Министерство науки и высшего образования РФ

федеральное государственное автономное  
образовательное учреждение высшего образования

«Омский государственный технический университет»

Факультет информационных технологий и компьютерных систем

Кафедра «Прикладная математика и фундаментальная информатика»

|  |
| --- |
| **Лабораторная работа №2** |
| по дисциплине **Системы управления базами данных**  **Тема: СОЗДАНИЕ НОВОЙ БАЗЫ ДАННЫХ В СРЕДЕ POSTGRESQL, СОЗДАНИЕ И СВЯЗЫВАНИЕ ТАБЛИЦ БАЗЫ ДАННЫХ В СРЕДЕ POSTGRESQL** |

|  |  |  |  |
| --- | --- | --- | --- |
| Студента | Кузиной Варвары Михайловны | | |
| Курс | 3 | Группа | МО-231 |
| Направление | 02.03.03 Математическое обеспечение и администрирование информационных систем | | |
| Руководитель | доц., к.н. | | |
|  | Моисеева Н.А. | | |
| Выполнил | 06.10 | | |
| Проверил |  | | |

Омск 2025

**Цель работы:**

Освоить базовые приёмы администрирования PostgreSQL: создание пользовательской базы данных sales, выбор кодировки и локали, проверка и документирование параметров созданной БД.

**Ход работы:**

1. Подключение к локальному серверу PostgreSQL в pgAdmin 4 (узел Servers → Local PostgreSQL).

2. Открытие редактора запросов (Tools → Query Tool) в базе postgres.

3. Создание базы данных sales. Для избежания конфликта с шаблонной базой использован «чистый» шаблон template0 и провайдер локали ICU:

CREATE DATABASE sales

WITH

OWNER = postgres

ENCODING = 'UTF8'

TEMPLATE = template0

LC\_COLLATE = 'Russian\_Russia.1251'

LC\_CTYPE = 'Russian\_Russia.1251'

TABLESPACE = pg\_default

CONNECTION LIMIT = -1

IS\_TEMPLATE = False;

4. Добавление комментария к базе данных:

COMMENT ON DATABASE sales IS 'База данных для службы оформления заказов на товары';

5. Обновление дерева объектов (ПКМ по Databases → Refresh) и подключение к созданной базе sales

**Пояснение параметров CREATE DATABASE:**

• OWNER = postgres — владелец БД суперпользователь postgres.

• ENCODING = 'UTF8' — текст хранится в кодировке UTF‑8.

• TEMPLATE = template0 — клонирование «чистой» шаблонной базы для смены локали.

• LC\_COLLATE = 'Russian\_Russia.1251' — локаль сортировки строк (как сравнивать/упорядочивать текст) через системный провайдер *libc* Windows. Значение Russian\_Russia.1251 даёт русские правила сортировки (Windows-1251 locale).

• LC\_CTYPE = 'Russian\_Russia.1251' — Локаль классификации символов (что считается буквой/цифрой, верхний/нижний регистр и т.д.). Должна совпадать с LC\_COLLATE, поэтому тоже Russian\_Russia.1251.

• TABLESPACE = pg\_default — табличное пространство по умолчанию.

• CONNECTION LIMIT = -1 — без ограничения на число одновременных подключений.

• IS\_TEMPLATE = False — базу нельзя клонировать всем пользователям, только суперпользователю/владельцу.

**Проверка параметров созданной БД:**

Параметры базы sales получены запросом (рисунок 1):

SELECT

d.datname AS database,

pg\_get\_userbyid(d.datdba) AS owner,

pg\_encoding\_to\_char(d.encoding) AS encoding,

d.datcollate,

d.datctype,

t.spcname AS tablespace,

d.datconnlimit AS connection\_limit,

d.datistemplate AS is\_template,

shobj\_description(d.oid, 'pg\_database') AS comment,

pg\_size\_pretty(pg\_database\_size(d.datname)) AS db\_size

FROM pg\_database AS d

LEFT JOIN pg\_tablespace AS t ON t.oid = d.dattablespace

WHERE d.datname = 'sales';

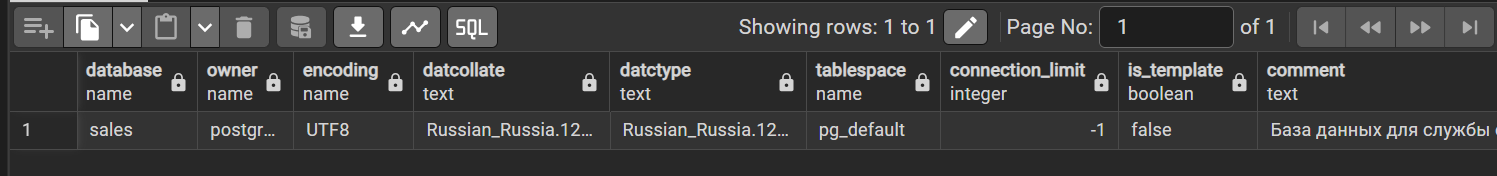


Рисунок 1 — Результат проверки

**Заключение:**

В ходе работы создана пользовательская база данных sales с кодировкой UTF-8 и русскими системными локалями (libc) — Russian\_Russia.1251 для LC\_COLLATE и LC\_CTYPE, на основе шаблона template0. Зафиксированы и проверены ключевые параметры БД: владелец postgres, табличное пространство pg\_default, отсутствие лимита подключений (CONNECTION LIMIT = -1), флаг IS\_TEMPLATE = False. Результаты подтверждены скриншотами из pgAdmin и контрольными SQL-запросами.

**Цель работы:**

Закрепить навыки проектирования и реализации схемы базы данных в PostgreSQL: создать таблицы предметной области, определить первичные ключи и обязательность полей, добавить внешние ключи и ограничения целостности, получить ER-диаграмму и сформировать отчёт.

**Ход работы:**

1. Подключились к базе данных sales в pgAdmin 4 (Servers → Local PostgreSQL → Databases → sales).

2. Открыли редактор SQL (Tools → Query Tool).

3. Выполнили скрипт создания таблиц (первичные ключи и NOT NULL определены в CREATE TABLE).

4. Отдельными ALTER TABLE добавили внешние ключи и правила каскадирования (ON DELETE/UPDATE CASCADE).

5. Обновили дерево объектов, проверили наличие таблиц и ограничений, построили ER-диаграмму (Tools → ERD Tool).

**SQL-скрипт: создание таблиц**

-- Таблица категорий  
CREATE TABLE public.categories (  
 cat\_ID integer NOT NULL,  
 cat\_name text NOT NULL,  
 cat\_overcat integer,  
 CONSTRAINT categories\_PK PRIMARY KEY (cat\_ID)  
);  
  
-- Таблица производителей  
CREATE TABLE public.manufacturers (  
 man\_ID integer NOT NULL,  
 man\_name text NOT NULL,  
 man\_address text NOT NULL,  
 CONSTRAINT manufacturers\_PK PRIMARY KEY (man\_ID)  
);  
  
-- Таблица товаров  
CREATE TABLE public.products (  
 prod\_ID integer NOT NULL,  
 prod\_name text NOT NULL,  
 prod\_price numeric(8,2) NOT NULL,  
 prod\_rest integer NOT NULL,  
 prod\_cat\_ID integer NOT NULL,  
 prod\_man\_ID integer NOT NULL,  
 CONSTRAINT products\_PK PRIMARY KEY (prod\_ID)  
);  
  
-- Таблица покупателей  
CREATE TABLE public.customers (  
 cust\_ID integer NOT NULL,  
 cust\_name text NOT NULL,  
 cust\_address text NOT NULL,  
 cust\_discount integer NOT NULL,  
 CONSTRAINT customers\_PK PRIMARY KEY (cust\_ID)  
);  
  
-- Таблица сотрудников  
CREATE TABLE public.employees (  
 emp\_ID integer NOT NULL,  
 emp\_name text NOT NULL,  
 emp\_position text NOT NULL,  
 CONSTRAINT employees\_PK PRIMARY KEY (emp\_ID)  
);  
  
-- Таблица заказов  
CREATE TABLE public.orders (  
 ord\_ID integer NOT NULL,  
 ord\_cust\_ID integer NOT NULL,  
 ord\_emp\_ID integer NOT NULL,  
 ord\_date date NOT NULL,  
 CONSTRAINT orders\_PK PRIMARY KEY (ord\_ID)  
);  
  
-- Таблица позиций заказа  
CREATE TABLE public.items (  
 item\_ID integer NOT NULL,  
 item\_ord\_ID integer NOT NULL,  
 item\_prod\_ID integer NOT NULL,  
 item\_prod\_count integer NOT NULL,  
 CONSTRAINT items\_PK PRIMARY KEY (item\_ID)  
);

**SQL-скрипт: внешние ключи и каскады**

-- Иерархия категорий (self-reference)  
ALTER TABLE public.categories  
 ADD CONSTRAINT categories\_FK  
 FOREIGN KEY (cat\_overcat)  
 REFERENCES public.categories (cat\_ID)  
 ON DELETE CASCADE  
 ON UPDATE CASCADE;  
  
-- products → categories  
ALTER TABLE public.products  
 ADD CONSTRAINT products\_FK1  
 FOREIGN KEY (prod\_cat\_ID)  
 REFERENCES public.categories (cat\_ID)  
 ON DELETE CASCADE  
 ON UPDATE CASCADE;  
  
-- products → manufacturers  
ALTER TABLE public.products  
 ADD CONSTRAINT products\_FK2  
 FOREIGN KEY (prod\_man\_ID)  
 REFERENCES public.manufacturers (man\_ID)  
 ON DELETE CASCADE  
 ON UPDATE CASCADE;  
  
-- orders → customers  
ALTER TABLE public.orders  
 ADD CONSTRAINT orders\_FK1  
 FOREIGN KEY (ord\_cust\_ID)  
 REFERENCES public.customers (cust\_ID)  
 ON DELETE CASCADE  
 ON UPDATE CASCADE;  
  
-- orders → employees  
ALTER TABLE public.orders  
 ADD CONSTRAINT orders\_FK2  
 FOREIGN KEY (ord\_emp\_ID)  
 REFERENCES public.employees (emp\_ID)  
 ON DELETE CASCADE  
 ON UPDATE CASCADE;  
  
-- items → orders  
ALTER TABLE public.items  
 ADD CONSTRAINT items\_FK1  
 FOREIGN KEY (item\_ord\_ID)  
 REFERENCES public.orders (ord\_ID)  
 ON DELETE CASCADE  
 ON UPDATE CASCADE;  
  
-- items → products  
ALTER TABLE public.items  
 ADD CONSTRAINT items\_FK2  
 FOREIGN KEY (item\_prod\_ID)  
 REFERENCES public.products (prod\_ID)  
 ON DELETE CASCADE  
 ON UPDATE CASCADE;

**Пояснение ключевых решений:**

• Первичные ключи (PRIMARY KEY) заданы в CREATE TABLE, что автоматически создаёт уникальные индексы.

• Внешние ключи вынесены в ALTER TABLE: так легче читать DDL, менять и удалять отдельные ограничения.

• Self-reference в categories (cat\_overcat → cat\_ID) строит иерархию категорий; каскады поддерживают целостность.

• Для учебной задачи выбрана схема public, чтобы все объекты были в одном пространстве имён.

**Проверка результатов(рисунок 2** — **3):**

-- Таблицы в схеме public  
SELECT tablename  
FROM pg\_catalog.pg\_tables  
WHERE schemaname = 'public'  
ORDER BY tablename;  
  
-- Внешние ключи  
SELECT conname,  
 conrelid::regclass AS table\_name,  
 confrelid::regclass AS ref\_table  
FROM pg\_constraint  
WHERE contype = 'f'  
ORDER BY conname;

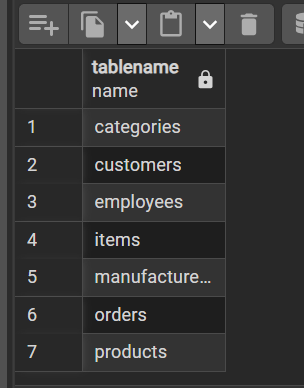


Рисунок 2 — Проверка таблиц

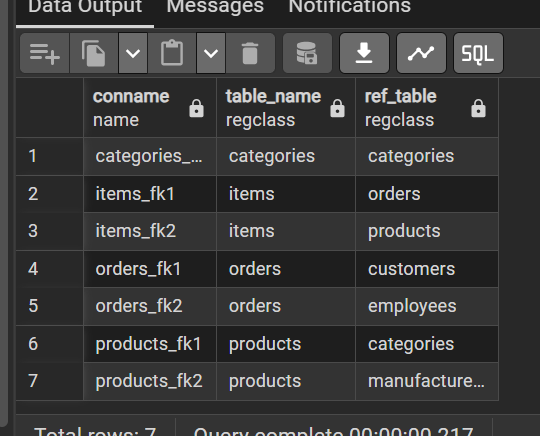


Рисунок 3 — Проверка внешних ключей

**ER-диаграмма в pgAdmin:**

Tools → ERD Tool → ПКМ на базе sales → ERD for Database ( рисунок 4).

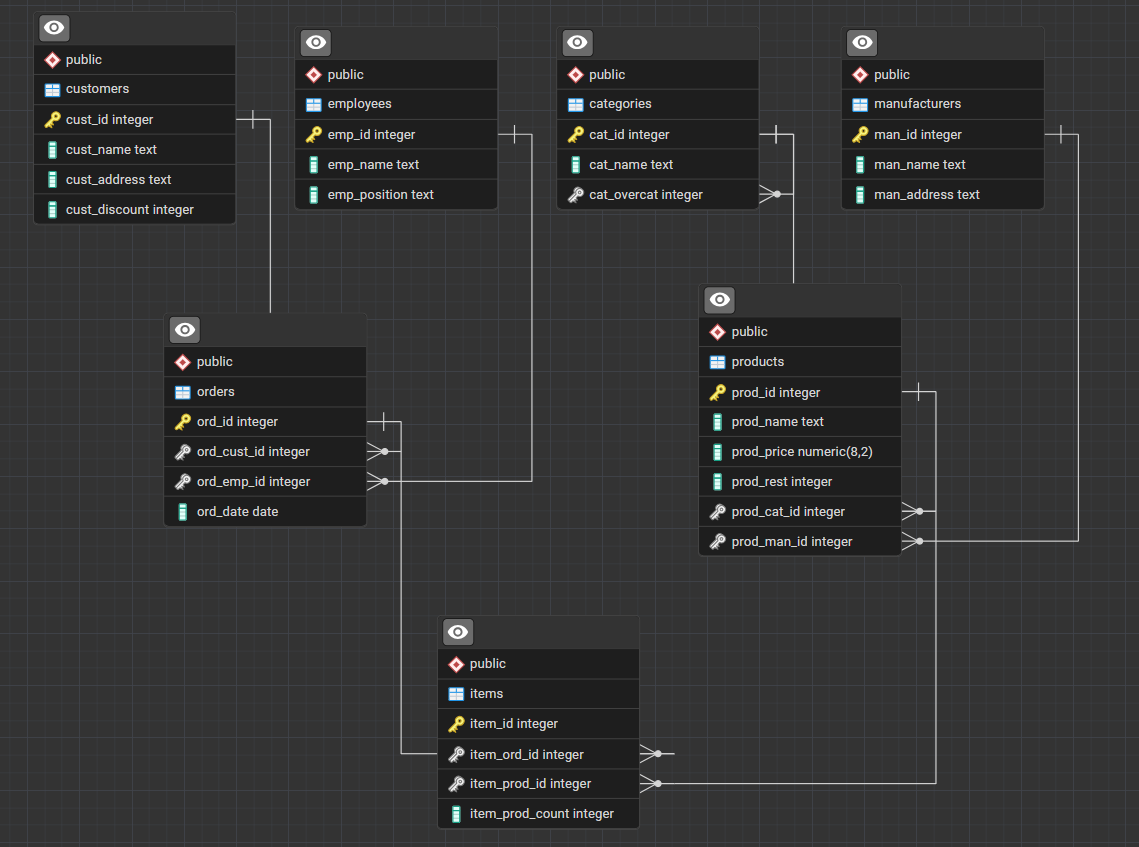


Рисунок 4 — ER-диаграмма

**Заключение:**

Создана схема базы данных sales: определены 7 таблиц, заданы первичные ключи и обязательность полей, добавлены внешние ключи с каскадными правилами. Результаты подтверждены проверочными запросами и ER-диаграммой.

**Цель работы:**

Спроектировать и реализовать новую базу данных в PostgreSQL по индивидуальному варианту: создать БД, задать таблицы и типы полей, реализовать связи и ограничения целостности, получить схему БД и подготовить отчёт с результатами.

**Описание предметной области (кратко):**

Учёт заявок, чеков и накладных на товары. Сущности: сотрудники, поставщики, каталог товаров, заявки, состав заявок, чеки, состав чеков, накладные, состав накладных. Между сущностями реализованы связи 1:M.

**Описание целостности:**

• По сущностям (первичные ключи):

employees(employee\_id),

suppliers(supplier\_code),

product\_catalog(product\_code),

orders(order\_number),

receipts(receipt\_number),

invoices(invoice\_number),

order\_composition(order\_number, product\_code),

check\_composition(receipt\_number, product\_code),

invoice\_composition(invoice\_number, product\_code).

• Семантическая целостность:

— даты и моменты времени хранятся в типах date/timestamp; — наименования и адреса — text; — цены — numeric(10,2); — идентификаторы — integer; — для составных таблиц первичный ключ составной, исключающий дубликаты позиций.

• Ссылочная целостность:

Все связи реализованы внешними ключами с каскадными действиями ON DELETE/UPDATE CASCADE для поддержания согласованности данных при изменениях.

**Шаг 1. Создание базы данных (рисунок 5):**

-- выполнить в подключении к postgres  
DROP DATABASE IF EXISTS rgr\_store;  
  
CREATE DATABASE rgr\_store

WITH

OWNER = postgres

ENCODING = 'UTF8'

TEMPLATE = template0

LC\_COLLATE = 'Russian\_Russia.1251'

LC\_CTYPE = 'Russian\_Russia.1251';  
  
COMMENT ON DATABASE rgr\_store  
 IS 'RGR: база данных для оформления заказов/чеков/накладных';

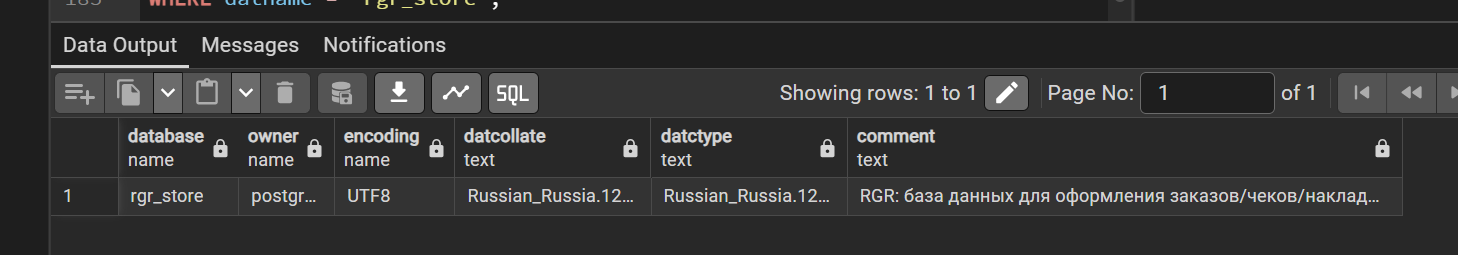


Рисунок 5 — Результат создания бд

**Шаг 2. Таблицы (PK/NOT NULL внутри CREATE TABLE) (рисунок 6):**

-- выполнять уже в базе rgr\_store  
  
-- При пересоздании  
DROP TABLE IF EXISTS invoice\_composition CASCADE;  
DROP TABLE IF EXISTS invoices CASCADE;  
DROP TABLE IF EXISTS check\_composition CASCADE;  
DROP TABLE IF EXISTS receipts CASCADE;  
DROP TABLE IF EXISTS order\_composition CASCADE;  
DROP TABLE IF EXISTS orders CASCADE;  
DROP TABLE IF EXISTS product\_catalog CASCADE;  
DROP TABLE IF EXISTS employees CASCADE;  
DROP TABLE IF EXISTS suppliers CASCADE;  
  
-- Сотрудники  
CREATE TABLE public.employees (  
 employee\_id integer PRIMARY KEY,  
 emp\_name text NOT NULL,  
 emp\_position text NOT NULL  
);  
  
-- Поставщики  
CREATE TABLE public.suppliers (  
 supplier\_code integer PRIMARY KEY,  
 supplier\_name text NOT NULL  
);  
  
-- Каталог товаров  
CREATE TABLE public.product\_catalog (  
 product\_code integer PRIMARY KEY,  
 product\_name text NOT NULL,  
 product\_category text NOT NULL,  
 product\_price numeric(10,2) NOT NULL  
);  
  
-- Заявки  
CREATE TABLE public.orders (  
 order\_number integer PRIMARY KEY,  
 order\_date date NOT NULL  
);  
  
-- Состав заявки  
CREATE TABLE public.order\_composition (  
 order\_number integer NOT NULL,  
 product\_code integer NOT NULL,  
 product\_quantity integer NOT NULL,  
 CONSTRAINT order\_composition\_pk PRIMARY KEY (order\_number, product\_code)  
);  
  
-- Чеки  
CREATE TABLE public.receipts (  
 receipt\_number integer PRIMARY KEY,  
 receipt\_date timestamp NOT NULL,  
 cash\_register\_number integer NOT NULL,  
 employee\_id integer NOT NULL  
);  
  
-- Состав чека  
CREATE TABLE public.check\_composition (  
 receipt\_number integer NOT NULL,  
 product\_code integer NOT NULL,  
 quantity\_in\_check integer NOT NULL,  
 CONSTRAINT check\_composition\_pk PRIMARY KEY (receipt\_number, product\_code)  
);  
  
-- Накладные  
CREATE TABLE public.invoices (  
 invoice\_number integer PRIMARY KEY,  
 invoice\_date date NOT NULL,  
 supplier\_code integer NOT NULL  
);  
  
-- Состав накладной  
CREATE TABLE public.invoice\_composition (  
 invoice\_number integer NOT NULL,  
 product\_code integer NOT NULL,  
 product\_quantity integer NOT NULL,  
 CONSTRAINT invoice\_composition\_pk PRIMARY KEY (invoice\_number, product\_code)  
);

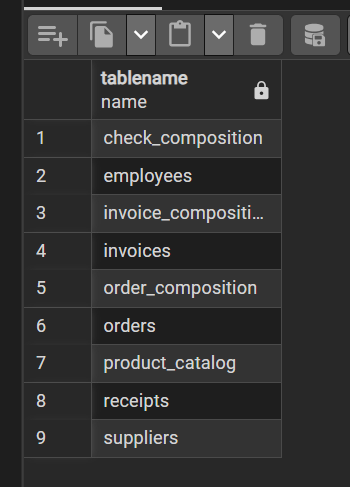


Рисунок 6 — Созданные таблицы

**Шаг 3. Связи и ограничения целостности (FK) (рисунок 7):**

-- order\_composition → orders / product\_catalog  
ALTER TABLE public.order\_composition  
 ADD CONSTRAINT oc\_fk\_order  
 FOREIGN KEY (order\_number)  
 REFERENCES public.orders (order\_number)  
 ON DELETE CASCADE ON UPDATE CASCADE;  
  
ALTER TABLE public.order\_composition  
 ADD CONSTRAINT oc\_fk\_product  
 FOREIGN KEY (product\_code)  
 REFERENCES public.product\_catalog (product\_code)  
 ON DELETE CASCADE ON UPDATE CASCADE;  
  
-- receipts → employees  
ALTER TABLE public.receipts  
 ADD CONSTRAINT receipts\_fk\_employee  
 FOREIGN KEY (employee\_id)  
 REFERENCES public.employees (employee\_id)  
 ON DELETE CASCADE ON UPDATE CASCADE;  
  
-- check\_composition → receipts / product\_catalog  
ALTER TABLE public.check\_composition  
 ADD CONSTRAINT cc\_fk\_receipt  
 FOREIGN KEY (receipt\_number)  
 REFERENCES public.receipts (receipt\_number)  
 ON DELETE CASCADE ON UPDATE CASCADE;  
  
ALTER TABLE public.check\_composition  
 ADD CONSTRAINT cc\_fk\_product  
 FOREIGN KEY (product\_code)  
 REFERENCES public.product\_catalog (product\_code)  
 ON DELETE CASCADE ON UPDATE CASCADE;  
  
-- invoices → suppliers  
ALTER TABLE public.invoices  
 ADD CONSTRAINT invoices\_fk\_supplier  
 FOREIGN KEY (supplier\_code)  
 REFERENCES public.suppliers (supplier\_code)  
 ON DELETE CASCADE ON UPDATE CASCADE;  
  
-- invoice\_composition → invoices / product\_catalog  
ALTER TABLE public.invoice\_composition  
 ADD CONSTRAINT ic\_fk\_invoice  
 FOREIGN KEY (invoice\_number)  
 REFERENCES public.invoices (invoice\_number)  
 ON DELETE CASCADE ON UPDATE CASCADE;  
  
ALTER TABLE public.invoice\_composition  
 ADD CONSTRAINT ic\_fk\_product  
 FOREIGN KEY (product\_code)  
 REFERENCES public.product\_catalog (product\_code)  
 ON DELETE CASCADE ON UPDATE CASCADE;

Проверка результата (вставить скриншоты вывода)

-- Таблицы в public  
SELECT tablename  
FROM pg\_catalog.pg\_tables  
WHERE schemaname='public'  
ORDER BY tablename;  
  
-- Все внешние ключи  
SELECT conname,  
 conrelid::regclass AS table\_name,  
 confrelid::regclass AS ref\_table  
FROM pg\_constraint  
WHERE contype='f'  
ORDER BY conname;

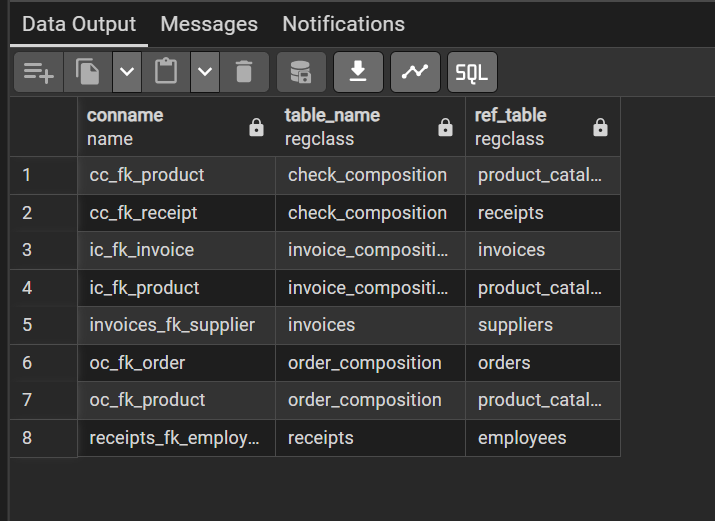


Рисунок 7 — Созданные связи

**Схема БД (ERD) (рисунок 8):**

Сформировать в pgAdmin: Tools → ERD Tool → ПКМ по rgr\_store → ERD for Database.

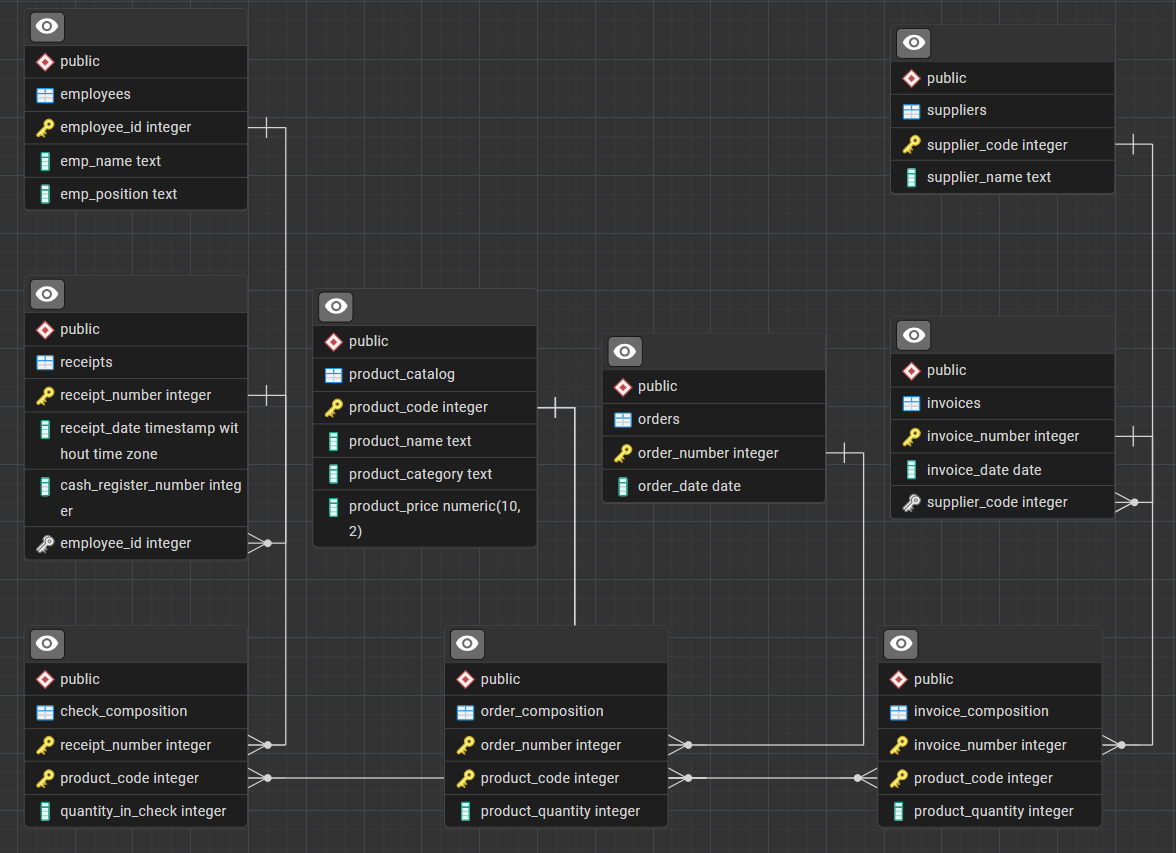


Рисунок 8 — Схема БД

**Заключение:**

Создана новая БД rgr\_store. Реализованы таблицы и связи предметной области, заданы первичные ключи, внешние ключи с каскадными действиями, обеспечены семантическая и ссылочная целостность. Результаты подтверждены выполнимыми скриптами и проверочными выборками.