Modern Storages and Data Warehousing
Week 5 - Queues

Попов Илья, i.popov@hse.ru

Recap прошлых занятий

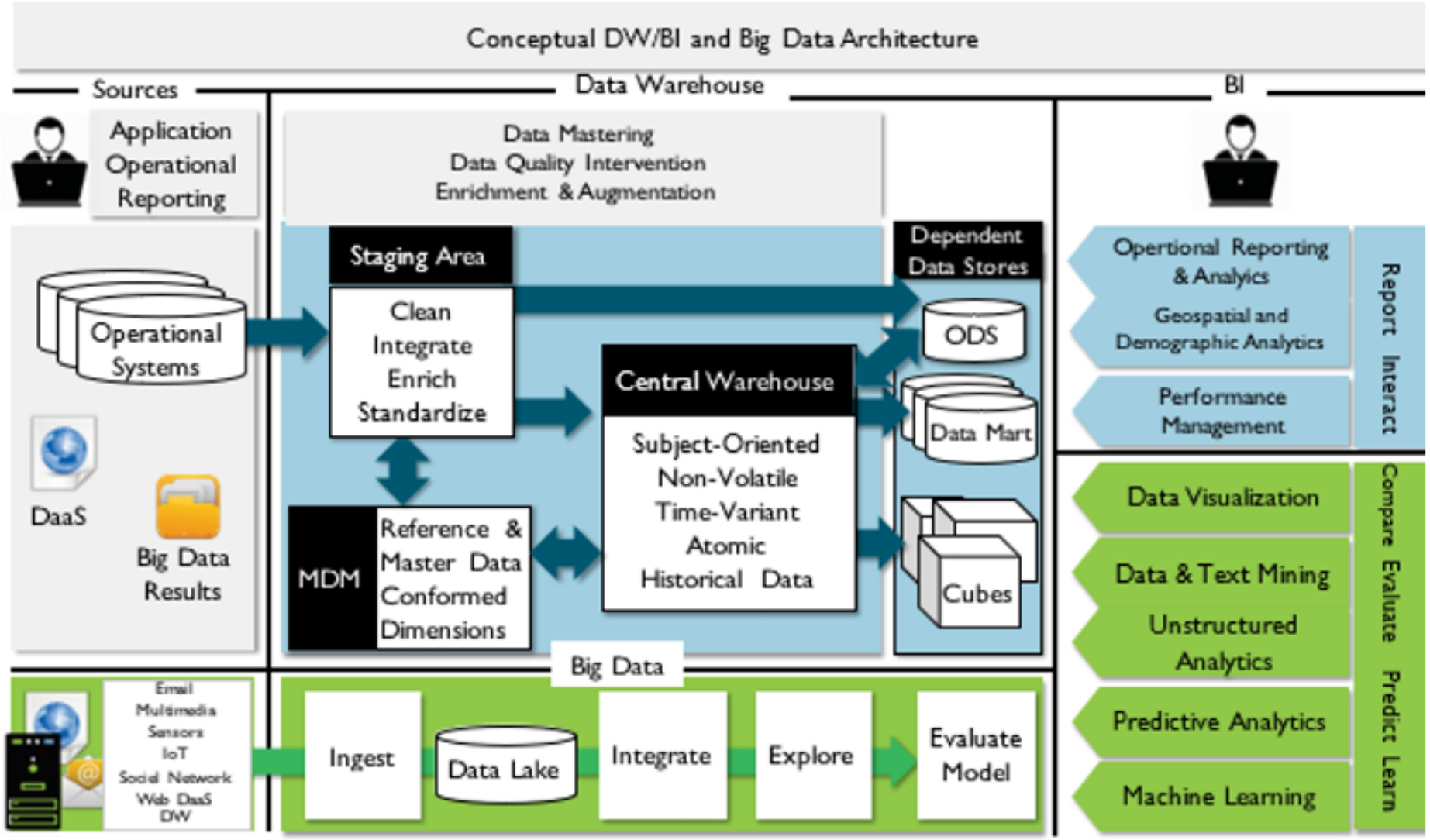


Figure 5: Date Warehouse Concept

1 - Очереди

Концепция Message broker

Message broker - архитектурный паттерн в распределённых системах. Приложение, которое преобразует сообщение по одному протоколу от приложения-источника в сообщение протокола приложения-приёмника, тем самым выступая между ними посредником.

Паттерн позволяет создать буфер, который может коммуницировать с различными системами по унифицированным протоколам, создавать канал коммуникации между приложениями или системами.

Концепция Message broker

Типы Message broker-ов:

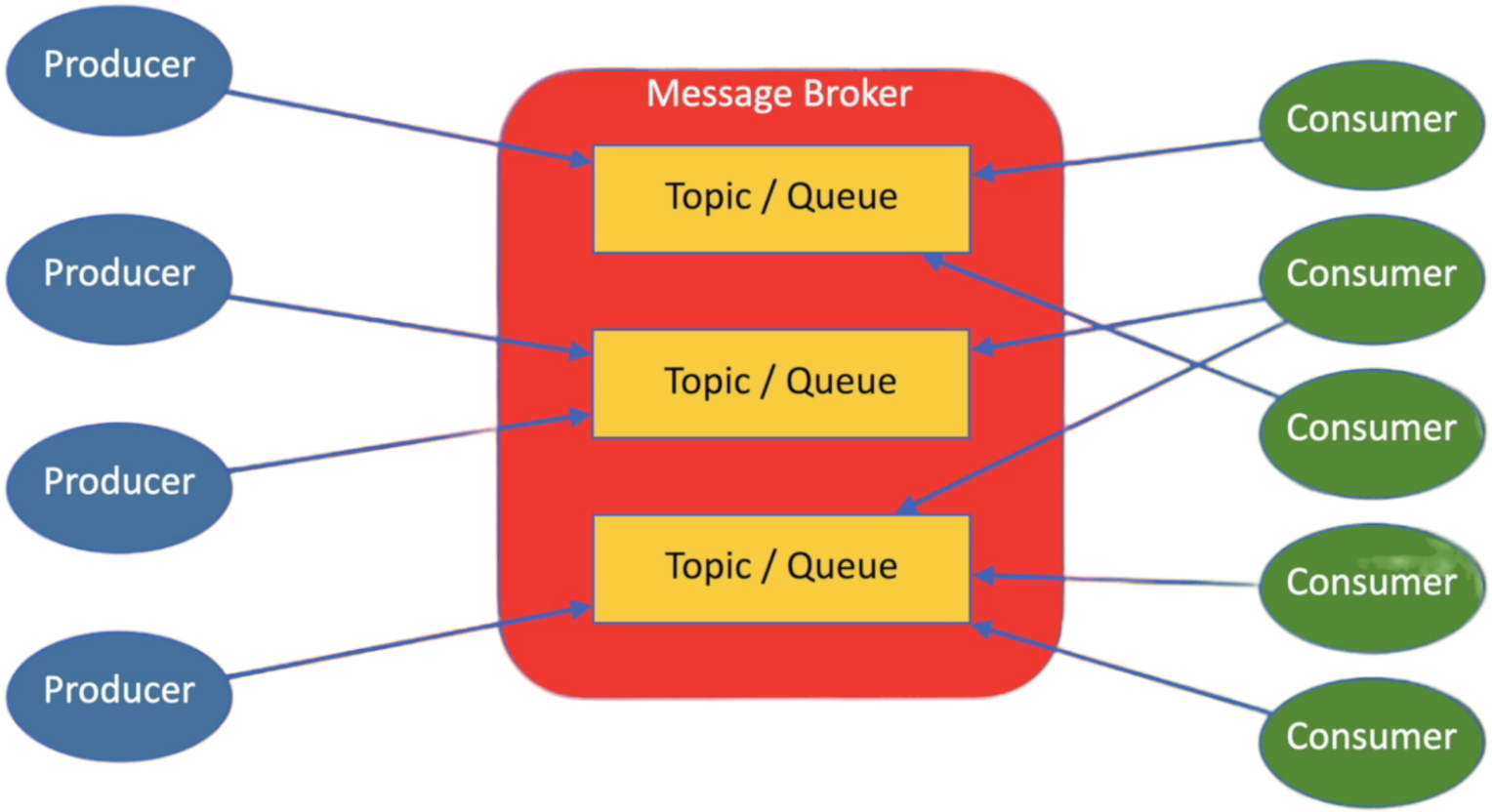
〉 **point-to-point**: брокеры, которые работают по принципу доставки конкретного сообщения. Применяется message-ориентированный подход, который основан на гарантированной доставке сообщений и строгой последовательности. Выполняется в виде очереди FIFO, в которую одна система пишет сообщения, а другая система вычитывает эти сообщения;

〉 **publish/subscribe**: источники (producer-ы) публикуют свои изменения, а потребители (consumer-ы) получают эти сообщения по подписке. Нет гарантии строгой последовательности, но являются более масштабируемыми.

Назначение Message broker

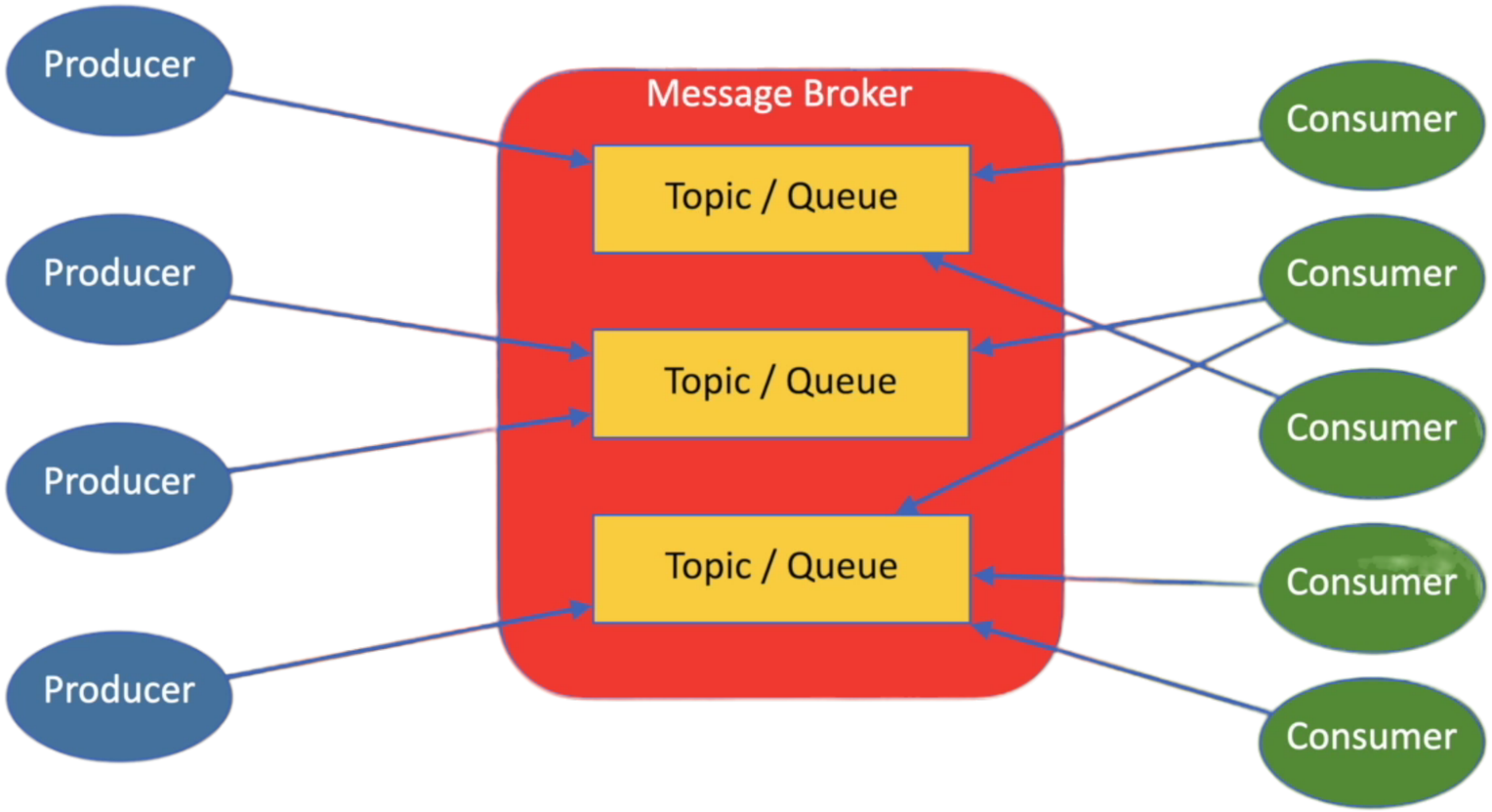
- **Интеграция систем с разными протоколами** – можем использовать различные языки, т.к. для большинства МВ существуют клиентские библиотеки, которые позволяют общаться с МВ
- **Роутинг сообщений** – можем настраивать правила отправки сообщений
- **Надёжное хранение** – при правильной настройке отправленное сообщение не будет потеряно
- **Гарантированная доставка** – гарантировано, что сообщение будет доставлено, но не все МВ гарантируют, что единожды отправленное сообщение не будет получено приёмником несколько раз
- **Масштабирование (как источников, так и потребителей)** – должны позволять подключать большее количество источников и потребителей
- **Преобразование сообщений** – сообщение может быть преобразовано внутри МВ (например, можно шифровать, маскировать сообщения)
- **Интеграция с внешними системами** – МВ может выступать не просто как канал передачи данных, но и взаимодействовать с внешними системами, и по результату каким-то образом проверить, обогатить, правильно маршрутизировать сообщение

Общая схема Message broker



- **Topic / Queue** – логический канал передачи информации
- **Producer** – системы-источники информации подключаются к MB и пишут свои сообщения в Topic (Queue)
- **Consumer** – системы-получатели либо самостоятельно выбирают информацию из Queue, либо подписываются на изменения данных в Topic

Общая схема Message broker



- Несколько Producer-ов могут писать в один Topic / Queue
- Несколько Consumer-ов могут получать данные из одного Topic / Queue
- Consumer может подписаться на несколько Topic / Queue

Apache Kafka

- Изначально - внутренний проект в LinkedIn
- Вышла в opensource в 2011 году
- Названа в честь Франца Кафки ¯_(ツ)_/¯

Apache Kafka

- Изначально - внутренний проект в LinkedIn
- Вышла в opensource в 2011 году
- Названа в честь Франца Кафки ¯_(ツ)_/¯

- Работает по модели publish/subscribe
- Имеет распределенную и гибко масштабируемую архитектуру

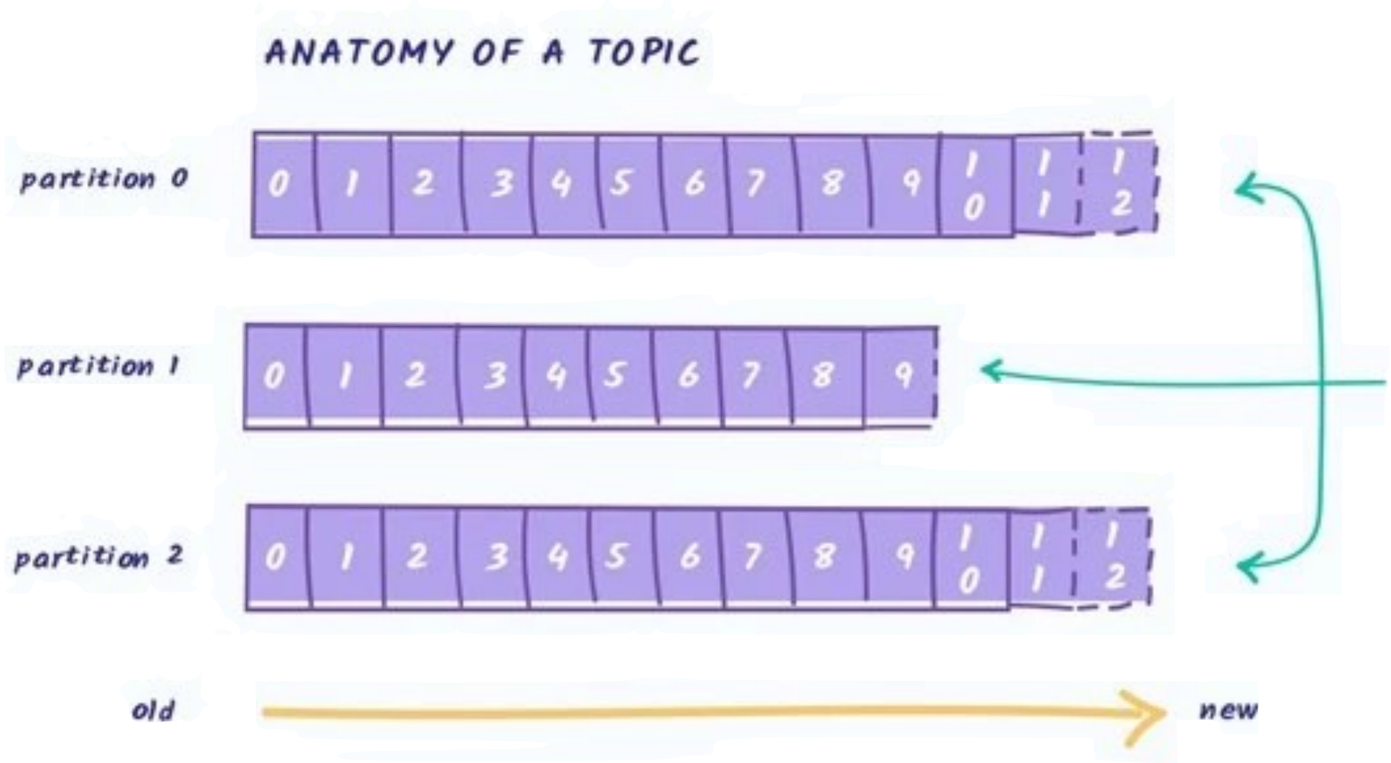
Apache Kafka

- Изначально - внутренний проект в LinkedIn
- Вышла в opensource в 2011 году
- Названа в честь Франца Кафки ¯_(ツ)_/¯

- Работает по модели publish/subscribe
- Имеет распределенную и гибко масштабируемую архитектуру

- Сообщения представляют собой произвольный набор байтов
- Сообщения группируются в topic
- Внутри топиков: key-value структура
- Настраиваются Log Retention и Cleanup Policy

Устройство Kafka

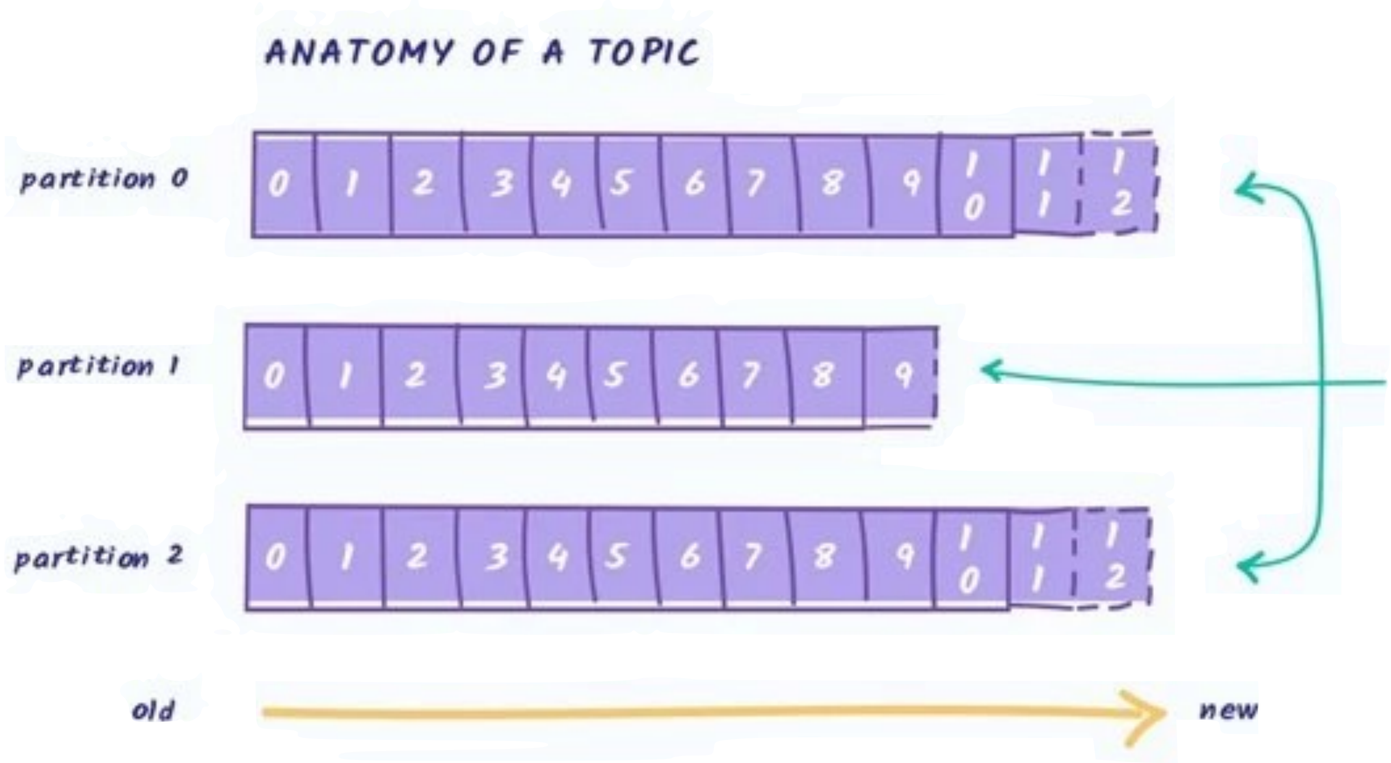


Topic:

〉 логическая append-only очередь сообщений (message): можем только добавлять сообщения, а удалить ошибочное сообщение не можем

〉 состоит из 1+ партиций (partition): для максимального распараллеливания обработки сообщений как с точки зрения записи, так и чтения

Устройство Kafka



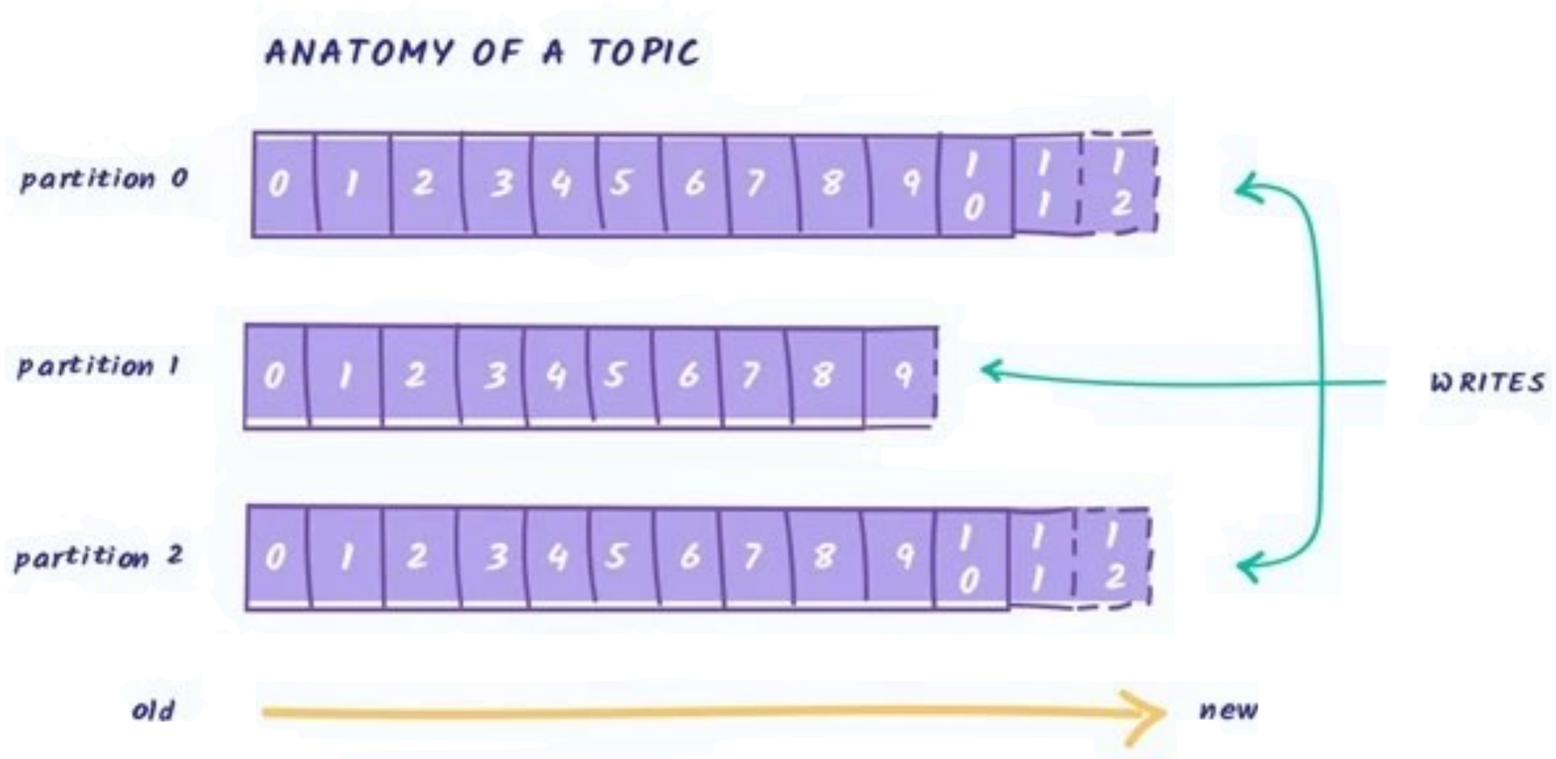
Partition:

〉 физическая единица хранения данных topic: состоит из файлов, которые хранятся на сервере

〉 в partition можно писать, но нельзя удалять

〉 в конкретный момент времени жёстко привязана к конкретному broker-у. Есть репликация, но только одна partition будет active, остальные follower

Устройство Kafka

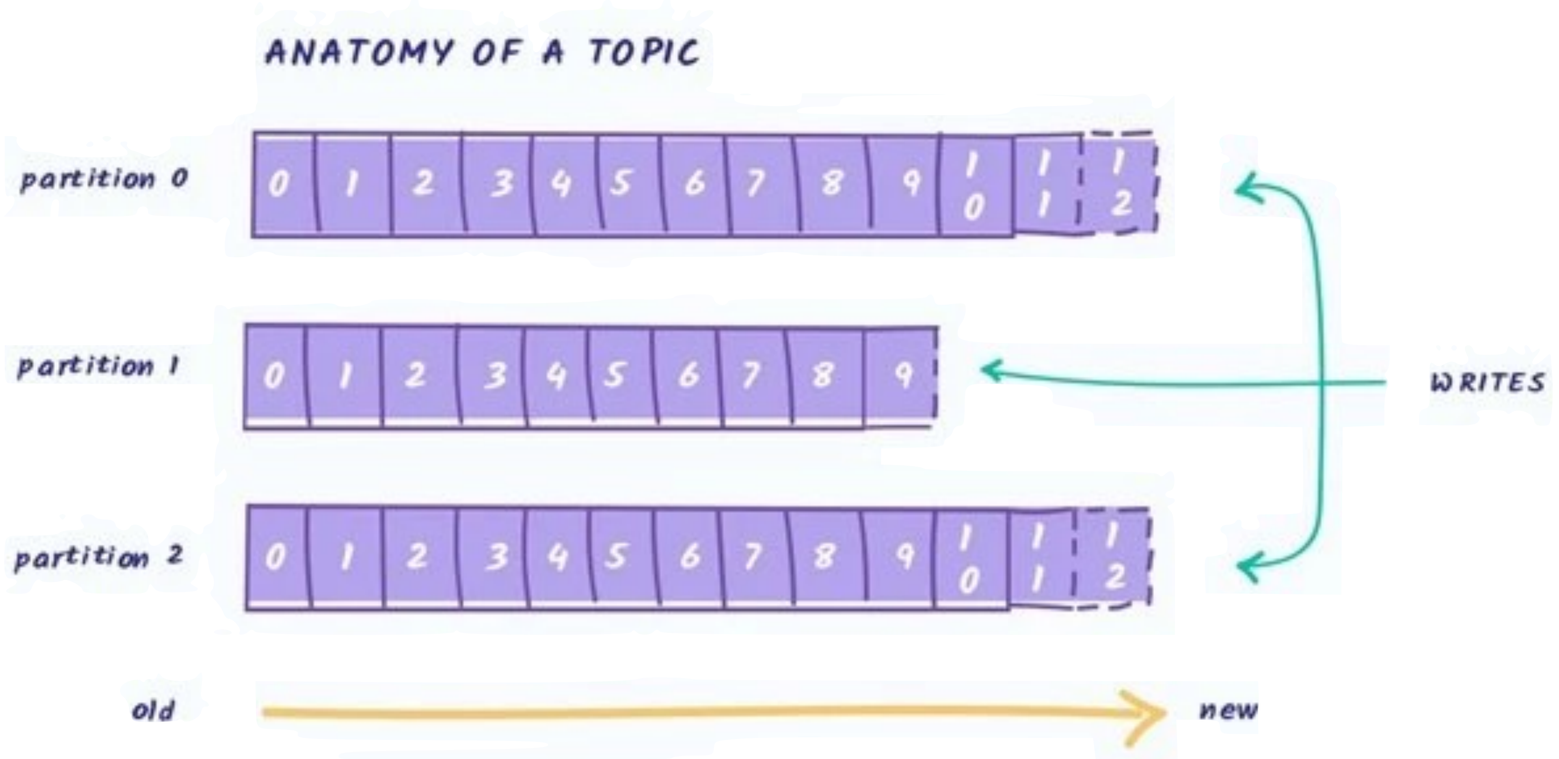


Запись и чтение из partition:

〉 Если используется ключ для сообщения, то для определения партиции используется hash(key) % кол-во партиций
Consumer, подключаясь к topic, понимает топологию
внутри topic и подключается к одной или нескольким partition

〉 Если не используется ключ для сообщения, то распределение между partition происходит по принципу round-robin – каждое следующее сообщение пишется в следующую partition и далее по кругу.

Устройство Kafka



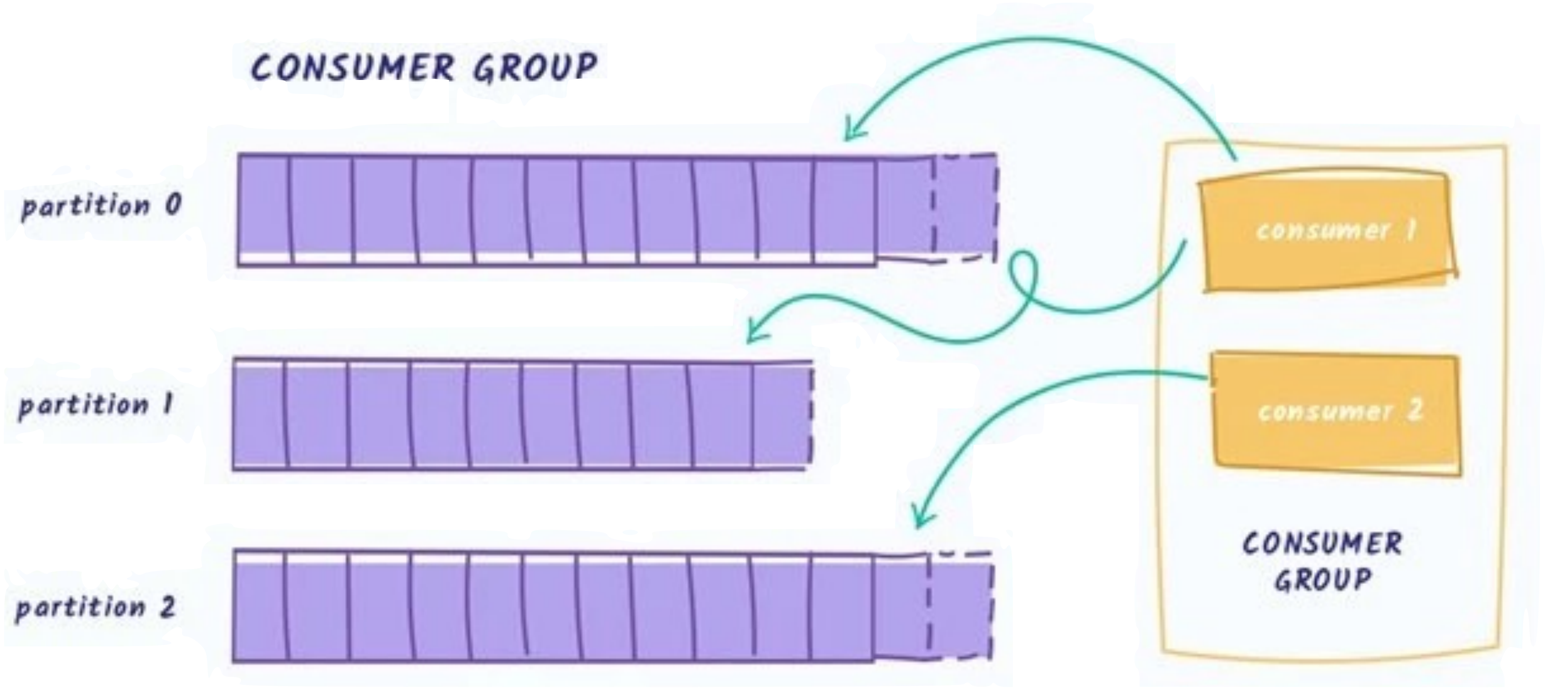
〉 Навигация по topic производится на основе offset - инкрементальный идентификатор события внутри каждой партиции

〉 Все сообщения нумеруются, и мы получаем сообщения в том порядке, в котором они были пронумерованы

〉 Offset не является глобальной величиной для topic, а работает на уровне партиции

〉 Если topic состоит из более чем одной партиции, то нет гарантии строгой последовательности получения сообщений

Устройство Kafka

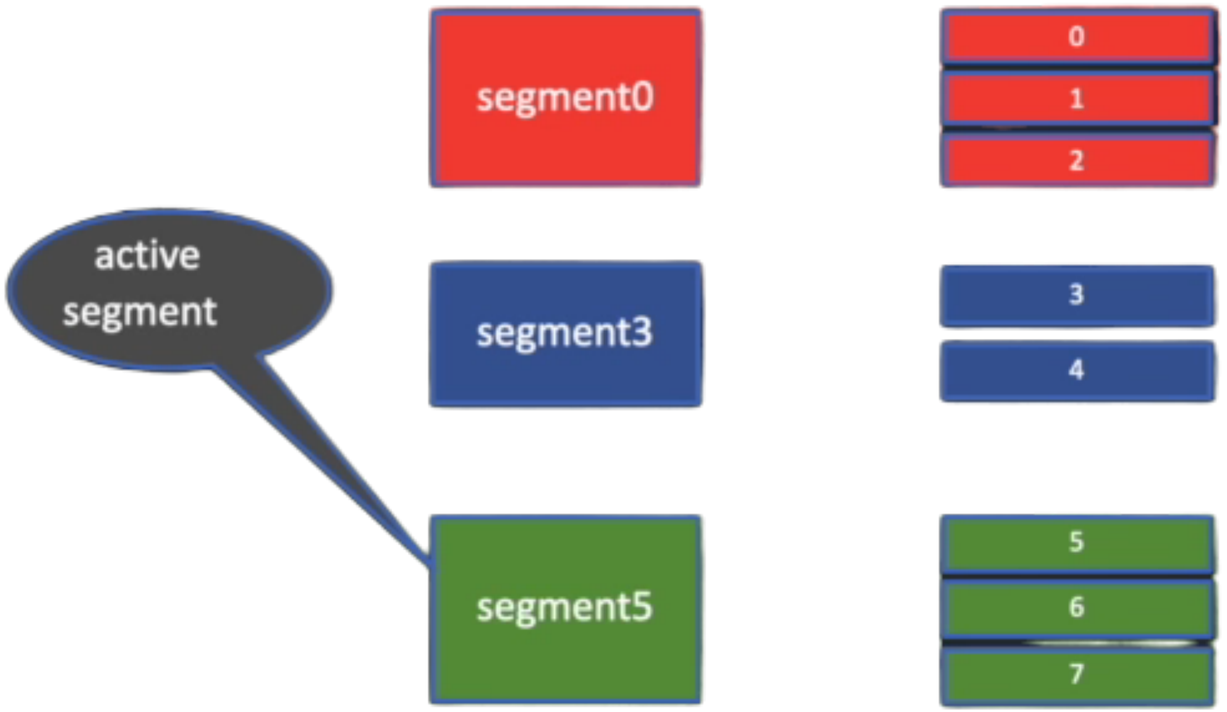


〉 Consumer’ы читают данные из партиций напрямую

〉 При чтении из партиции consumer делает **коммит оффсета**. Это необходимо для того, чтобы, если, например, текущий читатель упадёт, то следующий (новый читатель) начнёт с последнего коммита

〉 Consumer’ы объединяются в **consumer group**. При добавлении нового читателя или падении текущего, группа перебалансируется. Объединение в группы гарантирует, что сообщение внутри группы будет прочитано один раз

Устройство Kafka



Segment:

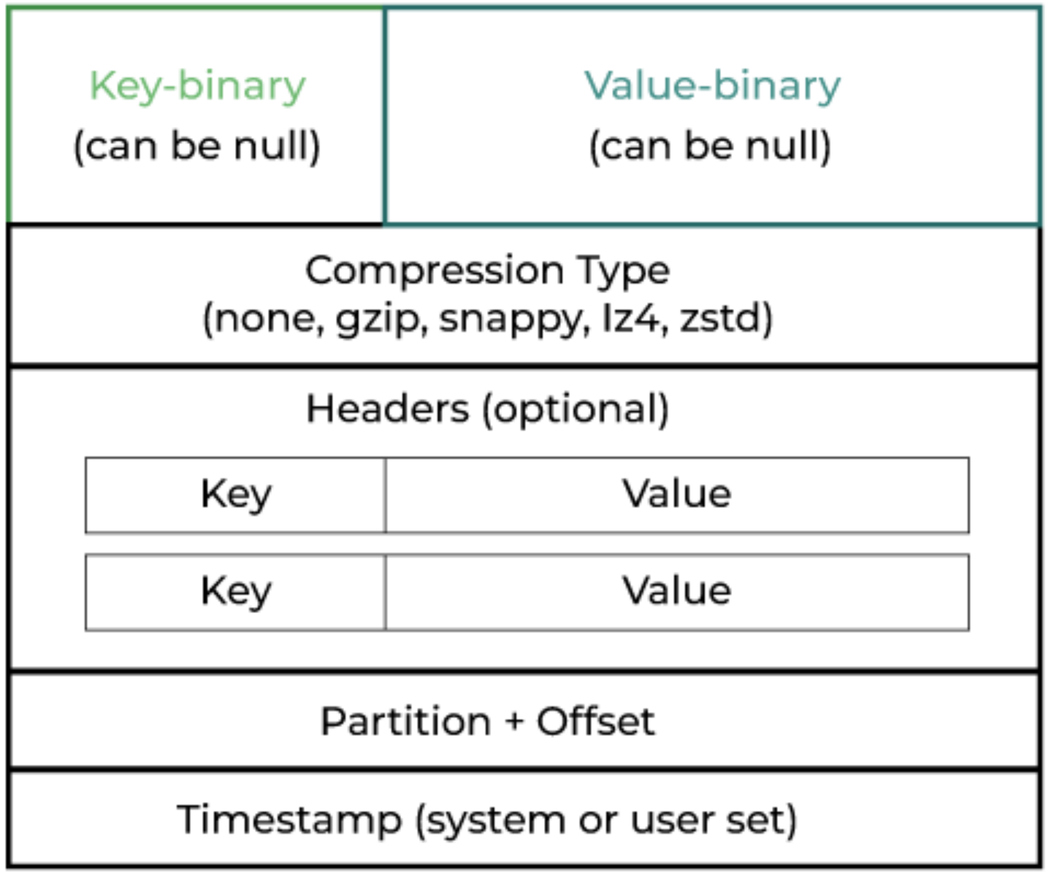
〉 Набор записей внутри partition

〉 Физически являются файлами в файловой системе

〉 Всегда есть один **active segment**, в который идёт запись новых message

〉 При превышении размера сегмента создаётся новый, который становится активным

Сообщение Kafka



Message:

〉 Headers:

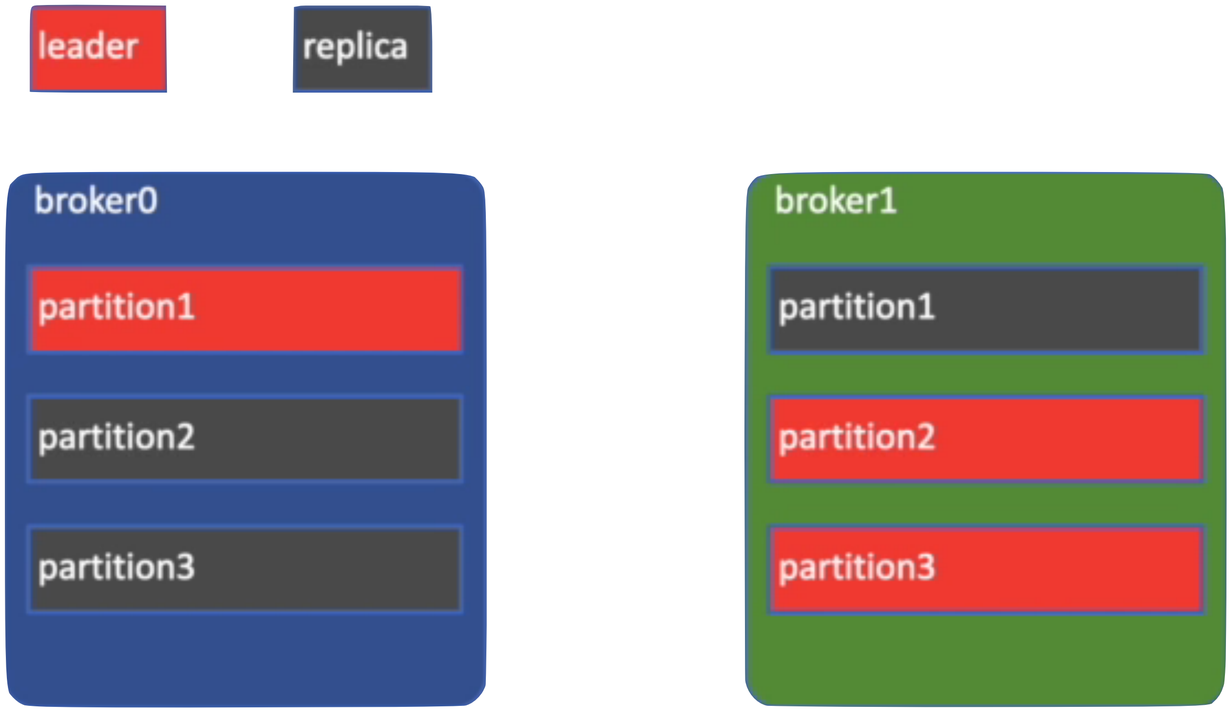
- Topic (название)
- Partition (партиция, из которой сообщение было вычитано)
- Offset (смещение внутри партиции)
- Timestamp (когда сообщение было доставлено в Kafka)
- Compression type

〉 Additional headers: дополнительные заголовки (Map[String, String]),

〉 Body: тело сообщения:

- key – bytes
- value – bytes

Структура кластера



Broker – каждый из серверов в Kafka:

〉 Обслуживает topic partitions

〉 Возможна репликация (leader + followers)

〉 Можно иметь несколько копий партиций в каждом топике

Демо 1

2 - Debezium

Вспоминаем прошлую лекцию

Особенности CDC на журналах транзакций:

Плюсы:

〉Асинхронный метод

〉Нет дополнительной нагрузки на СУБД

Минусы:

〉Отражает последовательность транзакций

〉**Сложно использовать из-за разных форматов журналов**

Решение проблемы

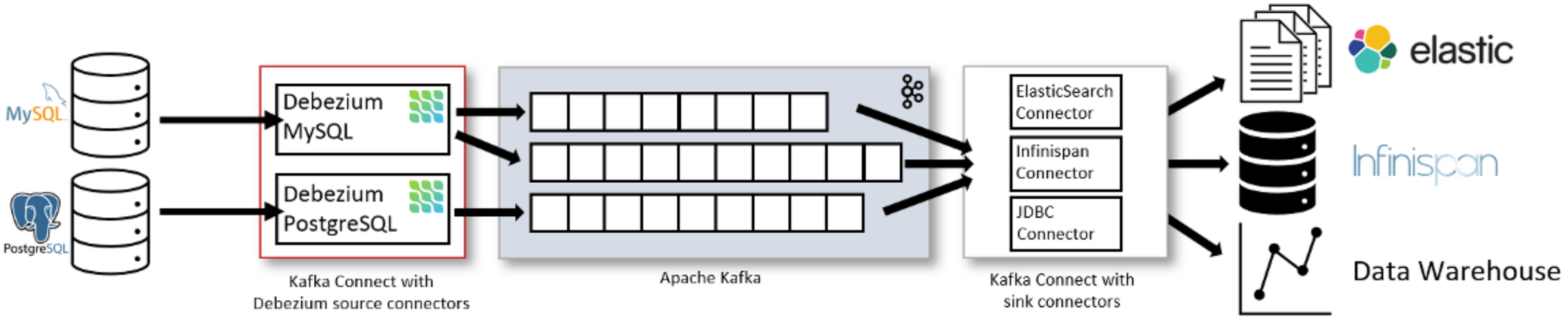
Давайте придумаем тулзу, которая умеет читать журналы транзакций популярных СУБД, и будет их преобразовывать в один формат ¯_(ツ)_/¯

Решение проблемы

Давайте придумаем тулзу, которая умеет читать журналы транзакций популярных СУБД, и будет их преобразовывать в один формат ¯_(ツ)_/¯

И такую тулзу придумали
Она называется **Debezium**

Как работает Debezium



Демо 2