Содержание

От издательства

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв на нашем сайте www.dmkpress.com, зайдя на страницу книги и оставив комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com; при этом укажите название книги в теме письма.

Если вы являетесь экспертом в какой-либо области и заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу http://dmkpress.com/authors/publish_book/ или напишите в издательство по адресу dmkpress@gmail.com.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы обеспечить высокое качество наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг, мы будем очень благодарны, если вы сообщите о ней главному редактору по адресу dmkpress@gmail.com. Сделав это, вы избавите других читателей от недопонимания и поможете нам улучшить последующие издания этой книги.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательство «ДМК Пресс» очень серьезно относится к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконной публикацией какой-либо из наших книг, пожалуйста, пришлите нам ссылку на интернет-ресурс, чтобы мы могли применить санкции.

Ссылку на подозрительные материалы можно прислать по адресу электронной почты dmkpress@gmail.com.

Мы высоко ценим любую помощь по защите наших авторов, благодаря которой мы можем предоставлять вам качественные материалы.

Предисловие

История хранилищ данных началась с того, что разработчики просто переносили данных из операционных баз данных в системы, более приспособленные для аналитики. Эксплуатация этих систем обходилась весьма недешево, и людям приходилось очень тщательно выбирать данные для загрузки.

С годами требования к объему обрабатываемых данных росли, значительно опережая рост производительности оборудования, описанный законом Мура. Для традиционных аналитических систем анализ такого объема данных стал непосильной задачей. Рост требований коснулся всех, но некоторые компании столкнулись с проблемами роста раньше остальных.

Facebook[1] стал одной из первых компаний, попытавшихся в 2012 году решить проблему производительности аналитических платформ. В то время в Facebook для интерактивной аналитики использовался Apache Hive. По мере роста объема данных выяснилось, что Hive не такой уж интерактивный, а попросту говоря – медленный. Во многом это связано с тем, что Hive является частью экосистемы Hadoop, где главным инструментом исполнения запросов является фреймворк MapReduce. Главный его недостаток в том, что он сохраняет промежуточные наборы данных на диске, что требует огромного объема ввода-вывода. Facebook разработал новый движок распределенных SQL-запросов – Presto, выполняющий промежуточные операции в памяти и не требующий дискового ввода-вывода для хранения временных данных. Такой подход позволил выполнять обработку данных на порядки быстрее: теперь многие запросы завершались в течение секунды. Пользователи – инженеры, менеджеры и аналитики – обнаружили, что анализ снова стал интерактивным и они могут выполнять множество запросов для проверки гипотез и создания графиков и диаграмм.

Facebook был одной из первых, но далеко не единственной компанией, столкнувшейся с проблемой роста объемов данных, опережающего рост возможностей аппаратуры. Для решения этой проблемы была разработана архитектура «озера данных» (data lake), где обработка данных отделена от их хранения. Такая архитектура позволила хранить данные в распределенных файловых системах, построенных на базе дешевого оборудования, а поз-

---

[1] Социальная сеть Facebook принадлежит компании Meta, признанной в России экстремистской организацией. Здесь и далее торговая марка Facebook употребляется исключительно в контексте созданных компанией технологий. – *Прим. перев.*

же – и в облачных хранилищах. Одновременно с дешевыми системами хранения развивались и вычислительные кластеры. Правда, очень долго было непонятно, как должен быть устроен инструментарий для интерактивной обработки данных. Часто, как в Facebook до 2012 года, аналитики пытались использовать инструменты, предназначенные для пакетной обработки, производительность которых их не устраивала.

В 2013 году исходный код Presto был открыт. Платформа быстро завоевала популярность и получила дополнительное развитие со стороны таких компаний, как Airbnb, Uber и Netflix. Дело в том, что задачи, стоящие перед аналитиками Facebook, не были уникальными – просто Facebook столкнулась с ними раньше всех. Со временем потребность в интерактивных запросах к озеру данных только росла. По мере роста использования росли и ожидания: пользователи требовали изоляции сеансов (то есть гарантии непротиворечивости результатов запросов при одновременно работающих ETL-процессах), эволюцию схемы данных, возможность запросов к предыдущим версиям данных. Чтобы выполнить растущие требования, разработчикам пришлось существенно изменить форматы таблиц. Они стали более сложными и функциональными, чем исходный формат Hive. Добавилась поддержка транзакций (ACID) и индексов. Архитектура Presto также менялась под действием новых требований. В конце концов архитектура «хранилища-озера» (lakehouse) стала отраслевым стандартом: дешевая распределенная система хранения данных, как в традиционном «озере», с аналитическим движком, производительность которого не уступает производительности традиционных аналитических СУБД. Возможности хранилищ, построенных по такому принципу, практически сравнялись с возможностями традиционных хранилищ, но «хранилища-озера» при этом не требуют сложных процедур загрузки и преобразования данных (ETL).

Зачем мы написали эту книгу?

Развертывание Presto для построения традиционного хранилища данных или «хранилища-озера» – весьма серьезное мероприятие. Чтобы все получилось так, как задумано, вам необходимо понять правила, по которым работает Presto, и неукоснительно им следовать. Мы написали эту книгу, чтобы рассказать вам об этих правилах и об инструментах, которые предоставляет Presto, одна из самых мощных распределенных платформ анализа данных. Вы не только поймете ее принципы, но и сможете развернуть и сконфигурировать собственную систему, воспользовавшись всеми преимуществами платформы. Часть книги посвящена экосистеме, сформировавшейся вокруг Presto, и способам интеграции Presto с другими популярными платформами с открытым исходным кодом, например Apache Pinot или Apache Hudi. Эта интеграция существенно расширит возможности вашей системы. Прочитав нашу книгу, вы сможете глубоко разобраться в платформе Presto и использовать ее в текущем проекте и во всех последующих.

Для кого эта книга?

Эта книга – для тех, кто создает хранилище данных. Для программистов, администраторов, архитекторов, инженеров данных и облачных инженеров. Для тех, кто создает хранилище, поддерживающее множество взаимосвязанных продуктов. В их обязанности входит обеспечение совместной работы всех компонентов. Они решают вопросы очистки и дедупликации данных, выстраивают процессы управления данными, разрабатывают процедуры преобразования данных, а также постоянно добавляют к системе новые инструменты и технологии, делающие ее удобнее и полезнее для конечного пользователя.

Немного об оформлении

В этой книге используется ряд типографских приемов.

Курсивом
: выделены новые термины, адреса веб-страниц и электронной почты, имена файлов.

`Моноширинный шрифт`
: используется для текстов программ, а также внутри обычного текста, чтобы выделить элементы программного кода – функции и переменные, типы данных, переменные среды, команды и ключевые слова.


Такой элемент содержит подсказку или предложение.


Это общее замечание.


А так обозначается предупреждение или предостережение.

Использование примеров кода

Дополнительные материалы (исходный код, примеры, упражнения и т. д.) можно загрузить из репозитория на GitHub, созданного специально для этой книги. Репозиторий находится по адресу https://github.com/alod83/Learning-and-Operating-Presto.

Если у вас возникли технические вопросы или проблемы с использованием примеров кода, напишите письмо по адресу bookquestions@oreilly.com.

Мы написали эту книгу, чтобы помочь вам в работе. В большинстве случаев, если вы хотите использовать пример из этой книги в своей программе или в документации (если только вы не воспроизводите значительную часть

кода), не нужно обращаться за специальным разрешением. Отдельное разрешение требуется, если вы хотите распространять или продавать примеры из книг издательства O'Reilly. Цитирование книги, как и использование примеров кода, разрешения не требует.

Обычно мы не требуем указания авторства, но очень это ценим. Если вы хотите сослаться на книгу, следует указать ее название, автора, издателя и ISBN. Например: *Анжелика Ло Дука, Тим Михан, Вивек Бхаратан, Ин Су.* Изучаем и используем Presto. М.: ДМК Пресс, 2024. ISBN 978-5-93700-294-5.

Благодарности

Эта книга является результатом работы многих людей, вклад каждого из которых бесценен. Мы хотим сказать спасибо всем, кто поделился с нами своим уникальным опытом и знаниями, кто заразил нас своей преданностью делу, благодаря которой книга вышла такой удачной. Прежде всего мы хотели бы выразить благодарность техническим рецензентам Чуньсю Тану (Chunxu Tang), Андреасу Кальтенбруннеру (Andreas Kaltenbrunner) и Скотту Хейнсу (Scott Haines), которые нашли время, чтобы дать конструктивные отзывы. Ваши замечания и предложения помогли значительно улучшить книгу, и мы очень ценим вашу помощь.

Кроме того, мы искренне благодарим Вэнь Фана (Wen Phan), щедро поделившегося с нами своим опытом. Его знания и идеи значительно обогатили книгу, особенно главу 5.

Мы также хотели бы выразить искреннюю благодарность редактору O'Reilly Аарону Блэку (Aaron Black) за веру в концепцию этой книги и за предоставленную нам возможность ее издать. Ваша помощь на протяжении всего процесса работы над книгой была просто бесценна. Спасибо техническому редактору Гэри О'Брайену (Gary O'Brien) за неизменную поддержку и за стремление сделать эту книгу как можно лучше. Ваше внимание к деталям и содержательные предложения действительно очень помогли.

И особая благодарность всей команде O'Reilly за трудолюбие, профессионализм и энтузиазм. Пусть ваш неустанный труд часто был скрыт от посторонних глаз, но именно благодаря ему эта книга наконец увидела свет.

Анжелика Ло Дука

В первую очередь я благодарю своего мужа Андреа за его терпение и поддержку во время написания этой книги. Спасибо за наши долгие беседы, которые обеспечили мне душевный комфорт и подняли мотивацию. Сердечное спасибо моим детям, которые проводили время за играми, пока я набирала текст. Пока я писала свою книгу, они писали свою. И особая благодарность Господу Богу Иисусу Христу за то, что дал мне возможность поделиться результатами своих занятий и исследований – ведь отдавать всегда приятнее, чем получать.

Тим Михан

Спасибо моей жене и сыну за их бесконечное терпение. Спасибо Гиришу Баллиге (Girish Balliga) за его идеи и за руководство Фондом Presto. Спасибо моим коллегам из Meta и IBM, в частности Джеймсу Сану (James Sun), Маше Басмановой (Masha Basmanova), Стивену Миху (Steven Mih) и Али Леклерк (Ali LeClerc), за дружбу, поддержку и вдохновение.

Вивек Бхаратан

Сердечное спасибо за вклад в эту книгу сообществу Presto. Спасибо за обсуждения, правки, идеи и многое другое. Особая благодарность Ретике Агравал (Reetika Agrawal), Девешу Агравалу (Devesh Agrawal) и Рохану Педнекару (Rohan Pednekar). Также не могу не отметить Али Леклерк (Ali LeClerc), чья помощь была просто неоценима. Спасибо Стивену Миху (Stephen Mih)[1] и Дипти Боркару (Dipti Borkar), которые буквально вели нас за собой. Спасибо Дэвиду Симмену (David Simmen) за руководство и наставления. И наконец, спасибо моей жене Прие и дочери Ракше, без поддержки которых эта работа была бы просто невозможна.

Ин Су

Спасибо моим дорогим родителям, которые подарили мне бесконечную любовь, заботу и поддержку. Спасибо Стивену Миху (Steven Mih), Дипти Боркару (Dipti Borkar)[2] и Дэвиду Симмену (David Simmen) за их дальновидность, смелость, неустанную работу, огромное доверие и поддержку. Спасибо моим коллегам из Ahana, IBM и Meta за нашу дружбу и за вдохновение.

---

[1] Steven Mih и Stephen Mih – один и тот же человек, CTO компании Ahana. В сети встречаются оба варианта написания имени. – *Прим. перев.*

[2] В оригинале – Dipti Borker, но это ошибка. – *Прим. перев.*

Об авторах

Анжелика Ло Дука – исследователь со степенью доктора компьютерных наук. Она работает в исследовательском подразделении Института информатики и телематики при Национальном исследовательском совете Италии. В сферу ее научных интересов входят подготовка и исследование данных, разработка веб-приложений, а также популяризация работы с данными при помощи лекций и статей. Ряд ее работ связан с сетевой безопасностью, блокчейном и с развитием технологий публикации данных – семантической паутиной и связанными данными. Кроме того, она является профессором Пизанского университета, где преподаёт журналистику данных.

Тим Михан большую часть своей карьеры посвятил вопросам обработки данных. Он работает над проектом Presto с 2018 года. В настоящее время трудится в IBM и возглавляет технический комитет Presto. До IBM он работал в таких компаниях, как Meta, Bloomberg, Goldman Sachs и других.

Вивек Бхаратан – соучредитель и главный инженер в компании Ahana, являющейся частью IBM. Ранее работал программистом в Uber, где управлял кластерами Presto, состоявшими из более чем двух с половиной тысяч узлов и обрабатывающими до 35 петабайт данных ежедневно, а также дорабатывал Presto, чтобы закрыть потребности Uber в интерактивной аналитике. До прихода в Uber Вивек был одним из первых сотрудников команды по оптимизации запросов в Vertica Systems и внес значительный вклад в разработку ядра Vertica и ее экосистемы, а его вклад в Presto включает в себя анализ частичных агрегаций. Ему принадлежит ряд алгоритмов для систем поддержки принятия решений и логических рассуждений, которые он разрабатывал в начале своей карьеры в Лаборатории исследований искусственного интеллекта. Вивек получил степень магистра компьютерных наук и инженерии в Университете штата Огайо.

Ин Су – архитектор в компании Ahana, где она работает над повышением производительности и эффективности сервисов на базе Presto и Velox. В прошлом она работала над SQL Server в Microsoft и над Presto в Meta. Она является коммиттером Presto и членом правления TSC.

Колофон

Животное на обложке этой книги – мабуя обыкновенная (*Mabuya mabouya*). Об этих ящерицах из семейства сцинковых известно не так уж много. Они – эндемики Мартиники. Мабуи обитают на суше и покрыты кожей бронзового цвета со светлыми полосками на верхней части туловища.

Всего известно более полутора тысяч видов сцинков. Ноги у сцинков короче относительно тела, чем у большинства ящериц, поэтому манера их передвижения напоминает змеиную. Они способны отбрасывать хвост, если на них нападают хищники, а потом в течение нескольких месяцев могут отрастить его заново – по крайней мере частично. Сцинки питаются в основном насекомыми, такими как сверчки, жуки, гусеницы и мухи.

Мабуя обыкновенная находится под угрозой исчезновения. Возможно, этот вид уже исчез из-за того, что человек завез на Карибские острова хищников, которые там никогда не жили. Многие животные на обложках книг O'Reilly находятся под угрозой исчезновения, и очень важно их сохранить.

Иллюстрация на обложке принадлежит кисти Карен Монтгомери (Karen Montgomery) и написана по мотивам старинной гравюры из книги *Pictorial Museum of Animated Nature*.

Глава 1

Введение в Presto

В последние несколько лет растущий объем данных, как создаваемых пользователями, так и автоматически генерируемых, ставит новые задачи перед организациями, желающими разобраться в этих данных для принятия более эффективных решений. Чтобы находить новые идеи, видеть новые возможности и меняться в соответствии с требованиями времени, критически важно ориентироваться именно на объективные данные. Появился даже новый термин – «data-driven organization», буквально – «организация, управляемая данными». Объем этих данных огромен, и их обработка требует значительных усилий, но полученные преимущества стоят затраченных усилий.

Данные извлекаются из разных источников в разных форматах, а для обращения к ним используются разные языки. Пользователи, пытающиеся извлечь из этих данных информацию, хотят получать ответы на свои запросы очень быстро, и поэтому им нужны высокопроизводительные системы обработки. Эти требования заставили Facebook (теперь Meta), Airbnb, Uber, Netflix и другие компании переосмыслить методы управления данными. Постепенно они перешли от старой парадигмы хранилищ данных (warehouse) к парадигме хранилищ-озер (lakehouse). Если традиционное хранилище данных содержит структурированные данные и историю их изменения, то хранилище-озеро также может содержать и неструктурированные, и слабоструктурированные данные, обновляемые в режиме реального времени.

Presto – один из инструментов, решающих задачу обработки данных в хранилищах-озерах. Presto представляет собой распределенный движок SQL, созданный и используемый в Facebook. Вы можете легко интегрировать Presto в свое хранилище данных, чтобы эффективно выполнять SQL-запросы независимо от физического местоположения и формата данных.

Эта глава познакомит вас с концепцией озера данных и расскажет, чем оно отличается от классического хранилища данных. Вы узнаете, что такое Presto, зачем этот движок был создан и почему его использует так много компаний. Вы также ознакомитесь с наиболее популярными сценариями применения Presto, такими как специальные (ad hoc) запросы, отчеты и информационные панели. Наконец, вы познакомитесь с хранилищем, которое будет использоваться в качестве примера далее во всех главах.

Хранилища данных и озера данных

Все данные можно разделить на три вида: *структурированные*, *слабоструктурированные* и *неструктурированные*. В табл. 1.1 собрана информация о видах данных: краткое описание, популярные форматы, достоинства и недостатки, а также примеры использования.

Таблица 1.1. Виды данных

	Структурированные	**Слабоструктурированные**	**Неструктурированные**
Описание	Структура данных жестко зафиксирована	Данные структурированы, но единой схемы нет	Данные не структурированы
Популярные форматы	SQL, CSV	JSON, XML	Аудио, видео, текст
Достоинства	Легко извлекать информацию	Более гибкий и универсальный подход, чем структурированные данные	Хорошо поддаются масштабированию
Недостатки	Масштабирование ограничено структурой	Метаданные не структурированы	Сложно что-либо искать
Примеры	База данных	Текст с аннотациями, например твиты с хештегами	Текст, цифровые фотографии

Разные классы хранилищ данных отличаются видами данных, которые они способны хранить и обрабатывать. Так, традиционные *хранилища данных* (*data warehouse*) – это централизованные системы, которые содержат *только* структурированные данные и используются для аналитики и отчетности. На рис. 1.1 изображена общая архитектура традиционного хранилища. Оно состоит из четырех основных компонентов:

- *структурированные данные,* собранные из разных источников, например из реляционных баз данных;
- *ETL – Extract, Transform, Load* – процесс, приводящий данные к требуемому формату;
- *слой детальных данных*[1], содержит данные, готовые к использованию конечными потребителями;
- *отчетность, информационные панели и интеллектуальный анализ данных.* Верхний уровень – инструменты, использующие информацию из хранилища, с которыми непосредственно работают пользователи.

[1] В оригинальном тексте data warehouse является частью data warehouse. Термин «слой детальных данных», detail data store (DDS), взят из других источников. – *Прим. перев.*

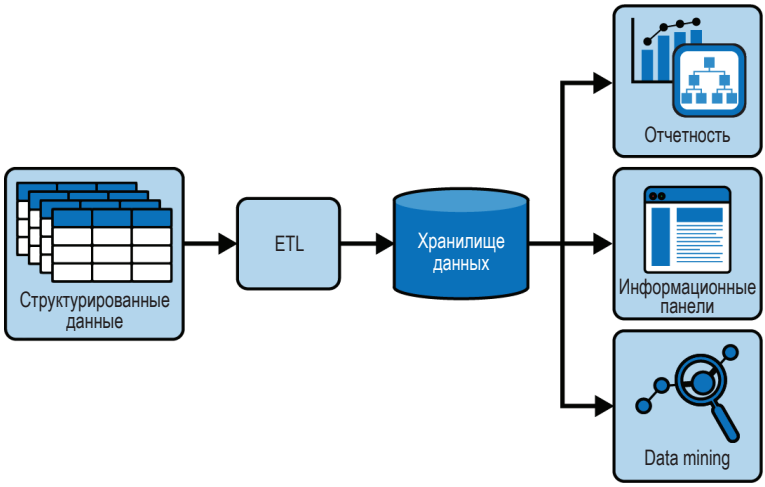

Рис. 1.1 ❖ Общая архитектура традиционного хранилища данных

С появлением больших данных выяснилось, что хранилища, построенные по традиционной архитектуре, не справляются с требуемыми объемами. Крупные компании, такие как Facebook, столкнулись с рядом проблем:

- *неструктурированные данные*. Поскольку хранилища предназначены для хранения и обработки структурированных данных, их нельзя использовать для хранения таких данных, как текст или аудио. Прежде чем записать эти данные в хранилище, их необходимо обработать;
- *масштабирование*. С ростом объема данных стоимость и техническая сложность традиционных хранилищ растет нелинейно;
- *предоставление данных в режиме реального времени*. Традиционное хранилище не способно предоставлять данные в режиме реального времени, поскольку перед загрузкой данные должны быть обработаны и преобразованы;
- *озеро данных* решает эти проблемы. На рис. 1.2 показана общая архитектура озера данных.

В отличие от традиционного хранилища, озеро данных хранит структурированные, слабоструктурированные и неструктурированные данные и предоставляет механизмы для их обработки и использования. Озеро может загружать сырые данные в режиме реального времени и затем хранить их наряду с историческими данными. Со временем концепция озера эволюционировала в *хранилище-озеро* (*data lakehouse*), включающее возможность выполнения транзакций поверх озера данных. На практике хранилище-озеро изменяет данные в озере, следуя семантике традиционного хранилища. Мы обсудим концепцию хранилища-озера и разработаем такое хранилище в главе 5.

Поначалу озера данных располагались целиком на оборудовании компаний, которым они принадлежали (*on-premise data lakes*). Их главным преимуществом был полный контроль системы компанией-владельцем. С появлением облачных вычислений озера данных постепенно переместились

в облака, где вопросы управления, обслуживания и безопасности решаются инженерами компании-провайдера. Разумеется, клиенты разделяют с облачными провайдерами ответственность за безопасность данных. Такой подход называется *облачным озером данных*, и его популярность непрерывно растет. Основные игроки на этом рынке – Amazon Web Service (AWS), Microsoft Azure и Google Cloud, а наиболее популярной инфраструктурой для хранения данных стало *объектное хранилище*[1].

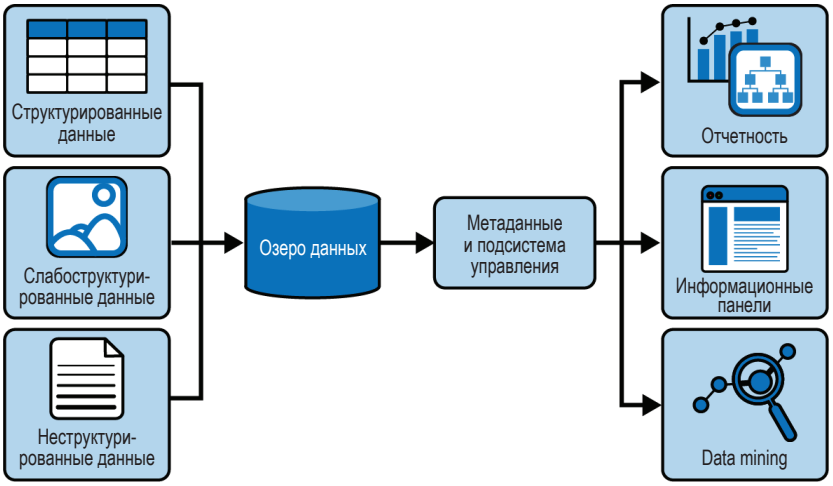

Рис. 1.2 ❖ Общая архитектура озера данных

Чтобы обеспечить доступ инструментам верхнего уровня (отчетность, информационные панели, инструменты интеллектуального анализа) к данным, в озере есть промежуточный уровень, называемый уровнем *управления и метаданных* (*metadata and governance*), который обеспечивает согласованность данных и управление доступом.

Роль Presto в озере данных

Presto – это распределенный движок SQL с открытым исходным кодом, который поддерживает в качестве источников как структурированные, так и слабоструктурированные данные. Вы можете использовать Presto для обращения к данным вне зависимости от того, где они находятся, что удобно для подключения к хранилищу-озеру. Для выполнения запроса нет необходимости перемещать данные. Запросы в Presto выполняются параллельно, что позволяет масштабировать эту платформу, а выполнение запросов в памяти обеспечивает высокую скорость.

[1] Имеется в виду сервис S3, simple storage service, впервые предложенный в 2006 году компанией Amazon. – *Прим. перев.*

Узел-координатор анализирует SQL-запрос (поддерживается стандарт ANSI SQL[1]), составляет план запроса, раздает задания рабочим узлам, непосредственно работающим с источниками данных, а затем возвращает пользователю результат запроса. В зависимости от сложности запроса план может содержать разное, иногда большое количество шагов. Например, если запрос выполняет соединение нескольких больших таблиц, ему может понадобиться несколько проходов, агрегирующих данные. Вы можете воспринимать эти промежуточные результаты как заметки, появляющиеся в блокноте в процессе решения сложной математической задачи.

Происхождение и истоки архитектуры Presto

Разработка Presto началась в компании Facebook в 2012 году. Целью было избавление от недостатков Apache Hive – распределенного SQL-движка, разработанного на фреймворке MapReduce в Hadoop. Одно из хранилищ данных Facebook было построено как раз на базе Apache Hive, и главной претензией пользователей к хранилищу была низкая скорость выполнения запросов над большими объемами данных.

Apache Hive

Apache Hive также был разработан в Facebook в 2010 году, и его код тоже был открыт. В основе Apache Hive, как и в основе других платформ в экосистеме Hadoop, лежит фреймворк MapReduce, предусматривающий запись промежуточных результатов выполнения запроса на диск. Такая архитектура требует частого доступа к диску и большого объема ввода-вывода.

Presto, новый распределенный SQL-движок, был призван решить эту проблему за счет обработки данных в памяти. У него не было потребности в массивном вводе-выводе, и в результате запросы стали выполняться на несколько порядков быстрее. Время выполнения многих запросов теперь не превышало одной секунды. Конечные пользователи – разработчики, аналитики, владельцы продуктов – получили возможность интерактивного выполнения запросов для проверки гипотез или создания диаграмм.

Рисунок 1.3 показывает, как выполняют запросы Presto и Hive. Hive использует фреймворк MapReduce, который записывает на диск промежуточные результаты как после фазы Map, так и после фазы Reduce. В отличие от него, Presto экономит время за счет сокращения ввода-вывода: все операции выполняются в памяти рабочих узлов.

---

[1] В книге не указана версия стандарта, которую поддерживает Presto. Обычно под ANSI SQL понимают стандарт SQL92, но на самом деле стандарт живет и развивается, и последняя на момент перевода версия – SQL:2023 (ISO/IEC 9075:2023). – *Прим. перев.*

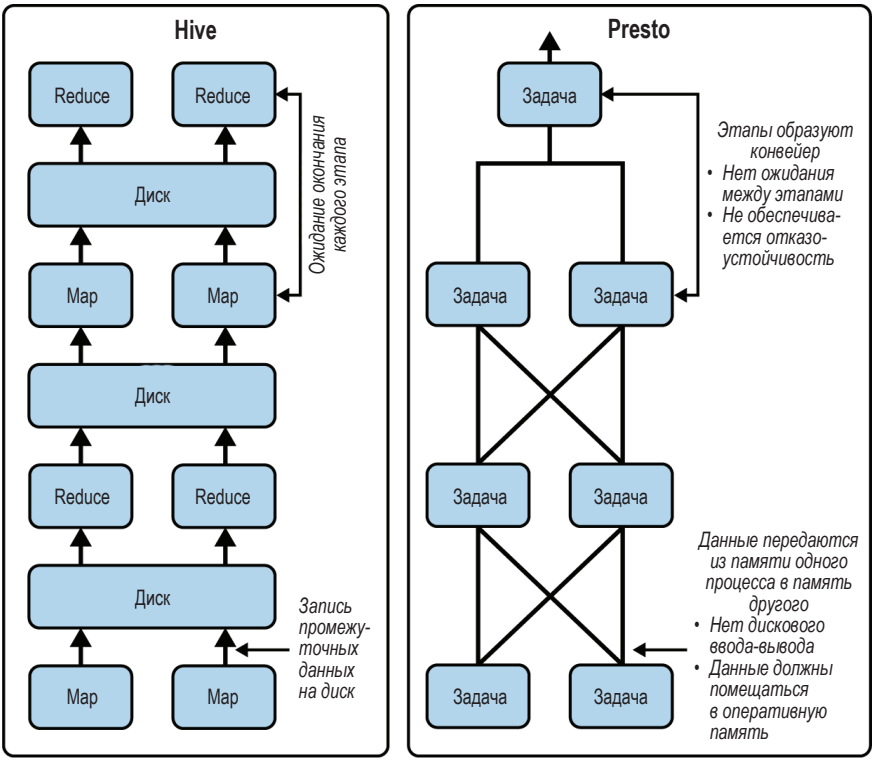

Рис. 1.3 ❖ Как Hive и Presto выполняют запросы

В 2013 году Facebook открыл исходный код Presto под лицензией Apache 2.0 и выложил репозиторий на GitHub. Позже Facebook спонсировал проект, передав его под управление Фонда Linux (Linux Foundation), который в свою очередь основал дочернюю организацию – фонд Presto (Presto Foundation).

Создатели Presto ставили перед собой следующие цели: высокая производительность, легкое масштабирование, соответствие стандарту ANSI SQL, объединение разрозненных данных и возможность запуска в облаке.

Высокая производительность

В оптимизаторе Presto реализовано множество правил, включая такие хорошо известные приемы, как проталкивание предикатов, исключение колонок, оптимизация коррелированных подзапросов. Сложный алгоритм в зависимости от возможностей конкретного источника выбирает, какие операции можно выполнить прямо на источнике. Например, СУБД могут самостоятельно выполнять фильтрацию, агрегацию, вычисление функций и т. д. Осуществляя эти операции максимально близко к данным, Presto существенно сокращает объем ввода-вывода и сетевой трафик. Часть запроса, например соединение данных из разных источников, выполняется непосредственно самим Presto.

Масштабирование

Благодаря особенностям архитектуры, с которыми вы познакомитесь в следующей главе, масштабирование Presto практически не ограничено, но при этом для первоначального запуска какая-то мощная инфраструктура вовсе не требуется. Прежде чем обрабатывать действительно большие наборы данных, вы можете разработать прототип вашей системы, используя маленькие инсталляции. Благодаря быстрому отклику платформы накладные расходы даже на запуск множества мелких запросов невелики.

Соответствие стандарту ANSI SQL

Синтаксис запросов Presto строго придерживается стандартов ANSI SQL. Поскольку большинство пользователей уже знают SQL, использование Presto не требует изучения нового языка. Практически все сценарии применения доступны прямо «из коробки».

Что значит «соответствует стандарту ANSI SQL»

Это значит, что такие команды, как `SELECT`, `UPDATE`, `DELETE`, `INSERT` и `JOIN`[1], в большинстве случаев делают именно то, что от них ожидается.

Объединение данных

Presto – это *федеративный SQL-движок* (*federated SQL-engine*), позволяющий запрашивать данные как из одного источника, так и из нескольких источников, в том числе и разнородных.

Для доступа к данным Presto использует коннекторы, позволяющие читать данные из разных источников и даже записывать их туда. Коннекторы для многих популярных баз данных и форматов уже написаны и доступны «из коробки».

Федеративные запросы

Федеративным запросом называется запрос, извлекающий данные из нескольких разнородных источников. Федеративный подход противопоставляется централизованному подходу, при котором данные перед обработкой должны быть сохранены в едином физическом формате. Presto изначально разработан для выполнения федеративных запросов.

На рис. 1.4 изображены базовые шаги, осуществляемые движком в процессе выполнения федеративного запроса[2]. Получив запрос, движок раз-

---

[1] В книге JOIN указывается именно как отдельная команда, хотя в стандарте SQL это только одна из частей команды SELECT. – *Прим. перев.*

[2] Kemele M. Endris, Maria-Esther Vidal, and Damien Graux, “Chapter 5, Federated Query Processing,” in Knowledge Graphs and Big Data Processing, ed. Valentina Janev, Damien Graux, Hajira Jabeen, and Emanuel Sallinger (Springer, 2020), https://oreil.ly/p7KFC.

бирает его (*стадия разбора, query parsing*) и выбирает один или несколько источников, данные из которых потребуются для выполнения запроса (*выбор источников, data source selection*). В результате этого шага запрос разбивается на подзапросы.

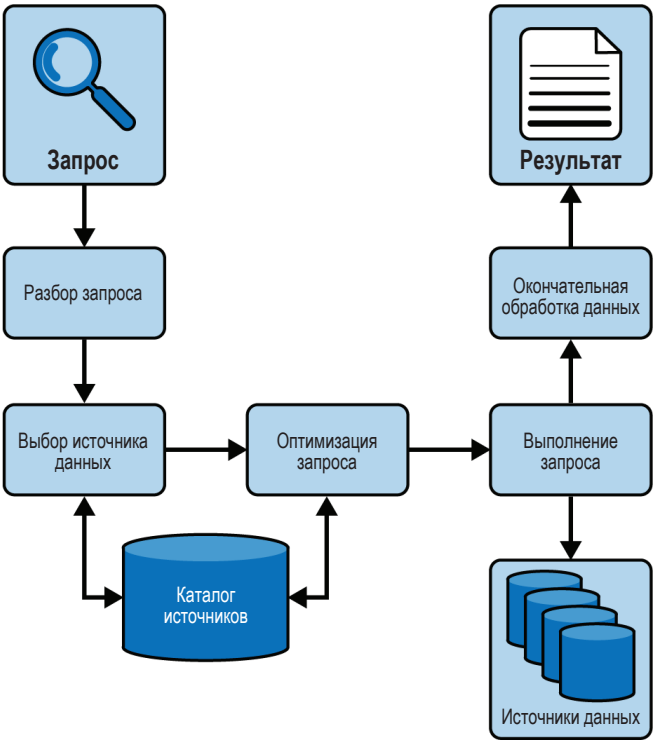

Рис. 1.4 ❖ Базовые шаги при обработке федеративного запроса

На следующем шаге строится логический план (*оптимизация, query optimization*), определяющий алгоритм выполнения запроса и набор операций (`JOIN`, `UNION`, `FILTER` и т. д.), которые будут использованы. Как правило, план представляется в виде *дерева* (*tree-based plan*), листья которого соответствуют подзапросам, а внутренние узлы представляют собой операции, которые будут выполнены самим движком.

После построения плана запроса наступает стадия *выполнения* (*query execution*)[1], во время которой подзапросы непосредственно выполняются, а затем их результаты приводятся к единому формату (*query reconciliation*) и из них собирается общий результат запроса.

[1] Авторы используют также термин «физический план» (physical plan) как синоним исполнения. – *Прим. перев.*