# Оглавление

# Предисловие

Идея написания этой книги оформилась, когда мы вместе преподавали в Международном университете Сараево. В ходе обсуждения с нашими студентами, которые работали в местных компаниях, мы поняли, что структуры для массивных данных получают все большее распространение в повседневном применении инженерами и исследователями данных. Эти технологии использовались по всему миру не только компаниями Google и Facebook, чтобы решать свои задачи обеспечения масштабируемости, но и компаниями с гораздо меньшими объемами данных, чьи системы начали сталкиваться с постоянно растущими потребностями в скорости обработки данных.

За обедом мы размышляли о том, куда студент, который учится внедрять структуры данных HyperLogLog или фильтр Блума в работающую производственную систему, мог бы обратиться за удобным обзором их применения. Оригинальные статьи, в которых эти структуры данных всесторонне представляются, нередко были очень глубокими с математической точки зрения, но с малым контекстом для инженера данных, пытающегося встроить эту структуру данных в реальную систему с реальными данными. Если не считать эпизодических блог-постов с описанием реализации структур данных, ресурсов, которые объединяли бы эти специфичные для массивных данных алгоритмические знания, было мало либо вообще не существовало.

Мы хотели написать книгу, которая могла бы представить эти высокотехничные предметы в дружественном тоне, а также дать более качественный ответ на вечный вопрос студентов: «где это можно использовать?» Сочетание вероятностных и потоковых структур данных, а также структур данных внешней памяти в живую экосистему массивных данных и демонстрация практических примеров использования были непростым делом для двух преподавателей в вельветовых пиджаках. Мы не были готовы полностью отказаться от математики, поэтому поставили перед собой задачу попытаться выразить как можно больше математических концепций в легко воспринимаемой на интуитивном уровне форме, не приводя ни одного доказательства.

Нам чрезвычайно повезло работать с Инес, иллюстратором с передовым инженерным образованием, которая создала действенные и очаровательные рисунки для иллюстрации сложных алгоритмических материалов. Если вы когда-либо объясняли кому-либо алгоритм, то знаете, что он по своей сути визуален, однако в книгах по компьютерным алгоритмам зачастую не так много визуальных подсказок. Будем надеяться, что эта книга станет еще одним маленьким шагом к тому, чтобы это изменить.

Каждая хорошая история нуждается в конфликте, и в этой книге главным является компромисс, возникающий из-за ограничений, налагаемых крупными данными. Главнейшая тема нашей книги – пожертвовать точностью структуры данных ради экономии пространства. Отыскание этой золотой середины в производительности и усвоение способов уравновешивания разных конкурирующих целей в сложном конвейере данных – вот главные вызовы, которые привносятся массивными данными в повестку дня, и ключевые уроки, которые можно извлечь из этой книги.

Мы признательны за то, что у нас была возможность написать книгу на такую захватывающую и важную тему. Мы невероятно благодарны всем, кто оставлял отзывы, пока книга находилась в разработке. Начав писать ее как ученые, мы закончили ее как инженеры в компаниях, специализирующихся на обработке данных (это действительно практическая книга!). Надеемся, что ознакомление с этим материалом обогатит ваш набор алгоритмических инструментов и позволит вам взяться за решение следующей задачи обработки больших данных с любопытством и уверенностью.

# Благодарности

С момента, когда начинаешь писать книгу, и до самого конца происходит много событий, и выдавать главы на-гора, ориентируясь во всех превратностях жизни и работы, не всегда легко. К счастью, у нас была целая деревня людей, которые нас поддерживали, подбадривали на протяжении всего процесса и приносили еду накануне сдачи этапов работы.

Во-первых, мы хотели бы поблагодарить наших родителей Мердзану и Сафера, а также Зикрету и Эсада. Ваши примеры и руководство на протяжении всей жизни дали нам свободу читать, учиться и осваивать, а также дали нам почувствовать, что мы можем – и должны – писать книги. Без этого данная книга никогда бы не появилась на свет. Мы также хотели бы поблагодарить наших дорогих братьев, сестер и племянниц, которые поддерживали нас на протяжении всего процесса написания: Дзейру, Энсар, Аджлу, Сериф, Мерсад, Далал и Лейну. Эмин также хотел бы поблагодарить свою тетю Индиру за то, что она терпела его во время учебы во Франкфурте.

Во-вторых, мы хотели бы поблагодарить наших друзей, которые неоднократно интересовались продвижением работы над книгой (даже зная, что им придется выслушивать реально длинный ответ). Многие наши друзья работают в областях, отличных от вычислительных наук, поэтому их желание ознакомиться с нашими первыми главами имело гораздо большее значение.

Мы хотели бы поблагодарить наших студентов из Сараевской школы науки и технологий и Международного университета Сараево, которые вдохновили нас на написание этой книги и в разное время ее рецензировали.

Завершением работы над этой книгой мы обязаны нашему редактору Карен Миллер, которая проделала великолепную работу, проведя нас через весь процесс с невероятным сочетанием профессионализма и доброты. Ее проницательность и опыт сыграли решающую роль в формировании данной книги.

В процессе написания мы были на связи со многими сотрудниками издательства Manning. Коллектив издательства стремится к совершенству и применяет гибкий подход к изданию книг с учетом ранней обратной связи, что мы, как авторы, сочли невероятно полезным и вдохновляющим.

Мы хотели бы поблагодарить рецензентов издательства Manning: Алехандро Беллогина, Алекса Гаута, Анто Аравинта, Арно Бастенхофа, Кристофера Коттмайера, Чунсу Тана, Клиффорда Тербера, Даниэля Васкеса, Диего Казеллу, Германа Гонсалеса-Морриса, Хильде Ван Гизель, Жан-Франсуа Морина, Йенса Кристиана Бредаля Мэдсена, Джима Амрейна, Хуана Хосе Дурильо Барриону-эво, Хуана Антонио Руфес де Висенте, Келум Сенанаяке, Ману Сарина, Маркуса Янга, Марка Бауэра, Ниака Васкеса, Раушана Джа, Руи Лю, Сатей Саху, Себастьена Джанаса, Стюарта Перкса, Тима ван Дюрзена, Трэвиса Нельсон и Юрия Кушча. Ваши предложения помогли сделать эту книгу лучше.

Наконец, мы хотели бы поблагодарить наших читателей, благодаря чьему участию и вкладу книга была значительно лучше приспособлена для целевой аудитории.

# Об этой книге

Книга «Алгоритмы и структуры для массивных наборов данных» предназначена для оказания помощи в разработке масштабируемых приложений и понимании алгоритмических строительных блоков, лежащих в основании систем обработки массивных данных. В книге рассматриваются разные алгоритмические аспекты разработки крупномасштабных приложений, которые предусматривают экономию места за счет использования вероятностных структур данных, обработку потоковых данных, работу с данными на диске и понимание компромиссов в производительности в системах управления базами данных.

## Кому следует прочитать эту книгу

Эта книга предназначена для читателей, разбирающихся в фундаментальных структурах данных и алгоритмах. Бóльшая часть содержимого этой книги основана на материале, который обычно рассматривается на ранних курсах по структурам данных / алгоритмам: большинство глав начинаются с демонстрации традиционного решения задачи и причины, по которой тот или иной алгоритм либо структура данных оказываются безуспешными в контексте массивных данных. Несмотря на то что вводные разделы глав предлагают некоторое обсуждение базовых алгоритмов, этот материал служит лишь кратким обзором тематик, с которыми читатель уже должен чувствовать себя удобно. Читатель данной книги также должен обладать промежуточными знаниями в области программирования и понимать основы теории вероятностей. Никаких знаний о какой-либо конкретной системе или фреймворке не требуется (в этом вся красота алгоритмов), кроме базового знакомства с языком Python и псевдокодом.

## Как эта книга организована: дорожная карта

Книга состоит из трех частей, охватываемых 11 главами. Часть I посвящена вероятностным лаконичным структурам данных, часть II – потоковым структурам данных и алгоритмам, а часть III – структурам данных и алгоритмам внешней памяти. Ниже приводится краткое изложение каждой главы.

- Глава 1 посвящена объяснению причины, по которой массивные данные представляют такую трудность для современных систем, и того, как эти трудности влияют на разработку алгоритмов и структур данных.

Часть I: Вероятностные лаконичные структуры данных

- Глава 2 посвящена хешированию и объяснению эволюции хеш-таблиц, чтобы удовлетворять требования со стороны крупных наборов данных и сложных распределенных систем (например, согласованное хеширование). Методы хеширования широко используются в последующих главах, поэтому данная глава служит подготовкой к другим главам части I.
- Глава 3 знакомит с задачей о приближенной принадлежности и двумя структурами данных, которые помогают ее решать: блумовским и порционным фильтрами. В главе представлены примеры использования и анализ частоты ложноположительных результатов, а также плюсы и минусы использования каждой структуры данных.
- Глава 4 посвящена описанию задачи оценивания частоты и вводит набросок count-min, структуру данных, которая решает задачу оценивания частоты чрезвычайно экономичным способом. В ней обсуждаются примеры использования в обработке естественного языка, обработке сенсорных данных и других областях, а также применение наброска count-min к таким задачам, как диапазонные запросы.
- Глава 5 посвящена углубленному изложению алгоритмов оценивания кардинального числа и алгоритмам HyperLogLog, а также их применениям. В этой главе используется мини-эксперимент, чтобы показать эволюцию точности от простого вероятностного подсчета до полной структуры данных HyperLogLog.

Часть II: Структуры и алгоритмы обработки потоковых данных

- Глава 6 представляет собой краткое введение в потоки данных как алгоритмическую концепцию и обработку (приложения по обработке) потоковых данных как проявление реального мира. С помощью нескольких практических примеров использования в архитектуре обработки потоковых данных мы покажем, как структуры данных из предыдущих глав вписываются в контекст потоковых данных.
- В главе 7 объясняется методика поддержания репрезентативной выборки из потока данных или из окна, скользящего над потоком. Мы объясняем ситуации, в которых могут интересовать смещенные выборки, и приводим примеры исходного кода, показывающие реализацию смещения выборки в сторону более недавно наблюдавшихся кортежей данных.
- Глава 8 посвящена вычислению приближенных квантилей на числовых данных из непрерывного потока данных. Мы опишем две конспективные структуры данных, или дайджесты: q-дайджест и t-дайджест. Мы дадим объяснение лежащих в их основе алгоритмов и сравним их друг с другом на реалистичном наборе данных.

Часть III: Структуры данных и алгоритмы внешней памяти

- Глава 9 посвящена введению в модель внешней памяти с несколькими примерами, демонстрирующими, как стоимость операций ввода-вывода преобладает над стоимостью центрального процессора при работе с данными в дистанционном хранилище. Эта глава открывает новые перспективы для разработчика алгоритмов, привыкшего думать об оптимизации алгоритмов с точки зрения стоимости центрального процессора.
- Глава 10 посвящена структурам данных, лежащих в основе магистральных баз данных, – B-деревьям и LSM-деревьям – и охватывает различные компромиссы между операциями чтения и записи при конструировании движка базы данных. Высокоуровневое понимание принципов работы этих структур данных должно помочь вам различать разные стили баз данных и выбирать правильную для вашего приложения.
- Глава 11 посвящена сортировке во внешней памяти и демонстрации оптимальных алгоритмов сортировки файлов на диске с использованием оптимизированных под внешнюю память версий сортировки слиянием и быстрой сортировки. В главе 11 сортировка используется в качестве примера, чтобы продемонстрировать виды оптимизации, которые можно применять для пакетных задач при их перемещении во внешнюю память.

Части I и II связаны друг с другом больше, чем с частью III, поскольку обе они касаются резидентных структур данных и темы максимизации точности при экономии пространства. Часть III имеет самостоятельную тему, и читатель, интересующийся исключительно ею, может пропустить ее вперед, не потеряв контекста. Кроме того, читать часть I перед частью II необязательно, но читатель, который сначала прочтет часть I, будет более подготовлен к пониманию части II, чем тот, кто сразу перейдет к ней.

Части II и III начинаются с главы, в которой объясняется модель и контекст (соответственно главы 6 и 9), и настоятельно рекомендуется прочитать эти главы, чтобы понять другие главы в соответствующих частях. Имея это в виду, не стесняйтесь обследовать книгу самостоятельно. Мы постарались написать все главы настолько самодостаточно, насколько это возможно. При необходимости вы всегда можете вернуться назад и получить более подробную информацию. Рекомендуем всем читателям прочитать главу 1, в которой объясняется причина, по которой массивные данные вызывают такой сдвиг парадигмы в части алгоритмов и структур данных, развертываемых в загруженных работой крупных инфраструктурах.

# Об исходном коде

Несколько глав книги содержат исходный код, и в некоторых более сложных алгоритмах и тех, где контекст значительно его усложнил бы (например, алгоритмах внешней памяти), мы возвращаемся к псевдокоду. В большинстве фрагментов исходного кода и для создания мини-экспериментов, демонстрирующих производительность структур данных в некоторых главах, используются языки Python и R. Читатель не должен испытывать каких-то затруднений в реализации упражнений по программированию на выбранном им языке, поскольку рассматриваемые темы не являются специфичными для какого-либо конкретного языка или технологии.

Эта книга содержит множество примеров исходного кода в виде отдельных нумерованных листингов и внутри обычного текста. В обоих случаях исходный код отформатирован `шрифтом фиксированной ширины, подобным этому`, чтобы отделять его от обычного текста. Иногда исходный код также выделяется **`жирным шрифтом`**, чтобы подчеркивать тот исходный код, который изменился по сравнению с предыдущими шагами в данной главе, например когда новая функциональная возможность добавляется в существующую строку исходного кода.

Во многих случаях изначальный исходный код был переформатирован; мы добавили переносы строк и переработали отступы, чтобы уместиться в доступное пространство страницы книги. В редких случаях даже этого было недостаточно, и листинги включали маркеры продолжения строки (➥). Вдобавок нередко комментарии в исходном коде из листингов удалялись, когда исходный код описывался в тексте. Многие листинги сопровождаются аннотациями к исходному коду, выделяющими важные понятия и концепции.

Вы можете получить исполняемые фрагменты исходного кода из онлайновой версии этой книги в Livebook по адресу **https://livebook.manning.com/book/algorithms-and-data-structures-for-massive-datasets**. Полный исходный код примеров книги доступен для скачивания с веб-сайта издательства Manning по адресу **https://www.manning.com/books/algorithms-and-data-structures-for-massive-datasets**.

# Об авторах

**Джейла Меджедович**, доктор философии, получила докторскую степень в лаборатории прикладных алгоритмов факультета вычислительных наук Университета Стоуни Брук, Нью-Йорк, в 2014 году. Джейла работала над рядом проектов в области алгоритмов обработки массивных данных, преподавала алгоритмы на различных уровнях, а также провела некоторое время в Microsoft. Увлечена преподаванием, продвижением образования в области вычислительных наук и передачей технологий. В настоящее время работает вице-президентом по обработке данных в Social Explorer, Inc.

**Эмин Тахирович**, доктор философии, получил докторскую степень по биостатистике в Пенсильванском университете в 2016 году и степень магистра теоретических вычислительных наук в Университете Гете во Франкфурте в 2008 году. Его статистическая методология и теоретические знания в области вычислительных наук делают его незаурядным исследователем в области науки о данных на стыке вычислительных методов и статистики. Работал в DBahn AG ИТ-консультантом и регулярно консультирует по проектам фармацевтические и технологические компании. Эмин работал доцентом кафедры разработки программного обеспечения в Международном университете Сараево. В настоящее время работает в HAProxy Technologies старшим исследователем данных.

**Доктор Инес Дедович** получила докторскую степень в Институте визуализации и компьютерного зрения на факультете электротехники Университета RWTH в Ахене, Германия. Работала научным сотрудником в исследовательском центре Юлиха и в настоящее время работает разработчиком программного обеспечения для систем видеонаблюдения в компании по автоматизации Jonas & Redmann. Инес более 10 лет также работала 3D-аниматором, художником комиксов и иллюстратором учебников. В этой книге она использует свои художественные и технические навыки для создания интуитивно понятных визуализаций технических концепций.

# Глава 1

# Введение

Эта глава охватывает следующие ниже темы:

- тема книги и ее структура;
- отличие этой книги от других книг по алгоритмам;
- влияние массивных наборов данных на устройство алгоритмов и структур данных;
- как эта книга поможет разрабатывать практические алгоритмы на практике;
- архитектуры компьютеров и систем, усложняющие работу с крупными объемами данных.

Раз вы взяли в руки эту книгу, то, вполне возможно, интересуетесь устройством алгоритмов и структур для массивных наборов данных и хотите разобраться в их отличии от «обычных» алгоритмов, с которыми вы сталкивались до сих пор. Означает ли название этой книги, что классические алгоритмы (например, двоичный поиск, сортировка слиянием, быстрая сортировка, поиск в глубину, поиск в ширину и многие другие фундаментальные алгоритмы), а также канонические структуры данных (например, массивы, матрицы, хеш-таблицы, деревья двоичного поиска, кучи) были созданы исключительно для малых наборов данных?

Ответ на этот вопрос не является ни коротким, ни простым, но если бы нужно было дать короткий и простой ответ, то он был бы «да». Объем понятия массивный набор данных имеет относительный характер и зависит от многих факторов, но суть такова: большинство простых алгоритмов и структур данных, о которых мы знаем и с которыми работаем на ежедневной основе, были разработаны с неявно принятым допущением о том, что все данные умещаются в основной, или оперативной, памяти (ОЗУ) компьютера. И поэтому после загрузки всех данных в оперативную память можно относительно быстро и легко обращаться к любому их элементу, и с этого момента конечной целью, с точки зрения эффективности, становится достижение максимальной производительности за наименьшее число

циклов центрального процессора. Именно этому и учит нас старый добрый анализ «О» большое (O(.)): он обычно выражает число базовых операций, требующихся в наихудшем случае, которые алгоритм должен выполнить, чтобы решить задачу. Указанными единицами работы могут быть сравнения, арифметика, побитовые операции, чтение/запись/копирование ячеек памяти или что-либо еще, что непосредственно транслируется в малое число циклов центрального процессора.

Однако если вы – исследователь данных[1], разработчик или инженер бэкендов, работающий в компании, которая собирает данные у своих пользователей, то хранение всех данных в рабочей памяти вашего компьютера зачастую нереализуемо. Сегодня многие приложения в таких областях, как банковское дело, электронная коммерция, научные приложения и интернет вещей, на рутинной основе манипулируют наборами данных размером в терабайт (Тб) или петабайт (Рб) (то есть вовсе не обязательно трудиться в Facebook или Google, чтобы на работе сталкиваться с массивными данными).

Возможно, вы задаетесь вопросом, а каким большим должен быть набор данных, чтобы можно было воспользоваться методами, показанными в этой книге. Мы намеренно избегаем навешивания ярлыков с указанием величины на так называемые массивные наборы данных или «компании по обработке больших данных», поскольку эта величина зависит от решаемой задачи, доступных инженеру вычислительных ресурсов, требований к системе и т. д. Некоторые компании, имея огромные наборы данных, к тому же располагают значительными ресурсами и могут позволять себе откладывать творческое осмысление проблем масштабируемости, инвестируя в инфраструктуру (например, покупая тонны оперативной памяти). Разработчик, оперирующий на среднекрупных наборах данных, но с ограниченным бюджетом на инфраструктуру и чрезвычайно высокими требованиями к производительности системы со стороны своего клиента, может извлечь выгоду из показанных в этой книге методов не меньше, чем кто-либо другой. Тем не менее, как мы увидим, даже компании с практически бесконечными ресурсами предпочитают заполнять эту дополнительную оперативную память продуманными пространственно-эффективными структурами данных.

Проблема массивных данных существует гораздо дольше, чем социальные сети и интернет. Одна из первых работ [1], в которой были представлены алгоритмы внешней памяти (класс алгоритмов, которые пренебрегают вычислительной стоимостью программы в пользу оптимизации гораздо более времязатратной стоимости передачи данных), появилась еще в 1988 году. В качестве практической мотивации того исследования авторы использовали пример крупных банков, которым приходилось ежедневно сортировать 2 млн чеков, причем чеки объемом около 800 Мб должны были сортироваться за ночь до начала следующего рабочего дня, используя рабочие элементы памяти того времени (~2–4 Мб). Выяснение

[1] Англ. data scientist. – *Прим. перев.*

того, как сортировать все чеки, имея возможность одновременно сортировать чеки объемом всего 4 Мб, и как это делать за наименьшее число обращений к диску, было актуальной задачей того времени, и с тех пор ее актуальность только возросла. С того времени объем данных вырос колоссально, но, что важнее, он рос гораздо быстрее, чем средний объем оперативной памяти.

Главным следствием ускоренного роста объема данных и главной идеей, мотивирующей алгоритмы в этой книге, является то, что сегодня большинство приложений характеризуются интенсивностью использования данных. Интенсивность использования данных (в отличие от интенсивности использования центрального процессора) означает, что узким местом приложения является передача данных туда и обратно и доступ к ним, а не выполнение вычислений с этими данными, после того как они станут доступными. Этот факт является центральным при разработке алгоритмов для крупных наборов данных, и именно отсюда проистекают идеи лаконичных структур данных и алгоритмов, ориентированных на внешнюю память. В разделе 1.4 мы подробнее рассмотрим причины, по которым доступ к данным на компьютере происходит намного медленнее, чем вычисления.

Картина становится еще сложнее, по мере того как мы расширяем поле зрения, переходя от одного-единственного компьютера. Большинство приложений сегодня представляют собой распределенные и сложные конвейеры данных, в которых тысячи компьютеров обмениваются данными по сетям. Базы данных и кеши распределены, и многочисленные пользователи одновременно добавляют и запрашивают крупные объемы контента. Форматы данных стали разнообразными, многомерными и динамичными. В целях поддержания эффективной работы современные приложения должны быть в состоянии очень быстро реагировать на изменения.

В приложениях по обработке потоков [2] данные фактически пролетают незаметно, без какого-либо промежуточного хранения, и приложению приходится улавливать релевантные признаки данных с такой степенью точности, которая делает их актуальными и полезными, без повторного сканирования. Этот новый контекст требует нового поколения алгоритмов и структур данных, нового набора инструментов разработчика приложений, оптимизированного под решение многих задач, характерных для систем массивных данных. Настоящая книга предназначена научить вас именно этому – основополагающим алгоритмическим методам и структурам данных для разработки масштабируемых приложений.

# 1.1 Пример

В целях иллюстрации главных тем этой книги давайте рассмотрим следующий пример: вы работаете в медиакомпании над проектом, связанным с комментариями к новостным статьям. Вам предоставили крупное хранилище комментариев со следующими базовыми метаданными:

```
{
    comment-id: 2833908010
    article-id: 779284
    user-id: 9153647
    text: для этого рецепта нужно больше сливочного масла
    views: 14375
    likes: 43
}
```

Вы осматриваете примерно 3 млрд пользовательских комментариев общим объемом 600 Гб. Вы хотели бы получить ряд ответов на вопросы о наборе данных, включая определение наиболее популярных комментариев и статей, классификацию статей в соответствии с темами и распространенными ключевыми словами, встречающимися в комментариях, и т. д. Но сначала необходимо решить задачу о дубликатах, которые накапливались в течение нескольких операций выскабливания данных из интернета, и определить общее число несовпадающих комментариев в наборе данных.

### 1.1.1 Решение задачи: пример

Для хранения уникальных элементов в структуре данных широко принято создавать словарь ключ-значение, в котором уникальный ИД каждого несовпадающего элемента соотносится с частотой его появления. Словари ключ-значение реализованы во многих библиотеках, таких как `map` в C++, `HashMap` в Java, `dict` в Python и т. д. Словари ключ-значение обычно реализуются в виде сбалансированного дерева двоичного поиска (например, красно-черного дерева в `map` C++) либо, как альтернативный вариант, в виде хеш-таблиц (например, `dict` в Python).

> **Реализации красно-черного дерева и хеш-таблицы в сравнении**
>
> Реализации древовидного словаря, помимо операций поиска/вставки/удаления, которые выполняются за быстрое логарифмическое время, предлагают столь же быстрые операции предшественник/преемник, то есть возможность эффективно просматривать данные взад и вперед, используя лексикографическое упорядочение. Большинству реализаций хеш-таблиц не хватает возможности эффективно выполнять обход элементов в лексикографическом порядке; с другой стороны, реализации хеш-таблиц обеспечивают высокую постоянно-временную производительность наиболее распространенных операций поиска/вставки/удаления.

Ради упрощения нашего примера давайте допустим, что мы работаем с хеш-таблицей Python `dict`. Использование атрибута `comment-id` в качестве ключа и числа появлений этого атрибута в качестве значения поможет нам эффективно устранить дубликаты (см. словарь `(comment-id -> frequency)` в левой части рис. 1.1).

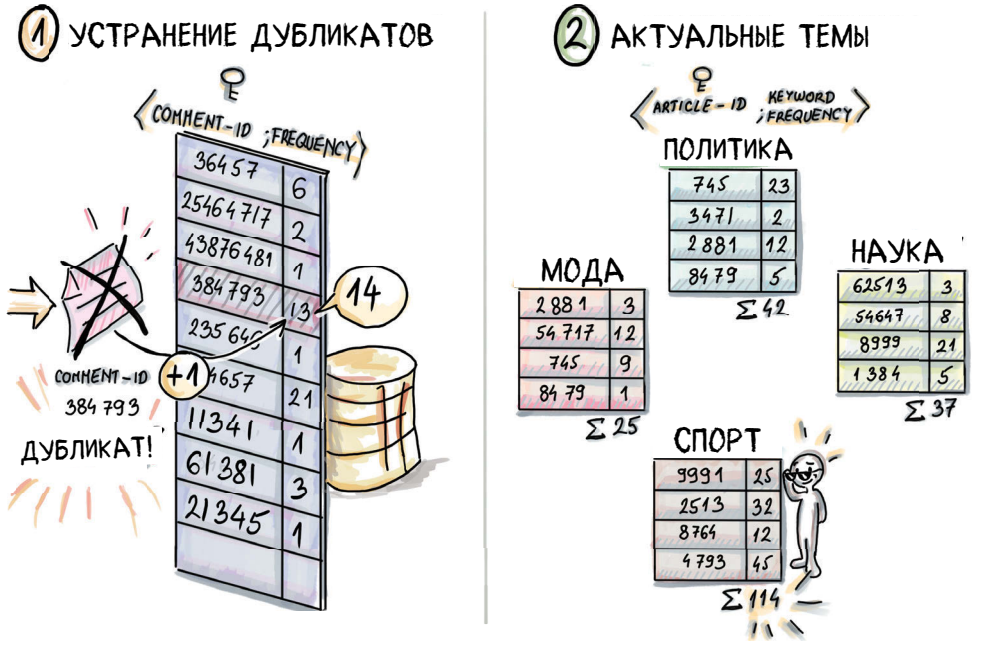


**Рисунок 1.1** В данном примере создается хеш-таблица (`comment-id`, `frequency`), которая помогает хранить несовпадающие идентификаторы `comment-id` с их частотами. Входящий идентификатор `comment-id 384793` в таблице уже содержится, и его частота возрастает. Помимо указанной хеш-таблицы, создаются хеш-таблицы, связанные с темами, в которых подсчитывается число раз, когда ассоциированные ключевые слова появлялись в комментариях к каждой статье (например, в спортивной теме ключевыми словами могут быть футбол, игрок, гол и т. д.). В крупном наборе данных в 3 млрд комментариев для таких структур данных может потребоваться от десятков до ста гигабайт оперативной памяти

Однако если использовать по 8 байт в расчете на пару (4 байта для `comment-id` и 4 байта для `frequency`), то для хранения пар `<comment-id, frequency>` нам может потребоваться до 24 Гб из 3 млрд комментариев. В зависимости от метода, используемого в реализации базовой хеш-таблицы, для служебных операций (с пустыми слотами, указателями и т. д.) структуре данных потребуется в 1.5–2 раза больше занимаемого элементами места, что приблизит нас почти к 40 Гб.

Если мы также хотим классифицировать статьи по определенным интересующим темам, то можем снова использовать словари (возможны и другие методы), создав отдельный словарь для каждой темы (например, спортивной, политической, научной и т. д.), как показано в правой части рис. 1.1. Здесь роль словарей (`article-id -> keyword_frequency`) заключается в подсчете числа появлений ключевых слов, связанных с той или иной темой, во всех комментариях; например, статья с `article-id` 745 содержит 23 ключевых слова в соответствующих комментариях, связанных с политикой. Мы предварительно фильтруем каждый `comment-id`, используя большой

словарь (`comment-id -> frequency`), чтобы учитывать только несовпадающие комментарии. Отдельная таблица такого рода может содержать десятки миллионов записей общим объемом около 1 Гб, и поддержание таких хеш-таблиц, скажем, для 30 тем может стоить до 30 Гб только для данных и примерно 50 Гб в общей сложности.

Будем надеяться, что приведенный выше небольшой пример наглядно демонстрирует, как можно начать с довольно распространенной и наивной задачи и, не успев опомниться, столкнуться с рядом неуклюжих структур данных, которые просто невозможно уместить в памяти.

Возможно, вы подумали: а разве нельзя заранее умножить пару чисел и легко предсказать будущую величину структуры данных? Дело в том, что в реальной жизни это зачастую работает не так. Люди редко начинают строить свои системы с нуля, уже имея в наличии массивные данные. Компании часто начинают с попыток создать работающую систему, а позже становятся жертвами собственного успеха, когда пользовательская база ускоренно вырастает за короткий промежуток времени и старая система, построенная покинувшими компанию разработчиками, должна справляться с этой новой взыскательной рабочей нагрузкой. Чаще всего части системы перестраиваются по мере возникновения необходимости.

Когда число элементов в наборе данных становится большим, каждый дополнительный бит в расчете на элемент увеличивает нагрузку на систему. Распространенные структуры данных, являющиеся хлебом насущным для каждого разработчика программного обеспечения, могут стать слишком большими, чтобы с ними можно было работать эффективно, и нам нужны более лаконичные альтернативы (см. рис. 1.2).

**Рисунок 1.2** Наиболее распространенные структуры данных, включая хеш-таблицы, становится трудно хранить и контролировать, имея крупные объемы данных