Инженерный вестник Дона, №4 (2017)
ivdon.ru/ru/magazine/archive/n4y2017/4511

Совершенствование процесса поиска неэффективных SQL-запросов в СУБД Oracle

С.М.М. Алгазали, В.Г.Айвазов, А.В. Кузнецова

Южно-Российский государственный политехнический университет (Новочеркасский политехнический институт) имени М.И. Платова

Аннотация: Рассматриваются пути решения проблемы поиска неэффективных SQL-запросов в больших системах при отсутствии явных причин падения производительности. Для облегчения и ускорения поиска проблемных запросов предлагается осуществлять их предварительную кластеризацию. Каждый SQL-запрос описывается тремя группами статистических параметров, которые можно получить при помощи системных средств СУБД: параметрами выбранного плана исполнения запроса, параметрами плана реального исполнения, характеристиками среды исполнения. Сформулирована задача кластеризации с учётом интеллектуального формирования параметрической модели SQL-запроса и возможностью начального формирования подмножества кластеров, содержащих заранее известные неэффективные запросы-образцы. Предлагаемое решение может быть распространено на другие СУБД, с учётом их особенностей.
Ключевые слова: SQL, запрос, СУБД, Oracle, неэффективный SQL-запрос, кластеризация, группировка.

Введение. Обычно информация о том, что имеют место проблемы с производительностью программных систем промышленного или коммерческого уровня исходит от пользователей и эта информация представляет собой субъективную оценку, выраженную в свободной форме. В качестве пользователей системы могут выступать как конечные пользователи (англ., End-User – EU), так и разработчики БД и ПО (англ., Data Base Developer – DBD, Software Developer – SD). Бòльшую часть претензий составляют жалобы на длительное исполнение запросов. Проблемы могут проявляться «зависанием» форм конечных приложений, увеличением длительности исполнения неинтерактивных задач, скриптов или программ, работающих с БД. Более явным и серьёзным сигналом могут служить ошибки времени исполнения, появлении которых может сообщаться через графический интерфейс конечных приложений, приложений разработки, консоль СУБД, индивидуальные лог-файлы задач, системные лог-файлы и

т.д. Наличие таких ошибок возможно только при крайнем ухудшении производительности – они сигнализируют об аварийном завершении запроса при невозможности продолжения его исполнения. При этом завершается с ошибкой не всегда тот запрос, который является причиной ошибки, запрос с плохой производительностью или высоким потреблением ресурсов.

Для устранения потенциально-неэффективных SQL-запросов и причин их появления специалистам DBD и SD требуется помощь со стороны от администраторов СУБД (англ., Data Base Administrator – DBA), которые имеют доступ к частной и агрегированной информации об исполнении запросов и состоянии среды. По результатам анализа DBA проводят мероприятия по оптимизации производительности, представляя пользователям отчёты о сделанных изменениях, либо дают рекомендации по действиям, которые необходимо произвести со стороны разработчиков.

Падение производительности может иметь как *случайный*, так и *системный* характер. Причинами случайного характера падения производительности могут служить ситуации, при которых 1) SQL-запросы INSERT, UPDATE ожидают окончания обработки других запросов и освобождения от блокировки тех данных, которые необходимы для собственного исполнения; 2) время исполнения запросов SELECT резко увеличивается из-за мгновенной нагрузки на сервере; 3) имеет место иная занятость дисковой системы сервера. Признаком такой случайности является однократное повторение явления при сравнительно небольшом количестве обрабатываемых данных. Причиной же системных проблем падения производительности являются: неоптимальный код SQL-скриптов, неудачная модель БД и/или её конфигурации, состояние и характеристики среды исполнения, а также устаревание статистической информации о распределении данных [1, 2].

В первом случае специалистам DBA необходимо определить, что является причиной длительных блокировок, а во втором – изучать планы исполнения запросов и статистические данные (далее просто статистику) результатов реального выполнения большого числа SQL-запросов. И чем крупнее программная система, тем больше кандидатов для изучения. За редким исключением неэффективность запросов прямо бросается в глаза. Такие очевидные показатели как высокая стоимость плана исполнения SQL-запроса, длительное время его исполнения, аварийное завершение с определёнными кодами ошибок далеко не всегда являются прямыми указателями на источник проблем. Определить эффективность плана запроса «на глаз», не имея представлений о задаче, структуре данных и настройках оптимизатора, практически невозможно. Не говоря уж о том, что просто не бывает запросов, одинаково эффективных во всех случаях, точнее даже не столько запросов, сколько их планов. В ряде случаев со второй задачей DBD и SD могут справиться самостоятельно, прибегая к помощи специализированных утилит и имея соответствующий уровень доступа.

Объём получаемой информации об одном запросе достаточно велик и составляет более полусотни числовых параметров. Количество запросов в больших программных системах составляет уже десятки тысяч, поэтому поиск неэффективных запросов, особенно при отсутствии явных признаков в работе программной системы, является далеко нетривиальной задачей. Для её решения следует произвести оптимальный (с точки зрения количества и значимости) отбор параметров и предварительную группировку запросов на основе методов разведочного и интеллектуального анализа данных.

Цель. Проблема поиска потенциально-неэффективных запросов приобретает всё бо̀льшую остроту, требуя изучения такого объёма информации, с которой человек справляется с трудом. Поэтому целесообразно предварительно локализовать те SQL-запросы, которые

требуют пристального изучения, а потом уже проводить их первоочередной анализ или анализ наиболее типичных представителей выделенного множества. Целью работы является повышение эффективности поиска проблемных SQL-запросов за счёт их предварительной группировки. При этом вектор признаков, описывающий SQL-запрос, формируется из параметров трех типов: оценок плана исполнения запроса, статистических параметров реального исполнения запроса, параметров среды и условий, в которых выполнялся запрос. Формирование вектора признаков должно основываться на интеллектуальных процедурах анализа состояния среды и условий, в которых исполняются запросы, чтобы обеспечить формирование оптимального набора параметров.

В качестве базовой выбрана СУБД Oracle как лидер среди серверов баз данных на платформах UNIX и Windows.

Материалы и объекты исследования. Под неэффективным SQL-запросом будем понимать запрос, приводящий к ощутимому замедлению работы клиентских приложений или сервера базы данных и потребляющий при этом значительные ресурсы дискового пространства, машинного времени, и оперативной памяти.

Работа запроса в любой СУБД состоит из следующих этапов: синтаксического разбора SQL-выражения, его семантической проверки, выработки плана запроса; исполнения запроса по выработанному плану. Соответственно и инструментальные средства, позволяющие получить статистические данные о запросах, можно разделить на две группы: утилиты, позволяющие получить предполагаемый план выполнения запроса; средства, позволяющие получить информацию о реальном исполнении запроса.

План SQL-запроса – это упорядоченный набор шагов, согласно которому СУБД будет осуществлять выборку данных. План представляет собой древовидную структуру, содержащую порядок шагов и связь между ними.

Зачастую возможно большое количество различных путей обработки данных, ведущих к одному и тому же результирующему набору данных, поэтому на основе лексического и синтаксического анализа текста запроса специальный компонент СУБД – оптимизатор – вырабатывает множество планов. В новых версиях СУБД Oracle по умолчанию используется современный оптимизатор «Cost based optimiser» (англ., CBO), в противовес устаревшему «Rule based optimiser» (англ., RBO). В результате преобразований и последующей оценки CBO выбирает наиболее подходящий план.

Основными показателями, отражающими эффективность плана исполнения запроса, являются *стоимость выполнения* (Cost) и *мощность* (Cardinality), которые представляют собой приблизительные оценки объёма работ по извлечению данных и количества возвращаемых строк соответственно [3]. Эти оценки вычисляются для каждой операции, входящей в план исполнения, и впоследствии суммируются для получения оценки производительности всего плана исполнения. Грубо говоря, эффективность запроса обратно пропорциональна суммарной величине этих параметров. Стоит отметить, что эти оценки далеко не всегда соответствуют реальным характеристикам производительности, получаемым при исполнении запроса. Мощность вычисляется с использованием информации о статистическом распределении данных в таблицах-источниках для SQL-выражений. Рассчитанная величина может быть не точной в связи с тем, что при изменении данных в БД информацию об их распределении невозможно поддерживать в полностью актуальном состоянии. Особенно это касается больших таблиц, в которых для сбора статистической информации используется только определённая доля данных. И эта доля может быть недостаточно репрезентативной. В большинстве случаев достаточно, чтобы мощность не отличалась от реального значения больше, чем на один и даже два порядка. Стоимость запроса складывается из стоимости отдельных

операций. К точности её расчёта предъявляются похожие требования. Поскольку оценка планов исполнения нужна оптимизатору для выбора наиболее эффективного способа исполнения одного и того же запроса, то его абсолютное значение не может напрямую использоваться для оценки реальной эффективности [4, 5].

Помимо вышеуказанных показателей для оценки SQL-запроса используются и другие параметры плана исполнения. Это: *стоимость ресурсов центрального процессора* (CPU Cost), *стоимость ввода-вывода* (IO Cost), показатель *использования дискового пространства* (Temp Space**).**

Параметр CPU Cost позволяет оценивать количество машинных циклов, необходимых для выполнения всех операций запроса и напрямую зависит от текущей загрузки сервера. Эта величина не является значимой для оценки эффективности запроса если экземпляр запущенной СУБД – инстанс (англ., instance) – не использует чрезмерные ресурсы центрального процессора. Специалисты Oracle рекомендуют учитывать оценку CPU Cost для тех планов исполнения, для которых в качестве цели оптимизации было задано наискорейшее получение всех строк результата запроса (хинт ALL_ROWS в SQL-выражении или параметр сессии ALL_ROWS). В случае, когда целью оптимизации является наискорейшее получение первых строк результата (хинт или режим оптимизатора FIRST_ROWS), CPU Cost теряет актуальность. В настоящее время CBO не располагает реальной информацией о нагрузке ЦПУ, поэтому величина CPU Cost имеет является приблизительной оценкой плана. Величина параметра IO Cost пропорциональна количеству дисковых (физических) чтений блоков данных, требующихся для выполнения всех операций плана. Однако CBO пока ещё не имеет предварительной информации о доле блоков таблиц, находящихся в буферном кэше, а значит не может достоверно различать затраты логическое и физическое чтение. Значение Temp Space свидетельствует о необходимости

использования дисковой памяти для проведения сортировок, группировок, сортировок объединений данных, что косвенно указывает на неэффективность SQL-запроса.

По мнению ряда специалистов-практиков эти и другие параметры плана исполнения являются отправными точками для поиска неэффективных запросов и в совокупности позволяют оценить длительность запроса, его ресурсоёмкость и, в конечном счёте, успешность исполнения. Получить предполагаемый план выполнения специалисты DDB, DS и DBA могут с помощью специальной команды explain plan for, предварительно создав специальную системную таблицу PLAN_TABLE. Для получения наиболее читабельного представления существуют утилиты Toad, SQL Navigator, PL/SQL Developer и др.

План, предлагаемый оптимизатором в Oracle, конечно же, зависит от текста запроса, но отнюдь не только от него, в особенности в CBO. Как упоминалось ранее, важную роль в расчёте параметров плана этим оптимизатором играет так называемая *статистика распределения* значений в столбцах таблиц, которые используются для задания условий выборки и операций соединения. Статистика распределений необходима для приблизительной оценки результирующего количества строк (мощности) и, соответственно, количества блоков данных, которые нужно прочесть и обработать. Если статистика распределения устаревает и не учитывает изменений, произошедших с данными после её сбора, имеют место несоответствия вычисленных параметров плана реальному количеству дисковых чтений, количеству места для временного хранения рабочих данных, количеству процессов для параллельного исполнения, времени существования блокировок для данных, а, следовательно, и времени исполнения запроса. Другими словами имеет место рост общей погрешности оценки планов исполнения и сам план перестаёт быть достоверной моделью

SQL-запроса. На достоверность отображения SQL-запроса выбранным для него планом исполнения накладывает отпечаток и несовершенство алгоритмов оптимизатора. CBO – программа, написанная исходя из общих предположений, а БД имеет конкретное наполнение, и поэтому вовсе не факт, что значения параметров плана точно соответствуют реальным характеристикам запроса, который исполняется в динамически изменяющейся среде.

Более точным вектором оценки SQL-запроса по сравнению с рядом параметров плана исполнения является *статистика производительности*, которую СУБД собирает во время исполнения запроса. Уровень детализации и подробности этой статистики зависят от настроек системы и от использования режима отладки (трассировка) для исполняющихся запросов. Некоторые значения статистики доступны только при включённой трассировке, но и без неё по умолчанию для каждого запроса собирается базовая информация о ходе его исполнения. Включение трассировки не всегда возможно, хоть и позволяет более подробно изучить проблему производительности запроса. В тех случаях, когда нельзя изолировать определённый запрос, т. е. получить его текст и исполнить его вне приложения в специальной среде разработки или консоли, трассировку можно включить только для приложения в целом. Такой подход вносит дополнительные накладки с точки зрения производительности приложения и последующего анализа полученных объёмов информации. В связи с этим для получения наиболее достоверных параметров SQL-запроса авторами предлагается сделать упор на базовую статистику. Её можно получить как с помощью системных запросов, так и с помощью вспомогательных утилит. И те и другие средства, как правило, находятся в распоряжении DBA.

Управляющие структуры, справочную информацию, буферы, кеш, области динамического выделения памяти для хранения временных структур

различного типа, и другие данные СУБД хранит в разделяемой памяти (англ., System Global Area, SGA), доступной как для системных, так и серверных процессов каждого инстанса [6]. Для удобства разработки, администрирования, анализа производительности и отладки СУБД Oracle предоставляет доступ к информации в SGA при помощи специальных реляционных объектов базы данных, хотя сами структуры памяти имеют не реляционную природу – это могут быть хеш-таблицы, индексы, списки, кучи и т.д. Эти объекты, называемые *фиксированными базовыми таблицами* (англ., fixed base tables, FBT), доступны на чтение для пользователей с определёнными привилегиями. Но любое изменение их содержимого или структуры невозможно. Кроме того, правила консистентности и целостности данных FBT не гарантируются и никакие транзакционные блокировки к ним не применимы. В связи с вре́менной природой структур, находящихся в оперативной памяти, информация доступная через эти фиксированные таблицы, на диск вместе с другими данными БД не сохраняется. Она может существовать только с момента старта инстанса и до момента его остановки.
Несмотря на наличие абстракции, позволяющей работать со структурами оперативной памяти инстанса при помощи SQL, эти таблицы не удобны в использовании. Для облегчения работы с FBT в СУБД Oracle существуют так называемые *представления*, определённые на этих таблицах и называемые *динамическими представлениями производительности* (англ., *dynamic performance views*, DPV). Через эти представления DBA может получить доступ к актуальной и хорошо детализированной информации для исследования большинства проблем производительности и возможности её улучшения [6].

Основными динамическими представлениями производительности СУБД Oracle, которые могут представлять интерес для создания параметрической модели SQL-запроса, являются:

– V$SESSION – содержит информацию обо всех сессиях работы с СУБД Oracle, открытых на данный момент;

– V$ACTIVE_SESSION_HISTORY – содержит историческую информацию об активных, не находящихся в состоянии простоя сессиях, периодически (по умолчанию с интервалом в 1 секунду) собираемую с момента старта инстанса. По структуре схожа с V$SESSION, с добавлением дополнительных полей, например, *sample_time,* указывающего на время производства «снимка» информации, отображаемой в V$SESSION;

– V$SQL – содержит информацию об обрабатываемых на данный момент SQL выражениях и всех объектах, связанных с ними: родительских и дочерних курсорах, планах исполнения, рабочих областей памяти для каждой операции определённых планов исполнения, статистики исполнения и т.д. В ряде случаев через это представление можно получить и информацию об SQL-выражениях, обрабатывавшихся в недавнем прошлом;

– V$SQLSTATS_PLAN_HASH – содержит базовую статистику для каждого плана исполнения каждого SQL-выражения. Производная от V$SQL;

– V$SQLAREA_PLAN_HASH – содержит информацию о разделяемых областях памяти для SQL-выражений. Производная от V$SQL;

– V$SQLTEXT – содержит полные тексты SQL-выражений;

– V$SQL_PLAN – содержит информацию обо всех планах исполнения, сгенерированных для исполнения SQL-выражений.

Не смотря на свою подробность и актуальность, все эти представления ограничены в размерах и вся информация, которая становится не нужной для работы СУБД (устаревает), очень быстро освобождается для последующего использования. В целях длительного сохранения диагностической информации и статистики производительности, для преодоления временны́х ограничений в СУБД Oracle существует специальное хранилище данных (англ., Advanced Workload Repository, AWR) [6]. В это хранилище

определённой периодичностью, по умолчанию равной одному часу, записывается предварительно отобранная и обработанная информация из динамических представлений производительности и фиксированных базовых таблиц. Доступ к AWR, в частности, можно получить из утилит Toad, SQL Developer, SQL*Plus и других средств работы с СУБД, но только при наличии необходимых привилегий.

С определённой периодичностью, по умолчанию равной одному часу, СУБД Oracle запускает специальную задачу, которая делает «снимок» (англ. snapshot) – собирает всю появившуюся и изменившуюся за период информацию из DPV и FBT; вычисляет разницу в кумулятивных счётчиках статистики; отбирает по определённым критериям (которые не раскрываются в официальной документации Oracle, хотя и понятны интуитивно) данные о сессиях, SQL-выражениях и других объектах времени исполнения записывает результаты в таблицы базы данных, входящие в состав AWR. Все данные, относящиеся к одному AWR-снимку, помечаются одним идентификационным номером, который затем можно получить из поля *snap_id* любой таблицы, относящейся к AWR. Помимо этого в таблицах AWR хранятся данные со всех вычислительных узлов кластера, обслуживающего отдельную базу данных. Для различия между ними используется порядковый номер узла в поле *instance_number*, а при настройке AWR для хранения данных с нескольких различных баз данных (окружений) и для различия между данными первичной и клонированной системы используется поле *dbid* – уникальный идентификационный номер БД. По умолчанию информация в AWR хранится в течение одной недели.

В AWR, кроме всего прочего, входит ряд таблиц, содержащих данные, аналогичные тем, которые формируются в упомянутых динамических представлениях производительности. Эти таблицы и предлагается использоваться в качестве источника информации для формирования

параметров исследуемых объектов. Таблица DBA_HIST_SQL_PLAN содержит информацию о структуре планов исполнения запросов и об оценках производительности, выданных оптимизатором для каждого шага плана исполнения, в том числе, и для исторических планов исполнения, которых на данный момент нет в памяти СУБД. Таблица DBA_HIST_SNAPSHOT содержит информацию о самих AWR-снимках. По этой информации можно определить время начала и окончания интервала для снимка, время старта инстанса и узнать дополнительную информацию об AWR-снимке. Таблица DBA_HIST_SQLTEXT содержит полные тексты SQL-выражений. Основной интерес в вышеперечисленных таблицах представляют столбцы, содержащие непосредственные данные об исполнении запросов. Наибольшее количество такой информации располагается в таблице DBA_HIST_SQLSTAT. Не смотря на то, что некоторые данные можно найти в DBA_HIST_ACTIVE_SESS_HISTORY, эта таблица преимущественно содержит данные не об отдельных запросах, а о сессиях базы данных и получение данных об SQL-запросах сопряжено с определёнными сложностями. Последняя таблица остается полезной для получения дополнительной, справочной информации о сессиях: с какого удалённого узла была инициализирована эта сессия; от имени какого пользователя, какая программа на удалённом сетевом узле инициализировала эту сессию; временны́е метки, по которым можно определить время активности; временно́й интервал, в котором существовала эта сессия. Вся эта информация оказывает существенную помощь в исследовании проблем производительности и может служить источником дополнительных параметров для SQL-запросов.

В таблице DBA_HIST_SQLSTAT статистические данные хранятся в двух представлениях: в суммарном для отдельного запроса и в виде изменения за интервал. В имени столбцов с данными первого типа и второго

типов присутствую соответственно суффиксы TOTAL и DELTA. Анализ статистических данных с суффиксом TOTAL в имени показал, что эти данные не пригодны для решения задачи предварительной группировки SQL-запросов по причине того, что эта статистика всё же иногда сбрасывается. Скорее всего, это происходит из-за устаревания курсора для запроса, но и однозначно – при перезапуске инстанса. В связи с этим для анализа предлагается использовать данные из аналогичных DELTA-столбцов. Они будут группироваться для каждого уникального сочетания sql_id и plan_hash_value, однозначно идентифицирующего SQL-запрос в Oracle, и некоторым образом агрегироваться. Число базовых статистических параметров, представляющих интерес для исследования производительности, составляет около четырёх десятков. Оперируя только этим набором, можно составлять различные комбинации, а значит и различные модели SQL-запроса, позволяющие делать акцент на разных аспектах его работы. Кроме того, в зависимости от конфигурации СУБД и особенностей исполняющихся SQL-выражений Oracle включает так называемое слежение (англ. *monitoring*) и собирает в указанных таблицах *дополнительные статистики.* Например, это касается запросов, исполняющихся более 5 секунд, распараллеленных запросов, запросов, явно выбранных пользователем.

Удаляя из рассмотрения номер инстанса из поля *instance_number,* можно оценивать производительность запросов на всей системе в целом, без учёта количества отдельных вычислительных узлов, входящих в один кластер. Учёт этого значения, напротив, позволит анализировать производительность одного или нескольких экземпляров запущенной СУБД.

Объём выборки для анализа может содержать десятки, а то и сотни тысяч единиц информации в зависимости от сложности задачи, стоящей перед DBA. Кроме того, для использования этой информации необходимо знание схемы данных AWR, назначений полей в её таблицах, особенностей

извлечения необходимой информации. Получение и обработка такого количества данных без применения статистических и/или интеллектуальных методов трудна и малоэффективна [7]. Для повышения качества технологии анализа производительности БД авторами предлагается проведение предварительного анализа и группировки (кластер-анализ) SQL-запросов на основе трёх групп параметров: параметров плана исполнения SQL-запросов, статистических данных о результатах исполнения запросов и характеристик среды, в рамках которой выполнялись запросы. При этом в процессе кластеризации будут использоваться не все, а только часть параметров каждой группы.

Формальная постановка задачи исследования. Пусть имеется множество SQL-запросов $R = \{r_1, r_2, \ldots, r_{NR}\}$, множество номеров (меток) кластеров $G = \{g_1, g_2, \ldots, g_{N_G}\}$. Каждый запрос r_i характеризуется набором трёх множеств предварительно пронормированных и (при необходимости) взвешенных количественных параметров: параметров плана исполнения $P^S = \{p_1^S, p_2^S, ..., p_{N_S}^S\}$, параметров производительности $P^C = \{p_1^C, p_2^C, ..., p_{NC}^C\}$, параметров среды $P^E = \{p_1^E, p_2^E, ..., p_{N_E}^E\}$: $R = P^S ? P^C ? P^E$. При этом параметры производительности P^C и параметры плана исполнения P^S имеют характер неопределённости, связанный, соответственно, со случайным состоянием среды исполнения запроса и настройками оптимизатора. Задана симметричная функция, характеризующая степень близости между объектами $\rho(r_i, r_j)$.

Требуется построить отображение $F = R \rightarrow G$ такое, чтобы на каждый входной элемент $r_i \in R$, определяемый подмножествами параметров P'^S, P'^C и P'^E из множеств P^S, P^C и P^E, формировался единственный элемент $g_j \in G$. Состав подмножеств P'^C, P'^S определяется составом подмножества P'^E, а он, в

свою очередь, зависит от характера проблем в работе программной системы, настроек оптимизатора, сетевых нагрузок и активности пользователей.

Конечным результатом работы искомого отображения F на всём множестве R является формирование непересекающихся подмножеств, называемых кластерами так, чтобы каждый кластер состоял из SQL-запросов, степень близости между которыми больше некоего порогового значения ρ. Соответственно, для запросов из разных кластеров степень близости должна быть существенно ниже ρ, а полученное разбиение должно удовлетворять заданному функционалу качества. Отображение F должно допускать эффективную компьютерную реализацию, т.е. представлять собой алгоритм.

Задача построения отображения F включает в себя анализ и выбор метода кластеризации исходя из особенностей предметной области, определение оптимальных доверительных интервалов его параметров, выбор функции для определения меры близости объектов и кластеров (расстояния), выбор одного или нескольких функционалов качества кластеризации [8]. Дополнением к данной постановке задачи кластеризации является необходимость уточнения общего числа кластеров N_G, которое требует привлечения дополнительных процедур либо, решения поставленной задачи для нескольких значений N_G с последующим выбором наиболее адекватного результата кластеризации [8, 9].

При реализации отображения F предлагается предусмотреть возможность принудительного формирования некоторого подмножества кластеров, содержащих заранее известные запросы-образцы, относящиеся к разряду неэффективных. Это позволит улучшить качество кластеризации и гарантированно отыскать запросы со сходными признаками среди десятков тысяч запросов сессии. Задачи кластеризации, в которых незначительная часть объектов распределена по классам, называются задачами с частичным обучением (англ., semisupervised learning). Из-за малого числа

идентифицированных объектов такие задачи не переходят в разряд классификационных, но и для их решения необходима модификация известных методов кластеризации.

Заключение. Кластеризация близких по совокупности параметров SQL-запросов позволит ускорить поиск неэффективных экземпляров, за счёт анализа наиболее «типичных представителей» каждой группы и, в первую очередь, анализа запросов, попавших в кластеры с заранее известными проблемными запросами. Предлагаемый подход предусматривает отказ от использования хэш-значений запросов и планов исполнения запросов при формировании кластеров, и основывается на решении задачи с точки зрения совокупного влияния параметров SQL-запроса на общую производительность. Положения, описанные в работе, могут быть распространены и на другие промышленные СУБД со своими алгоритмами создания планов, своими наборами статистик и способами их получения.

Литература

1. Михеичев В. Причины неэффективности SQLзапросов в Oracle. Оптимизация производительности SQLзапросов // Системный администратор. 2015. №6. С. 47-51.

2. Михеичев В. Опыт и рекомендации по оптимизации SQLзапросов // Интернетжурнал «FORS», 2013, URL: fors.ru/upload/magazine/07/index.html

3. Using EXPLAIN PLAN // Oracle Help Center URL: docs.oracle.com/cd/B19306_01/server.102/b14211/ex_plan.htm#g42231

4. Lewis J. Cost-based Oracle fundamentals. Berkeley, CA: Apress, 2006. – 506 p.

5. Fritchey G. SQL Server Execution Plans. Publishing house: Red Gate Books, 2012. – 205 p.

6. Automatic Performance Statistics // Oracle Help Center URL: docs.oracle.com/cd/B19306_01/server.102/b14211/ex_plan.htm#g42231

7. Миллсап К., Хольт Д. Oracle. Оптимизация производительности. Пер. с англ. СПб: Символ_Плюс, 2006. – 464 с.

8. Jadidinejad Amir H., Sadr Hossein Improving Weak Queries using Local Cluster Analysis as a Preliminary Framework Indian Journal of Science and Technology, Vol 8(15), July 2015 – URL: i-scholar.in/index.php/indjst/article/view/75325

9. Крашенинников А.М., Гданский Н.И., Рысин М.Л. Построение сложных классификаторов для объектов в многомерных пространствах // Инженерный вестник Дона, 2012, №3 URL: ivdon.ru/magazine/archive/n2y2013/1611

10. Гданский Н.И., Рысин М.Л., Крашенинников А.М. Линейная классификация объектов с использованием нормальных гиперплоскостей. Инженерный вестник Дона, 2012, №4. URL: ivdon.ru/magazine/archive/n4ply2012/1324

References

1. Mihei4ev V. Cistemnyj administrator. 2015. №6. pp. 47-51.

2. Mihei4ev V. Internetzurnal «FORS», 2013. URL: fors.ru/upload/magazine/07/index.html

3. Using EXPLAIN PLAN. Oracle Help Center URL: docs.oracle.com/cd/B19306_01/server.102/b14211/ex_plan.htm#g42231

4. Lewis J. Cost-based oracle fundamentals. Berkeley, CA: Apress, 2006. 506 p.

5. Fritchey G. SQL Server Execution Plans. Publishing house: Red Gate Books, 2012. 205 p.

6. Automatic Performance Statistics. Oracle Help Center URL: docs.oracle.com/cd/B19306_01/server.102/b14211/ex_plan.htm#g42231

7. Millsap K., Hol't D. Oracle. Optimizacija proizvoditel'nosti [Optimizing Performance]. SPb: Simvol_Pljus, 2006. 464 p.

8. Jadidinejad A. H., Sadr H. Indian Journal of Science and Technology, 2015 – URL: i-scholar.in/index.php/indjst/article/view/75325

9. Gdanskij N.I., Karpov A.V., Bugaenko A.A. Inzenernyj vestnik Dona (Rus), 2012, №3. URL: ivdon.ru/magazine/archive/n3y2012/936

10. Gdanskij N.I., Rysin M.L., Krasheninnikov A.M. Inzenernyj vestnik Dona (Rus), 2012, №4. URL: ivdon.ru/magazine/archive/n4ply2012/1324