Содержание

Предисловие

Компания Facebook раскрыла исходный код Cassandra в июле 2008 года. Оригинальная версия была написана преимущественно двумя людьми: выходцами из Amazon и Microsoft. Большое влияние на нее оказала программа Dynamo – распределенное хранилище ключей и значений, впервые разработанное в Amazon. В Cassandra реализована модель репликации в духе Dynamo, не имеющая точек общего отказа, но при этом добавлена более эффективная модель данных на основе «семейства столбцов».

Я подключился к проекту в декабре того же года, когда компания Rackspace предложила мне разработать распределенную базу данных для своих нужд. Момент был выбран очень удачно, так как к моим услугам были все наиболее важные на тот момент масштабируемые базы данных с открытым исходным кодом – только выбирай. Несмотря на то что в активе Cassandra была только одна крупная система, ее архитектура показалась мне самой удачной, и я направил свои усилия на улучшение кода и создание сообщества.

Проект Cassandra был включен в инкубатор Apache и к моменту выхода из него в марте 2010 стал примером настоящей истории успеха. Среди его разработчиков числились компании Rackspace, Digg, Twitter и другие, которые не смогли бы написать свою базу данных с нуля, но вместе создали нечто значительное.

Сегодня Cassandra – совсем не та ранняя система, которая лежала (и до сих пор лежит) в основе поиска по папкам «Входящие» в Facebook; она превратилась в «безусловного лидера в области эффективной обработки транзакций» (по выражению Тони Бэйна) и пользуется заслуженной репутацией по части надежности и производительности в сочетании с высокой масштабируемостью.

По мере своего становления Cassandra привлекала все больше крупных пользователей, и, наконец, стало ясно, что без коммерческой поддержки не обойтись. Поэтому в апреле 2010 года мы вместе с Мэттом Пфейлем (Matt Pfeil) основали компанию Riptano. Способствовать внедрению Cassandra оказалось очень интересно и поучительно, в частности потому, что мы могли знакомиться с такими приложениями, которые не обсуждаются публично.

Также выявилась нужда в такой книге, как эта. Как и многие другие проекты с открытым исходным кодом, Cassandra исторически отличалась не самой лучшей документацией. Но даже после того как дела с документацией наладились, представление материала в виде книги все равно полезно.

Благодарю Эбена за то, что он взял на себя труд раскрыть науку и искусство разработки в среде Cassandra и развертывания системы. А читателю предоставляется возможность познакомиться с методичным изложением новых концепций.

— Джонатан Эллис
руководитель проекта Apache Cassandra,
сооснователь и технический директор компании DataStax

Предисловие

Я очень волнуюсь, сочиняя предисловие для нового издания книги «Cassandra. Полное руководство». Вы спросите, почему? Да потому, что это новое издание! Когда появилось первое издание этой книги, проект Apache Cassandra только-только появился на свет. С годами изменилось так много, что тогдашние пользователи с трудом узнали бы в сегодняшней базе данных ту, прежнюю. Всем известно, как трудно угнаться за такими быстро развивающимися проектами, как Apache Cassandra, и я безумно благодарен Джеффу за то, что он решился поведать о текущем состоянии дел миру.

Одно из самых важных новшеств этого издания – раздел о моделировании данных. Я не раз публично заявлял: модель данных – это то, что отличает успешный проект на основе Apache Cassandra от провального. Значительная часть книги посвящена тому, как правильно построить модель. Но про эксплуатационников тоже не забыли. В современной базе Apache Cassandra есть такие вещи, как виртуальные узлы, а также многочисленные средства для обеспечения согласованности данных – и все это объясняется в новом издании. В общем, рассказать есть о чем, так что полное руководство окажется весьма кстати!

Вне зависимости от поставленной цели очень хорошо, что вы решили побольше узнать об Apache Cassandra. Сейчас самое время включить этот инструментарий в свой арсенал. А опытным пользователям стоит освежить знания, чтобы не отстать от жизни. Как показывают недавние опросы, специалисты, знакомые с Apache Cassandra, – одни из самых востребованных и высокооплачиваемых на рынке разработки приложений и построения инфраструктуры. И это отчетливая тенденция в нашей индустрии. Если организации нужна база данных с высоким уровнем масштабирования, размещенная в нескольких центрах обработки данных и постоянно готовая к работе, то лучше Apache Cassandra не найти. Первая же попытка поиска вернет сотни компаний, связавших свою судьбу с нашей любимой базой данных. И для такого доверия есть основания – вы убедитесь в этом, читая книгу. Приложения естественно мигрируют в облако, а Cassandra продолжает поддерживать работу с динамично изменяющимися глобальными данными. Эта книга научит вас применять Cassandra в соб-

ственных приложениях. Сделайте что-нибудь удивительное и предъявите свою историю успеха.

И наконец, приглашаю вас присоединиться к преуспевающему сообществу Apache Cassandra. Его членов можно найти в любом уголке мира, поэтому оно является одним из самых важных нетехнических ресурсов для новых пользователей. Нам повезло иметь такое процветающее сообщество, благодаря его совместной работе база данных Apache Cassandra стала еще лучше. Можете начать с простого – ходите на встречи и конференции, где сможете завязать знакомство со своими коллегами. Потом у вас, возможно, появится желание расширить свое участие, например писать статьи в блоге или проводить презентации, обогащая тем самым коллективный опыт и помогая новичкам, которые идут по вашим стопам. А там, глядишь, дойдет дело и до самого важного в любом проекте с открытым исходным кодом – деятельности технического характера. Напишите код, который исправит ошибку или добавит новую возможность. Отправьте в JIRA сообщение об ошибке или запрос на новую функциональность. Такие действия – показатель активности и доброго здравия проекта. Вам не понадобится никакой особый статус, просто создайте учетную запись – и вперед! А если столкнетесь с затруднениями, загляните снова в эту книгу или обратитесь к сообществу. Мы всегда готовы прийти на помощь.

Заинтересовались? Это хорошо!

Но хватит слов, пора перелистнуть страницу и приступить к учебе.

— *Патрик Макфейдин*,
главный пропагандист
Apache Cassandra, компания DataStax

Вступление

Почему именно Apache Cassandra?

Apache Cassandra – бесплатная распределенная система хранения данных с открытым исходным кодом, которая принципиально отличается от реляционных систем управления базами данных (РСУБД).

Проект Cassandra получил статус инкубаторного проекта Apache в январе 2009 года. Вскоре после этого команда разработчиков, возглавляемая Джонатаном Эллисом, выпустила версию Cassandra 0.3, а затем версии стали выходить регулярно. Систему Cassandra применяют некоторые крупнейшие интернет-компании, в т. ч. Facebook, Twitter и Netflix.

Во многом ее популярность объясняется выдающимися техническими характеристиками. Она надежна, легко масштабируется и допускает настройку уровня согласованности данных. Операции записи выполняет с фантастической скоростью, система может хранить сотни терабайтов данных, децентрализована и симметрична, так что точки общего отказа отсутствуют. База данных характеризуется высокой доступностью, предлагаются также средства моделирования данных на основе языка Cassandra Query Language (CQL).

Интересна ли вам эта книга?

Эта книга ориентирована на различные аудитории. Она будет полезна:

- разработчикам высокомасштабируемых приложений для работы с очень большими объемами данных, в частности социальных приложений для Web 2.0 и интернет-магазинов;
- архитекторам приложений или данных, которым нужно понимать, какие есть варианты создания высокопроизводительных, децентрализованных, эластичных хранилищ данных;
- администратору или разработчику баз данных, который знаком со стандартными реляционными СУБД и хочет понять, как можно реализовать отказоустойчивое, согласованное в конечном счете хранилище данных;

- руководителю, желающему разобраться в достоинствах (и недостатках) Cassandra и других столбцовых баз данных, чтобы принять решение о технической стратегии;
- студентам, аналитикам и исследователям, занимающимся проектированием систем, в которых участвует Cassandra или иное нереляционное хранилище.

Эта книга представляет собой техническое руководство. В ряде отношений Cassandra предлагает новый взгляд на данные. Многие разработчики, получившие профессиональные навыки в последние 15–20 лет, хорошо освоили рассуждения о данных в реляционных или объектно-ориентированных терминах. В Cassandra модель данных совершенно иная, и поначалу осмыслить ее нелегко, особенно тем из нас, кто имеет предвзятые идеи о том, что такое база данных и какой она должна быть.

Работать с Cassandra могут не только Java-разработчики. Однако система написана на Java, поэтому если вы хотите изучить исходный код, то без ясного понимания Java никак не обойтись. И хотя знать Java, строго говоря, необязательно, знакомство с этим языком поможет лучше разобраться в исключениях, понять, как строить исходный код и как пользоваться некоторыми популярными клиентами. Многие примеры в книге написаны на Java. Но существуют также интерфейсы к Cassandra из C#, Python, node.js, PHP, Ruby и других языков.

Наконец, предполагается, что читатель ясно понимает, как работает веб, умеет пользоваться интегрированной средой разработки (IDE) и хотя бы немного знаком с типичными проблемами, которые возникают в приложениях, управляемых данными. Даже профессиональный разработчик или администратор, начав изучать Cassandra, может столкнуться с незнакомыми ранее инструментами. Так, для сборки Cassandra используется Apache Ant, а для доступа к исходному коду – Git. В тех случаях, когда для работы с примерами требуется что-то установить или настроить самостоятельно, мы стараемся помочь советом.

О содержании книги

Книга построена так, что каждая глава является независимым руководством – в той мере, в какой это возможно. Это важно для книги о проекте Cassandra, который ориентирован на разных пользователей и быстро изменяется. По аналогии с программным обеспечением книга организована по «модульному» принципу. Начинающим

изучать Cassandra лучше читать ее по порядку. Но даже если для вас начальные стадии – пройденный этап, все равно некоторые главы могут оказаться полезными автономными руководствами.

Ниже кратко описано содержание глав.

Глава 1 «*За пределами реляционных баз*» описывает историю невероятного успеха реляционных баз данных и недавнего восхождения нереляционных технологий, к каковым относится и Cassandra.

Глава 2 «*Введение в Cassandra*» содержит введение в Cassandra, в ней обсуждаются наиболее интересные особенности, отличающие ее от других продуктов, истоки и преимущества системы.

Глава 3 «*Установка Cassandra*» описывает порядок установки и запуска Cassandra, а также выполнение некоторых простейших операций.

Глава 4 «*Cassandra Query Language*» рассматривает модель данных в Cassandra с упором на отличия от традиционной реляционной модели. Здесь мы также изучим, как выразить модель на языке Cassandra Query Language (CQL).

Глава 5 «*Моделирование данных*» содержит введение в принципы и процессы моделирования данных в Cassandra. Для создания работоспособной схемы мы проанализируем хорошо известную предметную область.

Глава 6 «*Архитектура Cassandra*» поможет понять, что происходит при операциях чтения и записи и как базе данных удается достичь выдающихся характеристик в части надежности и высокой доступности. Мы рассмотрим некоторые внутренние механизмы, в т. ч. протокол распространения сплетен, вручение напоминаний (hinted handoff), исправление на этапе чтения, деревья Меркла и др.

В *главе 7* «*Настройка Cassandra*» показано, как описывать распределители, стратегии размещения реплик и осведомители (snitch). Мы настроим кластер и понаблюдаем за последствиями выбора различных конфигурационных параметров.

Глава 8 «*Клиенты*» описывается клиентов для различных языков, в т. ч. Java, Python, node.js, Ruby, C# и PHP, позволяющих абстрагировать низкоуровневый API Cassandra. Мы поможем разобраться в общих функциях драйверов.

В *главе 9* «*Чтение и запись данных*» на базе всего ранее изложенного мы узнаем, что Cassandra делает «под капотом» для чтения и записи данных. Мы также обсудим пакеты, облегченные транзакции и разбиение на страницы.

Глава 10 «*Мониторинг*» – после того как кластер запущен, встает задача о мониторинге его работы, характере использования памяти

и потоков и оценке общего состояния. В Cassandra реализован развитый интерфейс Java Management Extensions (JMX), которым мы воспользуемся для решения этих и других задач.

Глава 11 «*Обслуживание*» – текущее обслуживание кластера Cassandra упрощается благодаря инструментам, входящим в комплект поставки сервера. Мы покажем, как вывести узел из эксплуатации, балансировать нагрузку, получать статистику и решать другие стандартные задачи.

Глава 12 «*Настройка производительности*» – одна из наиболее выдающихся характеристик Cassandra – чрезвычайно высокое быстродействие. Но можно добиться еще большего за счет правильной настройки параметров памяти и хранилища данных, выбора оборудования, кэширования, задания размеров буферов и т. п.

Глава 13 «*Безопасность*» – часто можно встретить пренебрежительное отношение к технологиям NoSQL как недостаточно безопасным. Но Cassandra предоставляет средства аутентификации, авторизации и шифрования, настройке которых посвящена эта глава.

Глава 14 «*Развертывание и интеграция*» – и в конце мы поговорим о планировании развертывания кластера, в том числе на облачных платформах Amazon, Microsoft и Google. Мы также вкратце расскажем о нескольких технологиях, которые нередко используются совместно с Cassandra с целью расширения ее возможностей.

Версии Cassandra, используемые в этой книге

При написании книги мы использовали версии семейства Cassandra 3.X и драйвер DataStax для Java версии 3.0.

При упоминании возможностей, добавленных, начиная с версии 2.0, мы указываем, в какой именно версии они появились. Это может быть полезно читателям, работающим с ранними версиями и планирующим переход на новую.

Что нового во втором издании

Первое издание книги «Cassandra. Полное руководство» было вообще первой книгой на тему Cassandra и на протяжении нескольких лет оставалось авторитетным. Но с 2010 года многое в Cassandra изменилось – как в части самой технологии, так и в отношении сообщества, которое эту технологию развивает и поддерживает. Ниже перечислены основные изменения, которые мы внесли, чтобы восстановить актуальность материала.

В ногу со временем

Первое издание относилось к версии 0.7, выпущенной в 2010 году. В 2016 году мы уже работаем с версиями семейства 3.X. Самое важное новшество – появление языка CQL и отказ от старого Thrift API. Из других архитектурных изменений отметим вторичные индексы, материализованные представления и облегченные транзакции. В главе 2 приведена история выпуска версий, она поможет проследить за изменениями. Упоминая новые возможности в тексте, мы часто указываем, в какой версии они появились.

Поддержка разработчиков

С годами техника разработки и тестирования систем на основе Cassandra претерпела серьезные изменения, чему способствовали внедрение оболочки CQL (`cqlsh`) и постепенная замена клиентов, разработанных сообществом, драйверами, предлагаемыми компанией DataStax. Подробному описанию `cqlsh` посвящены главы 3 и 4, а драйверам – главы 8 и 9. В главе 9 мы также приводим детальные сведения о выполнении операций чтения и записи в Cassandra, это поможет читателю разобраться во внутренних механизмах работы и понять, к каким последствиям приводят те или иные решения.

Новый уровень эксплуатации Cassandra

По мере роста количества внедрений Cassandra в производственных системах индивидуальными пользователями и организациями расширялась и база знаний о возникающих проблемах и способах их разрешения. Чтобы накопленные знания не остались втуне, мы добавили новые главы о безопасности (глава 13) и развертывании и интеграции (глава 14), а также значительно расширили главы о мониторинге, обслуживании и настройке производительности (главы 10–12).

Графические выделения

В книге применяются следующие графические выделения:

Курсив

Новые термины, имена и расширения имен файлов.

`Моноширинный`

Листинги программ, а также элементы кода в основном тексте: имена переменных и функций, базы данных, типы данных, переменные окружения, предложения и ключевые слова языка.

Моноширинный полужирный
Команды и иные строки, которые следует вводить буквально.

Моноширинный курсив
Текст, вместо которого следует подставить значения, заданные пользователем или определяемые контекстом.

Так обозначается замечание общего характера

Так обозначается предупреждение или предостережение.

О примерах кода

Примеры кода, встречающиеся в этой книге, можно скачать со страницы https://github.com/jeffreyscarpenter/cassandra-guide.

Эта книга призвана помогать вам в работе. Поэтому вы можете использовать приведенный в ней код в собственных программах и в документации. Спрашивать у нас разрешение необязательно, если только вы не собираетесь воспроизводить значительную часть кода. Например, никто не возбраняет включить в свою программу несколько фрагментов кода из книги. Однако для продажи или распространения примеров из книг издательства O'Reilly на компакт-диске разрешение требуется. Цитировать книгу и примеры в ответах на вопросы можно без ограничений. Но для включения значительных объемов кода в документацию по собственному продукту нужно получить разрешение.

Мы высоко ценим, хотя и не требуем, ссылки на наши издания. В ссылке обычно указываются название книги, имя автора, издательство и ISBN, например: «Cassandra: The Definitive Guide, Second Edition, by Jeff Carpenter. Copyright 2016 Jeff Carpenter, 978-1-491-93366-4».

Если вы полагаете, что планируемое использование кода выходит за рамки изложенной выше лицензии, пожалуйста, обратитесь к нам по адресу permissions@oreilly.com.

Как с нами связаться

Вопросы и замечания по поводу этой книги отправляйте в издательство:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472

800-998-9938 (в США и Канаде)
707-829-0515 (международный или местный)
707-829-0104 (факс)

Для этой книги имеется веб-страница, на которой публикуются списки замеченных ошибок, примеры и прочая дополнительная информация. Адрес страницы: http://bit.ly/cassandra2e.

Замечания и вопросы технического характера следует отправлять по адресу bookquestions@oreilly.com.

Дополнительную информацию о наших книгах, конференциях и новостях вы можете найти на нашем сайте по адресу http://www.oreilly.com.

Читайте нас на Facebook: http://facebook.com/oreilly.

Следите за нашей лентой в Twitter: http://twitter.com/oreillymedia.

Смотрите нас на YouTube: http://www.youtube.com/oreillymedia.

Благодарности

Мы благодарны многим замечательным людям, которые способствовали выходу книги в свет.

Спасибо нашим рецензентам: Стью Худу (Stu Hood), Роберту Шнейдеру (Robert Schneider) и Гэри Дасбейбеку (Gary Dusbabek), которые высказали немало ценных замечаний к первому изданию, а также Эндрю Бейкеру (Andrew Baker), Эвану Эллиоту (Ewan Elliot), Кирку Дэмрону (Kirk Damron), Кори Коулу (Corey Cole), Джеффу Джирса (Jeff Jirsa) и Патрику Макфейдину (Patrick McFadin), которые рецензировали второе издание. Замечания Криса Джадсона (Chris Judson) оказали решающее влияние на окончательный вид главы 14.

Мы признательны Джонатану Эллису и Патрику Макфейдину, написавшим предисловия к первому и второму изданиям соответственно. Патрик, спасибо тебе за вклад в раздел об интеграции со Spark в главе 14.

Спасибо нашим редакторам, Майку Лоукидесу (Mike Loukides) и Мари Божуро (Marie Beaugureau), за постоянную поддержку и стремление сделать книгу лучше.

Джефф благодарит Эбена за предоставленную возможность принять участие в обновлении пользующейся уважением книги и за поддержку на протяжении всей работы.

Наконец, нас вдохновляли многочисленные великолепные разработчики, внесшие свой вклад в проект Cassandra. Снимаем шляпу перед всеми, кто создал такую элегантную и эффективную базу данных.

Глава 1

За пределами реляционных баз данных

Если в первый момент идея не кажется
абсурдной, она безнадежна.

— Альберт Эйнштейн

Цель этой книги – помочь разработчикам и администраторам баз данных в освоении важной технологии. По ходу дела мы будем сравнивать Cassandra с традиционными реляционными СУБД и покажем, как включить ее в свою среду.

Что не так с реляционными базами данных?

Если бы я спросил людей, чего они хотят,
они бы попросили более быструю лошадь.

— Генри Форд

Представьте себе некую модель данных, придуманную небольшой командой разработчиков в компании, насчитывающей тысячу работников. Она была доступна по протоколу TCP/IP, а обращаться к ней можно было из программ на разных языках, включая Java и веб-службы. Поначалу в модели могли разобраться разве что самые квалифицированные специалисты по информатике, но с течением времени она распространялась все шире, и заложенные в ней идеи ста-

новились понятнее. Для использования базы данных, построенной на основе этой модели, пришлось выучить новые термины и по-новому взглянуть на хранение данных. Но по мере того как идея обрастала готовыми продуктами, ее принимали на вооружение все новые компании и правительственные учреждения, в немалой степени потому, что база данных работала быстро – могла выполнять тысячи операций в секунду. И приносила колоссальные доходы.

А затем появилась новая модель.

Новая модель несла в себе угрозу, в основном по двум причинам. Во-первых, она сильно отличалась от старой модели, демонстративно споря с ней. И это пугает, потому что понять и принять нечто новое и радикально отличающееся всегда трудно. Последующие споры могут лишь укрепить упорствующих в отстаивании своих взглядов – взглядов, которые, возможно, сложились на основе всего предшествующего опыта и условий работы. Во-вторых, – и это, пожалуй, более важное препятствие – новая модель несет угрозу, потому что компании вложили огромные деньги в старую модель и получают от ее использования немалую прибыль. Смена курса кажется нелепой, даже невозможной.

Разумеется, мы говорим об иерархической базе данных Information Management System (IMS), придуманной в компании IBM в 1966 году.

IMS создавалась с целью использования в ракете для полета на Луну Сатурн-5. Ее архитектор Верн Уоттс (Vern Watts) посвятил ей всю свою профессиональную карьеру. Многие из нас знакомы с базой данных DB2, разработанной в IBM. Эта весьма популярная СУБД получила имя от своей предшественницы – программы DB1, построенной на основе иерархической модели данных IMS. IMS была выпущена в 1968 году и затем получила широкое распространение как составная часть системы Customer Information Control System (CICS) и других приложений. Она используется и по сей день.

Но спустя несколько лет после изобретения IMS появилась новая, угрожающая ей модель – реляционная база данных.

В статье «A Relational Model of Data for Large Shared Data Banks», вышедшей в 1970 году, д-р Эдгар Ф. Кодд пропагандировал теорию реляционной модели данных, созданную им во время работы в исследовательской лаборатории IBM в Сан-Хосе. Эта статья, и сейчас доступная по адресу http://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf, легла в основу всех работ по реляционным системам управления базами данных.

Работа Кодда была полярно противоположна иерархической структуре IMS. Для понимания реляционной базы данных и работы с ней

понадобились новые термины: «отношение», «кортеж», «нормальная форма» и другие, которые были совершенно непривычны пользователям IMS. Новая система предлагала принципиальные улучшения по сравнению с предшествующей, в частности возможность выражать сложные связи между сущностями – намного более сложные, чем допускали иерархические базы данных.

За сорок лет эти идеи и их применения подверглись существенному развитию, но и до сих пор реляционные базы данных, без сомнения, являются одними из самых успешных программ в истории. Мы встречаем их как в форме персональных баз данных типа Microsoft Access, так и в гигантских транснациональных корпорациях, эксплуатирующих кластеры из сотен тщательно настроенных узлов, в совокупности составляющих многотерабайтное хранилище данных. В реляционных базах хранятся счета-фактуры, сведения о заказчиках, каталоги товаров, бухгалтерские книги, схемы аутентификации пользователей – чуть ли не все существующие в мире данные. Несомненно, реляционные базы – важнейшая грань современного технологического и делового ландшафта, и они пребудут с нами в различных формах еще много лет, как, впрочем, и IMS в ее разных ипостасях. Реляционная модель – это альтернатива IMS, и у каждой имеются свои применения.

Поэтому кратким ответом на вопрос «Что не так с реляционными базами данных?» будет «Все так».

Однако существует и более развернутый ответ: время от времени рождаются идеи, которые, на первый взгляд, меняют все и порождают всяческие революции. Но если взглянуть на это конструктивно, то такие революции – не более чем обычный ход истории. IMS, РСУБД, NoSQL. Лошадь, автомобиль, самолет. Новое всегда основывается на предшествующем ему, каждая новая технология стремится решить определенные проблемы; она хороша для одних целей и не столь хороша для других. Все они мирно сосуществуют, даже сейчас.

Давайте поговорим о том, почему на настоящем этапе развития стоит поискать альтернативу реляционной базе данных. Точно так же сам Кодд сорок лет назад анализировал систему Information Management System и думал, что это, возможно, не единственно допустимый способ организации информации и решения задач хранения данных. И что, возможно, для некоторых задач было бы более плодотворно подумать об альтернативах.

Когда приложение, в котором применяется реляционная база, становится популярным и активно используется, мы сталкиваемся

с проблемой масштабируемости. Даже в реляционных базах скромного размера никуда не деться от операции соединения, а она может оказаться медленной. Согласованность в базе данных обычно обеспечивается за счет транзакций, а это подразумевает блокировку некоторой части базы, из-за чего она становится недоступной другим клиентам. При очень высокой нагрузке это может стать нетерпимым, потому что из-за блокировок запросы пользователей выстраиваются в очередь в ожидании возможности выполнить операцию чтения или записи.

Обычно эти проблемы решаются одним или несколькими из перечисленных ниже способов, часто именно в указанном порядке.

- Бросить на решение проблемы аппаратные ресурсы: добавить память, поставить более быстрые процессоры, модернизировать диски. Такой подход называется *вертикальным масштабированием*. Он может спасти на некоторое время.
- Когда проблема возникает снова, применяется похожее решение: раз эта машина достигла своего потолка, добавим оборудование в виде дополнительных машин, образующих кластер базы данных. Теперь встает проблема репликации и обеспечения согласованности в обычном режиме работы и в случае отказа. Такой проблемы раньше не было.
- Нам придется изменить конфигурацию системы управления базами данных. Возможно, нужно будет оптимизировать каналы, по которым база данных производит запись в файловую систему. Мы отключаем протоколирование или журналирование, что обычно нежелательно (а иногда просто невозможно).
- Сделав все возможное на уровне СУБД, мы обращаем внимание на приложение. Пытаемся улучшить структуру индексов. Оптимизируем запросы. Но надо полагать, что при таком-то масштабе мы уже позаботились об индексах и запросах, так что они и так настроены оптимально. Наступает черед мучительной ревизии кода доступа к данным в попытках найти хоть какие-нибудь возможности для улучшения. Это может быть отказ от некоторых соединений или их реорганизация, выбрасывание таких накладных средств, как обработка XML в хранимой процедуре, и т. д. Разумеется, изначально у нас были причины обрабатывать XML, поэтому если где-то делать это необходимо, то придется перенести обработку на уровень приложения и молиться, чтобы нигде ничего не сломалось.
- Мы включаем в систему уровень кэширования. В больших системах это может быть какой-нибудь распределенный кэш типа

memcached, Redis, Riak, EHCache. Теперь возникает проблема согласованного обновления кэша и базы данных, которая к тому же осложняется наличием кластера.

- Мы снова обращаем внимание на базу данных и приходим к выводу, что теперь, когда приложение создано и мы понимаем, как выполняются основные запросы, можно пойти на дублирование некоторых данных, адаптировав их к структуре запросов. Этот процесс денормализации идет вразрез с пятью нормальными формами, характеризующими реляционную модель, и прямо нарушает 12 правил Кодда. Мы напоминаем себе, что живем в реальном, а не в идеальном мире и делаем все, чтобы приложение откликалось за приемлемое время, пусть даже в результате оно перестает быть «чистым».

Двенадцать правил Кодда

Кодд составил список из 12 правил (на самом деле их 13, занумерованных от 0 до 12), формализующих его определение реляционной модели. Это было сделано в ответ на отклонение коммерческих СУБД от оригинальных идей. Эти правила Кодд изложил в двух статьях в журнале CompuWorld за октябрь 1985 года и формализовал во втором издании своей книги «The Relational Model for Database Management», которая больше не переиздается.

Наверное, все это вам знакомо. Инженеры, имеющие дело с системами масштаба веба, вправе задаться вопросом, а не напоминает ли ситуация известное высказывание Генри Форда о том, что более быстрая лошадь – не то, что нужно людям. И в поисках ответа на этот вопрос они проделали интересную и впечатляющую работу.

В этот момент мы должны осознать, что реляционная модель – это всего лишь модель. То есть полезный взгляд на мир, применимый к некоторому классу задач. Она никогда не задумывалась как исчерпывающий способ представления данных, который не подлежит пересмотру и не оставляет места альтернативам. Обратившись к истории, мы убедимся, что в свое время модель Кодда подрывала основы. В ней использовалась совершенно новая терминология, например знакомое слово «кортеж» в новом, дотоле не употреблявшемся смысле. К реляционной модели относились с подозрением, она подвергалась ожесточенным нападкам. В оппозицию к ней встал даже работодатель Кодда – компания IBM, которая построила приносящий немалые прибыли набор продуктов на основе IMS и вовсе не желала делиться своим куском пирога со всякими выскочками.

Но теперь реляционная модель занимает, наверное, лучшее место в доме под названием «мир данных». У языка SQL широкая поддержка, его хорошо понимают разработчики. SQL преподают на вводных курсах в университетах. Существуют базы данных с открытым исходным кодом, уже установленные на сервере интернет-провайдера, которыми может воспользоваться всякий, кто платит 4,95 доллара в месяц за хостинг. Поставщики облачных решений PaaS (Platform-as-a-Service – платформа как услуга) – Amazon Web Services, Google Cloud Platform, Rackspace и Microsoft Azure предлагают доступ к реляционной базе данных как услугу, включающую автоматизированный мониторинг и обслуживание. Зачастую выбор базы данных продиктован архитектурными стандартами, принятыми в организации. Но даже если таких стандартов нет, имеет смысл поинтересоваться, какая платформа уже используется в организации. Коллеги из отделов разработки и инфраструктуры тяжелым трудом накопили много полезных знаний.

За многие годы мы привыкли считать (быть может, по инерции), что реляционная база данных – решение на все случаи жизни.

Поэтому, наверное, лучше спрашивать не «Что не так с реляционными базами данных?», а «Какая у вас проблема?».

То есть мы хотим убедиться, что решение соответствует имеющейся задаче. Для некоторых задач реляционные базы подходят идеально. Но лавинообразный рост веба и в особенности социальных сетей ведет к такому же росту объема данных, которые необходимо обрабатывать. Когда в начале 1990-х годов Тим Бернерс-Ли обдумывал веб, предполагалось, что эта сеть будет использоваться для обмена научными документами между сотрудниками физических лабораторий. Но теперь веб проник повсюду, им пользуются как ученые, так и легионы пятилетних ребятишек, которые шлют друг другу смайлики с котятами. Среди прочего это означает необходимость поддерживать гигантские объемы данных, и тот факт, что веб с этим справляется, – памятник прекрасно продуманной архитектуре.

Однако часть этой инфраструктуры начинает прогибаться под непомерным весом.

Краткий обзор реляционных баз данных

Скорее всего, все это вам известно, но давайте все же освежим в памяти некоторые фундаментальные концепции реляционных баз данных. Это позволит заложить основу для рассмотрения недавних идей

касательно компромиссов, характерных для распределенных систем, в особенности очень больших – таких, которые необходимы в масштабе веба.

РСУБД: великие и не очень

Причин, по которым реляционные базы данных добились такого ошеломительного успеха за последние сорок лет, немало. Одной из самых важных является язык Structured Query Language (SQL), обладающий весьма развитой функциональностью и вместе с тем простым декларативным синтаксисом. Впервые SQL был принят в качестве стандарта ANSI в 1986 году; с тех пор вышло несколько версий, а производители включили нестандартные расширения синтаксиса, например Microsoft T-SQL и Oracle PL/SQL, для поддержки зависящих от реализации возможностей.

Своей эффективностью SQL обязан нескольким факторам. Он позволяет пользователю представлять сложные связи между данными с помощью команд языка манипулирования данными (Data Manipulation Language – DML) для вставки, выборки, обновления, удаления, усечения и объединения данных. Применяя функции, основанные на реляционной алгебре, мы можем выполнять различные операции, например находить минимальное и максимальное значения во множестве или фильтровать и сортировать результаты. Команды SQL поддерживают группировку и агрегирование данных. Благодаря языку определения данных (Data Definition Language – DDL) SQL предоставляет средства для непосредственного создания, изменения и удаления структурных элементов схемы без перезагрузки. SQL также позволяет предоставлять и отзывать права для отдельных пользователей и групп пользователей.

SQL прост в использовании. Для изучения базового синтаксиса нужно совсем немного времени, да и вообще на концептуальном уровне SQL и РСУБД не представляют особых сложностей. Начинающие разработчики быстро приобретают необходимые навыки, а, как часто бывает в отрасли с высоким темпом изменений, жесткими сроками и растущими затратами, простота использования может оказаться решающим фактором. А в данном случае простота свойственна не только синтаксису; существует много надежных инструментов, включающих интуитивно понятный графический интерфейс для просмотра и манипулирования данными.

Отчасти благодаря стандартизации SQL позволяет без труда интегрировать РСУБД с разнообразными системами. Нужно лишь

установить драйвер для языка, на котором написано приложение, – и можно начинать, причем уровень переносимости очень высок. Если вы решите переписать приложение на другом языке (или сменить базу данных), то часто для этого не потребуется много усилий, если, конечно, вы не загнали себя в угол чрезмерно усердным применением нестандартных расширений.

Транзакции, свойства ACID и двухфазная фиксация

Помимо всего вышеупомянутого, РСУБД и SQL поддерживают *транзакции*. Важнейшая особенность транзакций заключается в том, что сначала они выполняются виртуально, т. е. программист может отменить (откатить) изменения, если во время выполнения что-то пошло не так; если же все хорошо, то транзакцию можно зафиксировать. По выражению Джима Грея, транзакция – это «преобразование состояния», обладающее свойствами ACID (см. статью «The Transaction Concept: Virtues and Limitations» по адресу http://research.microsoft.com/en-us/um/people/gray/papers/theTransactionConcept.pdf).

Акроним ACID расшифровывается как «Atomic, Consistent, Isolated, Durable» – атомарность, согласованность, изолированность, долговечность. Это те свойства, которые позволяют оценить, насколько правильно и успешно выполнена транзакция.

Атомарность

Атомарность означает «все или ничего». Иными словами, чтобы выполнение транзакции было признано успешным, все команды обновления внутри нее должны завершиться успешно. Не может быть частичного успеха, когда одно обновление выполнилось, а другое – связанное с ним – нет. Стандартный пример – перевод денежных средств с одного счета на другой, т. е. уменьшение суммы средств на одном счете и увеличение на другом. Эта операция неделима – обе ее части должны быть успешно выполнены.

Согласованность

Согласованность означает, что данные переходят из одного корректного состояния в другое, тоже корректное, не оставляя стороннему наблюдателю возможности увидеть недопустимую совокупность значений. Например, транзакция, которая пытается удалить клиента и всю историю его заказов, не может оставить строки заказов, ссылающиеся на первичный ключ удаленного клиента; такое состояние является несогласованным, и попытка прочитать такие записи о заказах привела бы к ошибке.