version"     INTEGER NOT NULL                    DEFAULT 0,
    "BuildNumber" INTEGER NOT NULL                    DEFAULT 0,
    CONSTRAINT distributives_pk PRIMARY KEY ("Id")
);

CREATE TABLE IF NOT EXISTS "Products"
(
    "Id"         UUID,
    "Price"      DECIMAL,
    "CustomerId" UUID,
    CONSTRAINT products_pk PRIMARY KEY ("Id"),
    CONSTRAINT product_customer_fk
        FOREIGN KEY ("CustomerId")
            REFERENCES "Customers" ("Id")
);

CREATE TABLE IF NOT EXISTS "Licences"
(
    "Id"             UUID,
    "Price"          DECIMAL,
    "SellDate"       DATE CHECK ( "SellDate" <= now() )       DEFAULT now(),
    "ExpirationDate" DATE CHECK ( "ExpirationDate" <= now() ) DEFAULT now(),
    "ProductId"      UUID,
    CONSTRAINT licences_pk PRIMARY KEY ("Id"),
    CONSTRAINT licences_products_fk
        FOREIGN KEY ("ProductId")
```

```
            REFERENCES "Products" ("Id")
);

CREATE TABLE IF NOT EXISTS "ProductDevelopers"
(
    "ProductId"   UUID,
    "DeveloperId" INTEGER,
    CONSTRAINT product_developers_pk
        PRIMARY KEY ("ProductId", "DeveloperId"),
    CONSTRAINT product_developers__product_fk
        FOREIGN KEY ("ProductId")
            REFERENCES "Products" ("Id"),
    CONSTRAINT product_developers__developer_fk
        FOREIGN KEY ("DeveloperId")
            REFERENCES "Developers" ("INN")
);

CREATE TABLE IF NOT EXISTS "ProductTesters"
(
    "ProductId" UUID,
    "TesterId"  INTEGER,
    CONSTRAINT product_testers_pk
        PRIMARY KEY ("ProductId", "TesterId"),
    CONSTRAINT product_testers__product_fk
        FOREIGN KEY ("ProductId")
            REFERENCES "Products" ("Id"),
    CONSTRAINT product_testers__tester_fk
        FOREIGN KEY ("TesterId")
            REFERENCES "Testers" ("INN")
);

CREATE TABLE IF NOT EXISTS "ProductDocumentations"
(
    "ProductId"       UUID,
    "DocumentationId" UUID,
    CONSTRAINT product_documentations_pk
        PRIMARY KEY ("ProductId", "DocumentationId"),
    CONSTRAINT product_documentations__product_fk
        FOREIGN KEY ("ProductId")
            REFERENCES "Products" ("Id"),
    CONSTRAINT product_documentations__documentation_fk
        FOREIGN KEY ("DocumentationId")
            REFERENCES "Documentations" ("Id")
);

CREATE TABLE IF NOT EXISTS "ProductTerms"
(
    "ProductId" UUID,
    "TermsId"   UUID,
    CONSTRAINT product_terms_pk
        PRIMARY KEY ("ProductId", "TermsId"),
    CONSTRAINT product_terms__product_fk
        FOREIGN KEY ("ProductId")
            REFERENCES "Products" ("Id"),
    CONSTRAINT product_terms__terms_fk
```

```
        FOREIGN KEY ("TermsId")
            REFERENCES "Terms" ("Id")
);

CREATE TABLE IF NOT EXISTS "ProductDistributives"
(
    "ProductId"      UUID,
    "DistributiveId" UUID,
    CONSTRAINT product_distributives_pk
        PRIMARY KEY ("ProductId", "DistributiveId"),
    CONSTRAINT product_distributives__product_fk
        FOREIGN KEY ("ProductId")
            REFERENCES "Products" ("Id"),
    CONSTRAINT product_distributives__distributives_fk
        FOREIGN KEY ("DistributiveId")
            REFERENCES "Distributives" ("Id")
);


-- Create indexes

CREATE UNIQUE INDEX developers_uindex ON "Developers" (
                                                       "INN" ASC
    );

CREATE UNIQUE INDEX testers_uindex ON "Testers" (
                                                 "INN" ASC
    );

CREATE UNIQUE INDEX organizations_uindex ON "Organizations" (
                                                             "Id" ASC
    );

CREATE UNIQUE INDEX customers_uindex ON "Customers" (
                                                     "Id" ASC
    );

CREATE UNIQUE INDEX products_uindex ON "Products" (
                                                   "Id" ASC,
                                                   "CustomerId" ASC
    );

CREATE INDEX licences_index ON "Licences" (
                                           "Id" ASC,
                                           "SellDate" DESC,
                                           "ExpirationDate" DESC
    );

CREATE INDEX licences_sell_date_index ON "Licences" (
                                                     "SellDate" DESC
    );

CREATE INDEX licences_expiration_date_index ON "Licences" (
                                                           "ExpirationDate"
DESC
```

```
    );

CREATE UNIQUE INDEX documentations_uindex ON "Documentations" (
                                                                "Id" ASC
    );

CREATE UNIQUE INDEX terms_uindex ON "Terms" (
                                            "Id" ASC
    );

CREATE UNIQUE INDEX distributives_uindex ON "Distributives" (
                                                            "Id" ASC
    );

CREATE UNIQUE INDEX product_developers__uindex ON "ProductDevelopers" (

"ProductId" ASC,

"DeveloperId" ASC
    );

CREATE UNIQUE INDEX product_testers__uindex ON "ProductTesters" (
                                                                 "ProductId"
ASC,
                                                                 "TesterId"
ASC
    );

CREATE UNIQUE INDEX product_documentations__uindex ON "ProductDocumentations"
(

"ProductId" ASC,

"DocumentationId" ASC
    );

CREATE UNIQUE INDEX product_distributives__uindex ON "ProductDistributives" (

"ProductId" ASC,

"DistributiveId" ASC
    );

CREATE UNIQUE INDEX product_terms__uindex ON "ProductTerms" (
                                                            "ProductId" ASC,
                                                            "TermsId" ASC
    );

CREATE UNIQUE INDEX individual_customers__uindex ON "Individuals" (
                                                                   "INN" ASC,
                                                                   "Id" ASC
    );

CREATE UNIQUE INDEX organizations_customers__uindex ON "Organizations" (
                                                                        "Id"
```

```
ASC
    );


-- Functions

CREATE OR REPLACE FUNCTION increment_version()
    RETURNS TRIGGER
AS
$body$
BEGIN
    NEW."Version" := NEW."Version" + 1;
    RETURN NEW;
END
$body$
    LANGUAGE plpgsql;

ALTER FUNCTION increment_version() OWNER TO CURRENT_USER;


-- Triggers

CREATE TRIGGER version_trigger_terms
    before update
    on "Terms"
    for EACH ROW
EXECUTE PROCEDURE increment_version();

CREATE TRIGGER version_trigger_documentations
    before update
    on "Terms"
    for EACH ROW
EXECUTE PROCEDURE increment_version();
```

Наведений код створить всі необхідні таблиці, тригери та функції, а також індекси для таблиць. В результаті виконання даного скрипта отримаємо таку схему БД.

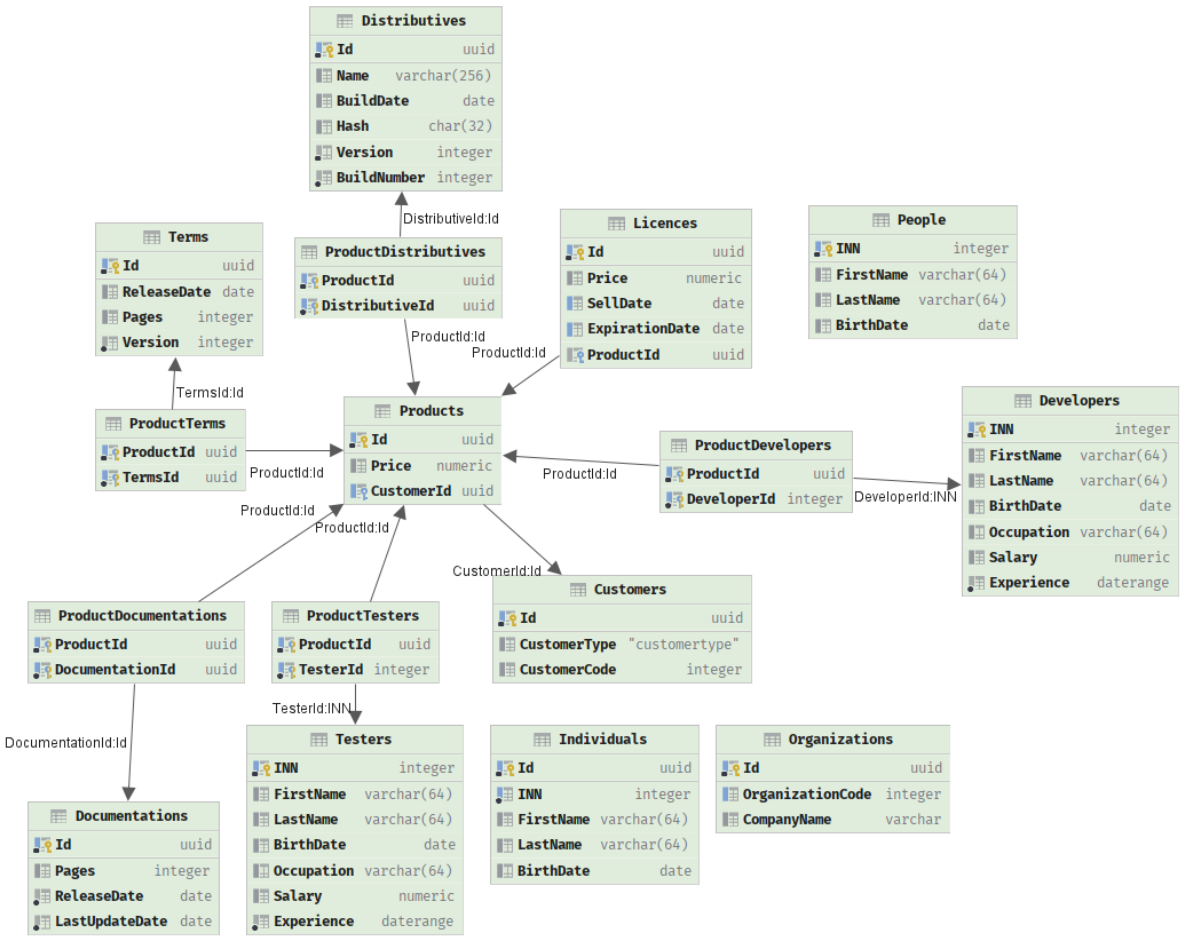
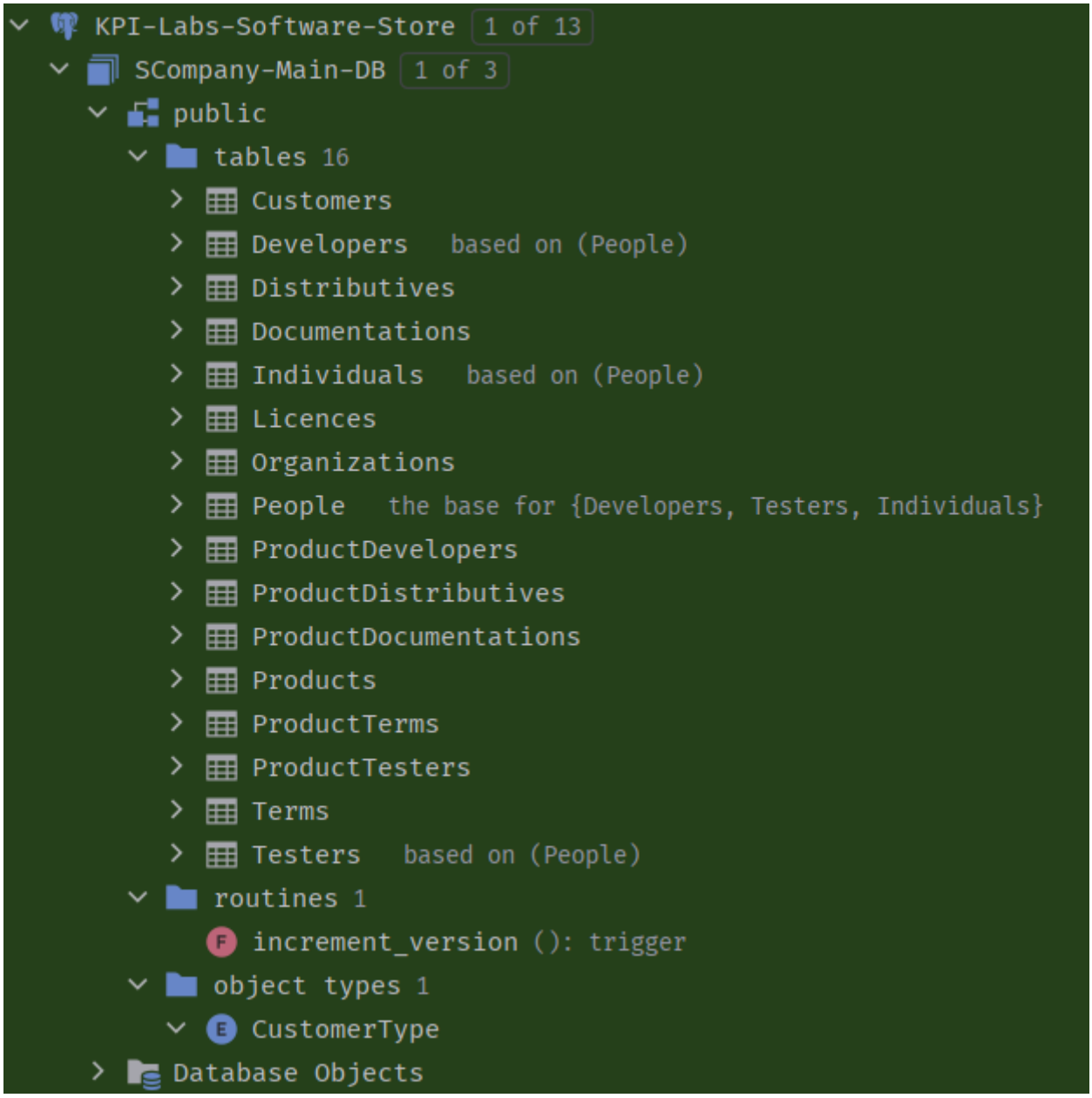


Рис. 2 — Схема створеної БД

А також такі елементи БД:



Заповнення бази даних тестовими даними

```
-- Fill developers table
INSERT INTO "Developers" ("INN", "FirstName", "LastName", "BirthDate",
"Occupation", "Salary", "Experience")
VALUES (1, 'Vladislav', 'Bardin', '2002-10-08', 'Junior .NET Developer',
1000, '[2021-02-05, 2021-09-19)'),
       (2, 'Igor', 'Erokhin', '2003-11-19', 'Junior Java Script Developer',
600, '[2021-08-05, 2021-10-21)');


-- Testers table
INSERT INTO "Testers" ("INN", "FirstName", "LastName", "BirthDate",
"Occupation", "Salary", "Experience")
VALUES (1, 'Maxime', 'Kurkin', '2002-09-10', 'Junior QA', 500, '[2021-08-05,
2021-10-21)');
```

```sql
-- Individual customers table
INSERT INTO "Individuals" ("INN", "Id", "FirstName", "LastName", "BirthDate")
VALUES (1, '95959856-05b3-42fd-a5f3-e3f3fa500112', 'Nastya', 'Stepanova',
'1987-07-03');


-- Organizations customers table
INSERT INTO "Organizations" ("Id", "OrganizationCode", "CompanyName")
VALUES ('aa08491d-d8fc-4216-ae45-d31459751382', '1', 'TranStar');


-- Customers table
INSERT INTO "Customers" ("Id", "CustomerType", "CustomerCode")
VALUES ('782e88d2-e7ba-433b-b3f9-7cea29f8fbfc', 'Individual', 1),
       ('100f6315-6fd2-41c8-9f14-4078c49795c9', 'Organization', 1);


-- Documentations table
INSERT INTO "Documentations" ("Id", "Pages")
VALUES ('47d7e17e-8b4b-4c1c-ab1e-3cf95ab6ecfa', 263),
       ('c864872b-c01d-4c69-92ea-a697854a6c39', 725),
       ('3c050e26-2209-4bf9-87d0-47ce455c5fe8', 315);


-- Terms table
INSERT INTO "Terms" ("Id", "Pages")
VALUES ('556cadac-f75a-46a3-bb95-0a6493135031', 104),
       ('405f241d-9ddc-4a7d-a3af-f3415a5c81d1', 536),
       ('0b9fe7dd-57f6-4949-a427-e363d3eaeff3', 917);


-- Distributives table
INSERT INTO "Distributives" ("Id", "Name", "BuildDate", "Hash", "Version",
"BuildNumber")
VALUES ('6993a04f-d313-4857-96f0-96b75c19a9fc', 'Vanity Fair', '2021-10-20',
'b08a86450cd4ede972c1b8ebe3aed3b8', 1, 1),
       ('bfd140da-cfae-4d35-997a-896e1f908cc2', 'X-Oreo', '2021-10-24',
'83f9825cbd677c74dfebf6a8079e60d5', 1, 2);


-- Products table
INSERT INTO "Products" ("Id", "Price", "CustomerId")
VALUES ('4a276c2e-ec71-416c-afc0-a28e24ad7b0c', 1491122, '782e88d2-e7ba-433b-
b3f9-7cea29f8fbfc'),
       ('f8b73f40-d501-4ed2-96d9-461204aaa191', 100000, '100f6315-6fd2-41c8-
9f14-4078c49795c9');


-- Licences table
INSERT INTO "Licences" ("Id", "Price", "ProductId")
VALUES ('6cdf8d30-9c0c-4be2-ac42-3304b2ab3742', 500, '4a276c2e-ec71-416c-
afc0-a28e24ad7b0c'),
       ('78a7cd89-c13e-4e2f-aa41-50b9f27d75fb', 649.99, 'f8b73f40-d501-4ed2-
96d9-461204aaa191');
```

```sql
-- ProductDevelopers table
INSERT INTO "ProductDevelopers" ("ProductId", "DeveloperId")
VALUES ('4a276c2e-ec71-416c-afc0-a28e24ad7b0c', 1),
       ('f8b73f40-d501-4ed2-96d9-461204aaa191', 2);


-- ProductTesters table
INSERT INTO "ProductTesters" ("ProductId", "TesterId")
VALUES ('4a276c2e-ec71-416c-afc0-a28e24ad7b0c', 1),
       ('f8b73f40-d501-4ed2-96d9-461204aaa191', 1);


-- ProductDistributives table
INSERT INTO "ProductDistributives" ("ProductId", "DistributiveId")
VALUES ('4a276c2e-ec71-416c-afc0-a28e24ad7b0c', '6993a04f-d313-4857-96f0-
96b75c19a9fc'),
       ('f8b73f40-d501-4ed2-96d9-461204aaa191', 'bfd140da-cfae-4d35-997a-
896e1f908cc2');


-- ProductDocumentations table
INSERT INTO "ProductDocumentations" ("ProductId", "DocumentationId")
VALUES ('4a276c2e-ec71-416c-afc0-a28e24ad7b0c', '47d7e17e-8b4b-4c1c-ab1e-
3cf95ab6ecfa'),
       ('f8b73f40-d501-4ed2-96d9-461204aaa191', 'c864872b-c01d-4c69-92ea-
a697854a6c39'),
       ('f8b73f40-d501-4ed2-96d9-461204aaa191', '3c050e26-2209-4bf9-87d0-
47ce455c5fe8');


-- ProductTerms table
INSERT INTO "ProductTerms" ("ProductId", "TermsId")
VALUES ('4a276c2e-ec71-416c-afc0-a28e24ad7b0c', '556cadac-f75a-46a3-bb95-
0a6493135031'),
       ('f8b73f40-d501-4ed2-96d9-461204aaa191', '405f241d-9ddc-4a7d-a3af-
f3415a5c81d1'),
       ('f8b73f40-d501-4ed2-96d9-461204aaa191', '0b9fe7dd-57f6-4949-a427-
e363d3eaeff3');
```

Для заповнення таблиць даними було використано наведений вище SQL код. Він дозволив додати до новоствореної БД певні тестові дані, а також створити готову для роботи БД на базі PostgreSQL.

Висновок:

В результаті виконання даної лабораторної роботи було написано SQL код для створення БД, таблиць, їх індексів, функцій, а також трігерів для маніпуляції даними. А також було усунуто дефекти 1-ї версії схеми з ЛР-1.