МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

Санкт-Петербургский национальный исследовательский университет ИТМО

Мегафакультет трансляционных информационных технологий

Факультет информационных технологий и программирования

Лабораторная работа №2

По дисциплине «Технологии управления данными»

Выполнил студент группы №M3306

Бочагова Екатерина

Проверил

Повышев Владислав Вячеславович

Санкт-Петербург

2025

Условие лабораторной работы:

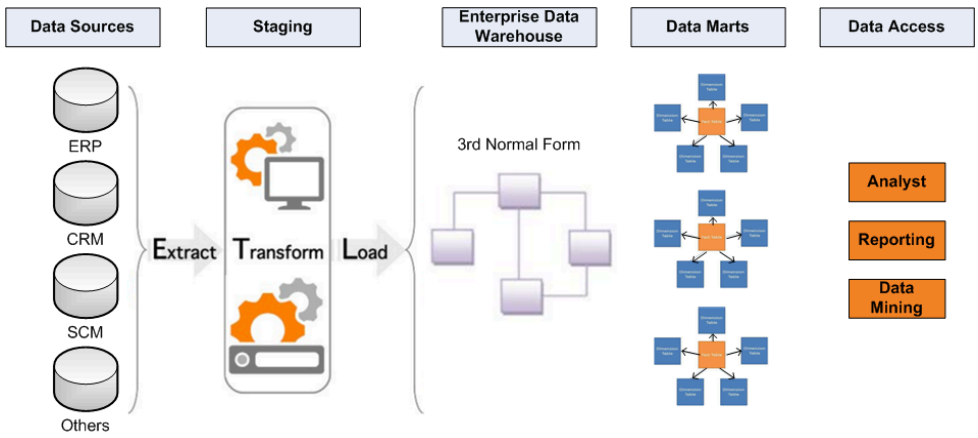
Разработка модели и скрипта создания хранилища данных. Хранилище данных должно реализовано с использованием реляционной СУБД, если иное не было оговорено заранее. Архитектура хранилища должна соответствовать одной из устоявшихся практик: Кимбалл, Инмон, Якорная модель, Data vault, или иная практика по согласованию с преподавателем. Студент обязан в отчете обосновать выбранную архитектуру. Наполнение будет производится в автоматизированном режиме из данных, поступающих из источников, разработанных в лабораторной работе №1. Для хранилища данных предусмотрена только операция добавления данных, операции модификации и удаления недопустимы. Пример: Сформировать модель «звезда», в которой основной таблицей фактов будет продажа конкретного товар, измерениями будут товары, покупатели и т.д. На основании имеющейся информации можно создать реляционную модель, утвердить ее у преподавателя. На основании утвержденной модели создать даталогическую модель, утвердить ее у преподавателя. На основании созданных моделей создать скрипт создания хранилища данных, проверить его работоспособность. Создать тестовый (!) скрипт наполнения хранилища, проверить работоспособность.

Теоретическое обоснование:

Кратко распишем, какие в целом бывают архитектуры хранилищ данных, сравним их между собой и выберем оптимальный вариант для выполнения задания

- Кимбалл

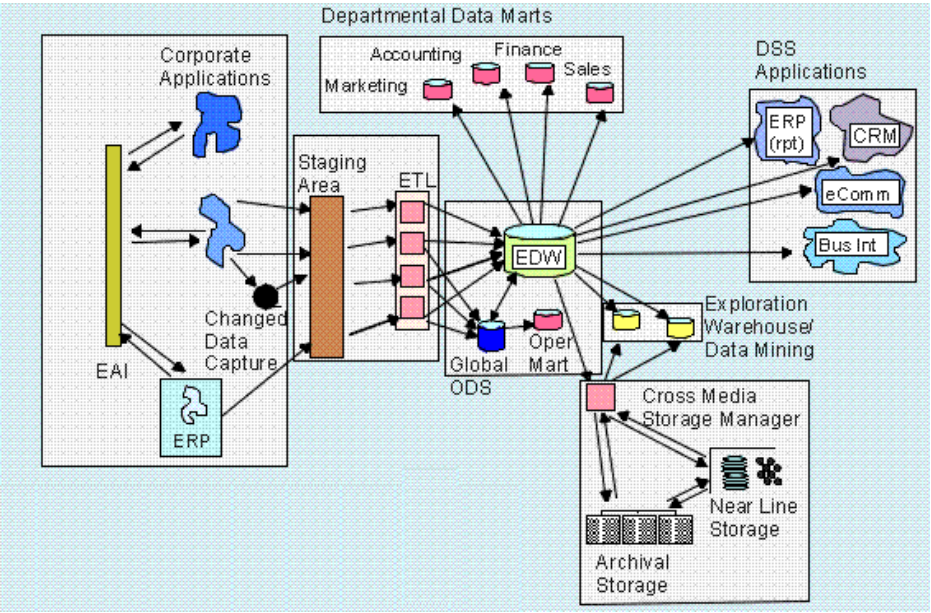
  Метод моделирования данных начинается с создания таблиц фактов(показатели) и измерений(атрибуты метрик из фактов).
  В архитектуре данные копируются из одной бд в другую, в так называемую область подготовки данных, далее строятся витрины



- Инмон

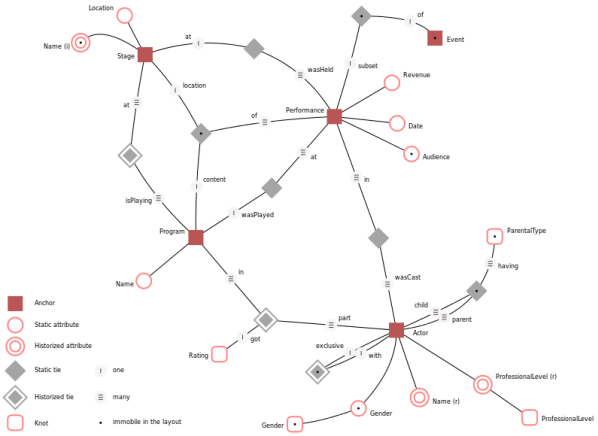
  Имеет несколько уровней организации данных.

Оперативный уровень – данные из систем обработки транзакций, предназначен для поддержки функционирования организации
Хранилище с атомарными данными – данные приводятся к единообразию из оперативного уровня.
Ведомственный уровень – данные поступают из атомарного уровня, в зависимости от специфики, сильно или слабо агрегированы.
Индивидуальный уровень – временный характер, проверка гипотез.



- Якорная модель

  Нацелена на разработку ХД в условиях Big Data. Высокая нормализация данных.
  Состоит из:
  Якорь – набор бизнес-ключей
  Атрибут – дополнительные сведения о якорях.
  Связь – отношение между якорями
  Узел – неизменяемый набор значений.
  Каждый объект имеет собственную таблицу.



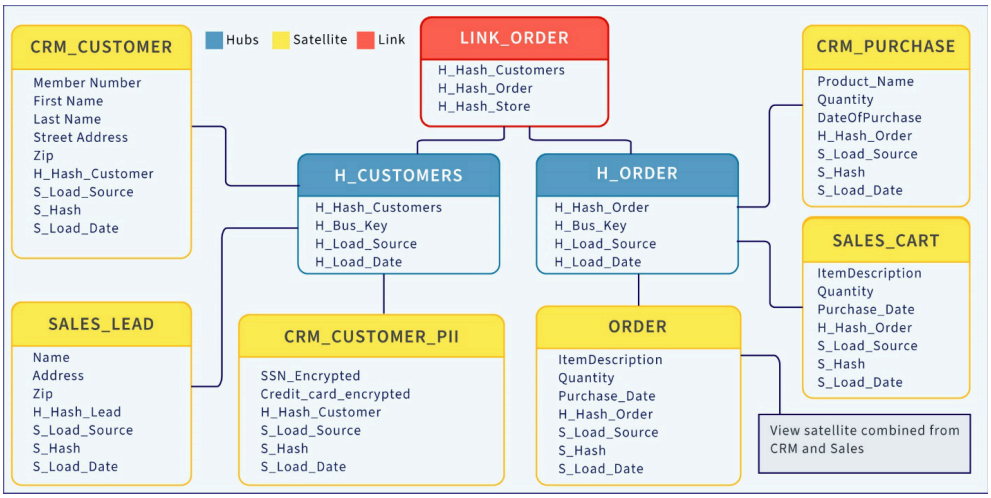
- Data vault

  Составляющие:

  Хаб – таблица, хранящая основное представление бизнес-сущности с функциональной позиции предметной области. Хаб состоит из уникального и неизменяемого бизнес ключа(guid).

  Связь – таблица, которая соединяет и масштабирует систему.

  Спутник – таблица с описанием информацией ключа хаба.



В рамках лабораторной работы буду использовать методологию Кимбалла по следующим причинам:

- Метод предназначен для аналитических целей — анализа данных, формирования отчетов, выявления закономерностей.

  Модель “звезда” упрощает формулирование запросов и повышает скорость вычислений при агрегировании данных.

- Архитектура Кимбалла строится по принципу: “факты в центре, измерения вокруг”, что делает модель интуитивно понятной, визуально наглядной и легко расширяемой
- Высокая производительность запросов - за счёт денормализации
- Добавление новых таблиц фактов или измерений не требует изменения существующей структуры.

Почему не были выбраны другие архитектуры

- Модель Data Vault подходит для корпоративных систем, где требуется:
  - хранить детализированную историю изменений,
  - интегрировать множество разнородных источников данных, построить масштабируемое, устойчивое к изменениям хранилище.

  Однако структура Data Vault не оптимальна для прямого аналитического доступа, поэтому для учебного проекта и ограниченного числа источников этот подход нецелесообразен
- Якорная модель (Anchor Modeling) - данный подход используется для высоко динамичных систем с частыми изменениями структуры данных. Он основан на сильной нормализации и разбиении данных на множество таблиц, что усложняет реализацию и работу с моделью.
  В рамках учебного проекта такая детализация избыточна и не дает практических преимуществ перед более наглядным методом Кимбалла.

Метод Кимбалла выбран как наиболее подходящий для целей лабораторной работы. Он хорошо демонстрирует взаимосвязь данных и легко расширяется при необходимости. Хранилище будет реализовано по схеме «звезда», где таблица фактов отражает факты (продажи), а таблицы измерений содержат описание связанных характеристик (товары, клиенты, даты, филиалы)

Опишем как должно выглядеть наше хранилище данных

1. fact_sales — таблица фактов - хранит все продажи
   sale_key – primary key
   sale_uid
   customer_key – ссылка на клиента
   product_key – ссылка на товар
   date_key – ссылка на дату
   quantity
   unit_price
   line_total

2. dim_customer — измерение клиентов - справочник клиентов
   customer_key – primary key
   customer_uid – исходный UUID
   customer_name – имя клиента

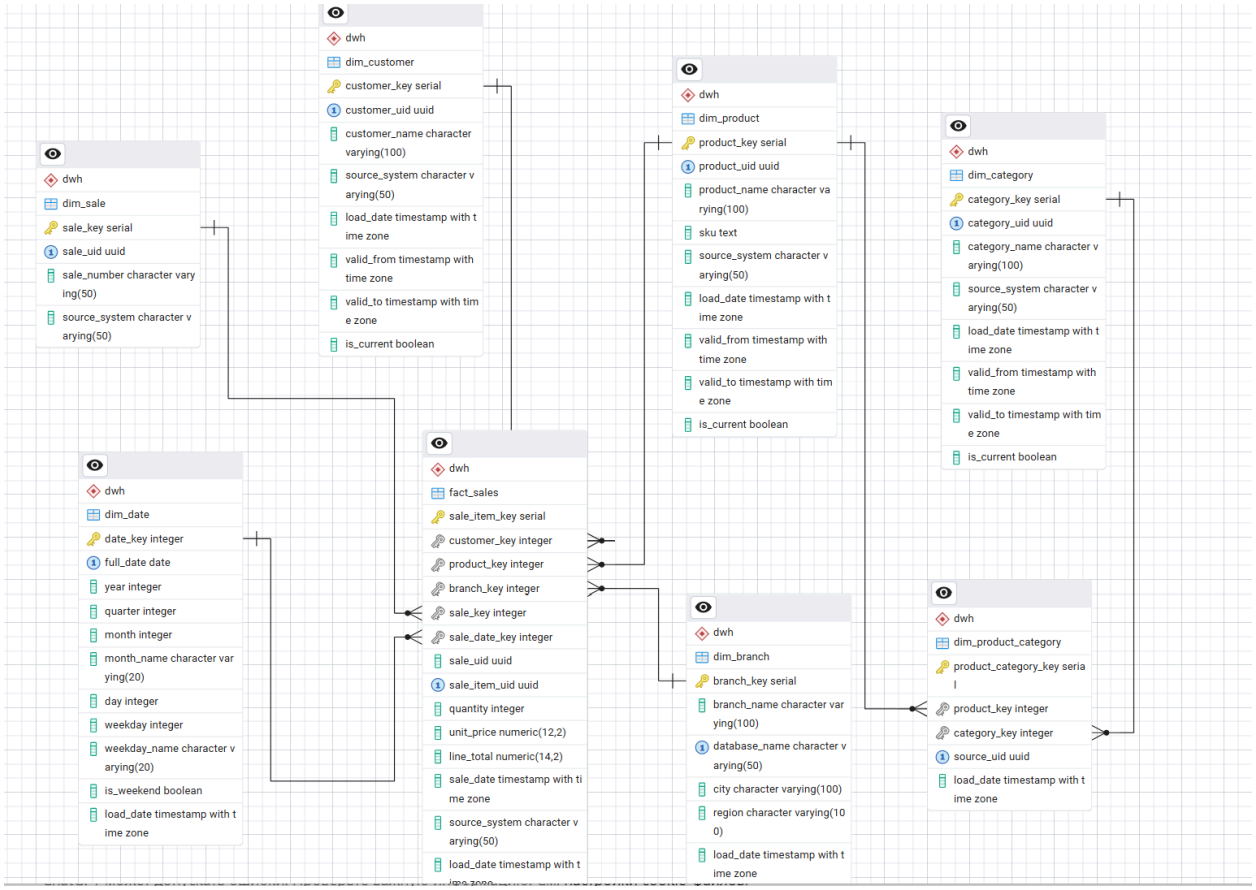
load_date – дата загрузки в DWH

3. dim_product — измерение товаров - справочник товаров
    product_key – primary key
    product_uid
    product_name
    sku
    price
    category_key – ссылка на категорию (dim_category) load_date

4. dim_category — измерение категорий
    category_key
    category_uid
    category_name
    load_date

5. dim_date — измерение времени - позволяет группировать данные по дате
    date_key
    full_date
    year
    month
    day
    weekday
    load_date

ERD-диаграмма



Перейдем к написанию скрипта создания хранилища данных:

Комментарии к коду:

На всякий случай нужно добавить первую строчку в этом файле - чтобы uuid корректно отработал

Исправила нейминг при помощи ии + отредактировала еще два измерения (чек, филиал, перенесла вычисление стоимости в факт)

Использую serial для ключей, чтобы они автоинкрементировались

```
CREATE EXTENSION IF NOT EXISTS pgcrypto;

CREATE SCHEMA IF NOT EXISTS dwh;

CREATE TABLE IF NOT EXISTS dwh.dim_customer (
   customer_key SERIAL PRIMARY KEY,
   customer_uid UUID UNIQUE NOT NULL,
   customer_name VARCHAR(100) NOT NULL,
   source_system VARCHAR(50) NOT NULL,
   load_date TIMESTAMPTZ NOT NULL DEFAULT now(),
   valid_from TIMESTAMPTZ NOT NULL DEFAULT now(),
   valid_to TIMESTAMPTZ DEFAULT '9999-12-31'::timestamptz,
   is_current BOOLEAN DEFAULT true
);
```

```sql
CREATE TABLE IF NOT EXISTS dwh.dim_category (
   category_key SERIAL PRIMARY KEY,
   category_uid UUID UNIQUE NOT NULL,
   category_name VARCHAR(100) NOT NULL,
   source_system VARCHAR(50) NOT NULL,
   load_date TIMESTAMPTZ NOT NULL DEFAULT now(),
   valid_from TIMESTAMPTZ NOT NULL DEFAULT now(),
   valid_to TIMESTAMPTZ DEFAULT '9999-12-31'::timestamptz,
   is_current BOOLEAN DEFAULT true
);

CREATE TABLE IF NOT EXISTS dwh.dim_product (
   product_key SERIAL PRIMARY KEY,
   product_uid UUID UNIQUE NOT NULL,
   product_name VARCHAR(100) NOT NULL,
   sku TEXT NOT NULL,
   source_system VARCHAR(50) NOT NULL,
   load_date TIMESTAMPTZ NOT NULL DEFAULT now(),
   valid_from TIMESTAMPTZ NOT NULL DEFAULT now(),
   valid_to TIMESTAMPTZ DEFAULT '9999-12-31'::timestamptz,
   is_current BOOLEAN DEFAULT true
);

CREATE TABLE IF NOT EXISTS dwh.dim_product_category (
   product_category_key SERIAL PRIMARY KEY,
   product_key INT NOT NULL,
   category_key INT NOT NULL,
   source_uid UUID UNIQUE NOT NULL,
   load_date TIMESTAMPTZ NOT NULL DEFAULT now(),
   FOREIGN KEY (product_key) REFERENCES dwh.dim_product(product_key),
   FOREIGN KEY (category_key) REFERENCES
dwh.dim_category(category_key),
   UNIQUE(product_key, category_key)
);

CREATE TABLE IF NOT EXISTS dwh.dim_branch (
   branch_key SERIAL PRIMARY KEY,
   branch_name VARCHAR(100) NOT NULL,
   database_name VARCHAR(50) UNIQUE NOT NULL,
   load_date TIMESTAMPTZ NOT NULL DEFAULT now()
);

CREATE TABLE IF NOT EXISTS dwh.dim_sale (
   sale_key SERIAL PRIMARY KEY,
```

```
    sale_uid UUID UNIQUE NOT NULL,
    sale_number VARCHAR(50),
    source_system VARCHAR(50) NOT NULL,
);

CREATE TABLE IF NOT EXISTS dwh.dim_date (
    date_key INT PRIMARY KEY,
    full_date DATE UNIQUE NOT NULL
    year INT NOT NULL,
    quarter INT NOT NULL,
    month INT NOT NULL,
    month_name VARCHAR(20) NOT NULL,
    day INT NOT NULL,
    weekday INT NOT NULL,
    weekday_name VARCHAR(20) NOT NULL,
    is_weekend BOOLEAN NOT NULL,
    load_date TIMESTAMPTZ NOT NULL DEFAULT now()
);

CREATE TABLE IF NOT EXISTS dwh.fact_sales (
    sale_item_key SERIAL PRIMARY KEY,

    customer_key INT NOT NULL,
    product_key INT NOT NULL,
    branch_key INT NOT NULL,
    sale_key INT NOT NULL,
    sale_date_key INT NOT NULL,

    sale_uid UUID NOT NULL,
    sale_item_uid UUID UNIQUE NOT NULL,

    quantity INT NOT NULL CHECK (quantity > 0),
    unit_price NUMERIC(12,2) NOT NULL CHECK (unit_price >= 0),
    line_total NUMERIC(14,2) GENERATED ALWAYS AS (quantity *
unit_price) STORED,

    sale_date TIMESTAMPTZ NOT NULL,
    source_system VARCHAR(50) NOT NULL,
    load_date TIMESTAMPTZ NOT NULL DEFAULT now(),

    FOREIGN KEY (customer_key) REFERENCES
dwh.dim_customer(customer_key),
    FOREIGN KEY (product_key) REFERENCES dwh.dim_product(product_key),
    FOREIGN KEY (branch_key) REFERENCES dwh.dim_branch(branch_key),
    FOREIGN KEY (sale_key) REFERENCES dwh.dim_sale(sale_key),
```

```
    FOREIGN KEY (sale_date_key) REFERENCES dwh.dim_date(date_key)
);

CREATE INDEX IF NOT EXISTS idx_dim_customer_uid ON
dwh.dim_customer(customer_uid);
CREATE INDEX IF NOT EXISTS idx_dim_customer_current ON
dwh.dim_customer(is_current);

CREATE INDEX IF NOT EXISTS idx_dim_product_uid ON
dwh.dim_product(product_uid);
CREATE INDEX IF NOT EXISTS idx_dim_product_sku ON
dwh.dim_product(sku);
CREATE INDEX IF NOT EXISTS idx_dim_product_current ON
dwh.dim_product(is_current);

CREATE INDEX IF NOT EXISTS idx_dim_category_uid ON
dwh.dim_category(category_uid);
CREATE INDEX IF NOT EXISTS idx_dim_category_current ON
dwh.dim_category(is_current);

CREATE INDEX IF NOT EXISTS idx_dim_sale_uid ON
dwh.dim_sale(sale_uid);
CREATE INDEX IF NOT EXISTS idx_dim_sale_customer ON
dwh.dim_sale(customer_key);
CREATE INDEX IF NOT EXISTS idx_dim_sale_branch ON
dwh.dim_sale(branch_key);
CREATE INDEX IF NOT EXISTS idx_dim_sale_date ON
dwh.dim_sale(sale_date);
CREATE INDEX IF NOT EXISTS idx_dim_sale_number ON
dwh.dim_sale(sale_number);

CREATE INDEX IF NOT EXISTS idx_fact_sales_customer ON
dwh.fact_sales(customer_key);
CREATE INDEX IF NOT EXISTS idx_fact_sales_product ON
dwh.fact_sales(product_key);
CREATE INDEX IF NOT EXISTS idx_fact_sales_branch ON
dwh.fact_sales(branch_key);
CREATE INDEX IF NOT EXISTS idx_fact_sales_sale ON
dwh.fact_sales(sale_key);
CREATE INDEX IF NOT EXISTS idx_fact_sales_date ON
dwh.fact_sales(sale_date_key);
CREATE INDEX IF NOT EXISTS idx_fact_sales_sale_date ON
dwh.fact_sales(sale_date);
CREATE INDEX IF NOT EXISTS idx_fact_sales_source ON
dwh.fact_sales(source_system);
```

```
CREATE INDEX IF NOT EXISTS idx_fact_sales_sale_uid ON
dwh.fact_sales(sale_uid);
```

Наполним хранилище данных тестовыми данными (при написании скрипта использовала ChatGPT)

```
INSERT INTO dwh.dim_branch (branch_name, database_name, city, region)
VALUES
    ('Центральный филиал', 'filial_central', 'Москва', 'Центральный
округ'),
    ('Северный филиал', 'filial_north', 'Санкт-Петербург',
'Северо-Западный округ'),
    ('Восточный филиал', 'filial_east', 'Новосибирск', 'Сибирский
округ')
ON CONFLICT (database_name) DO NOTHING;

INSERT INTO dwh.dim_customer (customer_uid, customer_name,
source_system)
VALUES
    (gen_random_uuid(), 'ООО Альфа', 'CRM'),
    (gen_random_uuid(), 'ООО Бета', 'CRM'),
    (gen_random_uuid(), 'ООО Гамма', 'ERP');

INSERT INTO dwh.dim_category (category_uid, category_name,
source_system)
VALUES
    (gen_random_uuid(), 'Электроника', 'PIM'),
    (gen_random_uuid(), 'Бытовая техника', 'PIM'),
    (gen_random_uuid(), 'Одежда', 'PIM');

INSERT INTO dwh.dim_product (product_uid, product_name, sku,
source_system)
VALUES
    (gen_random_uuid(), 'Смартфон X100', 'SKU100', 'ERP'),
    (gen_random_uuid(), 'Холодильник Frosty 2000', 'SKU200', 'ERP'),
    (gen_random_uuid(), 'Футболка Cotton', 'SKU300', 'ERP');

INSERT INTO dwh.dim_product_category (product_key, category_key,
source_uid)
VALUES
    (1, 1, gen_random_uuid()),
    (2, 2, gen_random_uuid()),
    (3, 3, gen_random_uuid())
ON CONFLICT DO NOTHING;
```

```
DO $$
DECLARE
    start_date DATE := '2024-01-01';
    end_date DATE := '2024-12-31';
    curr_date DATE := start_date;
BEGIN
    WHILE curr_date <= end_date LOOP
        INSERT INTO dwh.dim_date (
            date_key,
            full_date,
            year,
            quarter,
            month,
            month_name,
            day,
            weekday,
            weekday_name,
            is_weekend
        ) VALUES (
            TO_CHAR(curr_date, 'YYYYMMDD')::INT,
            curr_date,
            EXTRACT(YEAR FROM curr_date)::INT,
            EXTRACT(QUARTER FROM curr_date)::INT,
            EXTRACT(MONTH FROM curr_date)::INT,
            TO_CHAR(curr_date, 'FMMonth'),
            EXTRACT(DAY FROM curr_date)::INT,
            EXTRACT(DOW FROM curr_date)::INT,
            TO_CHAR(curr_date, 'FMDay'),
            EXTRACT(DOW FROM curr_date) IN (0, 6)
        )
        ON CONFLICT (date_key) DO NOTHING;

        curr_date := curr_date + INTERVAL '1 day';
    END LOOP;
END $$;

INSERT INTO dwh.dim_sale (sale_uid, sale_number, source_system)
VALUES
    (gen_random_uuid(), 'SALE-001', 'POS'),
    (gen_random_uuid(), 'SALE-002', 'POS'),
    (gen_random_uuid(), 'SALE-003', 'POS');

INSERT INTO dwh.fact_sales (
    customer_key,
    product_key,
```

```
    branch_key,
    sale_key,
    sale_date_key,
    sale_uid,
    sale_item_uid,
    quantity,
    unit_price,
    sale_date,
    source_system
)
VALUES
    (1, 1, 1, 1, 20240115, gen_random_uuid(), gen_random_uuid(), 2,
35000.00, '2024-01-15', 'POS'),
    (1, 2, 1, 1, 20240115, gen_random_uuid(), gen_random_uuid(), 1,
55000.00, '2024-01-15', 'POS'),
    (2, 3, 2, 2, 20240210, gen_random_uuid(), gen_random_uuid(), 3,
1500.00, '2024-02-10', 'POS'),
    (3, 1, 3, 3, 20240305, gen_random_uuid(), gen_random_uuid(), 1,
36000.00, '2024-03-05', 'POS');
```