УДК 519.68

ЛОКАЛЬНЫЕ ЭЛИМИНАЦИОННЫЕ АЛГОРИТМЫ ОБРАБОТКИ ЗАПРОСОВ В БАЗАХ ДАННЫХ[1]

© Щербина О.А.

University of Vienna
Fakultaet fuer Mathematik
Nordbergstr., 15, Vienna, 1090, Austria
e-mail: *oleg.shcherbina@univie.ac.at*

Abstract. The applying local elimination algorithms (LEA) for processing queries in relational databases is considered. The special features of realization of local algorithm using only a forward part are discussed.

Введение

Использование локальной информации [1] при изучении сложных дискретных систем, при разработке методов декомпозиции для решения больших разреженных дискретных задач является весьма актуальным, причем упомянутые задачи принадлежат как области дискретной оптимизации (ДО) [2], [3], так и области искусственного интеллекта (ИИ) [4], баз данных [5], линейной алгебры (мультифронтальные методы решения разреженных систем линейных уравнений [6], [7], имеющие декомпозиционный характер).

При изучении сложных дискретных систем не всегда возможно получить (или вычислить) глобальную информацию об объекте, поэтому представляет интерес получение информации об объекте, рассматривая его по частям, т.е. локально, используя предложенные *Ю.И. Журавлевым* локальные алгоритмы вычисления информации [1].

Важной особенностью локальных алгоритмов является вычисление и использование именно *локальной информации* (т.е. информации об элементах окрестности элемента) при решении разреженных дискретных задач. В [8] предложен общий класс локальных элиминационных алгоритмов вычисления информации для решения дискретных разреженных задач, позволяющих осуществлять вычисление *глобальной информации* о решении задачи с помощью *локальных вычислений* на основе анализа окрестностей элементов задачи. Локальные элиминационные алгоритмы (ЛЭА) вычисления информации относятся к графовым декомпозиционным подходам. Алгоритмическая схема ЛЭА представляет собой бесконтурный орграф, вершинами которого являются локальные подзадачи, соответствующие окрестностям элементов, а дуги – выражают информационную зависимость подзадач друг от друга. В [9] показано, что указанный орграф является *элиминационным деревом* [10]. Графовой интерпретацией ЛЭА является *элиминационная игра* (Parter [11]).

В [12] рассмотрено решение (неоптимизационной) задачи удовлетворения ограничений, структура которой задается графом взаимосвязей переменных, с помощью ЛЭА [8].

[1]Работа выполнена при финансовой поддержке австрийского фонда FWF, грант No. P20900-N13

В настоящей статье анализируются возможности применения локальных элиминационных алгоритмов к решению задач обработки запросов в реляционных базах данных (БД).

1. Основные понятия реляционной теории управления базами данных

Теория БД включает широкий круг вопросов, относящихся к теоретическим проблемам БД и системам управления БД (СУБД), в том числе создание языков запросов, теорию финитных моделей, теорию проектирования БД, теорию зависимостей и др. СУБД предназначены для организации хранения, поиска и манипулирования большими объемами данных.

1.1. **Реляционные базы данных.** Основы реляционной модели данных были впервые изложены в статье *Кодда* [13] в 1970 г. Эта работа послужила стимулом для большого количества исследований, в которых реляционная модель получила дальнейшее развитие. Сам термин «реляционное представление данных», впервые введенный *Коддом* [13], происходит от термина *relation – отношение*, поскольку в основе всей реляционной теории баз данных лежит понятие отношения, а БД в этой модели является конечным множеством отношений.

Введем необходимые понятия. Имеется конечное множество атрибутов $U = \{x_1, \ldots, x_m\}$ с соответствующими доменами $D_1, \ldots, D_m$. *Реляционной схемой* R называется подмножество U. *Схемой БД* называется множество реляционных схем $D = \{R_1, \ldots, R_n\}$. Отношением r_i с реляционной схемой R_i называется множество R_i-кортежей, где R_i-кортеж – отображение атрибутов из R_i на их домены. База данных d со схемой D – множество $\{r_1, \ldots, r_n\}$ отношений. Реляционную схему R (т.е. множество атрибутов) отношения r будем называть также *диапазоном отношения* r.

Отношение можно описать различными способами, однако в реляционных базах данных лучше всего использовать табличный способ. Отношение r может быть задано в виде таблицы, в которой каждый столбец озаглавлен *атрибутом*. Количество атрибутов называется *степенью* отношения. Фактически данными являются строки таблицы (*кортежи*). Количество кортежей отношения называется *мощностью отношения*. Порядок столбцов и строк в таблице несуществен.

Каждому отношению r степени n, определенному на диапазоне R, можно поставить в соответствие некоторый n-местный предикат $p(R)$, определяющий, будет ли кортеж принадлежать отношению (этот предикат принимает значение «истина» для всех кортежей r). Этот предикат называют *предикатом отношения*. Говорят, что кортеж f на диапазоне R удовлетворяет $p(R)$, если $p(R)$ принимает значение «истина» для этого кортежа. Это записывается в форме $f \models p(R)$. В свою очередь, каждый n-местный предикат $p(R)$ задает некоторое n-арное отношение на домене R, состоящее из всех кортежей, для которых $p(R)$ истинен, т.е. $\{f \in E_R : f \models p(R)\}$ (E_R – множество всех R-кортежей).

Таким образом, существует взаимно однозначное соответствие между n-арными отношениями и n-местными предикатами.

Если пользователю необходимо найти определенные данные из БД, он запрашивает их с помощью *запроса*, в котором выражены условия, связывающие атрибуты отношений, входящих в БД. СУБД обрабатывает запрос, находит требуемые данные и посылает их пользователю. Процесс запрашивания данных и получения результата называется запросом к базе данных.

Пример 1 ([15]). Реляционная БД состоит из следующих отношений.

Таблица 1.
Отношение $r_1(a,\ i)$

автор a	место жительства i
Edmonds	Waterloo
Cook	Toronto

Таблица 2.
Отношение $r_2(a,\ p)$

автор a	публикация p
Edmonds	P
Cook	NP
Shakespeare	P или NP

Таблица 3.
Отношение $r_3(p,\ s)$

публикация p	тема s
P	алгоритмы
NP	сложность
P или NP	драма

Таблица 4.
Отношение $r_4(p,\ j,\ y)$

публикация p	журнал j	год y
P	Can. J. Math.	1965
NP	Am. J. Math.	1971

Пример 2 (Реляционная алгебра и реляционное исчисление). Известны два эквивалентных способа манипулирования реляционными данными – *реляционная алгебра* и *реляционное исчисление*.

Фактическим стандартом доступа к реляционным данным стал язык запросов SQL (Structured Query Language), который представляет собой смесь операторов реляционной алгебры и выражений реляционного исчисления, использующий ключевые слова на английском языке и расширенный дополнительными возможностями, отсутствующими в реляционной алгебре и реляционном исчислении.

Фундаментальный результат, известный как *теорема Кодда* [16], устанавливает эквивалентность реляционной алгебры и реляционного исчисления в том смысле, что для любого запроса, заданного в терминах одного языка, можно эффективно построить (за полиномиальное время) запрос на другом языке, который эквивалентен, т.е. производит тот же ответ для любого наполнения БД.

Этот результат по сути дела означает, что мы можем декларативно задать, что мы хотим найти, а СУБД автоматически находит, как вычислить запрос с помощью реляционных операторов.

1) *Реляционное исчисление* похоже на исчисление предикатов первого порядка. Формулы здесь строятся из элементарных формул вида $x_1 = x_2$ и $p(R_i(x_1, \ldots, x_m))$ (где x_i – переменные или константы из соответствующих доменов), использующие операторы $\wedge$, $\vee$, $\neg$ булевой алгебры и квантификаторы $\forall$, $\exists$. Формула φ со свободными параметрами $y_1, \ldots, y_k$ задает *запрос* $\{y_1, \ldots, y_k \mid \varphi(y_1, \ldots, y_k)\}$; для данной

БД d результатом выполнения запроса является k-арное отношение, содержащее все кортежи $a_1, \ldots, a_k$ такие, что $\varphi(a_1, \ldots, a_k)$ истинна в d.

Например, запрос в описанном выше примере

$$\{j \mid (\exists a)(\exists p)\ p(R_1(a,\ Waterloo)) \wedge p(R_2(a,\ p)) \wedge p(R_4(p,\ j,\ 1965))\}$$

запрашивает журналы, в которых опубликовались авторы из Waterloo в 1965 г. (Здесь $p(R_i)$ – предикаты, соответствующие отношениям r_i $(i = 1, 2, 4)$).

Реляционное исчисление является декларативным, то есть оно задает, *что* мы хотим вычислить, но не задает, *как* это сделать.

2) *Реляционная алгебра* представляет собой набор операторов, использующих отношения в качестве аргументов, и возвращающих отношения в качестве результата. Таким образом, реляционный оператор выглядит как функция с отношениями в качестве аргументов и определяется с помощью элементарных операций над отношениями.

Реляционная алгебра является *замкнутой*, т.к. в качестве аргументов в реляционные операторы можно подставлять другие реляционные операторы, подходящие по типу. Таким образом, в реляционных выражениях можно использовать вложенные выражения сколь угодно сложной структуры.

Согласно Кодду [13], определяют восемь реляционных операторов, объединенных в следующие две группы.

Теоретико-множественные операторы: объединение, пересечение, вычитание, декартово произведение.

Специальные реляционные операторы: выборка, проекция, соединение, деление.

Не все эти операторы являются независимыми, т.к. операторы соединения, пересечения и деления могут быть выражены через другие реляционные операторы. Остальные реляционные операторы (объединение, вычитание, декартово произведение, выборка, проекция) являются примитивными операторами – их нельзя выразить друг через друга.

Рассмотрим некоторые *операторы реляционной алгебры.*

(1) *выборка* или селекция, обозначаемая $\sigma_F(r)$, находит кортежи отношения r, удовлетворяющие заданному условию F. Смысл операции выборки – выбрать кортежи отношения, удовлетворяющие некоторому условию. Таким образом, операция выборки дает «горизонтальный срез» отношения по некоторому условию.

(2) *проекция* $\pi_X(r)$ ограничивает отношение r подмножеством X атрибутов. Операция проекции дает «вертикальный срез» отношения, в котором удалены все возникшие при таком срезе дубликаты кортежей.

(3) *естественное соединение* $r_1 \bowtie r_2$ образует отношение над объединением атрибутов двух отношений r_1 и r_2 путем сочетания каждого кортежа t_1 отношения r_1 с каждым кортежем t_2 отношения r_2, которое согласуется по общим атрибутам отношений. Заметим, что если $R_1 = R_2$, то $r_1 \bowtie r_2 = r_1 \cap r_2$. Если все R_i различны, то $\bowtie r_i = \times r_i$.

(4) *соединение по условию* F: $r \underset{F}{\bowtie} s = \{(x, y) \in r \times s\ :\ F(x, y) =\ истина\}$.

(5) *переименование* $\varrho_{A|B}(r)$, изменяющее имя атрибута A на B.
(6) *объединение* $r_1 \cup r_2$ и *разность* $r_1 - r_2$ двух отношений над одним множеством атрибутов (в обычном теоретико-множественном смысле).

Отметим следующие свойства операторов реляционной алгебры: коммутативность и ассоциативность соединений, коммутативность выборки и проекции, коммутативность выборки и естественного соединения [14].

Пример 3. Запрос о журналах, в которых опубликовались авторы из Waterloo в 1965 г., записан ниже с помощью операторов реляционной алгебры: $\pi_j \sigma_{i=Waterloo} \sigma_{y=1965}(r_1 \bowtie r_2 \bowtie r_4)$. Обработка этого запроса с помощью операторов реляционной алгебры возможна различными способами, характеризующимися различной эффективностью. «Алгебраическая» оптимизация запросов [14] использует, в частности, правило как можно более раннего вычисления выборки (до соединения), поскольку вычисление соединения требует большой вычислительной работы.

Так, более эффективно выполнить выборку $y = 1965$ для отношения r_4 перед соединением его с r_2 (это сопряжено с меньшим объемом вычислений!). Выборка $\sigma_{i=Waterloo} r_1(a,\ i)$ достигается путем вычеркивания из таблицы отношения r_1 строк, в которых $i \neq Waterloo$.

Таблица 5.
$r_4(p,\ j,\ 1965)$**: Выборка отношения** $r_4^{'}(p,\ j,\ y) = \sigma_{y=1965} r_4(p,\ j,\ y)$

публикация p	журнал j	год y
P	Can. J. Math.	1965

Таблица 6.
$r_1(a,$ **'Waterloo'): Выборка отношения** $r_1^{'}(a,\ i) = \sigma_{i='Waterloo'} r_1(a,\ i)$

автор a	место жительства i
Edmonds	Waterloo

2. Запросы в базах данных

Оптимизация запросов является одним из наиболее важных и интересных направлений исследований в теории баз данных. Важность этого направления определяется тем, что от качества оптимизации запросов существенно зависит общая производительность СУБД. Обработка большинства запросов в реляционных базах данных может быть выполнена с помощью двух операторов: проекции и соединения.

2.1. ***Задачи удовлетворения ограничений и базы данных.*** Задачи удовлетворения ограничений (УО), известные в англоязычной литературе как constraint satisfaction problems (CSP) [12], широко используются при решении ряда практически важных задач ИИ, таких как составление расписаний, проектирование электронных схем, поддержка принятия решений.

Определение 1 ([12])**.** Задача УО (ЗУО) определяется множеством дискретных переменных $x_1, \ldots, x_n$, для каждой из которых задана область определения или домен $D_j = \{d_j^{(1)}, \ldots, d_j^{(p_j)}\}$ $(j = 1, \ldots, n)$, и множеством ограничений. *Ограничением* называется пара $(r, scope(r))$, где r – отношение, определенное на диапазоне $scope(r)$.

Решением ЗУО называется присвоение значений всем переменным, которое удовлетворяет всем ограничениям. *Целью* решения ЗУО может быть нахождение одного или всех решений.

Укажем на весьма тесную связь между задачами удовлетворения ограничений и базами данных, что показано в следующей таблице [17]:

Таблица 1.
Терминология удовлетворения ограничений и БД

Терминология удовлетворения ограничений		Терминология БД
задача удовлетворения ограничений	$\equiv$	база данных
переменная	$\equiv$	атрибут
домен	$\equiv$	домен атрибута
ограничение	$\equiv$	отношение
диапазон отношения	$\equiv$	реляционная схема
множество решений	$\equiv$	соединение всех отношений

Итак, поиск ответа на запрос в базе данных может быть сведен к решению задачи УО, которая может быть решена с помощью локального элиминационного алгоритма (ЛЭА) [12].

2.2. ***Графовые и гиперграфовые модели запросов.*** Графы и гиперграфы [18] используются для представления структуры запросов. Структура запроса в реляционной БД представляется в виде гиперграфа, вершины которого соответствуют атрибутам, а ребра – соответствуют реляционным схемам отношений R_i в одном запросе (см. рис. 1 a)). Класс гиперграфов, называемых *ациклическими*, соответствует схемам со многими желательными свойствами [5]. Было доказано еще в 1981 г. (*Yannakakis* [19]), что для решения различных NP-полных задач на схемах БД с «простыми» (ациклическими) гиперграфами имеются полиномиальные алгоритмы.

Часто вместо гиперграфового представления структуры запроса в реляционной БД используется более наглядное представление с помощью *первичного* и *двойственного* графов. Первичный граф называется обычно *графом взаимосвязей* [20] или *графом Гайфмана* (*Gaifman* [21]) запроса БД. В графе Гайфмана вершины соответствуют атрибутам запроса и неориентированное ребро между двумя вершинами соответствует наличию отношения в запросе, содержащего эти атрибуты (см. рис. 1 b)). Будем называть такие атрибуты *взаимосвязанными*. В дальнейшем нам потребуется понятие *окрестности атрибута*: множество атрибутов, взаимосвязанных с атрибутом x в графе взаимосвязей G, называется *окрестностью* $Nb(x)$ атрибута x в графе G. *Двойственным графом* гиперграфа $H = (V, S)$ с множеством вершин V и множеством ребер S называется граф, вершины которого соответствуют ребрам гиперграфа, причем пара таких вершин соединяется ребром в двойственном графе, если они имеют общие вершины из V (см. рис. 1 c)).

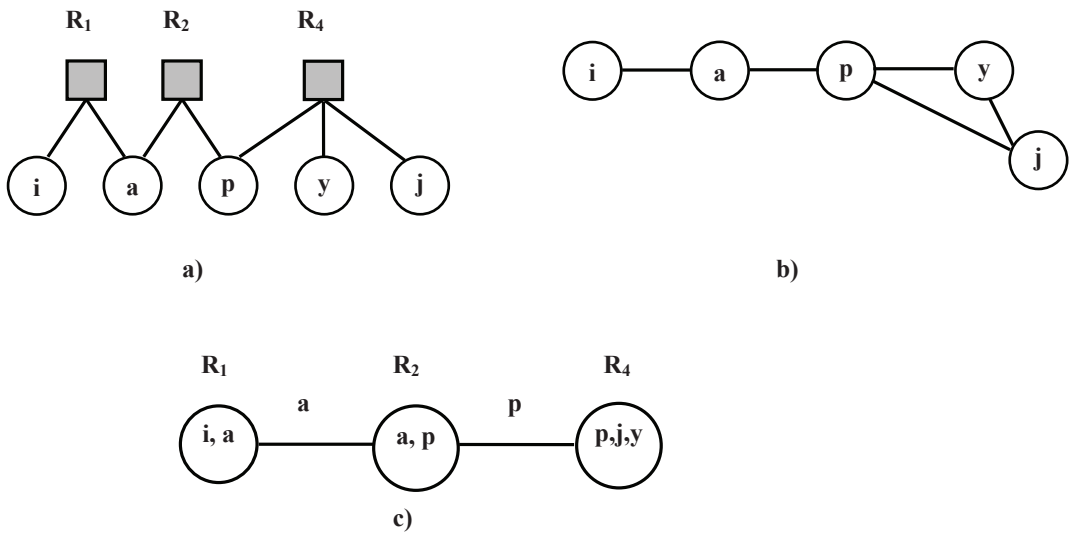

Рис. 1. Представление структуры запроса из примера 1: a) в виде гиперграфа; b) в виде графа Гайфмана (взаимосвязей атрибутов); c) в виде двойственного графа.

2.3. ***Обработка запросов в базах данных с помощью локального элиминационного алгоритма.*** Рассмотрим реляционную БД d, состоящую из множества отношений $\{r_1, \ldots, r_n\}$ с реляционными схемами $R_1, \ldots, R_n$. Запрос состоит в нахождении выборки σ_F от соединения отношений, сопровождающейся проекцией на некоторую область S

$$\pi_S(\sigma_F(r_1 \bowtie \ldots \bowtie r_n)). \tag{2.1}$$

Отметим, что запросы такого вида эквивалентны так называемым *конъюнктивным запросам* [22], которые являются наиболее часто встречающимся типом запросов.

Рассмотрим метод поиска ответа на запрос вида (2.1), основанный на последовательной элиминации атрибутов с помощью ЛЭА, который вычисляет информацию о локальных элементах (атрибутах) задачи нахождения ответа на запрос в реляционной БД, задаваемой структурным графом, записывая локальную информацию об этих элементах в виде новых отношений, добавляемых к задаче, затем элиминируя просмотренные элементы и использованные отношения. При этом рассматриваются все отношения, содержащие некоторый атрибут. После нахождения соединения ($\bowtie$) этих отношений [12], атрибут элиминируется с помощью проекции полученного отношения на множество оставшихся атрибутов. Эти операции проделываются далее до тех пор, пока не останется лишь одно отношение, которое и является решением задачи – ответом на данный запрос. Если последнее отношение непусто, то ответ на запрос имеется, если же пусто – то ответа нет.

Для решения задачи выполнения запроса, описываемого системой отношений $r_1, \ldots, r_n$ с атрибутами $x_1, \ldots, x_m$, при заданном упорядочении A_π атрибутов, ЛЭА выглядит следующим образом:

1. Выбрать очередной атрибут x из схемы БД согласно упорядочению A_π. Найти соединение отношений, соответствующих окрестности $Nb(x)$ элемента x в текущем структурном графе, сформировав новое отношение r с диапазоном $(x,\ Nb(x))$.
2. Спроектировать полученное отношение на множество атрибутов, соответствующих окрестности $Nb(x)$ атрибута x. В результате получится новое отношение $r^{'} = \pi_{Nb(x)} r(x)$ с диапазоном $Nb(x)$, добавляемое к отношениям задачи. Если отношение с тем же набором переменных уже имеется, найти их пересечение. Если пересечение пусто, то задача выполнения запроса решения не имеет.
3. Элиминировать атрибут x вместе с соответствующими отношениями. Из элементов его окрестности $Nb(x)$ создается клика в структурном графе (эта клика соответствует отношению $r^{'}$). Создание клик изменяет структурный граф и окрестности элементов.
4. Продолжать до тех пор, пока не останется нерассмотренных отношений.

Рассмотрим подробнее детали реализации ЛЭА при решении задачи выполнения запроса в БД в случае, когда структурный граф является *графом взаимосвязей* атрибутов [20], используя описанный выше пример.

После использования правила возможно более раннего вычисления выборок, описанный в примере запрос приобретает вид: $\pi_j((\sigma_{i=Waterloo} r_1(a,i)) \bowtie (r_2(a,p) \bowtie (\sigma_{y=1965} r_4(p,j,y))))$. Выборки $\sigma_{y=1965} r_4(p,j,y)$ и $\sigma_{i=Waterloo} r_1(a,i)$ являются отношениями, заданными соответственно таблицами 5 и 6.

Рассмотрим упорядочение $A_\pi = \{p, a\}$ для элиминации атрибутов. Для элиминации атрибута p выделим взаимосвязанные с ним атрибуты: $Nb(p) = \{a\}$ (рис. 1 b)) и отношения, содержащие p: $r_2(a,p)$ и $r_4^{'}(p,j,y) = \sigma_{y=1965} r_4(p,j,y)$. Вычислим соответствующее соединение $r_{2,4}(a,p,j,y) = r_2(a,p) \bowtie r_4^{'}(p,j,y)$ (см. табл. 7) и проекцию его на $\{a,j,y\}$: $r_{24}^{'}(a,\ j,\ y) = \pi_{\{a,j,y\}} r_{2,4}(a,p,j,y)$ (табл. 8). Для элиминации атрибута a вычислим соответствующее соединение $r_{124}(a,\ i,\ j,\ y) = r_1^{'}(a,i) \bowtie r_{24}^{'}(a,\ j,\ y)$ (табл. 9), а затем найдем проекцию $r_{124}^{'}(i,\ j,\ y) = \pi_{\{i,\ j,\ y\}} r_{124}(a,\ i,\ j,\ y)$ (табл. 10). Ответ на запрос $\pi_j r_{124}^{'}(i,\ j,\ y)$ показан в табл. 11.

Таблица 7.
Отношение $r_{24}(a,\ p,\ j,\ y)$

автор a	публикация p	журнал j	год y
Edmonds	P	Can. J. Math.	1965

Таблица 8.
Отношение $r_{24}^{'}(a,\ j,\ y)$

автор a	журнал j	год y
Edmonds	Can. J. Math.	1965

Таблица 9.
Отношение $r_{124}(a,\ i,\ j,\ y)$

автор a	место жительства i	журнал j	год y
Edmonds	Waterloo	Can. J. Math.	1965

Таблица 10.
Отношение $r'_{124}(i,\ j,\ y)$

место жительства i	журнал j	год y
Waterloo	Can. J. Math.	1965

Таблица 11.
Отношение $\pi_j r'_{124}(i,\ j,\ y)$

журнал j
Can. J. Math.

Как видно из решения примера, обратная часть ЛЭА здесь не нужна, так как в таблице 11 найден ответ на запрос.

Заключение

Локальный элиминационный алгоритм вычисления информации – перспективный подход, делающий возможным решение дискретных разреженных задач информатики. Использование графовых структур позволяет рационально построить вычислительную схему локального алгоритма. *Перспективными направлениями* дальнейших исследований являются разработка эффективных схем локального элиминационного алгоритма и использование соответствующих графовых структур при решении конкретных задач выполнения запросов в реляционных базах данных, обладающих специальной структурой.

Список литературы

1. *Журавлев Ю.И.* Избранные научные труды. — М.: Магистр, 1998. — 420 с.
2. *Сергиенко И.В., Шило В.П.* Задачи дискретной оптимизации: проблемы, методы решения, исследования. — К.: Наукова думка, 2003. — 162 с.
3. *Леонтьев В.К.* Дискретная оптимизация // Журнал вычислительной математики и математической физики. — 2007. — Т. 47. — С. 338–352.
4. *Dechter R.* Bucket elimination: A unifying framework for reasoning // Artificial Intelligence. — 1999. — V. 113. — P. 41–85.
5. *Beeri C., Fagin R., Maier D., Yannakakis M.* On the desirability of acyclic database schemes // J. ACM. — 1983. — V. 30. — P. 479–513.
6. *George A., Liu J.W.H.* Computer Solution of Large Sparse Positive Definite Systems. — Englewood Cliffs: Prentice-Hall Inc., 1981. — 324 p.
7. *Rose D.J.* A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations // Graph Theory and Computing /R.C. Read (ed.). — New York: Academic Press, 1972. — P. 183–217.
8. *Щербина О.А.* Локальные элиминационные алгоритмы решения разреженных дискретных задач // Журнал вычислительной математики и математической физики. — 2008. — Т. 48, N 1. — С. 161—177.
9. *Щербина О.А.* Роль графовых структур в теории локальных элиминационных алгоритмов) // Динамические системы. — 2008. — Вып 24. — С. 83–98.
10. *Liu J.W.H.* The role of elimination trees in sparse factorization // SIAM Journal on Matrix Analysis and Applications. — 1990. — V. 11. — P. 134–172.
11. *Parter S.* The use of linear graphs in Gauss elimination // SIAM Review. — 1961. — V. 3. — P. 119–130.
12. *Щербина О.А.* Локальные элиминационные алгоритмы для задач удовлетворения ограничений // Таврический вестник информатики и математики. — 2007. — №1. — С. 24–39.

13. *Codd E.F.* A relational model of data for large shared data banks // Commun. ACM. — 1970. — V. 13. — P. 377–387.
14. *Ullman J.* Principles of Database and Knowledge-Base Systems, 2 Volumes. — New York: Computer Science Press, 1988/89.
15. *Yannakakis M.* Graph-theoretic methods in database theory // Proc. 9th ACM PODS. — 1990. — P. 230–242.
16. *Codd E.F.* Relational Completeness of Data Base Sublanguages. IBM Research Report RJ987 (March 6th, 1972). — Republished in Randal J. Rustin (ed.), Data Base Systems: Courant Computer Science Symposia Series 6, Englewood Cliffs, N.Y.: Prentice-Hall, 1972.
17. *Pearson J., Jeavons P.* A survey of tractable constraint satisfaction problems. Technical Report CSD-TR-97-15, Royal Holloway University of London, 1997.
18. *Емеличев В.А., Мельников О.И., Сарванов В.И., Тышкевич Р.И.* Лекции по теории графов. — М.: Наука, 1990. — 384 с.
19. *Yannakakis M.* Computing the minimum fill-in is NP-complete // SIAM J. Alg. Disc. Meth. — 1981. — V. 2. — P. 77-–79.
20. *Bertele U., Brioschi F.* Nonserial Dynamic Programming. — New York: Academic Press, 1972. — 235 p.
21. *Gaifman H.* On local and non-local properties // Logic Colloquium '81. — North Holland, 1982.
22. *Arnborg S., Proskurowski A.* Characterization and recognition of partial k-trees // SIAM J. Alg. Discr. Meth. — 1986. — V. 7. — P. 305-314.

Статья поступила в редакцию 25.08.2009