1 Добро пожаловать в Kafka Streams

В этой главе

- ✓ Определение событий и потоковой обработки событий.
- ✓ Введение в Kafka — платформу потоковой обработки событий.
- ✓ Конкретный пример применения платформы.

Постоянный приток данных порождает все больше возможностей для потребителя, и все чаще пользователями этих данных становятся программные системы, обращающиеся к другим программным системам. Давайте посмотрим, например, как работает ваше любимое приложение потокового вещания, когда вы смотрите фильмы. Вы входите в приложение, выбираете фильм, смотрите его, а затем можете поставить свою оценку и оставить свой отзыв о фильме. Эта простая череда взаимодействий генерирует несколько событий, фиксируемых службой потокового вещания. Но эта информация нуждается в анализе, чтобы быть полезной для бизнеса. И здесь в игру вступает все остальное программное обеспечение.

1.1. ПОТОКОВАЯ ОБРАБОТКА СОБЫТИЙ

Программные системы потребляют и хранят всю информацию, полученную в процессе взаимодействий с вами и другими подписчиками. Затем другие программные системы используют эту информацию, чтобы предложить вам рекомендации и передать потоковому сервису информацию о том, что от него может потребоваться в будущем. Теперь представьте, что этот процесс происходит сотни тысяч или даже миллионы раз в день, и вы поймете, какой огромный объем информации должно получить

и обработать программное обеспечение, чтобы предприятие могло соответствовать требованиям и ожиданиям клиентов и оставаться конкурентоспособным.

С другой стороны, любые действия современных потребителей, от просмотра фильма онлайн до покупки пары обуви в обычном магазине, генерируют *события*. Чтобы организация выжила и преуспела в нашей цифровой экономике, у нее должен иметься эффективный способ получения этих событий и реагирования на них. Другими словами, компании должны искать способы идти в ногу с этим бесконечным потоком событий, если хотят удовлетворить клиентов и обеспечить устойчивый рост прибыли. Этот бесконечный поток разработчики так и называют — *потоком событий*. И все чаще для удовлетворения спроса на эту бесконечную цифровую деятельность привлекают *платформы потоковой обработки событий*, которые используют последовательность приложений потоковой обработки событий.

Платформа потоковой обработки событий подобна нашей центральной нервной системе, которая обрабатывает миллионы событий (нервных сигналов) и в ответ посылает сообщения соответствующим частям тела. Наши сознательные мысли и действия являются ответами на некоторые из этих сообщений. Когда мы голодны и открываем холодильник, центральная нервная система получает сообщение и посылает руке приказ потянуться за красивым красным яблоком на первой полке. Другие действия, такие как учащение пульса в ожидании захватывающих новостей, обрабатываются бессознательно.

Платформа потоковой обработки фиксирует события, генерируемые мобильными устройствами, взаимодействием клиентов с веб-сайтами, онлайн-активностью, отслеживанием поставок и другими бизнес-транзакциями. Но платформа, как и нервная система, делает больше, чем просто фиксирует события. Ей также нужен механизм для надежной передачи и хранения информации из этих событий в том порядке, в котором они произошли. Затем другие приложения могут обрабатывать или анализировать события, чтобы извлекать различные биты этой информации.

Обработка потока событий в реальном времени имеет большое значение для принятия решений, чувствительных к времени. Например, не выглядит ли подозрительной эта покупка у клиента X? Указывают ли сигналы от этого датчика температуры, что что-то в производственном процессе пошло не так? Была ли отправлена информация о маршрутизации в соответствующий отдел компании?

Но ценность платформы потоковой обработки событий не связана с немедленным получением информации. Наличие надежного хранилища позволяет нам вернуться и исследовать поток событий в первоначальном необработанном виде, выполнить некоторые манипуляции с данными для их более глубокого исследования или воспроизвести последовательность событий, чтобы попытаться понять, что привело к определенному результату. Например, сайт электронной коммерции предлагает фантастическую скидку на некоторые товары в выходные после большого праздника. Реакция на распродажу настолько сильная, что приводит к сбою нескольких серверов и останавливает работу бизнеса на несколько минут. Воспроизводя произошедшие события, инженеры могут лучше понять, что вызвало сбой и как исправить систему, чтобы она могла справиться с большим внезапным наплывом покупателей.

Итак, где нужны приложения потоковой обработки событий? Поскольку все в жизни можно считать событием, любая предметная область выиграет от потоковой

обработки событий. Но есть области, где это особенно важно. Вот несколько типичных примеров.

- *Мошенничество с платежными картами* — владелец платежной карты может не заметить ее кражи, но путем анализа покупок и сравнения их с устоявшимися паттернами (местоположение, общий характер потребительских расходов) можно заметить кражу платежной карты и оповестить ее владельца.
- *Обнаружение вторжений* — возможность выявлять аномальное поведение в режиме реального времени имеет решающее значение для защиты конфиденциальных данных и благополучия организации.
- *Интернет вещей* — при использовании технологии Интернета вещей датчики размещаются в самых разных местах и часто отправляют данные. Возможность быстро собирать и осмысленно обрабатывать эти данные имеет большое значение, а отсутствие такой возможности снижает эффект от развертывания этих датчиков.
- *Финансовый сектор* — для принятия удачных решений о покупке или продаже брокерам и потребителям необходимо иметь возможность отслеживать рыночные цены и тенденции в режиме реального времени.
- *Обмен данными в режиме реального времени* — крупным организациям, таким как корпорации или холдинги, имеющим множество приложений, необходимо обмениваться данными стандартным способом и в режиме реального времени.

Итог: если поток событий несет важную и полезную информацию, то предприятиям и организациям необходимы событийно-ориентированные приложения, чтобы извлечь выгоду из полученной информации.

Но потоковые приложения подходят не для всех ситуаций. Они становятся необходимыми, когда данные находятся в разных местах или генерируется большой объем событий, требующих распределенных хранилищ данных. Поэтому, если есть возможность обойтись одним экземпляром базы данных, то потоковая обработка не нужна. Например, небольшой бизнес электронной коммерции или сайт городской администрации с преимущественно статическими данными не лучшие кандидаты для использования потоковой обработки событий.

В этой книге вы познакомитесь с особенностями потоков событий, поймете, когда и почему они нужны и как использовать платформу потоковой обработки событий Kafka для создания надежных и отзывчивых приложений. Узнаете, как использовать различные компоненты платформы Kafka для захвата событий и передачи их другим приложениям. Мы вместе рассмотрим использование компонентов платформы для выполнения простых действий, таких как запись (производство) или чтение (потребление) событий, в расширенных приложениях с состоянием, требующих сложных преобразований, чтобы вы знали, как можно решать бизнес-задачи с помощью подходов, основанных на потоковой обработке событий. Эта книга подойдет любому разработчику, который хочет заняться созданием приложений потоковой обработки событий.

Название книги «Kafka Streams» дает понять, что основное внимание в ней уделяется Kafka Streams, но вообще эта книга описывает всю платформу Kafka от начала до конца. Эта платформа включает ряд важнейших компонентов, таких как производители, потребители и схемы, с которыми вам придется поработать перед созданием своих потоковых приложений, и о них я расскажу в первой части. Таким

образом, мы не будем углубляться конкретно в Kafka Streams до конца второй части, то есть до конца главы 6.

Но такое расширенное знакомство стоит потраченных времени и сил. Kafka Streams — это абстракция, выстроенная поверх компонентов платформы потоковой обработки событий Kafka, поэтому понимание этих компонентов поможет вам лучше осознать, как можно использовать Kafka Streams.

1.2. ЧТО ТАКОЕ СОБЫТИЕ

Итак, мы определили поток событий, но что такое само событие? Мы определим событие просто как «что-то, что произошло» (https://www.merriam-webster.com/dictionary/event). Термин *«событие»* часто вызывает у многих ассоциации с чем-то *примечательным*, например с рождением ребенка, свадьбой или спортивным событием, однако мы сосредоточимся на более мелких, более обыденных событиях, таких как совершение покупки клиентом (онлайн или очно), щелчок кнопкой мыши на ссылке или передача данных датчиком. Генерировать события могут как люди, так и машины. Именно последовательность событий и их постоянное течение составляют поток событий.

Концептуально события содержат три основных компонента:

- *ключ* — идентификатор события;
- *значение* — само событие;
- *отметка времени* — когда произошло событие.

Обсудим каждый из них более подробно. Ключ может служить идентификатором события и, как мы узнаем в последующих главах, используется для маршрутизации и группировки событий. Вообразите онлайн-покупку: идентификатор клиента — отличный пример ключа события покупки. Значение — это информация о событии. Значение может быть, например, сигналом от датчика, сообщающим, что кто-то открыл дверь, или результатом какого-либо действия, например покупки товара в онлайн-магазине. Наконец, отметка времени — это дата и время, определяющие, когда произошло событие. Далее практически во всех главах мы неизменно будем сталкиваться со всеми тремя компонентами событий.

В этом введении я использовал много разных терминов, поэтому завершу раздел таблицей определений (табл. 1.1).

Таблица 1.1. Определения

Событие	Что-то, что произошло, и сопутствующие характеристики произошедшего
Поток событий	Серия событий, зафиксированных в режиме реального времени из таких источников, как мобильные устройства или устройства Интернета вещей
Платформа потоковой обработки событий	Программное обеспечение для обработки потоков событий, способное создавать, потреблять, обрабатывать и хранить потоки событий
Apache Kafka	Ведущая платформа потоковой обработки событий, предоставляющая все компоненты платформы потоковой обработки событий в одном проверенном решении
Kafka Streams	Библиотека потоковой обработки событий для Kafka

1.3. ПРИМЕР ПОТОКА СОБЫТИЙ

Допустим, вы купили потоковый накопитель и с нетерпением ждете, когда покупка будет вам доставлена. Рассмотрим события, которые привели к моменту, когда вы получите свой новый накопитель, используя иллюстрацию на рис. 1.1 в качестве примера.

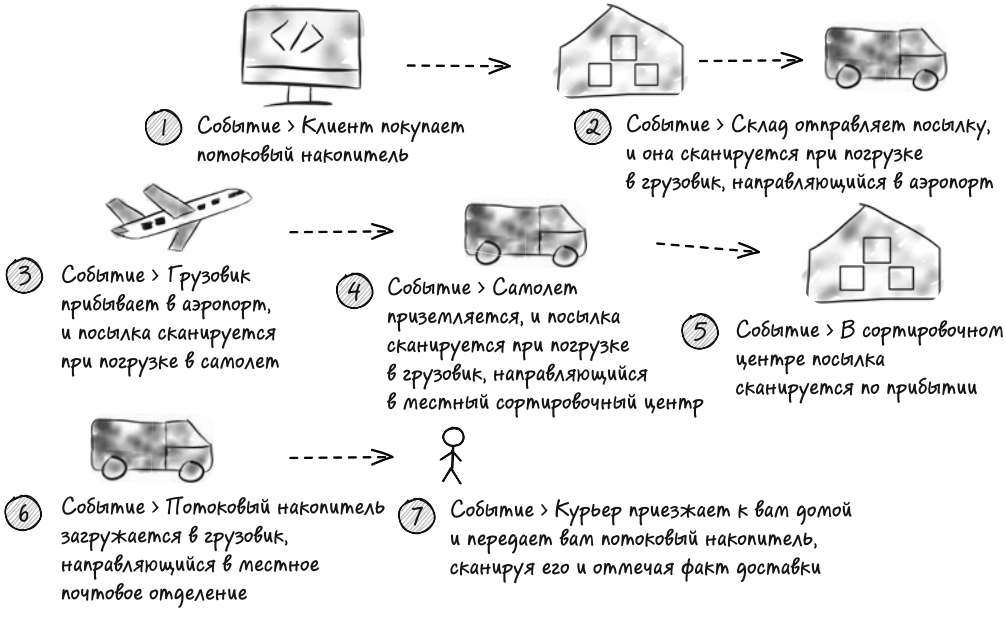

Рис. 1.1. Последовательность, составляющая поток событий, начинающийся с онлайн-покупки потокового накопителя

Рассмотрим шаги, ведущие к получению потокового накопителя.

1. Вы совершаете покупку на сайте, и сайт сообщает вам трек-номер.
2. Склад продавца получает информацию о событии покупки, упаковывает потоковый накопитель и грузит его в грузовик, фиксируя дату и время, когда ваша покупка покинула склад.
3. Грузовик прибывает в аэропорт, водитель выгружает потоковый накопитель в самолет и сканирует штрихкод, фиксируя дату и время.
4. Самолет приземляется, и посылка снова загружается в грузовик, направляющийся в местный сортировочный центр. Служба доставки регистрирует дату и время погрузки посылки в грузовик.
5. Грузовик приезжает в местный сортировочный центр. Сотрудник службы доставки выгружает посылку и, сканируя ее, фиксирует дату и время прибытия в сортировочный центр.
6. Другой сотрудник забирает посылку, сканирует, фиксирует дату и время и передает ее курьеру для доставки вам.
7. Курьер приезжает к вам домой, сканирует посылку в последний раз и передает ее вам. Теперь вы можете начать строить свою машину для путешествий во времени!

Наш пример показывает, как повседневные действия создают события, образуя их поток. Отдельные события — покупка, погрузка/выгрузка, прием на ответственное хранение и окончательная доставка. Этот сценарий представляет события, порожденные всего одной покупкой. А теперь вообразите потоки событий, созданных покупками на Amazon и различными отправителями товаров, — количество событий в них может исчисляться миллиардами или триллионами.

1.4. ЗНАКОМСТВО С APACHE KAFKA, ПЛАТФОРМОЙ ПОТОКОВОЙ ОБРАБОТКИ СОБЫТИЙ

Платформа Kafka предоставляет основные возможности для реализации приложений потоковой обработки событий от начала до конца. Мы можем разбить эти возможности на три основные области: публикация/потребление, долговременное хранение и обработка. Эти три этапа — перемещение, хранение и обработка — позволяют Kafka работать подобно центральной нервной системе для данных.

Прежде чем продолжить, нелишним будет проиллюстрировать, что значит быть центральной нервной системой для данных. Для этого будут показаны схемы до и после. Но сначала рассмотрим решение для потоковой обработки событий, в котором каждый источник входных данных требует отдельной инфраструктуры (рис. 1.2).

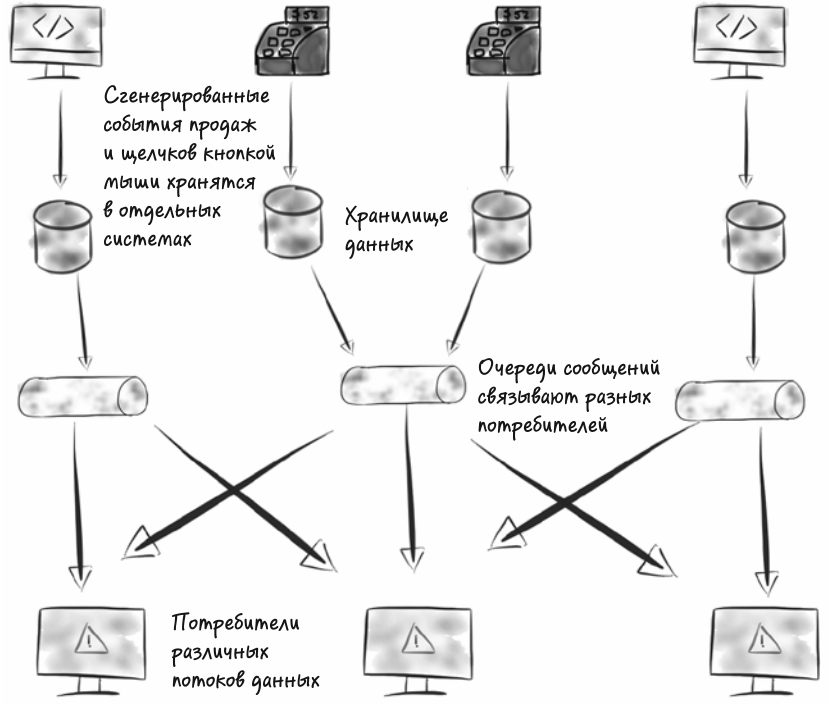

Рис. 1.2. Первоначальная архитектура потоковой обработки событий порождает массу сложностей, потому что различные отделы и источники потоков данных должны знать о других источниках событий

На иллюстрации видно, что каждый отдел создает свою инфраструктуру для удовлетворения своих потребностей. Однако другие отделы тоже могут быть заинтересованы в потреблении тех же данных, и добавление дополнительных связей для передачи потоков усложняет архитектуру. Взгляните на рис. 1.3, где показано, как платформа потоковой обработки событий Kafka может изменить ситуацию.

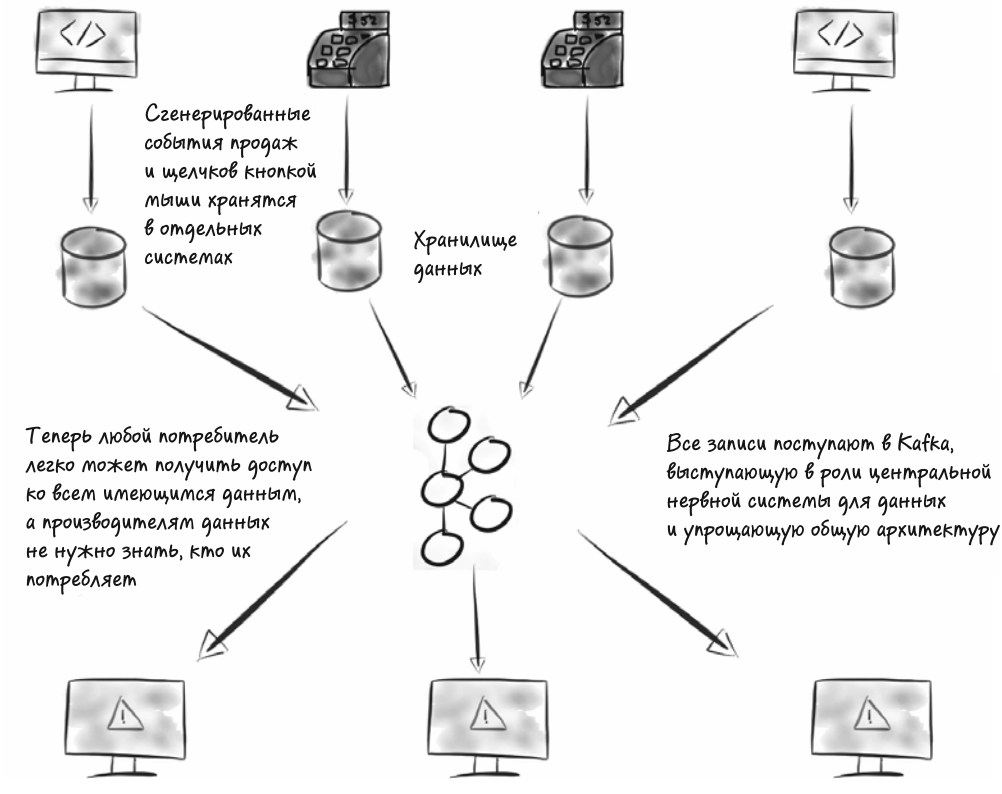

Рис. 1.3. Использование платформы Kafka помогает упростить архитектуру

Как можно видеть на этой обновленной иллюстрации, добавление платформы Kafka значительно упрощает общую архитектуру. Теперь все компоненты отправляют свои записи в Kafka, а потребители читают данные из Kafka, ничего не зная об их производителях.

Если говорить в общих чертах, то Kafka представляет собой распределенную систему серверов и клиентов. Серверы называются брокерами, а клиенты могут быть производителями событий, отправляющими записи брокерам, и потребителями, читающими записи для обработки событий.

1.4.1. Брокеры Kafka

Брокеры Kafka надежно хранят ваши записи в отличие от традиционных систем обмена сообщениями (RabbitMQ или ActiveMQ), где сообщения являются эфемерными. Брокеры хранят данные в виде пар «ключ — значение» (дополненных некоторыми другими полями метаданных) в байтовом представлении и являются своего рода черными ящиками.

Сохранение событий имеет глубокие последствия и определяет разницу между сообщениями и событиями. Сообщения можно рассматривать как «оперативную» информацию, которой обмениваются две машины, тогда как события представляют критически важные для бизнеса данные, потерять которые было бы очень нежелательно (рис. 1.4).

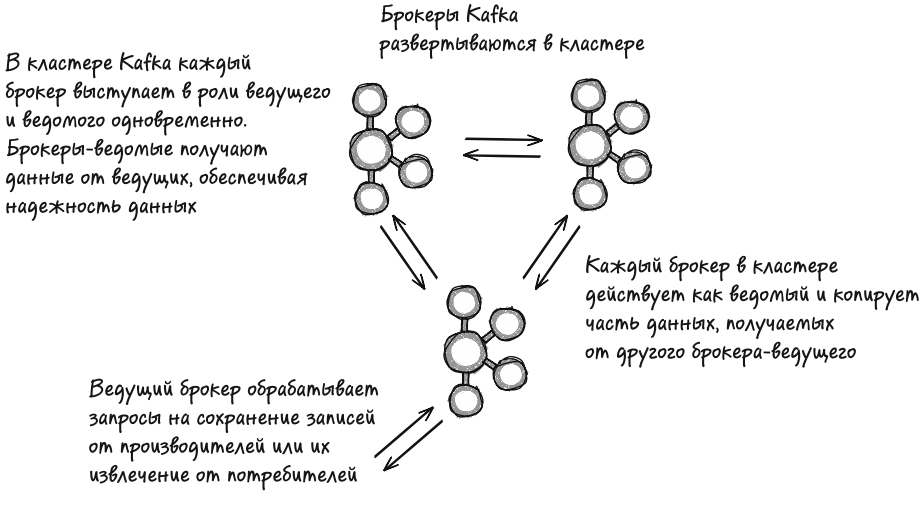

Рис. 1.4. Брокеры развертываются в кластере и реплицируют данные для долговременного хранения

Эта иллюстрация показывает, что брокеры Kafka являются уровнем хранения в трилогии потоковой обработки событий. Но, помимо хранения, брокеры предоставляют другие важные функции, такие как обслуживание клиентских запросов и координация с потребителями. Мы подробно рассмотрим работу брокеров в главе 2.

1.4.2. Schema Registry

Управление данными жизненно важно, и его важность только возрастает с ростом организации. Schema Registry хранит схемы записей с событиями (рис. 1.5). Схемы обеспечивают соблюдение контракта о формате данных между производителями

и потребителями. Schema Registry также предоставляет средства для сериализации и десериализации. Предоставление средств (де)сериализации означает, что вам не нужно писать свой код, осуществляющий сериализацию. Подробнее Schema Registry будет рассматриваться в главе 3. В приложении A также есть упражнение по миграции схем Schema Registry.

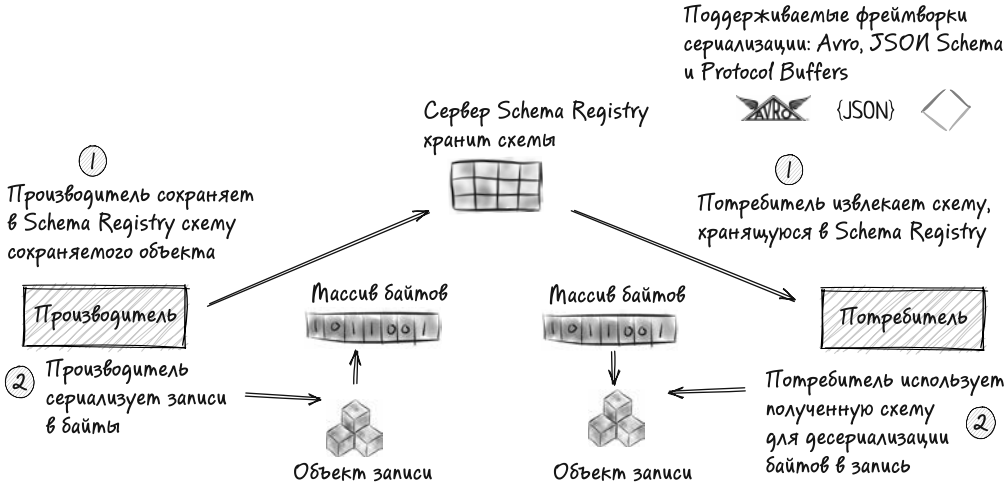

Рис. 1.5. Schema Registry обеспечивает единство моделей данных на всей платформе

1.4.3. Производители и потребители

Клиент-производитель отвечает за отправку записей в Kafka, а клиент-потребитель — за чтение записей из Kafka (рис. 1.6). Эти два клиента образуют основные строительные блоки для создания приложения, управляемого событиями, и не зависят друг от друга, что обеспечивает бóльшую масштабируемость. Клиент-производитель и клиент-потребитель также образуют основу для любой абстракции более высокого уровня, работающей с Apache Kafka. Более подробно мы рассмотрим клиенты в главе 4.

1.4.4. Kafka Connect

Kafka Connect — это абстракция поверх производителей и потребителей, предназначенная для импорта данных в Apache Kafka и экспорта данных из Apache Kafka (рис. 1.7). Компонент Kafka Connect необходим для подключения внешних хранилищ к Apache Kafka. Он также дает возможность выполнять простые преобразования данных с помощью Simple Message Transform (SMT) при экспорте или импорте данных. Мы поближе рассмотрим Kafka Connect в главе 5.

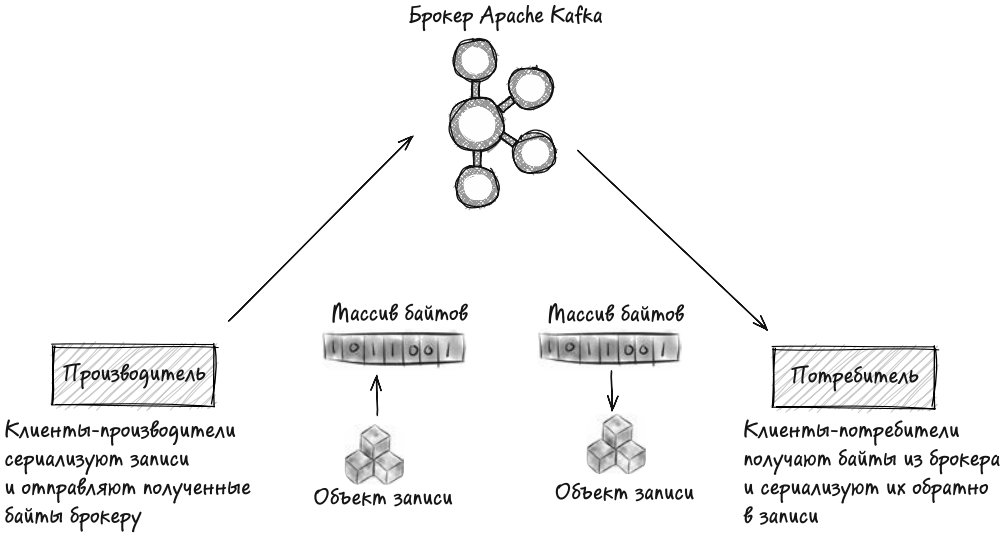

Рис. 1.6. Производители записывают записи в Kafka, а потребители читают их

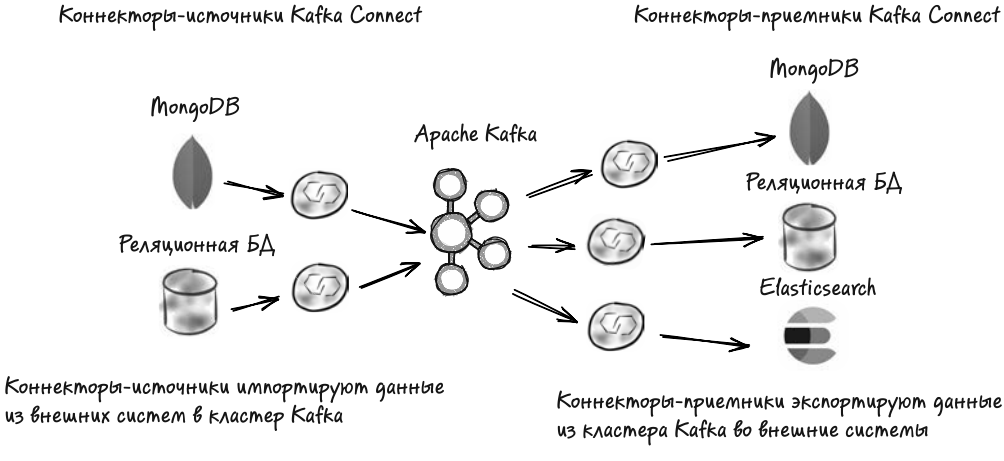

Рис. 1.7. Kafka Connect связывает внешние системы с Apache Kafka

1.4.5. Kafka Streams

Kafka Streams — это библиотека потоковой обработки для Kafka (рис. 1.8). Kafka Streams написана на языке Java и используется внешними клиентскими приложениями, взаимодействующими с кластером Kafka. Она поддерживает выполнение операций с данными событий, включая преобразования и операции с сохранением состояния,

такие как соединения и агрегирование. Kafka Streams — это то место, где вы будете работать с событиями. Подробнее о Kafka Streams рассказывается в главах 6–10.

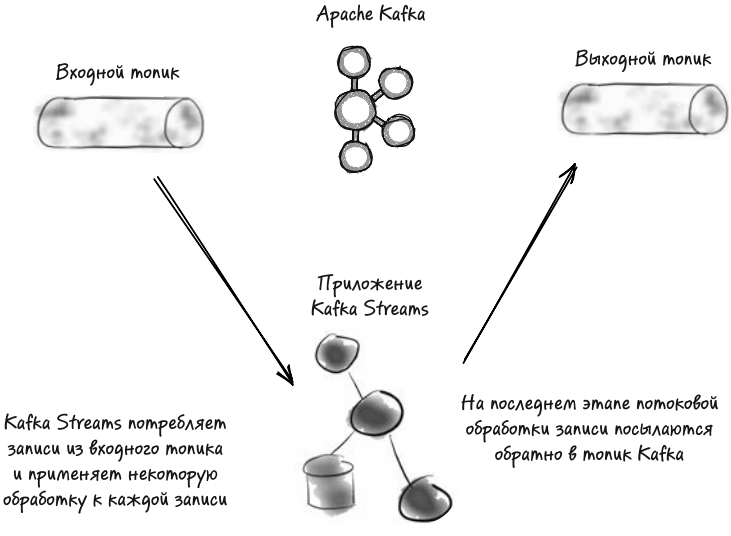

Рис. 1.8. Kafka Streams — это API потоковой обработки для Kafka

1.4.6. ksqlDB

ksqlDB — это база данных для хранения потоков событий (рис. 1.9). Она поддерживает интерфейс SQL для обработки потоков. Внутри для решения своих задач потоковой обработки событий ksqlDB использует Kafka Streams. Ключевое преимущество ksqlDB состоит в том, что она позволяет указывать операции потоковой обработки событий в виде SQL-кода; код на каком-либо языке программирования не требуется. Подробнее о ksqlDB рассказывается в главе 11.

```
CREATE TABLE activePromotions AS
    SELECT rideId
           qualifyPromotion(kmToDst) AS promotion
    FROM locations
    GROUP BY rideId
    EMIT CHANGES;
```

```
SELECT rideId, promotion
FROM activePromotions
WHERE ROWKEY = '6fd0fcdb'
```

Рис. 1.9. ksqlDB предоставляет возможности потоковой базы данных

Теперь, после знакомства с основными принципами работы платформы потоковой обработки событий Kafka и ее компонентами, предлагаю рассмотреть работу Kafka на конкретном примере розничной торговли.