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 №3** |
| по дисциплине **Системы управления базами данных**  **Тема: ВСТАВКА, УДАЛЕНИЕ И ОБНОВЛЕНИЕ ДАННЫХ** |

|  |  |  |  |
| --- | --- | --- | --- |
| 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 | 15.10 | | |
| Проверил |  | | |

Омск 2025

**Цель работы:**

Заполнить данными таблицы новой базы данных rgr\_store и проверить выполнение всех определённых ограничений целостности (NOT NULL, PRIMARY KEY/составные PK, FOREIGN KEY с ON DELETE/UPDATE CASCADE). Результаты представить скриптами SQL и скриншотами.

**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/время — date/timestamp; наименования и адреса — text; цены — numeric(10,2); идентификаторы — integer; для составных таблиц PK составной.

• Ссылочная целостность:  
все связи реализованы внешними ключами с каскадными действиями ON DELETE/UPDATE CASCADE.

**Шаг 1. Заполнение БД тестовыми данными:**

1. Очищаем таблицы командой TRUNCATE … CASCADE (чтобы повторные прогоны не конфликтовали с уже существующими строками).
2. Заполняем сначала «родительские» справочники (employees, suppliers, product\_catalog), затем «дочерние» документы и их состав (orders → order\_composition; receipts → check\_composition; invoices → invoice\_composition). Такой порядок соблюдает внешние ключи.

**SQL запрос:**

-- Очистка (при повторе)

TRUNCATE TABLE

invoice\_composition,

invoices,

check\_composition,

receipts,

order\_composition,

orders,

product\_catalog,

employees,

suppliers

RESTART IDENTITY CASCADE;

-- Сотрудникик,поставщики,продукты

INSERT INTO employees (employee\_id, emp\_name, emp\_position) VALUES

(1,'Иванова А.А.','кассир'),

(2,'Петров П.П.','менеджер'),

(3,'Сидорова С.С.','кладовщик');

INSERT INTO suppliers (supplier\_code, supplier\_name) VALUES

(10,'ООО «ФруТорг»'),

(20,'АО «Продукты+»');

INSERT INTO product\_catalog (product\_code, product\_name, product\_category, product\_price) VALUES

(100,'Яблоко','Фрукты',35.50),

(101,'Банан','Фрукты',49.90),

(200,'Молоко 1л','Молочные',79.00),

(300,'Хлеб батон','Хлеб',29.00);

-- Заявки и состав

INSERT INTO orders (order\_number, order\_date) VALUES

(5001, DATE '2025-03-01'),

(5002, DATE '2025-03-02');

INSERT INTO order\_composition (order\_number, product\_code, product\_quantity) VALUES

(5001,100,20),

(5001,200,10),

(5002,101,15),

(5002,300,25);

-- Чеки и состав

INSERT INTO receipts (receipt\_number, receipt\_date, cash\_register\_number, employee\_id) VALUES

(70001, TIMESTAMP '2025-03-05 10:15:00', 1, 1),

(70002, TIMESTAMP '2025-03-05 11:20:00', 1, 1);

INSERT INTO check\_composition (receipt\_number, product\_code, quantity\_in\_check) VALUES

(70001,100,3),

(70001,300,1),

(70002,200,2),

(70002,101,4);

-- Накладные и состав

INSERT INTO invoices (invoice\_number, invoice\_date, supplier\_code) VALUES

(9001, DATE '2025-03-03', 10),

(9002, DATE '2025-03-04', 20);

INSERT INTO invoice\_composition (invoice\_number, product\_code, product\_quantity) VALUES

(9001,100,100),

(9001,200,60),

(9002,101,80),

(9002,300,120);

# Шаг 2. Контрольные выборки после заполнения (рисунок 1 - 9)

**Что проверяем:** содержимое каждой таблицы; соответствие ключей и связей предметной области.

**SQL запрос:**

SELECT \* FROM employees ORDER BY employee\_id;

SELECT \* FROM suppliers ORDER BY supplier\_code;

SELECT \* FROM product\_catalog ORDER BY product\_code;

SELECT \* FROM orders ORDER BY order\_number;

SELECT \* FROM order\_composition ORDER BY order\_number, product\_code;

SELECT \* FROM receipts ORDER BY receipt\_number;

SELECT \* FROM check\_composition ORDER BY receipt\_number, product\_code;

SELECT \* FROM invoices ORDER BY invoice\_number;

SELECT \* FROM invoice\_composition ORDER BY invoice\_number, product\_code;

# 

# Рисунок 1 — Результат заполнения таблицы сотрудников

# 

# Рисунок 2 — Результат заполнения таблицы поставщиков

# 

# Рисунок 3 — Результат заполнения таблицы каталог товаров

# 

# Рисунок 4 — Результат заполнения таблицы заявок

# 

# Рисунок 5 — Результат заполнения таблицы состава заявок

# 

# Рисунок 6 — Результат заполнения таблицы чеков

# 

# Рисунок 7 — Результат заполнения таблицы состава чеков

# 

# Рисунок 8 — Результат заполнения таблицы накладных

# 

# Рисунок 9 — Результат заполнения таблицы состава накладных

# Шаг 3. Проверка ограничений целостности (негативные тесты) (рисунок 10 - 14)

Каждый запрос должен упасть с ошибкой — значит, ограничение работает.

**SQL запрос:**

BEGIN; -- <<< открыли явную транзакцию

-----------------------------

-- 1. CHECK и NOT NULL

-----------------------------

-- 1.1 employees: emp\_name не пустой (NOT NULL + not blank)

SAVEPOINT s1;

INSERT INTO public.employees(employee\_id, emp\_name, emp\_position)

VALUES (9999, NULL, 'кассир'); -- ожидаем ERROR: NOT NULL

ROLLBACK TO SAVEPOINT s1;

SAVEPOINT s1b;

INSERT INTO public.employees(employee\_id, emp\_name, emp\_position)

VALUES (9998, ' ', 'кассир'); -- ожидаем ERROR: employees\_name\_notblank

ROLLBACK TO SAVEPOINT s1b;

-- 1.2 employees: положительный ID

SAVEPOINT s2;

INSERT INTO public.employees(employee\_id, emp\_name, emp\_position)

VALUES (-1, 'Иванов', 'кассир'); -- ожидаем ERROR: employees\_id\_pos\_chk

ROLLBACK TO SAVEPOINT s2;

-- 1.3 suppliers: имя не пустое, код > 0

SAVEPOINT s3;

INSERT INTO public.suppliers(supplier\_code, supplier\_name)

VALUES (0, 'ООО Тест'); -- ожидаем ERROR: suppliers\_code\_pos\_chk

ROLLBACK TO SAVEPOINT s3;

SAVEPOINT s3b;

INSERT INTO public.suppliers(supplier\_code, supplier\_name)

VALUES (999, ' '); -- ожидаем ERROR: suppliers\_name\_notblank

ROLLBACK TO SAVEPOINT s3b;

-- 1.4 product\_catalog: цена неотрицательная, имя/категория не пустые, код > 0

SAVEPOINT s4;

INSERT INTO public.product\_catalog(product\_code, product\_name, product\_category, product\_price)

VALUES (100, 'Дубль', 'Разное', -1.00); -- PK-дубль + ERROR: products\_price\_nonneg\_chk

ROLLBACK TO SAVEPOINT s4;

SAVEPOINT s4b;

INSERT INTO public.product\_catalog(product\_code, product\_name, product\_category, product\_price)

VALUES (9999, ' ', 'Фрукты', 10); -- ERROR: products\_name\_notblank

ROLLBACK TO SAVEPOINT s4b;

-- 1.5 orders: номер > 0, дата не из будущего

SAVEPOINT s5;

INSERT INTO public.orders(order\_number, order\_date)

VALUES (0, CURRENT\_DATE); -- ERROR: orders\_num\_pos\_chk

ROLLBACK TO SAVEPOINT s5;

SAVEPOINT s5b;

INSERT INTO public.orders(order\_number, order\_date)

VALUES (9999, CURRENT\_DATE + INTERVAL '1 day'); -- ERROR: orders\_date\_past\_chk

ROLLBACK TO SAVEPOINT s5b;

-- 1.6 order\_composition: количество > 0

SAVEPOINT s6;

INSERT INTO public.order\_composition(order\_number, product\_code, product\_quantity)

VALUES (5001, 100, 0); -- ERROR: oc\_qty\_pos\_chk

ROLLBACK TO SAVEPOINT s6;

-- 1.7 receipts: номер/касса > 0, дата не из будущего

SAVEPOINT s7;

INSERT INTO public.receipts(receipt\_number, receipt\_date, cash\_register\_number, employee\_id)

VALUES (0, now(), 1, 1); -- ERROR: receipts\_num\_pos\_chk

ROLLBACK TO SAVEPOINT s7;

SAVEPOINT s7b;

INSERT INTO public.receipts(receipt\_number, receipt\_date, cash\_register\_number, employee\_id)

VALUES (70099, now() + INTERVAL '1 day', 1, 1); -- ERROR: receipts\_date\_past\_chk

ROLLBACK TO SAVEPOINT s7b;

SAVEPOINT s7c;

INSERT INTO public.receipts(receipt\_number, receipt\_date, cash\_register\_number, employee\_id)

VALUES (70098, now(), 0, 1); -- ERROR: receipts\_register\_pos\_chk

ROLLBACK TO SAVEPOINT s7c;

-- 1.8 check\_composition: количество > 0

SAVEPOINT s8;

INSERT INTO public.check\_composition(receipt\_number, product\_code, quantity\_in\_check)

VALUES (70001, 100, 0); -- ERROR: cc\_qty\_pos\_chk

ROLLBACK TO SAVEPOINT s8;

-- 1.9 invoices: номер > 0, дата не из будущего

SAVEPOINT s9;

INSERT INTO public.invoices(invoice\_number, invoice\_date, supplier\_code)

VALUES (0, CURRENT\_DATE, 10); -- ERROR: invoices\_num\_pos\_chk

ROLLBACK TO SAVEPOINT s9;

SAVEPOINT s9b;

INSERT INTO public.invoices(invoice\_number, invoice\_date, supplier\_code)

VALUES (9999, CURRENT\_DATE + 1, 10); -- ERROR: invoices\_date\_past\_chk

ROLLBACK TO SAVEPOINT s9b;

-- 1.10 invoice\_composition: количество > 0

SAVEPOINT s10;

INSERT INTO public.invoice\_composition(invoice\_number, product\_code, product\_quantity)

VALUES (9001, 100, 0); -- ERROR: ic\_qty\_pos\_chk

ROLLBACK TO SAVEPOINT s10;

-----------------------------

-- 2. PRIMARY KEY (уникальность)

-----------------------------

-- Дубликат PK в products (id 100 уже есть)

SAVEPOINT s11;

INSERT INTO public.product\_catalog(product\_code, product\_name, product\_category, product\_price)

VALUES (100, 'Дубль', 'Разное', 10.00); -- ERROR: duplicate key value ... pkey

ROLLBACK TO SAVEPOINT s11;

-- Составной PK: повтор товара в той же заявке/чеке/накладной

SAVEPOINT s12;

INSERT INTO public.order\_composition(order\_number, product\_code, product\_quantity)

VALUES (5001, 100, 1); -- ERROR: order\_composition\_pk

ROLLBACK TO SAVEPOINT s12;

SAVEPOINT s12b;

INSERT INTO public.check\_composition(receipt\_number, product\_code, quantity\_in\_check)

VALUES (70001, 100, 1); -- ERROR: check\_composition\_pk

ROLLBACK TO SAVEPOINT s12b;

SAVEPOINT s12c;

INSERT INTO public.invoice\_composition(invoice\_number, product\_code, product\_quantity)

VALUES (9001, 100, 1); -- ERROR: invoice\_composition\_pk

ROLLBACK TO SAVEPOINT s12c;

-----------------------------

-- 3. FOREIGN KEY (ссылочная целостность)

-----------------------------

-- Несуществующий сотрудник в чеке

SAVEPOINT s13;

INSERT INTO public.receipts(receipt\_number, receipt\_date, cash\_register\_number, employee\_id)

VALUES (70111, now(), 1, 999); -- ERROR: receipts\_fk\_employee

ROLLBACK TO SAVEPOINT s13;

-- Несуществующий товар в составе заявки

SAVEPOINT s14;

INSERT INTO public.order\_composition(order\_number, product\_code, product\_quantity)

VALUES (5001, 9999, 1); -- ERROR: oc\_fk\_product

ROLLBACK TO SAVEPOINT s14;

-- Несуществующий чек в составе чека

SAVEPOINT s15;

INSERT INTO public.check\_composition(receipt\_number, product\_code, quantity\_in\_check)

VALUES (79999, 100, 1); -- ERROR: cc\_fk\_receipt

ROLLBACK TO SAVEPOINT s15;

-- Несуществующий поставщик в накладной

SAVEPOINT s16;

INSERT INTO public.invoices(invoice\_number, invoice\_date, supplier\_code)

VALUES (90111, CURRENT\_DATE, 999); -- ERROR: invoices\_fk\_supplier

ROLLBACK TO SAVEPOINT s16;

Скриншоты ошибок:

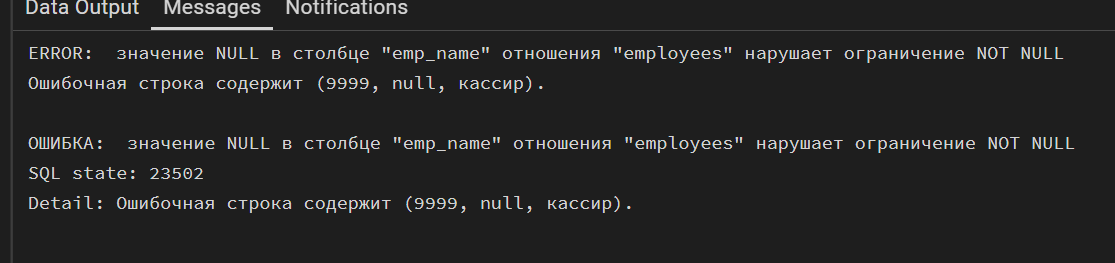


Рисунок 10 — Проверка ограничения NOT NULL

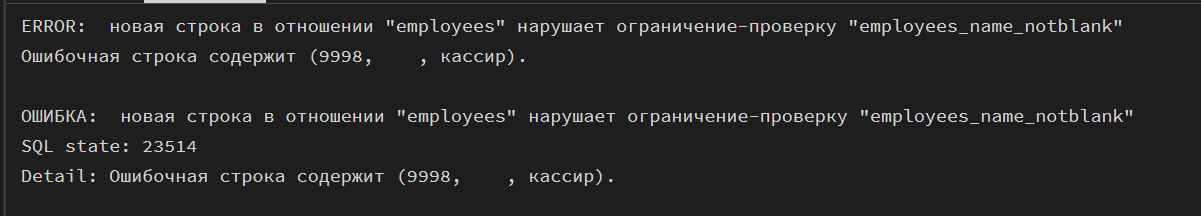


Рисунок 11 — Проверка ограничения установленных через CHECK в сотрудниках на пустое имя

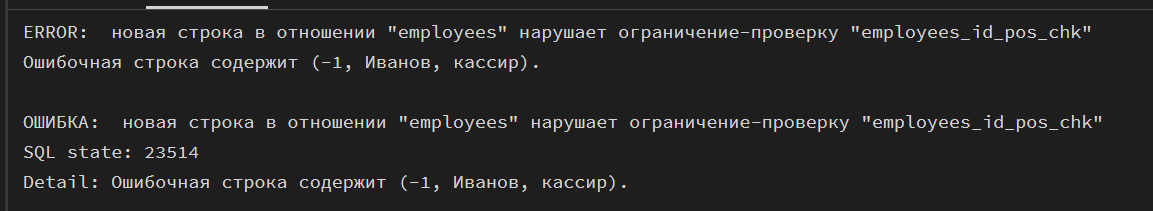


Рисунок 12 — Проверка ограничения установленного через CHECK в сотрудниках на отрицательный индекс

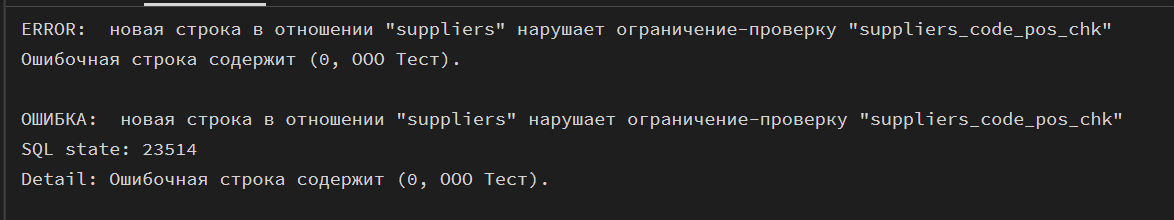


Рисунок 13 — Проверка ограничения установленного через CHECK в поставщиках на индекс равный 0

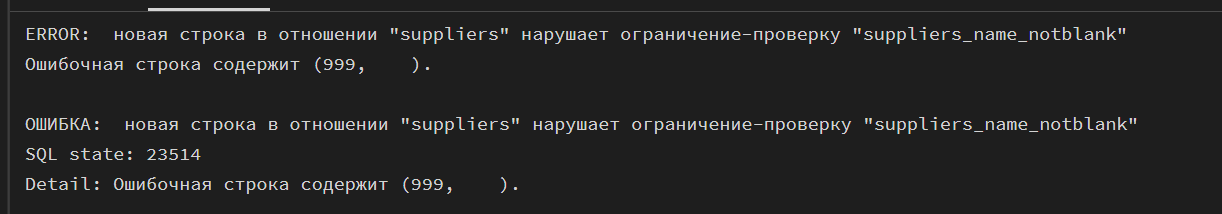


Рисунок 14 — Проверка ограничения установленного через CHECK в поставщиках на пустое имя

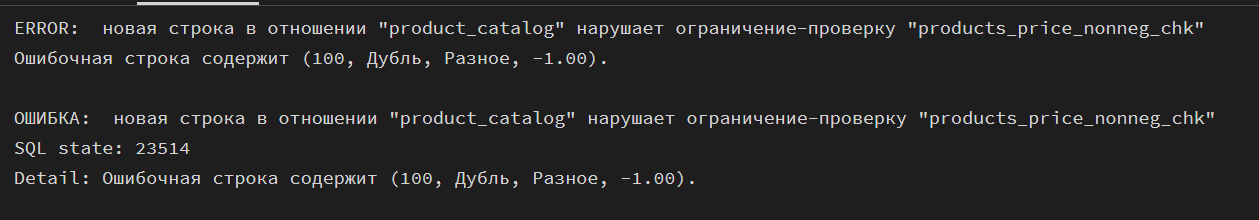


Рисунок 15 — Проверка ограничения установленного через CHECK в товарах на уникальность PK и отрицательную цену

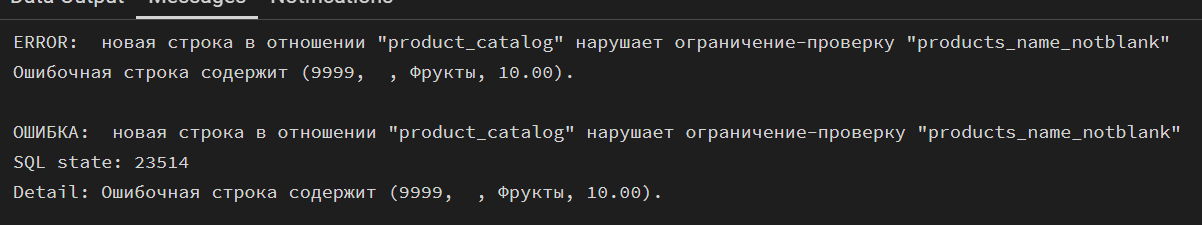


Рисунок 16 — Проверка ограничения установленного через CHECK в товарах на пустое имя

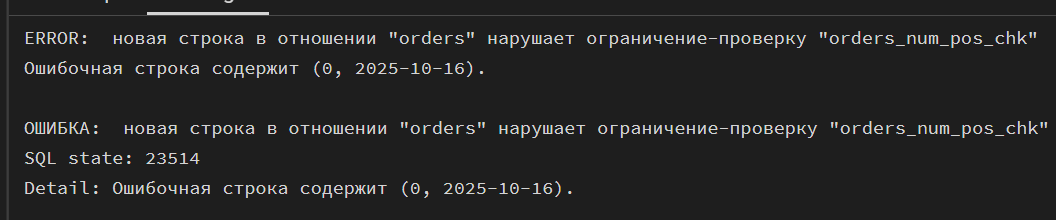


Рисунок 17 — Проверка ограничения установленного через CHECK в заявках на нулевой индекс

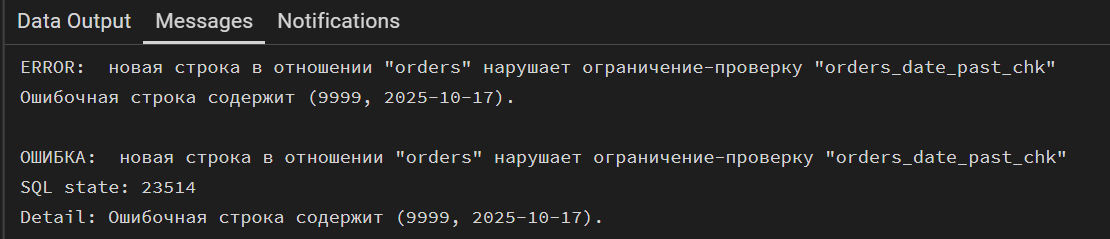


Рисунок 18 — Проверка ограничения установленного через CHECK в заявках на дату из будущего

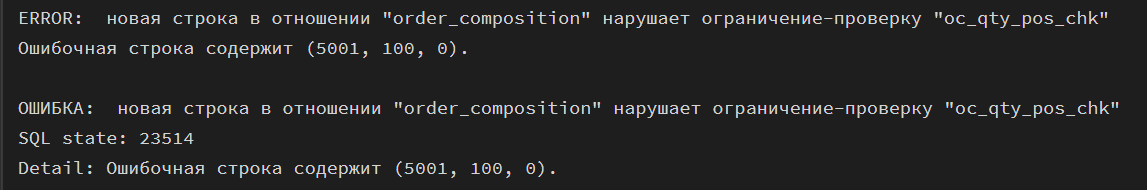


Рисунок 19 — Проверка ограничения установленного через CHECK в составе заявок на количество товаров

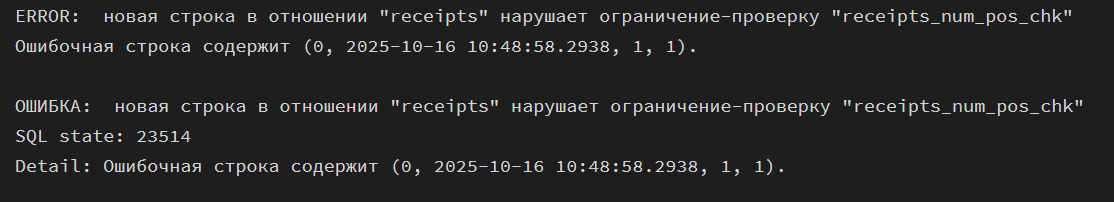


Рисунок 20 — Проверка ограничения установленного через CHECK в чеках на номер кассы

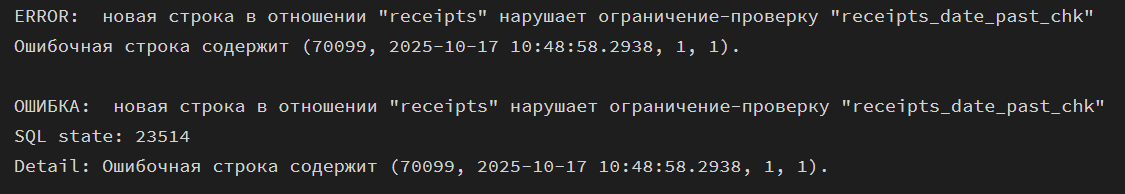


Рисунок 21 — Проверка ограничения установленного через CHECK в чеках на дату из будущего

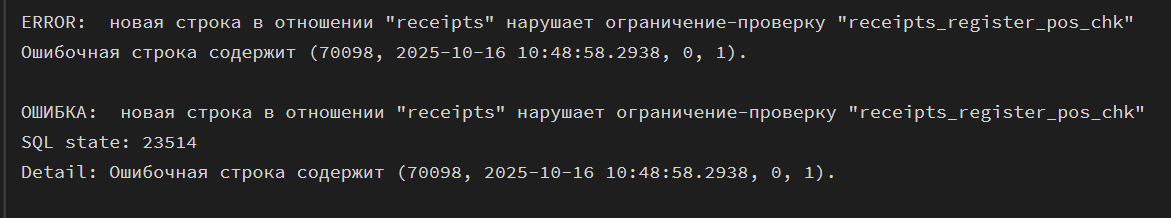


Рисунок 22 — Проверка ограничения установленного через CHECK в чеках на положительный номер кассы

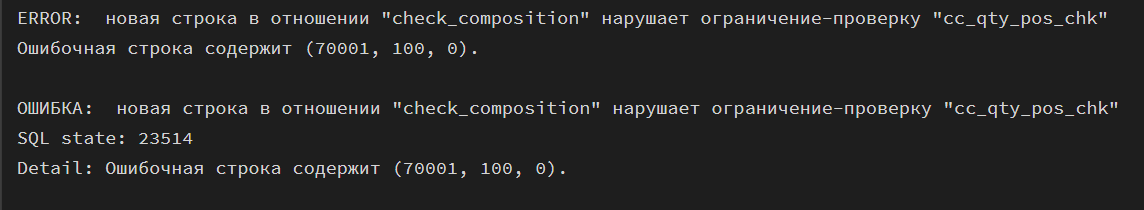


Рисунок 23 — Проверка ограничения установленного через CHECK в составе чеков на положительное количество товара

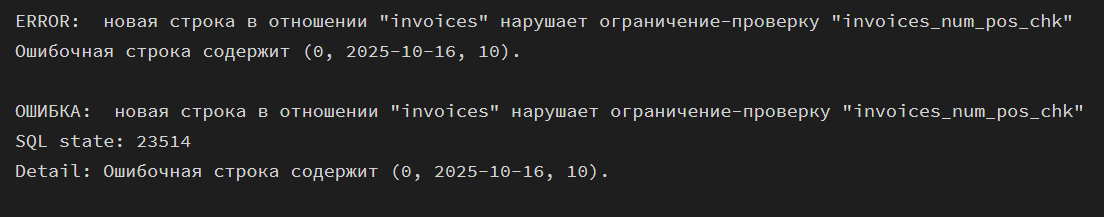


Рисунок 24 — Проверка ограничения установленного через CHECK в накладных на номер накладной

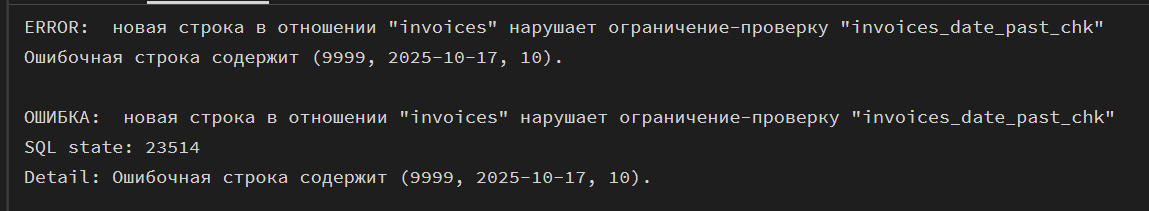


Рисунок 25 — Проверка ограничения установленного через CHECK в накладных на дату не из будущего

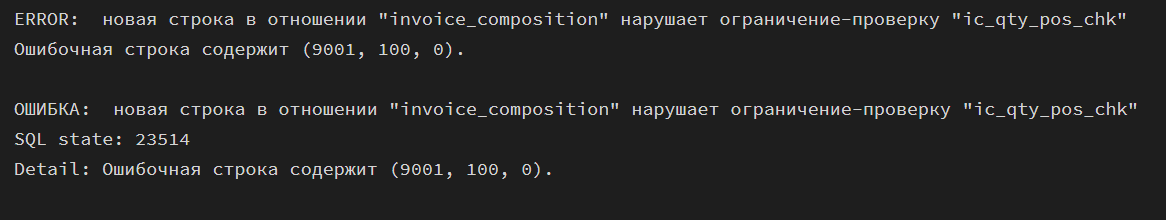


Рисунок 26 — Проверка ограничения установленного через CHECK в составе накладной на количество товара

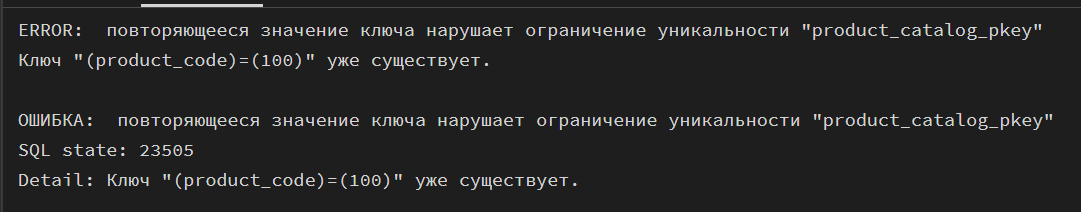


Рисунок 27 — Проверка ограничения уникальности PK в каталоге товаров

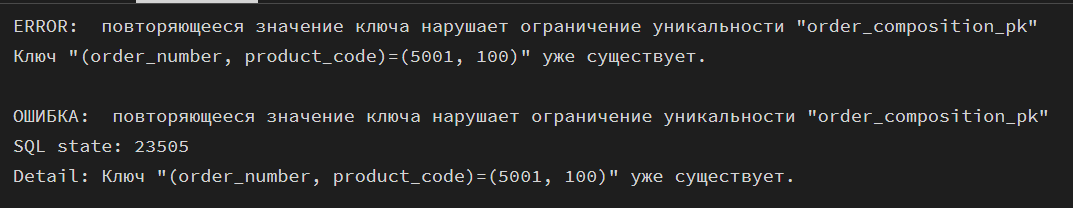


Рисунок 28 — Проверка ограничения уникальности PK в составе заявок

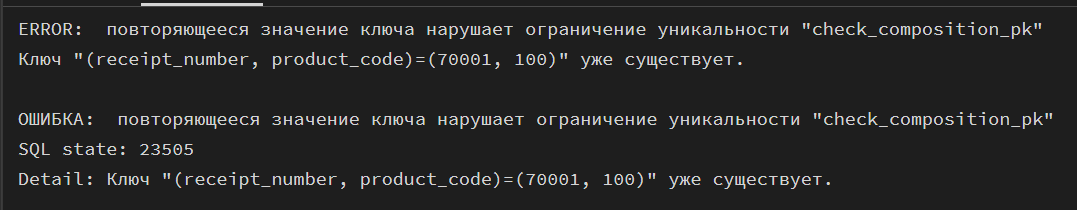


Рисунок 29 — Проверка ограничения уникальности PK в составе чеков

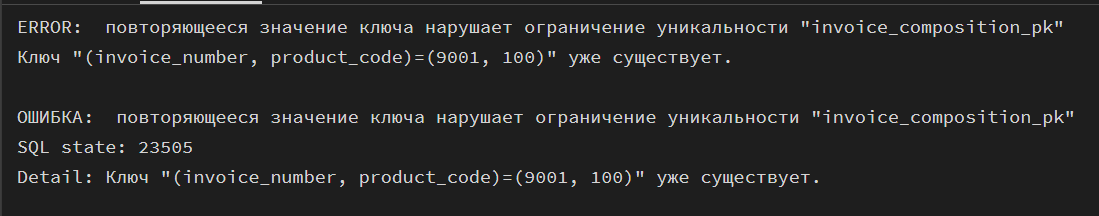
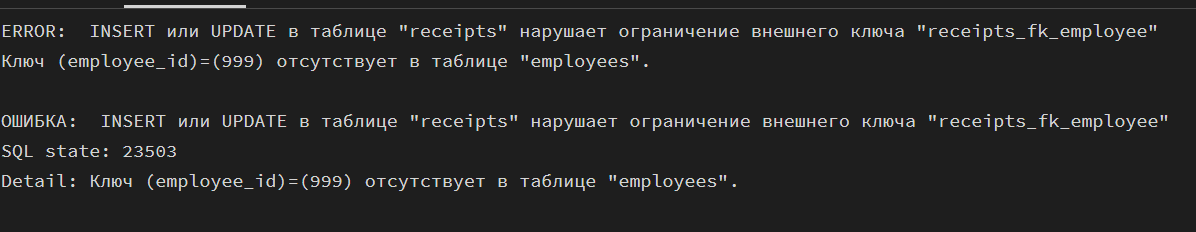


Рисунок 30 — Проверка ограничения уникальности PK в составе накладной

Рисунок 31 — Проверка несуществующего сотрудника в чеках

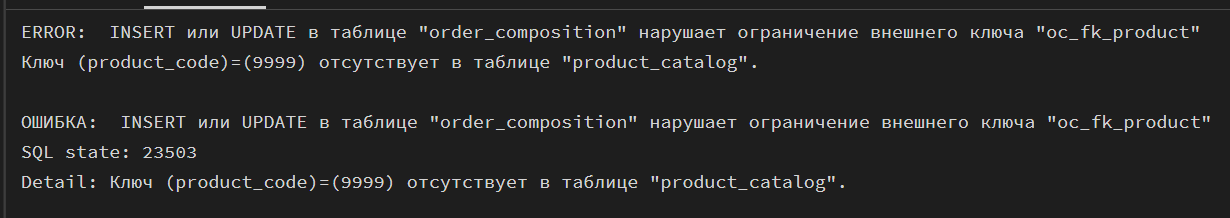


Рисунок 32 — Проверка несуществующего товара в составе заявки

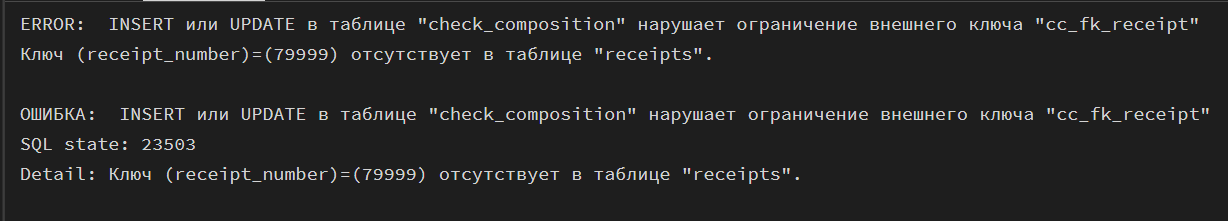


Рисунок 33 — Проверка несуществующего чека в составе чеков

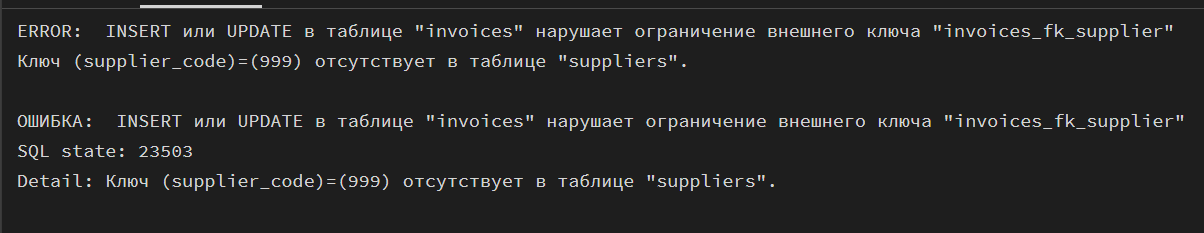


Рисунок 34 — Проверка несуществующего поставщика в составе накладной

# Шаг 4. Проверка каскадного удаления и обновления (рисунок 35 - 36)

**SQL запрос:**

-- 4.1 ON DELETE CASCADE: удаляем товар 300 → записи из check\_/invoice\_composition по нему должны исчезнуть

SAVEPOINT s17;

-- покажем, что строки по 300 есть:

SELECT 'before', count(\*) FROM public.check\_composition WHERE product\_code = 300;

SELECT 'before', count(\*) FROM public.invoice\_composition WHERE product\_code = 300;

DELETE FROM public.product\_catalog WHERE product\_code = 300; -- каскад на составы

SELECT 'after', count(\*) FROM public.check\_composition WHERE product\_code = 300; -- ожидается 0

SELECT 'after', count(\*) FROM public.invoice\_composition WHERE product\_code = 300; -- ожидается 0

ROLLBACK TO SAVEPOINT s17;

-- 4.2 ON UPDATE CASCADE: поменяем номер заявки → дочерние строки тоже обновятся

SAVEPOINT s18;

-- подготовим временную заявку и её строку

INSERT INTO public.orders(order\_number, order\_date) VALUES (6000, CURRENT\_DATE);

INSERT INTO public.order\_composition(order\_number, product\_code, product\_quantity)

VALUES (6000, 100, 1);

UPDATE public.orders SET order\_number = 7000 WHERE order\_number = 6000; -- каскад на order\_composition

-- дочерняя строка должна быть уже с 7000

SELECT \* FROM public.order\_composition WHERE order\_number = 7000 AND product\_code = 100;

ROLLBACK TO SAVEPOINT s18;

ROLLBACK;

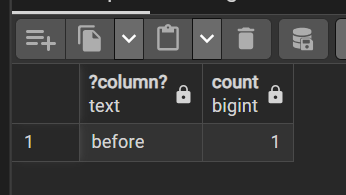
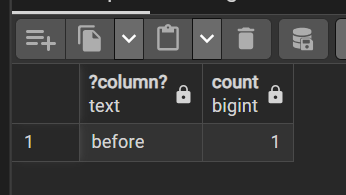


Рисунок 35 — До каскадного удаления в таблицах состав чека и состав накладной

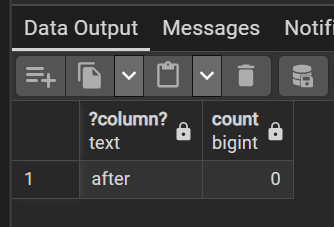
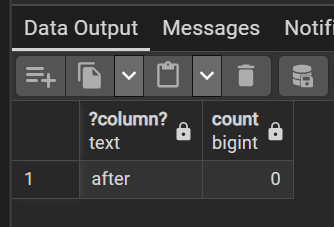


Рисунок 36 — Результат удаления продукта с кодом 300 в таблицах состав накладной и состав чека

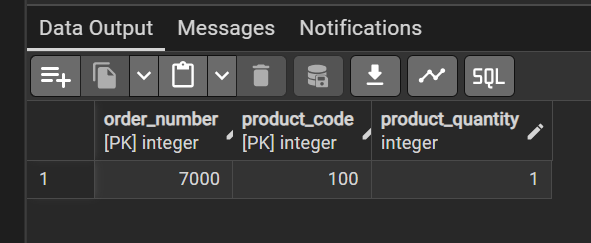


Рисунок 36 — Результат в таблице состав заявки после изменения номера заявки в таблице заявок

# Заключение:

Таблицы БД rgr\_store заполнены согласованными тестовыми данными. Ограничения целостности работают корректно: **NOT NULL** предотвращает пропуски, **PK/составные PK** исключают дубликаты, **FK** обеспечивает ссылочную целостность, **ON DELETE/UPDATE CASCADE** корректно поддерживает зависимые записи. Результаты подтверждены скриншотами выполнения скриптов, выборок и ожидаемых ошибок.