УДК 004.65 doi: 10.15827/0236-235X.142.196-201 2023. Т. 36. № 2. С. 196–201

Кластеризация данных на лету для СУБД PostgreSQL

Т.М. Татарникова

Ссылка для цитирования
Татарникова Т.М. Кластеризация данных на лету для СУБД PostgreSQL // Программные продукты и системы. 2023. Т. 36. № 2. С. 196–201. doi: 10.15827/0236-235X.142.196-201
Информация о статье
Поступила в редакцию: 28.01.2023 После доработки: 31.01.2023 Принята к публикации: 03.02.2023

Аннотация. В работе определена актуальность задачи кластеризации данных в реальном масштабе времени в виде динамически встраиваемой библиотеки для СУБД PostgreSQL с открытым исходным кодом. Сформулированы условия для выполнения кластеризации в реальном времени, заключающиеся в обеспечении достаточной производительности, при которой время определения кластеров не превышает время записи данных в таблицу и ограниченное количество данных для кластеризации. Методы PostgreSQL доступны в devel-библиотеке, что позволяет использовать их для взаимодействия с данными на уровне внутреннего представления и других языков программирования, выполняющих некоторые операции быстрее, чем язык запросов SQL. Схема взаимодействия между элементами для кластеризации включает БД, в которую установлены динамически встраиваемая библиотека и расширение TimescaleDB для организации хранения данных сервером БД; интерпретатор – программная прослойка для перевода данных из внутреннего представления в типы используемого языка перед кластеризацией и, наоборот, перевода результатов кластеризации во внутренний формат для их сохранения в БД; кластеризатор – программа, выполняющая кластеризацию переданных данных согласно алгоритму. Предлагаемая библиотека представляет собой реализацию триггерной функции, которая по сути является интерпретатором, связывающим кластеризатор с БД. Если это первое срабатывание функции для таблицы, то производится выбор начальных центроидов способом, заданным пользователем. В противном случае происходит считывание данных о центроидах из таблицы. Приведена демонстрация работы библиотеки. Набор данных для кластеризации сгенерирован случайным образом с концентрацией около заданных координат центроидов. Библиотека не ограничивает пользователя как в размерности точек, которые необходимо распределить по кластерам, так и в количестве таблиц, в которые может производиться вставка данных. Ввиду вычислительной сложности алгоритмов имеется ограничение на максимальное число данных для кластеризации.
Ключевые слова: кластеризация данных, СУБД, динамически встраиваемая библиотека, метод центроида, PostgreSQL

Кластеризация применяется в маркетинге для составления рекомендаций или разработки отдельных стратегий, data mining для повышения точности методов анализа, сегментации изображений для идентификации объектов и во множестве других задач [1].

Источником кластеризации являются данные, накапливаемые в БД. Таким образом, до непосредственно кластеризации необходимо построить серию запросов для получения выборочных данных и сохранить их в некоторой временной таблице. Очевидно, что этот процесс доступа к данным занимает значительное время. Вместе с тем есть приложения, требующие выполнения кластеризации на лету. Одним из известных подходов к решению кластеризации данных в реальном масштабе времени является построение динамически встраиваемой библиотеки для СУБД [2, 3].

В статье предложено решение задачи кластеризации данных в реальном масштабе времени в виде динамически встраиваемой библиотеки для СУБД PostgreSQL с открытым исходным кодом.

Постановка задачи

Кластеризация в реальном времени предполагает, что должны выполняться некоторые условия:

– обеспечение достаточной производительности, при которой время на определение кластеров не превышает время на поступление данных в таблицу [4];

– ограничение количества данных, для которых проводится кластеризация, так как время кластеризации сильно зависит от этого показателя.

Для выполнения этих требований учитывались следующие возможные решения.

1. Оптимизировать операции ввода и чтения данных из таблиц с помощью расширения TimescaleDB, которое позволяет минимизировать зависимость скорости чтения и записи данных от количества данных за счет разбиения таблиц на чанки – физические диски, выделяемые для хранения БД. Далее быстрый поиск по дискам будет осуществляться с помощью индексов.

2. Отказаться от использования языка запросов SQL в пользу более быстрых для сложных вычислений языков.

3. Реализовать возможность ограничения количества данных, участвующих в кластеризации.

Методы и технологии

Для ускорения кластеризации предложено сохранять часть данных о ней в некоторую внутреннюю таблицу, что позволит ускорить схождение алгоритма. Данные о кластеризации должны удаляться при удалении таблицы, для которой делается кластеризация.

Благодаря открытости PostgreSQL внутренние методы можно использовать как интерфейс при взаимодействии с данными на уровне внутреннего представления. Эти методы позволяют эффективнее получать доступ к данным, хранящимся в кластере, чем выполнение тех же операций с использованием SQL, и собраны в devel-библиотеки для нескольких языков программирования, среди которых есть удобный с точки зрения широкого выбора библиотек язык Python или более быстрый язык C/C++ [5, 6].

Опишем особенности devel-библиотеки [7].

- *Абстракция MemoryContext и динамическая память.* Методы выделения памяти из данной библиотеки не требуют освобождения выделенной памяти, так как они используют абстракцию контекстов памяти: вся память, выделенная на одном контексте, может быть легко очищена. Существуют также механизмы очистки контекстов памяти в конце транзакций.
- *Концепция внутреннего представления данных и дескрипторов.* Использование devel-библиотеки для чтения данных из БД позволяет получить не сами данные, а указатели на выделенную память с данными. Если данные читаются из некоторой таблицы, то будет сформирован дескриптор, позволяющий преобразовывать данные из внутреннего представления к типам, используемым в языке программирования.
- *Концепция прямого доступа к объектам БД.* Использование внутреннего представления и прямого доступа к объектам БД позволяет значительно увеличить скорость выполнения.
- *Концепция серверного программного интерфейса (SPI).* Devel-библиотека предоставляет набор функций для выполнения запросов из кода на языке C/C++ напрямую. Серверный программный интерфейс также использует специальные контексты памяти, которые позволяют отслеживать всю выделенную память за время работы с SPI. По окончании работы с SPI вся память, выделенная на специальном контексте, освобождается.
- *Вывод ошибок и логов.* Официальная библиотека для разработчиков предлагает удобный вывод сообщений, ошибок и логов со стороны сервера с использованием форматирования строк, подобного использованию функции printf в языке C. При написании динамически встраиваемых библиотек на C++ особое внимание стоит уделять поиску исключений, так как серверное устройство не способно обрабатывать их самостоятельно.

Динамически встраиваемые библиотеки в PostgreSQL обычно предполагают, что реализация методов будет происходить на стороне сервера. Схему взаимодействий опишем следующими элементами:

– БД – хранилище данных, в которое установлены динамически встраиваемая библиотека и расширение TimescaleDB;

– интерпретатор – прослойка, в которой происходят чтение данных для кластеризации из БД и их перевод из внутреннего представления в типы используемого языка, передача преобразованных данных в кластеризатор, а также перевод результатов кластеризации во внутренний формат и их сохранение в БД;

– кластеризатор – объект, выполняющий кластеризацию переданных в него данных согласно алгоритму кластеризации: k-means, k-means++ или k-medians [8].

Графическое описание схемы взаимодействий между компонентами библиотеки и БД представлено на рисунке 1.

В результате готовое расширение будет представлять собой триггерную функцию (интерпретатор), которая срабатывает при вставке

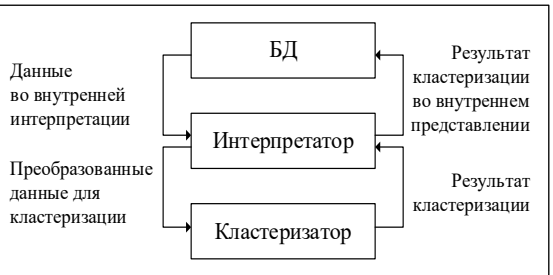


Рис. 1. Схема взаимодействий между компонентами библиотеки и БД

Fig. 1. A scheme of interactions between library and database components

значений в таблицу для кластеризации. В процессе работы считанные из таблицы данные преобразуются в стандартные типы языка программирования, отправляются в кластеризатор – класс, в котором будет производиться кластеризация, и передаются обратно в интерпретатор для сохранения результатов в БД.

Так как сервер БД редко работает с какими-либо файлами системы, принято решение о сохранении конфигурации и данных о кластеризации непосредственно в БД. Сохранение координат центров кластеров с предыдущих срабатываний триггерной функции позволит ускорить схождение алгоритмов [9]. Таким образом, в разрабатываемом расширении присутствуют две вспомогательные таблицы: таблица с данными кластеризации, в которой хранятся предыдущие центроиды кластеров, и таблица с настройками кластеризации для конкретной таблицы.

Данные кластеризации должны удаляться при удалении таблиц. Это обеспечивается с помощью внешних ключей.

Так как одно из требований – скорость кластеризации, языком разработки библиотеки стал C/C++. Помимо официальной devel-библиотеки, для разработки расширений для PostgreSQL используется фреймворк Qt. Он предоставляет множество STL-совместимых контейнеров, а также удобные методы для преобразования данных из строкового типа в другие. Для работы с json применяется библиотека nlohmann/json с множеством методов для доступа к полям json-объектов, а также проверки их типов.

Поскольку требования пользователя могут изменяться в зависимости от входных данных, было решено добавить возможность настраивать работу библиотеки под конкретный случай.

Объект в формате Json, представляющий собой настройки, имеет следующие поля:

key column – имя колонки, однозначно определяющей строку данных в таблице, текстовая строка;

clustering columns – имена колонок, по которым производится кластеризация, массив текстовых строк;

clustering algorithm – алгоритм кластеризации, текстовая строка; возможные значения: k-means, k-means++ и k-medoids;

center choice – способ выбора центров кластеров, текстовая строка; два возможных значения: manual – ручной или random – выбор случайного элемента из таблицы как начального центра кластера;

cluster count – количество центров кластеров, целое положительное число;

starting centers – опциональный объект, представляющий собой массив координат начальных центров кластеров и используемый только при ручном способе выбора центров; размерность координат должна совпадать с количеством колонок для кластеризации;

distance type – тип метрики расстояния между точками, текстовая строка; возможные значения: euclidian и manhattan;

cluster number column – имя колонки, в которую производится запись номеров кластеров, текстовая строка;

maximum data count – верхняя граница количества строк, считываемых для кластеризации, целое положительное число; используется для ограничения считываемых данных из таблицы при кластеризации;

minimum data count – нижняя граница количества строк, выше которой производится кластеризация, целое положительное число; если число строк в таблице меньше данного числа, то кластеризация не проводится; значение данного параметра должно быть больше, чем верхняя граница и количество кластеров.

Результаты

Разработанная библиотека представляет собой реализацию триггерной функции, которая производит кластеризацию при добавлении элементов в гипертаблицу из расширения TimescaleDB. Данная функция – это фактически интерпретатор, связывающий кластеризатор с БД. Описать работу триггерной функции можно следующим образом.

1. После срабатывания триггера производится проверка, что триггер был вызван перед вставкой значения, а также, что он был вызван для строки. Если триггер не соответствует этим условиям, пользователю будет выведена ошибка.

2. Так как из-за специфики работы гипертаблиц триггерная функция срабатывает не для гипертаблицы, а для ее чанка, определяются имя и схема таблицы, на чанке которой сработала триггерная функция.

3. По имени и схеме таблицы выполняются чтение конфигурации и проверка полей на типы и соответствие ограничительным условиям.

4. Производится получение данных для кластеризации во внутреннем формате. Если дан-

ных недостаточно, кластеризация не проводится, а вставка значения, на которое сработал триггер, успешно завершается. В противном случае осуществляется преобразование данных из внутреннего представления.

5. Если это первое срабатывание функции для таблицы, то выполняется выбор начальных центроидов способом, который задал пользователь. В противном случае данные о центроидах считываются из таблицы.

6. Данные для кластеризации и центроиды передаются в объект класса Clusterer, после чего производится кластеризация.

7. На основе результатов кластеризации выполняются обновление отношений данных к кластерам в таблице и успешная вставка значения, на которое сработал триггер, в таблицу. Новые данные о центроидах записываются в таблицу.

Продемонстрируем работу библиотеки в таблицах 1 и 2. Набор данных сгенерирован случайным образом, но с концентрацией около следующих координат центроидов: (5, 5, 5), (15, 5, 5) и (5, 15, 5) [10].

Начальные центры кластеров выбирались случайно.

Графики зависимости скорости работы библиотеки (время выполнения полной кластеризации *t*) от количества данных для кластеризации при использовании алгоритмов k-means и k-medians представлены на рисунке 2.

Изображенные зависимости приблизительно показывают, какой стоит сделать верхнюю границу количества данных для кластери-

Таблица 1

Вставка значений в количестве, меньшем нижней границы кластеризации

Table 1

Inserting values less than the lower clustering bound

No.	Uid uuid	Date_time integer	Val1 real	Val2 real	Val3 real	Cluster_number integer
1	123e45670-e89b-12d3-a456-426614174000	0	10	8	1	2
2	123e45670-e89b-12d3-a456-426614174001	1	3	2	8	0
3	123e45670-e89b-12d3-a456-426614174002	2	5	3	6	0
4	123e45670-e89b-12d3-a456-426614174003	3	6	4	5	0
5	123e45670-e89b-12d3-a456-426614174004	4	7	8	4	1
6	123e45670-e89b-12d3-a456-426614174005	5	12	6	7	2
7	123e45670-e89b-12d3-a456-426614174006	6	17	2	1	2
8	123e45670-e89b-12d3-a456-426614174007	7	11	1	7	2
9	123e45670-e89b-12d3-a456-426614174008	8	13	5	8	2
10	123e45670-e89b-12d3-a456-426614174009	9	11	3	2	2

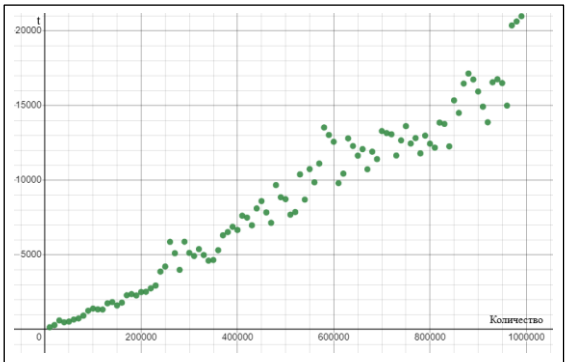
Таблица 2

Разделение на кластеры

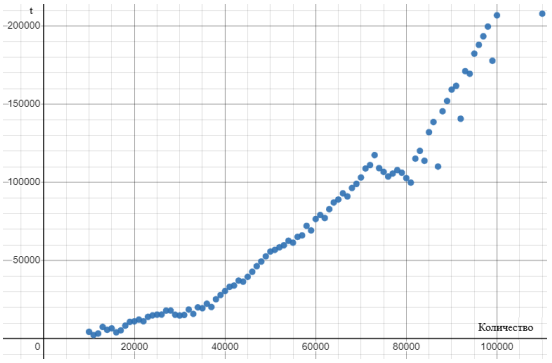
Table 2

Clustering

No.	Uid uuid	Date_time integer	Val1 real	Val2 real	Val3 real	Cluster_number integer
1	123e45670-e89b-12d3-a456-426614174000	0	10	8	1	2
2	123e45670-e89b-12d3-a456-426614174001	1	3	2	8	0
3	123e45670-e89b-12d3-a456-426614174002	2	5	3	6	0
4	123e45670-e89b-12d3-a456-426614174003	3	6	4	5	0
5	123e45670-e89b-12d3-a456-426614174004	4	7	8	4	1
6	123e45670-e89b-12d3-a456-426614174005	5	12	6	7	2
7	123e45670-e89b-12d3-a456-426614174006	6	17	2	1	2
8	123e45670-e89b-12d3-a456-426614174007	7	11	1	7	2
9	123e45670-e89b-12d3-a456-426614174008	8	13	5	8	2
10	123e45670-e89b-12d3-a456-426614174009	9	11	3	2	2
11	123e45670-e89b-12d3-a456-426614174010	10	6	15	6	1
12	123e45670-e89b-12d3-a456-426614174011	11	8	14	2	1
13	123e45670-e89b-12d3-a456-426614174012	12	2	16	7	1
14	123e45670-e89b-12d3-a456-426614174013	13	1	13	8	1
15	123e45670-e89b-12d3-a456-426614174014	14	9	15	5	1



а)



б)

Рис. 2. Время выполнения кластеризации в зависимости от количества данных: а) алгоритм k-means, б) алгоритм k-medians

Fig. 2. A clustering execution time depending on the amount of data: a) k-means algorithm, б) k-medians algorithm

зации. Например, при использовании алгоритма k-means для кластеризации данных второго порядка и вставках в таблицу раз в 20 000 мс стоит поставить ограничение числа строк для кластеризации – не более 900 000.

Выводы

Разработано расширение в виде динамически встраиваемой библиотеки для кластеризации данных в реальном времени на стороне сервера. Конфигурация для кластеризации дает возможность гибко настраивать библиотеку под решение задач с помощью моделей, основанных на близости данных к центроидам. Расширение позволяет применять для кластеризации данных различные метрики расстояний, несколько способов выбора начальных центров кластеров, а также использовать выбранный алгоритм кластеризации для конкретной таблицы.

Библиотека не ограничивает пользователя как в размерности точек, которые необходимо распределить по кластерам, так и в количестве таблиц, куда может производиться вставка данных. Из-за вычислительной сложности алгоритмов стоит ограничить максимальное число данных для кластеризации.

В перспективе расширение может развиваться в сторону оптимизации реализаций алгоритмов кластеризации и взаимодействия с БД (ввода больших данных в БД не по одной строке, а через реализацию триггера на оператор), увеличения видов и количества предоставляемых моделей кластеризации и расширения типов данных, подходящих для кластеризации.

Список литературы

1. Лакшманан В., Тайджани Дж. Google BigQuery. Все о хранилищах данных, аналитике и машинном обучении; [пер. с англ.]. СПб: Питер, 2021. 496 с.

2. Challawala S., Lakhatariya J., Mehta C., Patel K. MySQL 8 for Big Data: Effective Data Processing with MySQL 8, Hadoop, NoSQL APIs, and Other Big Data Tools. Birmingham, Packt Publ., 2017, 266 p.

3. Kaur M., Shaik B. PostgreSQL Development Essentials. Birmingham–Mumbai, Packt Publ., 2016, 210 p.

4. Шелест М.Н., Татарникова Т.М. Оценка граничных значений среднего времени отклика на запрос пользователя информационной системы // Программные продукты и системы. 2022. № 3. С. 488–492. doi: 10.15827/0236-235X.139.488-492.

5. Campbell L., Majors C. Database Reliability Engineering: Designing and Operating Resilient Database Systems. CA, Sebastopol, O'Reilly Media Publ., 2017, 294 p.

6. Фомин Д.С., Бальзамов А.В. Проблематика обработки транзакций при использовании микросервисной архитектуры // Изв. вузов. Поволжский регион. Технические науки. 2021. Т. 58. № 2. С. 15–23. doi: 10.21685/2072-3059-2021-2-2.

7. Попов С.Г., Фридман В.С. Обзор методов динамического распределения данных в распределенных системах управления базами данных // Информатика, телекоммуникации и управление. 2018. Т. 11. № 4. С. 82–107.

8. Nathiya G., Punitha S.C., Punithavalli M. An analytical study on behavior of clusters using K means, EM and K* means algorithm. Int. J. of Computer Science and Information Security, 2010, vol. 7, no. 3, pp. 185–190.

9. Bogatyrev V.A., Bogatyrev A.V., Bogatyrev S.V., Polyakov V.I. Redundant service of request copies by a sequence of groups of reserved nodes. CEUR-WS, 2018, pp. 135–140.

10. Кутузов О.И., Татарникова Т.М. Из практики применения метода Монте-Карло // Заводская лаборатория. Диагностика материалов. 2017. Т. 83. № 3. С. 65–70.

Software & Systems doi: 10.15827/0236-235X.142.196-201 2023, vol. 36, no. 2, pp. 196–201

On-the-fly data clustering for the PostgreSQL database management system

Tatiana M. Tatarnikova

For citation
Tatarnikova, T.M. (2023) 'On-the-fly data clustering for the PostgreSQL data management system', *Software & Systems*, 36(2), pp. 196–201 (in Russ.). doi: 10.15827/0236-235X.142.196-201
Article info
Received: 28.01.2023 After revision: 31.01.2023 Accepted: 03.02.2023

Abstract. The paper determines the relevance of the task of real-time data clustering in the form of a dynamically embedded library for the PostgreSQL open-source database management system. There are formulated conditions for performing real-time clustering, which consist in ensuring sufficient performance, in which the time for determining clusters does not exceed the time for writing data to the table and a limited amount of data for clustering. PostgreSQL methods are available in the devel-library, which allows them to be used to interact with data at the internal representation level and other programming languages that perform some operations faster than the SQL query language. The scheme of interaction between elements for clustering includes a database with a dynamically embedded library and the TimescaleDB extension to organize data storage by the database server; an interpreter – a software layer for translating data from the internal representation into the types of the language used before clustering, and vice versa, translating the clustering results into an internal format for saving them to the database; a clusterizer – a program that performs clustering of transmitted data according to an algorithm. The proposed library is an implementation of a trigger function, which in fact is an interpreter that connects the clusterizer with the database. If this is the first function operation for the table, then the initial centroids are selected in the way that the user specified. Otherwise, the centroid data is read from the table. There is a demonstration of the library work. The data set for clustering is randomly generated with a concentration around the given centroid coordinates. The library does not limit the user both in the dimension of points that need to be distributed among clusters, and in the number of tables for inserting data. Due to the computational complexity of the algorithms, there is a limit on the maximum amount of data for clustering.
Keywords: data clustering, database management system, dynamic link library, centroid method, PostgreSQL

Reference List

1. Lakshmanan, V., Tigani, J. (2019) *Google BigQuery: The Definitive Guide: Data Warehousing, Analytics, and Machine Learning at Scale*, O'Reilly Media Publ., 475 p. (Russ. ed.: (2021) St. Petersburg, 496 p.).
2. Challawala, S., Lakhatariya, J., Mehta, C., Patel, K. (2017) *MySQL 8 for Big Data: Effective Data Processing with MySQL 8, Hadoop, NoSQL APIs, and Other Big Data Tools*, Birmingham, Packt Publ., 266 p.
3. Kaur, M., Shaik, B. (2016) *PostgreSQL Development Essentials*, Birmingham–Mumbai, Packt Publ., 210 p.
4. Shelest, M.N., Tatarnikova, T.M. (2022) 'Boundary values evaluation for average response time to an information system user request', *Software & Systems*, (3), pp. 488–492. doi: 10.15827/0236-235X.139.488-492 (in Russ.).
5. Campbell, L., Majors, C. (2017) *Database Reliability Engineering: Designing and Operating Resilient Database Systems*, CA, Sebastopol, O'Reilly Media Publ., 294 p.
6. Fomin, D.S., Balzamov, A.V. (2021) 'The problem of transaction processing using microservice architecture', *University Proc. Volga Region. Engineering Sciences*, 58(2), pp. 15–23. doi: 10.21685/2072-3059-2021-2-2 (in Russ.).
7. Popov, S.G., Fridman, V.S. (2018) 'Review of methods for dynamic distribution of data in distributed database management systems', *Computing, Telecommunications and Control*, 11(4), pp. 82–107 (in Russ.).
8. Nathiya, G., Punitha, S.C., Punithavalli, M. (2010) 'An analytical study on behavior of clusters using K means, EM and K* means algorithm', *Int. J. of Computer Science and Information Security*, 7(3), pp. 185–190.
9. Bogatyrev, V.A., Bogatyrev, A.V., Bogatyrev, S.V., Polyakov, V.I. (2018) 'Redundant service of request copies by a sequence of groups of reserved nodes', *CEUR-WS*, pp. 135–140.
10. Kutuzov, O.I., Tatarnikova, T.M. (2017) 'Practical experience of using Monte Carlo method', *Industrial Laboratory. Diagnostics of Materials*, 83(3), pp. 65–70 (in Russ.).

Авторы
Татарникова Татьяна Михайловна [1], д.т.н., профессор, директор Института информационных технологий и программирования, tm-tatarn@yandex.ru

[1] Санкт-Петербургский государственный университет аэрокосмического приборостроения, г. Санкт-Петербург, 190000, Россия

Authors
Tatiana M. Tatarnikova [1], Dr.Sc. (Engineering), Professor, Director of the Information Technology and Programming Institute, tm-tatarn@yandex.ru

[1] Saint-Petersburg State University of Aerospace Instrumentation, St. Petersburg, 190000, Russian Federation