МИНИСТЕРСТВО ТРАНСПОРТА РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА (МИИТ)»

Кафедра ЦТУТП

**Отчёт**

**По лабораторной работе №2**  
по дисциплине «Информационные хранилища и аналитические системы»

Тема: «ПОСТРОЕНИЕ АНАЛИТИЧЕСКИХ ВИТРИН»

Выполнил: Шедания В. М.

Группа: УИС-411

Преподаватель: доц. Козьяков П. О.

­

Москва 2025 г.

**СОДЕРЖАНИЕ**

[ЦЕЛЬ ЛАБОРАТОРНОЙ РАБОТЫ 3](#_Toc195036875)

[ВАРИАНТ ЗАДАНИЯ 4](#_Toc195036876)

[МОДЕЛЬ ИСТОЧНИКА ДАННЫХ 5](#_Toc195036877)

[МОДЕЛЬ ДАННЫХ 6](#_Toc195036878)

[ИСХОДНЫЙ КОД ПРОГРАММЫ 9](#_Toc195036879)

[РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ 14](#_Toc195036880)

[ВЫВОД 17](#_Toc195036881)

# ЦЕЛЬ ЛАБОРАТОРНОЙ РАБОТЫ

Овладеть навыками проектирования аналитических витрин данных в схеме «Снежинка».

# ВАРИАНТ ЗАДАНИЯ

Вариант №9

1. Проанализировать структуру данных, предоставленных исходных файлов (вариант см. на рисунке 1).
2. Изучить теоретические основы построения схемы данных по варианту.
3. Выполнить проектирование витрины для имеющихся данных согласно схеме по варианту, оформить в виде диаграммы «Crow's foot».
4. Загрузить (импортировать) данные из файлов по варианту в Python в виде объектов DataFrame.
5. Выполнить очистку данных от лишних символов, приведение связанных между собой колонок к единому типу данных, строки с датами и временем, числовые показатели - преобразовать к соответствующим типам данных.
6. Сформировать из полученных наборов таблицы, согласно спроектированной в п.3 схеме данных (таблицы измерений и таблицу фактов).
7. Выгрузить (экспортировать) таблицу фактов и все таблицы измерений в единый excel-файл на отдельные листы с соответствующим наименованием.
8. Подготовить отчет о выполнении лабораторной работы.



# МОДЕЛЬ ИСТОЧНИКА ДАННЫХ

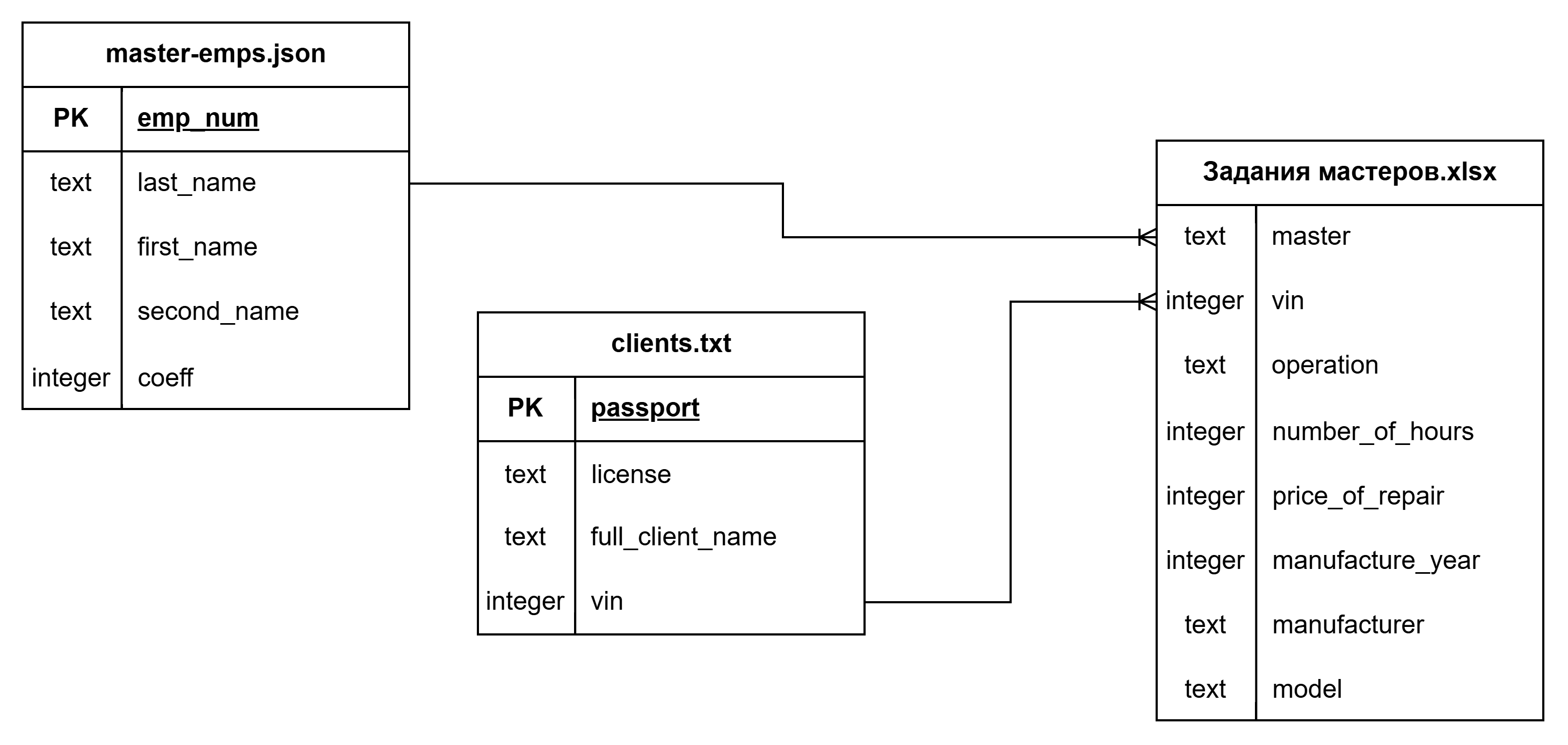


Рисунок 1 – Модель источника данных

# МОДЕЛЬ ДАННЫХ

Это логическая схема хранилища данных в виде снежинки, построенная вокруг таблицы фактов FACT\_Заказ, которая фиксирует данные о выполненных заказах на ремонт автомобилей. Каждая строка в этой таблице описывает конкретный заказ: какой клиент обратился, какой мастер выполнил работу, какая операция проводилась и на каком автомобиле. Дополнительно хранятся дата выполнения, количество часов работы и стоимость ремонта. Все ключи — составной первичный ключ, который обеспечивает уникальность каждой записи.

Измерение DIM\_Мастер содержит персональные данные мастеров и коэффициент (вероятно, влияет на стоимость работ). Измерение DIM\_Операция описывает вид выполняемых ремонтных работ, например: диагностика, покраска, замена деталей. Измерение Клиент содержит паспортные и личные данные клиентов, но не относится к общей префиксной нотации "DIM\_", что может указывать на логическое отличие или недоработку именования.

Цепочка DIM\_Автомобиль → DIM\_Модель → DIM\_Производитель представляет собой иерархическую структуру, нормализованную по всем правилам снежинки. DIM\_Автомобиль содержит VIN, год выпуска и внешний ключ на модель. DIM\_Модель — наименование модели и ссылка на производителя. А DIM\_Производитель хранит бренд или марку. Такой подход позволяет исключить дублирование информации, если множество автомобилей относятся к одной модели или марке.

Таким образом, структура соответствует схеме "снежинка": таблица фактов окружена нормализованными измерениями, иерархии вынесены в отдельные таблицы. Это удобно для поддержки бизнес-логики, обеспечивает компактность хранения, но требует дополнительных JOIN'ов при аналитических запросах.

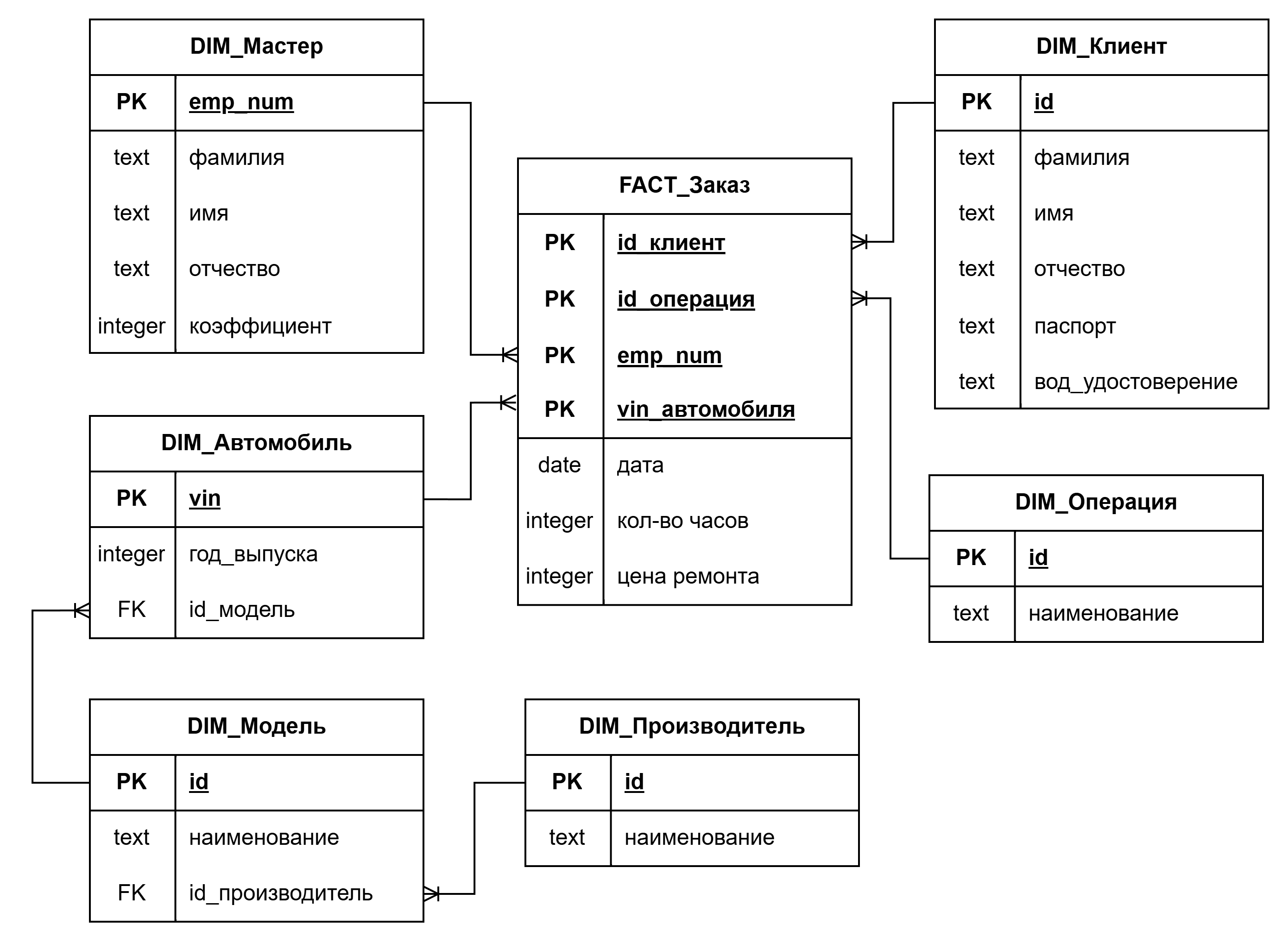


Рисунок 2 – Логическая схема «Снежинка»

Для обработки данных использованы язык программирования Python и библиотека pandas, которые обеспечивают удобную работу с табличными структурами и поддержку различных форматов файлов, включая TXT, JSON, Excel и Parquet. Выбор этих инструментов обусловлен их гибкостью, широкими возможностями для анализа данных и эффективностью в обработке больших объемов информации.

Библиотека pandas используется для загрузки, очистки и преобразования данных. Для работы с файловой системой применяется модуль pathlib, который позволяет удобно задавать пути к файлам. Регулярные выражения из модуля re используются для удаления сокращений в строках и очистки данных.

Загрузка данных выполняется с использованием специализированных функций pandas. Текстовый файл загружается с помощью pd.read\_fwf(), так как он имеет фиксированную ширину столбцов. JSON-файл обрабатывается через pd.read\_json(), поскольку он содержит список объектов. Excel-файл загружается с помощью pd.read\_excel(), при этом обрабатываются все листы, и каждому из них добавляется колонка с датой, соответствующей названию листа.

Предобработка данных включает удаление сокращений, таких как инициалы с точками, из полей last\_name, first\_name, second\_name. Также производится очистка столбца "Цена ремонта", из которого извлекаются только числовые значения, чтобы обеспечить корректную обработку информации о стоимости.

После очистки данные конвертируются в строковый формат (astype(str)) для предотвращения ошибок при сохранении. Итоговые данные экспортируются в формат Parquet с использованием библиотеки fastparquet, что обеспечивает их компактное хранение и быструю загрузку в дальнейшем. По завершении обработки информация о местоположении сохраненных файлов выводится на экран.

# ИСХОДНЫЙ КОД ПРОГРАММЫ

import pandas as pd

import re

from pathlib import Path

import os

# === Пути ===

BASE\_DIR = Path(\_\_file\_\_).resolve().parent

txt\_path = BASE\_DIR / "files" / "original" / "clients"

json\_path = BASE\_DIR / "files" / "original" / "master-emps.json"

excel\_path = BASE\_DIR / "files" / "original" / "master\_tasks.xlsx"

output\_excel\_path = BASE\_DIR / "files" / "processed" / "etl\_result.xlsx"

Path(output\_excel\_path).parent.mkdir(parents=True, exist\_ok=True)

# === Утилиты ===

def remove\_abbreviations(text):

    if isinstance(text, str):

        words = text.split()

        return ' '.join([w for w in words if not re.search(r'\.', w)])

    return text

def read\_txt\_fixed\_width(file\_path):

    raw\_lines = file\_path.read\_text(encoding="utf-8").splitlines()

    cleaned\_rows = []

    for line in raw\_lines:

        if "FULL\_CLIENT\_NAME" in line or "varchar" in line:

            continue

        full\_name = line[0:34].strip()

        passport = line[34:44].strip()

        license = line[44:50].strip()

        vin = line[52:69].strip()

        cleaned\_rows.append({

            "FULL\_CLIENT\_NAME": full\_name,

            "PASSPORT": passport,

            "LICENSE": license,

            "VIN": vin

        })

    return pd.DataFrame(cleaned\_rows)

def read\_and\_fix\_json(path):

    df = pd.read\_json(path)

    for col in ["last\_name", "first\_name", "second\_name"]:

        df[col] = df[col].apply(remove\_abbreviations)

    return df

def read\_excel\_all\_sheets(path):

    sheets = pd.read\_excel(path, sheet\_name=None, dtype=str)

    all\_data = []

    for name, df in sheets.items():

        df.columns = df.columns.str.strip()

        df = df.rename(columns={

            "Мастер": "master",

            "ВИН": "vin",

            "Операция": "operation",

            "кол-часов": "number\_of\_hours",

            "Цена ремонта": "price\_of\_repair",

            "Год": "manufacture\_year",

            "Производитель": "manufacturer",

            "Модель": "model"

        })

        if "price\_of\_repair" in df.columns:

            df["price\_of\_repair"] = df["price\_of\_repair"].str.replace(r"[^\d]", "", regex=True)

            df["price\_of\_repair"] = pd.to\_numeric(df["price\_of\_repair"], errors="coerce")

        df["date"] = name

        all\_data.append(df)

    return pd.concat(all\_data, ignore\_index=True)

# === Загрузка данных ===

txt\_df = read\_txt\_fixed\_width(txt\_path)

json\_df = read\_and\_fix\_json(json\_path)

excel\_df = read\_excel\_all\_sheets(excel\_path)

# === DIM\_Мастер ===

dim\_master = json\_df.rename(columns={

    "emp\_num": "emp\_num",

    "last\_name": "фамилия",

    "first\_name": "имя",

    "second\_name": "отчество",

    "coeff": "коэффициент"

})[["emp\_num", "фамилия", "имя", "отчество", "коэффициент"]].drop\_duplicates()

# === DIM\_Клиент (ID от 1, паспорт как str, добавлено LICENSE) ===

fio\_split = txt\_df["FULL\_CLIENT\_NAME"].str.strip().str.split(" ", expand=True)

fio\_split.columns = ["фамилия", "имя", "отчество"]

dim\_client = pd.concat([txt\_df["PASSPORT"], txt\_df["LICENSE"], fio\_split], axis=1)

dim\_client = dim\_client.rename(columns={

    "PASSPORT": "паспорт",

    "LICENSE": "вод\_удостоверение"

}).drop\_duplicates().reset\_index(drop=True)

dim\_client.insert(0, "id", dim\_client.index + 1)

dim\_client["паспорт"] = dim\_client["паспорт"].astype(str)

client\_id\_map = dim\_client.set\_index("паспорт")["id"].to\_dict()

# === DIM\_Производитель ===

dim\_manufacturer = excel\_df[["manufacturer"]].drop\_duplicates().reset\_index(drop=True)

dim\_manufacturer["id"] = dim\_manufacturer.index + 1

dim\_manufacturer = dim\_manufacturer.rename(columns={"manufacturer": "наименование"})

manufacturer\_lookup = dim\_manufacturer.set\_index("наименование")["id"].to\_dict()

# === DIM\_Модель ===

dim\_model = excel\_df[["model", "manufacturer"]].drop\_duplicates().reset\_index(drop=True)

dim\_model["производитель"] = dim\_model["manufacturer"].map(manufacturer\_lookup)

dim\_model["id"] = dim\_model.index + 1

dim\_model = dim\_model.rename(columns={"model": "наименование"})

model\_lookup = dim\_model.set\_index("наименование")["id"].to\_dict()

# === DIM\_Автомобиль ===

dim\_car = excel\_df[["vin", "manufacture\_year", "model"]].drop\_duplicates()

dim\_car["модель"] = dim\_car["model"].map(model\_lookup)

dim\_car = dim\_car.rename(columns={"vin": "vin", "manufacture\_year": "год\_выпуска"})

dim\_car = dim\_car[["vin", "год\_выпуска", "модель"]].drop\_duplicates()

# === DIM\_Операция ===

dim\_operation = excel\_df[["operation"]].drop\_duplicates().reset\_index(drop=True)

dim\_operation["id"] = dim\_operation.index + 1

dim\_operation = dim\_operation.rename(columns={"operation": "наименование"})

operation\_lookup = dim\_operation.set\_index("наименование")["id"].to\_dict()

# === FACT\_Заказ ===

fact = excel\_df.copy()

fact["id\_операция"] = fact["operation"].map(operation\_lookup)

fact["vin\_автомобиля"] = fact["vin"]

# Привязка мастера

fact["master\_clean"] = fact["master"].apply(remove\_abbreviations)

fact = fact.merge(json\_df[["emp\_num", "last\_name"]].rename(columns={"last\_name": "master\_last"}),

                  left\_on="master\_clean", right\_on="master\_last", how="left")

# Привязка клиента (паспорт как str)

fact = fact.merge(txt\_df[["PASSPORT", "VIN"]], left\_on="vin", right\_on="VIN", how="left")

fact["PASSPORT"] = fact["PASSPORT"].astype(str)

fact["id\_клиент"] = fact["PASSPORT"].map(client\_id\_map)

fact = fact.rename(columns={

    "number\_of\_hours": "кол-во часов",

    "price\_of\_repair": "цена ремонта",

    "date": "дата"

})

fact\_order = fact[["id\_клиент", "id\_операция", "emp\_num", "vin\_автомобиля", "дата", "кол-во часов", "цена ремонта"]]

# === ExcelWriter ===

with pd.ExcelWriter(output\_excel\_path, engine="xlsxwriter") as writer:

    dim\_master.to\_excel(writer, sheet\_name="DIM\_Мастер", index=False)

    dim\_client.to\_excel(writer, sheet\_name="DIM\_Клиент", index=False)

    dim\_manufacturer.to\_excel(writer, sheet\_name="DIM\_Производитель", index=False)

    dim\_model[["id", "наименование", "производитель"]].to\_excel(writer, sheet\_name="DIM\_Модель", index=False)

    dim\_car.to\_excel(writer, sheet\_name="DIM\_Автомобиль", index=False)

    dim\_operation.to\_excel(writer, sheet\_name="DIM\_Операция", index=False)

    fact\_order.to\_excel(writer, sheet\_name="FACT\_Заказ", index=False)

    # Excel форматирование для id\_клиент

    worksheet = writer.sheets["FACT\_Заказ"]

    workbook = writer.book

    text\_fmt = workbook.add\_format({'num\_format': '0'})

    worksheet.set\_column('A:A', 12, text\_fmt)

print(f"\n✅ Готово! Excel-файл со всеми таблицами сохранён в:\n{output\_excel\_path}"

# РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3 – Содержимое DIM\_Мастер

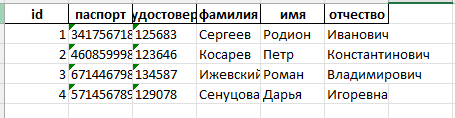


Рисунок 4 – Содержимое DIM\_Клиент

Изображение выглядит как текст, снимок экрана, число, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 5 – Содержимое DIM\_Производитель

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6 – Содержимое DIM\_Модель

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 7 – Содержимое DIM\_Автомобиль

Изображение выглядит как текст, снимок экрана, число, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 8 – Содержимое DIM\_Операция

Изображение выглядит как текст, снимок экрана, число, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 9 – Содержимое FACT\_Заказ

# ВЫВОД

В ходе выполнения лабораторной работы была успешно реализована аналитическая витрина данных в схеме «Снежинка». Были проведены все этапы ETL-процесса: анализ и нормализация структуры исходных данных, их очистка и трансформация, формирование таблиц измерений и таблицы фактов. Применение языка Python и библиотеки pandas позволило эффективно обрабатывать данные различных форматов (TXT, JSON, Excel) и обеспечить автоматизацию всех шагов.

Построенная модель соответствует требованиям аналитических систем: нормализованные измерения позволяют избежать дублирования информации и поддерживают расширяемость структуры, а таблица фактов содержит все ключевые параметры для последующего анализа заказов на ремонт. Конечный результат — экспорт всех таблиц в единый Excel-файл — обеспечивает удобство последующего использования и визуализации данных. Полученные навыки и применённые инструменты соответствуют современным требованиям к аналитике данных в бизнес-среде.