Введение

Архитектуры обработки данных

Google BigQuery — это бессерверное масштабируемое хранилище данных со встроенным механизмом запросов. Механизм запросов способен выполнять запросы SQL на терабайтах данных за считаные секунды, а на петабайтах — за считаные минуты. Чтобы получить такую производительность, вам не придется организовывать и поддерживать какую-либо инфраструктуру и создавать или перестраивать индексы.

Система управления реляционными базами данных

Информацию о сделках можно записывать в реляционную базу данных с оперативной обработкой транзакций (OnLine Transaction Processing, OLTP), например MySQL или PostgreSQL. Одним из ключевых преимуществ таких баз данных является поддержка запросов на языке структурированных запросов (Structured Query Language, SQL) — вашим сотрудникам не придется использовать высокоуровневые языки программирования, такие как Java или Python, чтобы получать ответы на возникающие вопросы. Они могут просто писать запросы, которые можно отправить серверу базы данных:

```
SELECT
  EXTRACT(YEAR FROM starttime) AS year,
  EXTRACT(MONTH FROM starttime) AS month,
  COUNT(starttime) AS number_one_way
FROM
  mydb.return_transactions
WHERE
  start_station_name != end_station_name
GROUP BY year, month
ORDER BY year ASC, month ASC
```

Не обращайте пока внимания на синтаксис, подробнее мы поговорим о SQLзапросах позже, а сейчас давайте сосредоточимся на преимуществах и недостатках баз данных OLTP.

Во-первых, обратите внимание, что язык SQL позволяет не просто получать исходные данные, хранящиеся в столбцах таблиц, — предыдущий запрос анализирует отметку времени и извлекает из нее год и месяц. Он также выполняет агрегирование (подсчитывает число строк), фильтрацию (выбирает сделки, в которых пункт, где было взято оборудование, не совпадает с пунктом его возвращения), группировку (по году и месяцу) и сортировку. Важным преимуществом SQL является возможность конкретизировать, что именно мы хотим получить, и позволить программному обеспечению базы данных определить оптимальный способ выполнения запроса.

К сожалению, базы данных OLTP довольно неэффективно выполняют подобные запросы. Основной упор в них делается на согласованность данных; дело в том, что базы данных позволяют считывать данные, даже когда в них одновременно производится запись. Это достигается за счет блокировок, обеспечивающих сохранение целостности данных. Чтобы фильтрация по

полю station_name выполнялась эффективно, необходимо создать индекс для поля с названием пункта проката. Только если название пункта проката индексируется, база данных будет выполнять специальные операции с хранилищем для оптимизации поиска, правда, при этом увеличение скорости чтения достигается за счет уменьшения скорости записи. Если название пункта проката не индексируется, фильтрация по этому полю будет работать довольно медленно. И даже если для названия пункта проката создать индекс, этот конкретный запрос все равно будет выполняться медленно из-за операций агрегирования, группировки и сортировки. Базы данных OLTP не созданы для таких ситуативных запросов, требующих выполнить обход всего набора данных.

Фреймворк MapReduce

Поскольку базы данных OLTP плохо подходят для ситуативных запросов и запросов, требующих обхода всего набора данных, специализированные виды анализа, требующие такого обхода, можно запрограммировать на языках высокого уровня, таких как Java или Python. В 2003 году Джефф Дин (Jeff Dean) и Санджай Гемават (Sanjay Ghemawat) заметили, что они и их коллеги из Google реализуют сотни таких специализированных вычислений для обработки больших объемов исходных данных. Для решения этой проблемы они разработали абстракцию, позволяющую выражать вычисления в виде двух шагов: функции map (отображения), которая обрабатывает пару ключ/значение и генерирует набор промежуточных пар ключ/значение, и функции reduce (свертки), которая объединяет все промежуточные значения, связанные с тем же промежуточным ключом. Эта парадигма, известная как MapReduce, оказала существенное влияние на фреймворки и привела к разработке Apache Hadoop.

Экосистема Hadoop начиналась как библиотека, написанная на Java, но теперь пользовательский анализ в кластерах Hadoop обычно выполняется с использованием Apache Spark (http://spark.apache.org/). Spark поддерживает программы, написанные на Python или Scala, а также позволяет выполнять специальные SQL-запросы к распределенным наборам данных.

То есть чтобы узнать количество платежей за аренду, можно организовать следующий конвейер данных:

1. Периодически экспортировать записи в текстовые файлы в формате CSV (Comma-Separated Values — значения, разделенные запятыми) в распределенной файловой системе Hadoop (Hadoop Distributed File System, HDFS). 2. Для выполнения ситуативного анализа написать программу Spark, которая:

а) выгружает данные из текстовых файлов в «DataFrame»;

б) выполняет SQL-запрос, подобный запросу, представленному в предыдущем разделе, за исключением того, что имя таблицы заменяется именем кадра данных DataFrame;

в) экспортирует результаты обратно в текстовый файл.

3. Выполнить программу Spark в кластере Hadoop.

На первый взгляд архитектура может показаться простой, тем не менее, в ней есть ряд скрытых недостатков. Для сохранения данных в HDFS необходим достаточно большой кластер. Еще одна немаловажная особенность архитектуры MapReduce, которой часто пренебрегают, заключается в том, что обычно вычислительным узлам требуется доступ к локальным для них данным. Соответственно, файловая система HDFS должна быть разбита на сегменты между вычислительными узлами кластера. Поскольку объемы данных и потребности в анализе быстро растут независимо друг от друга, часто происходит так, что кластеры испытывают нехватку или переизбыток вычислительных мощностей. То есть чтобы выполнять программы Spark в кластере Hadoop, сотрудникам вашей организации придется стать экспертами в управлении, мониторинге и настройке кластеров Hadoop. Это может не входить в ваши планы.

BigQuery: бессерверный распределенный движок SQL

А теперь представьте, что у вас есть возможность выполнять SQL-запросы как в системе управления реляционными базами данных (Relational Database Management System, RDBMS), эффективно выполнять обход распределенных наборов данных как в MapReduce и не обременять себя поддержкой инфраструктуры. Это третий вариант, и именно эти особенности делают BigQuery таким привлекательным. BigQuery — это бессерверная служба, позволяющая выполнять запросы, не требуя поддерживать свою инфраструктуру. Она дает возможность анализировать большие наборы данных и агрегировать их в считаные секунды или минуты.

Мы не призываем вас верить нам на слово. Попробуйте сами. Перейдите по ссылке https://console.cloud.google.com/bigquery (зарегистрируйтесь в Google Cloud Platform и, если потребуется, выберите свой проект), скопируйте и вставьте следующий запрос в окно, а затем щелкните на «Run query» («Выполнить»):

```
SELECT
  EXTRACT(YEAR FROM starttime) AS year,
  EXTRACT(MONTH FROM starttime) AS month,
  COUNT(starttime) AS number_one_way
FROM
  `bigquery-public-data.new_york_citibike.citibike_trips`
WHERE
  start_station_name != end_station_name
GROUP BY year, month
ORDER BY year ASC, month ASC
```

Когда мы запустили его, пользовательский интерфейс BigQuery сообщил, что запрос обработал 2.51 Гбайт данных и на это потребовалось примерно 2.7 секунды, как показано на рис. 1.1.

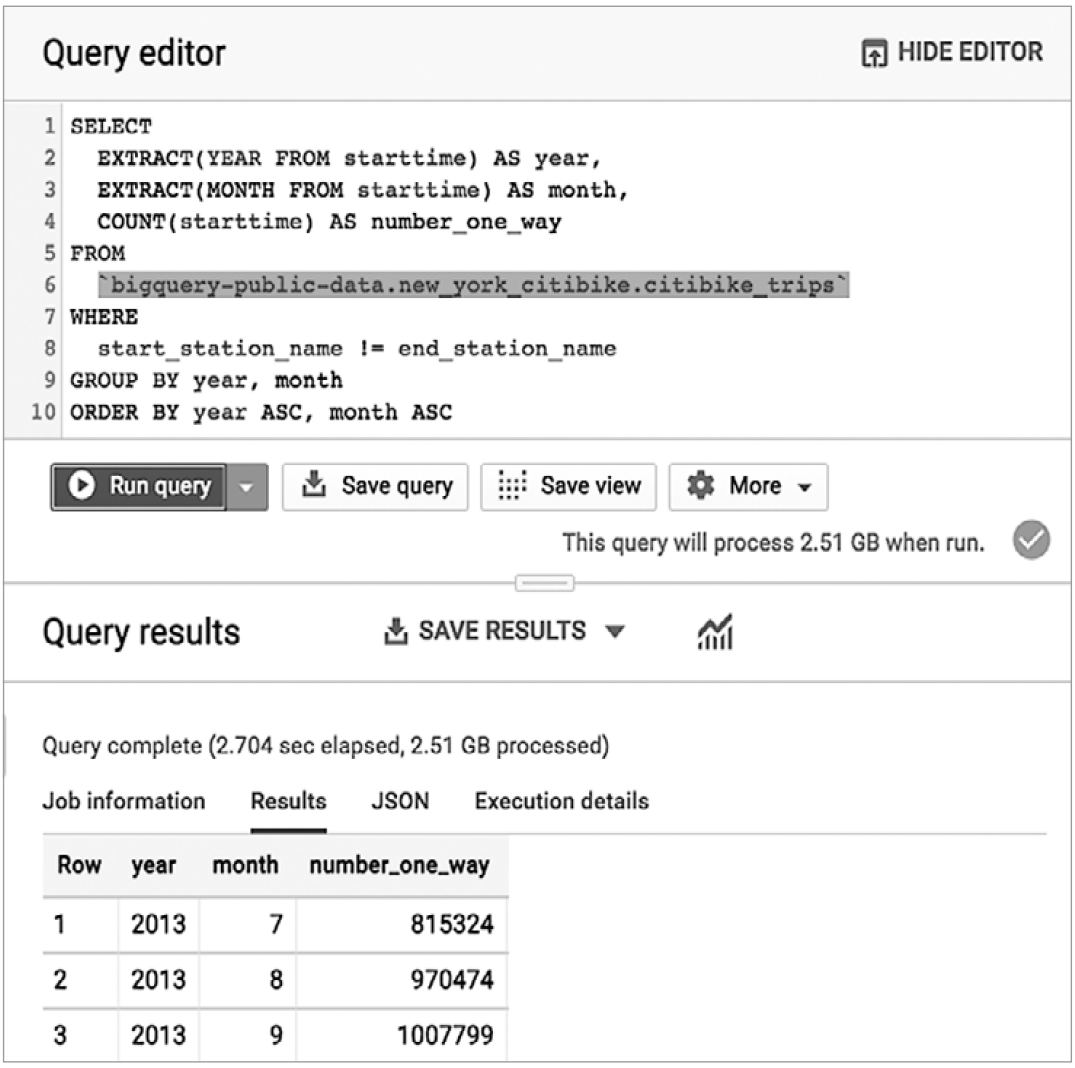

Рис. 1.1. Выполнение запроса для определения количества платежей за прокат в веб-интерфейсе BigQuery

Арендуемое оборудование — это велосипеды, таким образом, предыдущий запрос подводит помесячный итог проката велосипедов в Нью-Йорке по большому набору данных. Сам набор данных находится в открытом доступе (то есть любой желающий может запросить эти данные) и выпущен в Нью-Йорке в рамках инициативы «Открытый город». Результат этого запроса показывает, что в июле 2013 года в Нью-Йорке велосипеды арендовались 815 324 раза.

Обратите внимание на несколько моментов. Во-первых, вы смогли выполнить запрос к набору данных, который уже есть в BigQuery. Все, что должен сделать ответственный за проект, в котором размещены данные, — это предоставить вам доступ к этому набору данных для «просмотра». Вам не нужно запускать кластер или входить в него — вы просто отправляете запрос сервису и получаете результаты. Сам запрос написан на SQL:2011, который поддерживает синтаксис, хорошо знакомый аналитикам данных. Мы продемонстрировали пример обработки гигабайтов данных, но вообще сервис легко масштабируется и способен агрегировать терабайты и петабайты

данных. Такая масштабируемость возможна, потому что сервис распределяет обработку запросов между тысячами рабочих узлов практически мгновенно.

Работа с BigQuery

BigQuery — это хранилище данных с определенной степенью централизации. Запрос, продемонстрированный в предыдущем разделе, был применен к единственному набору данных. Однако преимущества BigQuery становятся еще более очевидными при объединении наборов данных из совершенно разных источников или при обращении к данным, хранящимся вне BigQuery.

Анализ наборов данных

Данные о прокате велосипедов поступают из Нью-Йорка. А можно ли объединить их с данными о погоде Национального управления океанических и атмосферных исследований США, чтобы узнать, как дождливая погода влияла на прокат велосипедов?

```
-- Дождливая погода влияла на прокат велосипедов?
WITH bicycle_rentals AS (
  SELECT
    COUNT(starttime) as num_trips,
    EXTRACT(DATE from starttime) as trip_date
  FROM `bigquery-public-data.new_york_citibike.citibike_trips`
  GROUP BY trip_date
),

rainy_days AS
(
  SELECT     date,
    (MAX(prcp) > 5) AS rainy
  FROM (
    SELECT
      wx.date AS date,
      IF (wx.element = 'PRCP', wx.value/10, NULL) AS prcp
    FROM
      `bigquery-public-data.ghcn_d.ghcnd_2016` AS wx
    WHERE
      wx.id = 'USW00094728'
  )
  GROUP BY   date )

SELECT
  ROUND(AVG(bk.num_trips)) AS num_trips,
wx.rainy
FROM bicycle_rentals AS bk
JOIN rainy_days AS wx
ON wx.date = bk.trip_date
GROUP BY wx.rainy
```

Пока оставим в стороне синтаксис запроса. Обратите внимание только на строки, выделенные жирным шрифтом. Здесь мы соединяем набор данных о прокате велосипедов и набор данных о погоде, взятый из совершенно другого источника. Результаты запроса подтверждают предположение, что жители Нью-Йорка слабаки — в дождливые дни берут велосипеды в прокат на 20% реже:

```
Row num_trips rainy
1       39107.0   false
2       32052.0   true
```

Что означает возможность совместного использования наборов данных и отправления запросов в контексте предприятия? Разные подразделения вашей компании могут хранить свои наборы данных в BigQuery и легко обмениваться ими с другими подразделениями компании и даже с партнерскими организациями. Бессерверные технологии BigQuery помогают устранить разрозненность подразделений и оптимизировать сотрудничество.

ETL, EL и ELT

Традиционный подход к работе с хранилищами данных включает три этапа: извлечение, преобразование и загрузку (Extract, Transform, Load; ETL), когда исходные данные извлекаются из источника, преобразуются и затем загружаются в хранилище данных. В действительности BigQuery поддерживает собственный высокоэффективный формат колоночного хранения, что делает методологию ETL особенно привлекательной. Конвейер обработки данных, обычно реализованный на основе Apache Beam или Apache Spark, извлекает необходимые исходные данные (потоковых данных или пакетных файлов), преобразует извлеченные данные, подготавливая их к очистке или агрегированию, а затем загружает их в BigQuery, как показано на рис. 1.2.

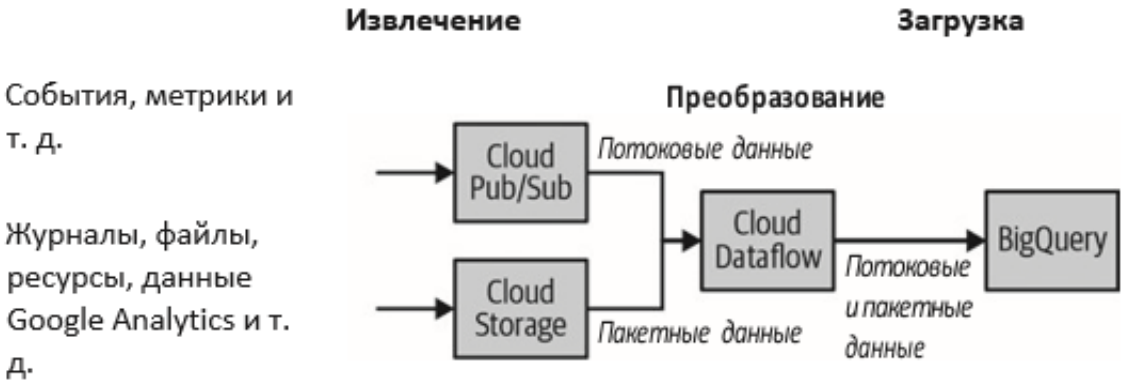

Рис. 1.2. Эталонная архитектура ETL в BigQuery использует конвейеры Apache Beam, выполняемые в Cloud Dataflow, и может обрабатывать как потоковые, так и пакетные данные с одним и тем же кодом

Apache Beam, выполняемые в Cloud Dataflow, и может обрабатывать как потоковые, так и пакетные данные с одним и тем же кодом

Хотя создание конвейера ETL в Apache Beam или Apache Spark весьма распространено, его можно создать исключительно в BigQuery. Так как BigQuery отделяет вычисления от хранилища, SQL-запросы BigQuery можно выполнять для файлов в формате CSV (а также JSON и Avro), которые хранятся в исходном виде в облачном хранилище Google Cloud Storage; эта возможность называется федеративным запросом. С помощью федеративных запросов можно извлекать данные, выполняя запросы SQL к Google Cloud Storage, преобразовывать данные внутри этих запросов SQL и сохранять результаты в собственных таблицах BigQuery.

Если преобразование не требуется, BigQuery может напрямую загружать стандартные форматы, такие как CSV, JSON или Avro, в свое хранилище — рабочий процесс EL (извлечение и загрузка). Такой подход, когда данные загружаются непосредственно в хранилище, используется потому, что хранение данных в собственном хранилище обеспечивает наибольшую эффективность запросов.

Мы настоятельно советуем реализовать процесс EL, если это возможно, и использовать процесс ETL, только если необходимы преобразования. По возможности выполните все необходимые преобразования в запросе SQL и сохраните весь конвейер ETL в BigQuery. Если преобразования трудно реализовать исключительно в SQL или если конвейер должен передавать данные в BigQuery по мере их поступления, создайте конвейер Apache Beam и выполняйте его бессерверным способом с использованием Cloud Dataflow. Другое преимущество реализации конвейеров ETL в Beam/Dataflow заключается в лучшем объединении таких конвейеров с системами непрерывной интеграции (CI) и модульного тестирования, потому что они содержат программный код.

Помимо процессов ETL и EL, BigQuery позволяет выполнять извлечение, загрузку и преобразование. Идея состоит в том, чтобы извлекать и загружать исходные данные «как есть» и использовать представления BigQuery для преобразования данных во время работы. Процесс ELT особенно удобен, если схема исходных данных постоянно меняется. Например, вы можете реализовать определение необходимости исправления конкретной временной метки с учетом местного часового пояса. Процесс ELT может пригодиться для создания прототипов и позволяет организации начать извлекать информацию из данных, не принимая потенциально необратимых решений на ранних этапах.

Весь этот винегрет может сбивать с толку, поэтому мы подготовили краткую сводку в табл. 1.1.

Таблица 1.1. Краткое описание процессов, примеры архитектур и сценарии, в которых они могут использоваться

Процесс	**Архитектура**	**Когда используется**
EL	Извлечение данных из файлов в Google Cloud Storage. Загрузка в собственное хранилище BigQuery. Можно вызывать из Cloud Composer, Cloud Functions или запланированных запросов	Пакетная загрузка архивных данных. Периодическая загрузка файлов журналов по расписанию (например, один раз в день)
ETL	Извлечение данных из Pub/Sub, Google Cloud Storage, Cloud Spanner, Cloud SQL и т. д. Преобразование данных с использованием Cloud Dataflow. Имеет конвейер Dataflow для записи данных в BigQuery	Когда требуется проверить качество, выполнить преобразование или обогатить исходные данные перед загрузкой в BigQuery. Когда загрузка данных должна происходить непрерывно, то есть если сценарий требует потоковой передачи. Когда необходимы интеграция с системами непрерывной интеграции/непрерывной доставки (CI/CD) и модульное тестирование всех компонентов
ELT	Извлечение данных из файлов в Google Cloud Storage. Хранение в BigQuery данных в формате, близком к исходному. Преобразование данных в процессе работы с использованием представлений BigQuery	При использовании экспериментальных наборов данных, когда еще неясно, какие преобразования необходимо применить, чтобы сделать данные пригодными для использования. С любыми промышленными данными, когда преобразование можно выразить в SQL

Процессы в табл. 1.1 перечислены в том порядке, в каком мы рекомендуем их применять.

Эффективная аналитика

Преимущества использования хранилища напрямую вытекают из видов анализа, которые можно выполнять с хранящимися в нем данными. Основной способ взаимодействия с BigQuery — выполнение запросов SQL, а поскольку BigQuery является движком SQL, вы можете использовать широкий спектр инструментов бизнес-аналитики (Business Intelligence, BI), таких как Tableau, Looker и Google Data Studio, для реализации разных видов анализа, визуальных диаграмм и отчетов на основе данных, хранящихся в BigQuery. Например, кликнув на «Explore in Data Studio» («Исследовать в Data Studio») в веб-интерфейсе BigQuery, можно быстро создать график помесячного изменения проката велосипедов, как показано на рис. 1.3.

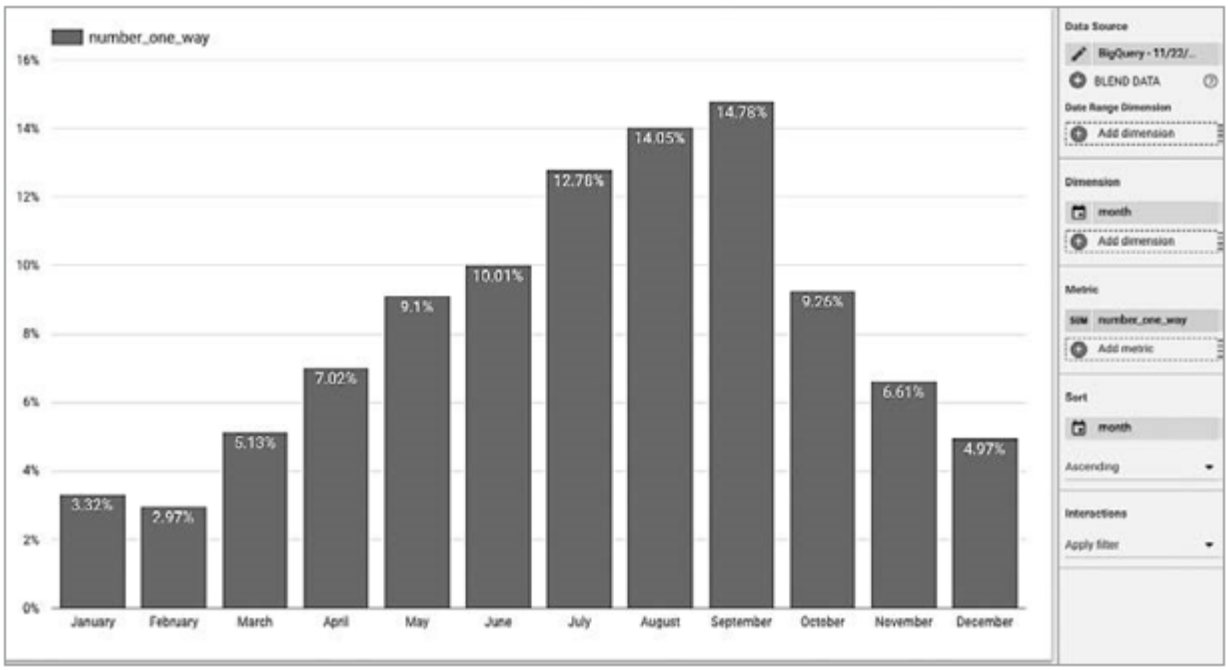

Рис. 1.3. Статистика в Data Studio проката велосипедов в зависимости от месяца; почти 15% проката в Нью-Йорке приходится на сентябрь

BigQuery предлагает полноценную поддержку SQL:2011, включая массивы и сложные соединения. В частности, поддержка массивов позволяет хранить в BigQuery иерархические данные (такие, как записи JSON) без преобразования вложенных и повторяющихся полей в плоские структуры. Помимо поддержки SQL:2011, BigQuery имеет несколько расширений, которые позволяют использовать этот сервис не только в качестве хранилища данных. Одним из таких расширений является поддержка широкого спектра геопространственных функций, позволяющих проверять в запросах местоположение, а также соединять таблицы на основе критериев расстояния или совпадения. Поэтому BigQuery является полезным инструментом для проведения описательной аналитики.

Другое расширение BigQuery — поддержка в стандартном SQL создания моделей машинного обучения и выполнения пакетных прогнозов. Мы подробно рассмотрим возможности машинного обучения в BigQuery в главе 9, но суть в том, что они позволяют обучать модели BigQuery и делать прогнозы, не экспортируя данные из BigQuery. Преимущества безопасности и локальности данных в этом случае огромны. То есть BigQuery — это хранилище данных, которое поддерживает не только описательную, но и прогнозную аналитику.

Идеология хранилища подразумевает возможность хранения данных разных типов. И действительно, BigQuery может хранить самые разные данные: числовые и текстовые данные как само собой разумеющееся, а также геопространственные и иерархические данные. BigQuery дает возможность хранить данные в упорядоченном виде, но это вовсе не обязательно, схемы могут быть очень разнообразными и сложными. Сочетание запросов с учетом пространственного местоположения, иерархических данных и машинного обучения делает BigQuery эффективным решением, не ограниченным рамками обычного хранилища и бизнес-аналитики.

BigQuery может принимать не только пакетные, но и потоковые данные. Данные можно передавать в BigQuery напрямую через REST API. Часто пользователи, которым требуется преобразовать данные — например, проводя вычисления с временным окном, — используют конвейеры Apache Beam, выполняемые сервисом Cloud Dataflow. И даже при передаче потоковых данных в BigQuery вы можете запросить их. Наличие общей инфраструктуры запросов для архивных (пакетных) и текущих (потоковых) данных открывает широкие возможности и упрощает многие процессы.

Простота управления

При разработке BigQuery немало внимания уделялось тому, чтобы обратить внимание пользователей на результаты анализа, а не на инфраструктуру. При вводе данных в BigQuery вам не нужно думать о разных типах хранилищ или о поиске золотой середины между скоростью работы и стоимостью услуги; хранилище полностью управляемое. На момент написания этой книги стоимость хранения автоматически снижается, если таблица не обновлялась в течение 90 дней.

Мы уже говорили, что в BigQuery не требуется настраивать индексы; ваши запросы SQL могут фильтровать результаты по любому столбцу, а BigQuery позаботится об их планировании и оптимизации. Более того, мы советуем писать запросы максимально понятно и разборчиво, а выбор стратегии оптимизации оставить для BigQuery. В этой книге мы будем говорить о настройке производительности, но в BigQuery она сводится в основном к логике и выбору соответствующих функций SQL. Вам не придется заниматься администрированием базы данных и решать такие задачи, как репликация, дефрагментация или восстановление после сбоев; обо всем этом позаботится BigQuery.

Запросы автоматически распределяются по тысячам машин и выполняются параллельно. Вам не нужно ничего предпринимать, чтобы запустить этот

процесс. Машины уже подготовлены для обработки различных этапов заданий; вам не придется их настраивать.
Отсутствие необходимости поддерживать инфраструктуру уменьшает число проблем, связанных с безопасностью. Данные в BigQuery автоматически шифруются как при хранении, так и при передаче. BigQuery заботится о безопасности многопользовательских запросов и изоляции заданий. Вы сможете организовать общий доступ к своим наборам данных с помощью сервиса Google Cloud Identity and Access Management (IAM), а также применять к наборам данных (а также таблицам и представлениям в них) различные меры безопасности, в зависимости от того, нужна ли вам открытость, возможность аудита или конфиденциальность.
В других системах обеспечение надежности, гибкости, безопасности и производительности инфраструктуры часто отнимает много времени. Учитывая, что BigQuery практически избавляет от необходимости администрирования, организации, использующие BigQuery, считают, что это дает их аналитикам дополнительное время, чтобы сосредоточиться на получении информации из своих данных.

Выводы
BigQuery — это масштабируемое хранилище данных, обеспечивающее быстрые бессерверные вычисления больших наборов данных с помощью SQL-запросов. Пользователи ценят масштаб и скорость BigQuery, но руководители компаний часто более высоко оценивают возможность перехода на более качественный уровень, который дает возможность выполнять специальные запросы с помощью бессерверных вычислений и позволяет принимать решения на основе данных во всех подразделениях компании.
Для загрузки данных в BigQuery можно использовать конвейер EL (часто применяется для периодической загрузки файлов журналов), конвейер ETL (когда необходимо обогащение данных или контроль качества) или конвейер ELT (для исследовательской работы).
Платформа BigQuery предназначена для аналитической обработки данных в реальном времени (OnLine Analytical Processing, OLAP) и предоставляет полноценную поддержку SQL:2011. Высокая скорость BigQuery достигается за счет инновационных инженерных решений, таких как использование колоночного хранилища, поддержка вложенных и повторяющихся полей, а также отделение вычислений от хранения, о котором Google продолжает публиковать статьи. BigQuery является частью экосистемы GCP инструментов анализа больших данных и тесно интегрируется как с элементами инфраструктуры (обеспечивающими, например, безопасность, мониторинг и ведение журналов), так и с элементами обработки данных и машинного обучения (такими, как потоковая передача, Cloud DLP и AutoML).

Основы запросов

Простые запросы

BigQuery поддерживает диалект SQL, совместимый с SQL:2011 (https://www.iso. org/standard/53681.html). Если спецификация неоднозначна или имеет пробелы, BigQuery следует требованиям, принятым в существующих движках SQL. Есть также области, для которых спецификации отсутствуют, например, машинное обучение; в таких случаях BigQuery определяет свой собственный синтаксис и семантику.

Извлечение записей с помощью SELECT

```
-- простая выборка
SELECT
  gender, tripduration
FROM
  `bigquery-public-data`.new_york_citibike.citibike_trips
LIMIT 5
```

Результаты должны выглядеть примерно так:

Row	gender	tripduration
1	male	371
2	male	1330
3	male	830
4	male	555
5	male	328

В табл. 2.1 перечислены три ключевых компонента имени `bigquery-publicdata`.new_york_citibike.citibike_trips.

Таблица 2.1. Основные объекты BigQuery и их описание[1]

Объект BigQuery	Имя	Описание
Проект	bigquery-public-data	Владелец хранилища, связанного с набором данных и его таблицами. Проект также регулирует использование всех других продуктов GCP
Набор данных	new_york_citibike	Наборы данных — это контейнеры верхнего уровня, которые используются для организации и управления доступом к таблицам и представлениям.

		Пользователь может иметь несколько наборов данных
Таблица/представление	citibike_trips	Таблица или представление должны принадлежать набору данных, поэтому перед загрузкой данных в BigQuery необходимо создать хотя бы один набор данных

Создание псевдонимов столбцов с помощью AS

По умолчанию имена столбцов в наборе результатов совпадают с именами столбцов в таблице, откуда извлекаются данные. Однако с помощью AS именам столбцов можно присвоить свои псевдонимы:

```
-- Определение псевдонимов для имен столбцов
SELECT
  gender, tripduration AS rental_duration
FROM
  `bigquery-public-data`.new_york_citibike.citibike_trips
LIMIT 5
```

Этот запрос вернул следующие результаты (у вас конкретные значения могут отличаться):

Row	gender	rental_duration
1	male	432
2	female	1186
3	male	799
4	female	238
5	male	668

Псевдонимы могут быть полезны при преобразовании данных. Например, без псевдонима следующий оператор:

```
SELECT
  gender, tripduration/60
FROM
  `bigquery-public-data`.new_york_citibike.citibike_trips
LIMIT 5
```

присвоит второму столбцу в наборе результатов имя автоматически:

Row	gender	f0_
1	male	6.183333333333334
2	male	22.166666666666668
3	male	13.833333333333334
4	male	9.25
5	male	5.466666666666667

Вы можете дать второму столбцу осмысленное имя, добавив псевдоним в запрос:

```
SELECT
  gender, tripduration/60 AS duration_minutes
FROM
  `bigquery-public-data`.new_york_citibike.citibike_trips
LIMIT 5
```

Этот запрос даст примерно такие результаты:

Row	gender	duration_minutes
1	male	6.183333333333334
2	male	22.166666666666668
3	male	13.833333333333334
4	male	9.25
5	male	5.466666666666667

Фильтрация с WHERE

Чтобы найти случаи, когда клиент брал велосипед в прокат меньше чем на 10 минут, можно отфильтровать результаты, возвращаемые оператором SELECT, добавив предложение WHERE:

```
SELECT
  gender, tripduration
FROM
  `bigquery-public-data`.new_york_citibike.citibike_trips
WHERE tripduration < 600
LIMIT 5
```

Как и ожидалось, теперь в набор результатов попали только записи, соответствующие поездкам, длившимся меньше 600 секунд:

Row	gender	tripduration
1	male	178
2	male	518
3	male	376
4	male	326
5	male	516

Предложение WHERE может содержать логические выражения. Например, вот как можно найти поездки, совершенные женщинами и длившиеся 5–10 минут:

```
SELECT
  gender, tripduration
FROM
  `bigquery-public-data`.new_york_citibike.citibike_trips
WHERE tripduration >= 300 AND tripduration < 600 AND gender =
'female' LIMIT 5
```

Также можно использовать ключевые слова OR и NOT. Например, следующее предложение WHERE поможет отобрать клиентов не женского пола (то есть мужчин и клиентов, не указавших свой пол):
WHERE tripduration < 600 AND NOT gender = 'female'
Для управления порядком вычислений можно использовать круглые скобки. Найти женщин, совершивших короткие поездки, а также всех мужчин можно так:
WHERE (tripduration < 600 AND gender = 'female') OR gender = 'male'
Предложение WHERE работает со столбцами в таблице, указанной в предложении
FROM; то есть внутри WHERE нельзя ссылаться на псевдонимы, указанные в SELECT. Иначе говоря, следующий запрос нельзя использовать для выбора только поездок, длившихся менее 10 минут:

```
SELECT
  gender, tripduration/60 AS minutes
FROM
  `bigquery-public-data`.new_york_citibike.citibike_trips
WHERE minutes < 10 -- ВНУТРИ WHERE НЕЛЬЗЯ ССЫЛАТЬСЯ НА
ПСЕВДОНИМЫ LIMIT 5
```

Вместо этого нужно повторить преобразование внутри предложения WHERE (более удачные альтернативы мы рассмотрим позже):

```
SELECT
  gender, tripduration / 60 AS minutes
FROM
  `bigquery-public-data`.new_york_citibike.citibike_trips
WHERE (tripduration / 60) < 10 LIMIT 5
```

SELECT *, EXCEPT, REPLACE

Из соображений экономии средств и производительности (которые мы подробно рассмотрим в главе 7) лучше выбирать только нужные столбцы. Однако если вы захотите выбрать все столбцы в таблице, то можете использовать оператор SELECT *:

```
SELECT
  *
FROM
  `bigquery-public-data`.new_york_citibike.citibike_stations
WHERE name LIKE '%Riverside%'
```

Здесь для поиска пунктов проката, которые в своем названии имеют слово Riverside, в предложении WHERE используется оператор LIKE.

Выбрать столбцы, исключив некоторые из них, можно с помощью оператора SELECT EXCEPT:

```
SELECT
* EXCEPT(short_name, last_reported)
FROM
  `bigquery-public-data`.new_york_citibike.citibike_stations
WHERE name LIKE '%Riverside%'
```

Этот запрос вернет те же результаты, что и предыдущий, за исключением двух столбцов (short_name и last_reported).

Чтобы выбрать все столбцы и изменить значения в одном из них, можно воспользоваться оператором SELECT REPLACE. Например, вот как можно увеличить на 5 число свободных велосипедов:

```
SELECT
* REPLACE(num_bikes_available + 5 AS num_bikes_available) FROM
  `bigquery-public-data`.new_york_citibike.citibike_stations
```

Подзапросы с WITH

Уменьшить повторяемость кода и получить возможность использовать псевдонимы можно с помощью подзапроса:

```
SELECT * FROM (
  SELECT
    gender, tripduration / 60 AS minutes
  FROM
    `bigquery-public-data`.new_york_citibike.citibike_trips
)
WHERE minutes < 10
LIMIT 5
```

Внешний оператор SELECT работает с данными, возвращаемыми внутренним подзапросом, заключенным в круглые скобки. Поскольку псевдоним определяется во внутреннем запросе, внешний запрос может использовать его в своем предложении WHERE.

Запросы с круглыми скобками могут быть довольно сложными для чтения, поэтому лучше использовать предложение WITH и заключать в него код, который в иных условиях определял бы подзапрос:

```
WITH all_trips AS (
  SELECT
    gender, tripduration / 60 AS minutes
  FROM
    `bigquery-public-data`.new_york_citibike.citibike_trips
)
SELECT * from all_trips
WHERE minutes < 10
LIMIT 5
```

В BigQuery предложение WITH действует как именованный подзапрос и не создает временных таблиц. Мы будем называть all_trips «промежуточным источником» («from_item») — это не таблица, но из него можно делать выборку.

Сортировка с ORDER BY

Управлять порядком следования записей в наборе результатов можно с помощью ORDER BY:

```
SELECT
  gender, tripduration/60 AS minutes
FROM
  `bigquery-public-data`.new_york_citibike.citibike_trips
WHERE gender = 'female'
ORDER BY minutes DESC
LIMIT 5
```

По умолчанию строки в результатах не упорядочены. При заданном значении столбца записи по умолчанию сортируются в порядке возрастания значений в этом столбце. Запросив вывести строки в порядке убывания и ограничив их число пятью, мы получили пять самых долгих поездок, совершенных женщинами:

Row	gender	minutes
1	female	250348.9
2	female	226437.93333333332
3	female	207988.71666666667
4	female	159712.05
5	female	154239.0

Обратите внимание, что мы упорядочиваем по значениям в столбце minutes — псевдониму. Поскольку предложение ORDER BY выполняется после SELECT, в нем можно использовать псевдонимы.

Агрегирование

В предыдущем разделе был показан пример, в котором мы преобразовывали секунды в минуты делением на 60, мы обрабатывали каждую запись в таблице и преобразовывали ее. Аналогично можно применять функции агрегирования ко всем записям, чтобы получить в результате только одну запись.

Агрегирование с GROUP BY

Вот как можно найти среднюю продолжительность поездок, совершенных мужчинами:

```
SELECT
  AVG(tripduration / 60) AS avg_trip_duration
FROM
  `bigquery-public-data`.new_york_citibike.citibike_trips
WHERE   gender = 'male'
```

Этот запрос вернет следующий результат:

Row	avg_trip_duration
1	13.415553172043886

Таким образом, средняя продолжительность поездки из числа совершенных мужчинами в Нью-Йорке составляет примерно 13.4 минуты. Однако поскольку набор данных постоянно обновляется, вы можете получить другой результат.
А что можно сказать о поездках, совершенных женщинами? Вы можете выполнить предыдущий запрос дважды, один раз для мужчин, а другой для женщин, но было бы слишком нерационально анализировать набор данных во второй раз, просто изменив предложение WHERE. Вместо этого можно воспользоваться предложением GROUP BY:

```
SELECT
  gender, AVG(tripduration / 60) AS avg_trip_duration
FROM
  `bigquery-public-data`.new_york_citibike.citibike_trips
WHERE
  tripduration is not NULL
GROUP BY   gender ORDER BY
avg_trip_duration
```

Этот запрос вернет следующие результаты:

Row	gender	avg_trip_duration
1	male	13.415553172043886
2	female	15.977472148805207
3	unknown	31.4395230232542

Теперь агрегированные значения вычисляются для каждой группы отдельно. Выражение SELECT может включать в себя поле, по которому выполняется группировка (gender), и функцию агрегирования (AVG). Обратите внимание, что на самом деле в наборе данных имеется три пола: мужской (male), женский (female) и неизвестный (unknown).

Подсчет записей с COUNT
Чтобы узнать, сколько записей участвовало в вычислении средних значений в предыдущем примере, можно добавить COUNT():

```
SELECT   gender,
  COUNT(*) AS rides,
```

```
  AVG(tripduration / 60) AS avg_trip_duration
FROM
  `bigquery-public-data`.new_york_citibike.citibike_trips
WHERE
  tripduration IS NOT NULL
GROUP BY   gender ORDER BY
avg_trip_duration
```

Этот запрос дал следующие результаты:

Row	gender	rides	avg_trip_duration
1	male	35611787	13.415553172043886
2	female	11376412	15.977472148805207
3	unknown	6120522	31.4395230232542

Фильтрация сгруппированных значений с HAVING

Отфильтровать результаты после группировки можно с помощью предложения HAVING. Вот как можно узнать, представители какого пола совершают поездки длительностью дольше 14 минут:

```
SELECT
  gender, AVG(tripduration / 60) AS avg_trip_duration
FROM
  `bigquery-public-data`.new_york_citibike.citibike_trips
WHERE tripduration IS NOT NULL
GROUP BY   gender
HAVING avg_trip_duration > 14
ORDER BY   avg_trip_duration
```

Этот запрос дал следующие результаты:

Row	gender	avg_trip_duration
1	female	15.977472148805209
2	unknown	31.439523023254203

Обратите внимание, что предложение WHERE можно использовать для фильтрации данных по полу или продолжительности поездки, но его нельзя применить

Агрегирование

для фильтрации по средней продолжительности, потому что она вычисляется только после группировки элементов (попробуйте и убедитесь сами!).

Поиск уникальных значений с DISTINCT

Какие гендерные значения присутствуют в столбце gender в наборе данных? Чтобы это выяснить, можно использовать GROUP BY, но есть более простой способ — SELECT DISTINCT:

```
SELECT DISTINCT   gender
FROM
  `bigquery-public-data`.new_york_citibike.citibike_trips
```

Этот запрос вернет набор результатов, содержащий всего четыре строки:

Row	gender
1	male
2	female
3	unknown
4	

Почему четыре? Что это еще за четвертая строка? Давайте посмотрим:

```
SELECT   bikeid,
tripduration,   gender
FROM
  `bigquery-public-data`.new_york_citibike.citibike_trips
WHERE gender = ""
LIMIT 100
```

Вот результаты этого запроса:

Row	bikeid	tripduration	gender
1	null	null	
2	null	null	
3	null	null	
...			

В данном случае отсутствие значений в столбце gender указывает на отсутствие или плохое качество данных. Мы обсудим отсутствующие данные (значения NULL) и способы их учета и преобразования в главе 3, а пока просто имейте в виду: если вы захотите отфильтровать значения NULL в предложении WHERE, используйте операторы IS NULL или IS NOT NULL, потому что другие операторы сравнения (=, !=, <, >) при применении к NULL будут возвращать NULL и поэтому никогда не будут соответствовать условию в предложении WHERE.

Теперь вернемся к запросу, определяющему уникальные значения в столбце gender с помощью DISTINCT. Важно отметить, что DISTINCT влияет на все результаты, возвращаемые оператором SELECT, а не только на столбец gender. Чтобы представить это наглядно, добавим второй столбец в список SELECT:

```
SELECT DISTINCT   gender,
usertype
FROM
  `bigquery-public-data`.new_york_citibike.citibike_trips
WHERE gender != ''
```

На этот раз набор результатов содержит шесть строк, то есть по одной строке для каждой уникальной пары значений gender и usertype (Subscriber или Customer), существующей в наборе данных:

Row	gender	usertype
1	male	Subscriber
2	unknown	Customer
3	female	Subscriber
4	female	Customer
5	male	Customer
6	unknown	Subscriber

Краткое руководство по массивам и структурам

В этом разделе мы предлагаем вашему вниманию краткое руководство по массивам. Оно необходимо, чтобы в следующей главе мы могли познакомить вас с многочисленными типами данных и функций на небольших наглядных наборах данных. Сочетание массивов (квадратные скобки в запросе) и UNNEST дает возможность поэкспериментировать с запросами, функциями и типами данных.

Например, если вы хотите узнать, как работает строковая функция SPLIT, попробуйте следующее:

```
SELECT
  city, SPLIT(city, ' ') AS parts
FROM (
  SELECT * from UNNEST([
    'Seattle WA', 'New York', 'Singapore'
  ]) AS city )
```

А вот результаты этого запроса:

Row	city	parts
1	Seattle WA	Seattle
		WA
2	New York	New
		York
3	Singapore	Singapore

Возможность жестко закодировать массив значений в самом запросе SQL позволяет экспериментировать с массивами и типами данных и избавляет от необходимости искать подходящий набор данных или ждать завершения долгих запросов. Более того, такие запросы обрабатывают 0 байт и, следовательно, за них не приходится платить.
Другой способ поэкспериментировать с набором значений основан на использовании UNION ALL для объединения однострочных операторов SELECT:

```
WITH example AS (
  SELECT 'Sat' AS day, 1451 AS numrides, 1018 AS oneways
  UNION ALL SELECT 'Sun', 2376, 936
  UNION ALL SELECT 'Mon', 1476, 736
)

SELECT * from example
WHERE numrides < 2000
```

Этот запрос вернет две записи из небольшого встроенного набора данных, в которых число поездок (numrides) меньше 2000:

Row	day	numrides	oneways
1	Sat	1451	1018
2	Mon	1476	736

В следующем уроке мы используем такие встроенные наборы данных с жестко заданными значениями, чтобы проиллюстрировать различные аспекты поведения некоторых типов данных и функций.