Департамент образования и науки города Москвы

Государственное автономное образовательное учреждение высшего образования города Москвы

«Московский городской педагогический университет»

Институт цифрового образования

Департамент информатики, управления и технологий

ДИСЦИПЛИНА:

Распределенные системы

**Лабораторная работа 3.1**

**Проектирование архитектуры хранилища данных**

Выполнил(а): Шведова С.С., группа: АДЭУ-211

Преподаватель: Босенко Т.М.

Москва

2024

**Цель работы**: разработать архитектуру хранилища больших данных для заданного сценария использования.

**Задача:** создать архитектуру хранилища больших данных для компании, занимающейся автомобилями.

**Определение требований**

Крупная автомобильная компания

- Объем данных: 500 ТБ в год, рост 40% ежегодно.

- Скорость получения: до 5000 событий в секунду.

- Типы данных: 50% структурированные, 40% полуструктурированные, 10% неструктурированные

- Требования к обработке: анализ данных с подключенных автомобилей, оптимизация производственных процессов

- Доступность: 99.95%, время отклика

- Безопасность: шифрование, соответствие 152-ФЗ и международным автомобильным стандартам

**Выбор модели хранилища данных**

Для выбранной компании подойдет модель Lakehouse, которая объединяет преимущества Data Lake и Data Warehouse, предоставляя возможность хранить разнообразные типы данных (структурированные, полуструктурированные, неструктурированные) и обеспечивая требуемый уровень доступности, безопасности и масштабируемости. Lakehouse поддерживает как аналитику в режиме реального времени, так и сложные аналитические запросы.

**Компоненты архитектуры**

1. **Источники данных:**

- Отзывы и обратная связь клиентов, социальные сети.

- Веб-сайты, мобильные приложения и камеры видеонаблюдения.

- Данные из систем учета и CRM.

**2. Слой сбора данных**

- Apache Kafka для обработки потоков данных

- Пользовательские коннекторы для CRM

- Logstash для сбора логов.

**3. Слой хранения данных**

- PostgreSQL для структурированных данных и метаданных.

- HDFS для работы с большими данными в экосистеме

- MongoDB для хранения полуструктурированных данных

**4. Слой обработки данных**

- Apache Spark является фреймворком для обработки и анализа больших объемов данных.

- Apache Flink является потоковым обработчиком данных, который может обрабатывать большие объемы событий с высокой скоростью.

- Microsoft Azure Stream Analytics для обработки данных в режиме реального времени с использованием облачных ресурсов

**5. Слой аналитики и машинного обучения**

- Power BI для визуализации и дашбордов

- PyCharm для интерактивной аналитики

- TensorFlow - это открытая платформа для машинного обучения

**6. Слой управления данными**

- Apache Ranger для контроля доступа и аудита.

- Apache Atlas для управления метаданными.

**7. Слой оркестрации и мониторинга**

- Apache Airflow предоставляет инструменты для оркестрации сложных вычислительных потоков данных.

- Zavgar Online это интернет-платформа для управления и мониторинга автопарка

На рисунке 1 изображена архитектура хранения данных «Звезда» для процессов.

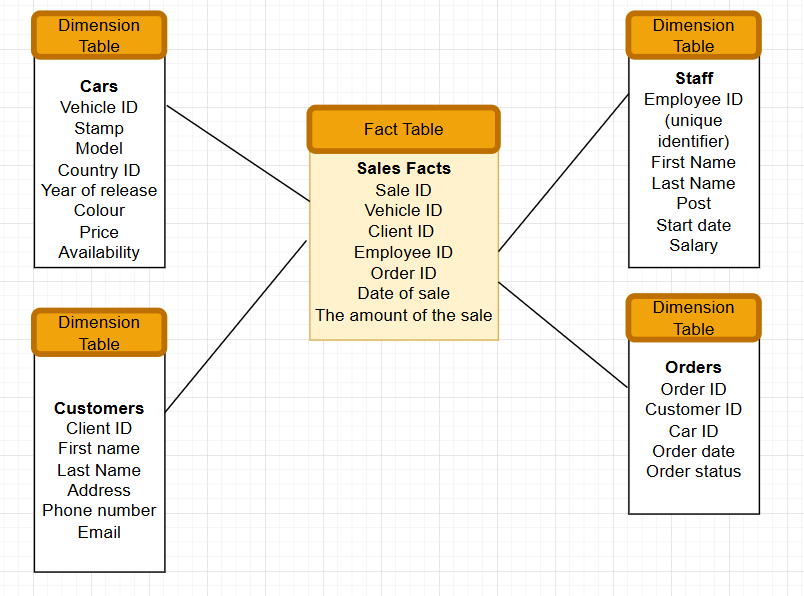


Рисунок 1. Архитектура хранения данных «Звезда»

На рисунке 2 изображена архитектура хранения данных «Снежинка» для самих данных.

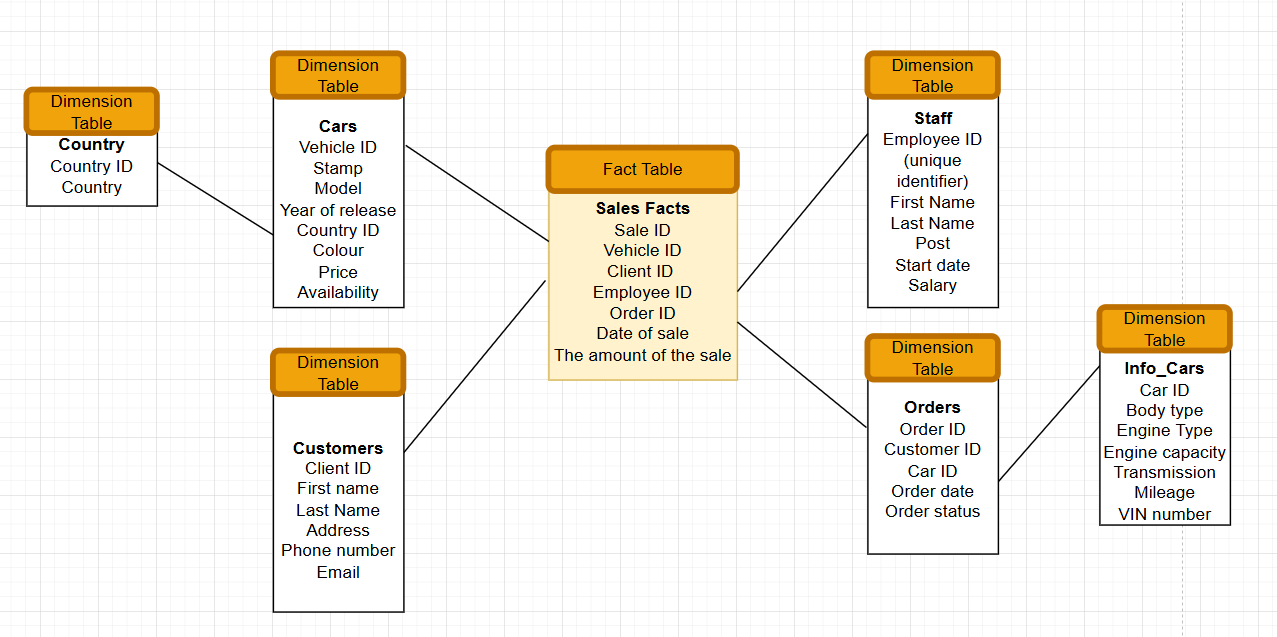


Рисунок 2. Архитектура хранения данных «Снежинка»

На рисунке 3 изображена схема архитектуры.

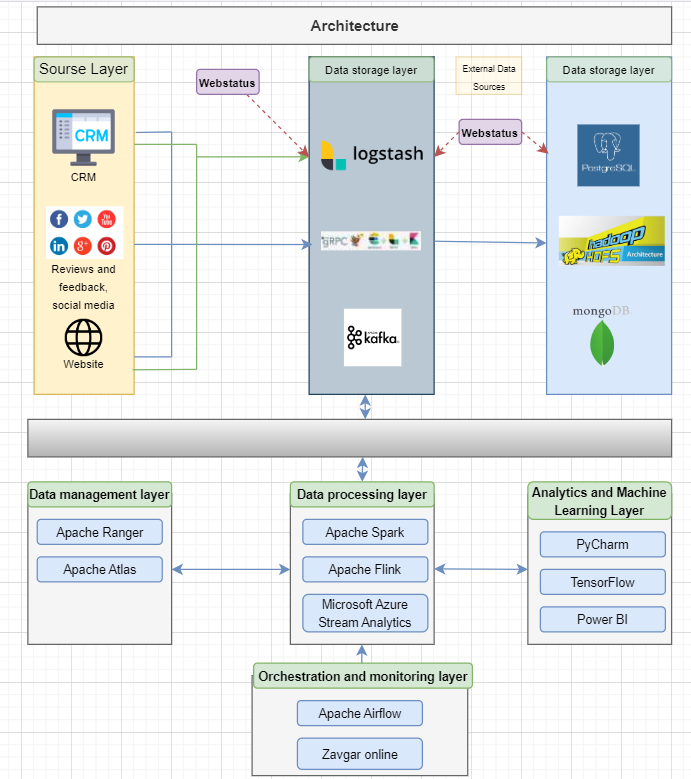


Рисунок 3. Схема архитектуры

**Процесс обработки данных**

1. Процесс обработки данных начинается с сбора информации из различных источников, включая социальные сети для неструктурированных данных, веб-сайты и мобильные приложения для полуструктурированных данных, и CRM системы для структурированных данных;
2. Пакетные задачи выполняются с помощью Spark;
3. Аналитики используют Jupyter Notebooks и Power BI для исследования данных и создания отчетов.
4. Модели машинного обучения обучаются на исторических данных (OLAP)

**Масштабирование и отказоустойчивость**

1. Apache Kafka для обработки потоков данных;
2. Пользовательские коннекторы для CRM;
3. Logstash для сбора логов.
4. Использование кластерной архитектуры Hadoop для горизонтального масштабирования.
5. Использование Kubernetes для оркестрации и автоматического масштабирования микросервисов.

**Безопасность**

1. Применение Apache Ranger для детального контроля доступа к данным.
2. Регулярное резервное копирование и план аварийного восстановления.

**Вывод:**

Созданная архитектура хранилища данных обеспечивает возможности обработки и хранения растущего объема разнообразной информации, а также обеспечивает масштабируемость, отказоустойчивость, безопасность и удовлетворение требований анализа данных для оптимизации производственных процессов.