

Выбрали 3NF, так как она решает главные проблемы денормализованной таблицы, представленной в задании:

1. **Устранение избыточности:** Сейчас, если один и тот же пользователь делает 10 заказов, его телефон (`user_phone`) дублируется 10 раз. То же самое касается адресов магазинов и телефонов водителей. 3NF выносит эти данные в справочники.
2. **Целостность данных (Data Integrity):** Если водитель сменит номер телефона, в денормализованной таблице нам придется обновлять тысячи строк с его старыми заказами. В 3NF мы меняем телефон в одном месте (таблица `Drivers`).
3. **Исключение аномалий:** Мы защищаемся от аномалий вставки (нельзя добавить магазин, пока нет заказа) и удаления (если удалить все заказы, исчезнет информация о товаре).

2. Процесс нормализации

Для достижения 3NF мы должны пройти три шага:

- **1NF (Атомарность):** Все атрибуты должны быть атомарными.
 - *Решение:* Атрибуты `address_text` и `store_address` формально содержат составные данные (город, улица), но в рамках данной задачи мы будем считать их атомарными строками адреса, так как нет требований фильтровать отдельно по улице.
- **2NF (Зависимость от всего ключа):** Все неключевые атрибуты должны зависеть от полного первичного ключа.
 - *Проблема:* У нас есть составной ключ "Заказ + Товар". Но `driver_phone` зависит только от `driver_id` (который привязан к заказу), а `item_category` зависит только от `item_id`, но не от `order_id`.
- **3NF (Отсутствие транзитивных зависимостей):** Неключевые атрибуты не должны зависеть от других неключевых атрибутов.
 - *Проблема:* `store_address` зависит от `store_id`, а `store_id` зависит от `order_id`. Это транзитивная зависимость.

3. Итоговая схема данных (Результат)

Мы разбиваем исходную таблицу на 6 логических таблиц.

1. Таблица Users (Пользователи)

Хранит информацию о клиентах.

- **PK:** user_id
- user_phone

2. Таблица Drivers (Курьеры/Водители)

Хранит информацию о персонале доставки.

- **PK:** driver_id
- driver_phone

3. Таблица Stores (Магазины)

Справочник торговых точек.

- **PK:** store_id
- store_address (Название, Город, Улица...)

4. Таблица Items (Товары/Каталог)

Справочник номенклатуры. Цена здесь не хранится, так как это цена каталога (может меняться), а нам важна цена на момент сделки.

- **PK:** item_id
- item_title
- item_category

5. Таблица Orders (Заказы)

Основная таблица факта совершения заказа. Содержит ссылки (Foreign Keys) на участников процесса.

- **PK:** order_id
- **FK:** user_id (Кто заказал)
- **FK:** store_id (Откуда везем)
- **FK:** driver_id (Кто везет, может быть NULL, если еще не назначен)

- `address_text` (Адрес доставки конкретного заказа)
- `created_at`
- `paid_at`
- `delivery_started_at`
- `delivered_at`
- `canceled_at`
- `payment_type`
- `delivery_cost`
- `order_discount` (Скидка на чек целиком)
- `order_cancellation_reason`

Полный скрипт нормализации

```
-- -----  
-- 1. ЭТАП ПОДГОТОВКИ: Создание схемы (DDL)  
-- -----
```

```
CREATE TABLE IF NOT EXISTS users (  
    user_id BIGINT PRIMARY KEY,  
    user_phone VARCHAR(50)  
);
```

```
CREATE TABLE IF NOT EXISTS drivers (  
    driver_id BIGINT PRIMARY KEY,  
    driver_phone VARCHAR(50)  
);
```

```
CREATE TABLE IF NOT EXISTS stores (  
    store_id BIGINT PRIMARY KEY,  
    store_address TEXT  
);
```

```
CREATE TABLE IF NOT EXISTS items (  
    item_id BIGINT PRIMARY KEY,
```

```

    item_title VARCHAR(255),
    item_category VARCHAR(100)
);

CREATE TABLE IF NOT EXISTS orders (
    order_id VARCHAR(50) PRIMARY KEY,
    user_id BIGINT,
    store_id BIGINT,
    driver_id BIGINT,
    address_text TEXT,
    created_at TIMESTAMP,
    paid_at TIMESTAMP,
    delivery_started_at TIMESTAMP,
    delivered_at TIMESTAMP,
    canceled_at TIMESTAMP,
    payment_type VARCHAR(50),
    delivery_cost DECIMAL(10, 2),
    order_discount DECIMAL(10, 2),
    order_cancellation_reason TEXT,

    -- Внешние ключи
    CONSTRAINT fk_orders_user FOREIGN KEY (user_id) REFERENCES users(user_id),
    CONSTRAINT fk_orders_store FOREIGN KEY (store_id) REFERENCES stores(store_id),
    CONSTRAINT fk_orders_driver FOREIGN KEY (driver_id) REFERENCES drivers(driver_id)
);

CREATE TABLE IF NOT EXISTS order_items (
    order_id VARCHAR(50),
    item_id BIGINT,
    item_quantity INT,
    item_price DECIMAL(10, 2),          -- Цена на момент покупки
    item_discount DECIMAL(10, 2),      -- Скидка на конкретный товар
    item_canceled_quantity INT,
    item_replaced_id BIGINT,           -- Ссылка на товар-замену

    -- Составной первичный ключ (один товар в одном заказе встречается один раз)
    PRIMARY KEY (order_id, item_id),

    -- Внешние ключи
    CONSTRAINT fk_order_items_order FOREIGN KEY (order_id) REFERENCES orders(order_id),

```

```
        CONSTRAINT fk_order_items_item FOREIGN KEY (item_id) REFERENCES items(item_id),
        CONSTRAINT fk_order_items_replaced FOREIGN KEY (item_replaced_id) REFERENCES
items(item_id)
);
```

```
-- -----
-- 2. ЭТАП ETL: Миграция данных из сырой таблицы
-- -----
```

```
INSERT INTO users (user_id, user_phone)
SELECT DISTINCT user_id, user_phone
FROM raw_data
WHERE user_id IS NOT NULL
ON CONFLICT (user_id) DO NOTHING; -- Защита от дублей, если они есть
```

```
INSERT INTO drivers (driver_id, driver_phone)
SELECT DISTINCT driver_id, driver_phone
FROM raw_data
WHERE driver_id IS NOT NULL
ON CONFLICT (driver_id) DO NOTHING;
```

```
INSERT INTO stores (store_id, store_address)
SELECT DISTINCT store_id, store_address
FROM raw_data
WHERE store_id IS NOT NULL
ON CONFLICT (store_id) DO NOTHING;
```

```
INSERT INTO items (item_id, item_title, item_category)
SELECT DISTINCT item_id, item_title, item_category
FROM raw_data
WHERE item_id IS NOT NULL
ON CONFLICT (item_id) DO NOTHING;
```

```
INSERT INTO items (item_id, item_title, item_category)
SELECT DISTINCT item_replaced_id, 'Unknown Replacement Item', 'Unknown'
FROM raw_data
WHERE item_replaced_id IS NOT NULL
    AND item_replaced_id NOT IN (SELECT item_id FROM items)
ON CONFLICT (item_id) DO NOTHING;
```

```
(
    order_id, user_id, store_id, driver_id, address_text,
    created_at, paid_at, delivery_started_at, delivered_at, canceled_at,
    payment_type, delivery_cost, order_discount, order_cancellation_reason
)
```

```
SELECT DISTINCT
    order_id,
    user_id,
    store_id,
    driver_id,
    address_text,
    created_at::TIMESTAMP,
    paid_at::TIMESTAMP,
    delivery_started_at::TIMESTAMP,
    delivered_at::TIMESTAMP,
    canceled_at::TIMESTAMP,
    payment_type,
    delivery_cost,
    order_discount,
    order_cancellation_reason
```

```
FROM raw_data
```

```
WHERE order_id IS NOT NULL
```

```
ON CONFLICT (order_id) DO NOTHING;
```

```
INSERT INTO order_items (
    order_id, item_id, item_quantity, item_price,
    item_discount, item_canceled_quantity, item_replaced_id
)
```

```
SELECT DISTINCT
    order_id,
    item_id,
    item_quantity,
    item_price,
    item_discount,
    item_canceled_quantity,
    item_replaced_id
```

```
FROM raw_data
```

```
WHERE order_id IS NOT NULL AND item_id IS NOT NULL
```

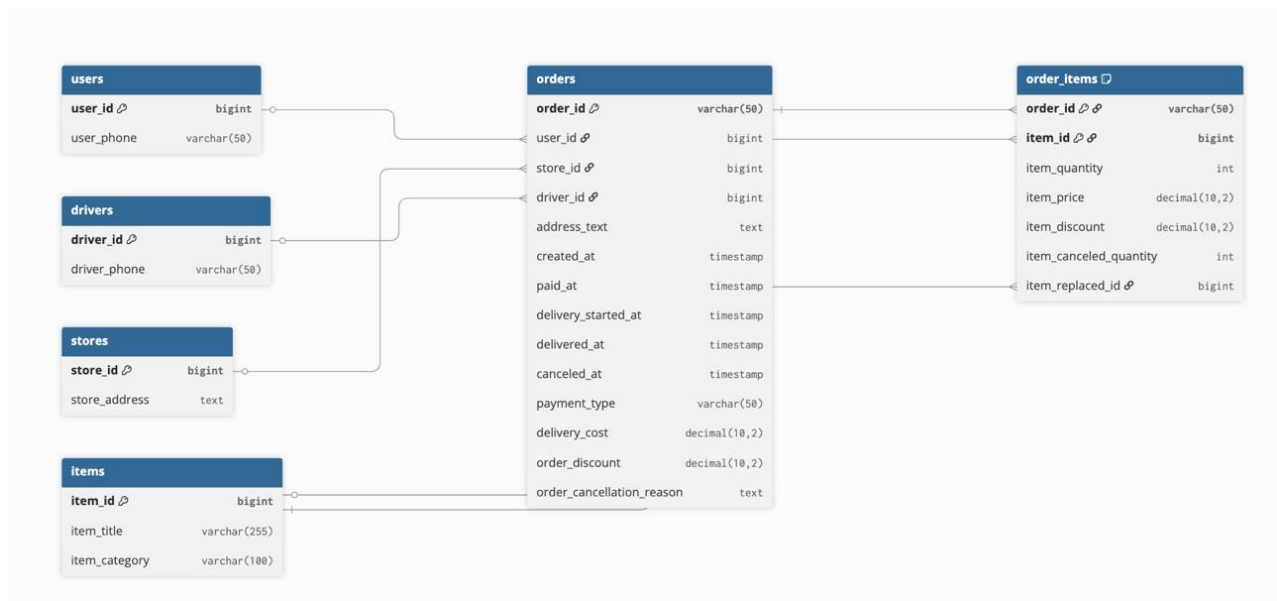
```
ON CONFLICT (order_id, item_id) DO NOTHING;
```

```
-- -----  
-- 3. ПРОВЕРКА (ОПЦИОНАЛЬНО)  
-- -----
```

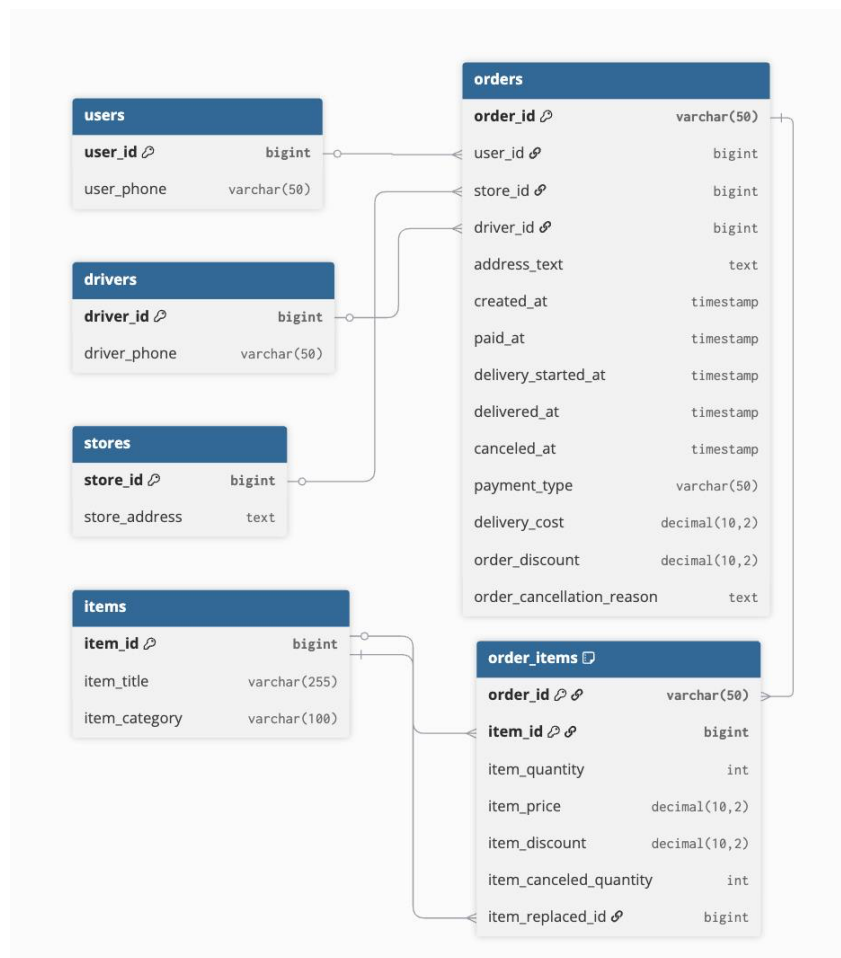
```
SELECT 'Orders count' as metric, count(*) FROM orders  
UNION ALL  
SELECT 'Items count', count(*) FROM items  
UNION ALL  
SELECT 'Users count', count(*) FROM users;
```

Ключевые моменты для проекта:

1. **ON CONFLICT DO NOTHING:** Эта команда (в PostgreSQL) позволяет скрипту не падать, если он встретит дубликаты ID с разными данными, просто пропуская их.
2. **Обработка item_replaced_id:** В пункте 2.4 есть хитрый запрос. Дело в том, что в сырых данных может быть ID товара замены, которого нет в колонке основного товара `item_id`. Если мы не добавим его в таблицу `items` заранее, при вставке в `order_items` база данных выдаст ошибку "Foreign Key Violation".
3. **Приведение типов (::TIMESTAMP):** В блоке 2.5 я добавил явное приведение типов, так как часто в сырых CSV/Excel дата приходит как строка текста. Если в исходной таблице уже стоит тип Date/Timestamp, эти касты (`::...`) можно убрать.



Реляционная схема 1



Реляционная схема 2