# АЛГОРИТМЫ ЭФФЕКТИВНОГО ВЫЧИСЛЕНИЯ КОНЪЮНКТИВНЫХ РЕГУЛЯРНЫХ ПУТЕВЫХ ЗАПРОСОВ

С. А. Афонин
*НИИ механики МГУ, Москва, Россия*
e-mail: serg@msu.ru

A problem of the processing of conjunctive regular path queries over semistructured databases (an NP complete problem) is considered. A number of heuristics for conjunctive query processing are introduced and an algorithm for query evaluation plan construction is proposed. Experimental results show the efficiency of the heuristics and corresponding query evaluation plans.

## Введение

Полуструктурированными называют данные с неоднородной, часто изменяющейся или заранее не известной структурой. Такие данные естественным образом возникают при децентрализованном управлении информацией, например, в сети Интернет или в крупных корпоративных или государственных информационных системах, отдельные компоненты которых (сайты или ведомственные информационные системы) могут содержать строго структурированные данные (общая структура HTML и единая структура базы данных), однако при объединении информации из нескольких таких источников данные, описывающие одни и те же объекты реального мира, могут существенно различаться структурой. Необходимость поиска информации одновременно в нескольких источниках приводит к задаче поиска данных с неоднородной структурой.

В работе 2005 года [1] было предложено еще более общее направление исследований — переход от баз данных к "пространствам данных". Авторы этой работы рассматривают задачу поиска разнородной информации в очень широкой постановке (в качестве примера используется поиск по всем файлам рабочей станции — электронным таблицам, текстовым документам, локальным базам данных и т. д.) и предлагают "программу исследований", направленную на достижение этой цели. Одним из таких направлений является обработка запросов к полуструктурированным данным на основе графовой модели представления данных.

Основными математическими моделями представления полуструктурированных данных являются ориентированные графы с помеченными ребрами, помеченные деревья и деревья с упорядоченными элементами [2]. Для каждой модели данных были предложены соответствующие языки запросов (регулярные путевые запросы, XPath, XQuery,

Datalog). Следует отметить, что эффективность применения той или иной модели данных зависит от предметной области, в которой будет использоваться система управления полуструктурированными данными. Например, модель данных, основанная на упорядоченных деревьях, которая в настоящее время находится в центре внимания специалистов по базам данных, ориентирована в первую очередь на работу с XML-документами. Для других задач, таких как поиск текстов с учетом онтологий [3 – 5], управление содержанием Web-сайтов или интеграция данных [6, 7], эта модель менее эффективна, поскольку в этих приложениях естественным образом возникают графовые структуры данных.

В данной работе рассматривается задача вычисления конъюнктивных регулярных путевых запросов при графовом педставлении данных. Язык регулярных путевых запросов по своей выразительной силе занимает промежуточное положение между относительно простыми языками типа XPath и такими гибкими языками, как Datalog. Следует отметить, что задача вычисления рассматриваемых далее запросов является NP-сложной, однако это связано с использованием *конъюнкции* регулярных путевых запросов, а не с наличием ограниченной формой рекурсии (итерацией Клини) в регулярных путевых запросах.

# 1. Конъюнктивные регулярные путевые запросы

Для графового представления полуструктурированных данных стандартом де-факто является так называемая Object Exchange Model, или OEM-модель, которая разрабатывалась для обеспечения унифицированного представления сложных информационных объектов для систем интеграции данных. В соответствии с этой моделью сложные структуры данных представляются в виде ориентированного графа с помеченными ребрами. Вершины графа соответствуют элементам первоначальной структуры данных, а ребра представляют отношения между ними. Формально *базой (полуструктурированных) данных* будем называть ориентированный граф с помеченными ребрами $\mathcal{B} = \langle V, \Sigma, E \rangle$, где $V$ — множество вершин графа; $\Sigma$ — конечное множество меток ребер; $E \subseteq V \times \Sigma \times V$ — множество $\Sigma$-помеченных ребер.

Представление данных в виде ориентированного графа с помеченными ребрами достаточно универсально. В рамках этой модели естественным образом реализуются конструкторы множества и кортежа, а данные, представленные в таком виде, являются *самоопределяющими*, т. е. содержат не только собственно данные, но и их структуру.

В рассматриваемой модели данных ***язык запросов*** основывается на понятии *регулярного путевого выражения*. Регулярные путевые выражения — это по сути регулярные языки, а пара вершин базы данных удовлетворяет регулярному путевому выражению, если между этими вершинами находится по крайней мере один ориентированный путь, метки которого образуют слово из этого языка (мы предполагаем известными основные понятия теории автоматов и формальных языков [8, 9]). Регулярное путевое выражение, таким образом, определяет бинарное отношение на множестве пар вершин базы данных. В качестве языка запросов будем рассматривать конъюнктивные регулярные путевые запросы.

*Конъюнктивным регулярным путевым запросом* (CRPQ-запросом) называется ориентированный помеченный граф $Q = \langle X, \mathrm{Reg}(\Delta), E_Q \rangle$, где $X$ — множество вершин запроса (переменных); $\mathrm{Reg}(\Delta)$ — множество всех регулярных языков над алфавитом $\Delta$; $E_Q \subset X \times \mathrm{Reg}(\Delta) \times X$ — множество помеченных ребер. Без ограничения общности можно считать, что алфавиты базы данных и запроса совпадают.

***Вычисление*** CRPQ-запроса состоит в нахождении всех отображений вершин запроса $Q$ в вершины базы данных $\mathcal{B}$, которые удовлетворяют следующим условиям:

1) образы различных вершин запроса различаются;

2) смежные вершины запроса, соединенные ребром с меткой $R \in \mathrm{Reg}(\Sigma)$, отображаются в вершины базы, соединенные по крайней мере одним путем, метки которого образуют слово из языка $L(R)$.

Задача вычисления CRPQ-запроса может быть за полиномиальное время сведена к задаче поиска подграфа в ориентированном мультиграфе с помеченными ребрами. Действительно, пусть ребра запроса $Q$ помечены языками $R_1, R_2, \ldots, R_k$. Введем вспомогательный алфавит $B = \{b_1, \ldots, b_k\}$ и построим по исходной базе данных ориентированный граф $G$ следующим образом. Вершинами $G$ служат вершины исходной базы данных, а пара вершин $u$ и $v$ соединена ребром с меткой $b_i$ тогда и только тогда, когда в исходной базе между этими вершинами есть путь с метками из языка $R_i$. Поскольку пара вершин базы данных может принадлежать результату нескольких элементарных запросов, граф $G$ в общем случае является мультиграфом.

Представим теперь запрос $Q$ в виде графа $H$, множество вершин которого совпадает с вершинами $Q$, а ребра помечены символами алфавита $B$. Для этого заменим регулярные языки на соответствующие символы алфавита $B$: вместо языка $R_1$ подставим $b_1$, вместо $R_2$ — $b_2$ и т. д. Очевидно, что вычисление запроса эквивалентно поиску всех подграфов графа $G$, изоморфных графу $H$. Отметим, что в частном случае, когда алфавит $\Sigma$ содержит один символ, а все встречающиеся в запросе регулярные выражения определяют однобуквенные языки, задача вычисления CRPQ-запроса в точности совпадает с задачей изоморфного вложения графа, которая относится к классу NP-полных [10]. Это означает, что вычисление CRPQ-запросов является NP-сложной задачей.

Сведение задачи вычисления запроса к задаче поиска подграфа позволяет использовать алгоритмы поиска подграфа для вычисления CRPQ-запроса. Наиболее эффективным алгоритмом поиска подграфа считается алгоритм Ульмана [11], реализующий метод перебора с отсечением. Этот алгоритм выбран в качестве основы для параллельного алгоритма вычисления CRPQ-запросов [12], который был реализован с использованием системы динамического распараллеливания программ OpenTS [13, 14].

## 2. Эвристические методы вычисления запроса

Основными стратегиями вычисления CRPQ-запросов являются *исчерпывающий поиск* и *метод слияния результатов* вычисления элементарных запросов. Первая стратегия — по сути алгоритм поиска подграфа. Вторая стратегия предполагает вычисление элементарных запросов и последующее слияние результатов. Как было показано в [15], эффективность применения данных стратегий существенно зависит от структуры вычисляемого запроса. Далее в данном разделе описываются различные эвристические методы повышения эффективности алгоритма вычисления конъюнктивных запросов, основанного на алгоритме поиска подграфа.

**Порядок просмотра вершин запроса.** Очевидно, что эффективность алгоритма вычисления запроса зависит от порядка обхода вершин запроса. Пусть, например, запрос представляется ациклическим графом. В этом случае можно ожидать, что порядок просмотра вершин, соответствующий естественному отношению частичного порядка, будет эффективнее порядка, при котором сначала просматриваются листовые вершины.

Последовательность просмотра вершин запроса можно определять либо статически, на основании структуры запроса, либо динамически, учитывая на каждом шаге алгоритма мощность множества допустимых образов. При статическом подходе порядок просмотра вершин определяется до начала вычисления запроса на основании числа входящих и исходящих ребер вершины запроса. Однако этот метод не учитывает специфики конкретного экземпляра базы данных, относительно которого вычисляется запрос. Простой эвристикой является выбор на каждом шаге вершины, множество допустимых образов которой минимально. Если мощность этих множеств для некоторых вершин окажется "примерно равной", то выбор можно осуществлять либо случайным образом, либо на основе статического порядка этих вершин.

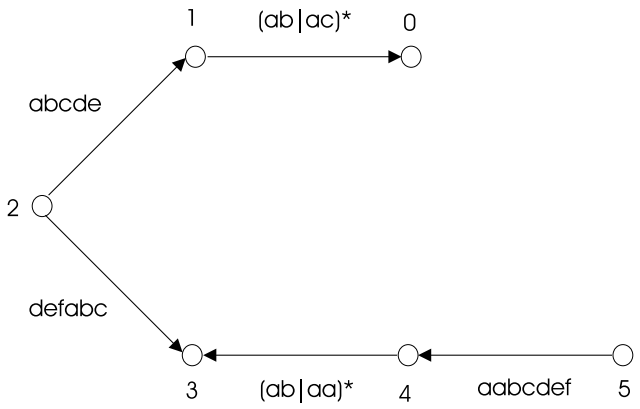
**"Обращение" ребер.** Предположим, что в запросе имеется ребро между вершинами $x$ и $y$, помеченное языком $R$. Если образ вершины $x$ зафиксирован, то можно построить множество допустимых образов $y$. С другой стороны, данное ребро можно просматривать в обратном направлении: сначала выбрать образ вершины $y$, а потом вычислить множество допустимых образов вершины $x$ (при этом ребра базы данных следует просматривать в обратном направлении). Если язык $R$ содержит большое число различных префиксов (например, $R = \Sigma^* abc$), то просмотр такого ребра в обратном порядке может оказаться более эффективным.

**Порядок обхода исходящих ребер.** В том случае, если на основании какой-либо дополнительной информации можно предположить, что вычисление элементарного запроса, соответствующего исходящему ребру текущей вершины, приведет к получению множества большой мощности, данное ребро можно временно не рассматривать. Приведем следующий пример.

Предположим, что исходящему ребру текущей вершины $x$ запроса соответствует язык $R$, задаваемый регулярным выражением $b((a+b)a^*(bb)^*)^*$. Пусть также известно, что каждая вершина базы данных имеет исходящее ребро, помеченное символами $a$ или $b$ с вероятностью 0.9. В такой ситуации можно предположить, что для любой вершины $v$ базы данных множество вершин, достижимых из $v$ по словам из языка $R$, будет содержать почти все вершины базы. Вычисление данного запроса трудоемкий процесс, так как требует обхода всей базы. Однако это ребро не дает существенной информации для уточнения множества допустимых образов вершин $x$ и $y$. В такой ситуации вычисление данного ребра можно отложить.

**Наложение локальных ограничений.** Приближение множества допустимых образов вершины запроса может быть получено проверкой *локальных ограничений*. Предположим, что чем больше исходящих ребер имеет вершина запроса, тем меньше множество ее допустимых образов. Минимально необходимое число исходящих ребер, которые должна иметь вершина базы данных, соответствующая данной вершине запроса, можно оценить путем рассмотрения множества однобуквенных префиксов языков, приписанных исходящим ребрам вершины запроса. Например, если вершина запроса имеет два исходящих ребра, метками которых являются языки $R_1 = b + ba$ и $R_2 = a + cb$, то ее образом может быть вершина, которая имеет как минимум два исходящих ребра (с метками $a$ и $b$). Локальные ограничения могут использоваться как дополнение статического метода выбора порядка просмотра вершин запроса.

**Вычисление запроса с учетом эвристик.** Для экспериментального сравнения эффективности применения описанных выше эвристических методов вычисления конъюнктивных запросов составлены тестовый запрос, представленный на рисунке, и шесть планов его вычисления, которые определяют различный порядок обхода вершин запроса и




Тестовый запрос для оценки эффективности эвристик.

направление прохождения ребер.

Результаты тестирования приведены в таблице. База данных представляла случайный граф и содержала 100 вершин. План, соответствующий просмотру вершин запроса в порядке возрастания их номеров, обозначен буквой $D$. Выполнение запроса с обращением всех ребер соответствует столбцу $R$, обращение некоторых ребер — столбцу $DR$. Различные планы вычисления запросов (изменяются порядок просмотра вершин и направления просмотра ребер) обозначены символами $P_1, \ldots, P_6$, число процессоров — через $N$.

Анализируя полученные результаты, можно сделать следующие выводы. Во-первых, порядок и направление обхода вершин существенно влияют на эффективность вычисления запроса. В рассмотренном случае применение эвристик повысило производительность более чем на два порядка.

Во-вторых, для некоторых планов вычисления запроса наблюдается снижение производительности при увеличении количества вычислительных узлов. Это может быть связано с тем, что ответы равномерно распределены в дереве перебора и при реализации перебора с отсечением на каждый вычислительный узел приходятся примерно равные фрагменты дерева, что ведет к равномерной загрузке и, как следствие, к росту производительности при увеличении числа вычислительных узлов. При оптимальном порядке обхода отсечение дерева перебора происходит на ранних этапах работы алгоритма и коммуникационные расходы становятся соизмеримы со временем счета.

Таким образом, знание “точного” плана выполнения запроса позволяет эффективно вычислять запрос на одном узле. В то же время увеличение числа вычислительных узлов приводит к росту производительности в случае неоптимальных планов выполнения запроса. Следовательно, основная цель применения эвристических методов заключается в построении приближения оптимального плана. Ошибка приближения может быть компенсирована за счет увеличения количества вычислительных узлов.

Среднее время вычисления, с, тестовых запросов при различных планах

| $N$ | $D$ | $R$ | $DR$ | $P_1$ | $P_2$ | $P_3$ | $P_4$ | $P_5$ | $P_6$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1069 | 1565 | 1237 | 21.8 | 77.1 | 5.8 | 60.4 | 6.4 | 1080.2 |
| 2 | 696 | 1535 | 805 | 20.0 | 51.8 | 5.2 | 38.2 | 5.1 | 569.4 |
| 4 | 429 | 1569 | 519 | 19.9 | 44.3 | 4.8 | 26.0 | 4.5 | 302.7 |
| 8 | 348 | 1572 | 416 | 24.2 | 52.1 | 7.9 | 36.1 | 8.1 | 213.1 |

# 3. Построение плана вычисления

Алгоритмы построения плана запроса можно разделить на следующие три категории:

— *синтаксические*: оптимальная последовательность выбирается только на основании структуры запроса;

— *стоимостные*: при выборе плана учитываются статистические характеристики конкретного экземпляра базы данных;

— *динамические*: план выполнения запроса корректируется в процессе его вычисления.

В данной работе предлагается алгоритм построения оптимальной последовательности прохождения вершин и ребер запроса, который основан на произвольной количественной характеристике сложности элементарного запроса. При использовании синтаксической оценки сложности элементарного запроса этот алгоритм приводит к построению синтаксического плана. Если же оценка сложности элементарного запроса учитывает статистику базы данных, то получающийся план является стоимостным.

*Ограничениями* вершины $x$ запроса $Q$ будем называть набор ребер запроса, инцидентных вершине $x$. Порождающую вершину ограничений $\pi$ будем обозначать как $x(\pi)$, а множество ребер — как $E(\pi)$. *Планом* вычисления запроса $Q$ называется такая последовательность ограничений, которая содержит все ребра запроса.

Сложность (эффективность) плана основывается на следующих интуитивных соображениях. Очевидно, что число вершин базы данных, удовлетворяющих заданным ограничениям, существенно зависит от числа ребер и структуры соответствующих регулярных языков. Ограничения назовем информативными, если им удовлетворяет незначительное число вершин базы данных. Аналогично, если в запросе есть ребро $(x, R, y)$, то при выборе образа вершины $x$ число допустимых образов вершины $y$ зависит от структуры языка $R$. Будем называть ребро $(x, R, y)$ запроса информативным, если мощность множества допустимых образов вершины $y$ зависит от выбора образа $x$. При построении плана вычисления запроса следует использовать следующие критерии:

1) вершины запроса, ограничения которых имеют наибольшую информативность, должны проверяться как можно раньше;

2) при выборе смежной вершины запроса предпочтение следует отдавать вершинам с наиболее информативными входящими ребрами (из ранее просмотренных вершин запроса).

Далее предлагается формализация понятий информативности ограничений и сложности плана.

Предположим, что заданы функции оценки информативности ограничений $I(\pi)$ и информативности ребра (регулярного языка) $I(L)$. Обозначим через $\pi(z)$ первое вхождение вершины $z$ запроса в плане $\Pi$. Определим на множестве вершин запроса отношение частичного порядка $y \prec_\Pi z$, которое означает, что в запросе есть ребро $(y, z)$ и вершина $y$ встречается в плане раньше вершины $z$. Обозначим через $L_{xy}$ регулярный язык, приписанный в запросе $Q$ ребру между вершинами $x$ и $y$, и определим информативность первого посещения вершины $z$ запроса в рамках плана $\Pi$ как

$$I_\Pi(z) = I(\pi(z)) \left( 1 + \sum_{y : y \prec_\Pi z} I(\pi(y)) \frac{I(L_{yz}) + I(L_{yz}^R)}{2} \right).$$

Поясним смысл данной формулы. Первый сомножитель — информативность ограничений вершины. Чем меньше это значение, тем меньше информативность вершины. Второй

сомножитель — весовой коэффициент, придающий больший вес тем вершинам, которые являются смежными с ранее рассмотренными вершинами запроса.

*Сложностью плана* $\Pi = \langle \pi_1, \ldots, \pi_K \rangle$ назовем величину

$$\mathrm{Cost}(\Pi) = \left[ \sum_{i=1}^{K} \exp(-i) I_{\Pi}(x(i)) \right]^{-1}.$$

План $\Pi$ назовем *оптимальным*, если для любого другого плана $\Pi'$ справедливо неравенство $\mathrm{Cost}(\Pi) \leqslant \mathrm{Cost}(\Pi')$.

Домножение на убывающую функцию $\exp(-i)$ придает больший вес первым элементам плана, а значит, планы, в которых первыми рассматриваются вершины с более информативными ограничениями, имеют меньшую сложность.

Так как число возможных планов выполнения запроса $Q$ конечно, задача нахождения оптимального плана алгоритмически разрешима. Для построения плана вычисления CRPQ-запроса предлагается использовать алгоритм со случайным выбором: производятся построения заданного числа случайных планов выполнения запроса и из этого множества выбирается лучший план. Построение случайного плана может выполняться по следующей схеме. Для каждого ребра запроса случайным образом определяется, в каком направлении это ребро будет проходиться при выполнении запроса. Далее генерируется случайная перестановка вершин запроса и для каждой вершины строятся ее ограничения, т. е. входящие и исходящие ребра, с учетом выбранного ранее направления.

Вернемся к задаче построения оценки информативности регулярного путевого запроса. В соответствии с изложенным выше в качестве меры сложности запроса рассмотрим вероятность того, что вершина базы данных удовлетворяет запросу. Отметим, что стандартные меры сложности регулярных языков, такие как число состояний минимального автомата, энтропия, $n$-различимость или автоматность [16], не отражают сложности регулярного запроса. Если язык $R_1$ вложен в язык $R_2$, то запрос $R_1$ более информативный, чем запрос $R_2$. Однако вложенность языков не связана с числом состояний соответствующих минимальных автоматов. Аналогичные примеры можно привести и для других мер сложности языков.

Оценку информативности заданного регулярного языка $R$ можно вычислить, предполагая, что база данных представляет собой дерево, построенное по следующему недетерминированному алгоритму:

— для любой буквы алфавита каждая вершина дерева имеет не более одного исходящего ребра, помеченного этой буквой;

— для каждой буквы $a \in \Sigma$ вероятность того, что эта вершина имеет исходящее ребро с этой меткой, есть $p(a)$.

*Сложность регулярного путевого запроса* $\mu(R)$ определим как вероятность того, что корневая вершина случайного дерева удовлетворяет запросу.

**Теорема.** *Для любого действительного числа* $\varepsilon > 0$*, любого отображения* $p : \Sigma \to [0, 1]$ *и любого регулярного языка* $R$ *значение* $\mu(R)$ *может быть вычислено с точностью* $\varepsilon$*.*

Доказательство проводится по следующей схеме. Предположим, что язык $R$ распознается детерминированным $\Sigma$-автоматом $A$ с $n$-состояниями $q_1, \ldots, q_n$, функцией переходов $\delta$ и множеством заключительных состояний $F$. Запись $\delta(q, a) = \varnothing$ будет означать, что не существует слова вида $w = au$, которое переводит состояние $q$ в некоторое заключительное состояние. Язык $R$ имеет не более $n$ различных производных, которые распознаются

автоматами $A_i, i = 1, \ldots, n$, получающимися из $A$ смещением начального состояния в состояние $q_i$. Очевидно, что производные являются регулярными языками и среди них есть язык $R$. Числа $\mu(A_i)$ удовлетворяют следующей системе соотношений:

$$\begin{cases} \mu(A_i) = 1 - \prod\limits_{\substack{a \in \Sigma \\ \delta(q,a) \neq \varnothing}} (1 - p(a)\mu(A_{\delta(q_i,a)})), & q_i \notin F, \\ \mu(A_i) = 1, & q_i \in F. \end{cases}$$

Утверждение теоремы следует из того, что отображение, которое задается данной системой, является сжимающим, а значит, имеет единственную неподвижную точку.

Одним из возможных методов учета особенностей конкретного экземпляра базы данных является вычисление частотности букв алфавита. Полученные значения можно использовать в качестве значений $p(a)$. Другая возможность состоит в подсчете частотности вхождения слов заданной длины и в построении стохастического автомата $M$, который имеет распределение вероятностей, близкое к полученному. В этом случае статистика экземпляра базы данных может быть учтена, если вместо автомата $A$ рассматривать автомат $A \times M$.

## 4. Результаты применения планов

Для запроса, представленного на рисунке, предложенный метод построения плана приводит к "почти оптимальному" плану $P_5$. При тестировании метода на случайных запросах в качестве критерия эффективности плана рассматривается число рекурсивных вызовов, которое делает алгоритм вычисления запроса при использовании заданного плана.

Случайный запрос представляет собой случайный граф (вероятность наличия ребра между двумя вершинами равнялась $2/n$, где $n$ — число вершин запроса), ребрам которого приписаны случайные регулярные языки над четырехбуквенным алфавитом. Для заданного случайного запроса строился оптимальный план его выполнения и сравнивалась эффективность оптимального плана с эффективностью случайных планов выполнения этого запроса. Для каждого запроса строилось множество случайных баз данных, для каждой из которых вычислялся запрос с использованием оптимального и нескольких случайных планов.

Результаты тестирования показали, что в некоторых случаях существуют планы выполнения запроса, которые оказываются значительно эффективнее построенного оптимального плана. Однако в среднем (в общей сложности было выполнено 200 запросов) оптимальные планы оказываются более эффективными. Отношение числа рекурсивных вызовов для оптимального плана к среднему значению этого параметра для случайных планов для полученных данных равняется 0.55461. Это означает, что оптимальные планы, построенные с использованием описанных алгоритмов, повышают эффективность алгоритма вычисления CRPQ-запросов.

## Заключение

На основании результатов анализа влияния различных методов вычисления запроса и предложенной меры сложности регулярных языков формализовано понятие плана вычисления конъюнктивного регулярного путевого запроса и предложен критерий оценки

сложности его вычисления. Результаты тестирования показали, что оптимальный с точки зрения предложенной меры сложности план выполнения запроса повышает эффективность алгоритма вычисления запроса. Приведенные данные для параллельного алгоритма вычисления запросов показывают, что использование оптимального плана вычисления запроса может снижать производительность системы при увеличении числа процессоров. Возможным направлением дальнейших исследований видится разработка методов декомпозиции исходного конъюнктивного запроса на несколько подзапросов. Подзапросы могут выполняться параллельно, а результат исходного запроса даст последующее слияние результатов подзапросов.

# Список литературы

[1] Franklin M. From databases to dataspaces: a new abstraction for information management // SIGMOD Rec. 2005. Vol. 34, N 4. P. 27–33.

[2] Гринев М. XML-модели данных, языки запросов к XML-базам данных и представление метаданных // Суперкомпьютерные вычислительно-информационные технологии в физических и химических исследованиях: Сб. лекций. Черноголовка, 2000. С. 43–84.

[3] Querying the semantic web with RQL / G. Karvounarakis, A. Magganaraki, S. Alexaki et al. // Comput. Networks. 2003. Vol. 42, N 5. P. 617–640.

[4] Васенин В.А. К разработке моделей эффективного поиска информации в сети Интернет // Сб. тр. Всерос. науч. конф. "Научный сервис в сети Интернет 2003". Абрау-Дюрсо. М.: Изд-во МГУ, 2003. С. 252–255.

[5] Афонин С.А. Поиск текстовых документов с учетом их логической структуры // XII Междунар. конф. по вычисл. механике и современным прикладным програмным системам (ВМСППС). М.: Изд-во МАИ, 2003.

[6] Васенин В.А. К созданию концепции интегрированной системы распределенных информационных ресурсов Московского государственного университета им. М.В. Ломоносова. М.: Изд-во МГУ, 2001.

[7] Vasenin V.A. To the problem of building an integrated system of university distributed information resources // Proc. of the Finnish Data Processing Week Conf. FDPW-2001, 2001. P. 152–177.

[8] Кудрявцев В.Б. Введение в теорию автоматов. М.: Наука, 1985.

[9] Трахтенброт Б.А. Конечные автоматы (поведение и синтез). М.: Наука, 1970.

[10] Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982.

[11] Ullmann J.R. An algorithm for subgraph isomorphism // J. of the ACM. 1976. Vol. 23, N 1. P. 31–42.

[12] Afonin S. Semistructured data search using dynamic parallelisation technology // Proc. of the 26th Intern. Convention MIPRO-2003, Opatija, Croatia. 2003. P. 152–157.

[13] Абрамов С.М., Васенин В.А., Мамчиц Е.Е. и др. Динамическое распараллеливание программ на базе параллельной редукции графов. Архитектура программного обеспечения новой версии Т-системы // Тр. Всерос. науч. конф. “Высокопроизводительные вычисления и их приложения”. Черноголовка, 2002. С. 261–265.

[14] Васенин В.А., Роганов В.А. GRACE: распределенные приложения в Internet // Открытые системы. 2001. Т. 5–6. С. 29–33.

[15] Афонин С.А. Стратегии вычисления регулярных путевых запросов // Информационные технологии и программирование. 2002. Т. 5, № 1. С. 9–16.

[16] Shallit J., Breitbart Y. Automaticity I: Properties of a measure of descriptional complexity // J. Computer and System Sciences. 1996. Vol. 53. P. 10–25.

*Поступила в редакцию 6 февраля 2007 г.*