УДК 004.8
DOI 10.52575/2687-0932-2021-48-3-578-593

Обнаружение ресурсоемких запросов к базам данных на основе применения самоорганизующихся карт и нечеткого вывода

[1] Хайлан А.М., [2] Польщиков К.А., [2] Алгазали С.М.М.
[1] Университет Ти-Кар, Ирак, 64001, г. Ти-Кар, ул. Мостафавия
[2] Белгородский государственный национальный исследовательский университет,
Россия, 308015, г. Белгород, ул. Победы, 85
E-mail: ahmad.m.hailan@utq.edu.iq, polshchikov@bsu.edu.ru, Salahm.ghazali@uokufa.edu.iq

Аннотация. Исследования ориентированы на выявление ресурсоёмких запросов, на обработку которых затрачивается недопустимое количество ресурсов времени, процессора, дисков и памяти. Проанализированы средства мониторинга и оптимизации запросов, используемые в современных системах управления базами данных, обозначены их недостатки. Обоснована актуальность разработки новых интеллектуальных средств своевременного и достоверного обнаружения ресурсоёмких запросов к базам данных. Сделан вывод о том, что для выявления ресурсоёмких запросов представляет интерес анализ расширенного набора статистических параметров. Исходную совокупность параметров запросов удалось сократить путем предварительного нормирования набора показателей с помощью сигмоидальной функции и последующего выбора конечного числа главных компонент на основе критерия Кеттелла. Выполнена кластеризация множества запросов с помощью самоорганизующейся карты Кохонена, во избежание переобучения которой разработан алгоритм поиска рекомендуемого значения радиуса топологической окрестности активных нейронов. Для разграничения кластеров предложен алгоритм нечеткого вывода. Экспериментальные исследования показали целесообразность практического использования полученных результатов.

Ключевые слова: ресурсоемкие запросы к базам данных, самоорганизующаяся карта, метод главных компонент, нечеткий вывод, корректность обнаружения запросов.

Для цитирования: Хайлан А.М., Польщиков К.А., Алгазали С.М.М. 2021. Обнаружение ресурсоемких запросов к базам данных на основе применения самоорганизующихся карт и нечеткого вывода. Экономика. Информатика, 48 (3): 578–593. DOI 10.52575/2687-0932-2021-48-3-578-593.

Detection of resource inquiries to databases on the basis of application of self-organizing maps and fuzzy output

[1] Ahmad M. Hailan, [2] Konstantin A. Polshchikov, [2] Salach M.M. Alghazali
[1] Thi-Qar University, Mostafavia St, Thi-Qar, 64001, Iraq
[2] Belgorod State University, 85 Pobeda St, Belgorod, 308015, Russia
E-mail: ahmad.m.hailan@utq.edu.iq, polshchikov@bsu.edu.ru, Salahm.ghazali@uokufa.edu.iq

Abstract. Research is focused on identifying resource-intensive requests that consume an unacceptable amount of time, processor, disks and memory resources. The tools for monitoring and optimizing queries used in modern database management systems are analyzed, their shortcomings are indicated. The urgency of the development of new intelligent tools for the timely and reliable detection of resource-intensive queries to databases has been substantiated. It is concluded that the analysis of an extended set of statistical parameters is of interest to identify resource-intensive queries. The initial set of query parameters was reduced by preliminary normalization of the set of indicators using the sigmoidal function and subsequent selection of a finite number of principal components based on the Cattell criterion. The clustering of a set of queries was performed using a self-organizing Kohonen map, in order to avoid overfitting of which an algorithm for finding the recommended radius of the topological neighborhood of active neurons was developed. To differentiate

clusters, a fuzzy inference algorithm is proposed. Experimental studies have shown the feasibility of the practical use of the results obtained.

Keywords: resource-intensive database queries, self-organizing map, principal component analysis, fuzzy inference, correct query detection.

For citation: Hailan A.M., Polshchikov K.A., Alghazali S.M.M. 2021. Detection of resource inquiries to databases on the basis of application of self-organizing maps and fuzzy output. Economics. Information technologies, 48 (3): 578–593 (in Russian). DOI 10.52575/2687-0932-2021-48-3-578-593.

Введение

Ресурсоемким запросом к базе данных будем считать запрос, на обработку которого сервером затрачивается недопустимое количество ресурсов времени, процессора, дисков и памяти. Поиск и преобразование ресурсоёмких запросов проводится специалистами DBA (Data Base Administrator) на основании информации со стороны клиентов. Постепенное снижение производительности распределенных клиентских систем требует принятия мер проактивной оптимизации.

Запросы к базам данных формируются с помощью специальных средств языка SQL (Structured Query Language) [Wang et al., 2017]. Используемые в настоящее время методы поиска и идентификации проблемных запросов, применяющиеся в системных утилитах, не всегда позволяют выявить ресурсоемкие SQL-операторы или пропускают те запросы, которые тоже можно отнести к категории ресурсоемких. Это связано с тем, что утилиты используют ограниченное число параметров производительности, несовершенные алгоритмы анализа, основанные на простом ранжировании. Функциональная избыточность и высокая стоимость таких утилит являются их дополнительными недостатками.

Ручной поиск позволяет с лучшим качеством обнаруживать проблемные SQL-запросы. Однако сложный и трудоёмкий процесс анализа больших объёмов информации не даёт возможности человеку обрабатывать данные с приемлемой скоростью. Ручной поиск предполагает большие временные затраты, требует наличия опыта и знаний архитектуры систем управления базами данных (СУБД), особенностей хранения системной информации, языка структурированных запросов, структур и моделей данных прикладных программ.

В данной статье изложены полученные авторами результаты исследований по анализу и разработке интеллектуальных средств, предназначенных для своевременного и достоверного автоматического обнаружения ресурсоемких запросов к базам данных. Цель исследования – совершенствование процесса обнаружения ресурсоемких запросов к базам данных на основе разработки алгоритмов нейросетевой кластеризации и нечеткого вывода.

Обзор литературы по теме исследования

Известны различные подходы, позволяющие предотвратить или минимизировать влияние межзапросных взаимодействий на общий кластер. Анализ научно-технической литературы показал, что применение имеющихся средств противодействия росту ресурсов, необходимых для обработки запросов к базам данных, на практике не дает возможности избежать проблем с производительностью информационных систем. Вследствие одновременной обработки многочисленных поступивших запросов нередко наблюдается чрезмерное время ожидания их выполнения [Kalmegh et al., 2018]. Современные методы оптимизации запросов к базам данных базируются на сборе статистической информации, касающейся обрабатываемых данных. Однако, если содержимое базы данных или детализация вычисления больших данных частично скрыты пользовательскими функциями, то получение необходимой статистической информации сопряжено с большими сложностями [Sikdar, Jermaine, 2020].

В целях повышения производительности выполнения сценариев реляционных СУБД без изменения существующих запросов к базам данных разработана программная платформа Spark [Bodepudi, 2020]. Создано программное обеспечение для диагностики аномалий прерывистых медленных запросов. Данное средство без вмешательства человека позволяет диагностировать основные причины прерывистых медленных запросов. Такие запросы вызывают серьезные проблемы для пользователей баз данных, отличаются более высокими показателями используемых ресурсов, чем другие медленные запросы [Ma et al., 2020]. Разработана система, предназначенная для помощи пользователям и специалистам в оценивании и корректировке характеристик SQL-запросов. Программное средство позволяет пользователям интерактивно «отслеживать» проблемные SQL-запросы, в том числе с коррелированными подзапросами [Miao et al., 2020].

В работе [Lekshmi, Meyer-Wegener, 2021] основное внимание уделяется аппаратной оптимизации запросов в реляционной СУБД с использованием расширенных правил и моделей затрат, а также уточнению стратегий оптимизации запросов для изменения состояний их выполнения. Предложенный в статье [Lan, 2021] оптимизатор запросов активно применяется во многих системах баз данных. Данный оптимизатор на основе учета ресурсных затрат использует алгоритм перерасчета планов исполнения запросов, а затем использует модель затрат для вычисления стоимости и выбора плана с наименьшими затратами. Исследование [Muniswamaiah, 2020] ориентировано на обеспечение промежуточной обработки запросов и оптимизации запросов для гетерогенных баз данных. Высокий уровень требований к ресурсам может быть снижен за счет минимизации времени выполнения запроса, что способствует повышению эффективности использования серверной памяти и вычислительных процессоров [Bachhav, 2021]. Для повышения производительности обработки запросов к базам данных в последние годы применяются разработки на основе параллельных СУБД [Zhou, Ordonez, 2020].

В процессе обработки запроса должен формироваться план выполнения SQL-оператора, позволяющий оперативно и с наименьшими затратами получить запрашиваемую из базы данных информацию. Анализ показал, что вышеуказанные разработки и другие современные средства мониторинга и оптимизации запросов, реализованные в том числе в СУБД Oracle, MS SQL Server, DB2, сортируют и оценивают SQL-операторы на основе значений одного, реже двух или трёх показателей, характеризующих скорость обработки запроса, а также используемые для этого ресурсы. Однако, как показывает опыт, это часто приводит к ошибочному диагностированию ресурсоёмких запросов [Alghazali et al., 2021].

Таким образом, с ростом объема анализируемых данных для выявления проблемных SQL-запросов требуются все более сложные инструменты мониторинга, превентивного анализа и упреждающей оптимизации. Вышеуказанные обстоятельства определяют актуальность разработки новых интеллектуальных инструментов для своевременного и корректного обнаружения ресурсоемких запросов к базам данных.

Обоснование числа параметров, учитываемых при обнаружении ресурсоемких запросов

Наиболее развитые технологии сбора данных для поиска ресурсоёмких запросов реализованы в автоматическом хранилище рабочей нагрузки СУБД Oracle [Fernandez, 2015]. К ним относится расширенное хранилище рабочей нагрузки (Advanced Workload Repository, AWR). Эта инфраструктура предоставляет службы компонентам СУБД Oracle для сбора, сопровождения и использования статистик с целью обнаружения проблем и самонастройки [Fattah, 2014]. AWR позволяет собирать различную информацию о запросах и работе системы. Наилучшим образом для поиска и последующей оптимизации ресурсоёмких запросов подходят статистические данные по выполненным SQL-операторам, содержащиеся в таблице DBA_HIST_SQLSTAT.

Основным источником информации, полезной для выявления проблемных SQL-операторов, являются статистические данные о периоде исполнения запроса. Чаще всего

определение интенсивно использующего ресурсы SQL-запроса производится по большому количеству операций чтения с диска (disk reads), по количеству операций логического чтения (logical reads) или чтения данных из буферов (buffer gets), по числу вызовов синтаксического анализа (parse calls), по количеству выполнений (executions). Однако исследования показали, что для выявления ресурсоёмких запросов представляет интерес расширенный набор, состоящий из 32 параметров.

Исследования показали, что расширенный набор параметров является избыточным. Для учета всех этих показателей с целью обнаружения ресурсоёмких запросов потребуются алгоритмы с большой вычислительной сложностью, что нежелательно в процессе выполнения практических задач. В целях сокращения числа параметров, характеризующих SQL-запросы, был использован метод главных компонент (Principal Component Analysis, PCA) [Belattar, 2020]. Применение метода PCA позволило построить новое пространство параметров меньшей размерности. Выбор конечного числа главных компонент осуществлялся на основе критерия Кеттелла, позволяющего сохранить от 65 % до 80 % информации, сосредоточенной в исходном наборе значимых признаков. Установлено, что для обнаружения проблемных запросов к базам данных достаточно использовать от 4 до 9 значимых параметров вместо исходных 32.

С использованием полученного ограниченного числа значимых параметров множество всех возможных запросов к базам данных необходимо разбить на некоторое число кластеров, отличающихся ресурсоемкостью выполнения SQL-операторов. С этой целью предлагается использовать аппарат самоорганизующихся карт Кохонена (Self-Organization Maps, SOM) [Sinha, 2010; Sakkari, 2020; Clovis et al., 2020], на основе которого разработан алгоритм обнаружения ресурсоемких запросов.

Разработка алгоритма обнаружения ресурсоемких запросов

Пусть в составе самоорганизующейся карты имеется N нейронов. Каждый нейрон представляется вектором весов:

$$W_k = \{w_{k1}, w_{k2}, ..., w_{ki}, ..., w_{kn}\}, \quad (1)$$

где W_k – вектор весов нейрона номер k, номер нейрона принимает значения $k = 1, 2, ..., N$; w_{ki} – вес номер i нейрона номер k; n – число весовых значений нейрона.

Разработан алгоритм обнаружения ресурсоемких запросов, состоящий из следующих шагов:

Шаг 1. На вход самоорганизующейся карты поступает вектор параметров запроса $X = \{x_1, x_2, ..., x_i, ..., x_n\}$, который требуется классифицировать с точки зрения ресурсоемкости.

Шаг 2. Выбирается нейрон номер $k = 1$.

Шаг 3. Для выбранного нейрона номер k самоорганизующейся карты вычисляется мера близости вектора его весов $W_k = \{w_{k1}, w_{k2}, ..., w_{ki}, ..., w_{kn}\}$ с вектором параметров поступившего запроса по формуле:

$$d_k = \sqrt{\sum_{i=1}^{n} (x_i - w_{ki})^2}. \quad (2)$$

Шаг 4. Номер выбранного нейрона увеличивается на 1.

Шаг 5. Проверяется выполнение условия:

$$k \le N \quad (3)$$

В случае выполнения данного условия осуществляется переход к шагу 3, в противном случае – к шагу 6.

Шаг 6. Выполняется поиск активного нейрона с минимальным значением d_k.

Шаг 7. Определяется кластер, к которому принадлежит активный нейрон, на основании чего принимается решение о том, к какому классу ресурсоемкости следует отнести поступивший запрос.

Используемая в данном алгоритме самоорганизующаяся карта должна быть настроена с учетом значений параметров поступающих запросов. С этой целью создается обучающая выборка, включающая значения параметров N запросов. Каждый запрос представляется вектором:

$$\overline{X}_q = \{\bar{x}_{q1}, \bar{x}_{q2}, ..., \bar{x}_{qi}, ..., \bar{x}_{qn}\}, \quad (4)$$

где $\overline{X}_q$ – вектор параметров запроса номер q, номер запроса принимает значения $q = 1, 2, ..., N$; $\bar{x}_{qi}$ – параметр номер i запроса номер q; n – число параметров в векторе запросов.

Для обучения нейронов самоорганизующейся карты предлагается использовать следующий алгоритм:

Шаг 1. Выполняется инициализация алгоритма – задаются исходные данные: число нейронов, начальные значения скорости обучения нейронов и радиуса топологической окрестности активного нейрона.

Шаг 2. Начинается цикл обучения номер $t = 1$; вводятся начальные значений весов нейронов на основе случайной нормализации на интервале $[0; 1]$ по формуле:

$$0{,}5 - \frac{1}{\sqrt{n}} \le w_{ki1} \le 0{,}5 + \frac{1}{\sqrt{n}}, \quad (5)$$

где w_{ki1} – вес номер i нейрона номер k в первом цикле обучения.

Шаг 3. Выбирается вектор обучающей выборки номер $q = 1$.

Шаг 4. На вход самоорганизующейся карты подается выбранный вектор номер q обучающей выборки; для каждого нейрона выполняется вычисление величины d_{qkt} – евклидова расстояния между q-м вектором обучающей выборки и вектором весов нейрона номер k в текущем цикле обучения номер t по формуле:

$$\bar{d}_{qkt} = \sqrt{\sum_{i=1}^{n} \left(\bar{x}_{qi} - w_{kit}\right)^2}, \quad (6)$$

где $\bar{x}_{qi}$ – значение i-го параметра q-го вектора обучающей выборки; w_{kit} – вес номер i нейрона номер k в цикле обучения номер t.

Шаг 5. По отношению к поданному на вход самоорганизующейся карты вектору номер q обучающей выборки определяется активный нейрон с минимальным значением $\bar{d}_{qkt}$, т. е. нейрон, весовые значения которого наиболее близки к значениям q-го вектора обучающей выборки.

Шаг 6. Корректируются весовые значения вектора активного нейрона и векторов нейронов его окрестности по формуле:

$$w_{ki(t+1)} = w_{kit} + \alpha_t h_{kt} (\bar{x}_{qi} - w_{kit}), \quad (7)$$

где $w_{ki(t+1)}$ – вес номер i нейрона номер k, скорректированный для следующего цикла обучения номер $(t+1)$; w_{kit} – вес номер i нейрона номер k в текущем цикле обучения номер t; α_t – значение скорости обучения нейронов в текущем цикле обучения номер t; h_{kt} – значение функции соседства нейронов в текущем цикле обучения номер t.

Скорость изменения нейронов уменьшается от цикла обучения к циклу обучения по линейному закону. Значение функции соседства нейронов вычисляется с использованием гауссовой функции:

$$h_{kt} = \exp\left(-\frac{\gamma_k}{2\sigma_t^2}\right), \quad (8)$$

где γ_k – расстояние между координатами нейрона номер k и активным нейроном на сетке самоорганизующейся карты; σ_t – радиус топологической окрестности активного нейрона в текущем цикле обучения номер t.

Величина радиуса топологической окрестности активного нейрона уменьшается с каждым циклом обучения на 1 от начального максимального значения $\sigma_{\max}$ до некоторого заданного значения.

Величина γ_k вычисляется по формуле:

$$\gamma_k = \|r_k - r_a\|, \quad (9)$$

где r_k – координаты нейрона номер k на сетке самоорганизующейся карты; r_a – координаты активного нейрона номер a на сетке самоорганизующейся карты.

Шаг 7. Значение номера q выбранного вектора обучающей выборки увеличивается на 1. Если $q \le N$, то осуществляется переход к шагу 4, в противном случае – к шагу 8.

Шаг 8. Номер цикла обучения t увеличивается на 1. Значение α_t уменьшается на величину $\Delta\alpha$, величина σ_t уменьшается на 1.

Шаг 9. Проверяется выполнение условия:

$$\sigma_t \le \sigma^*, \quad (10)$$

где σ^* – значение радиуса топологической окрестности активных нейронов, рекомендуемое во избежание переобучения самоорганизующейся карты.

В случае, если условие (12) не выполняется, осуществляется переход к шагу 3. При выполнении данного условия алгоритм обучения завершается. Нейроны самообучающейся карты считаются настроенными для выполнения дальнейшей кластеризации.

В представленном выше алгоритме во избежание переобучения нейронов самоорганизующейся карты используется величина рекомендуемого радиуса топологической окрестности активных нейронов. От выбора значения σ^* зависит правильность настройки нейронного слоя самоорганизующейся карты.

Разработка алгоритма поиска рекомендуемого значения радиуса топологической окрестности активных нейронов

Для обоснования значения σ^* предлагается вычисление ошибки обобщения в каждом текущем цикле обучения номер t:

$$E_t \le \frac{1}{nN}\sum_{i=1}^{n}\sum_{k=1}^{N}\left(R_{kit}^2 + D_{kit}\right), \quad (11)$$

где R_{kit} – разность средних значений i-го параметра векторов обучающей выборки и средних значений i-го параметра векторов контрольной выборки, для которых нейрон номер k был активен в текущем цикле обучения номер t; D_{kit} – усредненная оценка дисперсии i-го параметра векторов обучающей выборки и i-го параметра векторов контрольной выборки, для которых нейрон номер k был активен в текущем цикле обучения номер t.

Значение R_{kit} вычисляется с помощью выражения:

$$R_{kit} = \overline{A}_{kit} - \widetilde{A}_{kit}, \quad (12)$$

где $\overline{A}_{kit}$ – среднее значение i-го параметра векторов обучающей выборки, для которых нейрон номер k был активен в текущем цикле обучения номер t; $\tilde{A}_{kit}$ – среднее значение i-го параметра векторов контрольной выборки, для которых нейрон номер k был активен в текущем цикле обучения номер t.

Среднее значение i-го параметра векторов обучающей выборки, для которых нейрон номер k был активен в текущем цикле обучения номер t, вычисляется с помощью выражения:

$$\overline{A}_{kit} = \frac{1}{J_t}\sum_{j=1}^{J_t}\overline{x}_{kitj}, \tag{13}$$

где $\overline{x}_{kitj}$ – значение i-го параметра j-го вектора обучающей выборки, для которого нейрон номер k был активен в текущем цикле обучения номер t; J_t – число векторов обучающей выборки, для которых нейрон номер k был активен в текущем цикле обучения номер t.

Среднее значение i-го параметра векторов контрольной выборки, для которых нейрон номер k был активен в текущем цикле обучения номер t, вычисляется с помощью выражения:

$$\tilde{A}_{kit} = \frac{1}{M_t}\sum_{m=1}^{M_t}\tilde{x}_{kitm}, \tag{14}$$

где $\tilde{x}_{kim}$ – значение i-го параметра m-го вектора контрольной выборки, для которого нейрон номер k был активен в текущем цикле обучения номер t; M_t – число векторов контрольной выборки, для которых нейрон номер k был активен в текущем цикле обучения номер t.

Усредненная оценка дисперсии i-го параметра векторов обучающей выборки и i-го параметра векторов контрольной выборки, для которых нейрон номер k был активен в текущем цикле обучения номер t, вычисляется по формуле:

$$D_{kit} = \frac{1}{2}\left(\overline{D}_{kit} + \tilde{D}_{kit}\right), \tag{15}$$

где $\overline{D}_{kit}$ – оценка дисперсии i-го параметра векторов обучающей выборки, для которых нейрон номер k был активен в текущем цикле обучения номер t; $\tilde{D}_{kit}$ – оценка дисперсии i-го параметра векторов контрольной выборки, для которых нейрон номер k был активен в текущем цикле обучения номер t.

Для вычисления оценки дисперсии i-го параметра векторов обучающей выборки, для которых нейрон номер k был активен в текущем цикле обучения номер t, используется выражение:

$$\overline{D}_{kit} = \frac{1}{J_t}\sum_{j=1}^{J_t}\left(\overline{x}_{kitj} - \overline{A}_{kit}\right)^2. \tag{16}$$

Для вычисления оценки дисперсии i-го параметра векторов контрольной выборки, для которых нейрон номер k был активен в текущем цикле обучения номер t, используется выражение:

$$\tilde{D}_{kit} = \frac{1}{M_t}\sum_{m=1}^{M_t}\left(\tilde{x}_{kitm} - \tilde{A}_{kit}\right)^2. \tag{17}$$

Предлагается алгоритм поиска рекомендуемого значения радиуса топологической окрестности активных нейронов, состоящий из следующих шагов:

Шаг 1. Формируется массив обучающей выборки, состоящий из N векторов параметров запросов:

$$\overline{\boldsymbol{X}} = \left\{\overline{X}_1, \overline{X}_2, ..., \overline{X}_q, ..., \overline{X}_N\right\}, \tag{18}$$

где $\overline{X}_q$ – вектор номер q обучающей выборки.

Каждый вектор обучающей выборки формируется в виде выражения (4).

Шаг 2. Формируется массив контрольной выборки, состоящий из N векторов параметров запросов:

$$\tilde{\boldsymbol{X}} = \left\{\tilde{X}_1, \tilde{X}_2, ..., \tilde{X}_q, ..., \tilde{X}_N\right\}, \quad (19)$$

где $\tilde{X}_q$ – вектор номер q контрольной выборки.

Каждый вектор контрольной выборки формируется в виде:

$$\tilde{X}_q = \left\{\tilde{x}_{q1}, \tilde{x}_{q2}, ..., \tilde{x}_{qi}, ..., \tilde{x}_{qn}\right\}, \quad (20)$$

где $\tilde{x}_{qi}$ – значение i-го параметра вектора номер q контрольной выборки

Шаг 3. Выполняется инициализация: задаются размеры сетки нейронов, число циклов её обучения T; устанавливается начальный номер цикла обучения ($t = 1$); вводятся начальные значения весов нейронов на основе случайной нормализации на интервале $[0; 1]$ по формуле (5); задаются начальные значения параметров α_t и σ_t.

Шаг 4. Для подачи на вход самоорганизующейся карты выбирается начальный вектор (номер $q = 1$) обучающей выборки.

Шаг 5. На вход самоорганизующейся карты подается выбранный вектор номер q обучающей выборки. Для каждого нейрона по формуле (6) выполняется вычисление величины $\bar{d}_{qkt}$.

Шаг 6. По отношению к поданному на вход самоорганизующейся карты вектору номер q обучающей выборки определяется активный нейрон номер a в цикле обучения номер t.

Шаг 7. Корректируются весовые значения вектора активного нейрона по формуле:

$$w_{ai(t+1)} = w_{ait} + \alpha_t\left(\bar{x}_{qi} - w_{ait}\right), \quad (21)$$

где $w_{ai(t+1)}$ – вес номер i активного нейрона, скорректированный для следующего цикла обучения номер $(t+1)$; w_{ait} – вес номер i активного нейрона в цикле обучения номер t.

Шаг 8. Выбираются нейроны, попавшие в топологическую окрестность активного нейрона радиусом σ_t.

Шаг 9. Корректируются весовые значения векторов нейронов, попавших в топологическую окрестность активного нейрона по формулам (7)–(9).

Шаг 10. Значение номера q выбранного вектора обучающей выборки увеличивается на 1. Если $q \le N$, то осуществляется переход к шагу 5, в противном случае – к шагу 11.

Шаг 11. Для подачи на вход самоорганизующейся карты выбирается начальный вектор (номер $q = 1$) контрольной выборки.

Шаг 12. На вход самоорганизующейся карты подается выбранный вектор контрольной выборки. Для каждого нейрона выполняется вычисление величины евклидова расстояния между q-м вектором контрольной выборки и вектором весов нейрона номер k в текущем цикле обучения номер t по формуле:

$$\tilde{d}_{qkt} = \sqrt{\sum_{i=1}^{n}\left(\tilde{x}_{qi} - w_{kit}\right)^2}. \quad (22)$$

Шаг 13. По отношению к поданному на вход самоорганизующейся карты вектору контрольной выборки определяется активный нейрон с минимальным значением $\tilde{d}_{qkt}$.

Шаг 14. Значение номера q выбранного вектора контрольной выборки увеличивается на 1. Если $q \le N$, то осуществляется переход к шагу 12, в противном случае – к шагу 15.

Шаг 15. Вычисляется ошибка обобщения в текущем цикле обучения по формулам (11)–(17).

Шаг 16. Номер цикла обучения t увеличивается на 1. Значение α_t уменьшается на величину $\Delta\alpha$, величина σ_t уменьшается на 1. Если $t \le T$, то осуществляется переход к шагу 4, в противном случае – к шагу 17.

Шаг 17. Выбирается цикл обучения t^* с минимальным значением $E_{\min}$ ошибки обобщения, после которого начинается рост величины E_t. Выдается сообщение о том, что при принятии решения о выборе величины σ^* рекомендуется использовать значение радиуса топологической окрестности активного нейрона, полученное в цикле обучения t^*. Определяется кластер, к которому принадлежит активный нейрон, на основании чего принимается решение о том, к какому классу ресурсоемкости следует отнести поступивший запрос.

Разработка алгоритма разграничения кластеров самоорганизующейся карты

В процессе обучения самоорганизующейся карты значения весов её нейронов настраиваются в соответствии со значениями параметров запросов, содержащихся в векторах обучающей выборки. В результате образуются кластеры нейронов, имеющих близкие значения весов. Чтобы получить возможность использовать кластеры нейронов самоорганизующейся карты для классификации запросов с точки зрения ресурсоемкости, необходимо иметь данные о границах этих кластеров.

Визуальный анализ обученных самоорганизующихся карт показал размытость, нечеткость границ полученных кластеров. При этом очень сложно точно определить конкретный интервал численных значений $w_{k1}, w_{k2}, ..., w_{ki}, ..., w_{kn}$, по которым вектор весов W_k однозначно определяет принадлежность нейрона номер k к тому или иному кластеру. В связи с этим, для определения границ кластеров нейронов можно использовать аппарат нечеткого вывода, который успешно применяется для решения различных научно-технических задач [Polshchykov et al., 2014; Polshchykov et al., 2017; Polshchykov et al., 2019; Polshchykov et al., 2020].

Чтобы при разграничении кластеров можно было учесть значения всех весов нейронов, введем величины обобщенных весов нейронов. Для вычисления величины S_k – обобщенного веса нейрона номер k – можно применить нечёткие правила следующего вида:

$$\begin{aligned} &If\ (w_{k1} = w_1^+)\ and\ (w_{k2} = w_2^+)\ and\ ...\ and\ (w_{ki} = w_i^+)\ and\ ...\ and\ (w_{kn} = w_n^+) \\ &then\ (S_k = Y_1); \end{aligned}, \quad (23)$$

$$\begin{aligned} &If\ (w_{k1} = w_1^+)\ and\ (w_{k2} = w_2^+)\ and\ ...\ and\ (w_{ki} = w_i^+)\ and\ ...\ and\ (w_{kn} = w_n^-) \\ &then\ (S_k = Y_2); \end{aligned}, \quad (24)$$

…

$$\begin{aligned} &If\ (w_{k1} = w_1^+)\ and\ (w_{k2} = w_2^+)\ and\ ...\ and\ (w_{ki} = w_i^-)\ and\ ...\ and\ (w_{kn} = w_n^-) \\ &then\ (S_k = Y_y); \end{aligned}, \quad (25)$$

…

$$\begin{aligned} &If\ (w_{k1} = w_1^-)\ and\ (w_{k2} = w_2^-)\ and\ ...\ and\ (w_{ki} = w_i^-)\ and\ ...\ and\ (w_{kn} = w_n^-) \\ &then\ (S_k = Y_z), \end{aligned}, \quad (26)$$

где w_i^+ – нечеткое множество «высокое значение i-го веса нейрона»; w_i^- – нечеткое множество «низкое значение i-го веса нейрона»; $Y_1, Y_2, ..., Y_y, ..., Y_z$ – значения

индивидуальных выводов соответствующих нечетких правил; z – число нечетких правил, $z = n^2$.

Вычисление значений Y_y следует выполнять по формуле:

$$Y_y = \frac{v_1 b_1 + v_2 b_2 + ... + v_i b_i + ... + v_n b_n}{v_1 + v_2 + ... + v_i + ... + v_n}, \quad (27)$$

где v_i – значение доли объясненной дисперсии i-го параметра векторов запросов.

Коэффициенты $b_1, b_2, ..., b_i, ..., b_n$ вычисляются с учетом того, какому нечеткому множеству (w_i^+ или w_i^-) принадлежит величина w_{ki} в нечетком правиле номер y:

$$b_i = \begin{cases} 1, & w_{ki} = w_i^+; \\ 0, & w_{ki} = w_i^-. \end{cases} \quad (28)$$

Значения $w_{k1}, w_{k2}, ..., w_{ki}, ..., w_{kn}$ могут в большей или меньшей мере соответствовать нечетким множествам w_i^+ и w_i^-. Для вычисления величины этого соответствия воспользуемся функциями принадлежности $U^+(w_{ki})$ и $U^-(w_{ki})$. Их смысл заключается в том, что значение функции $U^+(w_{ki})$ показывает, с какой вероятностью значение w_{ki} принадлежит нечеткому множеству «высокое значение i-го веса нейрона», а значение функции $U^-(w_{ki})$ показывает,
с какой вероятностью значение w_{ki} принадлежит нечеткому множеству «низкое значение i-го веса нейрона». Широкое применение в исследовательской практике получили функции принадлежности, имеющие вид, представленный на рисунке 1.

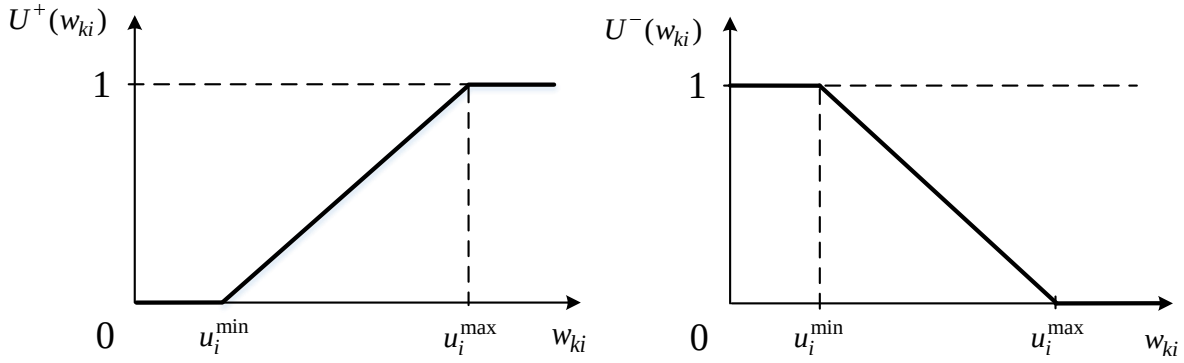

Рис. 1. Функции принадлежности $U^+(w_{ki})$ и $U^-(w_{ki})$

Fig. 1. Membership functions $U^+(w_{ki})$ and $U^-(w_{ki})$

На рисунке 1 обозначены следующие величины: $u_i^{\min}$ – минимальное значение параметра номер i в обучающей выборке векторов запросов; $u_i^{\max}$ – максимальное значение параметра номер i в обучающей выборке векторов запросов.

Чтобы вычислить показатель S_k на основе нечетких правил (23)–(26), прежде всего, необходимо выполнить фаззификацию, т. е. вычислить значения функций принадлежности величин $w_{k1}, w_{k2}, ..., w_{ki}, ..., w_{kn}$ нечетким множествам w_i^+ и w_i^-:

$$U^+(w_{ki})=\begin{cases}0, & w_{ki}<u_i^{\min};\\ \dfrac{w_{ki}-u_i^{\min}}{u_i^{\max}-u_i^{\min}}, & u_i^{\min}\le w_{ki}<u_i^{\max};\\ 1, & w_{ki}\ge u_i^{\max};\end{cases} \tag{29}$$

$$U^-(w_{ki})=\begin{cases}1, & w_{ki}<u_i^{\min};\\ \dfrac{u_i^{\max}-w_{ki}}{u_i^{\max}-u_i^{\min}}, & u_i^{\min}\le w_{ki}<u_i^{\max};\\ 0, & w_{ki}\ge u_i^{\max}.\end{cases} \tag{30}$$

Следующим этапом является агрегирование:

$$G_{k1}=U^+(w_{k1})\wedge U^+(w_{k2})\wedge\ldots\wedge U^+(w_{ki})\wedge\ldots\wedge U^+(w_{kn}), \tag{31}$$

$$G_{k2}=U^+(w_{k1})\wedge U^+(w_{k2})\wedge\ldots\wedge U^+(w_{ki})\wedge\ldots\wedge U^-(w_{kn}), \tag{32}$$

$$\ldots$$

$$G_{ky}=U^+(w_{k1})\wedge U^+(w_{k2})\wedge\ldots\wedge U^-(w_{ki})\wedge\ldots\wedge U^-(w_{kn}), \tag{33}$$

$$\ldots$$

$$G_{kz}=U^-(w_{k1})\wedge U^-(w_{k2})\wedge\ldots\wedge U^-(w_{ki})\wedge\ldots\wedge U^-(w_{kn}). \tag{34}$$

Заключительным этапом вычисления величины S_k является дефаззификация:

$$S_k=\frac{G_{k1}Y_1+G_{k2}Y_2+\ldots+G_{ky}Y_y+\ldots+G_{kz}Y_z}{G_{k1}+G_{k2}+\ldots+G_{ky}+\ldots+G_{kz}}. \tag{35}$$

В результате выполнения нечеткого вывода каждый нейрон самоорганизующейся карты будет иметь единственный вес S_k. В таком случае, чтобы разбить нейроны на C кластеров и получить границы кластеров, можно применить алгоритм *k*-средних для одномерного пространства.

Шаг 1. Из множества нейронов выбираются C нейронов, веса которых будут служить начальными значениями центров кластеров. На начальном этапе каждый кластер будет содержать только один нейрон, вес которого выбран в качестве центра этого кластера.

Шаг 2. Выбирается нейрон номер $k=1$.

Шаг 3. Для выбранного нейрона номер k вычисляются меры близости его веса S_k к значениям центра каждого кластера по формуле:

$$\delta_{kc}=\left|S_c-S_k\right|, \tag{36}$$

где S_c – значение центра кластера номер c; величина c может принимать значения от 1 до C.

Шаг 4. Определяется минимальное значение δ_{kc}. Выбранный нейрон включается в состав кластера, для которого значение δ_{kc} является минимальным.

Шаг 5. Пересчитывается значение центра кластера, в состав которого был включен новый нейрон:

$$S_c=\frac{1}{L_c}\sum_{l=1}^{L_c}S_l, \tag{37}$$

где S_l – значение веса l-го нейрона кластера номер c; L_c – число нейронов, входящих в кластер номер c.

Шаг 6. Номер выбранного нейрона увеличивается на 1.

Шаг 7. Проверяется выполнение условия:

$$k \le N . \quad (38)$$

В случае выполнения данного условия осуществляется переход к шагу 3, в противном случае – к шагу 8.

Шаг 8. Сохраняются составы кластеров, полученные на текущей итерации. Сравниваются составы кластеров, полученные на текущей и предыдущей итерациях. Если составы не совпали, то осуществляется переход к шагу 2 и выполняется следующая итерация разбиения нейронов на кластеры. Если составы совпали, то рекомендуется принять решение о текущем разбиении нейронов на кластеры. Конец алгоритма.

Проведение экспериментальных исследований по оценке корректности обнаружения ресурсоемких запросов

Представленные выше алгоритмы реализованы в виде программного обеспечения для кластеризации SQL-запросов с использованием самоорганизующейся карты (SOM-кластеризации). Разработка программных средств выполнена на языке Python на основе тензорных вычислений с помощью средств библиотеки TensorFlow.

Проведенный анализ показал, что множество запросов, поступающих к базам данных, следует сгруппировать в пределах четырех кластеров:

– 1-й кластер – ресурсоемкие «тяжелые» запросы, которые в процессе выполнения характеризуются интенсивным использованием системной памяти, процессоров, дискового пространства и каналов вывода;

– 2-й кластер – ресурсоемкие «медленные» запросы, для которых характерны процедуры частого исполнения и перекомпиляций;

– 3-й кластер – случайные проблемные запросы, возникающие по причине случайной временной нехватки ресурсов для их выполнения из-за пиковых нагрузок в сети, на серверах, в процессорах, дефицита ресурсов во время выполнения других запросов;

– 4-й кластер – остальные не ресурсоемкие (не проблемные) запросы.

Для идентификации различных видов запросов проведено более 500 экспериментов. При этом использовались исходные данные, представленные в таблице 1.

В процессе экспериментов выполнялись алгоритмы поиска рекомендуемого значения радиуса топологической окрестности активных нейронов, настройки самоорганизующейся карты и последующего обнаружения ресурсоемких запросов, поступающих к базам данных. Полученные результаты были использованы для вычисления показателей корректности обнаружения ресурсоемких запросов.

Корректность обнаружения запросов к базам данных оценивалась с помощью двух показателей:

1) вероятности обнаружения ресурсоемких запросов к базам данных;

2) вероятности ошибочного обнаружения ресурсоемких запросов к базам данных.

Значение вероятности обнаружения ресурсоемких запросов к базам данных вычислялось по формуле:

$$P_{\text{det}} = \frac{Q_{\text{det}}}{Q_{\Sigma}}, \quad (39)$$

где Q_{det} – число обнаруженных ресурсоемких запросов, поступивших к базам данных; Q_{Σ} – число всех ресурсоемких запросов, поступивших к базам данных.

Таблица 1
Table 1

Исходные данные
Initial data

Величины	Значения
N	2500
n	5
α_1	0,1
$\Delta\alpha$	0,01
$\sigma_{\max}$	15
T	20
C	4

Для вычисления вероятности ошибочного обнаружения ресурсоемких запросов к базам данных использовалось выражение:

$$P_{\text{err}} = \frac{Q_{\text{err}}}{Q_{\text{det}}}, \quad (40)$$

где Q_{err} – число поступивших к базам данных ресурсоемких запросов, ошибочно отнесенных к ресурсоемким.

Результаты экспериментальных исследований по оцениванию корректности обнаружения ресурсоемких запросов с помощью SOM-кластеризации представлены в таблице 2.

Таблица 2
Table 2

Результаты экспериментов
Experimental results

Величины	Значения
Среднее значение $E_{\min}$	1,214
Среднее значение σ^*	7,802
P_{det}	0,951
P_{err}	0,082

Кроме того, проведены многочисленные эксперименты по обнаружению ресурсоемких запросов с применением современных средств мониторинга и оптимизации запросов SQL Tuning Advisor и Oracle Cost-Based Optimizer. Полученные результаты исследований представлены в таблице 3.

Анализ данных, представленных в таблице 3, показывает, что предложенные алгоритмы, которые реализованы в виде программного обеспечения SOM-кластеризации, позволяют повысить вероятности обнаружения ресурсоемких запросов к базам данных на 12,68 % – 15,27 % и снизить вероятность ошибочного обнаружения ресурсоемких запросов к базам данных на 12,20 % – 24,39 % по сравнению с применяемыми на практике средствами настройки и оптимизации планов выполнения SQL-запросов.

Таблица 3
Table 3

Результаты оценивания корректности обнаружения ресурсоемких запросов к базам данных
Results of evaluating the correctness of detection of resource-intensive database queries

Анализируемые средства	Значения P_{det}	Значения P_{err}
SQL Tuning Advisor	0,825	0,094
Oracle Cost-Based Optimizer	0,844	0,102
SOM-кластеризация	0,951	0,082

Заключение

Таким образом, в процессе обнаружения проблемных SQL-операторов предложено использовать до 9 значимых статистических параметров, характеризующих скорость обработки запроса, а также используемые для этого ресурсы. Исходный набор 32 анализируемых параметров, содержащихся в таблице DBA_HIST_SQLSTAT хранилища рабочей нагрузки СУБД Oracle, удалось сократить путем предварительного нормирования набора показателей с помощью сигмоидальной функции и последующего выбора конечного числа главных компонент на основе критерия Кеттелла. Кластеризация множества SQL-операторов на основе SOM дала возможность сформировать 4 основных подмножества запросов. При этом ресурсоёмкие запросы оказались сосредоточены в 2 кластерах. В результате данного исследования авторам удалось обосновать корректность использования сокращенного набора статистических параметров для обнаружения ресурсоемких запросов. Это позволило успешно выполнить нейросетевую кластеризацию анализируемых запросов.

При выполнении исследований получены результаты, обладающие научной новизной:

1. Разработан алгоритм обнаружения ресурсоемких запросов к базам данных, который отличается использованием нейронной самоорганизующейся карты для обработки векторов параметров запросов. Применение алгоритма позволяет повысить вероятность корректного обнаружения ресурсоемких запросов и снизить вероятность их ошибочного обнаружения.

2. Разработан алгоритм поиска рекомендуемого значения радиуса топологической окрестности активных нейронов в самоорганизующейся карте, который учитывает характеристики параметров запросов обучающей и контрольной выборки. Применение алгоритма позволяет избежать переобучения нейронов в процессе настройки самоорганизующейся карты, предназначенной для обнаружения ресурсоемких запросов к базам данных.

3. Разработан алгоритм разграничения кластеров самоорганизующейся карты, предназначенной для классификации запросов к базам данных. Новизна алгоритма состоит в использовании нечеткого вывода для вычисления обобщенных весов нейронов. Применение алгоритма позволяет обеспечить получение границ кластеров для последующего нейросетевого обнаружения ресурсоемких запросов к базам данных.

Выполнение экспериментов по оцениванию корректности обнаружения ресурсоемких запросов с помощью разработанных алгоритмов показало преимущество их использования по сравнению с другими специализированными программными средствами, что подтверждает целесообразность применения на практике результатов проведенных исследований.

References

1. Alghazali S.M.M., Polshchykov K., Hailan A.M., Svoykina L. 2021. Development of Intelligent Tools for Detecting Resource-intensive Database Queries. International Journal of Advanced Computer Science and Applications, 12(7): 32–36.

2. Bachhav A., Kharat V., Shelar M. 2021. An Efficient Query Optimizer with Materialized Intermediate Views in Distributed and Cloud Environment. Tehnički glasnik, 15: 105–111.

3. Belattar S., Abdoun O., El khatir H. 2020. New Learning Approach for Unsupervised Neural Networks Model with Application to Agriculture Field. International Journal of Advanced Computer Science and Applications, 11(5): 360–369.

4. Bodepudi H. 2020. Faster The Slow Running RDBMS Queries With Spark Framework. International Journal of Scientific and Research Publications, 10(11): 287–291.

5. Clovis L.R., Scapim C.A., Pinto R.J.B. et al. 2020. Yield stability analysis of maize hybrids using the self-organizing map of Kohonen. Euphytica, 216: 161.

6. Fattah S.M.A., Mahmoud M.A., Abd-Elmegid L.A.E. 2014. An Adaptive Hybrid Controller for DBMS Performance Tuning. International Journal of Advanced Computer Science and Applications, 5(5): 151–156.

7. Fernandez I. 2015. Beginning Oracle Database 12c Administration. From Novice to Professional. Apress: 384.

8. Kalmegh P., Babu S., Roy S. 2018. Analyzing Query Performance and Attributing Blame for Contentions in a Cluster Computing Framework. arXiv:1708.08435v2.

9. Lan H., Bao Z., Peng Y.A. 2021. Survey on Advancing the DBMS Query Optimizer: Cardinality Estimation, Cost Model, and Plan Enumeration. Data Sci. Eng., 6: 86–101.

10. Lekshmi B.G., Meyer-Wegener K. 2021. COPRAO: A Capability Aware Query Optimizer for Reconfigurable Near Data Processors. 2021 IEEE 37th International Conference on Data Engineering Workshops (ICDEW): 54-59.

11. Ma M., Yin Z., Zhang S. et al. 2020. Diagnosing Root Causes of Intermittent Slow Queries in Cloud Databases. PVLDB, 13(8): 1176–1189.

12. Miao Z., Chen T., Bendeck A. et al. 2020. I-Rex: an interactive relational query explainer for SQL. Proc. VLDB Endow, 13: 2997–3000.

13. Muniswamaiah M., Agerwala T., Tappert C.C. 2020. Approximate Query Processing for Big Data in Heterogeneous Databases. 2020 IEEE International Conference on Big Data: 5765–5767.

14. Polshchykov K.A., Lazarev S.A., Konstantinov I.S., Polshchykova O.N., Svoikina L.F., Igityan E.V., Balakshin M.S. 2020. Assessing the Efficiency of Robot Communication. Russian Engineering Research, 40: 936–938.

15. Polshchykov K., Lazarev S., Polshchykova O., Igityan E. 2019. The Algorithm for Decision-Making Supporting on the Selection of Processing Means for Big Arrays of Natural Language Data. Lobachevskii Journal of Mathematics, 40 (11): 1831–1836.

16. Polshchykov K.O., Lazarev S.A., Zdorovtsov A.D. 2017. Neuro-Fuzzy Control of Data Sending in a Mobile Ad Hoc Network. Journal of Fundamental and Applied Sciences, 9(2S): 1494–1501.

17. Polshchykov K.O., Zdorenko Y.M., Masesov M.O. 2014. Method of telecommunications channel throughput distribution based on linear programming and neuro fuzzy predicting. Elixir International Journal. Network Engineering, 75: 27327–27334.

18. Sakkari M., Zaied M. 2020. A convolutional deep self-organizing map feature extraction for machine learning. Multimedia Tools and Applications, 79: 19451–19470.

19. Sikdar S., Jermaine C. 2020. MONSOON: Multi-Step Optimization and Execution of Queries with Partially Obscured Predicates. Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD '20): 225–240.

20. Sinha S., Singh T.N., Singh V.K., Verma A.K. 2010. Epoch determination for neural network by self-organized map (SOM). Computational Geosciences, 14: 199–206.

21. Wang C., Cheungc A., Bodik R. 2017. Synthesizing highly expressive SQL queries from input-output examples. Proceedings of the 38th ACM SIGPLAN Conf. on Programming Language Design and Implementation: 452–466.

22. Zhou X., Ordonez C. 2020. Matrix Multiplication with SQL Queries for Graph Analytics. 2020 IEEE International Conference on Big Data (Big Data): 5872–5873.

Конфликт интересов: о потенциальном конфликте интересов не сообщалось.
Conflict of interest: no potential conflict of interest related to this article was reported.

ИНФОРМАЦИЯ ОБ АВТОРАХ

Хайлан Ахмад, PhD, преподаватель кафедры компьютерных наук и математики, Университет Ти-Кар, г. Ти-Кар, Ирак

Польщиков Константин Александрович, доктор технических наук, доцент, директор института инженерных и цифровых технологий Белгородского государственного национального исследовательского университета, г. Белгород, Россия

Алгазали Салах Махди Мадлол, соискатель кафедры прикладной информатики и информационных технологий Белгородского государственного национального исследовательского университета, г. Белгород, Россия

INFORMATION ABOUT THE AUTHORS

Hailan Ahmad, PhD, lecturer of Department of Computer Science and Mathematics, Ti-Kar University, Ti-Kar, Iraq

Konstantin A. Polshchikov, Doctor of Technical Sciences, Associate Professor, Director of the Institute of Engineering and Digital Technologies of the Belgorod State National Research University, Belgorod, Russia

Alghazali Salach, postgraduate of the Department of Applied Informatics and Information Technologies of the Belgorod State National Research University, Belgorod, Russia