

Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт космических и информационных технологий
Кафедра «Программная инженерия»

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №2

по предмету “Серверное программирование”

Разработка архитектуры базы данных

тема

Преподаватель

А.А. Даничев

подпись, дата

инициалы, фамилия

Студент ЗКИ21-16БВВ 031625881

01.04.2025

Е.М.Хорошко

подпись, дата

инициалы, фамилия

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
1. Анализ схемы данных.....	4
1.1 Описание таблицы и ключевых видов связей.....	4
1.2 SQL SELECT JOIN запросы.....	6
2. Заполнение таблиц справочников.....	7
3. Генерация тестовых данных с помощью библиотеки Faker.....	7
4. Загрузка предварительно сгенерированных данных (copy from csv)...	8
5. Типовой SQL запрос приложения, использующий связи.....	8
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	10
Приложение 1.....	11

ВВЕДЕНИЕ

Цели и задачи работы:

В рамках данной практической работы необходимо реализовать связи между таблицами в соответствии с разработанной моделью данных, а также сгенерировать релевантный набор тестовых данных для дальнейших манипуляций.

В ходе практической работы необходимо выполнить следующие задачи:

1. Проанализировать схему данных и установить, какой вид связи подходит в каждом конкретном случае.
2. Реализовать оставшиеся связи.
3. Проверить корректность связывания используя SQL SELECT JOIN запросы.
4. Продемонстрировать использование различных видов JOIN
5. Заполнить таблицы справочники.
6. Сгенерировать тестовый набор данных используя выбранный подход/инструмент (не менее 10000 сущностей):
 - Ручная генерация (generate_series);
 - Загрузка предварительно сгенерированных данных (copy from csv);
 - Библиотеки генерации (faker).
7. Придумать и выполнить «типовой» SQL запрос приложения использующий связи.

1. Анализ схемы данных

1.1 Описание таблицы и ключевых видов связей

Описание таблиц:

- JewelryType: Типы изделий (кольца, серьги и т.д.).
- Material: Материалы (металлы/камни) с характеристиками.
- Jewelry: Ювелирные изделия (связь с типами и материалами).
- JewelryMaterial: Связь многие-ко-многим между изделиями и материалами.
- Customer: Клиенты.
- Orders: Заказы.
- OrderItem: Позиции заказа (связь заказов с изделиями).

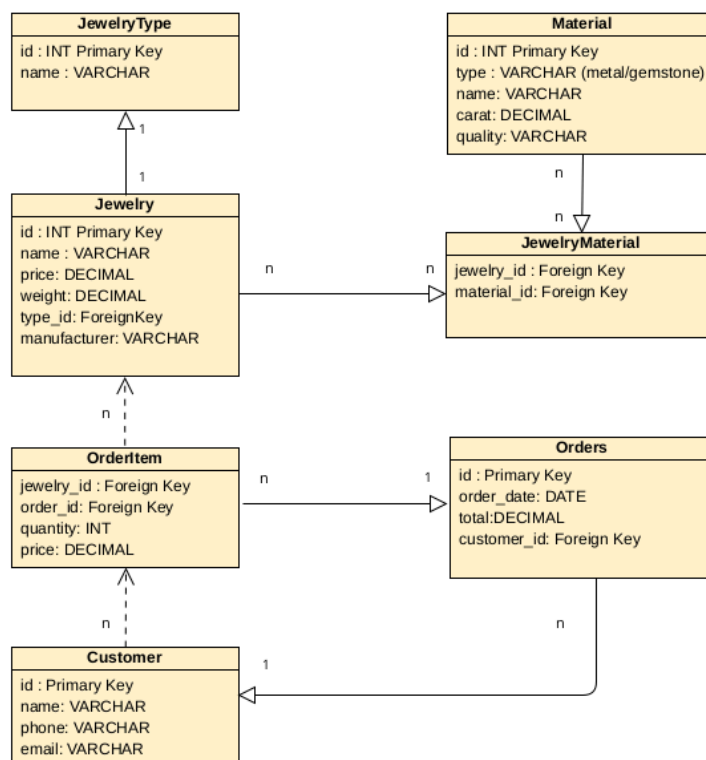


Рис.1 - UML схема базы данных

Ключевые связи:

1. Jewelry \rightarrow JewelryType

Каждое изделие относится к одному типу (`type_id` \rightarrow `JewelryType.id`).

2. Jewelry \leftrightarrow Material (через JewelryMaterial)

Одно изделие может содержать несколько материалов.

3. Customer \rightarrow Orders

Один клиент может иметь несколько заказов (`customer_id` \rightarrow `Customer.id`).

4. Orders \rightarrow OrderItem

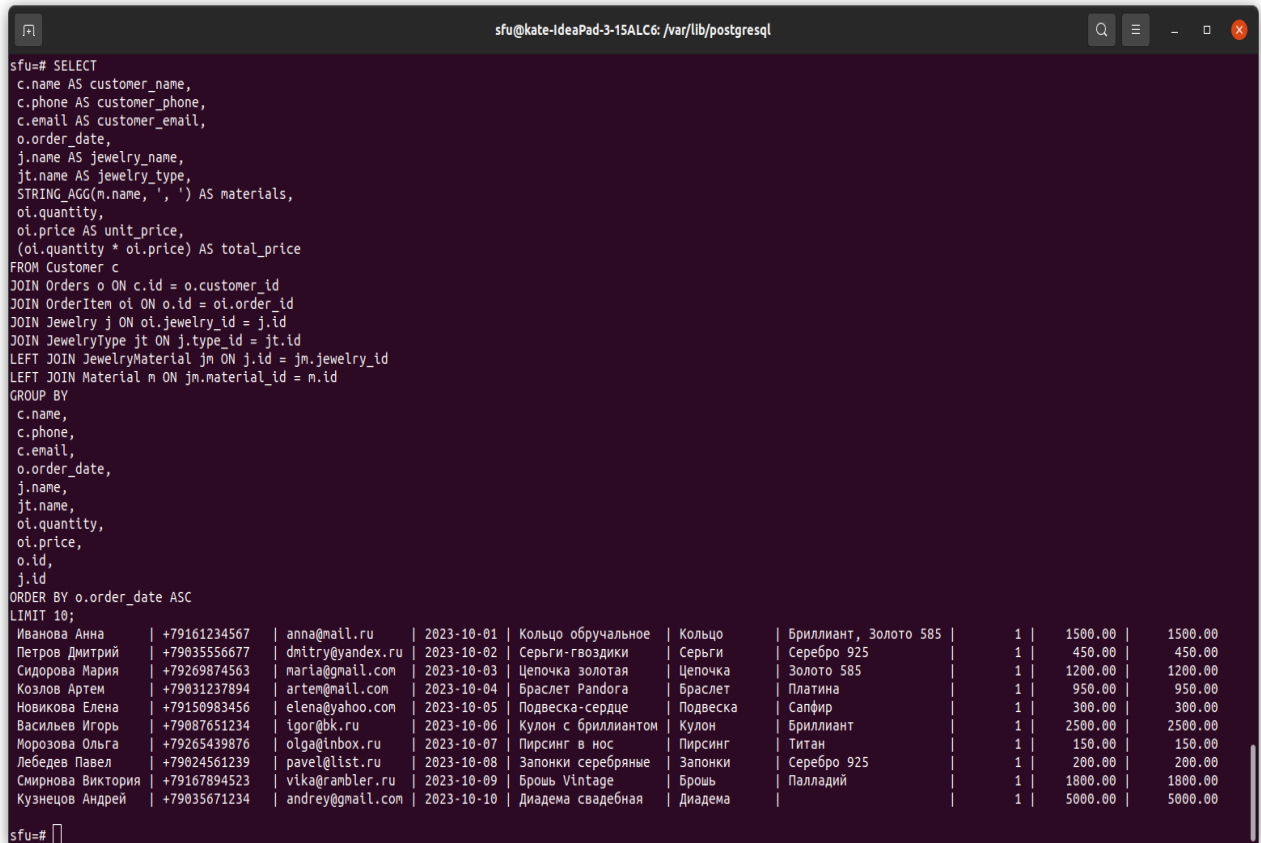
Один заказ содержит несколько позиций (`order_id` \rightarrow `Orders.id`).

5. OrderItem \rightarrow Jewelry

Каждая позиция заказа связана с одним изделием (`jewelry_id` \rightarrow `Jewelry.id`).

1.2 SQL SELECT JOIN запросы

Для демонстрации данных будем использовать запрос JOIN и LEFT JOIN, охватывающий основную информацию по всем таблицам.



```
sfu@kate-ideaPad-3-15ALC6: /var/lib/postgresql
sfu=# SELECT
  c.name AS customer_name,
  c.phone AS customer_phone,
  c.email AS customer_email,
  o.order_date,
  j.name AS jewelry_name,
  jt.name AS jewelry_type,
  STRING_AGG(m.name, ', ') AS materials,
  oi.quantity,
  oi.price AS unit_price,
  (oi.quantity * oi.price) AS total_price
FROM Customer c
JOIN Orders o ON c.id = o.customer_id
JOIN OrderItem oi ON o.id = oi.order_id
JOIN Jewelry j ON oi.jewelry_id = j.id
JOIN JewelryType jt ON j.type_id = jt.id
LEFT JOIN JewelryMaterial jm ON j.id = jm.jewelry_id
LEFT JOIN Material m ON jm.material_id = m.id
GROUP BY
  c.name,
  c.phone,
  c.email,
  o.order_date,
  j.name,
  jt.name,
  oi.quantity,
  oi.price,
  oi.id,
  j.id
ORDER BY o.order_date ASC
LIMIT 10;
```

Иванова Анна	+79161234567	anna@mail.ru	2023-10-01	Кольцо обручальное	Кольцо	Бриллиант, Золото 585	1	1500.00	1500.00
Петров Дмитрий	+79835556677	dmitry@yandex.ru	2023-10-02	Серьги-гвоздики	Серьги	Серебро 925	1	450.00	450.00
Сидорова Мария	+79269874563	maria@gmail.com	2023-10-03	Цепочка золотая	Цепочка	Золото 585	1	1200.00	1200.00
Козлов Артем	+79831237894	artem@mail.com	2023-10-04	Браслет Pandora	Браслет	Платина	1	950.00	950.00
Новикова Елена	+79150983456	elena@yahoo.com	2023-10-05	Подвеска-сердце	Подвеска	Сапфир	1	300.00	300.00
Васильев Игорь	+79887651234	igor@bk.ru	2023-10-06	Кулон с бриллиантом	Кулон	Бриллиант	1	2500.00	2500.00
Морозова Ольга	+79265439876	olga@inbox.ru	2023-10-07	Пирсинг в нос	Пирсинг	Титан	1	150.00	150.00
Лебедев Павел	+79824561239	pavel@list.ru	2023-10-08	Запонки серебряные	Запонки	Серебро 925	1	200.00	200.00
Смирнова Виктория	+79167894523	vika@rambler.ru	2023-10-09	Брошь Vintage	Брошь	Палладий	1	1800.00	1800.00
Кузнецов Андрей	+79835671234	andrey@gmail.com	2023-10-10	Диадема свадебная	Диадема		1	5000.00	5000.00

```
sfu=#
```

Рис.2 - Запрос к базе данных.

Запрос выполняет следующие действия:

1. Соединяет таблицы: Orders и Customer по ID клиента, Orders и OrderItem по ID заказа, OrderItem и Jewelry по ID изделия, JewelryType и Jewelry по ID типа изделия, Material и Jewelry через таблицу JewelryMaterial по ID изделия и ID материала соответственно.
2. Возвращает данные: имя клиента, номер телефона, почту, дату заказа, название ювелирного изделия, тип изделия, материал изделия, количество товара в позиции заказа, цену за единицу и общую стоимость позиции.

3. Упорядочивает результаты по дате заказа (свежие сверху) и имени клиента.
4. Ограничивает вывод в тестовых условиях до 10-ти записей.

2. Заполнение таблиц справочников

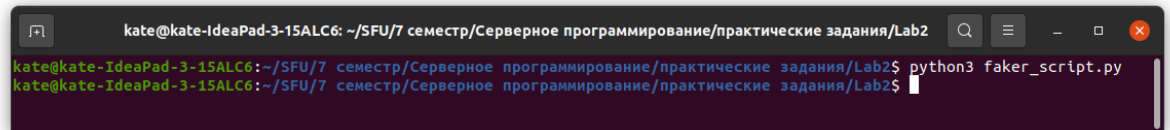
Таблицы-справочники (lookup tables) — это вспомогательные таблицы, которые хранят статичные данные для классификации или описания других сущностей. Справочники помогают нормализовать базу данных, уменьшить избыточность. В данной базе jewelry_db роль справочников выполняют таблицы JewelryType и Material.

```
sfu@kate-IdeaPad-3-15ALC6: ~  
jewelry_db=# SELECT * FROM JewelryType;  
id | name  
-----  
1 | Кольцо  
2 | Серьги  
3 | Цепочка  
4 | Браслет  
5 | Подвеска  
6 | Кулон  
7 | Пирсинг  
8 | Запонки  
9 | Брошь  
10 | Диадема  
(10 rows)
```

```
sfu=# SELECT * FROM Material;  
id | type | name | carat | quality  
-----  
1 | metal | Золото 585 | | Высшая проба  
2 | metal | Серебро 925 | | Стандарт  
3 | gemstone | Бриллиант | 1.20 | VVS1  
4 | metal | Платина | | 950 проба  
5 | gemstone | Сапфир | 0.80 | Синий  
6 | gemstone | Изумруд | 0.50 | AAA  
7 | metal | Титан | | Медицинский  
8 | gemstone | Рубин | 0.70 | Кровавый  
9 | metal | Палладий | | 999 проба  
10 | gemstone | Аметист | 0.30 | Фиолетовый  
(10 rows)  
sfu=#
```

3. Генерация тестовых данных с помощью библиотеки Faker

Создадим скрипт на языке python, используя библиотеку генерации фейковых данных Faker. (Приложение 1).

A screenshot of a terminal window with a dark background. The window title is "kate@kate-IdeaPad-3-15ALC6: ~/SFU/7 семестр/Серверное программирование/практические задания/Lab2". The terminal shows two lines of command history: "python3 faker_script.py" and a subsequent prompt "kate@kate-IdeaPad-3-15ALC6: ~/SFU/7 семестр/Серверное программирование/практические задания/Lab2\$".

```
kate@kate-IdeaPad-3-15ALC6: ~/SFU/7 семестр/Серверное программирование/практические задания/Lab2$ python3 faker_script.py
kate@kate-IdeaPad-3-15ALC6: ~/SFU/7 семестр/Серверное программирование/практические задания/Lab2$
```

Рис.3 - генерация фейковых данных faker_script.py

В результате генерации получим следующие файлы формата csv:

- jewelries.csv
- jewelry_types.csv
- materials.csv
- customers.csv
- orders.csv
- order_items.csv
- jewelry_materials.csv

4. Загрузка предварительно сгенерированных данных (copy from csv)

Выполним загрузку данных в базу данных jewelry_db:

- `psql -U sfu -d jewelry_db -c "\copy JewelryType FROM 'jewelry_types.csv' CSV HEADER"`
- `psql -U sfu -d jewelry_db -c "\copy Jewelry FROM 'jewelries.csv' CSV HEADER"`
- `psql -U sfu -d jewelry_db -c "\copy Jewelry FROM 'jewelries.csv' CSV HEADER"`
- `psql -U sfu -d jewelry_db -c "\copy Orders FROM 'orders.csv' CSV HEADER"`

- `psql -U sfu -d jewelry_db -c "\copy OrderItem FROM 'order_items.csv' CSV HEADER"`
- `psql -U sfu -d jewelry_db -c "\copy OrderItem FROM 'jewelry_materials.csv' CSV HEADER"`

5. Типовой SQL запрос приложения, использующий связи

Типовой запрос приложения может быть обращен к информации по заказам с клиентами и изделиями, ценой, с ограничением по дате оформления и количеству выводимых результатов. Результаты представлены в отсортированном виде по дате заказа в порядке увеличения:

```
SELECT
  c.name AS customer,
  o.order_date,
  j.name AS jewelry,
  oi.quantity,
  oi.price AS unit_price,
  (oi.quantity * oi.price) AS total_price
FROM Orders o
JOIN Customer c ON o.customer_id = c.id
JOIN OrderItem oi ON o.id = oi.order_id
JOIN Jewelry j ON oi.jewelry_id = j.id
WHERE o.order_date BETWEEN '2023-10-01' AND '2023-10-31'
ORDER BY o.order_date ASC
LIMIT 20;
```

```
sfu@kate-IdeaPad-3-15ALC6: ~  
^  
jewelry_db=# SELECT  
jewelry_db=#   c.name AS customer,  
jewelry_db=#   o.order_date,  
jewelry_db=#   j.name AS jewelry,  
jewelry_db=#   oi.quantity,  
jewelry_db=#   oi.price AS unit_price,  
jewelry_db=#   (oi.quantity * oi.price) AS total_price  
jewelry_db=# FROM Orders o  
jewelry_db=# JOIN Customer c ON o.customer_id = c.id  
jewelry_db=# JOIN OrderItem oi ON o.id = oi.order_id  
jewelry_db=# JOIN Jewelry j ON oi.jewelry_id = j.id  
jewelry_db=# WHERE o.order_date BETWEEN '2023-10-01' AND '2023-10-31'  
jewelry_db=# ORDER BY o.order_date ASC  
jewelry_db=# LIMIT 20;  
customer | order_date | jewelry | quantity | unit_price | total_price  
-----  
Кулаков Амос Абрамович | 2023-10-01 | Серебряное Подвеска | 2 | 2544.69 | 5089.38  
Савельев Исай Тимурович | 2023-10-01 | Серебряное Подвеска | 3 | 8174.54 | 24523.62  
Кулаков Амос Абрамович | 2023-10-01 | Золотое Кулон | 1 | 6457.05 | 6457.05  
Шаров Изяслав Георгиевич | 2023-10-01 | Золотое Пирсинг | 3 | 8761.96 | 26285.88  
Савельев Исай Тимурович | 2023-10-01 | Свадебное Браслет | 5 | 9460.95 | 47304.75  
Федосеева Кира Валентиновна | 2023-10-01 | Свадебное Подвеска | 5 | 8653.74 | 43268.70  
Тетерина Ульяна Степановна | 2023-10-01 | Серебряное Диадема | 5 | 6587.96 | 32939.80  
Кулаков Амос Абрамович | 2023-10-01 | Золотое Кулон | 1 | 7402.69 | 7402.69  
Кулаков Амос Абрамович | 2023-10-01 | Свадебное Цепочка | 4 | 3639.70 | 14558.80  
Шаров Изяслав Георгиевич | 2023-10-01 | Обручальное Серьги | 2 | 2379.00 | 4758.00  
Юлия Даниловна Евдокимова | 2023-10-01 | Серебряное Пирсинг | 5 | 2497.39 | 12486.95  
Савельев Исай Тимурович | 2023-10-01 | Обручальное Диадема | 4 | 3318.35 | 13273.40  
Кулаков Амос Абрамович | 2023-10-01 | Золотое Пирсинг | 2 | 8649.99 | 17299.98  
Шаров Изяслав Георгиевич | 2023-10-01 | Обручальное Подвеска | 5 | 6133.47 | 30667.35  
Шаров Изяслав Георгиевич | 2023-10-01 | Свадебное Брошь | 5 | 9124.82 | 45624.10  
Коновалов Харлампий Иосифович | 2023-10-02 | Обручальное Кольцо | 1 | 7362.07 | 7362.07  
Власова Зинаида Степановна | 2023-10-02 | Свадебное Кулон | 3 | 2099.78 | 6299.34  
Ксения Тимуровна Соколова | 2023-10-02 | Серебряное Пирсинг | 2 | 6555.79 | 13111.58  
Сазонов Ипполит Измаилович | 2023-10-02 | Серебряное Брошь | 5 | 9483.83 | 47419.15  
Бирюкова Ираида Георгиевна | 2023-10-02 | Свадебное Подвеска | 3 | 9995.73 | 29987.19  
(20 rows)  
jewelry_db=#
```

Рис.4 - типовой SQL запрос

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Генерация фейковых данных Faker. Python package - [Эл.ресурс]
<https://github.com/joke2k/faker>
2. Документация к PostgreSQL 17.4 - [Эл.ресурс]
<https://postgrespro.ru/docs/postgresql/17/index>

Листинг кода faker_script.py:

```
import csv
from faker import Faker
import random
from datetime import datetime, timedelta

fake = Faker('ru_RU')

# 1. Генерируем JewelryType (10 типов)
jewelry_types = [
    {"id": 1, "name": "Кольцо"},
    {"id": 2, "name": "Серьги"},
    {"id": 3, "name": "Цепочка"},
    {"id": 4, "name": "Браслет"},
    {"id": 5, "name": "Подвеска"},
    {"id": 6, "name": "Кулон"},
    {"id": 7, "name": "Пирсинг"},
    {"id": 8, "name": "Запонки"},
    {"id": 9, "name": "Брошь"},
    {"id": 10, "name": "Диадема"}
]

# 2. Генерируем Material (50 материалов)
materials = []
for i in range(1, 51):
    if i % 2 == 0:
        materials.append({
            "id": i,
            "type": "gemstone",
            "name": fake.word(ext_word_list=["Бриллиант", "Рубин", "Сапфир", "Изумруд", "Аметист"]),
            "carat": round(random.uniform(0.1, 5.0), 2),
            "quality": fake.word(ext_word_list=["VVS1", "VS1", "SI1", "Идеальная огранка"])
        })
    else:
        materials.append({
            "id": i,
            "type": "metal",
```

```

"name": fake.word(ext_word_list=["Золото 585", "Серебро 925", "Платина", "Титан", "Палладий"]),
"carat": None,
"quality": fake.word(ext_word_list=["Высшая проба", "Стандарт", "Медицинский"])
})

```

3. Генерируем Jewelry (2000 изделий)

```

jewelries = []
manufacturers = ["Tiffany", "Cartier", "Swarovski", "Pandora", "Local Jewelry"]
for i in range(1, 2001):
    jewelries.append({
        "id": i,
        "name": f"{fake.word(ext_word_list=['Обручальное', 'Свадебное', 'Золотое', 'Серебряное'])}
{random.choice(jewelry_types)['name']}",
        "price": round(random.uniform(100, 10000), 2),
        "weight": round(random.uniform(1.0, 100.0), 2),
        "type_id": random.randint(1, 10),
        "manufacturer": random.choice(manufacturers)
    })

```

4. Генерируем Customer (1000 клиентов)

```

customers = []
for i in range(1, 1001):
    customers.append({
        "id": i,
        "name": fake.name(),
        "phone": fake.phone_number(),
        "email": fake.email()
    })

```

5. Генерируем Orders (5000 заказов)

```

orders = []
start_date = datetime(2022, 1, 1)
for i in range(1, 5001):
    orders.append({
        "id": i,
        "order_date": (start_date + timedelta(days=random.randint(0, 730))).strftime('%Y-%m-%d'),
        "total": 0, # Будет рассчитано позже
        "customer_id": random.randint(1, 1000)
    })

```

6. Генерируем OrderItem (10000 позиций)

```
order_items = []
```

```
for i in range(1, 10001):
```

```
    jewelry = random.choice(jewelries)
```

```
    order_items.append({
```

```
        "order_id": random.randint(1, 5000),
```

```
        "jewelry_id": jewelry['id'],
```

```
        "quantity": random.randint(1, 5),
```

```
        "price": jewelry['price']
```

```
    })
```

7. JewelryMaterial (10000 связей)

```
jewelry_materials = []
```

```
for jewelry in jewelries:
```

```
    for _ in range(random.randint(1, 5)):
```

```
        jewelry_materials.append({
```

```
            "jewelry_id": jewelry['id'],
```

```
            "material_id": random.randint(1, 50)
```

```
        })
```

Сохранение в CSV

```
def save_to_csv(filename, data, columns):
```

```
    with open(filename, 'w', newline="", encoding='utf-8') as f:
```

```
        writer = csv.DictWriter(f, fieldnames=columns)
```

```
        writer.writeheader()
```

```
        writer.writerows(data)
```

```
save_to_csv('jewelry_types.csv', jewelry_types, ['id', 'name'])
```

```
save_to_csv('materials.csv', materials, ['id', 'type', 'name', 'carat', 'quality'])
```

```
save_to_csv('jewelries.csv', jewelries, ['id', 'name', 'price', 'weight', 'type_id', 'manufacturer'])
```

```
save_to_csv('customers.csv', customers, ['id', 'name', 'phone', 'email'])
```

```
save_to_csv('orders.csv', orders, ['id', 'order_date', 'total', 'customer_id'])
```

```
save_to_csv('order_items.csv', order_items, ['order_id', 'jewelry_id', 'quantity', 'price'])
```

```
save_to_csv('jewelry_materials.csv', jewelry_materials, ['jewelry_id', 'material_id'])
```