УДК 004.62, 004.65 **DOI: 10.14529/cmse200406**

ОБЗОР СОВРЕМЕННЫХ СИСТЕМ ОБРАБОТКИ ВРЕМЕННЫХ РЯДОВ

© 2020 Е.В. Иванова, М.Л. Цымблер

Южно-Уральский государственный университет
(454080 Челябинск, пр. им. В.И. Ленина, д. 76)
E-mail: elena.ivanova@susu.ru, mzym@susu.ru
Поступила в редакцию: 27.09.2020

Временной ряд представляет собой последовательность хронологически упорядоченных числовых значений, отражающих течение некоторого процесса или явления. В настоящее время одним из наиболее актуальных классов задач обработки временных рядов являются приложения Индустрии 4.0 и Интернета вещей. В данных приложениях типичной является задача обеспечения умного управления и предиктивного технического обслуживания сложных машин и механизмов, которые оснащаются различными сенсорами. Такие сенсоры имеют высокую дискретность снятия показаний и за сравнительно короткое время продуцируют временные ряды длиной от десятков миллионов до миллиардов элементов. Получаемые с сенсоров данные накапливаются и подвергаются интеллектуальному анализу для принятия стратегически важных решений. Обработка временных рядов требует специфического системного программного обеспечения, отличного от имеющихся реляционных СУБД и NoSQL-систем. Системы обработки временных рядов должны обеспечивать, с одной стороны, эффективные операции добавления новых атомарных значений, поступающих в потоковом режиме, а с другой стороны, эффективные операции интеллектуального анализа, в рамках которых временной ряд рассматривается как единое целое. В статье рассмотрены особенности обработки временных рядов в сравнении с данными реляционной и нереляционной природы, и даны формальные определения основных задач интеллектуального анализа временных рядов. Представлен обзор основных возможностей трех наиболее популярных современных систем обработки временных рядов: InfluxDB, OpenTSDB, TimescaleDB.

Ключевые слова: обработка и анализ временных рядов, NoSQL, реляционная СУБД, InfluxDB, OpenTSDB, TimescaleDB.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Иванова Е.В., Цымблер М.Л. Обзор современных систем обработки временных рядов // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2020. Т. 9, № 4. С. 79–97. DOI: 10.14529/cmse200406.

Введение

Временной ряд (time series) представляет собой последовательность хронологически упорядоченных числовых значений, отражающих течение некоторого процесса или явления [16, 17]. Временные ряды возникают в широком спектре предметных областей: мониторинг показателей функциональной диагностики организма человека [21] (временные ряды ЭКГ и ЭЭГ пациента), моделирование климата [65] (временные ряды температуры воздуха, силы ветра в некоторой локации), финансовое прогнозирование [22] (временные ряды курсов акций и валют), генная инженерия [35] (цепочки ДНК как временные ряды) и др.

В настоящее время приложения Индустрии 4.0 [10] и Интернета вещей [32] представляют собой один из наиболее актуальных классов задач обработки временных рядов. В данных приложениях типичной является задача обеспечения умного управления и предиктивного технического обслуживания сложных машин и механизмов, которые оснащаются различными сенсорами. Такие сенсоры, как правило, имеют высокую дискретность снятия показаний (например, десятки раз в секунду) и за сравнительно короткое время продуциру-

ют временные ряды длиной от десятков миллионов до миллиардов элементов. Получаемые с сенсоров данные временных рядов накапливаются и подвергаются интеллектуальному анализу, который позволяет выявить знания (скрытые тренды, аномалии и др.), необходимые для принятия стратегически важных решений.

Описанный сценарий говорит в пользу того, что обработка данных временных рядов требует специфического системного программного обеспечения, функционал которого должен отличаться как от традиционных систем управления базами данных (СУБД) на основе реляционной модели [18], так и от штатных решений на основе систем класса NoSQL [11]. Системы обработки временных рядов должны обеспечивать, с одной стороны, эффективные операции добавления новых атомарных значений, поступающих в потоковом режиме, а с другой стороны, эффективные операции интеллектуального анализа, в рамках которых временной ряд рассматривается как единое целое. Далее для краткого обозначения систем обработки временных рядов нами будет использоваться аббревиатура СУБД-ВР в соответствии с устоявшимися в англоязычной литературе терминами Time Series DBMS и Time Series Database (см. например, обзоры [5, 19, 41]).

По данным портала DB-Engines.com [12] на момент написания статьи насчитывается более тридцати коммерческих и свободных СУБД-ВР. При этом за последний год СУБД-ВР стабильно входят в тройку наиболее популярных категорий СУБД, где популярность системы представляет собой интегральный показатель частоты ее упоминания в социальных сетях, сервисе Google Trends, на сайтах с предложениями работы и др. Однако тщательный поиск отечественной и зарубежной научной литературы, предпринятый авторами, показывает, что, по-видимому, имеет место относительный недостаток обзорных статей по тематике СУБД-ВР. Например, с момента выхода обзора [41] в 2015 г. появилось большое количество новых систем, в обзоре [5] 2017 г. рассмотрены только свободные системы, а недавний обзор [19] 2020 г. рассматривает только немногочисленные системы с точки зрения поддержки ими краевых вычислений (Edge Computing) [7]. Целью настоящей статьи является попытка восполнить указанный пробел и дать обзор основных современных систем обработки данных временных рядов.

Статья организована следующим образом. В разделе 1 представлены особенности обработки временных рядов в сравнении с данными реляционной и нереляционной природы, и даны формальные определения основных задач обработки временных рядов. В разделе 2 рассмотрены наиболее популярные современные системы обработки временных рядов. Заключение резюмирует результаты исследования.

1. Особенности обработки данных временных рядов

В данном разделе рассматриваются концептуальные сходства и отличия СУБД-ВР от реляционных СУБД и NoSQL-систем (раздел 1.1) и основные задачи интеллектуального анализа данных временных рядов (раздел 1.2).

1.1. Сравнение СУБД-ВР с реляционными СУБД и NoSQL-системами

Реляционные СУБД предполагают два основных класса приложений: *обработка транзакций в реальном времени (OLTP, Online Transaction Processing)* [23] и *интерактивная аналитическая обработка данных (OLAP, Online Analytical Processing)* [48].

В приложениях OLTP под транзакцией понимают набор последовательных операций модификации информации в базе данных, которые рассматриваются как неделимая еди-

ница работы с данными и переводят базу данных из одного согласованного состояния в другое. Результатом транзакции СУБД является фиксация (успешное выполнение) или откат (неудачное выполнение) всех входящих в нее операций. Типичные сценарии использования временных рядов, рассмотренные выше, показывают, что СУБД-ВР не нуждаются в поддержке OLTP: данные накапливаются и могут архивироваться для экономии объема дисковой памяти, но операции модификации и удаления данных отсутствуют.

В этой связи СУБД-ВР имеют схожесть с реляционными СУБД, обслуживающими *хранилища данных (Data Warehouse)* [60]. Хранилище данных представляет собой предметно-ориентированную информационную базу данных, предназначенную для поддержки принятия решений в крупной организации. Данные в хранилище поступают из внешних источников, подвергаются очистке (исправление различного рода ошибок, заполнение пустых значений и др.) и интеграции (приведение к единым форматам и др.), но после добавления не корректируются и не удаляются.

Приложения OLAP строятся на основе хранилищ данных и предполагают подготовку суммарной (агрегированной) информации на основе многомерных данных. Агрегация может выполняться на основе дистрибутивных функций (минимум, максимум, сумма и др.), алгебраических функций (среднее, стандартное отклонение и др.) или целостных функций (медиана, мода и др.). Основной структурой в данном сценарии обработки является OLAP-куб (многомерный массив данных), создаваемый соединением таблиц хранилища в соответствии со схемой «звезда» или «снежинка». В центре схемы находится таблица фактов, содержащая сведения об объектах или событиях, совокупность которых подвергается анализу. Лучами в данной схеме являются таблицы измерений, связанных с таблицей фактов посредством внешнего ключа, которые содержат атрибуты событий, сохраненных в таблице фактов. Количество возможных вариантов агрегации информации определяется на основе числа смысловых уровней иерархии в каждом из измерений.

Однако в случае с СУБД-ВР обработка данных по сценарию OLAP востребована в урезанном варианте: в этом случае время является фактически единственным измерением, агрегация по которому не требует OLAP-куба, поскольку представляет собой примитивную операцию (например, нахождение минимального или максимального значения, стандартного отклонения временного ряда и др.).

Термин *NoSQL* используется для обозначения широкого класса систем обработки данных, не имеющих в своей основе реляционной модели и не использующих язык SQL: системы «ключ — значение», документоориентированные СУБД, графовые СУБД и др. [11]. NoSQL-системы отказываются от поддержки механизма транзакций (жертвуя свойствами атомарности и согласованности данных) для обеспечения масштабируемости системы и доступности данных при высоких нагрузках в распределенных системах обработки данных. В этой связи можно отметить, что СУБД-ВР, как и NoSQL-системы, не нуждаются в механизме транзакций и ориентированы на как можно более эффективное выполнение операций вставки новых данных.

Подобно реляционным СУБД и NoSQL-системам, СУБД-ВР должны обеспечивать пользователю (прикладному программисту) язык баз данных (аналог SQL), синтаксис которого предоставляет возможность формулировать запросы создания, манипулирования и выборки данных.

1.2. Основные задачи интеллектуального анализа временных рядов

Одной из наиболее важных особенностей СУБД-ВР является встроенная поддержка интеллектуального анализа данных временных рядов. К основным задачам относят [16, 17] выявление аномалий, обнаружение шаблонов (лейтмотивов), поиск по образцу, восстановление пропущенных значений и прогноз. Ниже приводится нотация и формализованные определения указанных задач.

Временной ряд представляет собой хронологически упорядоченную последовательность числовых значений: $T = (t_1, \ldots, t_n)$, $t_i \in \mathbb{R}$, длина ряда обозначается как $|T| = n$.

Подпоследовательность $T_{i,m}$ временного ряда T представляет собой непрерывное подмножество T, состоящее из m элементов и начинающееся с позиции i: $T_{i,m} = (t_i, \ldots, t_{i+m-1})$, $1 \leqslant i \leqslant n - m + 1,\ m \ll n$.

Пусть неотрицательная симметричная функция $\mathrm{Dist} : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ используется в качестве *функции расстояния*.

Поиск аномалий предполагает обнаружение подпоследовательностей временного ряда, которые наиболее непохожи на все остальные подпоследовательности ряда. Концепция *диссонанса (discord)*, предложенная в работе [30], уточняет и формализует понятие аномалии и в настоящее время признается научным сообществом как наиболее адекватный способ поиска аномалий во временном ряде [71]. Диссонанс определяется следующим образом.

Подпоследовательности $T_{i,m}$ и $T_{j,m}$ ряда T называются непересекающимися, если $|i - j| \geqslant m$. Подпоследовательность, которая является непересекающейся к данной подпоследовательности $T_{i,m}$, обозначается как $M_{T_{i,m}}$. Подпоследовательность $T_{i,m}$ является *диссонансом*, если

$$\forall T_{j,m}, M_{T_{i,m}} \in T\ \min(\mathrm{Dist}(T_{i,m}, M_{T_{i,m}})) > \min(\mathrm{Dist}(T_{j,m}, M_{T_{j,m}})). \quad (1)$$

Иными словами, диссонанс представляет собой подпоследовательность ряда, имеющую максимальное расстояние до наиболее близкой к ней непересекающейся подпоследовательности. В работах [30] и [70] предложены алгоритмы поиска диссонансов в ряде, целиком размещенном в оперативной памяти, и для случая временного ряда, хранящегося на диске, соответственно. В настоящее время поиск аномалий во временных рядах является сферой интенсивных научных исследований (см., например, обзоры [8, 9]).

Поиск лейтмотивов (шаблонов) предполагает нахождение пар непересекающихся подпоследовательностей временного ряда, наиболее похожих друг на друга и формально определяется следующим образом [40]. Пара подпоследовательностей $\{T_{i,m}, T_{j,m}\}$ ряда T называется *лейтмотивом (motif)*, если

$$\forall a, b,\ |a - b| \geqslant m,\ i, j,\ |i - j| \geqslant m\ \mathrm{Dist}(T_{i,m}, T_{j,m}) \leqslant \mathrm{Dist}(T_{a,m}, T_{b,m}). \quad (2)$$

В работах [39, 58] были предложены алгоритмы поиска приближенного лейтмотива во временном ряде. В работе [40] предложен алгоритм MK, который находит точный лейтмотив во временном ряде. В настоящее время в области разработки методов и алгоритмов поиска лейтмотивов во временном ряде ведутся интенсивные исследования (см., например, обзоры [62, 63]).

Поиск по образцу предполагает нахождение во временном ряде подпоследовательности, форма которой наиболее похожа на заданный пользователем существенно более короткий временной ряд (поисковый запрос) и формально определяется следующим образом [49].

Пусть имеется временной ряд Q, $m = |Q| \ll n = |T|$, тогда подпоследовательность $T_{i,m}$ является *наиболее похожей (best match)* на поисковый запрос, если

$$\forall j,\ 1 \leqslant j \leqslant n - m + 1\ \ \mathrm{Dist}(Q,\ T_{i,m}) \leqslant \mathrm{Dist}(Q,\ T_{j,m}). \tag{3}$$

Данные выше определения диссонанса, лейтмотива и наиболее похожей подпоследовательности могут быть естественным образом расширены для нахождения top-k соответствующих объектов временного ряда, где k — параметр поиска. Следует также отметить, что при реализации поиска диссонансов и лейтмотивов временного ряда в качестве метрики используется, как правило, евклидово расстояние или его производные. Однако для задачи поиска по образцу для временных рядов из большинства предметных областей мера схожести DTW (Dynamic Time Warping, динамическая трансформация времени) [6], которая имеет квадратичную временную сложность, считается наиболее адекватной [50]. Разработка методов и алгоритмов поиска похожих подпоследовательностей во временном ряде остается на сегодня областью интенсивных научных исследований [36, 49, 67].

В задаче ***восстановления пропущенных значений*** *(imputation of missing values)* предполагается, что временной ряд содержит один элемент или подпоследовательности элементов, имеющие пустое значение `NULL`, которые требуется заменить на синтетические правдоподобные значения. Пустые значения отражают типичную ситуацию отсутствия показаний сенсора вследствие аппаратного или программного сбоя либо человеческого фактора.

В задаче ***прогноза*** требуется сформировать одно или несколько синтетических значений временного ряда в будущем, основываясь на исторических данных этого временного ряда и/или временных рядов, семантически близких к нему. Задача прогноза может быть рассмотрена как задача восстановления будущих значений временного ряда.

Точность работы алгоритмов восстановления и прогноза (степень правдоподобия генерируемых ими синтетических значений) устанавливается с помощью тестового временного ряда, в котором часть реальных значений заменяется на `NULL`, и вычисляется одна из мер точности [25], например, среднеквадратическая ошибка (RMSE, Root Mean Square Error):

$$\mathrm{RMSE}(T, \hat{T}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(t_i - \hat{t}_i)^2}, \tag{4}$$

где T и $\hat{T}$ — временные ряды длины n с реальными и восстановленными значениями соответственно.

В настоящее время тематике восстановления пропущенных значений и прогноза во временных рядах посвящается большое количество научных работ. Соответствующие методы и алгоритмы разрабатываются как на базе аппарата статистики и методов интеллектуального анализа данных [3, 31], так и на основе использования технологий нейронных сетей [20, 57, 72].

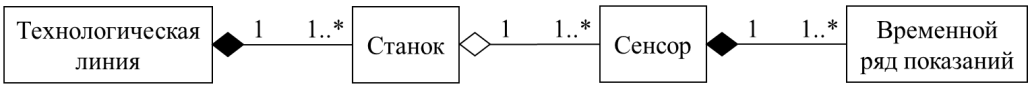
2. Основные классы СУБД-ВР и их основные представители

В работе [5] предложено разделять системы управления временными рядами на четыре следующих класса. В первый класс входят СУБД-ВР, в которых хранение временных рядов осуществляется с помощью сторонних реляционных СУБД или NoSQL-систем. Второй класс включает СУБД-ВР, самостоятельно выполняющие хранение временных рядов. В третий класс входят реляционные СУБД, обеспечивающие средства для хранения и обра-

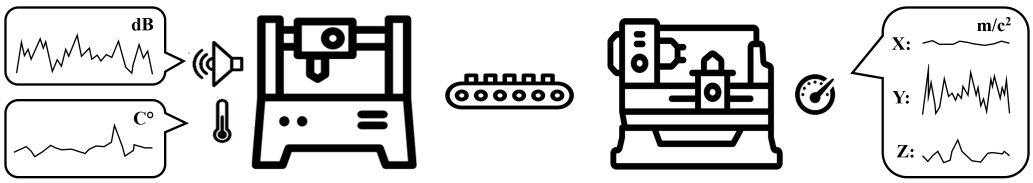
ботки временных рядов. Четвертый класс представляют коммерческие СУБД-ВР независимо от их базовой модели данных, использования сторонней СУБД либо NoSQL-системы для хранения временных рядов и др.

В настоящей статье предлагается разделить многообразие современных СУБД-ВР на два следующих класса: нативные и надстроечные. *Нативная СУБД-ВР* представляет собой самостоятельную проприетарную или свободную разработку с оригинальными языком запросов, машиной баз данных, системой хранения данных и др. Представителями данного класса являются системы InfluxDB [19, 27], Kdb+ [29], Prometheus [5, 47], Druid [68], LittleTable [51], FluteDB [34], PhilDB [37], EdgeDB [69], TSMMDB [33] и др.

Надстроечная СУБД-ВР реализуется на основе сторонней системы, обеспечивающей надстройке машину баз данных и систему хранения данных. В зависимости от модели данных, используемой сторонней системой, мы можем различать СУБД-ВР, которые являются надстройками над NoSQL-системой либо над реляционной СУБД. В настоящее время имеется широкий спектр СУБД-ВР надстроек над различными NoSQL-системами. СУБД-ВР OpenTSDB [5, 43] и Gorilla [44] функционирует на основе системы HBase [38]. СУБД-ВР BTrDB [2] может быть развернута на основе распределенных файловых систем, например, HDFS [59], GlusterFS [55], CephFS [55] и др., либо на базе одной из следующих NoSQL-систем: MongoDB [26], Cassandra, HBase [38]. Система KairosDB [5, 28] работает на основе системы Cassandra [38]. Система tsdb [13] разработана для функционирования в связке со встраиваемой СУБД Berkeley DB [54]. Базисом СУБД-ВР HeteroTSDB [64] является система Amazon DynamoDB [56]. СУБД-ВР Riak TS [19, 53] разработана как расширение системы Riak KV [52]. В подкласс СУБД-ВР, являющихся надстройками над реляционными СУБД, входят системы TimescaleDB [19, 61] (разработаны на основе СУБД PostgreSQL) и RecovDB [4] (разработана на основе СУБД MonetDB [26]).



а) Диаграмма классов



б) Иллюстративный пример

Рис. 1. Модельная предметная область

Далее представлен более детальный обзор возможностей трех систем: InfluxDB, OpenTSDB и TimescaleDB, — которые по данным портала DB-Engines.com [12] на сегодня являются наиболее популярными представителями перечисленных выше категорий СУБД-ВР. Для иллюстрации возможностей СУБД-ВР нами будет использоваться пример предметной области приложения Индустрии 4.0, представленный на рис. 1. Данный пример моделирует технологическую линию по производству изделий из металла, включающую в себя два станка. На станках установлены сенсоры для сбора данных, интеллектуальный анализ которых позволяет осуществлять предиктивное техническое обслуживание линии. На первом станке для контроля перегрева металла установлен сенсор температуры и для

улавливания волн, возникающих при изменениях в структуре металла (трещины, коррозия и др.) — сенсор акустической эмиссии. Каждый из указанных сенсоров выдает одно значение. На втором станке для контроля вибраций станка установлен виброакселерометр, который выдает три значения (виброускорение по осям X, Y, Z).

2.1. Нативная СУБД-ВР InfluxDB

InfluxDB [27] представляет собой свободную СУБД-ВР, написанную на языке программирования Go [14], распространяемую в виде исполняемого файла для основных операционных систем и аппаратных платформ. Для доступа к базе данных InfluxDB поддерживает интерфейсы командной строки и через протокол HTTP, а также клиентские библиотеки и плагины [46].

Организация хранения данных. В InfluxDB данные представляются в виде двумерной таблицы, называемой *измерением (measurement)*. В измерении имеется столбец с *метками времени (timestamp)*. Остальные столбцы измерения могут принадлежать одной из двух категорий: поле или тег. *Поле (field)* хранит данные временного ряда и состоит из *ключей (field keys)* и *значений (field values)*. *Тег (tag)* представляет собой метаданные поля и состоит из *ключей тегов (tag key)* и *значений тегов (tag values)*. Поля не индексируются, но для тегов могут быть созданы индексы. В InfluxDB отсутствует явная схема базы данных.

В InfluxDB поддерживаются понятия серии и точки. *Серия (series)* представляет собой набор данных, имеющих общие измерение, набор тегов и ключи полей. *Точка (point)* представляет собой элемент данных, состоящий из следующих компонентов: измерение, набор тегов, набор полей, метка времени. Точка однозначно идентифицируется по ее серии и метке времени.

```
1  CREATE DATABASE SensorDB
2  INSERT acoustic, machine=1 val=46
3  INSERT temperature, machine=1 val=21
4  INSERT accelerometer, machine=2 x=34.7, y=5.0, z=134.4
5  -- Просмотр схемы созданной базы данных
6  USE SensorDB
7  SHOW SERIES
8  acoustic, machine=1
9  temperature, machine=1
10 accelerometer, machine=2
```

Рис. 2. Создание базы данных в InfluxDB

На рис. 2 представлен пример создания базы данных для модельной предметной области, описанной на рис. 1. В данном примере создается база данных, содержащая три измерения: *acoustic*, *temperature* и *accelerometer*, — которые используются для хранения данных соответствующих сенсоров. Каждое из этих измерений имеет тег *machine* для указания станка, на котором устанавливается сенсор. Поля измерений — *val* для измерений *acoustic* и *temperature*, *x*, *y*, *z* для измерения *accelerometer* — служат для идентификации значений, измеряемых соответствующим сенсором. Создание указанных измерений выполняется попутно с добавлением точек данных в базу данных с помощью команды `INSERT`, поскольку в InfluxDB отсутствует возможность явного задания схемы данных. При добавлении точек данных метки времени добавляются автоматически. После этого с помощью

команды доступа к словарю СУБД-ВР `SHOW SERIES` выводится список созданных серий.

Хранение данных на физическом уровне в InfluxDB основано на использовании древовидной структуры данных LSM (Log-structured merge-tree) [42], которая используется в реляционных СУБД и обеспечивает быстрый доступ к данным в случае сценария работы, предполагающего частые запросы на вставку данных. В InfluxDB также поддерживается автоматическое сжатие данных для минимимизации объема хранимых данных.

Язык запросов. В InfluxDB поддерживается SQL-подобный язык запросов InfluxQL. На рис. 3 приведен пример запроса, вычисляющего минимальное значение точек данных температурного сенсора, установленного на первом станке.

```
SELECT MIN(val)
FROM "temperature"
WHERE "machine"='1'
```

Рис. 3. Запрос на выборку данных в InfluxQL

Из средств интеллектуального анализа временных рядов в языке InfluxQL обеспечивается прогнозирование значений ряда с помощью метода Холта—Винтерса [24, 66]. На рис. 4 приведен пример прогноза значений температурного сенсора.

```
-- 1 шаг: настройка параметров.
-- Получение данных сенсора для визуального определения параметров
-- (промежуток между "пиками" и "впадинами" и интервал смещения).
SELECT "val"
FROM "SensorDB"
WHERE "sensor"='temperature' AND
   time>='2020-08-22 22:12:00' AND time<='2020-08-28 03:00:00'
-- 2 шаг: формирование линии тренда по настроенным параметрам.
SELECT FIRST("val")
FROM "SensorDB"
WHERE "sensor"='temperature' AND
   time >= '2020-08-22 22:12:00' AND time <= '2020-08-28 03:00:00'
GROUP BY time(379m,348m)
-- 3 шаг: прогноз.
-- Прогноз 10 значений после 2020-08-28 03:00:00,
-- по 4 точки в каждом интервале смещения.
SELECT HOLT_WINTERS_WITH_FIT(FIRST("val"), 10, 4)
FROM "SensorDB"
WHERE "sensor"='temperature' AND
   time >= '2020-08-22 22:12:00' AND time <= '2020-08-28 03:00:00'
GROUP BY time(379m, 348m)
```

Рис. 4. Прогноз значения временного ряда в InfluxQL

InfluxDB поддерживает *непрерывные запросы (continous query)* — запросы, которые запускаются автоматически с заданной периодичностью. На рис. 5 приведен пример непрерывного запроса, который запускается ежечасно и находит минимальное значение показаний температурного сенсора за один час.

2.2. Надстроечная СУБД-ВР OpenTSDB на основе NoSQL-систем

OpenTSDB [43, 46] представляет собой свободную СУБД-ВР, написанную на языке программирования Java. OpenTSDB работает как надстройка над NoSQL-системами на основе

```
CREATE CONTINUOUS QUERY "cq_minimum" ON "SensorDB"
BEGIN
  SELECT MIN("val") INTO "min_temperature"
  FROM "SensorDB"
  WHERE "sensor"='temperature'
  GROUP BY time(1h)
END
SELECT * FROM "min_temperature"
```

Рис. 5. Непрерывный запрос в InfluxQL

семейства столбцов (column family store) HBase либо Cassandra [38]. Для доступа к базе данных OpenTSDB поддерживает интерфейсы командной строки и через протокол HTTP, а также клиентские библиотеки и плагины.

Организация хранения данных. Элемент временного ряда в OpenTSDB представляет собой набор, состоящий из вещественного значения, уникального идентификатора временного ряда (в терминах данной системы — метрика, metric), метки времени и непустого набора тегов. Тег представляет собой символьную строку для хранения метаданных.

OpenTSDB наследует способ организации данных от нижележащей системы класса NoSQL-системы. При этом NoSQL-система использует для хранения данных следующие системные таблицы с зарезервированными именами: `tsdb` для данных из временных рядов и `tsdb-uid` `tsdb-tree`, `tsdb-meta` для служебных данных. Запись таблицы `tsdb` представляет собой совокупность значения элемента ряда, метки времени и значения внешнего ключа, который ссылается на таблицу `tsdb-uid` и ассоциирует данную запись с определенным временным рядом. Таблица `tsdb-uid` хранит имена метрик и значения тегов временных рядов. Таблица `tsdb-tree` используется для задания и поддержки в OpenTSDB семантической иерархии хранимых временных рядов подобно файловой структуре в операционной системе. Таблица `tsdb-meta` позволяет хранить дополнительную информацию о временных рядах, задаваемую пользователем (например, текстовую аннотацию).

```
# Создание системных таблиц в HBase для работы OpenTSDB
env ./src/create_table.sh
# Создание таблиц в OpenTSDB и вставка в них данных
put acoustic 2020-08-22 22:12:00 46 machine=1
put temperature 2020-08-22 22:12:00 21 machine=1
put accelerometer.x 2020-08-22 22:14:00 34.7 machine=2
put accelerometer.y 2020-08-22 22:14:00 5.0 machine=2
put accelerometer.z 2020-08-22 22:14:00 134.4 machine=2
```

Рис. 6. Создание базы данных в OpenTSDB

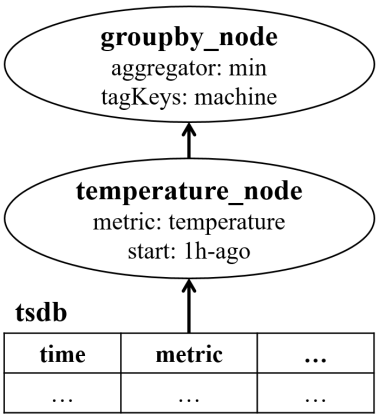
На рис. 6 представлен пример создания базы данных для модельной предметной области, описанной на рис. 1, когда в качестве базиса OpenTSDB выступает система HBase. Сначала HBase с помощью стандартного скрипта создает системные таблицы для хранения данных. Далее создается база данных, содержащая пять метрик: *acoustic*, *temperature* и *accelerometer.x*, *accelerometer.y*, *accelerometer.z*, — которые используются для хранения данных соответствующих сенсоров. Каждая из этих метрик имеет тег *machine* для указания станка, на котором устанавливается сенсор. Создание указанных измерений выполняется попутно с добавлением точек данных в базу данных с помощью команды `put`, поскольку

в OpenTSDB отсутствует возможность явного задания схемы данных. После команды `put` указывается имя метрики, отметка времени, значение точки данных и теги.

Язык запросов. В OpenTSDB запросы к базе данных записываются с помощью языка JSON. Запрос описывает ориентированный ациклический графа (execution graph), узлы которого определяют источники данных и операции преобразования данных. В запросах поддерживаются операции вычисления арифметических и логических выражений, фильтры, группировка и др., а также статистические и аналитические функции: понижение частоты дискретизации ряда (downsampling), восстановление пропущенных значений (interpolation) и др.

```
{ "start": "1h-ago",
  "executionGraph": [
  { "id": "temperature_node",
    "type": "TimeSeriesDataSource",
    "metric":
    { "type": "MetricLiteral",
      "metric": "temperature" } },
  { "id": "groupby_node",
    "type": "groupby",
    "aggregator": "min",
    "tagKeys": [ "machine" ],
    "sources": [ "temperature_node" ] } ] }
```

а) исходный текст запроса в формате JSON



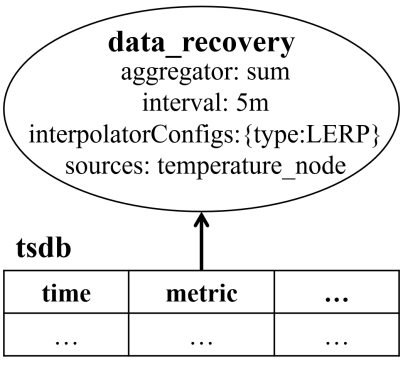
б) граф запроса

Рис. 7. Запрос на выборку данных в OpenTSDB

На рис. 7 приведен пример запроса, выполняющего операцию группировки с вычислением минимального значения по данным температурного сенсора, полученным за последний час. Граф запроса состоит из следующих узлов: узел запроса `temperature_node`, выполняющий чтение данных из метрики `temperature`, и узел преобразования данных `groupby_node`, выполняющий группировку данных по тегу `machine` и поиск минимального значения.

```
{ "id": "data_recovery",
  "aggregator": "sum",
  "interval": "5m",
  "fill": true,
  "interpolatorConfigs": [
  { "type": "LERP"
    "dataType": "numeric",
    "fillPolicy": "NAN",
    "realFillPolicy": "NONE" } ],
  "sources": [ "temperature_node" ] }
```

а) исходный текст запроса в формате JSON



б) граф запроса

Рис. 8. Восстановление значений временного ряда в OpenTSDB

На рис. 8 приведен пример запроса, выполняющего суммирование данных из метрики `temperature_node`, сгруппированных по интервалу времени 5 мин. Для получившихся пустых групп производится автозаполнение суммой, вычисляемой методом линейной интерполяции (LERP, linear interpolation) [45]. В случае, когда реальных данных для интерполяции недостаточно, в качестве суммы выдается неопределенное значение “`NaN`”.

За исключением линейной интерполяции, OpenTSDB не поддерживает развитых средств интеллектуального анализа временных рядов, однако допускает расширения от сторонних разработчиков, обеспечивающие указанную функциональность (например, библиотека R2Time [1], реализованная на языке программирования R).

2.3. Надстроечная СУБД-ВР TimescaleDB на основе реляционной СУБД

TimescaleDB [61] представляет собой СУБД-ВР с открытым исходным кодом, написанную на языке программирования C и распространяемую как расширение (extension) реляционной СУБД PostgreSQL. TimescaleDB работает в связке с экземпляром PostgreSQL и штатным образом поддерживает те же операции, что могут быть выполнены в PostgreSQL.

Организация хранения данных. В TimescaleDB данные временных рядов хранятся и обрабатываются в гипертаблицах. *Гипертаблица (hypertable)* задает именованный набор временных рядов и способ разбиения данных указанных рядов по физически хранимым реляционным таблицам. Сведения о разбиении используются для параллельной обработки указанных таблиц в PostgreSQL.

Гипертаблица **machine1 (time, val, sensor)**

Таблица **temperature_january**

time	val	sensor
2020-01-01 00:00:00	21.0	temperature
…		
2020-01-31 23:59:00	19.4	temperature

Таблица **acoustic_january**

time	val	sensor
2020-01-01 00:00:00	46.0	acoustic
…		
2020-01-31 23:59:00	50.4	acoustic

Таблица **temperature_december**

time	val	sensor
2020-12-01 00:00:00	53.2	temperature
…		
2020-12-31 23:59:00	23.4	temperature

Таблица **acoustic_december**

time	val	sensor
2020-12-01 00:00:00	34.1	acoustic
…		
2020-12-31 23:59:00	43.5	acoustic

Гипертаблица **machine2 (time, x, y, z)**

Таблица **accelerometer_january**

time	x	y	z
2020-01-01 00:00:00	34.7	5.0	134.4
…			
2020-01-31 23:59:00	33.2	15.2	131.7

Таблица **accelerometer_december**

time	x	y	z
2020-12-01 00:00:00	34.0	34.3	154.2
…			
2020-12-31 23:59:00	31.2	9.5	122.9

Рис. 9. Гипертаблицы в TimescaleDB

Рисунок 9 иллюстрирует концепцию гипертаблиц на примере данных для модельной предметной области, представленной на рис. 1. Для хранения временных рядов показаний сенсоров каждого из двух станков создаются две гипертаблицы. Гипертаблица *machine1* используется для хранения данных сенсоров, установленных на первом станке, и имеет следующие атрибуты: метка времени, показание сенсора и тип сенсора (сенсор акустической эмиссии либо температурный сенсор). Гипертаблица *machine2* определяет способ хранения показаний виброускорения по осям X, Y, Z с сенсора, установленного на втором станке. Обе гипертаблицы задают разбиение временного ряда на непересекающиеся подпоследовательности, соответствующие периодам в один месяц. Кроме того, гипертаблица *machine1* обеспечивает хранение показаний различных сенсоров в отдельных таблицах.

Рисунок 10 показывает пример создания базы данных в TimescaleDB с гипертаблицами, представленными на рис. 9. Сначала выполняется создание новой базы данных и ее расширение. Далее создаются таблицы, которые с помощью системной функции преобразуются в гипертаблицы с указанием способа разбиения. Вставка данных в гипертаблицу выполняется с помощью обычной SQL-команды `INSERT`.

```
-- Создание базы данных в PostgreSQL и ее расширение с помощью TimescaleDB
CREATE DATABASE SensorsDB;
\connect SensorsDB
CREATE EXTENSION timescaledb;
-- Создание таблиц для сенсорных данных и их преобразование в гипертаблицы
CREATE TABLE machine1    (
    time        TIMESTAMP,
    val         REAL,
    sensor      TEXT        )
CREATE TABLE machine2    (
    time        TIMESTAMP,
    x, y, z     REAL        )
SELECT create_hypertable('machine1', 'time', 'sensor', 2,
                         chunk_time_interval => INTERVAL '1 month');
SELECT create_hypertable('machine2', 'time',
                         chunk_time_interval => INTERVAL '1 month');
-- Вставка данных в гипертаблицы
INSERT INTO machine1(time, val, sensor) VALUES (NOW(), 46.0, 'acoustic');
INSERT INTO machine1(time, val, sensor) VALUES (NOW(), 21.0, 'temperature');
INSERT INTO machine2(time, x, y, z) VALUES (NOW(), 34.7, 5.0, 134.4);
```

Рис. 10. Создание базы данных в TimescaleDB

Язык запросов. Выборка данных из гипертаблиц в TimescaleDB осуществляется с помощью запросов SQL (команда SELECT с широким набором стандартных возможностей языка: подзапросы, сортировка, группировка и др.). Кроме того, в TimescaleDB язык запросов дополнен функциями, позволяющими выполнять статистический анализ временных рядов: вычисление медианы, скользящего среднего и процентилей, построение гистограмм, группировка по заданному временному интервалу и др.

TimescaleDB поддерживает *непрерывные агрегаты (continuous aggregates)* — представления, которые в фоновом режиме автоматически вычисляют и материализуют результаты специфицированного запроса. Непрерывные агрегаты похожи на материализованные представления (materialized view) в PostgreSQL, но, в отличие от последних, непрерывные агрегаты не нужно обновлять вручную: представление будет обновляться автоматически по мере добавления или изменения данных. На рис. 11 приведен пример непрерывного агрегата, который вычисляет среднее значение показаний температурного сенсора и группирует эти значения по периодам, равным одному часу.

```
CREATE VIEW ca_minimum WITH (timescaledb.continuous) AS
SELECT
  time_bucket(INTERVAL '1 hour', time) AS bucket, MIN(val)
FROM temperature
GROUP BY bucket;
SELECT * FROM ca_minimum;
```

Рис. 11. Непрерывный агрегат в TimescaleDB

TimescaleDB не предоставляет штатных функций интеллектуального анализа данных временных рядов, однако наследует от PostgreSQL возможность интеграции с библиотеками сторонних разработчиков, реализующими функции интеллектуального анализа данных внутри СУБД (например, Apache MADlib [15]), а также поддерживает возможность реализации пользовательских функций (UDF, user-defined function) для интеллектуального анализа временных рядов на языках программирования R и Python.

Заключение

В статье представлен обзор основных современных систем, обеспечивающих эффективное хранение и обработку данных временных рядов. Временной ряд представляет собой последовательность хронологически упорядоченных числовых значений, отражающих течение некоторого процесса или явления. Временные ряды возникают в широком спектре предметных областей: мониторинг показателей функциональной диагностики организма человека, моделирование климата, финансовое прогнозирование, генная инженерия, умное управление и предиктивное техническое обслуживание сложных машин и механизмов в приложениях Индустрии 4.0 и Интернета вещей и др.

Рассмотрена специфика обработки данных временных рядов, требующая системного программного обеспечения, отличного от имеющихся реляционных СУБД и NoSQL-систем. Обработка временных рядов не предполагает наличие транзакций в реальном времени (сценарий OLTP): данные временных рядов накапливаются, операции модификации и удаления данных, как правило, отсутствуют. В этом смысле системы обработки временных рядов близки к реляционным СУБД для хранилищ данных, однако интерактивная аналитическая обработка данных (сценарий OLAP) востребована здесь в урезанном виде: время является фактически единственным измерением, агрегация по которому не требует OLAP-куба, поскольку представляет собой примитивную операцию (нахождение минимального или максимального значения, стандартного отклонения временного ряда и др.). Системы обработки временных рядов, как и NoSQL-системы, не нуждаются в механизме транзакций и ориентированы на как можно более эффективное выполнение операций вставки новых данных. Подобно реляционным СУБД и NoSQL-системам, системы обработки временных рядов должны обеспечивать язык баз данных (аналог SQL) для формулирования запросов на создание, манипулирование и выборку данных. Важной особенностью систем обработки временных рядов является необходимость поддержки эффективных операций интеллектуального анализа данных, в рамках которых временной ряд рассматривается как единое целое. Даны формальные определения основных задач интеллектуального анализа временных рядов: выявление аномалий, обнаружение шаблонов (лейтмотивов), поиск по образцу, восстановление пропущенных значений и прогноз.

Предложено деление современных систем обработки временных рядов на два класса: нативные и надстроечные. Нативная система представляет собой самостоятельную проприетарную или свободную разработку с оригинальными языком запросов, машиной баз данных, системой хранения данных и др. Надстроечная система реализуется на базе существующей системы класса NoSQL либо реляционной СУБД, обеспечивающей надстройке машину баз данных и систему хранения. Представлен обзор основных возможностей следующих наиболее популярных современных систем обработки временных рядов: нативная система InfuxDB, система OpenTSDB, являющаяся надстройкой над NoSQL-системой, и система TimescaleDB, являющаяся надстройкой над реляционной СУБД. Возможности указанных систем проиллюстрированы на примере модельной предметной области приложениях Индустрии 4.0. Следует отметить, что большинство современных систем обработки временных рядов предоставляют достаточно узкий спектр встроенных средств интеллектуального анализа данных. Как правило, указанные средства позволяют решать лишь задачи восстановления пропусков или/и прогноза значений временного ряда. В соответствии с этим актуальной задачей является разработка методов и подходов для расширения спектра средств интеллектуального анализа данных, выполняемого в рамках систем обработки временных рядов.

Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (грант № 20-07-00140) и Министерства науки и высшего образования РФ (государственное задание FENU-2020-0022).

Литература

1. Agrawal B., Chakravorty A., Rong C., Wlodarczyk T.W. R2Time: A framework to analyse Open TSDB time-series data in HBase // Proceedings of the 6th International Conference on Cloud Computing Technology and Science, CloudCom 2014 (Singapore, December, 15–18, 2014). IEEE, 2014. P. 970–975. DOI: 10.1109/CloudCom.2014.84.

2. Andersen M.P., Culler D.E. BTrDB: Optimizing storage system design for timeseries processing // Proceedings of the 14th USENIX Conference on File and Storage Technologies, FAST 2016 (Santa Clara, United States, February, 22–25, 2016). P. 39–52. URL: https://www.usenix.org/system/files/conference/fast16/fast16-papers-andersen.pdf (дата обращения: 30.07.2020).

3. Andiojaya A., Demirhan H. A bagging algorithm for the imputation of missing values in time series // Expert Syst. Appl. 2019. Vol. 129. P. 10–26. DOI: 10.1016/j.eswa.2019.03.044.

4. Arous I., Khayati M., Cudre-Mauroux P., et al. RecovDB: Accurate and efficient missing blocks recovery for large time series // Proceedings of the 35th International Conference on Data Engineering, ICDE 2019 (Macao, Macao, April, 8–11, 2019). IEEE Computer Society, 2019. P. 1976–1979. DOI: 10.1109/ICDE.2019.00218.

5. Bader A., Kopp O., Falkenthal M. Survey and comparison of open source time series databases // Proceedings of the Workshop on Business, Technologies and Web, BTW 2017 (Stuttgart, Germany, March, 6–7, 2017). Gesellschaft fur Informatik, 2017. P. 249–268. URL: https://dl.gi.de/bitstream/handle/20.500.12116/922/paper31.pdf (дата обращения: 16.07.2020).

6. Berndt D.J., Clifford J. Using Dynamic Time Warping to find patterns in time series // Knowledge Discovery in Databases: Papers from the 1994 AAAI Workshop (Seattle, Washington, USA, July 1994). 1994. P. 359–370.

7. Cao K., Liu Y., Meng G., Sun O. An overview on Edge Computing research // IEEE Access. 2020. Vol. 8. P. 85714–85728. DOI: 10.1109/ACCESS.2020.2991734.

8. Chandola V., Banerjee A., Kumar V. Anomaly detection for discrete sequences: A survey // IEEE Trans. Knowl. Data Eng. 2012. Vol. 24, no. 5. P. 823–839. DOI: 10.1109/TKDE.2010.235.

9. Cook A.A., Misirli G., Fan Z. Anomaly detection for IoT time-Series data: A Survey // IEEE Internet Things Journal. 2020. Vol. 7, no. 7. P. 6481–6494. DOI: 10.1109/JIOT.2019.2958185.

10. Da X.L., Duan L. Big data for cyber physical systems in Industry 4.0: a survey // Enterp. Inf. Syst. 2019. Vol. 13, no. 2. P. 148–169. DOI: 10.1080/17517575.2018.1442934.

11. Davoudian A., Chen L., Liu M. A survey on NoSQL stores // ACM Comput. Surv. 2018. Vol. 51, no. 2. P. 40:1–40:43. DOI: 10.1145/3158661.

12. DB-Engines Ranking of Time Series DBMS. URL: https://db-engines.com/en/ranking/time+series+dbms (дата обращения: 16.07.2020).

13. Deri L., Mainardi S., Fusco F. tsdb: A compressed database for time series // Proceedings of the 4th International Workshop on Traffic Monitoring and Analysis, TMA 2012 (Vienna, Austria, March, 12, 2012). P. 143–156. DOI: 10.1007/978-3-642-28534-9_16.

14. Donovan A.A.A., Kernighan B.W. The Go programming language. Addison-Wesley, 2015. 380 p. ISBN: 978-0134190440.

15. Hellerstein J.M., Re C., Schoppmann F., et al. The MADlib analytics library or MAD skills, the SQL // PVLDB. 2012. Vol. 5, no. 12. P. 1700–1711. DOI: 10.14778/2367502.2367510.

16. Esling P., Agon C. Time-series data mining // ACM Comput. Surv. 2012. Vol. 45, no. 1. P. 12:1–12:34. DOI: 10.1145/2379776.2379788.

17. Fu T.C. A review on time series data mining // Eng. Appl. of AI. 2011. Vol. 24, no. 1. P. 164–181. DOI: 10.1016/j.engappai.2010.09.007.

18. Garcia-Molina H., Ullman J.D., Widom J. Database systems – the complete book. Pearson, 2009. 1203 p.

19. Grzesik P., Mrozek D. Comparative analysis of time series databases in the context of Edge computing for low power sensor networks // Proceedings of the 20th International Conference on Computational Science, ICCS 2020 (Amsterdam, The Netherlands, June, 3–5, 2020). Part V. 2020. P. 371–383. DOI: 10.1007/978-3-030-50426-7_28.

20. Guo Z., Wan Y., Ye H. A data imputation method for multivariate time series based on generative adversarial network // Neurocomputing. 2019. Vol. 360. P. 185–197. DOI: 10.1016/j.neucom.2019.06.007.

21. Hamdi S., Chaabane N., Bedoui M.H. Intra and Inter Relationships between Biomedical Signals: A VAR Model Analysis // Proceedings of the International Conference on Information Visualisation, IV 2019 (Paris, France, July, 2–5, 2019). P. 411–416. DOI: 10.1109/IV.2019.00076.

22. Hanif M. Relationship between oil and stock markets: Evidence from Pakistan stock exchange // International Journal of Energy Economics and Policy. 2020. Vol. 10, no. 5. P. 150–157. DOI: 10.32479/ijeep.9653.

23. Harizopoulos S., Abadi D.J., Madden S., Stonebraker M. OLTP through the looking glass, and what we found there // Making Databases Work: the Pragmatic Wisdom of Michael Stonebraker / Ed. by Brodie M.L. ACM / Morgan & Claypool, 2019. P. 409–439. DOI: 10.1145/3226595.3226635.

24. Holt C.E. Forecasting seasonals and trends by exponentially weighted averages // International Journal of Forecasting. 2004. Vol. 20, no. 1. P. 5–10. DOI: 10.1016/j.ijforecast.2003.09.015.

25. Hyndman R.J., Koehler A.B. Another look at measures of forecast accuracy // International Journal of Forecasting. 2006. Vol. 22, no. 4. P. 679–688. DOI: 10.1016/j.ijforecast.2006.03.001.

26. Idreos S., Groffen F., Nes N., et al. MonetDB: Two decades of research in column-oriented database architectures // IEEE Data Engineering Bulletin. 2012. Vol. 35, no. 1. P. 40–45.

27. InfluxDB 1.8 Documentation. URL: `https://docs.influxdata.com/influxdb/v1.8/` (дата обращения: 27.09.2020).

28. KairosDB documentation. URL: `https://kairosdb.github.io/docs/build/html/` (дата обращения: 27.09.2020).

29. Kdb+ and q documentation. URL: https://code.kx.com/ (дата обращения: 27.09.2020).

30. Keogh E., Lin J., Fu A. HOT SAX: efficiently finding the most unusual time series subsequence // Proceedings of the 5th IEEE International Conference on Data Mining, ICDM'05 (Houston, Texas, November, 27–30, 2005). 2005. P. 8. DOI: 10.1109/ICDM.2005.79.

31. Khayati M., Cudré-Mauroux P., Böhlen M.H. Scalable recovery of missing blocks in time series with high and low cross-correlations // Knowl. Inf. Syst. 2020. Vol. 62, no. 6. P. 2257–2280. DOI: 10.1007/s10115-019-01421-7.

32. Kumar S., Tiwari P., Zymbler M. Internet of Things is a revolutionary approach for future technology enhancement: a review // Journal of Big Data. 2019. Vol. 6. Article 111. DOI: 10.1186/s40537-019-0268-2.

33. Lan L., Shi R., Wang B., et al. A lightweight time series main-memory database for IoT real-time services // Proceedings of the 6th International Conference on Internet of Vehicles, Technologies and Services Toward Smart Cities, IOV 2019 (Kaohsiung, Taiwan, November, 18–21, 2019). P. 220–236. DOI: 10.1007/978-3-030-38651-1_19.

34. Li C., Li B., Bhuiyan M.Z.A., et al. FluteDB: An efficient and scalable in-memory time series database for sensor-cloud // J. Parallel Distributed Comput. 2018. Vol. 122. P. 95–108. DOI: 10.1016/j.jpdc.2018.07.021.

35. Lin T., Kaminski N., Bar-Joseph Z. Alignment and classification of time series gene expression in clinical studies // Bioinf. 2008. Vol. 24, no. 13. P. 147–155. DOI: 10.1093/bioinformatics/btn152.

36. Liu X.-Y., Ren C.-L. Fast subsequence matching under time warping in time-series databases // Proceedings of the International Conference on Machine Learning and Cybernetics, ICMLC 2013 (Tianjin, China, July, 14–17, 2013). P. 1584–1590. DOI: 10.1109/ICMLC.2013.6890855.

37. MacDonald A. PhilDB: the time series database with built-in change logging // PeerJ Comput. Sci. 2016. Vol. 2. P. e52. DOI: 10.7717/peerj-cs.52.

38. Matallah H., Belalem G., Bouamrane K. Evaluation of NoSQL databases: MongoDB, Cassandra, HBase, Redis, Couchbase, OrientDB // Int. J. Softw. Sci. Comput. Intell. 2020. Vol. 12, no. 4. P. 71–91. DOI: 10.4018/IJSSCI.2020100105.

39. Meng J., Yuan J., Hans M., Wu Y. Mining motifs from human motion // Proceedings of the Eurographics 2008 – Short Papers (Crete, Greece, April, 14–18, 2008). Eurographics Association, 2008. P. 71–74. DOI: 10.2312/egs.20081024.

40. Mueen A., Keogh E.J., Zhu Q., Cash S., Westover M.B. Exact Discovery of Time Series Motifs // Proceedings of the SIAM International Conference on Data Mining, SDM 2009 (Sparks, Nevada, USA, April, 30 – May, 2, 2009). SIAM, 2009. P. 473–484. DOI: 10.1137/1.9781611972795.41.

41. Namiot D. Time series databases // Selected Papers of the XVII International Conference on Data Analytics and Management in Data Intensive Domains, DAMDID/RCDL 2015 (Obninsk, Russia, October, 13–16, 2015). P. 132–137. URL: http://ceur-ws.org/Vol-1536/paper20.pdf (дата обращения: 16.07.2020).

42. O'Neil P., Cheng E., Gawlick D., O'Neil E. The log-structured merge-tree (LSM-tree) // Acta Informatica. 1996. Vol. 33. P. 351–385.

43. OpenTSDB 3.0 Documentation. URL: `http://opentsdb.net/docs/3x/build/html/` (дата обращения: 27.09.2020).

44. Pelkonen T., Franklin S., Cavallaro P., et al. Gorilla: A fast, scalable, in-memory time series database // Proc. VLDB Endow. 2015. Vol. 8, no. 12. P. 1816–1827. DOI: 10.14778/2824032.2824078.

45. Petersen D., Middleton D. Linear interpolation, extrapolation, and prediction of random space-time fields with a limited domain of measurement // IEEE Transactions on Information Theory. 1965. Vol. 11, no. 1. P. 18–30. DOI: 10.1109/TIT.1965.1053734.

46. Petre I., Boncea R., Radulescu C.Z., et al. A time-series database analysis based on a multi-attribute maturity model // Studies in Informatics and Control. 2019. Vol. 2, no. 2. P. 177–188. DOI: 10.24846/v28i2y201906.

47. Prometheus Documentation. URL: `https://prometheus.io/docs/` (дата обращения: 27.09.2020).

48. Queiroz-Sousa P.O., Salgado A.C. A review on OLAP technologies applied to information networks // ACM Trans. Knowl. Discov. Data. 2020. Vol. 14, no. 1. P. 8:1–8:25. DOI: 10.1145/3370912.

49. Rakthanmanon T., Campana B.J.L., Mueen A., et al. Searching and mining trillions of time series subsequences under Dynamic Time Warping // The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'12 (Beijing, China, August, 12–16, 2012). 2012. P. 262–270. DOI: 10.1145/2339530.2339576.

50. Ratanamahatana C.A., Keogh E.J. Three myths about Dynamic Time Warping data mining // Proceedings of the 2005 SIAM International Conference on Data Mining, SDM 2005 (Newport Beach, CA, USA, April, 21–23, 2005). 2005. P. 506–510. DOI: 10.1137/1.9781611972757.50.

51. Rhea S., Wang E., Wong E., et al. LittleTable: A time-series database and its uses // Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference 2017 (Chicago, IL, USA, May, 14–19, 2017). P. 125–138. DOI: 10.1145/3035918.3056102.

52. Riak KV Documentation. URL: `https://docs.riak.com/riak/kv/` (дата обращения: 27.09.2020).

53. Riak TS Documentation. URL: `https://docs.riak.com/riak/ts/` (дата обращения: 27.09.2020).

54. Seltzer M.I. Berkeley DB: A retrospective // IEEE Data Eng. Bull. 2007. Vol. 30, no. 3. P. 21–28. URL: http://sites.computer.org/debull/A07Sept/seltzer.pdf (дата обращения: 30.07.2020).

55. Sim H., Khan A., Vazhkudai S.S., Lim S.-H., Butt A.R., Kim Y. An Integrated Indexing and Search Service for Distributed File Systems // IEEE Transactions on Parallel and Distributed Systems. 2020. Vol. 31, no. 10. P. 2375–2391. DOI: 10.1109/TPDS.2020.2990656.

56. Sivasubramanian S. Amazon dynamoDB: a seamlessly scalable non-relational database service // Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data (Scottsdale Arizona, USA, May, 2012). P. 729–730. DOI: 10.1145/2213836.2213945.

57. Shen Z., Zhang Y., Lu J., et al. A novel time series forecasting model with deep learning // Neurocomputing. 2020. Vol. 396. P. 302–313. DOI: 10.1016/j.neucom.2018.12.084.

58. Shieh J., Keogh E.J. *i*SAX: Indexing and mining terabyte sized time series // Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (Las Vegas, Nevada, USA, August, 24–27, 2008). ACM, 2008. P. 623–631. DOI: 10.1145/1401890.1401966.

59. Shvachko K., Kuang H., Radia S., Chansler R. The Hadoop Distributed File System // Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies, MSST 10 (May, 2010). P. 1–10. DOI: 10.1109/MSST.2010.5496972.

60. Song I.-Y. Data Warehouse // Encyclopedia of Database Systems (2nd ed.). Ed. Liu L., Özsu M.T. Springer, 2018. DOI: 10.1007/978-1-4614-8265-9_882.

61. TimescaleDB Documentation. URL: `https://docs.timescale.com/` (дата обращения: 27.09.2020).

62. Torkamani S., Lohweg V. Survey on time series motif discovery // Wiley Interdiscip. Rev. Data Min. Knowl. Discov. 2017. Vol. 7, no. 2. DOI: 10.1002/widm.1199.

63. Truong C.D., Anh D.T. A survey on time series motif discovery // Int. J. Bus. Intell. Data Min. 2019. Vol. 15, no. 2. P. 204–227. DOI: 10.1504/IJBIDM.2019.101266.

64. Tsubouchi Y., Wakisaka A., Hamada K., et al. HeteroTSDB: An extensible time series database for automatically tiering on heterogeneous key-value stores // Proceedings of the 43rd IEEE Annual Computer Software and Applications Conference, COMPSAC 2019 (Milwaukee, WI, USA, July, 15–19, 2019). Vol. 1. P. 264–269. DOI: 10.1109/COMPSAC.2019.00046.

65. Vibhute A., Halder S., Singh P., et al. Decadal variability of tropical Indian Ocean sea surface temperature and its impact on the Indian summer monsoon // Theoretical and Applied Climatology. 2020. Vol. 141, no. 1-2. P. 551–566. DOI: 10.1007/s00704-020-03216-1.

66. Winters P.R. Forecasting sales by exponentially weighted moving averages // Management Science. 1960. Vol. 6. P. 324–342. DOI: 10.1287/mnsc.6.3.324.

67. Wu J., Wang P., Pan N., et al. KV-Match: A subsequence matching approach supporting normalization and time warping // Proceedings of the 35th IEEE International Conference on Data Engineering, ICDE 2019 (Macao, China, April, 8–11, 2019). P. 866–877. DOI: 10.1109/ICDE.2019.00082.

68. Yang F., Tschetter E., Léauté X., et al. Druid: a real-time analytical data store // Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, SIGMOD '14 (New York, NY, US, June, 2014). P. 157–168. DOI: 10.1145/2588555.2595631.

69. Yang Y., Cao Q., Jiang H. EdgeDB: An efficient time-series database for Edge Computing // IEEE Access. 2019. Vol. 7. P. 142295–142307. DOI: 10.1109/ACCESS.2019.2943876.

70. Yankov D., Keogh E.J., Rebbapragada U. Disk aware discord discovery: Finding unusual time series in terabyte sized datasets // Proceedings of the 7th IEEE International Conference on Data Mining, ICDM 2007 (Omaha, Nebraska, USA, October, 28–31, 2007). IEEE Computer Society, 2007. P. 381–390. DOI: 10.1109/ICDM.2007.61.

71. Yeh C.-C.M., Zhu Y., Ulanova L., et al. Time series joins, motifs, discords and shapelets: a unifying view that exploits the matrix profile // Data Min. Knowl. Discov. 2018. Vol. 32, no. 1. P. 83–123. DOI: 10.1007/s10618-017-0519-9.

72. Zhang Y.-F., Thorburn P.J., Xiang W., Fitch P. SSIM – A deep learning approach for recovering missing time series sensor data // IEEE Internet Things Journal. 2019. Vol. 6, no. 4. P. 6618–6628. DOI: 10.1109/JIOT.2019.2909038.

Иванова Елена Владимировна, к.ф.-м.н., кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Цымблер Михаил Леонидович, д.ф.-м.н., доцент, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

DOI: 10.14529/cmse200406

OVERVIEW OF MODERN TIME SERIES MANAGEMENT SYSTEMS

© 2020 E.I. Ivanova, M.L. Zymbler

South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia)

E-mail: elena.ivanova@susu.ru, mzym@susu.ru

Received: 27.09.2020

A time series is a sequence of chronologically ordered numerical values that reflect some process or phenomenon. Currently, one of the most topical applications related to time series processing are Industry 4.0 and Internet of Things. In these applications, the typical task is to provide intelligent control and predictive maintenance of complex machines and mechanisms that are equipped with various sensors. Such sensors have a high frequency, and in a relatively short time interval produce time series from tens of millions to billions of elements. The data obtained from the sensors is accumulated and mined to make strategic decisions. Time series processing requires specific system software that is different from the existing relational DBMS and NoSQL systems. Time series database systems should provide, on the one hand, efficient operations for adding new atomic values arriving in streaming mode, and on the other hand, efficient mining operations where time series is considered as a whole. The paper discusses the features of time series processing in comparison with data of a relational and non-relational nature, and gives formal definitions of the basic tasks of time series mining. The paper also presents an overview of three most popular modern time series database systems, namely InfuxDB, OpenTSDB, TimescaleDB.

Keywords: time series management and mining, NoSQL, relational DBMS, InfuxDB, OpenTSDB, TimescaleDB.

FOR CITATION

Ivanova E.V., Zymbler M.L. Overview of Modern Time Series Management Systems. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2020. Vol. 9, no. 4. P. 79–97. (in Russian) DOI: 10.14529/cmse200406.