САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Математико-механический факультет

Кафедра системного программирования

Савельев Николай Геннадьевич

Реализация обобщенных операторов реляционной алгебры в системе AsterixDB BDMS

Курсовая работа

Научный руководитель:
Новиков Б.А., профессор, зав. кафедры информационно-аналитических систем

Санкт-Петербург
2014

Оглавление

Введение

Объем данных, отличающихся различной структурой, растет с каждым годом. Обработка такой информации требует специальных подходов, методологий и инструментов. Системы, способные эффективно работать с различными форматами данных и информационными ресурсами, позволяют решить эту проблему. Такие системы должны обладать мощным языком запросов, оптимизатором на основе обобщенной реляционной алгебры и модели стоимостей и обработчиками данных, способными работать с различными информационными форматами.

Схема исполнения запросов в системах с гетерогенными источниками может не отличаться от схем в обычных реляционных базах данных. Общие стадии исполнения запроса описаны в [10] и состоят из:

- парсер языка запросов;
- оптимизатор;
- генератор кода;
- исполнитель запросов.

Если мы рассматриваем реляционные базы данных, то все эти стадии находятся в рамках одной системы. Минусом таких систем является необходимость в хранении всех обрабатываемых данных в некоторых заранее определенных форматах, которые не всегда совпадают при переходе от одной системы к другой, что делает невозможным обработку информации с различных источников. Именно поэтому и возникает потребность в системах, способных обрабатывать гетерогенные источники информации.

Одним из способов создания такой системы является представление отдельных систем управления базами данных как информационных источников, что избавляет разработчиков от реализации самой обработки информации. В таком случае для поддержки информационного ресурса требуется реализовать вычисление дополнительной информации об объектах с разных источников для их унифицированной обработки, трансляцию операторов расширенной реляционной алгебры, которыми оперирует оптимизатор, в язык исполнения запросов целевой платформы, а также модель стоимостей, позволяющей оптимизатору выбирать не

только оптимальный план, но и платформу, на которой запрос или его часть будет исполняться максимально эффективно. На рис. 1 представлена схема подобной системы.

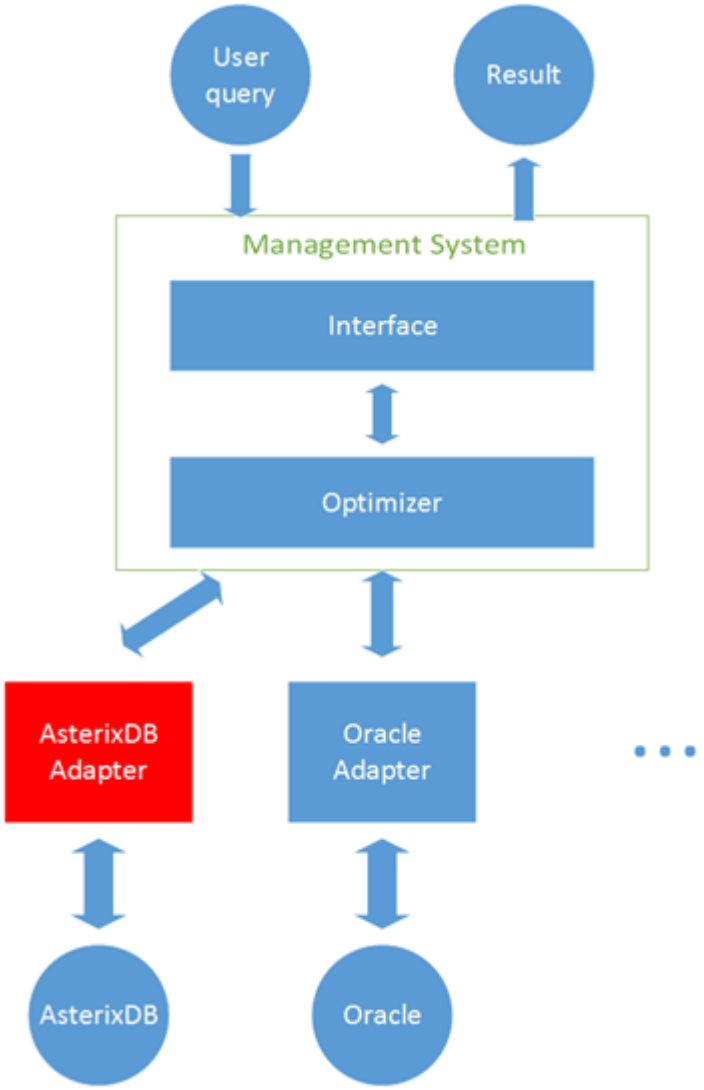

Рисунок 1. Общая схема системы обработки гетерогенных источников информации

Как видно из рис.1 пользователь имеет унифицированный интерфейс для взаимодействия с с различными источниками данных. В то же время оптимизатор взаимодействует с этими источниками через адаптеры, которые выполняют трансляцию операторов расширенной реляционной алгебры, для возвращаемых от источника данных вычисляют дополнительную информацию, чтобы система могла корректно с ними работать (оценки, идентификаторы, аттрибуты), а также предоставляют функции модели стоимостей для поиска оптимального

планы запроса. Главной целью данной работы является подобного адаптера для системы управления базами данных AsterixDB [2].

В следующем разделе будет проведен обзор литературы и работ в таких направления как реляционные алгебры, модели стоимостей, системы управления большими объемами данных и обработка информации. Далее, будет представлена расширенная реляционная алгебра из [1], используемая в данной работе, и описана система AsterixDB. После этого будет приведена схема трансляции операторов расширенной реляционной алгебры и представлена модель стоимостей для этих операторов.

1. Обзор литературы и работ

1.1. Реляционные алгебры

В данной работе представлена реализация операторов расширенной реляционной алгебры из [1], которая может использоваться оптимизатором в системе с гетерогенными источниками данных. В ней определены такие операторы как проекция, соединение, аггрегирование и др. Авторы описывают алгебру с универсальным подходом к представлению, оптимизации, параметризации и исполнению сложных запросов.

Обобщение классической реляционной алгебры с введением нечетких операций представлено в [4]. Авторы определяют язык запросов, который совмещает в себе традиционное реляционное исчисление и обработку нечетких значений. Этот набор алгебраических операторов включает в себя соединение, умножение, проекцию, выборку, объединение, пересечение и разность множеств, а также предикаты схожести и веса.

Реляционная алгебра на основе схожести объектов с весами описана в [3]. В этой алгебре могут быть представлены сложные запросы с различной интерпретацией значений схожести объектов и с различными алгоритмами их исполнения. Алгебра поддерживает такие операторы как соединение, слияние, выборка, распределение, объединение, пересечение и разность множеств.

1.2. Модели стоимостей

Для нахождения оптимального плана исполнения запроса оптимизатору должна быть предоставлена модель стоимостей для оценки конкретных планов. Ниже представлены несколько подходов к созданию моделей стоимостей для расширенный реляционной алгебры.

Модель стоимостей на основе оценки мощности и статистики выборки описана в [3].

Авторы [9] вводят нечеткую модель стоимостей для исполнения запросов в системе с несколькими базами данных, основанную эвристиках исполнения и статистике выборки данных.

Создание модели стоимостей для алгоритмов параллельных соединений описано в [13]. Автор строит модель для различных алгоритмов и производит их сравнение.

1.3. Платформы исполнения

Описание системы управления большими объемами данных AsterixDB представлено в [5]. Эта система имеет большой набор операций и предназначена для хранения полуструктурированных данных.

Другие системы хранения больших объемов данных:

- Apache Cassandra [8] – распределенная система хранения данных, в которой данные хранятся в виде ключ-значение, где ключом выступает хэш значения.
- Apache Hadoop [6] – фреймворк для хранения больших объемов данных и их обработки.
- Apache Hive [7] – хранилище данных, построенное на основе Hadoop для исполнения запросов и анализа данных.

2. Расширенная реляционная алгебра

В данной работе план запроса выражается оптимизатором с помощью расширенной реляционной алгебры, представленной в [1]. Центральной концепцией в данной алгебре является q–set – тройка (q, B, S), где:

- q – запрос;
- B – базовое множество объектов для обработки;
- S – функция оценки для объектов из B.

Q-set инкапсулирует в себе как сам запрос, так и результат его исполнения совместно с оценками, вычисленными с помощью функции оценки. В этой модели не важно как выглядит сам запрос (q), но каждый объект из (B) должен обязательно иметь дополнительное поле, хранящее его оценку, которая может представлять точность вычисления объекта, точность его источника и др. В общем случае оценка представляет собой релевантность объекта запросу. Она может использоваться в нечетких запросах и приближенном вычислении их результатов. Функция оценки (S) используется для изменения оценки объектов и зависит от запроса.

Все операторы данной расширенной реляционной алгебры определяются с помощью q – set. Это такие операторы как:

- filter (фильтрация) – обобщение оператора проекции с вычислением оценок;
- join (соединение);
- aggregate (группировка);
- fusion (слияние) и др.

Ниже представлены примеры использования этих операторов:

- filter(рестораны, ‘итальянские’) – выделяет из базового множества ресторанов все итальянские без модификации оценок объектов;
- filter(B, функция_оценки=’нормализация’) – нормализует оценки объектов из множества B;

- Group_join(
  filter (магазины, ‘есть спагетти‘),
  filter (рестораны, ‘итальянские’),
  ‘расстояние < 20 км’,
  функция_оценки =’произведение’) – группирует два соединенных множества объектов, расстояние между которыми не превосходит 20 км, и для каждой группы вычисляет новую оценку, являющуюся произведением оценок объектов, составляющих группу.

Полный набор операторов расширенной реляционной алгебры может быть найден в [1]. Важно заметить, что операторы могут иметь множество параметров (алгоритмы вычисления, временные рамки исполнения и пр.) и исполняться на внешних узлах, не входящих в саму систему.

3. AsterixDB BDMS

AsterixDB BDMS – система управления большими объемами данных, использующая полуструктурированную модель данных, основанную на расширении формата JSON свойствами объектов из объектно-ориентированных баз данных, а также имеющая мощный язык запросов, который поддерживает большое количество операций для обработки и анализа полуструктурированных данных. Главной причиной для использования данной системы в этой работе являются язык запросов, с помощью которого легко выражаются операторы расширенной реляционной алгебры, использование конвейера в движке вычисления запросов без материализации промежуточных результатов и обработка данных, которые могут напрямую не храниться в системе.

4. Соединение системы AsterixDB с оптимизатором

Как уже пояснялось раннее, одной из идей данной работы является использование системы AsterixDB как платформы исполнения для оптимизатора. Для этого должны быть реализованы генератор кода (транслятор) и модель стоимостей, объединенные в один модуль. Этот модуль соединяет оптимизатор и систему AsterixDB, дополняет объекты оценками и метаинформацией, транслирует план запроса, полученный от оптимизатора и выраженный с помощью операторов реляционной алгебры, в язык запросов AsterixDB и предоставляет функции модели стоимости для определения оптимизатором наилучшего плана исполнения запроса.

Сам оптимизатор строит дерево запроса, где алгебраические операторы находятся в узлах, а источники информации и отношения - в листьях. После этого дерево перестраивается для оптимизации вычисления результата. Для этого оптимизатор может не только использовать алгебраические свойства операторов, но также выбирать наиболее удобную платформу исполнения для вычисления результата какого-либо поддерева. Также результат вычисления некоторого поддерева запроса на одной платформе исполнения может быть передан другой платформе в качестве информационного источника для вычисления верхних узлов дерева.

5. Трансляция запросов

Данный раздел описывает процесс трансляции плана запроса, полученного от оптимизатора и выраженного с помощью операторов реляционной алгебры, в язык запросов AsterixDB.

5.1. Транслятор

Главной частью разработанного модуля для соединения с оптимизатором является транслятор. На вход он получает поддерево плана запроса, переводит его в код на языке запросов AsterixDB (AQL), после этого, в зависимости от требований оптимизатора, он либо рассчитывает функции стоимости по полученному плану, либо отправляет преобразованный код на исполнение.

5.2. Представление q-set

Так как все операторы расширенной реляционной алгебры выражаются в терминах троек q-set (q,B,S), которые были описаны раннее, то для трансляции самих операторов нужно иметь представление q-set в системе AsterixDB.

Q-set может быть представлен в различных формах, но, с практической точки зрения, выгоднее представлять эту тройку (q, B, S) как шаблон запроса (q) с выделенными параметрами для обрабатываемого базового множества (B) и функции оценки (S). Важно заметить, что базовым множеством может служить другой оператор алгебры, поэтому необходимо предусмотреть возможность вызовов цепоцек разных операторов.

Язык запросов AsterixDB (AQL) [2] спроектирован на основе языка xQuery – языка для обработки полуструктурированных и неструктурированных данных. Запрос строится с применением конструкции FLWOR – (for, let, where, order by, return), в которой происходит последовательный доступ к элементом обрабатываемого отношения. Тогда нетрудно представить шаблон для тройки q-set (q, B, S):

q = “for $item in **B** return **S** ($item)”

Этот запрос модифицирует оценку объектов посредством функции оценки S. Важно заметить, что элемент B также может быть выражен с

помощью конструкции FLWOR, что в значительной мере упрощает трансляцию цепочек операторов.

Таким образом, все операторы расширенной реляционной алгебры будут транслироваться в похожие конструкции на языке AQL, о которых будет рассказано ниже. Остается добавить, что для поддержки работы функции оценки необходимо каждый объект обеспечивать дополнительным полем, содержащим его оценку, если такового не имеется.

5.3. Трансляция операторов

Транслятор запросов оперирует шаблонами запросов, представляющими конкретные операторы реляционной алгебры. Никакой дополнительной оптимизации запроса не требуется, так как за оптимизацию плана запроса в терминах операторов реляционной алгебры отвечает оптимизатор, а за оптимизация кода запроса на языке AQL – сама система AsterixDB. Далее будут рассмотрены более детально решения для трансляции операторов фильтрации и соединения.

5.3.1. Трансляция оператора фильтрации

Оператор фильтрации исполняет необходимую последовательность действий над каждым объектом базового множества. Это может быть функция оценки, предикат или другая функция.

Главное правило трансляции для оператора фильтрации представлено ниже:

Получаемый транслятором оператор:

filter(BaseSet, Expression(arguments),ScoringFunction);

Трансляция оператора в язык AQL:

```
for $obj in BaseSet
    where Expression (arguments)
return ScoringFunction($obj);
```

Как показано в примере, выражение для фильтрации проверяется в блок ‘where’, а функция оценки вычисляется для каждого объекта при возврате результата.

5.3.2. Трансляция оператора соединения

Оператор соединения представляется в языке AQL вложенными конструкциями FLWOR, но при этом, данный синтаксис позволяет системе AsterixDB варьировать алгоритмы вычисления соединения в зависимости от условия.

Главное правило для трансляции точного оператора соединения представлено ниже:

Получаемый транслятором оператор:

join(FirstSet, SecondSet, Expression(arguments), ScoringFunction);

Трансляция оператора в язык AQL:

```
for $first in FirstSet
    for $second in SecondSet
        where Expression (arguments)
    return {
        $first,
        $second,
        ScoringFunction($first, $second)
    };
```

Как можно заметить, данный запрос соединяет два множества по определенному условию (выражению), которое может быть параметризовано. Результатом является новое множество, состоящее из троек объектов: соединенных объектов из первого и второго множества и вычисленная для этой пары оценка. То, как она будет вычисляться, зависит от самой функции.

6. Модель стоимостей

В данной работе на модель стоимостей возлагается только одна задача – по предложенному плану запроса определить примерное время его исполнения. Для этого было сделано несколько предположений об обработке данных в системы AsterixDB. Эта система имеет неразделяемую между узлами память, поэтому все данные равномерно распределены между ними. Тогда становятся важными две детали: количество узлов системы и время считывания данных с узлов. Будем считать, что система имеет P узлов. Время считывания данных принято определять временем считывания одного блока данных, в которых система хранит элементы множеств. Многие системы управления базами данных работают с данными на уровне блоков, а не самих элементов. Система AsterixDB не является исключением. В ней размер блока равен 1Мб. Тогда время считывания блока данных с диска или с другого узла обозначим за C_b.

6.1. Оператор фильтрации

Оператор фильтрации имеет линейную алгоритмическую сложность. В функции времени исполнения этого оператора берутся в расчет количество считанных блоков, в которых хранится отношение, его мощность и временная константа вычисления выражения на каждом элементе:

$$t_f = \frac{C_b N_b + C_e |N|}{P}, \quad \text{где}$$

- $t_f -$ время исполнения оператора фильтрации;
- $C_b -$ время считывания одного блока данных;
- $N_b -$ количество считанных блоков данных;
- $C_e -$ временная константа для оператора фильтрации;
- $|N| -$ мощность отношения N;
- $P -$ количество узлов в системе.

Все рабочие узлы системы для вычисления результата оператора фильтрации используют хранящиеся в них данные, не прибегая к их распространению на другие узлы.

6.2. Оператор соединения

Для создания функции времени исполнения оператора соединения должен быть известен алгоритм параллельного соединения, который используется системой AsterixDB. Было сделано предположение о том, что в данной системе используется алгоритм разделения и распространения (divide and broadcast parallel join algorithm) [14]. Этот алгоритм состоит из двух частей – разделение данных по узлам с предоставлением доступа к требуемым данным и локальное соединение. Первый этап состоит в разделении большего по размеру отношения на непересекающиеся подмножества и распределение их по имеющимся в системе рабочим узлам. Также всем узлам предоставляется полностью меньшее по размеру отношение. Второй этап состоит в локальном соединении имеющихся на узлах частей отношений. В данном случае каждая часть большего по размеру отношения будет соединена со вторым отношением, доступ к которому будет предоставлен каждому узлу. В системе AsterixDB все данные уже изначально распределены по узлам, поэтому в первом этапе алгоритма распределение частей большего отношения больше не требуется.

Локальное соединение вычисляется с помощью трех алгоритмов. Для каждого алгоритма существуют определенные условия. Именно поэтому были созданы три функции расчета времени для оператора соединения. В них подразумевается, что оператор соединения применяется для двух отношений N и M, где первое отношение занимает меньше блоков данных, и именно части этого отношения будут распространяться на все узлы при вычислении оператора соединения.

Далее будут кратко описаны алгоритмы локального соединения и представлены функции расчета времени исполнения. Каждая формула состоит из двух главных слагаемых, где первое отвечает за время считывания блоков данных, а второе – за время работы самого алгоритма.

6.2.1. Соединение на основе хэш-таблиц

Данный алгоритм по одному из соединяемых отношений строит хэш-таблицу. После этого происходит проход по элементам второго отношения на предмет равенства ключей. Соответственно, функция времени исполнения оператора соединения при условии равенства выглядит следующим образом:

$$t_{hj} = \frac{C_b}{P}\left((P-1)N_b + M_b\right) + C_e\left(|N| + \frac{|M|}{P}\right), \text{ где}$$

- $t_{hj} -$ время исполнения оператора фильтрации;
- $C_b -$ время считывания одного блока данных;
- $N_b -$ количество считанных блоков данных отношения N;
- $M_b -$ количество считанных блоков данных отношения N;
- $C_e -$ временная константа для оператора фильтрации;
- $|N| -$ мощность отношения N;
- $P -$ количество узлов в системе.

Алгоритм на основе построения хэш-таблицы может использоваться только при эквисоединении (соединении на основе равенства аттрибутов) и имеет линейную алгоритмическую сложность в среднем.

6.2.2. Соединение на основе сортировки и слияния

Данный алгоритм начинает свою работу с сортировки отношений по соединяемому аттрибуту (набору аттрибутов). После этого происходит параллельный проход по двум отсортированным отношениям и поиск элементов, где условие соединения будет выполняться. Тогда, функция времени исполнения оператора соединения при транзитивном условии выглядит следующим образом:

$$t_{mj} = \frac{C_b}{P}\left(M_b \log\left(\frac{M_b}{P}\right) + N_b \log\left(\frac{N_b}{P}\right) + (P-1)N_b + M_b\right) + C_e\left(\frac{|N| log\left(\frac{|N|}{P}\right)}{P} + \frac{|M| \log\left(\frac{|M|}{P}\right)}{P} + |N| + \frac{|M|}{P}\right), \quad \text{где}$$

- $t_{mj} -$ время исполнения оператора фильтрации;
- $C_b -$ время считывания одного блока данных;
- $N_b -$ количество считанных блоков данных отношения N;
- $M_b -$ количество считанных блоков данных отношения N;
- $C_e -$ временная константа для оператора фильтрации;
- $|N| -$ мощность отношения N;
- $P -$ количество узлов в системе.

Этот алгоритм предполагает, что отношения могут быть отсортированы по условию соединения, что не позволяет использовать этот алгоритм в сложных случаях.

6.2.3. Соединение на основе вложенных циклов

Данный алгоритм использует два вложенных цикла для нахождения элементов из разных отношений, в которых будет выполняться условие соединения. Тогда функция времени исполнения для алгоритма вложенных циклов выглядит следующим образом:

$$t_{nj} = \frac{C_b(P-1)N_bM_b}{P^2} + \frac{C_e|N||M|}{P}, \text{ где}$$

- $t_{mj} -$ время исполнения оператора фильтрации;
- $C_b -$ время считывания одного блока данных;
- $N_b -$ количество считанных блоков данных отношения N;
- $M_b -$ количество считанных блоков данных отношения N;
- $C_e -$ временная константа для оператора фильтрации;
- $|N| -$ мощность отношения N;
- $P -$ количество узлов в системе.

Для этого алгоритма нет ограничений, но он имеет квадратичную алгоритмическую сложность, что может негативно сказываться на времени исполнения, поэтому данный алгоритм используют только когда нельзя использовать остальные.

7. Заключение

В данной работе был рассмотрен подход к использованию системы AsterixDB в качестве платформы-исполнителя в рамках системы, способной обрабатывать разнородные данные. Для этого были реализован и описан модуль трансляции и соединения с оптимизатором, а также создана модель стоимостей на основе имеющихся данных о работе исполняющей системы.

Список литературы

[1] B. Novikov, A. Vassilieva, A. Yarygina. Querying Big Data. In CompSysTech ’12 Proceedings of the 13th International Conference of Computer Systems and Technologies, pages 1-10, 2012.

[2] AsterixDB BDMS Web site. http://asterix.ics.uci.edu/

[3] Fagin, R.. Fuzzy queries in multimedia database systems, Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, New York, NY, USA, 1998, PODS '98, pages 1-10, ACM.

[4] Deshpande, A., Z. G. Ives, V. Raman. Adaptive query processing, Foundations and Trends in Databases, vol. 1, no. 1, pages 1-140, 2007.

[5] Vinayak R. Borkar, Michael J. Carey, Chen Li. Inside “Big Data Management”: Ogres, Onions, or Parfaits?, EDBT 2012.

[6] Apache Hadoop Web site. http://hadoop.apache.org/

[7] Apache Hive Web site. http://hive.apache.org/

[8] Apache Cassandra Web site. http://cassandra.apache.org/

[9] Qiang Zhu, Larson, P.-A. Establishing a fuzzy cost model for query optimization in a multidatabase system. Proc. of 27th ACM/IEEE Hawaii Int'l Conf. on Syst. Sci., Feb. 1994

[10] Ioannidis, Y.E., Query optimization, *ACM Comput. Surv.*, vol. 28, no. 1, pages. 121–123,1996

[11] B. Novikov, A. Yarygina. Query optimization problem in distributed environment of heterogeneous information resources. In Mathematics, Economic Sciences, and Management: the Centenary of L.V. Kantorovich, pages 57-59. Saint-Petersburg State University, 2012.

[12] A. Yarygina. Execution and optimization techniques for approximate queries in heterogeneous systems. Programming and Computer Software, pages 309-317, 2013.

[13] Alice Pigul. Generalized parallel join algorithms and designing cost models. In Proceedings of the Eighth Spring Researchers' Colloquium on Database and Information Systems, SYRCoDIS '12, pages 29-40, 2012.

[14] D. Taniar, C. H. C. Leung, W. Rahayu, S. Goel. High-performance parallel database processing and grid databases. A John Wiley & Sons, Inc., Publication, New Jersey, 2008.