А.В. СМИРНОВ

Проблема оптимизации запросов в расширяемой СУБД SciDB

УДК 004.657

Муромский институт (филиал) ФГБОУ ВПО «Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых», г. Муром

В статье рассматриваются проблемы оптимизации запросов в современной пользовательски-расширяемой СУБД.

The article discusses problems of query optimization in modern user extensible DBMS.

Возросший интерес, а также доступность технологических средств в последнее десятилетие привел к росту исследований и множеству различных имплементаций NoSQL-СУБД (англ. not only SQL, не только SQL) [1].

NoSQL в информатике — термин, обозначающий ряд подходов, проектов, направленных на реализацию моделей баз данных, имеющих существенные отличия от используемых в традиционных реляционных СУБД с доступом к данным средствами языка SQL. Описание схемы данных в случае использования NoSQL-решений может осуществляться через использование различных структур данных: хеш-таблиц, деревьев и других.

Основная идея при разработке такого типа систем — предоставление разработчикам более гибких средств по модификации схемы данных на этапе эксплуатации, отказе от транзакционного контроля в пользу более естественной поддержки распределённых вычислений.

Использование NoSQL резко выросло в 2009 году, благодаря увлечению количества компаний связанных с использованием больших объемов данных. Появилось большое количество систем позволяющих организовывать и прозрачно поддерживать огромные массивы данных, обрабатывать и контролировать эти данные.

Один из ярких примеров использования NoSQL — СУБД для научных исследований. Огромное количество данных постоянно собирается из окружающей среды с помощью сенсоров различного типа, присоединенных к таким устройствам, как спутники и микроскопы, или генерируются искусственно с помощью высокоточных научных и инженерных симуляций. Например телескоп LSST, планирует собирать до 30 ТБ необработанных данных за ночь, а в процессе всей работы планируется собрать несколько петабайт необработанных и обработанных данных в год в течении 10 лет, что по приблизительным оценкам составит около 90 петабайт данных. Причем сырые данные планируется хранить постоянно. Анализ таких данных — ключ к лучшему пониманию физических явлений. Подобный анализ становится все более обычным во многих областях научных исследованиях.

Эффективный анализ и запросы к столь большим базам данных требует высокоэффективных многомерных индексированных структур и специфичных к областям применения техник обработки и агрегации [2].

Ключевой момент в обработке столько огромных баз данных — эффективные алгоритмы для статистического анализа. Разработка СУБД использующей новые подходы к обработке данных, новые модели данных, масштабируемость, требует отказа или как минимум пересмотра традиционных используемых алгоритмов, средств и технологий. Оптимизатор запросов, как ключевой узел обработки запросов не исключение в данном случае. Учитывая колоссальный размер обрабатываемых данных, качество конечного запроса может влиять на время его выполнения радикально.

SciDB представляет собой расширяемую высокопроизводительную распределенную многомерную СУБД для обработки научных данных петабайтных размеров. Под многомерностью понимается то, что все данные в СУБД организованы в виде многомерных массивов, ячейки которых содержат атрибуты, непосредственно содержащие данные. Подобная модель позволяет решать множество исследовательских задач, например выявление на снимках скопления объектов, определение их характеристик, скорости перемещения и различные аномальные явления [3]. Оптимизация в SciDB происходит достаточно примитивным и простым способом, осно-

ванным на правилах преобразований, что позволяет сократить стоимость выполнения с помощью спускания предикатов в запросе и перестановке операторов перераспределения. Однако потенциал оптимизируемой запросов теоретически намного больше, в частности сейчас не учитываются пользовательские операторы, которые составляют большую часть запросов. Отсутствие на данный момент развитого оптимизатора запросов в SciDB, а также уникальность этой СУБД предоставляет возможность попытаться решить научную задачу.

Целью данной статьи является анализ варианта проектирования оптимизатора запросов новой СУБД, являющейся типичным представителем NoSQL, в качестве которого была взята современная развивающаяся СУБД SciDB, а также выявление основных проблем.

Большинство СУБД предоставляет доступ к своим данным через высокоуровневый интерфейс в виде, какого либо языка запросов. Непосредственное «вычисление» запроса проходит через 2 ключевых компонента СУБД: оптимизатор и подсистему выполнения (рис. 1).

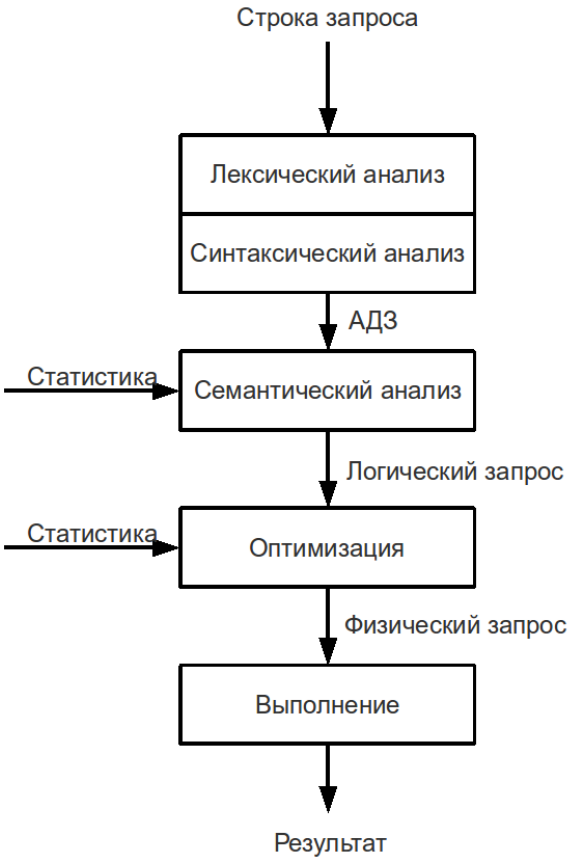
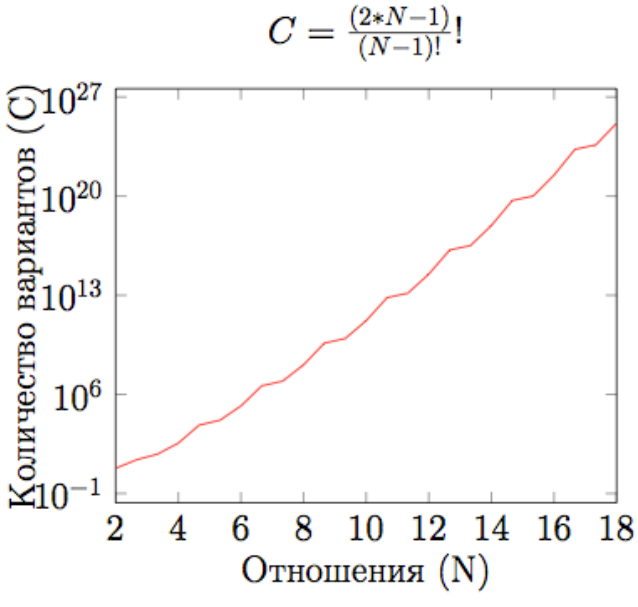


Рис. 11. Общая схема обработки запроса в СУБД

В подсистеме выполнения запросов используются физические реализации операторов. Операторы берут входные данные и производят над ними операции, возвращая выходные. Каждый из таких физических операторов не обязательно соответствует один к одному какому-либо логическому оператору.

Оптимизатор запросов ответственен за генерирование входных данных для подсистемы выполнения. Он берет логическое представление запроса и выполняет поиск наиболее эффективного соответствующему ему физическому в пространстве возможных планов выполнения.

Данная задача не проста, как кажется на первый взгляд, так как количество деревьев выполнения в пространстве поиска может быть огромно. Так, алгебраическое представление плана запроса может быть трансформировано во множество других эквивалентных ему; каждому из логических операторов в запросе может соответствовать несколько различных физических имплементаций (например, несколько алгоритмов соединения JOIN в реляционных СУБД). На рисунке 2 показано отношение количества альтернативных вариантов запросов в зависимости от количества используемых в запросе отношений в реляционной СУБД. Видно, что рост вариантов экспоненциальный и каждое новое отношение усложняет поиск наилучшего плана больше чем на порядок [4].



#

Рис. 12. Зависимость количества вариантов запросов от количества отношений для SQL СУБД

Проблема оптимизации состоит во-первых в нахождении всех возможных альтернативных вариантов запроса и во-вторых — в выборе наиболее эффективного варианта из множества альтернативных.

Обе части проблемы нетривиальны. Прежде всего требуется найти все корректные планы выполнения запроса и не упустить какой-либо план, который является наиболее эффективным. Для облегчения второй задачи требуется сократить пространство найденных корректных планов, оставив только те планы, которые потенциально эффективны. Обе задачи являются не полностью формализуемыми, так как отсутствуют точные математические критерии выбора [5].

Первая часть проблемы решается с помощью определения множества правил преобразования запросов. То есть программной логикой, которая из исходного запроса генерирует альтернативные ему варианты.

Для второй же требуется формальные критерии отбора, например стоимость выполнения запроса предложенный Патрицией Селинджер [6]. Сокращение пространства поиска или поиск пути в пространстве множества альтернативных планов — отдельная нетривиальная задача, для решения которой зачастую применяются эвристические и генетические алгоритмы, а также методы динамического программирования [7, 8].

#

Рис. 13. Элементы оптимизатора запросов

Схематично в оптимизаторе запросов можно выделить следующие части (рис. 3):

1. Пространство возможных планов или пространство поиска получаемых с помощью преобразований запроса;
2. Метод определения стоимости планов;
3. Алгоритм перебора планов пространства поиска — поисковый движок.

Можно сказать, что хороший оптимизатор запросов это такой оптимизатор, пространство поиска которого содержит планы с низкой стоимостью, метод подсчета стоимости точен, а алгоритм перебора — эффективен.

Разберем по-отдельности ключевые особенности SciDB и каждый из узлов оптимизатора. Как они формируются и влияют друг на друга?

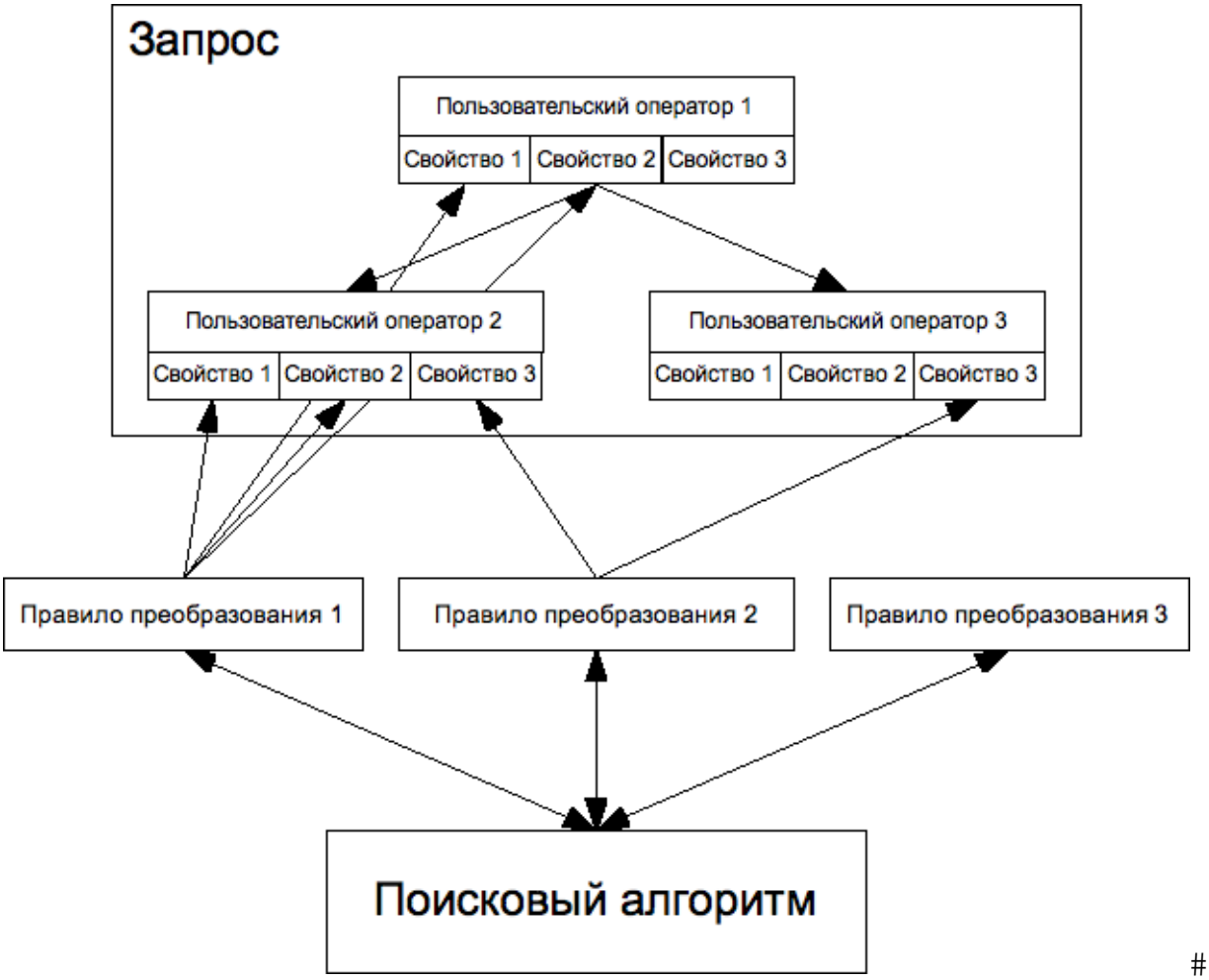
Логические и физические операторы в SciDB представляет собой классы на языке C++, экземпляры которых конструируется во время разбора запроса и его компиляции. Операторы могут быть как встроенными, так и загружаемыми из разделяемых библиотек. На

этом расширяемость SciDB заканчивается, так как оптимизатор ничего не знает о пользовательских операторах.

Работы по созданию расширяемого оптимизатора ведутся достаточно давно, так например один из первых генераторов оптимизаторов Starburst [9] был вполне успешным каркасом для построения расширяемого оптимизатора. Однако расширяемость до текущего времени так и осталась лишь на уровне исходного кода во всех фреймворках и СУБД. В действительности она и не требовалась в существующих СУБД, ограничиваясь настраиваемыми предикатами с пользовательскими скалярными функциями, для увеличения точности предсказания селективности. Популяризация MapReduce [10] тем не менее привела к попыткам оптимизации запросов содержащих пользовательские операторы. Например в [11] выполняется непосредственная оптимизация с помощью анализа кода пользовательских операторов. Несомненно такая оптимизация возможна, но стоит признать, что далеко не всегда, так как сложность и реализацию пользовательского кода предугадать невозможно.

Оптимизаторы Cascades [12] и OPT++ [13] активно используют понятие свойств операторов, однако только для внутренних целей, например свойство «sort» говорит о том, что оператору требуется сортированный вход, что приводит к принудительной вставке оператора сортировки, остальные же преобразования запросов производятся с помощью фиксированных наборов правил, что исключает использование их, как расширяемых пользовательскими операторами.

Идеальным для расширяемого оптимизатора было бы использование исключительно свойств операторов, то есть все правила преобразований абстрактны от операторов и руководствуются только свойствами операторов (рис. 4), сами же операторы рассматриваются, как черный ящик. В таком случае можно реализовать не только расширение для операторов, но и для правил преобразования. Таким образом библиотека пользовательских операторов может содержать не только операторы, но и правила преобразований запросов с ними.

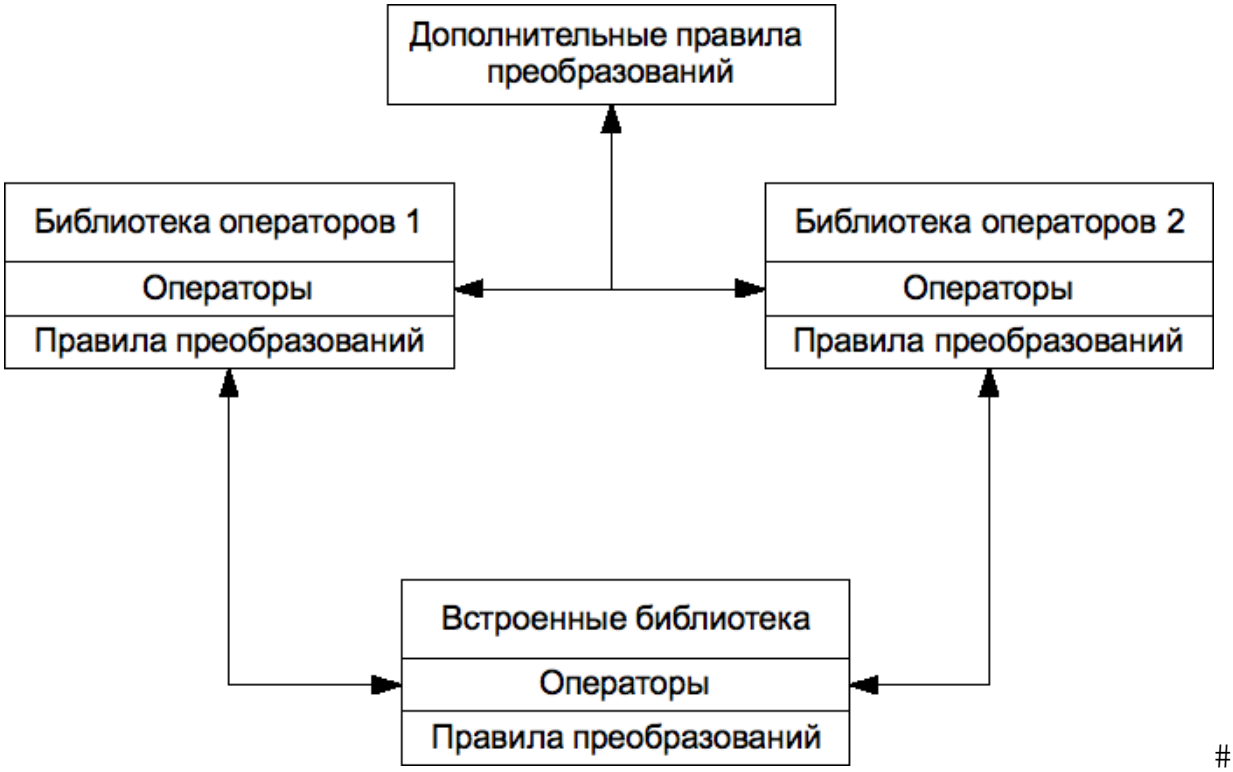


#

Рис. 14. Преобразование запроса руководствуясь свойствами

Однако встает вопрос об интеграции в оптимизаторе наборов операторов независимых разработчиков. Вполне возможно, что такая проблема может решиться только администратором СУБД, которому придется добавить новые «промежуточные» правила преобразования запросов (рис. 5).

В некоторой степени облегчает задачу то, что в большинстве случаев SciDB использует только два набора операторов: встроенный и пользовательский разработанный непосредственно для использования в какой либо области применения. Тем не менее этот вопрос важен для полноценной расширяемой СУБД и требует проработки, так как добавление правил преобразования и исследования совместимости между ними — задача часто требующая большой квалификации и времени.



#

Рис. 15. Преобразование между двумя сторонними библиотеками операторов

Для внутреннего представления запросов SciDB использует дерево, где каждый узел представляет собой оператор и входы вышестоящих операторов, причем объект класса оператора отделен от узла, также как логическое и физическое представление запроса.

Данный подход имеет преимущество, так как производить некоторые операции преобразования над деревом очень просто, например для обмена двух операторов местами производится всего лишь обменом адресами и не затрагивает их входы и выходы. Однако есть и ряд недостатков: хранение пространства поиска в таком виде ведет к большим накладным расходам, так как на каждый альтернативный вариант приходится иметь полную копию всего плана запроса (рис. 7}); невозможно оптимизировать запрос частично, потому что логический план не может содержать в себе физические операторы. Данная проблема может быть решена с помощью использования групп: наборов альтернативных вариантов подпланов, то есть каждое изменение плана производящее альтернативные изменения в пределах подплана, выделяется в виде группы (рис. 8). Такое разбиение не только позволяет экономить память, но и упрощает дальнейшую навигацию в пространстве поиска. Данная техника была

предложена в Volcano [14] и зарекомендовала себя впоследствии в коммерчески успешном Cascades [15] и его последователе Columbia [16].

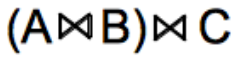


Рис. 16. Изначальный запрос

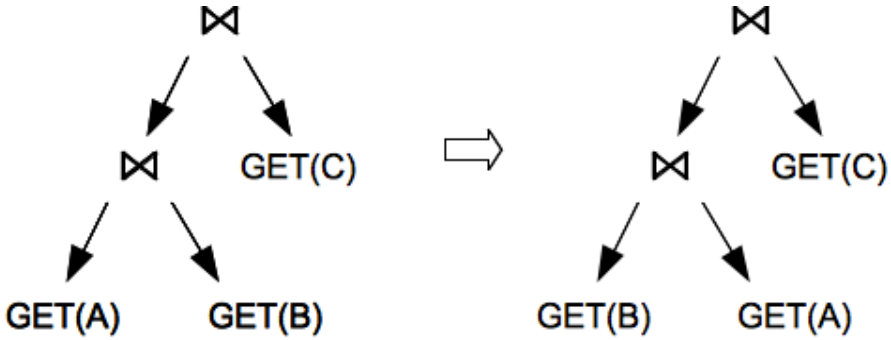


Рис. 17. Дублирование альтернативных вариантов

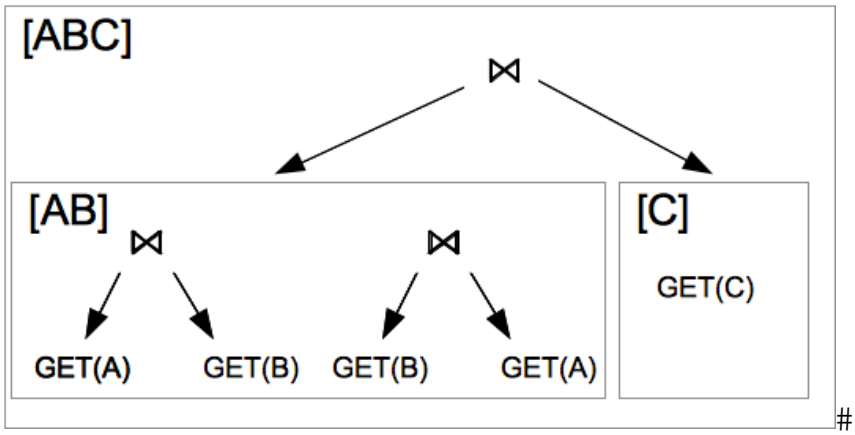


Рис. 18. Альтернативные варианты упаковываются в группы

Разработка наборов правил преобразований наиболее трудоемкий этап данной части. Успех его в большей степени зависит от изученности области применения, для которой был написан набор операторов. Так, операторы и модель данных реляционной алгебры формализована и известна. В то же время оптимизация запросов с использованием операторов нелинейной алгебры может стать серьезным препятствием при реализации.

Разработка модели оценки стоимости запроса предполагает изучение модели данных. SciDB использует вертикальную модель

многомерных массивов. Логически массивы выглядят, как многомерные кубы, в ячейках которых хранятся атрибуты. Под вертикальной моделью понимается, что физически каждый атрибут хранится отдельно. Также в SciDB используется понятие «чанка» (англ. chunk, кусок). Чанк является единицей считывания/записи и обработки данных. Таким образом размер чанка — основная величина участвующая в расчете стоимости доступа к диску.

Подобным образом необходимо проработать каждый используемый ресурс (память, процессор, GPU, сеть), однако, как говорилось ранее SciDB использует расширяемый язык запросов, где оператор является черным ящиком для системы, поэтому узнать какие чанки и в каком порядке прочитаны и переданы по сути невозможно.

Вполне логично решить, что определять стоимость оператора должен сам оператор. Данное решение вполне вписывается в концепцию использования свойств оператора для манипуляции запросом: оператору (а точнее программисту написавшего этот оператор) достаточно знать алгоритм работы, чтобы вычислить использование того или иного ресурса компьютера. Статистическую информацию о данных над которыми будет выполняться оператор может предоставить оптимизатор. Таким образом интерфейс SciDB должен содержать необходимые данные о ресурсах компьютера (а возможно и кластера в целом), а также методы получения статистической информации.

Второй подход, который можно применить в данной ситуации — это эмпирический подход, то есть получение стоимости выполнения непосредственно после выполнения оператора [16, 17]. Не зная ничего о стоимости оператора можно «прогнать» его на данных базы с целью профилирования и впоследствии корректировать значения, при поступлении новых данных.

В любом из этих двух случаев модель стоимости должна быть формализована.

Поисковый алгоритм или стратегия поиска — ядро оптимизатора, которое полностью задает поведение оптимизатора. Именно в нем происходит сужение области поиска, что напрямую влияет на производительность и качество оптимизации, то есть задается баланс между скоростью оптимизации и скоростью результирующего плана.

Алгоритм сам по себе не стоит делать расширяемым, так как это необоснованно увеличит сложность его реализации. Однако иметь несколько динамически переключающихся алгоритмов поиска в зависимости от ситуации, вполне возможно, как например поступили разработчики PostgreSQL задействовавшие два алгоритма: динамического программирования по-умолчанию и генетического, при использовании большого количества отношений.

Важность реализации алгоритма поиска в расширяемой СУБД — адекватная работа с пользовательскими операторами. Стратегия не должна основываться на знании операторов, как это выполнено сейчас, иначе это полностью нивелирует попытки создания расширяемого оптимизатора. Хорошей отправной точкой может послужить стратегия поиска Columbia [15] — Глобальное Эпсилон Отсечение, основанное на стоимости высших групп запроса.

Последняя особенность SciDB, которая должна быть учтена в оптимизаторе — это распределенность. Причем использование кластера может быть использовано как по-прямому назначению, так и косвенно. Под прямым назначением, подразумевается глобальная оптимизация или мультизапросная оптимизация [18]. В отличии от СУБД общего назначения, SciDB не нацелена на выполнения множества запросов одновременно, поэтому предиктивное перераспределение данных, создание и перестановка очередей запросов — путь к наилучшей утилизации ресурсов кластера. В частности данный вопрос был рассмотрен Самаревым Р. С. в работе «Методы и модели проектирования параллельных СУБД» [19]

Под косвенным назначением оптимизатор запросов может использовать мощности кластера для увеличения точности поиска наилучшего плана. Очевидно, что линейное наращивание размеров кластера не может помочь в условиях экспоненциального роста сложности запроса, однако распределенный анализ может помочь более глубоко исследовать подающие надежды варианты запросов. Также за счет использования свободных мощностей кластера можно увеличить точность эмпирической стоимости, выполняя предварительные запуски, вместо того, чтобы дожидаться следующего запуска запроса, что, учитывая характер хранения данных «только для чтения» в SciDB, теоретически должно дать выигрыш в данном случае.

Итак, мы рассмотрели описание оптимизатора запросов SciDB. Из всего вышесказанного можно сделать следующие выводы:

1. Приемы построения оптимизатора запросов, десятилетиями совершенствующиеся на реляционных СУБД вполне применимы к расширяемым СУБД.

2. Расширяемость СУБД значительно увеличивает сложность разработки оптимизатора. Можно с уверенностью сказать, что на данный момент системы активно использующие логическую оптимизацию запросов с пользовательскими операторами отсутствуют (хотя и ведутся исследования), что дает большое поле для исследований в частности — это исследования расширяемой модели стоимости, генераторов вариантов и поисковых алгоритмов.

3. Элементы оптимизатора, которые зависят от моделей используемых в СУБД практически не используемы повторно, так как модель стоимости и правила преобразования запросов жестко связаны с используемой моделью данных и алгеброй запросов. Тем не менее принципы их использования практически не разнятся, однако формальная модель должна быть выведена всегда.

Естественно — это очень грубый набросок системы, причем далеко не единственный ее возможный вариант. Как мы видим даже классический способ разработки оптимизатора запросов предоставляет большое поле для изучения и деятельности. В то же время всегда существует место для применение альтернативных подходов, как например полностью эмпирический оптимизатор запросов MongoDB [20]. Логичным продолжением данной работы должно стать исследование каждой из обозначенных проблем в оптимизаторе SciDB, что послужит положительным развитием как самой SciDB, так и возможным улучшением других NoSQL СУБД.

Непосредственно области, которые предполагается исследовать:

1. Моделирование и прототипирование модели динамически расширяемого оптимизатора.

2. Разработка и исследование стоимостной модели данных характерной SciDB.

3. Разработка и исследование поискового алгоритма, который соответствует модели динамически расширяемого оптимизатора.

4. Разработка и исследование алгоритма распределенного анализа и оптимизации запросов.

Литература

1. *Кузнецов С.* Будущее транзакционных систем / Открытые системы. — 2011. — № 4

2. *Stonebraker M.* One size fits all: An Idea Whose Time Has Come and Gone / *M. Stonebraker, U. Cetintemel* // Proceedings of the 21st International Conference on Data Engineering. — ICDE ’05. — Washington, DC, USA: IEEE Computer Society, 2005.— Pp. 2–11. http://dx.doi. org/10.1109/ICDE.2005.1.

3. *Brown P. G.* Overview of scidb: large scale array storage, processing and analysis / Proceedings of the 2010 international conference on Management of data. — SIGMOD ’10. — New York, NY, USA: ACM, 2010. — Pp. 963–968.

4. *Дж. Дейт К.* Введение в системы баз данных / М.: Наука, 1980. — 463 с.

5. *Кузнецов, С.* Оптимизация запросов: вечнозеленая область / Открытые системы. — 2003

6. Access path selection in a relational database management system / *P. G. Selinger, M. M. Astrahan, D. D. Chamberlin et al.* // Proceedings of the 1979 ACM SIGMOD international conference on Management of data. — SIGMOD ’79. — New York, NY, USA: ACM, 1979. — Pp. 23–34.

7. System R: Relational Approach to Database Management / *M. M. Astrahan, H. W. Blasgen, D. D. Chamberlin et al.* // ACM Transactions on Database Systems. — 1976. — Vol. 1. — Pp. 97–137.

8. *Bennett, K.* A genetic algorithm for database query optimization / K. Bennett, M. C. Ferris, Y. E. Ioannidis // In Proceedings of the fourth International Conference on Genetic Algorithms.— Morgan Kaufmann Publishers, 1991. — Pp. 400–407.

9. *Pirahesh, H.* Extensible/rule based query rewrite optimization in starburst / H. Pirahesh, J. M. Hellerstein, W. Hasan // SIGMOD Rec. — 1992.—June.— Vol. 21.— Pp. 39–48. http://doi.acm.org/10.1145/ 141484.130294.

10. *Dean, J.* MapReduce: simplified data processing on large clusters / *J. Dean, S. Ghemawat, G.* Inc // In OSDI’04: Proceedings of the 6th conference on Symposium on Opearting Systems Design and Implementation. — USENIX Association, 2004.

11. *Iu, M.-Y.* HadoopToSQL: a mapReduce query optimizer / *M.-Y. Iu, W. Zwaenepoel* // EuroSys’10. — 2010. — Pp. 251–264.

12. *Graefe, G.* The cascades framework for query optimization / *G. Graefe* // IEEE Data Eng. Bull. — 1995. — Vol. 18, no. 3. — Pp. 19–29.

13. *Kabra, N.* Opt++ : an object-oriented implementation for extensible database query optimization / *N. Kabra, D. J. DeWitt* // The VLDB Journal. — 1999. — April. — Vol. 8. — Pp. 55–78.

14. *Graefe, G.* The volcano optimizer generator: Extensibility and efficient search / *G. Graefe* // ICDE. — 1993. — Pp. 209–218.

15. *Xu, Y.* Efficiency in the columbia database query optimizer: Tech. rep. */ Y. Xu*: Portland State University, 1998.

16. *Reddy, N.* Analyzing plan diagrams of database query optimizers / *N. Reddy, J. R. Haritsa* // Proceedings of the 31st international conference on Very large data bases. — VLDB ’05. — VLDB Endowment, 2005. — Pp. 1228–1239.

17. *Valluri, S. R.* Towards empirically driven query optimization: Tech. rep. / *S. R. Valluri, K. Karlapalem, A. Hulgeri*: Centre for Data Engineering International Institute of Information Technology, 2009.

18. On multi-query optimization: Tech. rep. / *R. Choenni, M. L. Kersten, J. van den Akker et al.*: in Proc. COMAD ’97 8th Int. Conference on Management of Data, 1996.

19. *Самарев Р. С.* Методы и модели проектирования параллельных субд. — 2007.

20. *Stern M.* Indexing, Query Optimization, the Query Optimizer — MongoAustin. — 2011.

E-MAIL: SMIRNOFFJR@GMAIL.COM

ТЕЛЕФОН: +7920944330

НАУЧНЫЙ РУКОВОДИТЕЛЬ:

Д.Т.Н. АНДРИАНОВ Д.Е.