4. ХРАНЕНИЕ ВРЕМЕННЫХ РЯДОВ

Очень часто ценность данных временного ряда возрастает с увеличением срока их давности — данные, получаемые в режиме реального времени, оказываются менее полезными, чем собранные много лет назад. Чтобы иметь возможность обрабатывать временные ряды в будущем, нужно побеспокоиться об их хранении.

Правильное хранилище данных характеризуется высокой надежностью и простотой доступа и не требует значительных инвестиций в вычислительное оборудование.

В это разделе мы обсудим характеристики наборов данных, на которые следует обращать наиболее пристальное внимание при передаче временных рядов на хранение. Мы также обсудим преимущества реляционных и нереляционных баз данных и некоторых других файловых форматов.

Разработка универсального решения, отвечающего за хранение временных рядов, является сложной задачей, поскольку они бывают самых разных типов, каждый из которых имеет свой формат хранения, чтения/записи и обработки данных. Заметьте, что одни данные сохраняются и обрабатываются с заданной периодичностью, тогда как другие востребованы в течение предельно короткого периода времени, после чего их можно безбоязненно удалить.

Перечислим наиболее распространенные сценарии использования хранилищ временных рядов с разными форматами запроса, чтения и записи данных.

1. Временной ряд используется для хранения данных о производительности промышленного оборудования. Сведения о производительности необходимо хранить годами, но чем старше данные, тем меньше деталей они должны включать. Следовательно, вам нужен формат хранения, в котором данные автоматически урезаются и сокращаются по мере устаревания информации.

2. Имея доступ к удаленному хранилищу временных рядов с открытым исходным кодом, вам удобнее работать с локальной копией данных в силу существующих ограничений на доступ к сетевому окружению. В удаленном репозитории временные ряды хранятся в виде отдельных файлов, доступных для загрузки с веб-сервера, но для большего удобства их лучше свести в единую базу данных. В конечной базе данных информация должна представляться в неизменном виде и храниться неограниченный период времени, так как она представляет резервную копию удаленного репозитория.

3. Вы создали собственный временной ряд данных, собранных из многих источников с разными временными масштабами, подвергая их предварительной обработке и форматированию. Сбор и обработка заняли много времени и сил. Вы хотели бы сохранить данные в их окончательном формате, чтобы избежать повторной предварительной обработки. Наряду с этим не стоит пренебрегать сохранением копии необработанных данных на случай возникновения необходимости в их анализе в исходном виде. Вы ожидаете частого пересмотра обработанных и необработанных данных вследствие разработки новых алгоритмов машинного обучения, усовершенствования старых и создания новых моделей, а также пополнения наборов новыми свежими данными. Вы не планируете сокращать или очищать данные в хранилище, даже если это требуется в конечных задачах их анализа.

Как видите, рассмотренные выше сценарии сильно различаются требованиями к системе.

*Масштабирование и производительность*

В первом сценарии требуется решение, включающее сценарии автоматического удаления старых данных. Здесь не стоит беспокоиться о масштабировании системы, как в случае больших данных, поскольку в хранилище помещаются только небольшие наборы данных. Наряду с этим в остальных сценариях ожидается, что хранилище будет заполняться исключительно большими наборами данных: неизменных (сценарий 2) или регулярно пополняемых (сценарий 3).

*Случайный и последовательный доступ к данным*

Во втором сценарии предполагается одинаковая частота обращения ко всем значениям набора, так как данные всего временного ряда имеют одинаковый “возраст” хранения и представляют одну и ту же величину. Напротив, в остальных сценариях (особенно в первом, учитывая предыдущее замечание) ожидается, что последние данные будут востребованы чаще более ранних данных.

*Сценарии автоматизации*

Первый сценарий просто-таки напрашивается на использование средств автоматизации, тогда как во втором сценарии от автоматизации можно полностью отказаться (данные остаются неизменными). Третий сценарий предполагает наименьшую автоматизацию, но наибольший уровень выборки и обработки всех, а не только последних, данных. В первом сценарии нужно использовать решение для хранения данных, поддерживающее работу со сценариями автоматизации и хранимыми процедурами, тогда как в третьем сценарии оно должно обеспечивать свободную перенастройку инструментов обработки данных. Мы рассмотрели всего три ограниченных сценария, но даже они дают представление о многих аспектах разработки универсальных решений для хранения данных временных рядов.

В реальных задачах хранения данных вы сможете адаптировать имеющиеся готовые решения и не заниматься поиском инструментов, удовлетворяющих требованиям всех возможных сценариев. Как бы там ни было, вам придется выбирать из небольшого числа форматов, которые можно представить следующим списком возможностей.

• Реляционные базы данных (SQL).

• Нереляционные базы данных (NoSQL).

• Файлы обычных форматов.

В этом разделе мы обсудим все три возможности, а также рассмотрим преимущества и недостатки каждой из них. Конечно, формат, в котором будут храниться данные, в наибольшей степени зависит от избранного сценария хранения временных рядов, и в этом разделе вы узнаете об основных аспектах, на которые нужно обращать внимание при его выборе.

Сначала мы обсудим требования к хранилищам данных, на которые нужно обращать самое пристальное внимание, а затем перейдем к сравнению реляционных и нереляционных баз данных и рассмотрению наиболее популярных решений по хранению временных рядов. В конце разделамы займемся настройкой политик, отвечающих за умышленное устаревание и удаление более старых данных временных рядов.

*Определение требований*

Определяясь с технологиями хранения временных рядов, постарайтесь ответить на несколько вопросов.

• Как много данных временного ряда подлежит хранению? Насколько быстро увеличивается их объем? Решение для хранения нужно выбирать, исходя из скорости увеличения объема хранимых данных. Администраторы баз данных, переходящие к работе с временными рядами, содержащими данные о финансовых транзакциях, искренне удивляются тому, насколько сильно они увеличиваются в размере со временем.

• Исследованию подлежит непрерывный поток восполняемых данных (например, сетевой трафик) или дискретный набор значений (например, почасовые данные о воздушных перевозках в дни государственных праздников США за последние 10 лет)? Если временные ряды собираются из непрерывного потока, то анализу, скорее всего, будут подлежать только последние данные. С другой стороны, если анализируемые данные представлены конечными временными рядами, то более отдаленные во времени значения могут представлять даже больший интерес, чем последние. В последнем варианте наиболее предпочтительным будет случайный порядок доступа к данным.

• Данные собираются через регулярные или нерегулярные временные интервалы? При равномерном сборе данных можно заранее определить общее количество временных точек и частоту выборки. Если же данные собираются через нерегулярные временные интервалы, то будьте готовы к непредсказуемому порядку выборки данных, который характеризуется периодами простоя и сбора разной длительности.

• Сбор данных осуществляется в течение бесконечного периода времени или у исследовательского проекта имеется строго оговоренная дата окончания? В случае существования конечной даты сбора данных всегда можно рассчитать общий объем хранимых данных. Как это ни странно, в этом сценарии самым трудным оказывается прекратить сбор данных в строго оговоренную дату. Нередки ситуации, когда временные ряды определенного типа продолжают собираться даже после завершения всех необходимых исследований!

• Каково назначение временных рядов? Нужно ли визуализировать их в реальном времени? Будут ли данные подвергаться предварительной обработке нейронной сетью в несколько тысяч проходов? Доступны ли сегментированные данные широкому кругу мобильных пользователей? Тип доступа к данным (последовательный или свободный), их формат и время ожидания в полной мере определяются избранным сценарием хранения.

• Каким образом отбраковываются данные и понижается частота их выборки? Как предотвращается их бесконечный рост? Каким должен быть жизненный цикл отдельной точки данных во временном ряду? Сохранить данные всех событий за весь бесконечный период времени просто невозможно. Именно поэтому лучше заранее выработать принципы автоматического удаления больше ненужных данных, чем заниматься их очисткой вручную. Чем эффективнее будет ваше решение, тем проще вам будет определиться с форматом хранения данных. Подробнее об этом рассказано в следующем разделе.

Ответив на такие вопросы, вы сможете определиться со многими важными аспектами хранения — данные будут записываться в необработанном или в обработанном виде, располагаться в памяти в соответствии со временем сбора или с некой другой характеристикой и нужно ли их представлять в специальном формате, упрощающем чтение и запись. Сценарии хранения могут изменяться в процессе принятия решений, поэтому не ленитесь их пересматривать для каждого нового набора данных.

Сегментированными называются данные, являющиеся частью общей структуры данных, которая разделяется на мелкие, более удобные в управлении части, зачастую хранящиеся на разных серверах.

***Оперативные и хранимые данные***

Выбирая оптимальный способ хранения, не забывайте о жизненном цикле данных. Чем реалистичнее вы подойдете к оценке фактического сценария, тем меньший объем данных вам придется хранить и тем быстрее определитесь с оптимальной системой хранения, поскольку вряд ли сможете превысить зарезервированный объем в предельно короткие сроки. Очень часто исследователи, боясь потерять важные данные, записывают временные ряды повторно, что совершенно неоправданно: иметь большой объем неразрешимых данных — не намного лучше, чем заниматься сбором данных в течение неприлично большого временного периода.

Краткосрочные временные ряды, например с данными о производительности

системы, предназначенные только для ознакомления, обычно не подлежат хранению,

по крайней мере в течение долгого периода времени. Такой подход в наибольшей

степени проявляется при использовании данных, управляемых событиями,

когда интерес представляют не отдельные события, а общие статистические

показатели процесса.

Представьте, что с помощью специального серверного программного обеспечения

собираются и сохраняются сведения о количестве времени, затрачиваемого

каждым мобильным устройством для полной загрузки веб-страницы.

В качестве результата вы получите неравномерные временные ряды, выглядящие

примерно так, как показано в табл. 4.1.

Таблица 4.1. Временные ряды, собираемые веб-сервером

Временная метка Время загрузки страницы, с

5 апреля, 2018 22:22:24 23

5 апреля, 2018 22:22:28 15

5 апреля, 2018 22:22:41 14

5 апреля, 2018 22:23:02 11

Сами по себе временные метки событий загрузки страницы не представляют ценности для анализа сразу по многим причинам. Поэтому подвергать анализу лучше всего средние значения величин (например, среднее время загрузки страницы за минуту). Но результаты статистических исследований усредненных показателей будут актуальны только непродолжительный период времени. Предположим, что требуется изучить данные, собранные сервером в течение ночи, чтобы убедиться в его высокой производительности. Упростите исходный временной ряд

до единственной точки данных, в которой будет указываться усредненное за все

время наблюдений (12 часов) значение показателя. Полученный результат несет

больше полезной информации, чем исходно собранные данные (табл. 5.2).

Таблица 5.2. Упрощенные точки данных для периодов наблюдений

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Временной период | Наиболее популярное время доступа | Количество загрузок | Среднее время загрузки, с | Максимальное время загрузки, с |
| 5 апреля 2018 года 20:00-8:00 | 23:00 | 3470 | 21 | 45 |

Как видите, в таком сценарии можно отказаться от долговременного хранения данных всех регистрируемых событий. Вместо этого используйте для хранения такое решение, которое обеспечивает их сохранность в течение небольшого периода времени — до представления в окончательном, наиболее информативном формате. Тем самым вы избежите неоправданно быстрого роста объема данных, помещаемых на хранение. Вместо 3470 отдельных событий, которые по отдельности не представляют особого интереса, долговременному хранению будут подвергаться компактные данные с заведомо важной информацией. Старайтесь всегда, когда только возможно, сокращать объем хранящихся данных за счет усреднения и исключения из временного ряда дублирующих значений.

Далее мы рассмотрим несколько возможностей по сокращению данных без потери важной информации.

*Медленно изменяющиеся переменные*

При хранении переменных состояния рассмотрите возможность записи только тех точек данных, в которых они претерпевают изменения. Например, если вы снимаете показания термометра через каждые пять минут, то кривая изменения температуры будет выглядеть как пошаговая функция, особенно если вы округляете измеренные значения до ближайшего градуса. В таком случае можно отказаться от хранения повторяющихся значений, сэкономив место для более значимых данных.

*Зашумленные, высокочастотные данные*

Если данные зашумлены, то нет никаких оснований дорожить каждой отдельно рассматриваемой точкой данных. Рассмотрите возможность частичного объединения точек данных перед записью в хранилище — высокий уровень шума делает любое отдельное наблюдение малоинформативным. Конечно, все зависит от рассматриваемой задачи, и вам нужно убедиться в том, что при необходимости исходный шум, присутствующий в данных, все еще можно оценить.

*Устаревшие данные*

Чем старше данные, тем меньше вероятность того, что они будут привлекаться к анализу временных рядов, разве что в самом общем виде. Всякий раз, собираясь сохранять новый набор временных рядов, попытайтесь понять, в какой момент времени они станут неактуальными.

• Существует ли у данных естественный срок годности?

• Если нет, то по возможности ознакомьтесь с результатами исследований аналитического отдела и посмотрите, насколько давние данные принимаются в них в расчет? Например, узнайте, когда в последний раз сценарий запрашивал самые старые значения временных рядов из репозитория GitHub.

Автоматизировав операцию удаления данных без нарушения процедуры анализа временных рядов, вы уменьшите сложность их хранения и дальнейшей обработки, нивелировав проблему масштабируемости или медленно обрабатываемых запросов к непомерно раздутым наборам данных.

*Правовые аспекты*

Если вы работаете в крупной организации, которая занимается обработкой больших массивов данных, предоставляемых сторонним организациям (например, регулирующим органам), то при выборе средств хранения обязательно учитывайте юридические аспекты решаемой задачи. Постарайтесь оценить, как часто будут обрабатываться запросы на получение данных, какой объем данных передается при их выполнении и регулирует ли нормативная документация формат хранения и передачи данных.

Изучение правовых аспектов при выборе средств хранения данных может показаться излишней мерой предосторожности, но последние исследования, проводимые специалистами по обработке данных в Европе и США, свидетельствуют о том, что наибольший интерес у аналитиков вызывают наборы данных, доступные для использования в машинном обучении (подпадающие под действие норм общего регламента по защите данных Европейского союза).

До настоящего момента мы рассматривали только общие требования к решениям, предназначенным для хранения временных рядов. Вы также познакомились с особенностями получения и анализа временных рядов как основного фактора определения формата хранения данных. Теперь можно переходить к рассмотрению двух наиболее распространенных вариантов хранения временных рядов: в базах данных и в файлах.

*Хранение временных рядов в базах данных*

База данных является интуитивно понятным и знакомым инструментом хранения данных практически для любого специалиста по анализу и обработке информации. При этом базы данных прекрасно подходят для хранения не только реляционных данных, но и временных рядов. В наибольшей степени они подходят для решения задач, в которых к системам хранения данных выдвигаются следующие, давно ставшие классическими, требования.

• Масштабирование данных на нескольких серверах.

• Выполнение операций чтения/записи с низкой задержкой.

• Поддержка функции вычисления часто определяемых статистических показателей (например, среднего значения в запросах группировки, где операция группировки может применяться в том числе к временным величинам).

• Снабжение инструментами сопровождения и устранения неполадок, которые можно использовать для настройки производительности системы и анализа узких мест.

Из всех возможных вариантов хранения временных рядов (https://perma.сс/K994-RXE9) существуют веские причины отказаться от использования отдельных файлов в пользу баз данных, особенно при работе с новыми наборами данных. В базах данных, особенно в нереляционных, данные находятся в гибком состоянии. Наличие строгой структурной организации является еще одним преимуществом баз данных: можно существенно ускорить запуск проектов с данными временных рядов по сравнению с решениями, основанными на отдельных файлах. Но даже если вы предпочитаете хранить данные временных рядов в файлах (в отличие от большинства исследователей), база данных поможет вам определиться со структурой файлов, количество которых неизбежно увеличивается по мере усложнения проекта и пополнения временных рядов новыми данными.

В оставшейся части этого раздела мы рассмотрим преимущества реляционных баз данных перед NoSQL-решениями, а затем обсудим самые популярные в настоящее время типы баз данных, предназначенные для хранения временных рядов.

Хорошая новость заключается в стремительном развитии графовых СУБД (https: //perma.cc/RQ79-5AX7), в настоящее время являющихся наиболее перспективной технологией управления базами данных, и есть все основания полагать, что в ближайшем будущем появится несколько достойных внимания решений.

*Реляционные и нереляционные базы данных*

Технологии SQL и NoSQL привлекают не менее пристальное внимание, чем базы данных временных рядов. Многие опытные администраторы баз данных настаивают на том, что SQL — единственно верная технология управления базами данных и что не существует данных, которые нельзя описать правильно подобранным набором реляционных таблиц. Тем не менее при попытках масштабирования SQL-решений при хранении больших временных рядов часто наблюдается падение производительности, что подталкивает к переходу на нереляционные технологии управления базами данных, особенно при поиске открытого решения, обеспечивающего свободное масштабирование данных при сборе временных рядов для бесконечного временного горизонта.

При работе с временными рядами могут применяться как реляционные, так и нереляционные технологии управления базами данных, но оба случая сопряжены с целым рядом трудностей при управлении базами данных временных рядов.

Для правильного их применения нужно понять, чем временные ряды отличаются

от других типов данных, для управления которыми изначально разрабатывались

СУБД указанных типов.

*Данные, с которыми работают СУБД*

Для лучшего понимания принципов несоответствия реляционных баз данных потребностям в хранении временных рядов достаточно изучить историю разработки решений, основанных на технологии SQL. Исходно SQL-решения предназначались для обработки транзакционных данных — достаточных для полного описания дискретного события. В транзакции обрабатываются данные атрибутов, которые отражены во многих первичных ключах, таких как назначение, имя клиента, дата исполнения и стоимость транзакции. Обратите внимание, что время также может выступать первичным ключом, указывая на одно из многих, но не привилегированное направление измерения информации.

Существуют две важные особенности транзакционных данных, которые отличают их от данных временных рядов.

• Точки данных часто обновляются.

• Доступ к данным осуществляется случайным образом, так как порядок их обработки никак не регламентируется.

*Характеристики временных рядов*

Если данные временного ряда описывают историю некого процесса, то транзакция указывает только конечное состояние системы. Следовательно, данные временных рядов обычно не требуют обновления, что делает операции записи данных со случайным доступом крайне маловероятными.

Таким образом, требования к производительности инструментов, которые были ключевыми на протяжении многих десятилетий разработки СУБД, оказываются несущественными при работе с временными рядами. Фактически, выдвигая требования к системам хранения временных рядов, мы преследуем несколько иные цели, поскольку управление такими хранилищами построено на принципах, отличных от принятых в реляционных базах данных. Для лучшего понимания обозначим ключевые требования к средствам хранения временных рядов.

• Операции записи преобладают над операциями чтения.

• Данные записываются, считываются и обновляются не случайным образом, а во временном порядке.

• Одновременное считывание данных выполняется намного чаще, чем в случае проведения транзакций.

• Кроме времени, допускается использовать (если они заданы) всего несколько первичных ключей.

• Массовое удаление данных выполняется чаще, чем удаление отдельных точек данных.

Большинство из этих функций поддерживается NoSQL-решениями — многие нереляционные базы данных общего назначения предлагают многое из того, что требуется реализовать в хранилище временных рядов — в первую очередь, из того, что касается превалирования операций записи над операциями чтения.

Концептуально технологии NoSQL хорошо согласуются с требованиями, которые выдвигаются к средствам хранения временных рядов, изначально отражая многие специальные возможности, например заполнения полей данных не для всех точек данных. Гибкость структуры NoSQL оказывается естественной для временных рядов. По правде говоря, стремительно растущий интерес к нереляционным технологиям в большой степени вызван потребностью в системах управления и хранения временных рядов.

По этой же причине готовые NoSQL-решения показывают лучшие результаты в операциях записи, чем реляционные СУБД. На рис. 4.1 приведены графики сравнения производительностей операций добавления (записи) новых точек временных рядов в стандартные реляционную базу данных и базу данных NoSQL.

*Трудности выбора*

Может показаться, что я пытаюсь подтолкнуть вас к варианту использования нереляционных решений, но у реляционных баз данных тоже есть важные преимущества. Изучая имеющиеся данные, помните о главном аспекте, который нужно учитывать как в реляционных базах данных и NoSQL-решениях, так и при хранении временных рядов в старом добром текстовом файле: одновременно запрашиваемые данные должны храниться одном и том же месте. Это самый важный фактор, независимо от предпочитаемого решения.

Рис. 4.1. Главная особенность базы данных временных рядов состоит в неизменности скорости добавления новых точек данных при увеличении ее общего размера. Это важнейшее преимущество перед традиционными реляционными СУБД

На многих тематических конференциях и в блогах NoSQL-решения описываются как хранилища данных, в полной мере отвечающие требованиям высокой скорости записи, низкой частоты обновления и последовательной организации данных. Наряду с этим реляционные базы данных также относятся к оправданным и часто востребованным решениям для хранения временных рядов. В частности, внеся незначительные архитектурные изменения в традиционные базы данных и внутреннюю структуру памяти, можно преодолеть большинство недостатков с сохранением большинства преимуществ реляционных СУБД (https://perma.cc/3ZUQ-B5WC). Например, такое простое и, казалось бы, очевидное решение, как структурирование внутренней памяти реляционных таблиц, обеспечивающее учет времени, позволяет разительно увеличить общую производительность системы.

В конечном счете преимущества и недостатки как реляционных, так и нереляционных баз данных сильно зависят от их реализации и не настолько концептуальны и важны, как предполагают технологии, лежащие в их основе. Допустим, что выбор конкретной реализации одной из технологий определяют исследуемые данные. Рассматривая атрибуты временных рядов и возможные сценарии реализации, помните о следующих важных ограничениях.

Преимущества хранения временных рядов в реляционных базах данных

• При хранении в реляционной базе данных временной ряд можно легко связать с соответствующими перекрестными данными этой же базы данных.

• Иерархические данные временных рядов естественным образом встраиваются в реляционные таблицы. Образованная структура позволяет предельно точно группировать связанные временные ряды и четко обозначать иерархические отношения, в то время как в NoSQL-решении эта задача представляется намного сложнее и требует более системного подхода в выполнении.

• При создании временных рядов на основе транзакционных данных, хранящихся в реляционной базе данных, самым оправданным решением будет сохранять временные ряды в этой же базе данных, упрощая их дальнейшую проверку, наполнение перекрестными связями и т.д.

Преимущества хранения временных рядов в нереляционных базах данных

• Высокая скорость записи.

• Прекрасно подходят для разработки функциональных и высокопроизводительных решений в системах, в которых мало что известно о будущих данных.

• Начинающим специалистам по анализу данных нереляционные базы данных кажутся более производительным, готовым к использованию решением, не способным к образованию малопригодной производственной схемы.

*Популярные базы данных временных рядов и файловые решения*

Настало время обсудить наиболее популярные решения для хранения временных рядов. Вам предстоит узнать о возможностях, которые предоставляются в дополнение к традиционным базам данных. Обратите внимание, что обсуждаемые далее решения относятся к нерегулярно задействуемым технологиям. Те из них, которые пользуются популярностью в этом году, могут оказаться маловостребованными в следующем году. Таким образом, дальнейшие рассуждения следует воспринимать не как строгие технологические рекомендации, а скорее как набор советов, помогающих ориентироваться в текущем состоянии рынка программных решений, предназначенных для хранения данных.

*Базы данных для хранения временных рядов и инструменты мониторинга*

Сначала давайте познакомимся со специальными инструментами хранения и мониторинга временных рядов. В частности, рассмотрим базу временных рядов InfluxDB и специальный инструмент, отвечающий за мониторинг производительности системы, который также может использоваться как решение для хранения временных рядов, — Prometheus. Преимущества каждого инструмента в полной мере отражают их назначение и способ использования.

InfluxDB. Согласно описанию проекта на GitHub база данных InfluxDB исходно ориентирована на хранение временных рядов.

InfluxDB — это база данных временных рядов с открытым исходным кодом. Ее лучше всего использовать для сбора статистических показателей, регистрации событий и результатов аналитических исследований.

В InfluxDB данные представлены временными рядами. Точка данных в InfluxDB состоит из следующих компонентов.

• Временная метка.

• Подпись, описывающая измеряемую величину.

• Одно или несколько полей “ключ/значение” (например, temperature=25.3).

• Пары “ключ/значение”, содержащие теги метаданных.

Как учитывающая время база данных InfluxDB автоматически расставляет временные метки для всех точек данных, лишенных их. Кроме того, в InfluxDB используются SQL-подобные запросы, например такие.

SELECT \* FROM access\_counts WHERE значение> 10000

База данных InfluxDB характеризуется следующими преимуществами.

• Поддержка инструментов хранения данных, которые позволяют легко автоматизировать

операции назначения и удаления устаревших данных.

• Высокая скорость получения и высокая степень сжатия данных.

• Возможность добавления меток к временным рядам, которые соответствуют определенному критерию, для быстрой индексации.

• Полноправное подключение к TICK (https://www.influxdata.com/time-series-platform/) — платформе для сбора, хранения, мониторинга и отображения временных рядов.

Существует много других баз данных, ориентированных на хранение временных рядов. На данный момент InfluxDB является наиболее популярной из них, поэтому с ней вы, скорее всего, и столкнетесь в первую очередь. Ее функции стандартны для специализированных баз своего типа и отражают наиболее востребованные возможности по хранению временных рядов.

Как база данных InfluxDB основана на технологии "проталкивания" (push-технология) — в базу данных заносятся данные, предложенные ей. Такой подход заметно отличается от представленного далее в описании базы данных Prometheus.

С учетом рассмотренных спецификаций InfluxDB является полнофункциональным решением, основанным на общепринятых принципах. Это означает, что при ее использовании вы получаете в свое распоряжение как стандартные SQL-инструменты, так и специализированные функции, отвечающие за сбор и контроль размера временных рядов. Кроме того, InfluxDB включает команды быстрой записи данных, часто востребованные при сборе данных временных рядов непосредственно в процессе их создания.

***Prometheus.*** Программное обеспечение Prometheus (https://github.com/prometheus/prometheus) описывается как “система мониторинга и база данных временных рядов”, работающая по протоколу HTTP. В описании недвусмысленно указывается его основное функциональное назначение: сначала — мониторинг и только потом — хранение данных. Главная особенность Prometheus заключается в том, что она основана на pull-технологии, — параметры извлечения данных временного ряда, а также сериализации легко поддаются проверке и настройке.

Система Prometheus — это ценный инструмент на случай непредвиденных ситуаций благодаря атомарной и самодостаточной структуре. Однако поддержка технологии pull не предполагает обновление данных с абсолютной точностью и своевременностью. А так как Prometheus, в первую очередь, предназначена для оперативного мониторинга производительности, она точно не подойдет для систем, в которых первостепенную важность имеет 100-процентная точность данных. Запросы в Prometheus пишутся на языке функциональных выражений PromQL.

access\_counts> 10000

Кроме того, система Prometheus снабжена программным интерфейсом PromQL, включающим команды выполнения многих распространенных задач обработки временных рядов, в том числе такие сложные, как прогнозирование (predict linear ()) и вычисление скорости роста временного ряда за единицу времени (rate()). А еще он обеспечивает агрегацию показателей по периодам времени. В Prometheus акцент сделан на мониторинге и анализе временных рядов, а не на обслуживании базы данных, поэтому по сравнению с InfluxDB в нем намного меньше команд автоматической обработки данных.

Систему Prometheus можно смело использовать для хранения временных рядов, особенно в потоковых приложениях и в случаях, когда превыше всего ценится доступность данных. Она не самая простая для обучения благодаря использованию специального языка написания сценариев, а также необычной архитектуре и не характерному для базы данных программному интерфейсу. Тем не менее она широко используется и нравится многим разработчикам.

*Нереляционные базы данных*

В то время как базы данных, специально предназначенные для временных рядов, имеют много преимуществ, для их хранения можно попробовать использовать нереляционные базы данных общего назначения. Структура таких баз данных основана на документах, а не таблицах, а сами они обычно включают много явных функций обработки временных рядов данных.

Несмотря на это гибкость состояния нереляционных баз данных прекрасно сочетается с обработкой временных рядов, особенно в новых проектах, в которых скорость сбора данных и количество входных каналов могут изменяться в течение всего срока хранения набора данных. Например, временной ряд может изначально включать всего один канал данных, но постепенно расширяться за счет включения большего количества типов данных, снабженных временной разметкой. В дальнейшем, ввиду бесполезности, некоторые из входных каналов всегда можно отключить.

Сохранение подобных данных в реляционной таблице представляет собой трудно решаемую задачу сразу по нескольким причинам и приводит к включению большого количества значений NaN, указывающих на недоступность данных.

В нереляционной базе данных, чтобы избежать загромождения хранилища данных множеством значений NaN, достаточно отключить соответствующие каналы.

В качестве примера популярной и производительной нереляционной базы данных можно привести MongoDB. Превыше всего она ценится именно как база данных временных рядов, а также за активную поддержку 1оТ-технологий на архитектурном и программном уровнях. Она включает функции агрегации высокого уровня, которые обеспечивают агрегацию информации по времени и группировку данных по временным признакам, а также предлагает множество инструментов автоматического разделения данных по принятым в человеческом мире временным отрезкам — дням недели или месяца.

• $dayOfWeek

• $dayOfMonth

• $hour

Кроме того, разработчики системы MongoDB приложили значительные усилия для документирования ее функций обработки временных рядов (“Schema Design for Time Series Data in MongoDB” (https: //perma. cc/E8XL-R3SY), “Time Series Data and MongoDB: Part 1 — An Introduction’(https://perma.cc/A2D5-HDFB) и “Time Series Data and MongoDB: Part 2 — Schema Design Best Practices” (https : / /perma. cc/2LU7-YDHX). Противоположная точка зрения изложена в статье “How to store time-series data in MongoDB, and why that is a bad idea” (https : //perma . cc/T4KM-Z2B4)). Полный набор документации и акцент на временных рядах однозначно указывают на то, что пользователям стоит ожидать появления более удобных функций управления временными зависимостями.

Однако наиболее важной из всех функциональных особенностей MongoDB является гибкость изменения схемы со временем. Она позволяет избежать многих неприятностей при работе с быстро эволюционирующими наборами временных рядов, характеризующихся нерегулярным способом сбора данных.

Например, проследим за тем, как могут изменяться способы сбора временных рядов при запуске одного из возможных стартапов в области здравоохранения.

1. Стартап начинается с выпуска приложения для измерения артериального давления и обеспечивает сбор всего двух показателей: систолического и диастолического артериального давлений, которые рекомендуется измерять пользователям несколько раз в день.

2. Однако пользователи желают получить советы по здоровому образу жизни и готовы предоставить вам больше данных. Вы получаете все необходимые сведения о пользователях от дат их рождения до ежемесячных показателей веса, количества ежечасно пройденных шагов и расходованных калорий. Очевидно, что эти разные типы данных собираются с совершенно разными скоростями (частотами).

3. Вскоре вы понимаете, что некоторые из типов данных не представляют для приложения никакой ценности, и поэтому прекращаете их сбор.

4. Но пользователи не просматривают выводимые им сообщения, в том числе

о ненадобности отдельных показателей, и вы вынуждены продолжать сбор в том числе ненужных данных...

5. И затем правительство одного из крупнейших рынков вашей целевой аудитории принимает закон о защите персональных данных о состоянии здоровья пользователей, и вам необходимо либо удалить все данные, либо зашифровать их — в базу данных нужно добавить еще одно, зашифрованное, поле.

6. Изменения продолжаются до бесконечности.

Если в вашем проекте важна гибкость в запросах и структурной организации данных, подобная описанной выше, то можете смело обращаться к услугам нереляционной базы данных, обеспечивающей разумный баланс между общей гибкостью и временной точностью системы.

Еще одно преимущество нереляционной базы данных общего назначения, в отличие от базы данных, ориентированной на хранение временных рядов, заключается в возможности быстрой интеграции в систему данных, не относящихся к временным рядам, для образования перекрестных связей между наборами данных. Иногда реляционная база данных общего назначения — это просто удачное сочетание высокопроизводительных решений и SQL-подобных инструментов, полученных без понимания того, как правильно оптимизировать реляционную схему под задачи управления временными рядами.

В этом разделе мы рассмотрели решения, основанные на нереляционных СУБД (Рассмотрение принципов оптимизации реляционных баз данных для обработки временных рядов выходит за рамки материала курса ввиду высокой их сложности. Этому вопросу посвящены публикации, упомянутые в конце раздела) и некоторые другие широко используемые технологии. Несмотря на то что доминирующие на сегодняшний момент технологии будут непрерывно совершенствоваться, общие принципы их построения, особенности использования и конкурентные преимущества останутся неизменными. В качестве краткого итога перечислим достоинства упомянутых ранее технологий баз данных.

• Высокая скорость чтения или записи (либо и то, и другое).

• Гибкая структура данных.

• Технологии передачи данных push и pull.

• Автоматизированные инструменты сбора данных.

Базы данных, ориентированные на хранение временных рядов, предлагают наиболее высокую степень автоматизации операций по обработке временных рядов, но при этом характеризуются меньшей гибкостью и предоставляют меньше возможностей по интегрированию перекрестных данных, чем нереляционные базы данных общего назначения.

По большому счету базы данных характеризуются большей гибкостью, чем хранилища из плоских файлов, но это означает, что базы данных менее оптимизированы и производительны, чем файловые решения, о которых мы поговорим далее.

*Файловые решения*

В общем понимании база данных — это программное обеспечение, в котором удачно объединяются сценарии автоматизации и хранилище данных. А по своей сути это обычный файл, обернутый в специальное программное обеспечение, которое делает его максимально безопасным и простым в использовании.

Иногда имеет смысл отказаться от внешней оболочки и взять ответственность за хранение данных на себя. Хотя это не очень распространено в бизнес-приложениях, так часто поступают в научных программах и некоторых типах промышленного программного обеспечения (например, в приложениях высокочастотной биржевой торговли), в которых производительность имеет первостепенное значение.

В таких ситуациях разработка гораздо более сложного конвейера данных, обеспечивающего распределение памяти для хранения данных, открытие, чтение и закрытие файлов, их защита и так далее выполняются специалистами по анализу данных вручную. Разумеется, это намного более сложная задача, чем написание запросов к базе данных.

Решение на основе плоских файлов будет неплохим вариантом при выполнении любого из следующих требований.

• Формат данных выбран правильно, позволяя собирать всю необходимую информацию за достаточно долгий период времени.

• Скорость обработки данных зависит от скорости их ввода-вывода, поэтому нет смысла тратить время и усилия на ее повышение.

• Свободный доступ к данным не требуется; достаточно считывать данные последовательно.

Далее мы кратко остановимся на некоторых популярных решениях, основанных на плоских файлах. Не забывайте, что вы всегда можете создать собственный формат хранения файлов (https: //perma. cc/DU98-FWW9), хотя это довольно сложно. Так поступать стоит только в случае владения языком программирования, отличающимся высокой производительностью, например C++ или Java.

Если ваша система достаточно зрелая и производительная для работы с плоской файловой системой, то такая реализация будет обладать рядом неоспоримых преимуществ, даже несмотря на высокую трудоемкость процесса переноса данных из базы данных в отдельные файлы.

• Формат плоского файла не зависит от системы. Для обмена данными просто сделайте файлы общедоступными. При этом не нужно просить коллегу открывать удаленный доступ к базе данных или создавать зеркальную базу данных.

• Временные затраты на ввод (вывод) данных в обычном файле несравнимо меньшие, чем при работе с базой данных, поскольку определяются простой операцией чтения файла (записи в файл) без необходимости поиска и извлечения из базы данных.

• Порядок, в котором должны считываться данные, указывается в самом формате файла, что менее характерно для баз данных. Такой подход повышает скорость последовательного считывания данных, что благосклонно сказывается на производительности, например, обучения алгоритмов глубокого обучения.

• Конечные данные занимают в памяти гораздо меньший объем, чем при использовании базы данных, благодаря возможности максимально увеличить степень сжатия. Вы можете предельно точно настроить степень сжатия данных, чтобы добиться оптимального баланса между потребностями в минимизации размера хранилища данных и скорости ввода-вывода данных. Высокая степень сжатия позволяет уменьшить объем данных, но понижает скорость операций ввода-вывода.

*Библиотека NumPy*

Если данные временных рядов носят исключительно числовой характер, то для их хранения можно смело применять массивы библиотеки NumPy языка Python. Такие массивы могут иметь самые разные типы данных, а их относительная производительность освещается в большом количестве публикаций. Самостоятельно оценку производительности операций с различными типами массивов NumPy можно выполнить с помощью решения, представленного в репозитории array\_storage\_benchmark (https://perma.cc/ZBS7-PR56) на GitHub.

Главный недостаток массива NumPy заключается в невозможности хранения в нем сразу нескольких типов данных. Это означает, что вы не можете автоматически сохранять в ней разнородные данные временных рядов и должны придумать способ приведения их к единственному типу данных в необработанном или обработанном виде (это ограничение можно обойти). Еще один важный недостаток массивов NumPy обусловливается тем фактом, что в нем не предусмотрена возможность добавления подписей к строкам и столбцам, поэтому, например, нет простого способа временной разметки отдельно каждой строки массива.

Преимущество использования массива NumPy состоит в поддержке большого количества форматов сохранения, включая сжатый двоичный формат, который позволяет сэкономить место и обеспечивает быстрый ввод-вывод по сравнению с традиционными базами данных. С точки зрения анализа и хранения данных массив NumPy является необычайно производительной структурой данных.

*Библиотека Pandas*

Для более простой разметки данных или сохранения разнородных данных временных рядов (или и того, и другого) обратитесь к более структурно сложному и гибкому фрейму данных пакета Pandas. Он станет незаменимым при работе с временными рядами, состоящими из большого количества разнородных данных, в том числе включающих счетчики событий (целочисленные значения), характеристики состояния (числа с плавающей точкой) и подписи (текстовые строки или программный код). В таких случаях фреймы данных пакета Pandas (“pandas” происходит от словосочетания “panel data” (панельные данные), представляющего привычный формат в большинстве случаев использования) оказываются наиболее естественным решением.

Фреймы данных Pandas находят широкое применение в анализе временных рядов — результаты оценки их производительности при хранении данных разных форматов приведены на тематических площадках (https://perma.cc/BNJ5-EDGM).

*Разнозначные инструменты языка R*

Собственный формат хранения объектов языка R представлен классами. .Rds и .Rdata. В них данные представлены в битовой форме, что обусловливает более высокую, чем в текстовом формате, степень сжатия и скорость выполнения операций ввода-вывода. В этом понимании объекты языка R во многом напоминают фреймы данных Pandas языка Python. Более того, как в R, так и в Python фреймы данных можно сохранять в независимом от языка битовом формате feather (https://perma.cc/4C3J-TBK8). В R наибольшую производительность будут показывать, конечно же, собственные двоичные форматы. В табл. 4.3 приво.дятся сравнительные характеристики разных файловых форматов.

Таблица 4.3. Сравнение размера и производительности файловых форматов

Название Относительный размер Относительное время загрузки

.RDS 1х 1х

feather 2х 1х

CSV 3х 20х

Как видите, собственный формат выступает явным победителем как по объему хранящихся данных, так и по скорости операций вводы-вывода. Наряду с этим аутсайдером выступает текстовый формат, отличающийся более низкими, чем в двоичном формате, оценками производительности.

*Пакет Хаrrау*

При работе с многомерными временными рядами обратите внимание на одно из промышленных решений для хранения данных — в частности, на пакет Хаrrау, выделяющийся рядом полезных возможностей.

• Именованные размерности.

• Векторизованные математические операции, подобные применяемым в NumPy.

• Групповые операции, как в Pandas

• Функциональные особенности, свойственные базам данных, включая команды индексации по временным диапазонам

• Различные форматы хранения файлов

Хаrrау — это структура данных, которая поддерживает множество функций обработки данных временных рядов, в том числе индексации и повторной временной выборки, интерполяции данных и работы с отдельными компонентами даты и времени. Пакет Хаrrау разрабатывался как высокопроизводительный научный вычислительный инструмент и остается крайне недооцененным решением для анализа временных рядов.

Хаrrау представляется отдельной структурой данных в языке Python, обеспечивающей несколько вариантов хранения временных рядов. В ней возможны как сериализация, так и представление данных в еще одном двоичном формате — netCDF , — универсальном формате обмена научными данными, поддерживаемом многими платформами и языками. Для тех, кто ищет эффективный инструмент для работы с временными рядами в языке Python, пакет Хаrrау будет очень благоразумным выбором.

ак вы могли заметить, существует большое количество способов хранения данных временных рядов в плоских файлах. Одни из них предполагают использование ассоциированных функций (Хаrrау), а другие — работают только с упрощенными, исключительно числовыми наборами данных (NumPy). При переносе информации из конвейера базы данных в конвейер, основанный на использовании отдельных файлов, неизбежно будут возникать определенные трудности, связанные с необходимостью упрощения данных и переписывания сценариев — формализация логики базы данных в явных сценариях ETL (Extract-Transform-Load — извлечение-преобразование-загрузка). В случаях, когда первостепенное значение имеет производительность, перемещение базы данных в отдельные файлы окажется наиболее правильным шагом по оптимизации, позволяющим существенно снизить задержки в обработке данных. Давайте посмотрим, в каких именно случаях такой шаг оказывается наиболее оправданным.

• Чувствительное к временным задержкам прогнозирование, выполняемое для программного обеспечения, ориентированного на пользователя

• Операции ввода-вывода с частым повторным доступом к данным, например в задачах обучения алгоритмов глубокого обучения

С другой стороны, удобство использования, масштабируемость и гибкость базы данных оказываются более важными, чем производительность, факторами во многих приложениях. В результате оптимальным считается решение для хранения данных, при разработке которого учитываются как характер и назначение данных, так и способ их обработки.

***Дополнительные источники***

*• Базы данных временных рядов*

Jason Moiron, “Thoughts on Time-Series Databases” jmoiron blog, June 30, 2015, https://perma.cc/8GDC-6CTX

Популярный пост в блоге 2015 года, дающий представление о раннем периоде разработки баз данных временных рядов и культе сбора данных “на все случаи жизни” Содержит подробный обзор решений для хранения временных рядов и типичных сценариев их реализации. Будет полезен для начинающих специалистов по администрированию и разработке баз данных.

Preetam Jinka, “List of Time Series Databases,” Misframe blog, April 9, 2016, https://perma.cc/9SCQ-9G57

Длинный, регулярно обновляемый список баз данных временных рядов, предоставляющих сведения о наиболее популярных в настоящий момент времени решениях. В описании каждой базы данных проводится сравнение ее с ближайшими конкурентами и предшественниками.

Peter Zaitsev, “Percona Blog Poll: What Database Engine Are You Using to Store Time Series Data?”Percona blog, February 10, 2017, https: //perma. cc/5PXFBF7L

Опрос инженеров баз данных, проведенный в 2017 году, показал, что реляционные базы данных занимают доминирующе положение на рынке, причем 35% опрашиваемых указали, что используют такие базы данных для хранения временных рядов. Лидерами опроса стали такие системы, как ElasticSearch, InfluxDB, MongoDB и Prometheus.

Rachel Stephens, “The State of the Time Series Database Market,” RedMonk, April 3, 2018, https://perma.cc/WLA7-ABRU

Недавний отчет, основанный на изучении данных и выполненный специалистом по анализу данных, включает описание наиболее популярных решений для хранения временных рядов методом эмпирического исследования активности сервисов GitHub и Stack Overflow. Отчет указывает на высокий уровень специализации решений для хранения временных рядов, что обусловлено широким выбором сценариев реализации, а также тенденцией к разработке преимущественно узкопрофильных баз данных.

Prometheus.io, “Prometheus Documentation: Comparison to Alternatives,” n.d.https://perma.cc/M83E-NBHQ

Чрезвычайно подробный и полный список различий в функциональных особенностях системы Prometheus и других решений для хранения временных рядов. Его можно использовать в качестве краткого справочника по структурам данных и средствам хранения, наиболее часто применяемым в альтернативных решениях. Прекрасный источник сведений о наиболее востребованных технологиях, а также преимуществах и недостатках баз данных, предназначенных для хранения временных рядов.

*• Адаптация баз данных общего назначения*

Gregory Trubetskoy, “Storing Time Series in PostreSQL Efficiently,” Notes to Self blog, September 23, 2015, https: / /perma. cc/QP2D-YBTS

Старое, но до сих пор актуальное сообщение в блоге, в котором описываются общие принципы хранения временных рядов в базе данных Postgres, обеспечивающей высокую скорость обработки данных. В нем Трубецкой указывает на недостатки “наивного” подхода, основанного на размещении значений и временных меток в отдельных столбцах, и дает практические рекомендации по работе с массивами Postgres.

Josiah Carlson, “Using Redis as a Time Series Database,” InfoQ, January 2, 2016, https://perma.cc/RDZ2-YM22

Статья содержит подробные советы по работе и примеры использования системы Redis в качестве базы данных временных рядов. Несмотря на то что Redis исходно разрабатывалась как решение для хранения временных рядов, работа в ней сопряжена с рядом трудностей и требует использования специальных структур данных, которые можно применять во многих других решениях для хранения временных рядов. Статью можно рассматривать как краткое справочное руководство по использованию баз данных Redis для хранения временных рядов, а также для демонстрации принципов выполнения операций над временными рядами с помощью общепринятых инструментов.

Mike Freedman, “Time Series Data: Why (and How) to Use a Relational Database Instead of NoSQL,” Timescale blog, April 20, 2017, https: //perma. cc/A6CU-6XTZ

Пост в блоге одного из разработчиков базы данных TimescaleDB, в котором описаны способы построения реляционной базы данных временных рядов, включая изменение схемы распределения памяти под задачи управления временными данными. Разработчики TimescaleDB утверждают, что основная проблема использования традиционных реляционных баз данных для хранения временных рядов заключается в низком уровне масштабируемости, предопределяющем недостаточную производительность операций передачи и извлечения данных из памяти при выполнении запросов, связанных с обработкой временных значений. TimescaleDB предлагает перенастраивать схемы распределения памяти и отображения объектов в памяти таким образом, чтобы отразить временную природу данных и сократить обмен разрозненными данными.

Hilmar Buchta, “Combining Multiple Tables with Valid from/to Date Ranges into a Single Dimension,” Oraylis blog, November 17, 2014, https://perma.cc/B8CT-BCEK

Несмотря на “заумный” заголовок, статья будет полезной всем, кто не считает себя экспертом по реляционным базам данных. В ней описан эффективный способ обработки данных, которые рассматриваются как действительные только в течение определенного промежутка времени, и показано, как правильно объединять такие данные, содержащиеся сразу в нескольких таблицах. Это необычайно распространенная и на удивление сложная задача. В рассматриваемом примере изучаются таблицы распределения кадровых ресурсов, в том числе для действительных дат трудоустройства сотрудника, для определенного отдела, для офиса с определенным расположением и для служебного автомобиля.

Весьма вероятной задачей было бы определить статус сотрудника в определенный день — выходил ли он на работу, и если да, то в каком отделе и офисе, а также имел ли он в своем распоряжении служебный автомобиль? Звучит как интуитивно понятное решение, но использовать реляционные таблицы для получения ответа на поставленную задачу очень сложно. Тем не менее в этой статье он рассмотрен достаточно детально.