Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

ПСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет информатики и вычислительной техники

Кафедра вычислительной техники

ПРЕДМЕТ «Базы Данных»

# *КУРСОВОЙ ПРОЕКТ*

***«Вариант 12»***

РАЗРАБОТКА БАЗЫ ДАННЫХ В СУБД POSTGRESQL

Выполнил студент группы 0024-01

Невзоров Л.Е.

Проверил преподаватель

Вертешев А.С.

Псков

2022

**Задание**

Тема: учет обмена жилплощади.

Объекты: районы, клиенты, жилье, обмены жилья, тип жилья.

Для моделирования задачи необходимо хранить следующую информацию:

\* ФИО клиента

\* Адрес места работы клиента с указанием района

\* Характеристика жилья (адрес, район, размер жилья)

\* Тип жилья (1, 2, 3, -комнатная квартира)

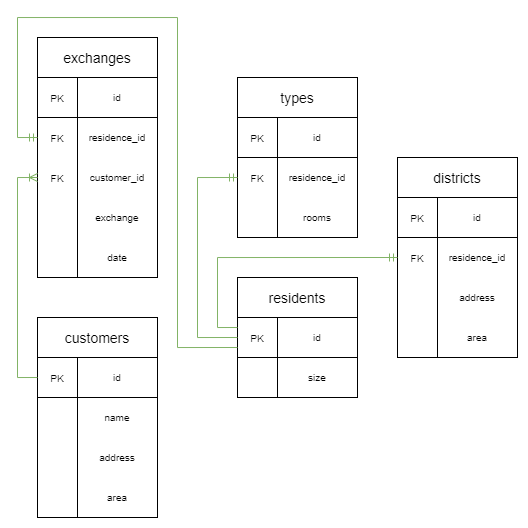
\* Обмен жилплощади (было, стало)

\* Дата обмена

Дополнительные условия:

\* один клиент может произвести несколько обменов жилья.

**Схема**



**Создание таблиц**

**Создание таблицы customers:**

CREATE TABLE "customers" (

  id VARCHAR PRIMARY KEY,

  name TEXT NOT NULL,

  address VARCHAR(100) NOT NULL,

  area VARCHAR(100) NOT NULL

);

INSERT INTO customers (id,name,address,area)

VALUES

  ('EEP25VSP7KO','Valentine Wise','P.O. Box 311, 762 Accumsan Avenue','Mazowieckie'),

  ('WJM65UOL8BE','Baxter Pierce','Ap #515-1237 Natoque St.','North Maluku'),

  ('EQV62NND5LI','Ruby Lindsey','Ap #236-245 Cursus. Avenue','Agder'),

  ('HSK10IBQ6IB','Cameron Chen','7705 Sit Av.','Mykolaiv oblast'),

  ('HGX94TOM8ZS','Jena Potter','Ap #787-3794 Non, St.','New South Wales');

**Создание таблицы exchanges:**

CREATE TABLE "exchanges" (

  id SERIAL PRIMARY KEY,

  residence\_id VARCHAR(7) NOT NULL REFERENCES residents(id) ON DELETE CASCADE,

  customer\_id VARCHAR(11) NOT NULL REFERENCES customers(id) ON DELETE CASCADE,

  exchange TEXT NOT NULL,

  date VARCHAR(19)

);

INSERT INTO exchanges (residence\_id,customer\_id,exchange,date)

VALUES

  ('D3J 4S3','EEP25VSP7KO','True','2021-11-10 18:47:25'),

  ('N4Y 6B5','EEP25VSP7KO','False',''),

  ('C8W 4O3','WJM65UOL8BE','False',''),

  ('Q1J 5L5','WJM65UOL8BE','True','2021-11-07 14:11:36'),

  ('M6C 4P1','EQV62NND5LI','False','');

**Создание таблицы residents:**

CREATE TABLE "residents" (

  id VARCHAR(7) PRIMARY KEY,

  size INTEGER

);

INSERT INTO residents (id,size)

VALUES

  ('D3J 4S3',15),

  ('N4Y 6B5',30),

  ('C8W 4O3',60),

  ('Q1J 5L5',25),

  ('M6C 4P1',40);

**Создание таблицы types:**

CREATE TABLE "types" (

  id SERIAL PRIMARY KEY,

  residence\_id VARCHAR(7) UNIQUE NOT NULL REFERENCES residents(id) ON DELETE CASCADE,

  rooms INTEGER

);

INSERT INTO types (residence\_id,rooms)

VALUES

  ('D3J 4S3',2),

  ('N4Y 6B5',1),

  ('C8W 4O3',3),

  ('Q1J 5L5',3),

  ('M6C 4P1',2);

**Создание таблицы districts:**

CREATE TABLE "districts" (

  id SERIAL PRIMARY KEY,

  residence\_id VARCHAR(7) UNIQUE NOT NULL REFERENCES residents(id) ON DELETE CASCADE,

  address VARCHAR(100) NOT NULL,

  area VARCHAR(100) NOT NULL

);

INSERT INTO districts (residence\_id,address,area)

VALUES

  ('D3J 4S3','Ap #797-2017 Bibendum. Rd.','Noord Holland'),

  ('N4Y 6B5','Ap #690-5187 Tempor Rd.','Connacht'),

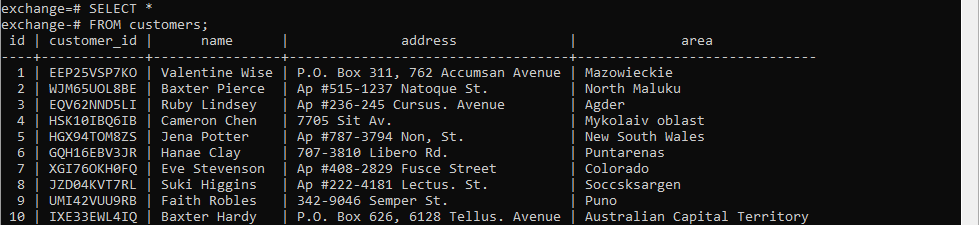
  ('C8W 4O3','P.O. Box 199, 8252 Ullamcorper. Ave','Calabria'),

  ('Q1J 5L5','P.O. Box 832, 7181 Sit Av.','South Island'),

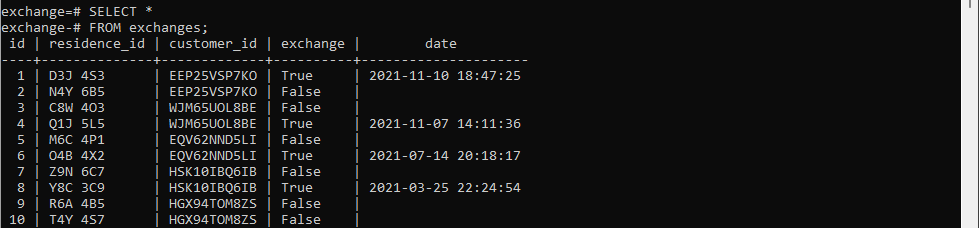
  ('M6C 4P1','Ap #350-3512 Morbi Rd.','Limburg');

**Таблицы**

**customers**



**exchanges**



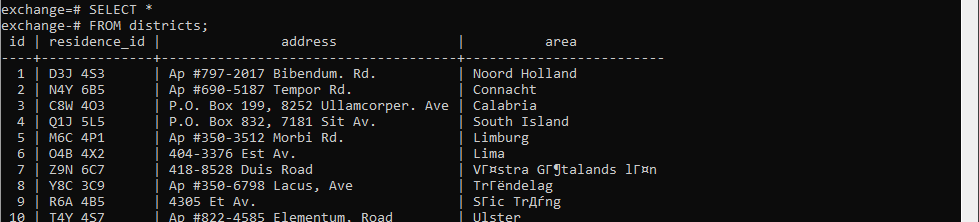
**residents**



**types**



**districts**



**Запросы**

1)Три запроса на выборку с использованием join, использовать не мене 3 таблиц

SELECT cu.name,

       ex.exchange,

       re.size

FROM customers AS cu

JOIN exchanges AS ex ON cu.customer\_id=ex.customer\_id

JOIN residents AS re ON ex.residence\_id=re.residence\_id;



SELECT re.size,

       ty.rooms,

       di.area

FROM residents AS re

JOIN types AS ty ON re.residence\_id=ty.residence\_id

JOIN districts AS di ON re.residence\_id=di.residence\_id;



SELECT cu.name,

       ex.exchange,

       re.size,

       ty.rooms,

       di.area

FROM customers AS cu

JOIN exchanges AS ex ON cu.customer\_id=ex.customer\_id

JOIN residents AS re ON ex.residence\_id=re.residence\_id

JOIN types AS ty ON re.residence\_id=ty.residence\_id

JOIN districts AS di ON re.residence\_id=di.residence\_id;



[2) Запросы на объединение, пересечение, разность, произведение, проекцию, деление](#_Toc91305837)

SELECT customer\_id,

       name

FROM customers

WHERE name LIKE '%S%'

UNION

SELECT customer\_id,

       name

FROM customers

WHERE name LIKE '%T%'

LIMIT 10;



SELECT residence\_id,

       rooms

FROM types AS ty

WHERE rooms > 0

INTERSECT

SELECT residence\_id,

       rooms

FROM types AS ty

WHERE rooms < 3

LIMIT 10;



SELECT residence\_id,

       rooms

FROM types

EXCEPT

SELECT residence\_id,

       rooms

FROM types

WHERE rooms < 3

LIMIT 10;



SELECT size,

       L AS letter

FROM residents as re CROSS JOIN (VALUES ('A'), ('B'), ('C')) b(L)

WHERE size < 40

  AND size > 20

LIMIT 10;



[3) Запросы на соединение: left join, right join, full outer join](#_Toc91305838)

Были добавлены строки в customers и exchanges, которые не имеют связи между таблицами, для демонстрации LEFT, RIGHT и FULL JOIN’ов:

SELECT cu.name,

       ex.exchange

FROM customers AS cu

LEFT OUTER JOIN exchanges AS ex ON cu.id=ex.customer\_id;



SELECT cu.name,

       ex.exchange

FROM customers AS cu

RIGHT OUTER JOIN exchanges AS ex ON cu.id=ex.customer\_id;



SELECT cu.name,

       ex.exchange

FROM customers AS cu

FULL OUTER JOIN exchanges AS ex ON cu.id=ex.customer\_id;



4) 2 подзапроса: простой, сложный

SELECT cu.name,

       ex.exchange,

       ex.date,

       re.size

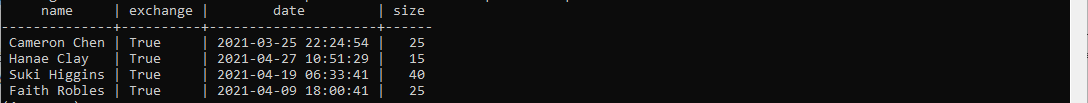
FROM customers AS cu

LEFT OUTER JOIN exchanges AS ex ON cu.customer\_id=ex.customer\_id

LEFT OUTER JOIN residents AS re ON ex.residence\_id=re.residence\_id

WHERE ex.date BETWEEN '2021-01-01' AND '2021-06-31'

  AND ex.exchange = 'True';



SELECT cu.name,

       ex.exchange,

       ex.date,

       re.size

FROM customers AS cu

LEFT OUTER JOIN exchanges AS ex ON cu.customer\_id=ex.customer\_id

LEFT OUTER JOIN residents AS re ON ex.residence\_id=re.residence\_id

WHERE re.size > (SELECT AVG(re.size)

                 FROM residents AS re

                 JOIN exchanges AS ex ON re.residence\_id=ex.residence\_id

                 JOIN types AS ty ON re.residence\_id=ty.residence\_id

                 WHERE ex.exchange = 'True'

                   AND ty.rooms > 1);



5) Запросы с различными уровнями вложенности подзапросов в частях select, From where, having

SELECT re.residence\_id,

       re.size,

       ty.rooms,

       di.address,

       di.area

FROM residents AS re

JOIN types AS ty ON re.residence\_id=ty.residence\_id

JOIN districts AS di ON re.residence\_id=di.residence\_id

WHERE re.size < (SELECT SUM(re.size) - AVG(re.size) \* 5

                 FROM residents AS re

                 JOIN exchanges AS ex ON re.residence\_id=ex.residence\_id

                 JOIN types AS ty ON re.residence\_id=ty.residence\_id

                 WHERE ex.exchange = 'True'

                   AND ty.rooms > 1);



SELECT cu.name,

       COUNT(ex.exchange)

FROM customers AS cu

JOIN exchanges AS ex ON cu.customer\_id=ex.customer\_id

WHERE ex.exchange = 'True'

GROUP BY cu.name

HAVING COUNT(ex.exchange) >= (SELECT MIN(tbl.count)

                              FROM (SELECT cu.name,

                                           COUNT(ex.exchange)

                                    FROM customers AS cu

                                    JOIN exchanges AS ex ON cu.customer\_id=ex.customer\_id

                                    WHERE ex.exchange = 'True'

                                    GROUP BY cu.name) AS tbl);



SELECT tbl.name,

       tbl.date,

       tbl.exchange

FROM (SELECT cu.name,

       ex.date,

       ex.exchange,

       re.size,

       ty.rooms,

       di.area

FROM customers AS cu

JOIN exchanges AS ex ON cu.customer\_id=ex.customer\_id

JOIN residents AS re ON ex.residence\_id=re.residence\_id

JOIN types AS ty ON re.residence\_id=ty.residence\_id

JOIN districts AS di ON re.residence\_id=di.residence\_id) AS tbl

WHERE tbl.name LIKE 'Valentine%'

   OR tbl.name LIKE '%Potter'

   OR tbl.size > 50;



SELECT tbl.name,

       tbl.date,

       tbl.exchange,

       (SELECT COUNT(address)

        FROM districts

        WHERE tbl.name LIKE 'Valentine%'

           OR tbl.name LIKE '%Potter'

           OR tbl.size > 50) AS address\_count

FROM (SELECT cu.name,

       ex.date,

       ex.exchange,

       re.size,

       ty.rooms,

       di.area

FROM customers AS cu

JOIN exchanges AS ex ON cu.customer\_id=ex.customer\_id

JOIN residents AS re ON ex.residence\_id=re.residence\_id

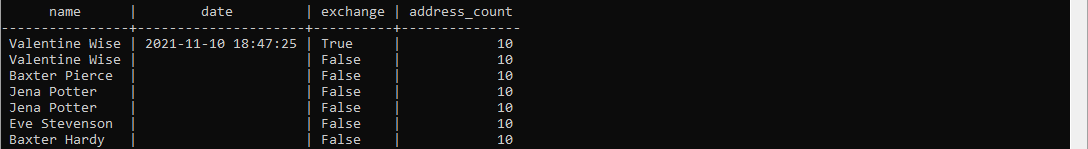
JOIN types AS ty ON re.residence\_id=ty.residence\_id

JOIN districts AS di ON re.residence\_id=di.residence\_id) AS tbl

WHERE tbl.name LIKE 'Valentine%'

   OR tbl.name LIKE '%Potter'

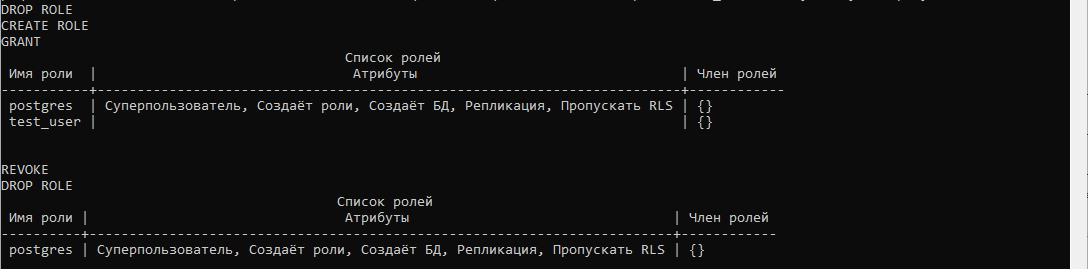
   OR tbl.size > 50;



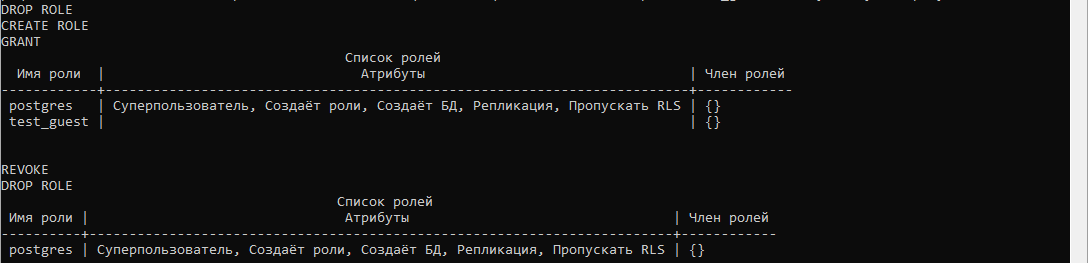
[6) Создание ролей пользователей user, guest](#_Toc91305840)

CREATE ROLE test\_user WITH LOGIN PASSWORD 'password';

GRANT ALL ON DATABASE exchange TO test\_user;

  
  
CREATE ROLE test\_guest WITH LOGIN PASSWORD 'guest!password';

GRANT ALL ON residents TO test\_guest;



Проверка:





[7) Создание двух транзакций с тремя точками восстановления на удаление, создание транзакции на вставку](#_Toc91305841)

UPDATE balance SET balance = balance - 400000.00

    WHERE customer\_id = 'GQH16EBV3JR';

SAVEPOINT save\_point;

UPDATE balance SET balance = balance + 400000.00

    WHERE customer\_id = 'XGI76OKH0FQ';

SAVEPOINT save\_point;

UPDATE balance SET balance = balance - 400000.00

    WHERE customer\_id = 'GQH16EBV3JR';

SAVEPOINT save\_point;

UPDATE balance SET balance = balance + 400000.00

    WHERE customer\_id = 'JZD04KVT7RL';

ROLLBACK TO save\_point;

UPDATE balance SET balance = balance + 400000.00

    WHERE customer\_id = 'XGI76OKH0FQ';

COMMIT;



[8) Создание двух представлений: изменяемое, неизменяемое](#_Toc91305842)

-- создадим изменяемое представление

CREATE OR REPLACE VIEW district\_view AS

    SELECT \*

    FROM districts

    WHERE area = 'Noord Holland'

       OR area = 'Connacht';

SELECT \*

FROM district\_view;

-- изменим представление

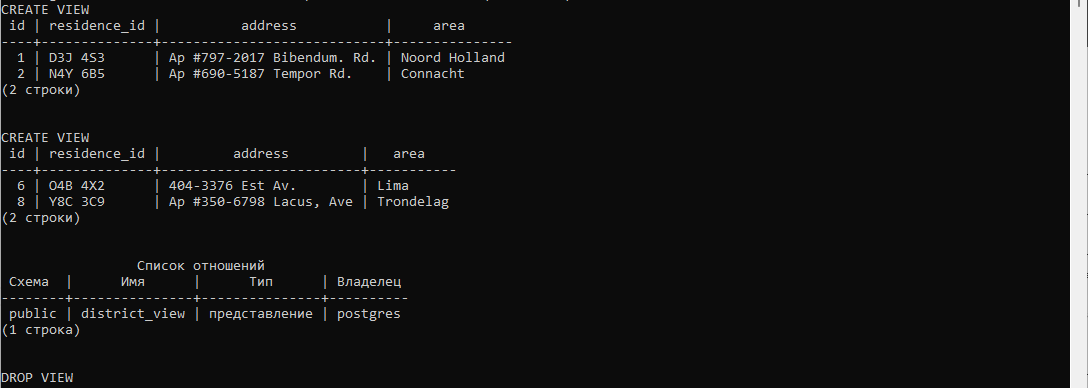
CREATE OR REPLACE VIEW district\_view AS

    SELECT \*

    FROM districts

    WHERE area = 'Lima'

       OR area = 'Trondelag';



-- создадим неизменяемое представление

CREATE VIEW customer\_view AS

    SELECT \*

    FROM customers

    WHERE name = 'Valentine Wise'

       OR name = 'Baxter Pierce'

    WITH CASCADED CHECK OPTION;

SELECT \*

FROM customer\_view;

-- попробуем его изменить

CREATE VIEW customer\_view AS

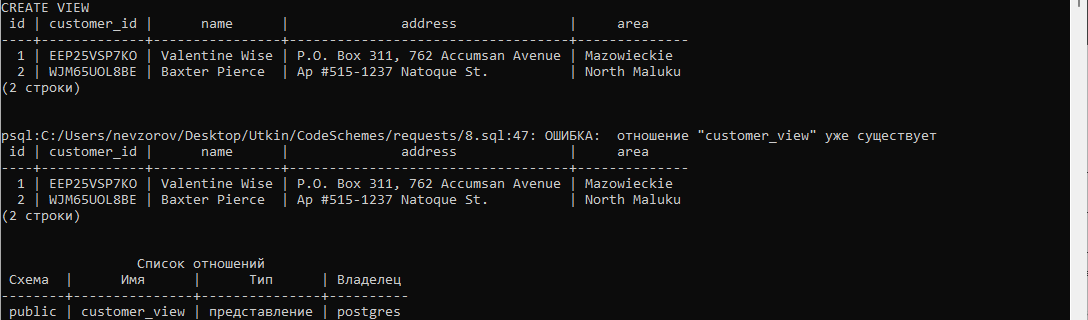
    SELECT \*

    FROM customers

    WHERE name = 'Hanae Clay'

       OR name = 'Eve Stevenson'

    WITH CASCADED CHECK OPTION;



[9) Создание функции](#_Toc91305843)

CREATE FUNCTION mean(x float, y float) RETURNS INTEGER AS $$

     SELECT x / y;

$$ LANGUAGE SQL;

SELECT mean((SELECT SUM(balance) FROM balance),

            (SELECT COUNT(balance) FROM balance)) AS mean\_balance;



[10) Создание триггера](#_Toc91305844)

-- создание функции

CREATE FUNCTION customer\_stamp() RETURNS trigger AS $customer\_stamp$

    BEGIN

        IF NEW.name IS NULL THEN

            RAISE EXCEPTION 'name cannot be null';

        END IF;

        IF NEW.address IS NULL THEN

            RAISE EXCEPTION '% address cannot be null', NEW.address;

        END IF;

        IF NEW.area IS NULL THEN

            RAISE EXCEPTION '% area cannot be null', NEW.area;

        END IF;

        RETURN NEW;

    END;

$customer\_stamp$ LANGUAGE plpgsql;

-- создание триггера

CREATE TRIGGER customer\_stamp BEFORE INSERT OR UPDATE ON customers

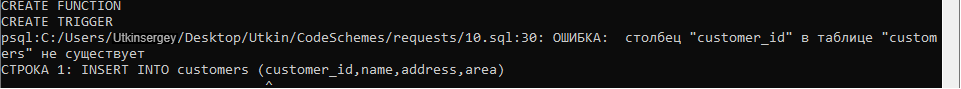
    FOR EACH ROW EXECUTE FUNCTION customer\_stamp();

-- проверка неправильными данными

INSERT INTO customers (name, address, area)

VALUES

  ('Alfred', NULL, 'Briton Beach');



[11). Создание процедуры](#_Toc91305845)

-- удалние через процедуру

CREATE PROCEDURE clean\_emp() AS '

    DELETE FROM exams

        WHERE mark < 3;

' LANGUAGE SQL;

CALL clean\_emp();



[12) Шифрование столбцов](#_Toc91305846)

CREATE EXTENSION IF NOT EXISTS pgcrypto;

CREATE TABLE exam\_table (

   id SERIAL PRIMARY KEY,

   discipline\_id VARCHAR(150) NOT NULL,

   student\_id VARCHAR(150) NOT NULL,

   mark INTEGER NOT NULL

);

INSERT INTO exam\_table (discipline\_id,student\_id,mark)

VALUES

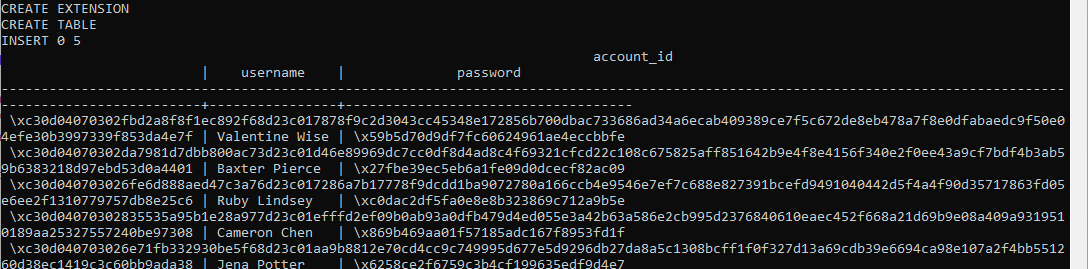
  (pgp\_sym\_encrypt('6','arSlan!PassWorD'), encrypt('667511','arSlan!PassWorD', 'aes'), 5),

  (pgp\_sym\_encrypt('1','arSlan!PassWorD'), encrypt('717505','arSlan!PassWorD', 'aes'), 4),

  (pgp\_sym\_encrypt('7','arSlan!PassWorD'), encrypt('791997','arSlan!PassWorD', 'aes'), 4),

  (pgp\_sym\_encrypt('3','arSlan!PassWorD'), encrypt('418318','arSlan!PassWorD', 'aes'), 3),

  (pgp\_sym\_encrypt('5','arSlan!PassWorD'), encrypt('351472','arSlan!PassWorD', 'aes'), 5);



13) Расшифровка столбцов

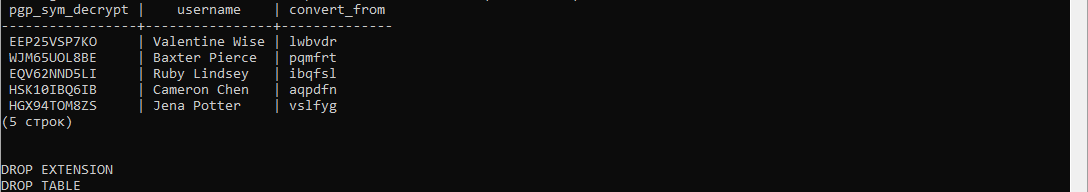
-- прочитаем расшифрованные пароли

SELECT  pgp\_sym\_decrypt(discipline\_id::bytea, 'arSlan!PassWorD'),

        convert\_from(decrypt(student\_id::bytea, 'arSlan!PassWorD', 'aes'), 'SQL\_ASCII'),

        mark

FROM exam\_table;



14) Создание объектных типов данных

CREATE TYPE gender\_type AS ENUM ('M', 'W', 'N/S');

CREATE TABLE gender\_table (

    id serial,

    name text,

    gender gender\_type

);

INSERT INTO gender\_table (id, name, gender)

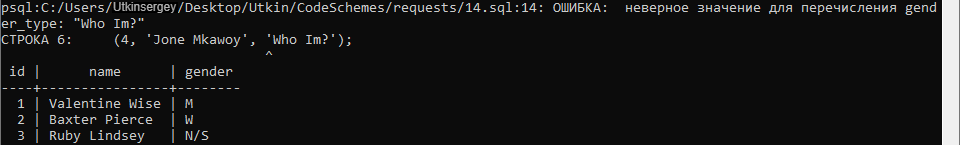
VALUES

    (1, 'Valentine Wise', 'M'),

    (2, 'Baxter Pierce', 'W'),

    (3, 'Ruby Lindsey', 'N/S'),

    (4, 'Jone Mkawoy', 'Who Im?');



15) Перекрестные запросы

CREATE EXTENSION IF NOT EXISTS tablefunc;

CREATE TABLE cross\_request (

    name TEXT,

    area TEXT,

    balance INTEGER

);

INSERT INTO cross\_request (name,area,balance)

VALUES

    ('Valentine Wise', 'Mazowieckie', 135),

    ('Valentine Wise', 'Colorado', 752),

    ('Baxter Pierce', 'Mazowieckie', 159),

    ('Baxter Pierce', 'Colorado', 231),

    ('Ruby Lindsey', 'Mazowieckie', 852),

    ('Ruby Lindsey', 'Colorado', 178);

SELECT \*

FROM crosstab(

    'SELECT name,

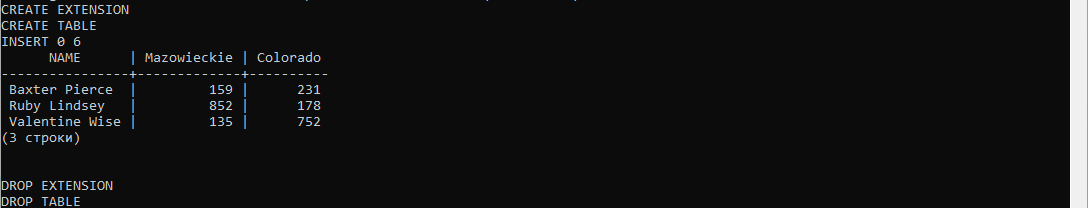
            area,

            balance

     FROM   cross\_request

     ORDER  BY 1'

   ) AS ct ("NAME" text, "Mazowieckie" int, "Colorado" int);



16) Запрос на перевод в формат JSON

SELECT array\_to\_json(array\_agg(row\_to\_json (cu))) FROM (

    SELECT cu.name,

           cu.area,

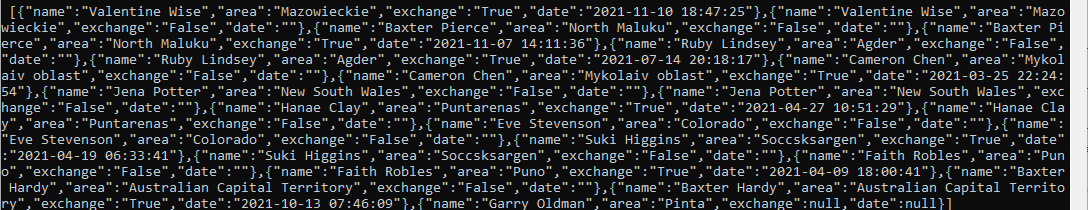
           ex.exchange,

           ex.date

    FROM customers AS cu

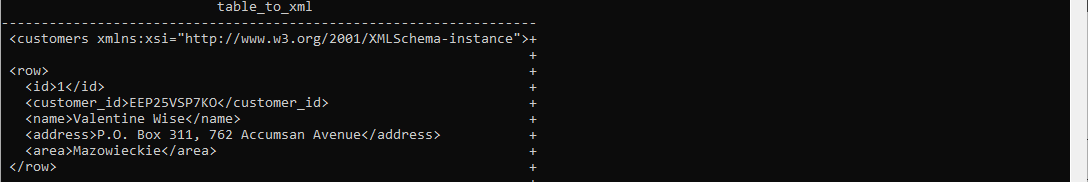
    LEFT OUTER JOIN exchanges AS ex ON cu.customer\_id=ex.customer\_id

) cu;



17) Запрос на перевод в формат XML

SELECT table\_to\_xml('customers', true, false, '');



SELECT query\_to\_xml('SELECT \* FROM exchanges', true, false, '');

