Содержание

Пролог

Вездесущие графы, или Рождение известных нам графовых баз данных

Это было в 1999 году, мы работали по 23 часа в сутки. По крайней мере, чувствовали себя именно так. Каждый день приносил очередную новость о сумасшедшей идее, только что получившей финансирование в миллионы долларов. У наших конкурентов были сотни инженеров, а наша команда разработчиков состояла всего из 20 человек. Этого было явно недостаточно, причем 10 из наших инженеров большую часть времени проводили в борьбе с реляционной базой данных.

Нам потребовалось время, чтобы выяснить, почему так происходит. После тщательного анализа архитектуры данных, хранимых нашим приложением управления контентом, выяснилось, что программа должна не просто управлять массой отдельных изолированных *дискретных* элементов данных, но и учитывать *связи* между ними. Несмотря на то что наши дискретные данные легко размещались в таблицах реляционной базы данных, с хранением связей между ними возникли сложности, а запросы, извлекающие их, выполнялись чрезвычайно медленно.

Просто от отчаяния я вместе с Йоханом и Питером, соучредителями компании Neo, начал экспериментировать с другими моделями представления данных, в частности основанными на графах. Нас увлекла идея замены табличной семантической модели данных графо-ориентированной, которая должна была значительно облегчить навигацию по связанным данным. Мы поняли, что, вооружившись графовой моделью, наша команда перестанет тратить половину своего времени на борьбу с базой данных.

Конечно, мы понимали, что не являемся первопроходцами. Теория графов существует уже в течение почти 300 лет, и примеры ее применения в целом ряде разнообразных математических задач широко известны. Естественно, должны быть и базы данных, основанные на графах!

Ну, мы и прочесали с помощью AltaVista[1] весь молодой Интернет и ничего не нашли. Через несколько месяцев бесплодных поисков мы

---

[1] Это будет шоком для молодых читателей, но в истории человечества было время, когда Google не существовал. В те далекие времена землей правили динозавры и поисковые системы AltaVista, Lycos и Excite в основном использовались для поиска электронных порталов, предлагающих через Интернет приобрести корм для домашних животных.

(смело) приступили к созданию с нуля базы данных, которая изначально предназначалась для работы с графами. В нашем представлении она должна была сохранить все проверенные временем функции реляционной базы данных (транзакции, ACID, триггеры и т. д.), но при этом использовать модель данных XXI века. Так родился проект Neo, а вместе с ним графовые базы данных, в том виде, в котором они известны сегодня.

В первом десятилетии нового тысячелетия возникло несколько изменивших мир коммерческих проектов, в том числе Google, Facebook и Twitter. Им присуща одна общая особенность: они сделали ставку на графовую модель представления взаимосвязей между данными, сделав ее основой своего бизнеса. И вот, через 15 лет после появления, графовая модель используется повсеместно.

Facebook, например, базируется на идее, что, несмотря на значимость дискретной информации о людях, их именах, профессиях и т. д., еще большую ценность представляют сведения о *взаимосвязях* между ними. Основатель Facebook Марк Цукерберг (Mark Zuckerberg) построил свою империю, основываясь на понимании моделирования этих взаимосвязей с помощью *социального графа*.

Точно так же основатели компании Google – Ларри Пейдж (Larry Page) и Сергей Брин (Sergey Brin) – придумали, как хранить и обрабатывать не только отдельные веб-документы, но и как связать их между собой. Google создала *граф Интернета*, что и позволило ей стать самой впечатляющей компанией последнего десятилетия.

Сегодня графы успешно используются не только гигантами Интернета. Одна из самых больших в мире логистических компаний использует графовую базу данных для прокладки в режиме реального времени маршрутов отправлений, ведущая авиакомпания использует графы для управления метаданными медиаконтента, и финансовые структуры топ-уровня переводят всю свою инфраструктуру на Neo4j. Практически никому не известные несколько лет назад, графовые базы данных в настоящее время широко применяются в таких областях, как здравоохранение, розничная торговля, газонефтедобыча, средства массовой информации, разработка игр и др., ускоряя развитие каждой из них.

Реализация этих идей потребовала разработки инструментов нового типа, основанных на технологиях управления базами данных, учитывающих взаимосвязи между данными и позволяющих мыслить категориями графов, а это именно те инструменты, о которых мы мечтали, мучась с реляционной базой данных в 1999 году.

Я надеюсь, что эта книга станет вашим путеводителем по прекрасному развивающемуся миру графовых технологий и вдохновит вас на использование графовой базы данных в вашем следующем проекте, чтобы вы тоже смогли ощутить необычайную мощь графов.

Удачи!

Эмиль Эифрем (*Emil Eifrem*),
соучредитель Neo4j и генеральный директор Neo Technology
Менло-Парк, Калифорния
Май 2013 года

Предисловие

Сегодня решения, основанные на графовых базах данных, очень важны для ведения успешного бизнеса: управление, учитывающее сложные и динамичные отношения тесно связанных между собой данных, предоставляет преимущество в конкуренции. Везде, где требуется учитывать взаимосвязи между клиентами, абонентами телефонного центра или сети обработки данных, организаторами представлений и зрителями или же генами и белками, способность воспринимать и анализировать огромные графы тесно взаимосвязанных данных будет играть определяющую роль для компаний, которые превзойдут своих конкурентов в течение следующего десятилетия.

Для данных любого объема и значения графовые базы данных являются лучшим способом представления и извлечения взаимосвязанных данных. Взаимосвязи между данными представляют собой данные, описание и оценка которых нужны нам для понимания, каким образом связаны между собой составляющие элементы. Для достижения такого понимания потребуются идентификация и квалификация взаимосвязей между элементами данных.

Многие крупные корпорации, осознав эту необходимость некоторое время назад, начали создавать собственные запатентованные технологии обработки графов, но мы живем в эпоху, когда доступ к технологиям становится все более демократичным. Сегодня общедоступные графовые базы данных являются реальностью, позволяя основной массе пользователей испытать преимущества работы с взаимосвязанными данными без необходимости инвестировать в разработку собственной графовой инфраструктуры.

Примечательным является возрождение интереса к отображению данных посредством графов, притом что теория графов сама по себе не нова. Теория графов была впервые описана Эйлером в XVIII веке и до сих пор активно исследуется и улучшается математиками, социологами, антропологами и другими специалистами-практиками. Тем не менее только в последние несколько лет теория графов и представление с помощью графов стали применяться к управлению информацией. Графовые базы данных помогли решению важных проблем в области социальных сетей, управления нормативно-справочной информацией, картографии, экспертных рекомендаций и многих других. Такое повышенное внимание к графовым базам данных обусловлено двумя факторами: огромным коммерческим успехом таких компаний, как Facebook, Google и Twitter, бизнес-модели которых основываются

на собственных запатентованных технологиях графов, и появлением общедоступных графовых баз данных.

О втором издании книги

Первое издание этой книги была написано, когда Neo4j 2.0 находился в стадии активного развития и еще не были окончательно закреплены формы меток, индексов и ограничений целостности. Сейчас версия 2.x Neo4j уже просуществовала часть своего жизненного цикла (на момент написания книги доступна версия 2.2, и скоро выйдет версия 2.3) и мы уже с уверенностью можем включить в текст книги новые элементы графовой модели.

Для второго издания этой книги мы пересмотрели все примеры на языке запросов Cypher, чтобы привести их в соответствие с современным синтаксисом языка. Мы добавили метки для запросов и схем, включили пояснения декларативной индексации и дополнительных ограничений целостности Cypher. В другом месте мы добавили дополнительные правила моделирования, переработали описание внутренних свойств Neo4j в соответствии с изменениями его внутренней архитектуры и обновили тестовые примеры, применив в них новейшие инструменты.

О чем эта книга

Цель этой книги – познакомить разработчиков, специалистов по базам данных и лиц, принимающих решение о выборе технологий, с графами и графовыми базами данных, в основном ориентируясь на практическое применение этих технологий. Прочтя эту книгу, вы научитесь применять графовые базы данных на практике. Мы покажем, как графовая модель «формирует» данные и как мы извлекаем, обосновываем, понимаем и *воздействуем на* данные с помощью графовых баз данных. Мы рассмотрим все виды проблем, которые легко решаются при применении графовых баз данных, сопроводив пояснения практическими примерами, и опишем проектирование и реализацию решений, основывающихся на графовых базах данных.

Соглашения

В этой книге приняты следующие типографские соглашения:

Курсив
: Выделение новых терминов, URL-адресов, адресов электронной почты, имен файлов и расширений файлов.

`Моноширинный шрифт`
: Используется для оформления программного кода и фрагментов программного кода внутри абзацев, таких как имена переменных и функций, типы данных, переменные окружения, операторы и ключевые слова.

`Жирный моноширинный шрифт`
: Команды или другой текст, который должен быть введен пользователем.

`Наклонный моноширинный шрифт`
: Текст, который должен быть заменен пользовательскими значениями или значениями, определяемыми контекстом.

Так обозначаются советы, предложения и примечания общего характера.

Так обозначаются предупреждения и предостережения.

Использование примеров кода

Сопроводительные материалы (примеры кода, упражнения и т. д.) можно загрузить со страницы https://github.com/iansrobinson/graph-databases-use-cases.

Данная книга призвана оказать вам помощь в решении ваших задач. Вы можете свободно использовать примеры программного кода из этой книги в своих приложениях и в документации. Вам не нужно обращаться в издательство за разрешением, если вы не собираетесь воспроизводить существенные части программного кода. Например, если вы разрабатываете программу и используете в ней несколько отрывков программного кода из книги, вам не нужно обращаться за разрешением. Однако в случае продажи или распространения компакт-дисков с примерами из этой книги вам необходимо получить разрешение от издательства O'Reilly. Если вы отвечаете на вопросы, цитируя данную книгу или примеры из нее, получения разрешения не требуется. Но при включении существенных объемов программного кода примеров из этой книги в вашу документацию необходимо получить разрешение издательства.

Мы приветствуем, но не требуем добавлять ссылку на первоисточник при цитировании. Под ссылкой на первоисточник мы подразумеваем указание авторов, издательства и ISBN. Например: «Graph Databases by Ian Robinson, Jim Webber, and Emil Eifrem (O'Reilly). Copyright 2015 Neo Technology, Inc., 978-1-491-93089-2.».

За получением разрешения на использование значительных объемов программного кода примеров из этой книги обращайтесь по адресу permissions@oreilly.com.

Как связаться с нами

С вопросами и предложениями, касающимися этой книги, обращайтесь в издательство:

O'Reilly Media, Inc.
1005 Gravenstein Highway North Sebastopol, CA 95472
800-998-9938 (США или Канада)
707-829-0515 (международный и местный)
707-829-0104 (факс)

Список опечаток, файлы с примерами и другую дополнительную информацию вы найдете на странице книги http://bit.ly/modern_php.

Свои пожелания и вопросы технического характера отправляйте по адресу bookquestions@oreilly.com.

Ищите нас в Facebook: http://facebook.com/oreilly.

Следуйте за нами в Twitter: http://twitter.com/oreillymedia.

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

Благодарности

Мы хотели бы поблагодарить наших технических рецензентов: Майкла Хангера (Michael Hunger), Колина Джека (Colin Jack), Марка Нидхама (Mark Needham) и Прамода Садалага (Pramod Sadalage).

Выражаем признательность и благодарность редактору первого издания Натану Джепсону (Nathan Jepson).

Наши коллеги из Neo Technology внесли свой вклад в написание этой книги, потратив на это массу времени и сил. В частности, спасибо Андерсу Наврозу (Anders Nawroth) за его неоценимую помощь, Андресу Тейлору (Andrés Taylor) за помощь во всем, что касается Cypher, и Филиппу Расли (Philip Rathle) за его полезные советы.

Большое спасибо всем членам Neo4j-сообщества за огромный вклад в разработку графовой базы данных.

Отдельное спасибо нашим семьям за их любовь и поддержку: Лотти, Тигер, Эллиот, Кэт, Билли, Мадлен и Нооми.

Это второе издание стало возможным благодаря работе Кристины Эскалант (Cristina Escalante) и Майкла Хангера (Michael Hunger). Спасибо вам обоим за вашу неоценимую помощь.

Глава 1

Введение

Большая часть книги посвящена графовым моделям данных, но она не рассказывает о теории графов[1]. Для использования графовых баз данных не требуются глубокие теоретические познания, достаточно наличия общих представлений о графах. Давайте освежим наши сведения о графах.

Что такое граф?

Формально граф – это набор *вершин* и *ребер*, или, говоря более простым языком, набор *узлов* и *взаимосвязей* между ними. В графах объекты представлены узлами, а способы, которыми эти объекты соединены между собой, – взаимосвязями. Эта универсальная и выразительная структура позволяет моделировать всевозможные сценарии, от постройки космической ракеты до строительства системы дорог, от поставок продуктов питания до историй болезни населения, и многое другое.

Графы повсюду

Графы чрезвычайно полезны при анализе самых разных наборов данных в таких областях, как наука, государственное управление и бизнес. Реальный мир, в отличие от основанной на шаблонах устаревшей модели реляционных баз данных, разнообразен и взаимосвязан: в одних местах равномерно и упорядочено, в других случайно и нерегулярно. После освоения графов вы начнете замечать их присутствие повсюду. Гартнер (Gartner, http://www.gartner.com/id=2081316), например, выделяет в мире бизнеса пять видов графов: социальный, целевой, потребления, интересов и мобильности – и утверждает, что способность использовать эти графы обеспечивает «устойчивое преимущество в конкурентной среде».

---

[1] Информацию о теории графов можно найти в книгах Ричарда Дж. Труде (Richard J. Trudeau), «Introduction To Graph Theory» (Dover, 1993) и Гарри Чартранд (Gary Chartrand), «Introductory Graph Theory» (Dover, 1985). О применении теории графов для анализа сложных событий и поведения можно узнать из книги Дэвида Исли (David Easley) и Джон Клейнберг (Jon Kleinberg) «Networks, Crowds, and Markets: Reasoning about a Highly Connected World» (Cambridge University Press, 2010).

Например, данные в Twitter легко представить в виде графа. На рис. 1.1 изображена небольшая сеть пользователей Twitter. Каждый узел помечен как «пользователь», с указанием его роли в сети. Узлы соединены взаимосвязями, определяющими дополнительный семантический контекст, а именно: Билли следует за Гарри, а Гарри, в свою очередь, следует за Билли. Рут и Гарри так же следуют друг за другом, но, к сожалению, хотя Рут следует с Билли, Билли (пока) не ответил взаимностью.

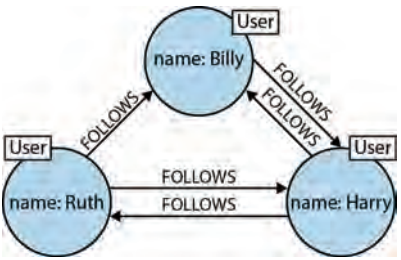

Рис. 1.1 ❖ Малый социальный граф

Конечно же, реальный граф Twitter в сотни миллионов раз больше, чем изображенный на рис. 1.1, но работает по тому же принципу. На рис. 1.2 мы расширили граф, включив в него сообщения, опубликованные Рут.

Несмотря на свою простоту, рис. 1.2 демонстрирует выразительную мощь графового моделирования. Из него легко понять, что Рут опубликовала последовательность сообщений. Ее самое последнее сообщение можно найти по взаимосвязи, помеченной как `ТЕКУЩЕЕ`. Далее следуют сообщения Рут, помеченные как `ПРЕДЫДУЩЕЕ`.

Модель графов с метками и свойствами

При обсуждении рис. 1.2 мы неформально представили самую популярную графовую модель – *графовую модель с метками и свойствами* (в приложении А более подробно будут рассмотрены альтернативные графовые модели). Графовая модель с метками имеет следующие характеристики:

- содержит узлы и взаимосвязи;
- у узлов есть свойства (пары ключ-значение);
- узлы должны быть помечены одной или более метками;
- взаимосвязи имеют имя и направление, для них всегда определен начальный и конечный узлы;
- у взаимосвязей также имеются свойства.

Большинство считает, что графовая модель со свойствами является простой и интуитивно понятной. Но, несмотря на свою простоту, именно она используется в подавляющем большинстве случаев при применении графов для отображения данных.

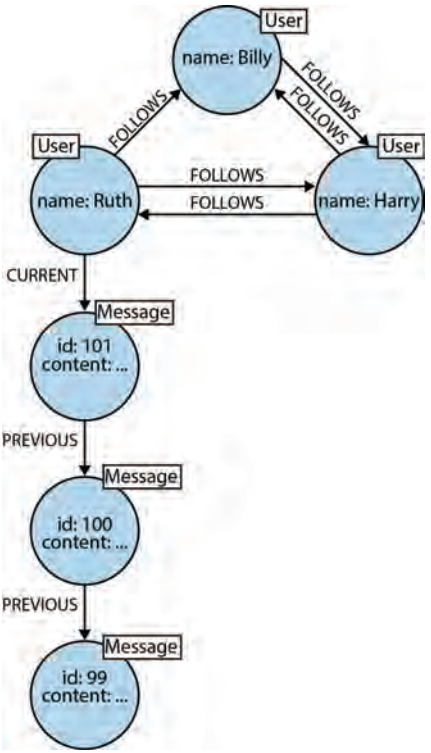

Рис. 1.2 ❖ Опубликованные сообщения

Обзор областей применения графов

В последние годы появилось множество проектов и готовых решений для управления, обработки и анализа графов. Огромное их количество затрудняет мониторинг инструментов и различий между ними, даже для тех, кто активно ими пользуется. Этот раздел содержит обобщенную их классификацию, чтобы помочь сформировать представление об областях применения графов.

С высоты птичьего полета области применения графов можно разделить на две части:

1) *технологии организации транзакционных графовых хранилищ, как правило, используемых в режиме реального времени непосредственно из приложения*. Эти технологии называются *графовыми базами данных*, и им в этой книге будет уделено основное

внимание. Они являются аналогом «нормальной» обработки транзакций в масштабе реального времени (On-Line Transaction Processing, OLTP) реляционных баз данных.

2) *технологии автономного анализа графов, обычно реализующие серии пакетных шагов*. Эти технологии часто называют *механизмами вычисления графов* (*graph compute engine*). Их можно отнести к той же категории, что и другие приемы анализа объемных данных, например интеллектуальный анализ данных и аналитическая обработка данных в реальном времени (On-Line Analytical Processing, OLAP).

Существует и еще один способ классификации применения графов, основанный на видах графовых моделей. Имеются три доминирующие графовые модели: графы со свойствами, триплексы схем описания ресурсов (Resource Description Framework, RDF) и гиперграфы. Они подробно рассмотрены в приложении А. Большая часть популярных графовых баз данных, представленных на рынке, использует графовые модели со свойствами, поэтому именно эта модель и будет использована в остальной части этой книги.

Графовые базы данных

Система управления графовыми базами данных (далее *графовые базы данных*) поддерживает методы создания (Create), чтения (Read), изменения (Update) и удаления (Delete) (CRUD), основанные на графовой модели данных. Графовые базы данных, как правило, поддерживают систему транзакций реального времени (OLTP). Соответственно, они оптимизированы для выполнения транзакций и спроектированы с учетом транзакционной целостности и оперативности.

Имеются две особенности графовых баз данных, которые необходимо учитывать при рассмотрении применяемой ими технологии:

1) **принцип хранения.** Некоторые графовые базы данных используют *специализированные хранилища графов*, предназначенные и оптимизированные для хранения и обработки именно графов. Но такую технологию хранения используют не все графовые базы данных. Некоторые сериализуют графы и размещают их в реляционной, объектно-ориентированной или какой-то другой базе данных или хранилище;
2) **порядок обработки.** Некоторые определения требуют, чтобы графовая база данных использовала *смежность без индексов*

(*index-free adjacency*), т. е. физическое соединение узлов друг с другом[1]. Мы придерживаемся более широких взглядов: любую базу данных, которая с точки зрения пользователя *ведет* себя как графовая (т. е. предоставляет графовую модель данных через CRUD-операции), квалифицируется как графовая база данных. Но мы признаем значительные преимущества в производительности при использовании смежности без индексов и, следовательно, используем термин *специализированная обработка графов* для графовых баз данных, использующих смежность без индексов.

Важно отметить, что принципы специализированного хранения графов и специализированной обработки графов не хороши и не плохи – они просто являются классическими инженерными компромиссами. Преимущество специализированного хранения графов – в том, что оно предусматривает стек, специально разработанный для повышения производительности и масштабируемости. Преимуществом неспециализированного хранения графов является использование зрелых неграфовых интерфейсов (например, MySQL), особенности которых хорошо знакомы разработчикам. Специализированная обработка графов (смежность без индексов) демонстрирует лучшую производительность обхода, но некоторые запросы приводят к использованию больших объемов памяти.

Взаимосвязи в графовой модели данных являются гражданами первого сорта. Здесь к ним относятся не так, как в других системах управления базами данных, где для отображения взаимосвязей применяются такие механизмы, как внешние ключи или внешние операции, например MapReduce. Собирая абстракции узлов и взаимосвязей в связанные структуры, графовая база данных позволяет строить модели любой сложности, лучше всего отражающие предметную область. Полученные модели проще и в то же время нагляднее, чем те, что создаются с помощью традиционных реляционных баз данных или других NOSQL-хранилищ.

На рис. 1.3 приведен графический обзор некоторых графовых баз данных из представленных сегодня на рынке, основанных на разных моделях хранения и обработки.

---

[1] Более подробную информацию можно найти в статье Родригеса Марко А. (Rodriguez Marko A.), Питера Неубауэр (Peter Neubauer) «The Graph Traversal Pattern». 2011 (http://arxiv.org/abs/1004.1001), в сборнике «Graph Data Management: Techniques and Applications», под ред. Шерифа Сарка (Sherif Sakr), Эрика Пардеде (Eric Pardede), 29–46. Hershey, PA: IGI Global.

Рис. 1.3 ❖ Обзор графовых баз данных

Механизмы вычисления графов

Механизмы вычисления графов позволяют выполнять глобальные графовые вычислительные алгоритмы для больших наборов данных. Они предназначены для решения таких задач, как идентификация кластеров данных или получение ответов на такие вопросы, как: «Сколько всего взаимосвязей, сколько их в среднем, полна ли социальная сеть?»

Из-за своей направленности на глобальные запросы механизмы вычисления графов, как правило, оптимизированы для сканирования и пакетной обработки больших объемов информации, и в этом отношении они похожи на другие технологии пакетного анализа, такие как интеллектуальный анализ данных (data mining) или аналитическая обработка в реальном времени (OLAP), используемые в реляционном мире. Некоторые механизмы вычисления включают в себя и средства хранения графов, а другие (большинство) заботятся только об обработке данных, получаемых из внешнего источника, а затем возвращают результаты для сохранения в другом месте.

Рисунок 1.4 иллюстрирует типовую архитектуру развертывания механизмов вычисления графов. Она включает в себя систему записи (System of Record, SOR) базы данных со свойствами OLTP (например, MySQL, Oracle или Neo4j), которая обслуживает запросы и отвечает на запросы, поступающие от приложений (и в конечном счете от пользователей). Периодически задания на извлечение, преобразование и загрузку данных (Extract, Transform, Load, ETL) перемещают

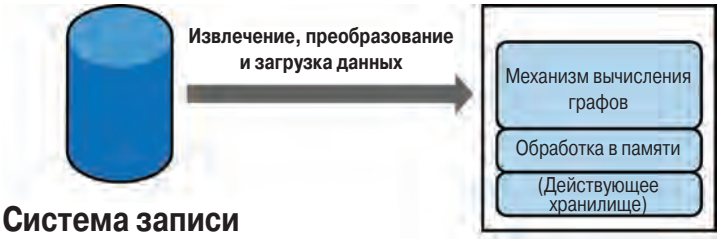

Рис. 1.4 ❖ Укрупненная схема типичной среды движков расчетов графов

данные из системы записи базы данных в механизм вычисления графов для выполнения автономных запросов и анализа.

Существуют разные типы механизмов вычисления графов. Наиболее известными из них являются *одномашинные* (*in-memory/single machin)*, такие как Cassovary (https://github.com/twitter/cassovary), и *распределенные*, такие как Pegasus (http://www.cs.cmu.edu/~pegasus/) и Giraph (http://giraph.apache.org/). В основе большинства распределенных механизмов вычисления графов лежит идея, изложенная в статье «Pregel: a system for large-scale graph processing», опубликованной на сайте Google (http://dl.acm.org/citation.cfm?id=1807184), которая описывает движок Google для классификации страниц.

Эта книга посвящена только графовым базам данных

В предыдущем разделе был приведен общий обзор областей применения графов. Остальная часть этой книги будет посвящена исключительно графовым базам данных. Далее речь пойдет о *концепции* графовых баз данных. При необходимости их рассмотрение будет иллюстрироваться примерами, взятыми из собственного опыта разработки решений с использованием графовой модели с метками и свойствами, и базы данных Neo4j. Независимо от того, какая графовая модель или база данных использована в примерах, иллюстрируемые ими идеи применимы и к другим графовым базам данных.

Преимущества графовых баз данных

Несмотря на то что практически все можно представить в виде графа, мы живем в прагматичном мире бюджетов, проектов с жесткими графиками, корпоративных стандартов и коммерческих правил. Предоставляемый графовыми базами данных новый мощный метод моделирования данных сам по себе не является достаточным основанием для замены устоявшихся и понятных платформ обработки данных – от

этого должна быть незамедлительная и весьма значительная практическая польза. Для графовых баз данных такой мотивацией может послужить применение ее в тех случаях и к таким моделям данных, когда при переходе на графовую модель будет достигнуто увеличение производительности на один и более порядков. Вместе с выигрышем в производительности графовые базы данных предоставляют чрезвычайно гибкую модель данных и способ развертывания, соответствующий современным способам развертывания программного обеспечения.

Производительность

Одной из веских причин выбора графовой базы данных является большой прирост производительности при работе со взаимосвязанными данными, по сравнению с реляционными базами данных и NOSQL-хранилищами. В отличие от реляционных баз данных, где учет взаимосвязей интенсивно ухудшает производительность запросов на больших наборах данных, производительность графовых баз данных остается неизменной с увеличением объема хранимых данных. Это связано с тем, что запросы локализуются в определенной части графа. В результате время выполнения каждого запроса зависит от размера части графа, которую требуется обойти для удовлетворения запроса, а не от общего размера графа.

Гибкость

Разработчикам и проектировщикам необходимо организовать взаимосвязи между данными, согласно требованиям области применения, структура данных должна соответствовать изменяющимся потребностям, а не навязываться заранее и оставаться неизменной. В графовых базах данных эта задача легко решается. Как мы увидим в главе 3, графовая модель данных отражает и охватывает потребности бизнеса таким образом, что может *изменяться со скоростью изменения самого бизнеса*.

Присущая графам возможность расширения означает, что можно добавлять новые виды взаимосвязей, новые узлы, новые метки и новые подграфы в существующую структуру, не нарушив при этом существующих запросов и функционала приложения. Это положительно влияет на производительность разработки и снижает риски для проекта. Благодаря гибкости графовой модели не требуется предварительно моделировать задачу в мельчайших подробностях, что очень неудобно из-за быстро меняющихся бизнес-требований. Способность графов к расширению также позволяет уменьшить ко-

личество миграций, что снижает нагрузку при обслуживании данных и уменьшает риск потери данных.

Оперативность

Модель данных должна не отставать от прочих составных частей приложения и использовать технологии, соответствующие современным итерационным методам развертывания программного обеспечения. Современные графовые базы данных оснащены всем необходимым для разработки и системного обслуживания. В частности, встроенная графовая модель данных, лишенная схем, в сочетании со встроенным программным интерфейсом (API) и языком запросов позволяет эффективно вести разработку приложений.

В то же время благодаря отсутствию схемы графовые базы данных не предполагают наличия ориентированных на схемы механизмов контроля данных, которые широко применяются в реляционном мире. Но в этом нет ничего страшного, здесь они заменены гораздо более удобными и действенными видами контроля. Как мы увидим в главе 4, контроль выполняется в программной форме, с помощью тестов для моделей данных и запросов, а также с помощью определения бизнес-правил, основанных на графе. Сейчас такая методика уже не вызывает сомнений: разработка с помощью графовых баз данных полностью соответствует современным методикам гибкой и надежной разработки программного обеспечения, что позволяет разработке приложений с использованием графовых баз данных не отставать от бизнес-среды.

Итоги

В этой главе мы рассмотрели графовую модель со свойствами, представляющую собой простой, но удобный инструмент для работы со взаимосвязанными данными. Графовая модель со свойствами хорошо моделирует области ее применения, а графовые базы данных облегчают разработку приложений, которые реализуют графовые модели.

В следующей главе мы сравним несколько различных технологий обработки взаимосвязанных данных, начнем с реляционных баз данных, затем перейдем к агрегированным NOSQL-хранилищам и закончим графовыми базами данных. Обсудив их, мы узнаем, почему графы и графовые базы данных являются лучшим средством для моделирования, хранения и выборки взаимосвязанных данных. Затем, в последующих главах, будут описаны проектирование и реализация решений, основывающихся на графовых базах данных.