ГЛАВА 1

Основы потоковой обработки

> Путешествие героя всегда начинается с зова. Так или иначе должен появиться проводник, который скажет: «Оглянись, ты в Стране снов. Проснись. Пора отправляться в путешествие. Существует целый аспект твоего сознания, твоего существа, который еще не затронут. Ты здесь чувствуешь себя как дома? Что ж, для тебя этого маловато». И так все начинается.
>
> *Джозеф Кэмпбелл. Reflections on the Art of Living: A Joseph Campbell Companion*

Концепция потоковой БД родилась из более чем десятилетнего опыта деятельности в сфере обработки и предоставления данных. Корни эволюции, приведшей к появлению потоковых баз данных, уходят глубже в историю развития систем управления базами данных, обработки данных и меняющихся требований цифровой эпохи. Для изучения этой эволюции мы совершим историческое путешествие по ключевым вехам развития потоковых БД.

Появление Интернета и взрывной рост объема цифровых данных в конце XX века привели к необходимости создания более масштабируемых и гибких решений для управления данными. Хранилища данных и ориентированные на пакетную обработку фреймворки, такие как Hadoop, создавались в ту эпоху для решения задач в ходе работы с большими объемами данных.

Термин «большие данные» использовался и продолжает использоваться для обозначения не только объема данных, но и всех решений хранения и обработки данных огромного объема. Большие данные не могут поместиться на одном компьютере или сервере. Приходится разделять их объем на более мелкие части одинакового размера и хранить их на нескольких компьютерах. Такие системы, как Hadoop и MapReduce, стали популярными благодаря возможности распределенного хранения и обработки данных.

Это привело к возникновению идеи применения распределенных потоков для перемещения больших объемов данных в Hadoop. Платформа Apache Kafka стала одним из сервисов обмена сообщениями для работы с большими данными. Платформа не только обеспечивала возможность переноса данных из системы в систему, но и предоставляла доступ к данным в движении — в режиме реального времени. Эта разработка породила новую волну спроса на решения для потоковых сценариев использования, работающих в режиме реального времени.

Были созданы новые технологии, такие как Apache Flink и Apache Spark, и они смогли оправдать повышенные ожидания. Это распределенные фреймворки для пакетной и потоковой обработки, способные работать с данными на многих серверах и выдавать аналитические результаты. В сочетании с Kafka это трио обеспечило решение для поддержки потоковой аналитики в режиме реального времени. Более подробно о потоковых процессорах мы поговорим в главе 2.

В середине 2010-х годов появились более простые и совершенные парадигмы потоковой передачи данных, позволяющие увеличить масштаб обработки данных в режиме реального времени. В их число вошли два новых фреймворка для обработки потоков — Apache Kafka Streams (KStreams) и Apache Samza. В KStreams и Samza впервые были воплощены материализованные представления, которые сделали поток более похожим на базу данных.

Мартин Клеппман пошел еще дальше в объединении баз данных и потоковой обработки. В своей презентации *Turning the Database Inside-Out* (https://oreil.ly/EPsaz) в 2015 году он описал способ реализации потоковой обработки с использованием внутренних возможностей базы данных, выведенных за ее пределы в виде потоков в режиме реального времени. Этот подход позволил создать более масштабируемые, отказоустойчивые системы обработки потоков в режиме реального времени.

Одной из проблем потоковой обработки была и остается бóльшая сложность ее применения по сравнению с пакетной. Здесь меньше абстракций и гораздо больше сложных технологий. Чтобы реализовать потоковую обработку данных для своих целей, инженеры по обработке данных должны были учесть порядок данных, согласованность для точной обработки, отказоустойчивость, надежность, масштабируемость и многое другое. Эти условия стали препятствием для команд по работе с данными, пытающихся использовать потоковую обработку. В результате большинство из них предпочли и дальше использовать базы данных для преобразования и пакетной обработки, что привело к снижению требований к производительности.

С этой книгой мы попытаемся сделать потоковую обработку более доступной для тех, кто привык работать с базами данных. Мы начнем, как и Клеппман, с рассказа о том, как вывернуть базу данных наизнанку.

Выворачиваем базу данных наизнанку

Мартин Клеппман — выдающийся разработчик программного обеспечения, который выступил с наводящим на размышления докладом *Turning the Database Inside-Out* («Выворачивая базу данных наизнанку»). Он представил вычислительную среду Apache Samza как новый способ осуществления потоковой обработки с использованием внутренних возможностей базы данных и их реализацией в потоках реального времени. Его идейное лидерство привело к изменению парадигмы — внедрению материализованных представлений в потоковую обработку.

> На самом деле это скрытая попытка взять известную архитектуру баз данных и вывернуть ее наизнанку.
>
> *Мартин Клеппман. Turning the Database Inside-Out (https://oreil.ly/EPsaz)*

Однако потоковая обработка по-прежнему сложна, и поэтому многие инженеры по обработке данных все еще используют БД для преобразования данных и их пакетной обработки. Даже если это означает несоблюдение требований *соглашения об уровне услуг* (Service Level Agreement, SLA).

По мере развития повествования в книге мы попытаемся сделать потоковую передачу и обработку более доступными для инженеров по обработке данных, перенеся эти процессы в БД. Но прежде чем мы сможем это сделать, необходимо понять, почему Клеппман решил разобрать базу данных на части и в своей новой парадигме выбрал конкретные функции базы данных для получения возможности обработки данных в режиме реального времени.

Вынос функциональности базы данных за ее пределы

Клеппман выделил две важные функциональные возможности базы данных: *механизм упреждающей записи* (Write-Ahead Log, WAL) и материализованное представление. Как оказалось, эти функции сами по себе обладают потоковыми характеристиками, которые обеспечивают лучший способ обработки данных в режиме реального времени.

Механизм упреждающей записи

WAL — это механизм, позволяющий базам данных обеспечивать долговечность и согласованность данных. Жесткие диски для записи данных из БД не предлагают транзакций. Поэтому перед СУБД стоит задача — обеспечить выполнение транзакций на устройстве без такой возможности. Механизмы WAL предоставляют способ их реализации без использования транзакционной памяти.

Транзакция в базе данных — это последовательность одной или нескольких операций над БД, выполняемых как единое целое. Это могут быть операции вставки (`INSERT`), изменения (`UPDATE`) или удаления (`DELETE`) данных (рис. 1.1).

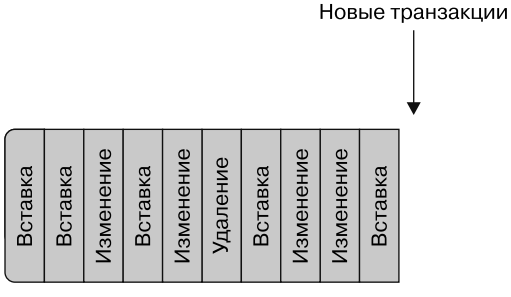

Рис. 1.1. Механизм упреждающей записи с фиксацией событий изменений в БД

Механизм WAL работает как буфер — его можно перезаписывать по мере внесения изменений. Он сохраняет изменения на диске (рис. 1.2).

Рис. 1.2. Запись БД на диск с помощью механизма упреждающей записи

При сохранении транзакций на диске выполняются следующие действия.

1. Клиент начинает транзакцию, используя оператор `BEGIN`.
2. База данных заносит в WAL запись с указанием начала транзакции.
3. Клиент вносит изменения в данные базы данных.
4. Клиент фиксирует транзакцию, применяя оператор `COMMIT`.
5. База данных записывает в WAL запись с указанием фиксации транзакции.
6. Изменения, внесенные транзакцией, записываются на диск.

Когда транзакция начинается, база данных заносит в WAL запись, указывающую на начало транзакции. После этого СУБД приступает к внесению изменений

в данные базы. Однако изменения не записываются на диск до тех пор, пока транзакция не будет зафиксирована. Кроме того, в случае сбоя в работе БД или обрыва питания изменения можно воспроизвести из журнала и БД будет восстановлена до согласованного состояния[1].

WAL предоставляет механизм для захвата транзакций базы данных в режиме реального времени, позволяя внешним системам подписываться на него. Один из таких случаев — аварийное восстановление баз данных. Считывая WAL, можно реплицировать данные во вторичную базу данных. Если основная база данных выходит из строя, ее клиенты могут переключиться на вторичную БД — копию основной (рис. 1.3).

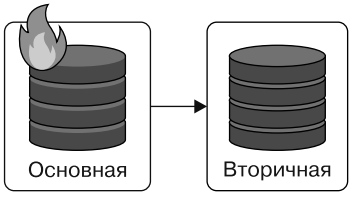

Рис. 1.3. Механизм WAL используется для репликации данных из основной базы данных во вторичную для защиты в случае выхода из строя основной БД

Поскольку журналы WAL получают транзакции в режиме реального времени, они, естественно, обладают идеальной семантикой для потоковой передачи. Клиенты могут подписаться на WAL и передавать свои транзакции на потоковую платформу, чтобы другие системы могли их использовать. И другие системы также могут создавать копии, представляющие исходную основную базу данных. Семантика конструкции WAL имитируется в потоковых платформах, таких как Kafka, в их реализации хранения данных. Потоковые платформы расширяют базы данных WAL извне, чтобы его могли использовать другие приложения и системы.

Есть и другие концепции WAL, связанные с потоковой обработкой. После фиксации транзакций журнал WAL очищается не сразу. В нем используется механизм под названием *«контрольная точка»* — его действие заключается в периодическом перемещении транзакций WAL в основные файлы данных. Установка контрольных точек служит нескольким целям, одна из которых — обеспечить постоянную запись некоторых зафиксированных изменений в файлы данных. Это позволяет сократить *воспроизводимый* при восстановлении после сбоя объем данных и, следовательно, ускоряет процесс восстановления. Кроме того, со временем по мере фиксации транзакций журнал WAL увеличивается. Контрольные точки помогают контролировать размер журнала, перемещая часть его содержимого в файлы данных. Таким образом, журнал WAL не становится слишком большим и не занимает слишком много места на диске. Контрольная

[1] Существуют и другие типы алгоритмов восстановления, помимо описанных нами (накат изменений/воспроизведение). Изменения в актуальных данных можно применить и до выполнения операции COMMIT (учитывая, что WAL содержит старое значение), а незафиксированные транзакции — откатить.

точка и воспроизведение транзакций — это функциональные возможности, применяющиеся в потоковой обработке по очень похожим причинам.

Конструкция WAL, обычно существующая внутри базы данных, может быть представлена во внешних потоковых платформах, таких как Kafka. Они обеспечивают подобную WAL семантику при репликации данных от системы к системе.

Платформы потоковой обработки

Платформы потоковой обработки данных, такие как Apache Kafka, — это распределенные, масштабируемые и отказоустойчивые системы для обработки потоков данных в режиме реального времени. Они представляют собой мощную инфраструктуру для поглощения больших объемов непрерывных данных из различных источников, их хранения и обработки.

В большинстве потоковых платформ есть конструкция, называемая *разделом* (partition). Эти конструкции имитируют журналы WAL из баз данных. Транзакции добавляются в разделы, как и транзакции в журналы WAL. Потоковые платформы могут содержать множество разделов для распределения потоковой нагрузки, что позволяет горизонтально масштабировать эти платформы. Разделы группируются в абстракции — *топики* (topic). Приложения либо публикуют транзакции в них, либо потребляют из них.

Публикуя транзакцию на потоковой платформе, мы делаем это для всех подписчиков. Это называется *моделью публикации и подписки*, которая позволяет нескольким отдельным потребителям использовать эти транзакции.

Для других потоковых платформ названия этих конструкций могут быть иными.

В табл. 1.1 перечислены некоторые альтернативные потоковые платформы. Apache Kafka — самая популярная из современных платформ такого типа. В ней абстракция этих конструкций называется топиком, а разделы более низкого уровня — разделами.

В книге мы будем использовать понятия «топик» (topic) и «раздел» (partition) как названия конструкций потоковой платформы для хранения потоковых данных в режиме реального времени.

Поскольку Kafka — самая популярная потоковая платформа, в последней графе табл. 1.1 указано, поддерживает ли платформа Kafka-клиентов. Это позволит заменить Kafka в приложениях на другую потоковую платформу, совместимую с Kafka.

Таблица 1.1. Альтернативные потоковые платформы

Название	Описание	Реализация	Топик	Раздел	Совместимость с Kafka
Memphis	Альтернатива традиционным брокерам сообщений с открытым исходным кодом нового поколения	GoLang	Station (Станция)	Stream (Поток)	Нет
Apache Pulsar	Платформа распределенного обмена сообщениями и потоковой передачи данных с открытым исходным кодом. Разработана в компании Yahoo!	Java	Topic (Топик)	Ledger (Леджер)	Да. Обертка Pulsar Kafka поддерживает большинство операций, предлагаемых API Kafka
Redpanda	Потоковая платформа с открытым исходным кодом. Разработана для обеспечения высокопроизводительной, масштабируемой и надежной обработки потоков данных в режиме реального времени	C++	Topic (Топик)	Partition (Раздел)	Да
WarpStream	Совместимая с Kafka потоковая платформа, созданная непосредственно на основе S3	GoLang	Topic (Топик)	Partition (Раздел)	Да
Gazette	Легкая потоковая платформа с открытым исходным кодом	GoLang	Selector (Селектор)	Journal (Журнал)	Нет
Pravega	Потоковый процессор, обеспечивающий абстракцию потокового хранения для непрерывно генерируемых и неограниченных данных	Java	Stream (Поток)	Stream Segment (Сегмент потока)	Доступен адаптер для Kafka

Деление на разделы — это механизм платформ для масштабирования. Чем больше разделов содержится в топике, тем больше можно разделить нагрузку от данных. Это позволяет большему количеству экземпляров потребителей обрабатывать транзакции параллельно. Для распределения транзакций по разделам используется присвоенный транзакции ключ. На рис. 1.4 журнал WAL из базы данных считывается и сохраняется в топике потоковой платформы — на более высоком уровне абстракции, чем просто диск.

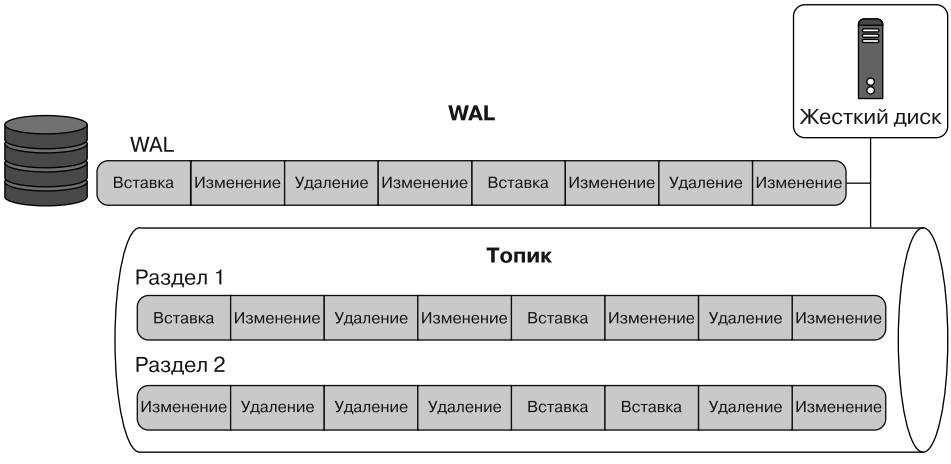

Рис. 1.4. Топики в потоковой платформе могут имитировать журнал WAL и передавать его другим системам для создания копий исходной базы данных

Вместо хранения данных для последующего запроса потоковая платформа реконструирует WAL и распределяет транзакции по отдельным разделам[1]. Реконструкция WAL открывает доступ к транзакциям для других систем данных, которые смогут создать копию основной базы данных.

Разделы — это неизменяемые журналы с возможностью только добавления данных, которые используются потоковыми платформами для сбора и обслуживания транзакций. С помощью смещений на них может подписаться множество потребителей. Смещение соответствует индексу или позиции транзакции в разделе[2]. Каждый потребитель топика содержит указатель смещения для отслеживания своей позиции в разделе. Это дает возможность потребителям читать и обрабатывать транзакции в разделе в своем темпе. Побочным эффектом такого подхода стала необходимость хранения транзакций в разделах дольше, чем базы данных хранят свои транзакции в журналах WAL. По умолчанию в Kafka срок хранения составляет семь дней, что дает медленным потребителям достаточно времени на обработку транзакций в топике. Время хранения можно настроить, чтобы еще увеличить его.

Что касается рис. 1.4, то подход к публикации транзакций в топике должен сильно отличаться от записи на диск. Важно отметить, что при публикации транзакций в топики потоковых платформ они все еще рассматриваются как

---

[1] Транзакции могут охватывать несколько записей. В этом случае ключ для распределения данных по разделам не совпадает напрямую с первичным ключом в исходной базе данных.

[2] Для простоты описания наша транзакция содержит только одну запись.

потоковые. Для лучшего понимания воспользуемся метафорой с водой. Когда вы пьете воду из крана, она кажется вам свежей. То же самое происходит и с потоковыми платформами. Когда вы потребляете транзакции из топика, они тоже считаются свежими. И наоборот, если принести в дом литр воды и не пить ее некоторое время, она будет считаться несвежей. В несвежей или застоявшейся воде размножаются бактерии, и она считается нечистой. Литр такой воды больше похож на пакетные данные.

В то же время, если не открывать краны больше месяца, в воде, поступающей из них, может появиться ржавчина или мусор — это говорит о том, что вода стала несвежей. В таком случае вода в кране не всегда будет свежей. Как правило, в потоковых платформах предусмотрен механизм, защищающий их от хранения несвежих транзакций. Чтобы избежать публикации несвежих транзакций, к топикам применяется политика хранения. Транзакция может быть удалена после окончания периода хранения, настроенного пользователем потоковой платформы.

Суммируя, напомним, что основные базы данных OLTP при хранении данных на жестких дисках ведут запись в WAL. Эти журналы можно использовать для репликации данных во вторичную базу данных OLTP в сценариях аварийного восстановления системы. Потоковые платформы, такие как Kafka, подходят для предоставления журналов базы данных WAL в другие системы с помощью абстрагированных по топикам разделов. Такие системы подписываются на топик, чтобы получить возможность построить свою копию таблиц из оригинальной первичной базы данных OLTP точно так же, как это выполнялось для вторичной базы данных OLTP (рис. 1.5). Таким образом, потоковые платформы можно использовать для того, чтобы сделать журналы WAL, ранее скрытые в системах баз данных OLTP, общедоступными и превратить их в инструмент для синхронизации систем БД всей организации.

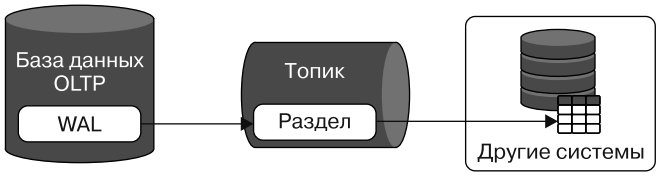

Рис. 1.5. Разделы в топике могут хранить транзакции из исходной базы данных OLTP и публиковать их для других систем с целью создания копий таблиц

Используя аналогичный подход, можно создавать материализованные представления в платформах обработки потоков.

Материализованные представления

В типичных базах данных OLTP *материализованные представления* — это специальные типы объектов БД, которые хранят результаты предварительно вычисленного запроса или агрегации. Обычные представления виртуальны и динамически генерируют свои результаты на основе базовых данных. А материализованные представления хранят фактические данные, а значит, физически хранят в базе данных.

Цель материализованных представлений — повысить производительность сложных запросов или агрегаций, предварительно вычислив и сохранив результаты. Когда запрос ссылается на материализованное представление, база данных может быстро получить предварительно вычисленные данные из этого представления вместо того, чтобы пересчитывать их на лету из таблиц. Это может значительно сократить время выполнения запроса и повысить общую производительность базы данных, особенно если запросы большие и ресурсоемкие.

Материализованные представления обычно приходится обновлять вручную, чтобы сохранить актуальность хранимых результатов. В примере 1.1 показано, как обновить материализованное представление в базе данных Postgres — популярной базе данных OLTP.

Пример 1.1. Обновление материализованного представления в Postgres

```
REFRESH MATERIALIZED VIEW CONCURRENTLY product_sales;
```

Из-за возможности обновления материализованных представлений хранимые данные всегда будут актуальными, то есть данными реального времени. Благодаря этой особенности материализованные представления органично вписываются в потоковые фреймворки.

Ранее мы рассмотрели, что потоковые платформы могут хранить транзакции из журналов WAL баз OLTP. Эти разделы имитируют конструкцию WAL, поэтому другие системы могут создавать копии таблиц исходной базы данных OLTP. Этот же подход можно применить в потоковых процессорах для построения табличных структур (рис. 1.6).

Подробнее о потоковой обработке поговорим в главе 2. А главу 3 мы посвятили материализованным представлениям, поскольку их важность для потоковых БД очень велика. Чтобы лучше объяснить принципы работы потоковых БД, нужно создать простой пример практического применения и проследить ход его выполнения до конца. Попутно мы определим каждую систему, необходимую для достижения цели сценария использования.