|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ  Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |

Институт комплексной безопасности и специального приборостроения

Кафедра КБ-14 «Цифровые технологии обработки данных»

### **Программные средства манипулирования данными**

**Отчёт**

по практическим работам №2-6

Выполнил:

Студент 3 курса

Группа БСБО-05-22

Шифр 22Б0723

Парамонов

Павел

Константинович

Москва, 2024

Архитектура Базы данных

**Таблица Помещения(Places)**

|  |  |  |
| --- | --- | --- |
| **Ру. Название** | **Название в СУБД** | **Пояснения** |
| Id помещения | place\_id |  |
| Город | city |  |
| Адрес | address |  |
| Почтовый индекс | index |  |
| Телефонный номер | number |  |
| Количество сотрудников | employer\_count | Создать функцию или что-то в духе чтобы считать автоматически |

**Таблица Тип обслуживания в помещении(places\_service\_type)**

|  |  |  |
| --- | --- | --- |
| **Ру. Название** | **Название в СУБД** | **Пояснения** |
| Id помещения | place\_id | На случай если помещение обслуживает и то и то |
| Тип обслуживания | service\_type |

**Таблица сотрудников(Employers)**

|  |  |  |
| --- | --- | --- |
| **Ру. Название** | **Название в СУБД** | **Пояснения** |
| Id сотрудника | employer\_id | Возможно лучше использовать данные паспорта |
| Город | city |  |
| ФИО | fist/middle/last \_name | Разделить на 3 |
| Контактные данные | email/number | Разделить на 2 |
| Стаж | exp |  |
| Возраст | age |  |
| Id должности | Profession\_id |  |
| Краткая информация | info | мега странная колонка мб кучу сторонних таблиц надо |

**Таблица должность(profession)**

|  |  |  |
| --- | --- | --- |
| **Ру. Название** | **Название в СУБД** | **Пояснения** |
| Id должности | Profession\_id |  |
| Должность | profession |  |

**Таблица должность-зарплата(proffesion\_salary) доп балл**

|  |  |  |
| --- | --- | --- |
| **Ру. Название** | **Название в СУБД** | **Пояснения** |
| Id должности | Profession\_id |  |
| зарплата | salary |  |

Решил объединить в одну^^^

**Таблица Мест работы(test\_name){пока не придумал нормальное название}**

|  |  |  |
| --- | --- | --- |
| **Ру. Название** | **Название в СУБД** | **Пояснения** |
| id помещения | place\_id |  |
| id сотрудника | employer\_id |  |
| Город | city |  |

**Таблица Клиентов(Clients)**

|  |  |  |
| --- | --- | --- |
| **Ру. Название** | **Название в СУБД** | **Пояснения** |
| id клиента | client\_id |  |
| ФИО | fist/middle/last \_name | Разделить на 3 |
| Контактные данные | email/number | Разделить на 2 |
| Статус клиента | status | тоже мега хитрый столбец со сложными вычислениями |
| Бонусы | bonuses | Ура очередной столбец с хитрыми вычеслениями/ функция сгорания |

**Таблица услуг(services)**

|  |  |  |
| --- | --- | --- |
| **Ру. Название** | **Название в СУБД** | **Пояснения** |
| id услуги | service\_id |  |
| Краткая информация | info | мега странная колонка мб кучу сторонних таблиц надо |
| Тип обслуживания | service\_type | Мото или авто |
| Цена | price |  |
| Время | hours |  |

**Таблица предоставления услуг по помещениям(services\_in\_places)**

|  |  |  |
| --- | --- | --- |
| **Ру. Название** | **Название в СУБД** | **Пояснения** |
| id услуги | service\_id |  |
| Id помещения | place\_id |  |

**Таблица деталей(details)**

|  |  |  |
| --- | --- | --- |
| **Ру. Название** | **Название в СУБД** | **Пояснения** |
| Id детали | Detail\_id |  |
| Характеристики | \_\_ | Зависит от желания упоротся |
| Цена | Price |  |

**Таблица наличия деталей на складах(details\_in\_places)**

|  |  |  |
| --- | --- | --- |
| **Ру. Название** | **Название в СУБД** | **Пояснения** |
| Id детали | Detail\_id |  |
| Id помещения | place\_id |  |

**Таблица заказов(orders)**

|  |  |  |
| --- | --- | --- |
| **Ру. Название** | **Название в СУБД** | **Пояснения** |
| id заказа | order\_id |  |
| id клиента | client\_id | мб паспорт |
| id помещения | place\_id |  |
| Мастер выполняющий заказ | master |  |
| Срок выполнения | deadline |  |
| Статус заказа | order\_status |  |
| Цена | final\_price | Очередные хитрые вычесления просто прелесть |

**Таблица Часть заказа-Детали(order\_part\_details)**

|  |  |  |
| --- | --- | --- |
| **Ру. Название** | **Название в СУБД** | **Пояснения** |
| id заказа | order\_id |  |
| id детали | detail\_id |  |
| Количество | amount |  |

**Таблица Часть заказа-Услуги(order\_part\_details)**

|  |  |  |
| --- | --- | --- |
| **Ру. Название** | **Название в СУБД** | **Пояснения** |
| id заказа | order\_id |  |
| id услуги | service\_id |  |
| Количество | amount |  |

*\*\*Чек будет являться запросом подсасывающим данные из трех таблиц выше\*\**

**Создание таблиц в СУБД**

CREATE TABLE places (

place\_id SERIAL PRIMARY KEY,

city VARCHAR(255) NOT NULL,

adress VARCHAR(255) NOT NULL,

post\_code VARCHAR(20),

phone\_number VARCHAR(20),

emp\_count int

);

CREATE TABLE places\_service\_type (

place\_id INT, FOREIGN KEY (place\_id) REFERENCES places (place\_id),

service\_type VARCHAR(10) NOT NULL

);

CREATE TABLE professions (

profession\_id SERIAL PRIMARY KEY,

profession VARCHAR(100) NOT NULL,

salary INT NOT NULL

);

CREATE TABLE employers (

employer\_id SERIAL PRIMARY KEY,

city VARCHAR(100) NOT NULL,

first\_name VARCHAR(255) NOT NULL,

middle\_name VARCHAR(255),

last\_name VARCHAR(255) NOT NULL,

email VARCHAR(255),

phone\_number VARCHAR(20) UNIQUE,

expirience INT,

age INT NOT NULL,

profession\_id INT, FOREIGN KEY (profession\_id) REFERENCES professions(profession\_id),

employer\_info VARCHAR(500)

);

CREATE TABLE where\_works (

place\_id INT, FOREIGN KEY (place\_id) REFERENCES places (place\_id),

employer\_id INT, FOREIGN KEY (employer\_id) REFERENCES employers (employer\_id),

city VARCHAR(100)

);

CREATE TABLE clients (

client\_id SERIAL PRIMARY KEY,

first\_name VARCHAR(255) NOT NULL,

middle\_name VARCHAR(255),

last\_name VARCHAR(255) NOT NULL,

email VARCHAR(255),

phone\_number VARCHAR(20) UNIQUE,

client\_status VARCHAR(20) NOT NULL,

bonus INT DEFAULT(0) NOT NULL

);

CREATE TABLE services (

service\_id SERIAL PRIMARY KEY,

info VARCHAR(500) NOT NULL,

service\_type VARCHAR(10),

price MONEY,

work\_time TIME

);

CREATE TABLE services\_in\_places (

service\_id INT, FOREIGN KEY (service\_id) REFERENCES services (service\_id),

place\_id INT, FOREIGN KEY (place\_id) REFERENCES places (place\_id)

);

CREATE TABLE details (

detail\_id SERIAL PRIMARY KEY,

detail\_type VARCHAR(10),

detail\_info VARCHAR(500) NOT NULL,

price MONEY NOT NULL

);

CREATE TABLE details\_in\_places (

detail\_id INT, FOREIGN KEY (detail\_id) REFERENCES details (detail\_id),

place\_id INT, FOREIGN KEY (place\_id) REFERENCES places (place\_id)

);

CREATE TABLE orders (

order\_id SERIAL PRIMARY KEY,

client\_id INT, FOREIGN KEY (client\_id) REFERENCES clients (client\_id),

employer\_id INT, FOREIGN KEY (employer\_id) REFERENCES employers (employer\_id),

place\_id INT, FOREIGN KEY (place\_id) REFERENCES places (place\_id),

deadline DATE,

order\_status VARCHAR(100),

final\_price MONEY

);

CREATE TABLE order\_part\_details (

order\_id INT, FOREIGN KEY (order\_id) REFERENCES orders (order\_id),

detail\_id INT, FOREIGN KEY (detail\_id) REFERENCES details (detail\_id),

amount INT NOT NULL

);

CREATE TABLE order\_part\_services (

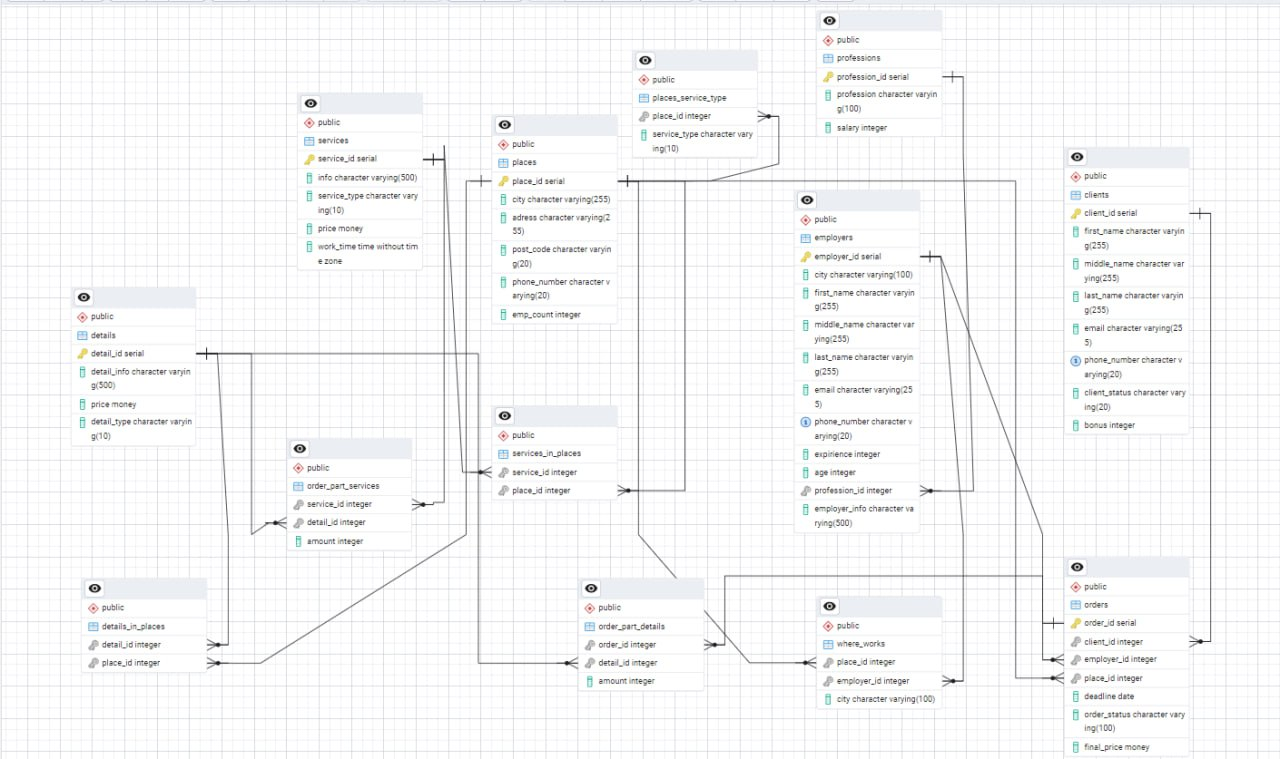
order\_id INT, FOREIGN KEY (order\_id) REFERENCES orders (order\_id),

service\_id INT, FOREIGN KEY (dservice\_id) REFERENCES services (service\_id),

amount INT NOT NULL

);

Получившаяся ERD-диаграмма



**Заполним таблицы тестовыми данными**

**Заполнение places**

INSERT INTO places(city,adress,post\_code,phone\_number)

VALUES

('Москва','Новослобская, д30','111252','88005553535'),

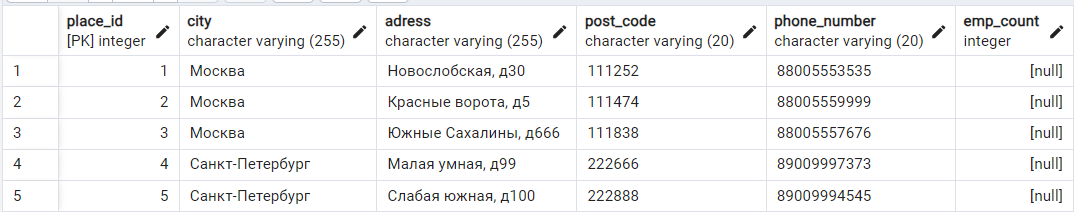
('Москва','Красные ворота, д5','111474','88005559999'),

('Москва','Южные Сахалины, д666','111838','88005557676'),

('Санкт-Петербург','Малая умная, д99','222666','89009997373'),

('Санкт-Петербург','Слабая южная, д100','222888','89009994545');

Emp\_count будет заполняться позже с использованием функций и тригеров



**Заполнение Places\_service\_type**

INSERT INTO places\_service\_type(place\_id,service\_type)

VALUES

(1,'Мото'),

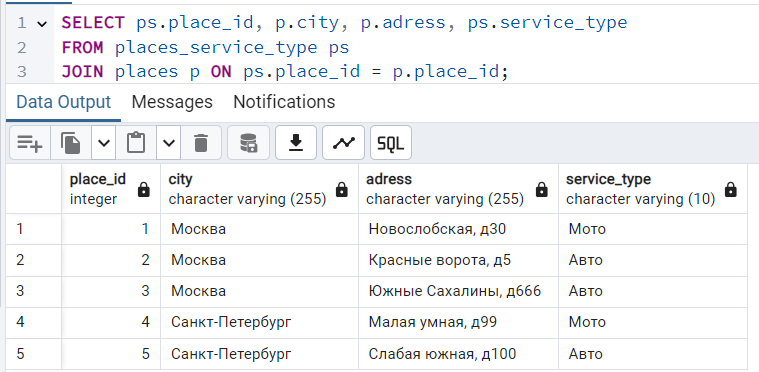
(2,'Авто'),

(3,'Авто'),

(4,'Мото'),

(5,'Авто');

Т.к таблица связана с таблицей places



**Заполнение Professions**

INSERT INTO professions(profession,salary)

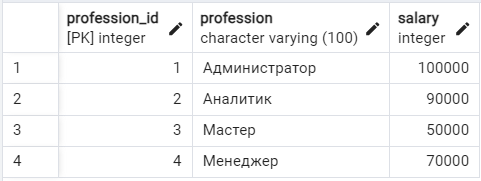
VALUES

('Администратор',100000),

('Аналитик',90000),

('Мастер',50000),

('Менеджер',70000);



**Заполнение Employers**

INSERT INTO employers(city,first\_name,middle\_name,last\_name,email,phone\_number,expirience,age,profession\_id,employer\_info)

VALUES

('Москва', 'Иван', 'Сергеевич', 'Петров', 'ivan.petrov@example.com', '+79161234567', 10, 35, 1, 'Опытный администратор автосервиса'),

('Санкт-Петербург', 'Ольга', 'Игоревна', 'Смирнова', 'olga.smirnova@example.com', '+79261234568', 5, 29, 2, 'Специалист по аналитике данных'),

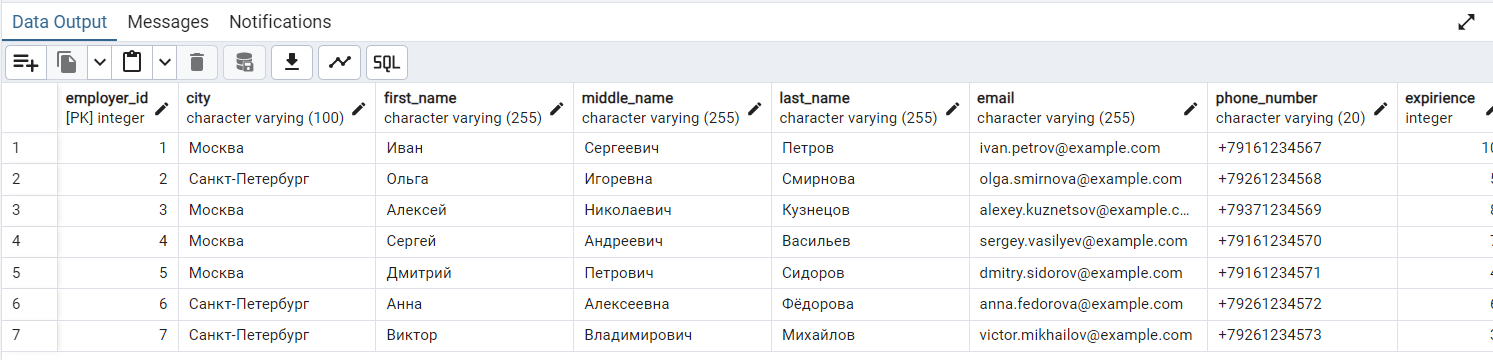
('Москва', 'Алексей', 'Николаевич', 'Кузнецов', 'alexey.kuznetsov@example.com', '+79371234569', 8, 33, 4, 'Управляет клиентами и заказами автосервиса'),

('Москва', 'Сергей', 'Андреевич', 'Васильев', 'sergey.vasilyev@example.com', '+79161234570', 7, 40, 3, 'Специалист по ремонту двигателей'),

('Москва', 'Дмитрий', 'Петрович', 'Сидоров', 'dmitry.sidorov@example.com', '+79161234571', 4, 28, 3, 'Механик по работе с подвеской'),

('Санкт-Петербург', 'Анна', 'Алексеевна', 'Фёдорова', 'anna.fedorova@example.com', '+79261234572', 6, 32, 3, 'Специалист по шиномонтажу'),

('Санкт-Петербург', 'Виктор', 'Владимирович', 'Михайлов', 'victor.mikhailov@example.com', '+79261234573', 3, 25, 3, 'Мастер по диагностике электроники');



**Заполнение Where\_works**

INSERT INTO where\_works(place\_id,employer\_id)

VALUES

(1,4),

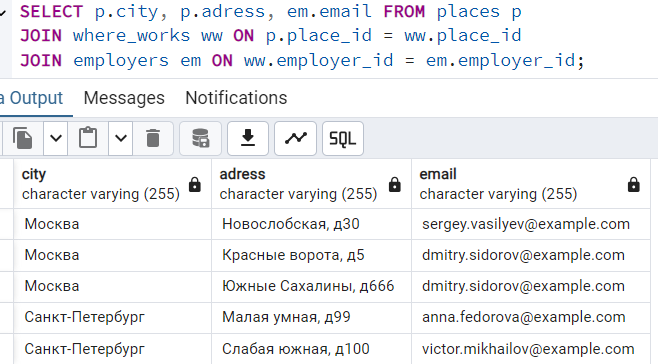
(2,5),

(3,5),

(4,6),

(5,7);

Пример использования данной таблицы, так же реализовано, что один сотрудник может работать в нескольких местах



**Заполнение Clients**

INSERT INTO clients(first\_name, middle\_name, last\_name,email, phone\_number, client\_status)

VALUES

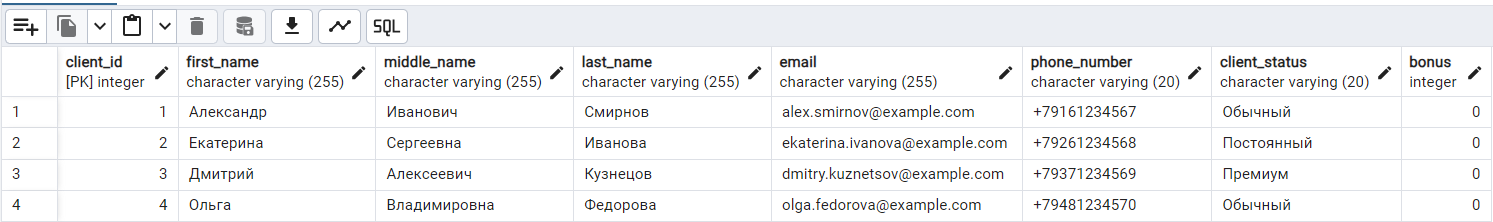
('Александр', 'Иванович', 'Смирнов', 'alex.smirnov@example.com', '+79161234567', 'Обычный'),

('Екатерина', 'Сергеевна', 'Иванова', 'ekaterina.ivanova@example.com', '+79261234568', 'Постоянный'),

('Дмитрий', 'Алексеевич', 'Кузнецов', 'dmitry.kuznetsov@example.com', '+79371234569', 'Премиум'),

('Ольга', 'Владимировна', 'Федорова', 'olga.fedorova@example.com', '+79481234570', 'Обычный');

Функционал бонусов и их сгорания будет добавлен с помощью функций и тригеров



**Заполнение Services**

INSERT INTO services(info, service\_type, price, work\_time)

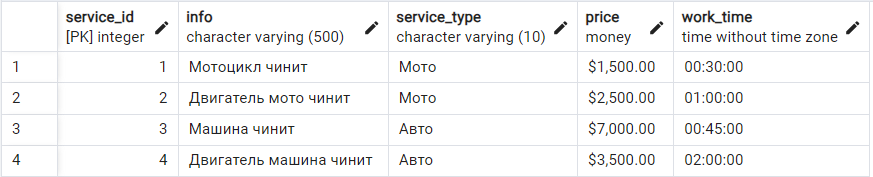
VALUES

('Мотоцикл чинит', 'Мото', 1500.00, '00:30:00'),

('Двигатель мото чинит', 'Мото', 2500.00, '01:00:00'),

('Машина чинит', 'Авто', 7000.00, '00:45:00'),

('Двигатель машина чинит', 'Авто', 3500.00, '02:00:00');



**Заполнение Services\_in\_places**

INSERT INTO services\_in\_places(service\_id,place\_id)

VALUES

(1,1),

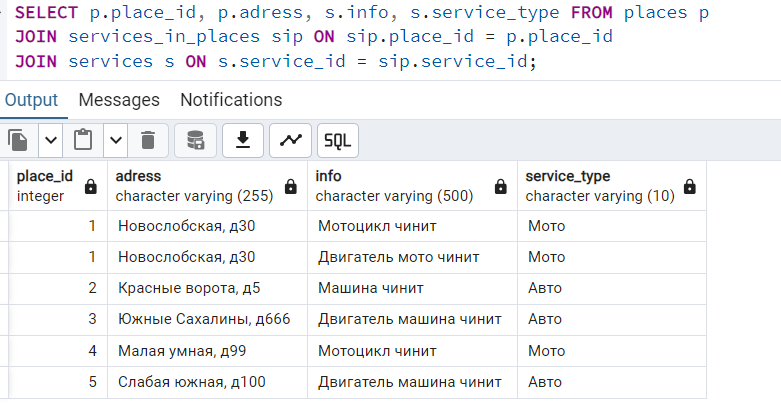
(2,1),

(3,2),

(4,3),

(1,4),

(4,5);

Пример использования таблицы, так же благодаря данному заполнения может появиться ситуация, что нужной услуги в данном салоне может не быть и клиенту посоветуют обратиться в другой салон.  


**Заполнение Details**

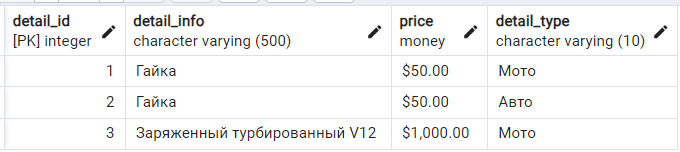
INSERT INTO details(detail\_info, price, detail\_type)

VALUES

('Гайка',50.00,'Мото'),

('Гайка',50.00,'Авто'),

('Заряженный турбированный V12',1000.00,'Мото');



**Заполнение Detais\_in\_places**

INSERT INTO details\_in\_places(detail\_id,place\_id)

VALUES

(1,1),

(3,1),

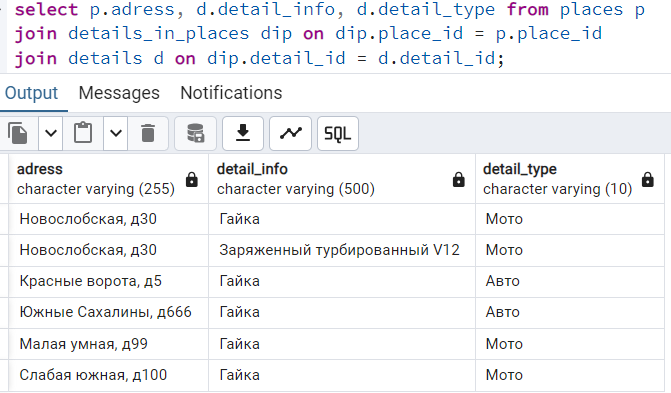
(2,2),

(2,3),

(1,4),

(1,5);

Пример использования

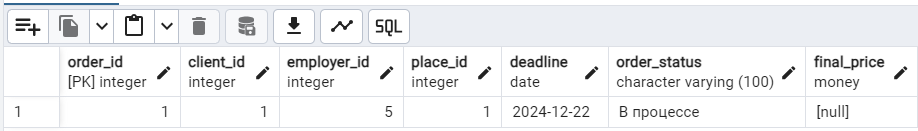


**Заполнение Orders**

INSERT INTO orders(client\_id,employer\_id,place\_id,deadline,order\_status)

VALUES

(1,5,1,'2024-12-22','В процессе');



**Заполнение Order\_part\_details**

INSERT INTO order\_part\_details(order\_id, detail\_id, amount)

VALUES

(1,1,1);

**Заполнение Order\_part\_service**

INSERT INTO order\_part\_services(order\_id, service\_id, amount)

VALUES

(1,1,1),

(1,2,1);

**Создание ролей и групп ролей и назначение привилегий.**

**Роль Администратора**

CREATE ROLE admin\_role;

GRANT ALL PRIVILEGES ON DATABASE big\_moto\_big\_auto TO admin\_role;

GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO admin\_role; GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA public TO admin\_role; GRANT ALL PRIVILEGES ON ALL FUNCTIONS IN SCHEMA public TO admin\_role;

**Роль Аналитика**

CREATE ROLE analyst\_role;

GRANT CONNECT ON DATABASE big\_moto\_big\_auto TO analyst\_role;

GRANT USAGE ON SCHEMA public TO analyst\_role;

GRANT SELECT ON ALL TABLES IN SCHEMA public TO analyst\_role;

ALTER DEFAULT PRIVILEGES IN SCHEMA public GRANT SELECT ON TABLES TO analyst\_role;

**Роль Мастера**

CREATE ROLE master\_role;

GRANT CONNECT ON DATABASE big\_moto\_big\_auto TO master\_role;

GRANT USAGE ON SCHEMA public TO master\_role;

GRANT SELECT, INSERT, UPDATE ON TABLE orders TO master\_role;

GRANT SELECT, UPDATE ON TABLE clients TO master\_role;

**Роль Менеджера**

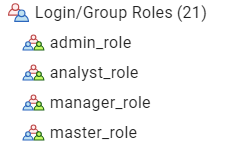
CREATE ROLE manager\_role;

GRANT CONNECT ON DATABASE big\_moto\_big\_auto TO manager\_role;

GRANT USAGE ON SCHEMA public TO manager\_role;

GRANT SELECT, INSERT, UPDATE ON TABLE clients TO manager\_role;

GRANT SELECT, INSERT, UPDATE ON TABLE orders TO manager\_role;



**Настройка политик безопасности**

Менеджер может работать только с клиентами и заказами, связанными с его автосервисом, однако я считаю очень не эффективным, что клиент может быть привязан к сервису, в случае, если клиент вдруг обратиться в другой сервис, то нужно будет либо заново регистрировать пользователя, но уже в новом сервисе и все его бонусы сгорят, либо добавить систему где его будут “привязывать” к каждому сервису на уровне БД. Поэтому я считаю, что лучше сделать так, что менеджер будет иметь доступ ко всем пользователям на чтение, и изменение их статуса, но иметь доступ только к заказам в его салоне

ALTER TABLE clients ENABLE ROW LEVEL SECURITY;

ALTER TABLE orders ENABLE ROW LEVEL SECURITY;

CREATE OR REPLACE POLICY manager\_clients\_policy ON clients

FOR SELECT USING (true);

CREATE OR REPLACE POLICY manager\_clients\_update\_policy ON clients

FOR UPDATE USING (true)

WITH CHECK (client\_status IS NOT NULL);

CREATE OR REPLACE POLICY manager\_orders\_policy ON orders

FOR SELECT USING (

EXISTS (

SELECT 1

FROM where\_works ww

WHERE ww.employer\_id = current\_user::INT

AND ww.place\_id = orders.place\_id

)

);

CREATE OR REPLACE POLICY manager\_orders\_update\_policy ON orders

FOR UPDATE USING (

EXISTS (

SELECT 1

FROM where\_works ww

WHERE ww.employer\_id = current\_user::INT

AND ww.place\_id = orders.place\_id

)

);

CREATE OR REPLACE POLICY manager\_orders\_insert\_policy ON orders

FOR INSERT USING (

EXISTS (

SELECT 1

FROM where\_works ww

WHERE ww.employer\_id = current\_user::INT

AND ww.place\_id = NEW.place\_id

)

);

GRANT SELECT, UPDATE ON TABLE clients TO manager\_role;

GRANT SELECT, INSERT, UPDATE ON TABLE orders TO manager\_role;

Мастер может работать только с заказами и клиентами, с которыми он напрямую связан.

CREATE POLICY master\_orders\_policy ON orders

FOR SELECT USING (

orders.employer\_id = current\_user::INT

);

CREATE POLICY master\_orders\_update\_policy ON orders

FOR UPDATE USING (

orders.employer\_id = current\_user::INT

);

CREATE POLICY master\_clients\_policy ON clients

FOR SELECT USING (

EXISTS (

SELECT 1

FROM orders

WHERE orders.client\_id = clients.client\_id

AND orders.employer\_id = current\_user::INT

)

);

GRANT SELECT, INSERT, UPDATE ON TABLE orders TO master\_role;

GRANT SELECT ON TABLE clients TO master\_role;

ALTER TABLE clients FORCE ROW LEVEL SECURITY;

ALTER TABLE orders FORCE ROW LEVEL SECURITY;

**Автоматизация функционала за счёт добавление функций, процедур , триггеров.**

Функция обновления emp\_count

CREATE OR REPLACE FUNCTION update\_emp\_count()

RETURNS TRIGGER AS $$

BEGIN

UPDATE places

SET emp\_count = (

SELECT COUNT(\*)

FROM where\_works ww

WHERE ww.place\_id = NEW.place\_id

)

WHERE place\_id = NEW.place\_id;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER emp\_count\_trigger

AFTER INSERT OR DELETE OR UPDATE ON where\_works

FOR EACH ROW

EXECUTE FUNCTION update\_emp\_count();

Пример



Функция начисления бонусов, бонус зависит от статуса клиента

CREATE OR REPLACE FUNCTION set\_bonus\_based\_on\_status()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.client\_status = 'Обычный' THEN

NEW.bonus := 0;

ELSIF NEW.client\_status = 'Постоянный' THEN

NEW.bonus := 10;

ELSIF NEW.client\_status = 'Премиум' THEN

NEW.bonus := 25;

ELSE

NEW.bonus := 0;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER set\_bonus\_trigger

BEFORE INSERT OR UPDATE ON clients

FOR EACH ROW

EXECUTE FUNCTION set\_bonus\_based\_on\_status();

Функция для сгорания бонусов через год

CREATE OR REPLACE FUNCTION downgrade\_inactive\_clients()

RETURNS VOID AS $$

BEGIN

UPDATE clients

SET client\_status = 'Обычный', bonus = 0

WHERE client\_status != 'Обычный'

AND client\_id NOT IN (

SELECT DISTINCT client\_id

FROM orders

WHERE deadline >= NOW() - INTERVAL '1 year'

);

END;

$$ LANGUAGE plpgsql;

Проверять данную функцию можно при помощи планировщика задач

Функция применения бонуса к заказу

CREATE OR REPLACE FUNCTION apply\_bonus(order\_id INT)

RETURNS VOID AS $$

DECLARE

client\_bonus INT;

original\_price NUMERIC;

new\_price NUMERIC;

BEGIN

SELECT c.bonus, o.final\_price

INTO client\_bonus, original\_price

FROM orders o

JOIN clients c ON o.client\_id = c.client\_id

WHERE o.order\_id = order\_id;

IF client\_bonus IS NULL OR original\_price IS NULL THEN

RAISE EXCEPTION 'Invalid order\_id or client\_id';

END IF;

new\_price := original\_price \* (client\_bonus/100);

UPDATE orders

SET final\_price = new\_price

WHERE order\_id = order\_id;

END;

$$ LANGUAGE plpgsql;

**Оптимизация работы за счёт индексов**

CREATE INDEX idx\_places\_city ON places (city);

CREATE INDEX idx\_clients\_client\_status ON clients (client\_status);

CREATE INDEX idx\_orders\_client\_id ON orders (client\_id);

**Задания с повышенной сложностью**

Система должна обеспечивать клиентам возможность бронировать время для получения услуг в автосервисе или мотосервисе. Менеджеры предлагают клиентам выбрать тип услуги, удобное время и дату, а также указать предпочтительного мастера (если такая опция предусмотрена). Забронированное время резервируется менеджером для клиента и отображается в системе.

Для данного задания создадим таблицу time\_slots

CREATE TABLE time\_slots (

slot\_id SERIAL PRIMARY KEY,

place\_id INT REFERENCES places(place\_id) ON DELETE CASCADE,

master\_id INT REFERENCES employers(employer\_id) ON DELETE CASCADE,

start\_time TIMESTAMP NOT NULL,

end\_time TIMESTAMP NOT NULL,

is\_booked BOOLEAN DEFAULT FALSE

);

Заполним её тестовыми данными

INSERT INTO time\_slots(place\_id, master\_id, start\_time, end\_time)

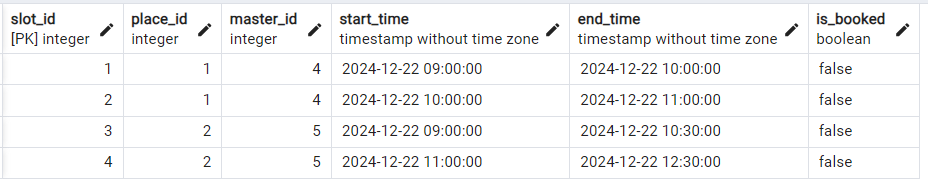
VALUES

(1, 4, '2024-12-22 09:00:00', '2024-12-22 10:00:00'),

(1, 4, '2024-12-22 10:00:00', '2024-12-22 11:00:00'),

(2, 5, '2024-12-22 09:00:00', '2024-12-22 10:30:00'),

(2, 5, '2024-12-22 11:00:00', '2024-12-22 12:30:00');



Так же создадим функцию которая будет автоматически добавлять промежутки времени

(данная функция заполняет данные на 10 дней вперёд для определённого мастера в определённом салоне, в зависимости от нужд функцию можно модифицировать)

CREATE OR REPLACE FUNCTION create\_time\_slots(master\_id INT, place\_id INT)

RETURNS VOID AS $$

DECLARE

current\_day DATE := CURRENT\_DATE;

slot\_start TIMESTAMP;

slot\_end TIMESTAMP;

day\_counter INT;

BEGIN

FOR day\_counter IN 0..9 LOOP

FOR slot\_start, slot\_end IN

SELECT

ts::TIMESTAMP,

(ts::TIMESTAMP + INTERVAL '1 hour')

FROM UNNEST(ARRAY[

current\_day::TIMESTAMP + INTERVAL '10:00',

current\_day::TIMESTAMP + INTERVAL '12:00',

current\_day::TIMESTAMP + INTERVAL '14:00'

]) AS ts

LOOP

INSERT INTO time\_slots (place\_id, master\_id, start\_time, end\_time)

VALUES (place\_id, master\_id, slot\_start, slot\_end);

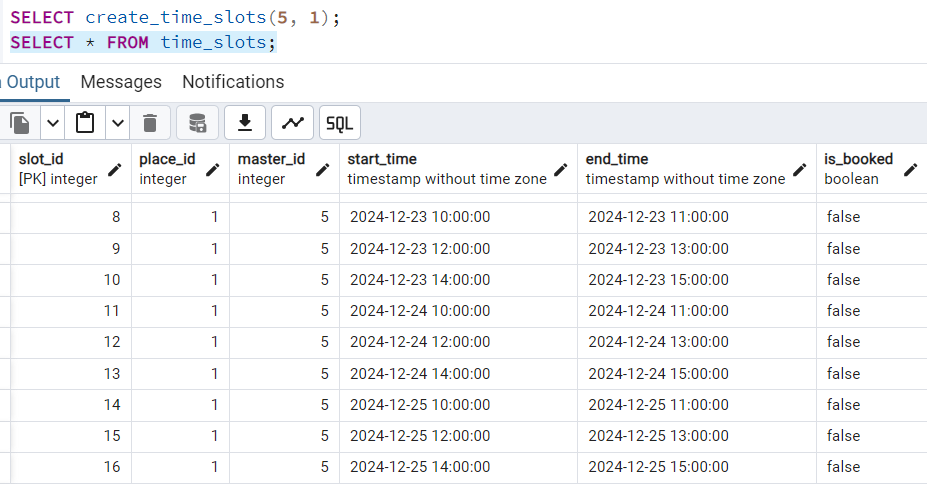
END LOOP;

current\_day := current\_day + INTERVAL '1 day';

END LOOP;

END;

$$ LANGUAGE plpgsql;



Для брони создадим таблицу bookings

CREATE TABLE bookings (

booking\_id SERIAL PRIMARY KEY,

client\_id INT REFERENCES clients(client\_id) ON DELETE CASCADE,

slot\_id INT REFERENCES time\_slots(slot\_id) ON DELETE CASCADE,

booking\_time TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

status VARCHAR(20) DEFAULT 'Active' -- 'Active', 'Completed', 'Cancelled'

);

Создадим функцию бронирования

CREATE OR REPLACE FUNCTION book\_time\_slot(client\_id INT, slot\_id INT)

RETURNS VOID AS $$

BEGIN

IF EXISTS (

SELECT 1

FROM time\_slots

WHERE time\_slots.slot\_id = $2

AND is\_booked = FALSE

) THEN

UPDATE time\_slots

SET is\_booked = TRUE

WHERE time\_slots.slot\_id = $2;

INSERT INTO bookings (client\_id, slot\_id)

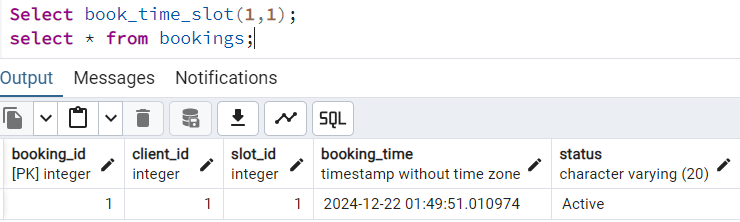
VALUES ($1, $2);

ELSE

RAISE EXCEPTION 'Selected time slot is not available';

END IF;

END;

$$ LANGUAGE plpgsql;

Если все доступные временные слоты для обслуживания заняты, система должна автоматически предложить клиенту альтернативные варианты. Например, система может предложить обратиться в другой филиал компании или оставить заявку на ожидание, указав предпочтительное время начала работы. Необходимо произвести автоматизацию процесса.

Создадим таблицу waitlist  
CREATE TABLE waitlist (

waitlist\_id SERIAL PRIMARY KEY,

client\_id INT REFERENCES clients(client\_id) ON DELETE CASCADE,

preferred\_time TIMESTAMP NOT NULL,

preferred\_service\_id INT REFERENCES services(service\_id) ON DELETE CASCADE,

notes VARCHAR(255)

);

А так же функцию которая будет предлагать алтернативу либо добавлять пользователя в список ожидания

CREATE OR REPLACE FUNCTION suggest\_alternatives(client\_id INT, preferred\_time TIMESTAMP)

RETURNS TABLE (place\_id INT, master\_id INT, start\_time TIMESTAMP, end\_time TIMESTAMP) AS $$

BEGIN

RETURN QUERY

SELECT ts.place\_id, ts.master\_id, ts.start\_time, ts.end\_time

FROM time\_slots ts

WHERE ts.is\_booked = FALSE

AND ts.start\_time >= preferred\_time

ORDER BY ts.start\_time;

IF NOT FOUND THEN

INSERT INTO waitlist (client\_id, preferred\_time, preferred\_service\_id)

VALUES (client\_id, preferred\_time, service\_id);

END IF;

END;

$$ LANGUAGE plpgsql;

В последствии можно будет реализовать функционал, отправки пользователю сообщения на мобильный телефон с указанными доступными интервалами, соответствующие требованиям в списке ожидания.

Реализовать систему ведения статистики о бронировании и выполненных услугах. Это может включать в себя отчеты о количестве бронирований по датам, наиболее востребованные услуги и другие показатели, необходимые для анализа и улучшения работы сервиса.

Добавим таблицу Statistics

CREATE TABLE statistics (

stat\_id SERIAL PRIMARY KEY,

report\_date DATE NOT NULL,

total\_bookings INT DEFAULT 0,

most\_requested\_time TIMESTAMP

);

И функцию сбора статистики

CREATE OR REPLACE FUNCTION update\_statistics()

RETURNS VOID AS $$

BEGIN

INSERT INTO statistics (report\_date, total\_bookings, most\_requested\_time)

SELECT

CURRENT\_DATE,

COUNT(\*),

(SELECT start\_time FROM time\_slots WHERE is\_booked = TRUE GROUP BY start\_time ORDER BY COUNT(\*) DESC LIMIT 1)

FROM bookings

WHERE booking\_time::DATE = CURRENT\_DATE;

END;

$$ LANGUAGE plpgsql;