САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

На правах рукописи

Лукичёв Максим Сергеевич

Оптимизация запросов в слабоструктурированной модели данных

05.13.11 — Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

АВТОРЕФЕРАТ
диссертации на соискание ученой степени
кандидата физико-математических наук

Санкт-Петербург
2009

Работа выполнена на кафедре информатики математико-механического факультета Санкт-Петербургского государственного университета.

Научный руководитель: доктор физико-математических наук,
профессор НОВИКОВ Борис Асенович

Официальные оппоненты: доктор технических наук,
КУЗНЕЦОВ Сергей Дмитриевич
(Институт системного программирования РАН)

кандидат физико-математических наук,
КУРАЛЁНОК Игорь Евгеньевич
(ЗАО "Яндекс")

Ведущая организация: Южно-Уральский государственный университет

Защита диссертации состоится "___"_____ 2009 года в __ часов на заседании совета Д212.232.51 по защите докторских и кандидатских диссертаций при Санкт-Петербургском государственном университете по адресу: 198504, Санкт-Петербург, Петродворец, Университетский пр., 28, математико-механический факультет, ауд. 405.

С диссертацией можно ознакомиться в Научной библиотеке им. М. Горького Санкт-Петербургского государственного университета по адресу: 199034, Санкт-Петербург, Университетская наб., 7/9.

Автореферат разослан "___"_____ 2009 года.

Ученый секретарь
диссертационного совета
доктор физико-математических наук,
профессор Даугавет И.К.

Общая характеристика работы

Актуальность темы. Высокоуровневые языки запросов принято рассматривать как одно из наиболее важных средств, предоставляемых СУБД. Обладая очень большой выразительностью, декларативные языки допускают высокоэффективное выполнение запросов, достигаемое в процессе оптимизации.

Наиболее полно методы оптимизации развиты для СУБД, основанных на реляционной модели данных, и, в частности, ее промышленного аналога SQL. Оптимизаторы современных промышленных СУБД способны генерировать планы очень высокого качества. Однако, для применения в контексте слабоструктурированной модели данных, в частности, XML, эти методы должны быть существенно пересмотрены. Учитывая неизменно возрастающую интенсивность использования XML в качестве модели данных и постоянно растущий объём таких данных, задача оптимизации запросов к XML выходит на первый план. В качестве языка XML-запросов в диссертации рассматривается XQuery, в силу наибольшей распространённости этого стандарта.

Цель работы. Исследование и разработка методов высокоэффективного выполнения запросов к слабоструктурированным данным, представленных на языке XQuery. Для достижения этой цели в диссертации решаются следующие задачи:

- Разработка гибкой алгебры, обладающей необходимыми свойствами для предоставления достаточно широкого пространства допустимых планов запроса.
- Разработка эффективного метода поиска оптимального (субоптимального) плана в терминах разработанной алгебры.
- Разработка прототипа с целью экспериментальной верификации полученных результатов

Основные результаты. В работе получены следующие основные результаты:

- Предложена новая алгебра XQuery-запросов, удовлетворяющая требованиям, сформулированным на основе анализа существующих подходов к оптимизации XQuery-запросов, которым должна отвечать алгебра для построения высокопроизводительных систем выполнения запросов.

- Доказано, что операции предложенной алгебры обладают необходимыми алгебраическими свойствами, такими как ассоциативность и коммутативность. А также, доказано, что в терминах этой алгебры можно получить более эффективные планы, чем при использовании ранее известных алгебр.

- Сформулированы ограничения на пространство допустимых планов, позволяющих применять алгоритмы блочной оптимизации. Доказано, что данные ограничения не приводят к потере оптимального плана.

- Экспериментально продемонстрировано, что при использовании предложенной алгебры могут быть получены значительно более эффективные планы, чем в известных XML СУБД.

Научная новизна. Научной новизной обладают следующие результаты работы:

1. Критерии и требования к алгебре, необходимой для построения высокопроизводительных исполнителей запросов.

2. Алгебра XQuery-запросов, удовлетворяющая этим требованиям.

3. Конкретизация блочного алгоритма для оптимизации XQuery-запросов на основе предложенной алгебры.

Теоретическая ценность и практическая значимость. С теоретической точки зрения в работе формально определены операции, формирующие базис алгебры, и доказаны тождества, обосновывающие допустимость оптимизирующих преобразований. Формализация понятий гарфа

частичных планов и блоков в нём позволяют использовать предложенный блочный метод для оптимизации на основе различных алгебр.

Практическая значимость работы состоит в том, что, разработанная алгебра может служить основой для построения стоимостных оптимизаторов как для XML СУБД, так и для автономных исполнителей запросов промышленных систем. Предложенный метод оптимизации поблочно может существенно ускорить процесс оптимизации, что особенно важно, учитывая возможную сложность XQuery-запросов. Экспериментально показано, что применение предложенных методов может существенно повысить эффективность выполнения запросов в XML СУБД.

Апробация работы. Результаты работы докладывались:

- на Двенадцатой Восточно-Европейской конференции "Advances in Databases and Information Systems" (Пори, Финляндия, сентябрь 2008),
- на Четвёртом коллоквиуме "Spring Colloquium for Young Researchers in Databases and Information Sytems (SYRCoDIS)" (Москва, июнь 2007),
- на семинарах группы теории баз данных при лаборатории исследования операций НИИММ,
- на семинаре Московской секции ACM SIGMOD (Москва, январь 2009).
- на семинаре в Институте системного программирования РАН (Москва, январь 2009).

Публикация результатов. Основные результаты представлены в работах [1-3]. Статья [1] опубликована в журнале, входящем в перечень ВАК. В статье [2] соискателю принадлежат определения компонент алгебры, соавтору - техническое оформление и примеры. В статье [3] соискателю принадлежит метод группировки по соседям (NG), соавтору - метод группировки в порядке документа (DG).

Структура и объем диссертации.

Работа состоит из шести глав, включая введение и заключение, и списка литературы. Общий объём диссертации составляет 120 страниц машинописного текста. Список литературы содержит 73 наименования. Рисунки и таблицы нумеруются по главам.

Основное Содержание работы.

Введение содержит предварительную информацию о предмете исследования.

Первая глава демонстрирует актуальность оптимизации запросов в СУБД, основанных на слабоструктурированной модели данных, в частности XML. За последнее десятилетие XML приобрёл значительное распространение как универсальное средство обмена данными, став одним из основных стандартов обмена информации между информационными системами. В результате существенного роста размера XML-данных появилась острая необходимость в создании эффективных и удобных методов извлечения данных. Это стало предпосылкой для появления таких языков, как Lorel, Quilt, XQL, XML-QL, XPath и XQuery. Наибольшую популярность среди них получили XQuery и его подмножество XPath, являющиеся результатом работы XML Query Working Group в World Wide Web Consortium (W3C). Первая часть первой главы диссертации посвящена краткому описанию модели данных XML и языка запросов XQuery.

Вторая часть посвящена основным принципам оптимизации. Для одного и того же запроса существуют различные способы выполнения (физические планы), вырабатывающие одинаковые результаты, но значительно различающиеся по стоимости выполнения. Эти планы формируют пространство поиска. Задача оптимизации состоит в том, чтобы выбрать среди них обладающий наименьшей стоимостью. При этом, в качестве метрики стоимости обычно используют приблизительную оценку времени выполнения плана. Задача поиска оптимального плана относится к классу задач дискретного математического программирования. Существует множество алгоритмов для решения таких задач, например, метод ветвей и границ, стохастический поиск и др. Во второй части приводится краткое описание

основных компонент оптимизатора и краткий обзор наиболее распространённых алгоритмов, используемых для поиска или построения оптимальных планов. Несмотря на то, что эти алгоритмы были разработаны в контексте оптимизации реляционных запросов, они являются универсальными и могут быть использованы для оптимизации XML-запросов.

В третьей части приводится обзор наиболее важных алгоритмов, используемых при выполнении запросов, к которым относятся универсальные алгоритмы сортировки и хэширования для больших объёмов данных, альтернативные алгоритмы для операций соединения и некоторые алгоритмы реализации операций, специфичных для XML баз данных.

Четвёртая часть посвящена алгебрам запросов. Алгебра является одной из наиболее важных компонент оптимизатора, формируя пространство поиска планов, эквивалентных исходному. Наличие в алгебре операций, допускающих эффективную реализацию, и достаточно большого количества эквивалентных трансформаций потенциально создают условия для построения плана лучшего качества. Для реляционных баз данных существует общепринятая алгебра, чего нельзя сказать об XML базах данных. Это обусловлено относительной новизной данной области и, особенно, использованием различных подходов к организации баз данных (XML-прирождённые или XML-приспособленные СУБД).

В четвёртой части первой главы на основе анализа существующих XML-алгебр формулируются критерии, которым должна удовлетворять алгебра, необходимая для построения высококачественного оптимизатора, использующего стоимостные оценки:

1. Операции алгебры определены над структурами, позволяющими группировать элементы.

2. Все операции алгебры атомарны и определены над множествами кортежей (ST-операции). Результат вычисляется только на основе входных множеств кортежей и может быть использован в качестве операндов других операций.

3. Наличие тождеств, позволяющих преобразовывать операции со слож-

ными предикатами к комбинациям операций с простыми предикатами.

4. Наличие тождеств, позволяющих изменять порядок следования операций, представляющих *xpath* и *FLWOR*-конструкции.

Также в данном разделе демонстрируется, что существующие XML-алгебры не в полной мере удовлетворяют этим требованиям.

Во **второй главе** представлена новая XQuery-алгебра, именуемая XAnswer, разработанная автором и показано, что она удовлетворяет обозначенным требованиям. В первой и второй частях описаны структуры данных алгебры, а также определены основные операции, формирующие базис алгебры.

Операции XAnswer определены над структурой, называемой *подборкой (envelop).*

Определение 1. *Подборка* ($< he|be|re >$) *представляет собой тройку из заголовка* (he)*, тела* (be) *и результирующего атрибута* (re)*. Заголовок* (he) *- это неупорядоченное множество уникальных (в рамках этого заголовка) имен, называемых атрибутами. Результирующий атрибут* (re) *- это некоторый атрибут заголовка. Тело подборки - это упорядоченное множество кортежей* (τ)*, а каждый кортеж - неупорядоченное множество пар* (a, v)*, по одной паре для каждого* $a \in he$*;* v *- либо элемент (то есть атомарное значение или узел), либо последовательность элементов* (ς)*.*

Базис алгебры составляют 11 основных операций:

1. Вычисление функции (f_f): $f_f(< h|\tau(e_1) \dots \tau(e_n)|r >) =$
   $=< h, i = nextId()|\tau(e_1, f(e_1)) \dots \tau(e_n, f(e_n))|i >$.

2. Выборка (σ_{pr}): $\sigma_{pr}(< h|\tau(e_1) \dots \tau(e_n)|r >) =< h|\tau(e'_1) \dots |r >$, где $(e')_i \subseteq (e)_i$ и $pr(e_i) = true$. В качестве предиката pr может быть любое логическое выражение.

3. Проекция ($\pi_{h'}$): $\pi_{h'}(< h|\tau(e_1)\dots\tau(e_n)|r >) =< h'|\tau(e_1\ |_{h'})\dots|r\ |_{h'}>$, где $h' \subseteq h$.

4. Сортировка ($sort_{h'}$): $sort_{h'}(< h|\tau(e_1)\dots\tau(e_n)|r >) =$
$=< h'|\tau(e'_1)\dots|r >$, где $(e')_i = Sort_{h'}(e)_i$.

5. Нумерация ($index_i$): $index_i(< h|\tau(e_1)\dots\tau(e_n)|r >) =$
$=< h, i|\tau(e_1, 1)\dots\tau(e_n, n)|r > .$

6. Копирование (dup_{h_i}): $dup_{h_i}(< h|\tau(e_1)\dots\tau(e_n)|r >) =$
$=< h, nextId()|\tau(e_1, e_1\ |_{h_i})\dots|r >$, где $h_i \in h$.

7. Объединение ($\cup$): $< h|\tau(e_1)\dots|r > \cup < h|\tau(e'_1)\dots|r >=$
$=< h|\tau(e_1)\dots\tau(e_n), \tau(e'_1)\dots\tau(e'_m)|r > .$

8. прямое произведение ($\times$): $< h|\tau(e_1)\dots|r > \times < h'|\tau(e'_1)\dots|r' >=$
$=< h, h'|\tau(e_1, e'_1)\dots\tau(e_1, e'_n), \tau(e_2, e'_1)\dots|r' > .$

9. Левое внешнее соединение ($\bowtie^l_{pr}$): $< h|\tau(e_1)\dots|r > \bowtie^l_{pr}$
$< h'|\tau(e'_1)\dots|r' >=< h, h'|\tau(e_1, e''_1)\dots\tau(e_1, e''_n), \tau(e_2, e''_1)\dots|r' >$, где $e''_i = e'_i$, если $pr(e_i, e'_i) = true$, иначе $e''_i = ()$.

10. Группировка ($nest_{h'}$): $nest_{h'}(< h|\tau(e_1)\dots|r >) =$
$=< h|\tau(p_{i_1}\ |_{h'}, s^{l_1}_{i_1}\dots s^{l_m}_{i_1})\dots|r >$, где $h' \subset h$, p_{i_j} - уникальные кортежи $\pi_{h'}(E)$, $s^{l_k}_{i_j} = \varsigma(\pi_{l_k}(\sigma_{e|_{h'}=p_{i_j}|_{h'}}(E))), \forall l_k \in h \backslash h'$.

11. Операции разгруппировки ($unnest_{h'}$):
$unnest_{h_i}(< h|\tau(e^b_1, s^i_1, e^e_1)\dots|r >) =$
$=< h|\tau(e^b_1, v^{i,1}_1, e^e_1), \tau(e^b_1, v^{i,1}_2, e^e_1)\dots|r >$, где $h_i \in h$, $s^i_j = \varsigma(v^{i,j}_1\dots v^{i,j}_m)$.

Основные тождества алгебраических операций, определяющие пространство эквивалентных планов (пространство поиска), описаны и доказаны в третьей части.

В четвёртой части описывается процесс построения алгебраических выражений для запросов, содержащих основные конструкции XQuery, не

включая рекурсивные функции. Процесс состоит из двух фаз. На первой фазе, называемой нормализацией, производится преобразование XQuery-запроса в эквивалентный запрос, но удовлетворяющий определённым требованиям. На второй фазе (фазе трансляции) производится непосредственное построение алгебраического выражения, путём последовательного обхода конструкций нормализованного запроса в порядке их следования и применения описанных правил для каждой из них.

В пятой части главы описываются оптимизации, позволяющие повысить качество планов, получаемых в результате построения. Повышение качества достигается за счёт переупорядочивания, исключения и замещения избыточных операций. Целью переупорядочивания является уменьшение размеров промежуточных результатов за счёт выполнения операций, сокращающих размеры результатов, как можно раньше (опускание селективных операций). Учитывая алгебраические свойства, опускание селективных операций зачастую возможно только наряду с опусканием других, связанных с ними. Для этого предложен алгоритм, позволяющий переупорядочивать (опускать) *блоки операций*.

Для формализации преобразований вводится понятие *графа плана запроса*, с помощью которого, в свою очередь, определяются *блоки операций*, являющихся ключевым элементом при оптимизации. Вершинами графа являются операции, а дуги соединяют операции с операндами, имея направление от первых к последним.

Определение 2. Блок операций - *это подграф, который имеет не более одной вершины (*результирующей вершины*), соединённой входящими дугами с вершинами вне подграфа, а все вершины, имеющие исходящие дуги, которые ведут вне подграфа, соединены этими дугами с единственной вершиной. Такие вершины называются* входными вершинами*.*

Если блок оперирует с атрибутами, которые были добавлены в заголовок результирующей подборки другого блока, такой блок *зависит* от последнего. Блоки *независимы*, если они не *зависят* друг от друга. Блоки операций, соответствующие *for*, *let*, *where* и др. конструкциям, будем именовать *блоками конструкций*. Заметим, что два *блока конструкций* неза-

висимы, если одна конструкция нормализованного запроса не зависит от переменной, определённой в другой.

Алгоритм опускания блоков заключается в следующем. Представим план $P = B_s(B_r)$, где B_s и B_r блоки операций и B_s - блок, который требуется опустить.

1. Найти *блок конструкции* B_m, $B_r = B_e(B_m(B_b))$: B_s зависит от B_m. Если такого B_m нет, тогда $B_e = B_r$.

2. Рекурсивно выполнить алгоритм для B_m, получив в результате B'_m,B'_b,B'_e.

3. Поменять местами B_s и B'_e, получив: $P = B'_e(B_s(B'_m(B'_b)))$.

Частным случаем преобразований, направленных на опускание блоков операций, является поддержка ленивых вычислений, которые необходимы для эффективной обработки кванторных выражений, позиционных предикатов, условий в where, if-then-else и typeswitch конструкциях. Эти оптимизации имеют специфику, связанную со структурой полученного в результате построения плана, и так же описаны в пятой части второй главы.

Третья глава описывает предложенный алгоритм поиска оптимального плана. Чрезмерно широкое пространство допустимых планов существенно усложняет задачу поиска оптимального или субоптимального плана, в особенности для сложных запросов. Можно ожидать, что при использовании XQuery доля сложных запросов будет выше, чем в системах, использующих SQL, поскольку выразительные средства XQuery, такие, как навигационные выражения а также вложенные for- и let-конструкции позволяют скрыть фактическую сложность реализации.

Сложность задачи оптимизации можно значительно понизить, преобразовав задачу в блочную, так как сложность оптимизации каждого из блоков существенно ниже, чем исходная оптимизационная задача. В силу однородности модели данных и языка запросов выделение блоков в реляционных системах практически невозможно и, поэтому, блочные алгоритмы оптимизации неприменимы и ранее не рассматривались в контексте оптимизации запросов.

В первой части главы вводятся основные понятия и показываестя, что задача оптимизации может быть решена поблочно, если пространство эквивалентных алгебраических преобразований запроса удовлетворяет определённым условиям. Для формализации блочной задачи оптимизации и ограничений на пространство поиска вводится граф классов частичных планов.

Рассмотрим множество V классов эквивалентности частичных планов для некоторого запроса. На множестве V можно определить структуру графа: пусть $v \in V$ - некоторый класс эквивалентности, $p \in v : p = \alpha(p1, p2, \ldots, ps)$ - некоторый представитель этого класса, а $p_i \in v_i$ - частичные планы. Тогда в графе содержатся дуги $v \to v_i$.

Вершина, соответствующая полному плану выполнения запроса, имеет только выходящие дуги. Любому плану соответствует некоторый набор путей в этом графе, начинающихся в корневой вершине и заканчивающихся в классах операций выборки хранимых данных, из которых дуги выходить не могут. Заметим, что каждой вершине из V соответствует некоторый запрос (не обязательно являющийся подзапросом исходного запроса).

Определение 3. *Подмножество $B \subset V$ называется* блоком графа*, если существует вершина $v_B \in B$ такая, что для любой вершины $b \in B$ любой путь, проходящий через b, проходит также через v_B.*

Важность понятия блока графа связана с тем, что любой план, содержащий какую-нибудь вершину из блока, обязательно содержит вершину v_B, поэтому оптимизация подзапроса, соответствующего блоку, может быть выполнена в некотором смысле независимо от других частей запроса.

Неформально, блочный алгоритм состоит в применении некоторого другого алгоритма оптимизации по отдельности к разным частям запроса. При этом, не имеет значения, применяется ли один и тот же алгоритм для всех блоков или нет. Также не имеет значения, является ли этот алгоритм точным (например, алгоритмом динамического программирования или ветвей и границ) или приближенным (основанным на эвристиках или стохастическим). Конечно, качество получаемого плана будет зависеть от используемых алгоритмов, но это не оказывает влияния на принципиаль-

ную схему блочного алгоритма.

Во второй части третьей главы приводится описание блочного алгоритма. Неформально, блочный алгоритм состоит в применении некоторого другого алгоритма оптимизации по отдельности к разным частям запроса.

Предположим, что в пространстве планов выделены некоторые блоки $B_1, B_2, \ldots B_m$. Стоит отметить, что блоки могут быть как листовыми, то есть не имеющими вершин, связанных с вершинами других блоков исходящими дугами, так и промежуточными, обладающими такими вершинами.

Тогда для решения задачи оптимизации может быть применен следующий алгоритм:

1. Найти (суб)оптимальный план для каждого блока, применяя выбранный алгоритм оптимизации. Возможно, что при вычислении блока используются подвыражения, оценки стоимости и кардинальности которых еще не вычислены. В таких случаях используются грубые оценки для подвыражений. Например, можно применить оценки, полученные по произвольному плану из класса эквивалентности.

2. Выполнить алгоритм оптимизации для исходного запроса, в котором каждый блок заменен на одну неделимую операцию (с оценками стоимости, полученными при оптимизации блоков).

3. Если исчерпан лимит времени на оптимизацию, закончить работу.

4. Если в результате шага 2 изменились оценки стоимости операций, на основе которых выполнялась оптимизация промежуточных блоков, повторить шаг 1 для тех блоков, для которых изменились оценки, и перейти к шагу 2, иначе (если оценки не изменились) закончить работу.

Корректность этого алгоритма непосредственно следует из леммы 1.

Лемма 1. *Пусть p - оптимальный план и существует путь, построенный по плану p, проходящий через вершину $v \in V$. Тогда подплан p_v плана p является оптимальным планом для v.*

Поиск блоков в пространстве планов сам по себе вычислительно сложен, поэтому предлагается использовать априорное выделение блоков, соответствующих подвыражениям специального вида в исходном запросе. Для того, чтобы эти подзапросы действительно образовывали блоки, необходимо ввести некоторые ограничения на применение алгебраических соотношений. По определению, блок характеризуется отсутствием путей, ведущих в него не через его корневую вершину, то есть отсутствием дуг, ведущих в другие вершины блока извне. Поскольку мы будем выделять блоки априорно, запрет на использование таких дуг эквивалентен запрету на использование выражений, лежащих вне блока и использующих его внутренние вершины, но не корень. Исключение таких выражений, в свою очередь, означает запрет на использование эквивалентных трансформаций, приводящих к появлению нежелательных дуг.

В третьей части главы описывается применение блочного алгоритма для оптимизации XQuery-запросов с использованием алгебры XAnswer. Предлагается выделять в качестве блоков множества планов, соответствующих навигационным выражениям *let* и *for*-конструкций нормализованного запроса. В терминах алгебры такое выделение блоков означает запрет на ассоциативное преобразование между операциями соединения, имеющими предикаты разной природы (то есть структурный предикат и предикат по значению). Далее показывается что запрет таких преобразований не приводит к потере оптимального плана для широкого класса запросов. В остальных случаях это оказывает влияние на выделение блоков, что учитывается на следующих итерациях алгоритма, тем самым, сохраняя возможность получения субоптимальных планов высокого качества.

Четвертая глава посвящена описанию экспериментов. Для экспериментальной верификации предложенной техники выполнения запросов был реализован прототип исполнителя на основе СУБД eXist. В рамках прототипа были реализованы и использованы следующие элементы: синтаксический анализатор с открытым исходным кодом (eXist), преобразователь, транслятор, планировщик без стоимостной оптимизации, исполнитель и драйвер для хранилища eXist. Общая архитектура исполнителя запросов описывается в первой части главы.

Некоторые из подходов, основанных на существующих XQuery-алгебрах, за счёт применения специальных оптимизаций позволяют использовать атомарные операции над множествами (ST) вместо итеративных (NT), для относительно простых вложенных подзапросов. Это, в свою очередь, может значительно повысить производительность за счёт возможности использования алгоритма хэширующего соединения вместо вложенных циклов. Тем не менее, они не могут использовать ST-вычисления вместо NT для тех запросов, которые содержат кванторы, сложные условные *(if-then-else, typeswitch)* или *where*-конструкции с вложенными подзапросами. Такие запросы будем называть *усложнёнными*. При этом, при использовании ST-операций для вычисления *усложнённых запросов*, появляется необходимость поддержки ленивых вычислений.

Усложнённые запросы часто возникают, например, в задачах анализа больших объёмов XML-данных, имея в качестве операндов условных выражений и логических операций выражения большой вычислительной сложности. Одним из примеров таких задач является анализ информации в Wikipedia, представленной в виде XML, объёмом более 20 гигабайт, и содержащей несколько миллионов связанных друг с другом статей сложной структуры.

В экспериментах анализируется производительность различных подходов выполнения запросов: основанных на использовании алгебры W3C, генерируемые в СУБД eXist; использующие ST-операции наряду с NT-операциями для условных выражений (аналогично подходам, основанным на существующих XQuery-алгебрах; использование алгебры XAnswer и предложенных оптимизаций. Для экспериментов использовался эталонный набор данных и запросов XMark, включая дополнительно усложнённые запросы.

Показано, что планы для относительно простых запросов, получаемые при помощи XAnswer, сопоставимы по производительности с планами, которые можно получить, используя существующие алгебры, а для сложных запросов - значительно превосходят их.

Заключение содержит список основных результатов, полученных в работе.

Публикации автора по теме диссертации

Статьи в журналах, рекомендованных ВАК:

1. Лукичёв М.С. Оценка Селективности XPath-запросов в XML-СУБД. *Программные продукты и системы, Выпуск номер 2.* — Тверь, июнь 2009. — стр. 20-22.

Другие публикации:

2. M.Lukichev, D.Barashev. XML Query Algebra for Cost-based optimization.*Spring Young Researchers Colloquium on Databases and Information Systems (SYRCoDIS).* — Москва, июнь 2007. — стр. 17-26.

3. Y.Soldak, M.Lukichev. Enabling XPath Optional Axes Cardinality Estimation Using Path Synopses. *Proceedings of the 12-th East-European Conference on Advances in Databases and Information Systems.* — Пори, Финляндия, сентябрь 2008. — стр. 279-294.