УДК 004.652+004.655+004.657

Д. Б. БУЙ[1], В. Г. СКОБЕЛЕВ[2]

[1] ***Киевский национальный университет имени Тараса Шевченко, г. Киев, Украина***
[2] ***Институт прикладной математики и механики НАН Украины, г. Донецк, Украина***

МОДЕЛИ, МЕТОДЫ И АЛГОРИТМЫ ОПТИМИЗАЦИИ ЗАПРОСОВ В БАЗАХ ДАННЫХ (ОБЗОР)

В данной работе сделана попытка исследовать современное состояние проблемы обеспечения эффективности функционирования систем управления базами данных (СУБД) с позиции моделей и методов, предназначенных для оптимизации запросов. На основе анализа особенностей процесса проектирования СУБД, основных моделей данных, основных понятий, используемых при оценке сложности алгоритмов, показано, что в этом контексте следует рассматривать именно связку «БД–СУБД». Охарактеризованы основные два типа математических моделей, предназначенных для теоретического анализа эффективности функционирования связки «БД–СУБД» в терминах анализа сложности операций. Во-первых, это модели, конструируемые в терминах формальной теории, построенной на основе синтеза математической логики, теории моделей и прикладной теории алгоритмов. Во-вторых, это модели многомерных связок «БД–СУБД», конструируемые в терминах теории категорий (что, в частности, дает возможность эффективно использовать язык схем). Рассмотрена эволюция методов «оптимизации запросов» (являющихся, по своей сути, эвристическими методами) в контексте развития моделей связок «БД–СУБД».

Ключевые слова: *базы данных, СУБД, оптимизация запросов.*

Введение

Известно, что основной формой организации информации в современных информационных системах (ИС) являются связки «база данных (БД) – система управления базой данных (СУБД)».

Парадигма БД сформировалась в середине 60-х годов XX столетия в результате многочисленных попыток избавиться от недостатков файловых систем организации информации, возникающих при многопользовательском доступе к данным. Основными из этих недостатков являются [1]:

1) изолированность данных (параллельно работающие приложения не могут одновременно изменять записи в одном и том же файле, а совместная обработка нескольких файлов достаточно сложная);

2) зависимость программ от данных (описание структуры данных в прикладной программе требует при изменении структуры файла внесение соответствующих изменений в программу и последующую ее перекомпиляцию);

3) дублирование данных (различные приложения, использующие данные об одном и том же объекте, хранимые в своих независимых файлах, непроизводительно расходуют память, и могут привести к противоречивости данных);

4) отсутствие описания данных (в файлах данные, обрабатываемые прикладными программами, хранятся без их описания, что приводит к сложности документирования ИС и появлению ошибок).

Естественным средством устранения основных этих недостатков является отделение процесса хранения данных от процесса их обработки, что и реализуется связкой «БД–СУБД». В этой связке БД – это информационная модель предметной области, представленная в виде совокупности данных, хранимых в памяти компьютера и связанных между собой правилами, определяющими общие принципы их описания, хранения и манипулирования, а СУБД – это совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования БД многими пользователями.

Отметим, что ядром БД является модель данных, т.е. совокупность структур данных (именно они обеспечивают возможность использования того или иного алгоритма) и операций их обработки. Кроме того, СУБД по своему предназначению делятся на операционные (т.е. обладающие высокой скоростью реакции на запрос, извлечения и представления информации) и предназначенные для работы с хранилищами данных, содержащими очень большой объем информации, подготовка представления которой занимает значительное время.

Проблема проектирования связок «БД–СУБД» является одной из центральных в процессе разработки ИС, так как от ее решения, во многом, зависит качество создаваемой ИС. Этой проблеме посвящены усилия многочисленных исследователей во всем мире. Большое число подходов к ее решению (см.,

напр., [1-12]) обусловлено следующим. В результате стремительного развития средств вычислительной техники ИС стали применяться практически во всех сферах жизнедеятельности современного общества. Как следствие, происходит рост как многообразия моделей данных и способов их представления на физических носителях (т.е. типов БД), так и многообразия средств взаимодействия с БД (т.е. СУБД).

Основными требованиями, предъявляемыми к связкам «БД–СУБД» являются их безопасность (как ИС) и эффективность функционирования на протяжении всего жизненного цикла.

Проблеме обеспечения безопасности связок «БД–СУБД» посвящены многочисленные исследования, и в настоящее время в ее решении достигнуты значительные успехи [13].

Совершенно иная ситуация имеет место с проблемой обеспечения эффективности функционирования связок «БД–СУБД». На практике, как правило, разработчики руководствуются принципом «мы делаем так, как умеем», не проводя детального теоретического анализа этой проблемы в конкретной ситуации. Такой подход обусловлен, прежде всего, большой внутренней сложностью этой проблемы, природа которой состоит в следующем. В терминах классической связки В.М. Глушкова «управляющий автомат – операционный автомат» [14] связка «БД–СУБД» представляет собой операционный автомат с переменной структурой. Как известно, анализ и синтез таких автоматов является одной из наименее изученных проблем теории автоматов. Кроме того, широкому спектру задач, решаемых с использованием информационных технологий, соответствует широкий спектр типов автоматов, являющихся моделями связок «БД–СУБД».

Целью настоящего обзора и является попытка охарактеризовать современное состояние проблемы анализа эффективности функционирования связок «БД–СУБД». Структура работы следующая. В п. 1 дана общая характеристика особенностей проектирования связок «БД–СУБД». В п. 2 кратко представлены основные понятия, используемые при оценке сложности алгоритмов. В п. 3 рассмотрен современный подход к теоретическому анализу связок «БД–СУБД». В п. 4 охарактеризованы существующие подходы к решению проблемы эффективной организации обработки запросов. Заключение содержит ряд выводов.

1. Особенности процесса проектирования связки «БД–СУБД»

Процесс проектирования связки «БД–СУБД» состоит из следующих 4-х этапов:

1. Системный анализ предметной области.

2. Построение инфологической (*концептуальной*) *модели предметной области*. Эта модель не зависит от СУБД (и, следовательно, от среды хранения данных), и представляется в терминах выбранной семантической модели (одной из наиболее распространенных семантических моделей является разработанная П. Ченом ER-модель (Entity Relationship) [15]).

3. Построение логической модели. Выбирается модель данных и тип СУБД. Строится схема БД и подсхемы для различных пользователей. Создается набор возможных типовых запросов. Определяются спецификации для программного обеспечения (ПО).

4. Построение физической модели. Выбирается размещение БД на внешних носителях и определяется используемая СУБД. Создается реальная БД. Программируются и отлаживаются приложения, которые будут работать с БД.

Результатом процесса проектирования связки «БД–СУБД» является готовая к заполнению реальная БД и готовое к использованию ПО.

Таким образом, в связке «БД–СУБД» выделяются инфологический, логический и физический уровни. Первые два уровня называются внешними и предназначены для поддержки санкционированного доступа к данным БД. Третий уровень называется внутренним и отображает организацию данных в среде хранения.

С позиции логической модели многообразие типов связок «БД–СУБД» в значительной мере определяется используемыми моделями данных. В настоящее время основными являются следующие модели данных.

1. Иерархическая модель. Разработана в [16]. Модель представляет собой набор ориентированных корневых деревьев. Каждая вершина дерева соответствует записи (или упорядоченному набору записей) со встроенными указателями на непосредственного предка (кроме корня дерева) и потомков вершины (кроме висячих вершин). Поддерживает связи «один к одному» и «один ко многим». Модель эффективна при выполнении запросов, естественно формулируемых в терминах операций модификации двухсвязных списков.

2. Сетевая модель. Разработана в [17, 18]. Представляет собой набор ориентированных связных графов с рядом ограничений на их структуру. Расширение возможностей по сравнению с предыдущей моделью состоит в том, что за счет двух встроенных связей «один ко многим» поддерживается связь «многие ко многим». Модель эффективна при выполнении запросов, естественно формулируемых в терминах любых операций над двухсвязными списками.

3. Реляционная модель. Разработана в [21]. Мо-

дель основана на понятии «отношение» [19, 20] – одном из основных понятий математики. Представляет собой набор двумерных таблиц, строки которых имеют одинаковую структуру. Модель эффективна при выполнении запросов, естественно формулируемых в терминах операций над отношениями. В настоящее время эта модель де-факто является «промышленным стандартом».

4. Объектно-ориентированная модель. Возникла в результате формирования в 80-х годах XX столетия объектно-ориентированного подхода к программированию [22-25]. Ее суть состоит в следующем. Основными понятиями являются «объект» (имеющий уникальный идентификатор) и «литерал». Объекты (они разбиваются на атомарные и структурированные) инкапсулируют свое состояние (определяемое значениями набора свойств объекта) и поведение (т.е. операции, которые могут быть выполнены объектом или над ним). Свойства объекта делятся на атрибуты (не являющиеся объектами и принимающими в качестве значения литерал или идентификатор) и связи (представленные посредством ссылочных атрибутов). Объекты, имеющие одинаковые атрибуты и отвечающие на одни и те же сообщения, образуют класс. Отношение наследования дает возможность определять один класс как частный случай другого класса. Таким образом, логическая структура этой модели похожа на структуру иерархической модели (отличие состоит в методах манипулирования данными). В настоящее время объектно-ориентированная модель недостаточно проработана теоретически. Как следствие, отсутствует ее стандарт.

5. Объектно-реляционная модель. Представляет собой расширение реляционной модели за счет поддержки основных концепций (кроме наследования классов) объектно-ориентированного программирования [26]. В этой модели сняты ограничения неделимости записей в таблицах, допускаются поля, значениями которых являются таблицы, встроенные в таблицы. Это дает возможность создавать типы данных любой степени сложности. Достоинством модели является возможность использовать существующие реляционные БД для вновь разрабатываемых приложений. В настоящее время на практике при объектно-ориентированном подходе к программированию используется именно объектно-реляционная модель.

6. Многомерная модель. Является обобщением реляционной модели. Представляет собой набор многомерных таблиц [27, 28]. Модель предназначена для аналитической обработки больших объемов информации, связанной со временем (в частности, для решения задач прогнозирования). Основными понятиями являются «измерение» и «ячейка». Измерение состоит в выделении множества однотипных данных, соответствующих грани многомерной таблицы. Ячейка – это поле, значение которого определяется фиксированным набором измерений. Если все таблицы имеют одинаковую размерность и совпадающие типы данных при всех измерениях, то модель называется гиперкубической, иначе – поликубической. В модели реализованы специальные операции «срез», «вращение», «агрегация» и «детализация», дающие возможность осуществлять анализ информации на различных уровнях ее обобщения.

С позиции физической модели многообразие типов связок «БД–СУБД» можно разбить на следующие классы.

1. По среде размещения. Связка «БД–СУБД» является:

1) *локальной* (*сосредоточенной*), если она реализована на одном компьютере;

2) *распределенной*, если она реализована в компьютерной сети.

2. По способу доступа. Связка «БД–СУБД» является:

1) *мейнфреймовой*, если рабочее место – это текстовый или графический терминал, а вся информация обрабатывается на сервере, т.е. компьютере или компьютерах где хранится связка;

2) *файл-серверной*, если она размещена на сервере, доступ к данным и прикладным программам осуществляется через локальную сеть, а ядро СУБД реализовано на каждом клиентском компьютере;

3) *клиент-серверной*, если она размещена на сервере, доступ к данным осуществляется через локальную сеть, а ядро СУБД и прикладные программы реализованы на клиентских компьютерах;

4) *встраиваемой*, если она является программной библиотекой, т.е. на локальных компьютерах организовано унифицированное хранение больших объемов данных, а доступ к ним происходит посредством запросов или вызовом функций библиотеки из приложения пользователя.

3. По подходу к организации вычислений. Связка «БД–СУБД» является:

1) *классической* (фон-неймановской), если при обработке запросов используется только классический (т.е. последовательный) подход к организации вычислений;

2) *нео-классической*, если при обработке запросов используются параллельные и/или распределенные вычисления.

В реальной связке «БД–СУБД» время обработки любого запроса состоит из времени, затраченного на операцию загрузки данных с медленного носителя в оперативную память, времени, затраченного на вычисление, и времени, затраченного на возвраще-

ние результата в соответствующий медленный носитель. Поэтому естественно выделяются следующие два подхода к анализу эффективности обработки запросов в связке «БД–СУБД».

Первый подход основан на использовании симбиоза математической логики, теории моделей и прикладной теории алгоритмов. Предметом его исследования является математическая модель связки «БД–СУБД». Цель состоит в построении нижних оценок асимптотической временной сложности функционирования связки «в наихудшем случае» или «в среднем».

Второй подход основан на исследовании реальной связки «БД–СУБД». Как правило, он осуществляется методами статистического анализа. Цель состоит в оценке реального времени обработки запросов (возможно, модельными) реализациями связки при тех или иных условиях.

Подчеркнем, что для получения адекватной картины оба эти подхода к анализу эффективности обработки запросов должны осуществляться как на этапе проектирования, так и в процессе эксплуатации связки «БД–СУБД», особенно при ее структурных модификациях и создании новых версий.

2. Анализ сложности алгоритмов

В процессе работы любой алгоритм преобразования информации обрабатывает некоторые входные данные того или иного «размера», характеризующего «объем памяти», необходимой для их хранения. К определению «размера» применяется один из следующих двух подходов [29-34]. При первом подходе используется *равномерный вес*, т.е. предполагается, что для записи числа n используется единица памяти, а каждая элементарная операция выполняется в течение единицы времени. При втором подходе используется *логарифмический вес*, т.е. предполагается, что для записи числа n используется $\lceil \log n \rceil$ единиц памяти, и для каждой элементарной операции вычисляется время ее выполнения как функция от размера используемых в ней данных.

Анализ временной сложности алгоритма при его работе над входом фиксированного размера при равномерном весе, состоит в оценке числа выполненных элементарных операций, а при логарифмическом весе – в оценке времени, затраченного на все сработавшие элементарные операции. Использование равномерного веса проще, чем использование логарифмического веса. При анализе временной сложности обработки запросов связкой «БД–СУБД» существенную роль играют операции ввода-вывода. Поэтому при таком анализе более адекватным является использование логарифмического веса. Отметим, что использование равномерного веса дает возможность получить некоторые грубые оценки временной сложности обработки запросов связкой «БД–СУБД». Однако подчеркнем еще раз, в каждом случае вопрос о том, насколько приемлемы такие оценки, требует отдельного изучения.

Выделяют три типа временной сложности алгоритма при обработке данных фиксированного размера: *сложность в наихудшем случае*, *сложность для почти всех входов* и *сложность в среднем*. Исследование временной сложности алгоритма в наихудшем случае существенно проще, чем исследование его сложности для почти всех входов или сложности в среднем. В последнем случае предполагается, что для множества данных фиксированного размера задано адекватное распределение вероятностей появления его элементов при работе алгоритма (что является достаточно сложной задачей). При анализе временной сложности обработки запросов связкой «БД–СУБД» наиболее приемлемым является использование сложности в среднем. Именно она дает возможность оценить эффективность обработки запросов связкой при условиях, определенных в техническом задании. Отметим, что использование сложности в наихудшем случае или сложности для почти всех входов также дает возможность получить некоторые грубые оценки временной сложности обработки запросов связкой «БД–СУБД». Однако в каждом случае вопрос о том, насколько приемлемы такие оценки требует отдельного изучения.

Вычисление временной сложности алгоритма (за исключением тривиальных случаев) приводит к громоздким, мало обозримым формулам. Для того чтобы избежать такой ситуации, оценивают порядок роста этой сложности при неограниченном росте размера входных данных. Такая оценка называется *асимптотической временной сложностью алгоритма*. При ее вычислении для оценки функции $g(n)$ ($n \in \mathbf{N}$, $\mathbf{N}$ – множество натуральных чисел) используют следующие асимптотические обозначения (где $f(n)$ – достаточно простая функция):

1) $g(n) = o(f(n))$ $(n \to \infty)$ означает, что $\lim\limits_{n\to\infty} \dfrac{g(n)}{f(n)} = 0$;

2) $g(n) = \omega(f(n))$ $(n \to \infty)$ означает, что $f(n) = o(g(n))$;

3) $g(n) = O(f(n))$ $(n \to \infty)$ означает, что существуют такое число $n_0 \in \mathbf{N}$ и константа $c > 0$, что для всех $n > n_0$ истинно неравенство $g(n) < cf(n)$;

4) $g(n) = \Omega(f(n))$ $(n \to \infty)$ означает, что существуют такое число $n_0 \in \mathbf{N}$ и константа $c > 0$, что

для всех $n > n_0$ истинно неравенство $g(n) > cf(n)$;

5) $g(n) = \Theta(f(n))$ означает, что существуют такое число $n_0 \in \mathbf{N}$ и константы $c_1, c_2 > 0$, что для всех $n > n_0$ истинны неравенства $c_1 f(n) < g(n) < c_2 f(n)$.

При анализе асимптотической временной сложности алгоритма чаще всего предпринимается попытка получить оценку вида $O(f(n))$ (ее легче получить, чем оценку вида $\Theta(f(n))$, и она более информативная, чем оценки вида $o(f(n))$, $\omega(f(n))$ или $\Omega(f(n))$).

Таким образом, для теоретического анализа эффективности обработки запросов в процессе эксплуатации связки «БД–СУБД» наиболее приемлемым является вычисление асимптотической временной сложности в среднем при использовании логарифмического веса.

Рассмотрим кратко структуры данных, используемые при анализе эффективности обработки запросов связкой «БД–СУБД» [22-36].

Массив представляет собой линейную структуру, состоящую из n «ячеек» (число n – размер массива), в каждую из которых может быть записан любой элемент, принадлежащий фиксированному множеству S однотипных элементов (ниже предполагается, что на множестве S задано равномерное распределение вероятностей). Выделяют следующие типы массивов:

1. Статический массив. Его размер n задаётся на стадии объявления и не может быть изменен во время работы алгоритма. Выборка или запись элемента по заданному индексу в такой массив осуществляется за время

$$T = \begin{cases} O(\log n), \text{ если } n \to \infty, \ |S| \text{ фиксировано;} \\ O(\log |S|), \text{ если } |S| \to \infty, \ n \text{ фиксировано;} \\ O(\log n \cdot \log |S|), \text{ если } n \to \infty \text{ и } |S| \to \infty. \end{cases} \quad (1)$$

2. Вектор. Является динамическим массивом с возможностью изменения размера n. Он обеспечивает те же операции, что и статический массив, а также дает возможность удалять и добавлять элементы в концы вектора. Выделяют следующие типы векторов:

1) *стек*, если элемент может быть добавлен только в один конец вектора, и удален только из этого конца вектора (отметим, что именно стек лежит в основе организации рекурсивных процедур);

2) *очередь*, если элемент может быть добавлен только в конец вектора, а удален только из другого конца вектора;

3) *дек*, если добавление и удаление элемента возможно для любого из концов вектора.

Операция удаления элемента из вектора осуществляется за время, определенное формулой (1). С операцией добавления элемента в вектор ситуация следующая. Если размер n вектора меньше некоторого числа n_0 (определяемого используемым ПО), то время выполнения этой операции определяется формулой (1). Если же размер вектора достиг числа n_0, то для выполнения операции добавления элемента требуется выделить совершенно новый кусок памяти и перенести туда все элементы вектора, т.е. операция добавления элемента выполнима за время

$$T = \begin{cases} O(n \log n), \text{ если } n \to \infty, \ |S| \text{ фиксировано;} \\ O(n \log |S|), \text{ если } |S| \to \infty, \ n \text{ фиксировано;} \\ O(n \log n \cdot \log |S|), \text{ если } n \to \infty \text{ и } |S| \to \infty. \end{cases} \quad (2)$$

3. Ассоциативный массив. Является динамической структурой, в которой реализована возможность обращения к значению по ключу (при этом ключом может быть практически любой тип данных). В современных системах web-разработки под массивом понимают именно ассоциативный массив.

4. Хеш-массив. Является ассоциативным массивом, в котором данные распределены по m «корзинам». Выбор корзины осуществляет хэш-функция, аргумент которой – ключ, а значение – номер корзины. Хэш-функция должна быть, во-первых, легко вычислимой, а, во-вторых, равномерно распределять ключи по корзинам. Такая конструкция дает возможность уменьшать время вычисления в определенное число раз.

Вектор является основой для построения *списков*. Выделяют следующие типы списков:

1) *односвязный список* состоит из двух векторов одного и того же размера; 1-й вектор хранит элементы списка, а 2-й вектор – указатели либо на следующий элемент списка (правосторонний список), либо на предыдущий элемент списка (левосторонний список);

2) *двусвязный список* состоит из трех векторов одного и того же размера; 1-й вектор хранит элементы списка, 2-й вектор – указатели на предыдущий элемент списка, а 3-й вектор – указатели на следующий элемент списка.

Время вставки элемента в список – такое же, как и для вектора. Удаление элемента и расщепление списка на два подсписка по значению указателя, выполнимы за время $T = O(n \log n)$ $(n \to \infty)$. Конкатенация списков размеров n_1 и n_2 выполнима за время $T = O(n_1 \log n_1)$ $(n_1 \to \infty)$.

Списки представляют собой основу для построения эффективных структур данных, предна-

значенных для представления множеств, деревьев и графов (см., напр., [32-35]).

В современных связках «БД–СУБД» (особенно предназначенных для работы с хранилищами данных) используются специальные библиотеки функций, в основе которых лежит обработка списков.

Типичным таким примером является библиотека MapReduce [37-39], разработанная в компании Google. Ее основой являются функции map и reduce. Входом функции map является список значений и функция-трансформатор, а выходом – список результатов применения трансформатора к каждому значению. Вход функции reduce – список значений, стартовое значение аккумулятора и функция свертки (принимающая два значения и возвращающая одно). Функция reduce последовательно передаёт в функцию свертки значение аккумулятора и очередное значение из списка, а результат помещает в аккумулятор. Результат выполнения функции reduce – финальное значение аккумулятора. Практика показывает, что широкий класс сложных задач обработки данных эффективно сводится к последовательному выполнению комбинаций функций map и reduce.

3. Математические модели связок «БД-СУБД»

Формальный подход к теоретическому анализу алгоритмов преобразования информации основан на симбиозе математической логики, теории моделей и прикладной теории алгоритмов. Такой подход, по своей сути, состоит в следующем [40-48].

Над фиксированным множеством M конечных структур (т.е. моделей) строится теория (по крайней мере 1-го порядка) T. Множество формул этой теории представляет собой некоторый язык L. Основными являются следующие три задачи:

1) *проверка выполнимости* (satisfiability checking), состоящая в том, чтобы ответить на вопрос: существует ли модель $S \in M$, для которой формула $\psi \in L$ выполнима?

2) *проверка модели* (model checking), состоящая в том, чтобы ответить на вопрос: верно ли, что для модели $S \in M$ и формулы $\psi \in L$ истинно соотношение $S \models \psi$ (т.е. формула ψ истинна в модели S)?

3) *эволюция запроса* (query evolution), состоящая в том, чтобы для модели $S \in M$ и формулы $\psi(\vec{x}) \in L$ (где $\vec{x}$ – множество свободных переменных) вычислить множество $\{\vec{a} \mid S \models \psi(\vec{a})\}$ (состоящее, говоря содержательно, из всех значений свободных переменных формулы ψ, при которых эта формула истинна в модели S).

Входными данными для проверки модели являются $S \in M$ и $\psi \in L$. Поэтому естественно возникают следующие задачи:

1) как выражается сложность решения задачи проверки модели через сложность модели $S \in M$ и формулы $\psi \in L$ (такая сложность называется комбинированной сложностью (combined complexity))?

2) какова сложность вычисления множества $\{\psi \in L \mid S \models \psi\}$ для заданной модели $S \in M$ (такая сложность называется выразительной сложностью (expression complexity))?

3) какова сложность вычисления множества $\{S \in M \mid S \models \psi\}$ для заданной формулы $\psi \in L$ (такая сложность называется структурной сложностью (structure complexity))?

В случае анализа связки «БД–СУБД» построение теории T осуществляется над некоторой алгеброй $A = (\Xi, B)$, где Ξ – множество структур данных, а B – множество базовых алгоритмов их обработки. Алгебра A представляет собой основу для построения некоторого множества (конечных) структур D. Объектом исследования является язык запросов L, а предметом исследования – анализ разрешимости и выразительности языка L [49-53], анализ временной сложности алгоритмов, представленных формулами языка L [54-60], а также выделение тех классов формул языка L, которые представляют запросы, реализуемые с полиномиальной временной сложностью [61-65].

Такой подход исследовался в деталях для основных моделей данных: иерархической [66-69], сетевой [70-74] и реляционной [75-89]. В п. 1 было отмечено, что в настоящее время «промышленным стандартом» является реляционная модель связки «БД–СУБД». Этому обстоятельству (кроме успешных применений на практике) способствовало теоретическое обоснование преимущества реляционной модели перед остальными двумя, установленное в процессе исследования перечисленных выше задач.

Кроме того, именно исследование указанных выше трех моделей данных сформировало теорию сложности интервального поиска в терминах информационных графов [90-96], предназначенную, в частности, для теоретического обоснования выбора физической организации данных.

Рассмотренный выше формальный подход получил дальнейшее развитие (и существенное усложнение) в процессе построения математических моделей *многомерной* связки «БД–СУБД».

В [28], исходя из анализа существующих моделей хранилищ данных [97-109], построена следующая общая модель многомерной связки «БД–СУБД» в терминах теории категорий, что дает возможность эффективно использовать язык схем.

Основа этой модели – многомерный объект. Он состоит из согласованных между собой схемы фактов $S=(F,D)$ (где F – множество типов фактов, а D – частично упорядоченное множество категорий типов размерности, содержащее наименьший и наибольший элементы) и множеств F, D и R, соответственно, фактов, размерностей и отношений «факт-размерность». На многомерном объекте определена алгебра. Доказано, что эта алгебра замкнута (т.е. что результат выполнения операций над многомерным объектом – многомерный объект) и что она является не менее мощной, чем реляционная алгебра с функциями агрегирования.

Посредством операторов квантования времени транзакций и времени валидации выделяется согласованное во времени семейство многомерных объектов. Для этого семейства развит математический аппарат, предназначенный для регулирования оценок запросов и представления неточных результатов. Показано, что выполнены все основные требования, предъявляемые к современным моделям многомерных связок «БД–СУБД». Рассмотрены особенности реализации построенной модели на основе использования технологии построения реляционных связок «БД–СУБД».

Отметим, что большая внутренняя сложность теоретического анализа математических моделей связок «БД–СУБД» в значительной мере обусловлена следующим обстоятельством, которое остается «в тени» в работах, посвященных исследованию этой задачи.

Принципиальные ограничения применения при разработке ПО теорий, основанных на классической логике, привели к разработке композиционного подхода в программировании [110]. Важной составляющей этого подхода являются исчисления квазиарных предикатов [111]. В основе таких исчислений лежат частичные отображения, заданные на произвольных наборах именованных значений, т.е., по своей сути, частичные словарные функции. В настоящее время уделяется большое внимание дескриптивному аспекту исследования исчислений квазиарных предикатов. Однако алгоритмический и метрический аспекты исследования этих исчислений (т.е. разработка алгоритмов, основанных на исчислении квазиарных предикатов, и анализ их сложности) находятся только в стадии становления.

4. «Алгоритмы оптимизации» запросов

Одной из основных сложных задач, связанных с появлением связок «БД–СУБД», является «оптимизация обработки» запросов СУБД. Слово «оптимизация» является математическим термином и подразумевает выбор наилучшего (или близкого к нему в соответствии с заданным критерием) объекта или алгоритма, принадлежащего формально определенному классу, соответственно, объектов или алгоритмов. Ясно, что такой подход неприменим ни с теоретической, ни с практической точки зрения к связке «БД–СУБД» (как и ко многим другим прикладным системам). Поэтому под «оптимизацией обработки» запросов СУБД понимают стратегии, предназначенные для повышения эффективности процедур обработки запросов. Такие стратегии представляют собой эвристические методы, целесообразность использования которых обосновывается статистическими методами, т.е. обработкой результатов проведенных экспериментов.

Понятие «запрос», по своей сути, является выражением некоторого языка и рассматривается, по крайней мере, в следующих трех контекстах:

1) как стандартное требование доступа пользователя к данным БД (в этом случае «оптимизация» осуществляется за счет программирования соответствующих процедур поиска данных и преобразования входа пользователя в результат требуемого формата);

2) как транзакция, осуществляющая изменение данных БД на основании их текущего значения;

3) как выражение, которое СУБД использует для авторизации пользователя [112], обеспечения целостности данных [113] и синхронизации многопользовательского доступа к БД [114].

Для того, чтобы говорить об «оптимизации обработки» запросов, прежде всего требуется понимать, что понимается под «стоимостью» запроса. Выделяют следующие составляющие этого понятия:

1) *коммуникационная стоимость*, т.е. стоимость передачи данных из их местоположения во вторичную память (т.е. память, из которой загружаются данные для вычисления) плюс стоимость перемещения результата в его местоположение;

2) *стоимость доступа к вторичной памяти*, т.е. стоимость загрузки порций данных из вторичной памяти в основную память, используемую при вычислении;

3) *стоимость запоминания*, т.е. стоимость времени использования вторичной памяти и памяти буферов (что особенно актуально для связок «БД–СУБД», обладающих «узкими местами», изменяющимися в зависимости от запросов);

4) *стоимость вычисления*, т.е. стоимость времени, используемого непосредственно для вычисления.

Таким образом, «оптимизация обработки» запросов представляет собой нетривиальную многокритериальную задачу.

Методы «оптимизации обработки» запросов,

разработанные в [115-120], привели к пониманию необходимости разработки полномасштабного унифицированного подхода к решению этой задачи, основанного на нисходящем подходе (top-down approach). Анализ состояния этих исследований на 1984 г. представлен в обзоре [121].

В течение последнего 20-летия XX столетия сформировались следующие три направления исследования задачи «оптимизация обработки» запросов.

I. Разработка методов «оптимизации обработки» последовательности запросов. Формирование этого направления обусловлено попыткой снижения стоимости обработки запроса за счет использования того фактора, что, как правило, в последовательности составных запросов содержится значительное количество общих подвыражений.

Первые методы «оптимизации обработки» последовательности запросов были основаны на исчерпывающемся поиске [122-126]. Эти методы использовали дважды экспоненциальную память от размера запроса и являлись неэффективными на практике. Ощутимый прорыв произошел в связи с разработкой методов «оптимизации обработки» последовательности запросов, основанных на использовании представления запросов ориентированными ациклическими И-ИЛИ графами (AND-OR DAG representation) [127-135]. Некоторые из них представляют собой прототипы методов, реализованных в SQL Microsoft Server. Более того, именно разработка этих методов дала возможность выработать ряд общих рекомендаций увеличения производительности SQL Microsoft Server 6.5, основные из которых состоят в следующем:

1) выделите серверу столько оперативной памяти, сколько он выдержит;

2) используйте массивы RAID (эти массивы распределяют запросы на чтение по нескольким физическим дискам) уровня 0 или 5 для распараллеливания получения информации из БД;

3) обеспечьте для функции Max Async I/O возможность использования всех преимуществ вашего компьютера;

4) установите пороги расширения блокировок (Lock Escalation) на всю таблицу, позволяющие избежать «накладных расходов», связанных со значительным числом блокировок;

5) создайте кластеризованные индексы для запросов, считывающих диапазоны значений;

6) выделите некластеризованные индексы для запросов на поиск уникальных значений;

7) создайте составные индексы для поддержания последовательности запросов, что дает существенный выигрыш, когда, в основном, осуществляется чтение данных, и выполняются операции UPDATE и INSERT;

8) индексируйте соединенные столбцы, что дает возможность существенно уменьшить (иногда на несколько порядков) время выполнения соединения таблиц;

9) используйте преимущества покрывающих индексов, т.е. содержащих все столбцы, указанные в операторах SELECT, UPDATE или DELETE, что уменьшает время выполнения этих операций до тех пор, пока суммарная длина всех входящих в индекс столбцов значительно меньше, чем длина строки таблицы.

II. Разработка методов лексической и семантической «оптимизации» запросов. Лексическая «оптимизация» запроса состоит в его преобразовании, направленном на устранение избыточности на основе анализа ограничений и условий, содержащихся в нем. Существуют следующие два подхода к решению этой задачи:

1. Преобразование запроса, представленного на нереляционном языке (а при необходимости и данных), к реляционной форме, и применение к ней методов лексической «оптимизации», разработанных для реляционных СУБД [136-139]. Обзор работ, посвященных таким методам, содержится в [140].

2. Построение новых алгебр, предназначенных для представления запросов с учетом особенностей новых языков, и разработка методов «оптимизации» выражений этих алгебр [141-144]. Обзор работ, посвященных таким методам, содержится в [145, 146].

Семантическая «оптимизация» запроса представляет собой валидацию и преобразование синтаксического дерева запроса к виду, пригодному для выполнения дальнейших шагов оптимизации. При этом осуществляется преобразование запросов в каноническую форму (т.е. раскрытие представлений, преобразование подзапросов в соединения, спуск предикатов), упрощение условий, распределение предикатов и преобразование дерева условий в пути выборки [147-149].

III. Разработка методов «оптимизации обработки» запросов в параллельной распределенной связке «БД–СУБД». По-видимому, именно эти исследования определили следующие три направления исследований, наиболее интенсивно развивающиеся в настоящее время.

Во-первых, это разработка детерминированных и вероятностных методов «оптимизации» запросов для различных моделей параллельных распределенных связок «БД–СУБД» [150-158]. Отметим, что в работе [157] предложен способ выбора архитектуры параллельной распределенной связки «БД–СУБД» по критерию стоимости с ограничением на верхнюю границу доверительного интервала времени выполнения запроса. При этом для конфигурации SE [152,

156] исследована зависимость математического ожидания времени выполнения запроса от числа процессоров. Показано, что с ростом числа процессоров сначала время убывает благодаря распараллеливанию обработки запроса, а затем возрастает из-за перегрузки системы ввода/вывода, которая является разделяемым ресурсом.

Во-вторых, это построение моделей связок «БД–СУБД» на основе использования графических процессоров (GPU) [159-165]. В состав обычного CPU входят от 1 до 16 процессоров, а в состав GPU – несколько сотен простых процессоров. Под «простыми» понимаются процессоры, не содержащие компонент проверки условий, кэш и т.д. Архитектура GPU представляет собой SIMT (single instruction multiple thread). Имея функцию, выполняющую некоторую работу, разработчик назначает ее исполнение на тот или иной GPU. Модель программирования GPU описывает 6 типов областей памяти, различных по скорости доступа, прав на запись-чтение, объему. Наличие большого числа процессоров и быстрой памяти обеспечивает эффективность выборки данных со сложным условием, так как каждый поток исполняется независимо от других, оперируя только с малой частью обрабатываемых данных. В результате существенно снижается различие между временем загрузки данных с медленного носителя в оперативную память и временем, затрачиваемым на исполнения запроса и возвращения результата.

В-третьих, это разработка моделей и методов обработки и хранения больших данных. Под термином «большие данные» понимают электронные данные, характеризующиеся большим объемом, разнообразием и скоростью, с которой структурированные и неструктурированные данные поступают по сетям передачи в процессоры и хранилища, а также наличием эффективных процессов переработки данных в требуемую информацию [166]. Типичными примерами больших данных являются системы компьютерной поддержки функционирования ведущих фондовых бирж мира, социальная сеть Facebook, проект Internet Archive, системы компьютерной поддержки уникальных экспериментальных установок исследования процессов микромира, таких, как Большой андронный коллайдер [167].

Именно актуальность разработки средств эффективной обработки больших данных привела к появлению концепции NoSQL (Not Only SQL) [168-170]. Ее суть состоит в проектировании архитектуры, обладающей свойством адаптации к возрастающим объемам данных. При этом сохранение эффективности процессов обработки данных обеспечивается за счет высокой пропускной способности и потенциально неограниченного горизонтального масштабирования. В [171] предложена следующая классификация существующих в настоящее время NoSQL решений по типу используемых хранилищ:

1. Хранилище ключ-значение. Представляет собой «словарь», предназначенный для работы с данными по ключу. Дает возможность обеспечить высокую производительность обработки данных (за счет усложнения структуры запросов, так как в словаре отсутствует информация о структуре значений данных).

2. Документное хранилище. Представляет собой словарь с логическим объединением множеств пар ключ-значение в структуру, называемую «документ». Документы могут иметь вложенную структуру и объединяться в коллекции. Работа с документами осуществляется по ключу и/или по значениям атрибутов.

3. Колоночные хранилища. Значения данных хранятся в виде интерпретируемых байтовых массивов, адресуемых кортежами вида (a_1, a_2, a_3), где a_1 – ключ строки, a_2 – ключ столбца и a_3 – метка времени [172]. Основой модели является «колонка». Число колонок в таблице потенциально неограниченно. Колонки по ключам объединяются в семейства, обладающие заданным набором свойств. Отметим, что значительное число публикаций, посвященных «оптимизации запросов» именно для этой модели (см., напр., [173-177]), обусловлено ее большой схожестью с реляционными связками «БД–СУБД».

4. Хранилища на графах. Используются для объектов с четко выраженной сетевой структурой (таких, как социальные сети, системы управления авиаперелетами, стратегические системы обеспечения безопасности страны, глобальные системы оповещения о природных катастрофах и т.д.). Модель состоит из вершин (в которых сосредоточены данные) и ребер (размеченных свойствами). Работа с данными осуществляется методом того или иного обхода графа по ребрам с заданными свойствами.

Обзор существующих разработок, предназначенных для работы с большими данными, содержится в [178]. Отметим, что в настоящее время уделяется значительное внимание разработкам, предназначенным для обработки потоков событий в реальном времени.

Заключение

В настоящей работе сделана попытка охарактеризовать современное состояние проблемы исследования сложности операций в связках «БД–СУБД». Значимость этой проблемы состоит в том, что, в конечном итоге, именно ее решение представляет

собой основу обеспечения эффективности функционирования связок «БД–СУБД» на протяжении их жизненного цикла.

В работе кратко рассмотрены основные этапы проектирования связки «БД–СУБД», основные модели данных и теоретические основы анализа сложности алгоритмов. С этих позиций охарактеризованы математические модели, предназначенные для теоретического обоснования сложности исполнения запросов связкой «БД–СУБД». В контексте направлений исследований, наиболее интенсивно развивающихся в настоящее время, рассмотрены существующие эвристические методы «оптимизации обработки» запросов связкой «БД–СУБД». По-видимому, общим недостатком всех таких математических моделей и эвристических методов является отсутствие средств, предназначенных для эффективного представления особенностей функционирования связки «БД–СУБД» в конкретной реальной среде. По этой причине на практике основным является тестирование эффективности связки «БД–СУБД» в реальных условиях.

Следует отметить, что практически отсутствуют публикации, содержащие данные о производительности в реальных условиях даже таких широко известных реляционных СУБД, как Oracle и Microsoft SQL Server. Более того, авторы настоящей статьи не располагают информацией о существовании методик тестирования в реальных условиях для постреляционных связок «БД–СУБД».

Таким образом, современное состояние проблемы исследования сложности операций в связках «БД–СУБД» весьма далеко от того, что принято называть проработанной технологией.

Вне рамок данного обзора остался ряд аспектов, связанных с проблемой исследования сложности операций в связках «БД–СУБД». Укажем на следующие два из них.

Во-первых, это исследование рассматриваемой проблемы в контексте связок «БД–СУБД», предназначенных для использования в экспертных системах. Такая связка является дедуктивной, т.е. состоит из двух частей: экстенциональной, содержащей факты, и интенциональной, содержащей правила логического вывода новых фактов на основе экстенциональной части и запроса пользователя. При этом правила интенциональной части, а также запросы пользователей могут содержать рекурсию. Отметим, что интенциональную часть дедуктивной связки «БД–СУБД» можно рассматривать как базу знаний.

Во-вторых, это исследование рассматриваемой проблемы в контексте использования подхода, основанного на «сборке» ПО из готовых, возможно использовавшихся ранее, компонент [179-181].

Литература

1. Исаченко, А. Н. Модели данных и СУБД [Текст] / А. Н. Исаченко, С. П. Бондаренко. – Минск : Изд-во БГУ, 2007. – 205 с.

2. Мартин, Дж. Организация баз данных в вычислительных системах [Текст] / Дж. Мартин. – М. : Мир, 1980. – 622 с.

3. Ульман, Дж. Основы систем баз данных [Текст] / Дж. Ульман. – М. : Финансы и статистика, 1983. – 334 с.

4. Мейер, Д. Теория реляционных баз данных [Текст] / Д. Мейер. – М. : Мир, 1987. – 540 с.

5. Редько, В. Н. К основаниям теории реляционных моделей баз данных [Текст] / В. Н. Редько, Д. Б. Буй // Кибернетика и системный анализ. – 1996. – № 4. – С. 3-12.

6. Хансен, Г. Базы данных: разработка и управление [Текст] / Г. Хансен, Дж. Хансен. – М. : Бином, 1999. – 504 с.

7. Яргер, Р. MySQL и mSQL. Базы данных для небольших предприятий и Интернета [Текст] / Р. Яргер, Дж. Риз, Т. Кинг. – СПб. : Символ-Плюс, 2000. – 560 с.

8. Дейт, К. Введение в системы баз данных [Текст] / К. Дейт. – М. : Вильямс, 2001. – 1072 с.

9. Власовец, А. М. Основы проектирования реляционных баз данных [Текст] / А. М. Власовец. – СПб. : Изд-во СПбГУЭФ, 2001. – 144 с.

10. Карпова, Т. С. Модели, разработка, реализация [Текст] / Т. С. Карпова. – СПб. : Питер, 2001. – 304 с.

11. Конноли, Т. Базы данных: проектирование, реализация и сопровождение. Теория и практика [Текст] / Т. Конноли, К. Бег. – М. : Вильямс, 2001. – 1120 с.

12. Кудрявцев, К. Я. Создание бах данных [Текст] / К. Я. Кудрявцев. – М. : НИЯУ МИФИ, 2010. – 155 с.

13. Database security [Text] / S. Castano, M. G. Fugini, G. Martello et al. – N.Y. : Addison Wesley Publishing Company, 1995. – 456 p.

14. Глушков, В. М. Синтез цифровых автоматов [Текст] / В. М. Глушков. – М. : Физматлит, 1962. – 476 с.

15. Chen, P. P. The entity-relationship model: towards a unified view of data [Text] / P. P. Chen // ACM Transactions on Database Systems. – 1976. – № 1. – P. 9-36.

16. Tsichritzis, D. C. Hierarchical data-base management: a survey [Text] / D. C. Tsichritzis, F. H. Lochovsky // ACM Computing. Survey – 1976. – № 8. – P. 67-103.

17. Bachman, C. W. Software for random access processing [Text] / C. W. Bachman // Datamation. – 1965.– № 11. – P. 36-41.

18. Bachman, C. W. Data structure diagrams [Text] / C. W. Bachman // Data Base. – 1969. – № 2. – P. 4-10.

19. Риге, Ж. Бинарные отношения, замыкания,

соответствия Галуа [Текст] / Ж. Риге // Кибернетический сборник. – 1963. – Вып. 7. – С. 129-185.

20. Вагнер, В. В. Теория отношений и алгебра частичных отображений [Текст] / В. В. Вагнер // Теория полугрупп и ее приложения. – 1965. – Вып. 1. – С. 3-178.

21. Codd, E. F. A relationship model for large stored date banks [Text] / E. F. Codd // Communications of the ACM. – 1970. – № 13. – P. 377-387.

22. Booch, G. Object-Oriented Development [Text] / G. Buch // IEEE Transactions on Software Engineering. – 1986. – № 2. – P. 211-221.

23. Booch, G. Object-Oriented Analysis and Design with Applications [Text] / G. Buch. – Redwood City : Benjamin/Cummings, 1991. – 691 p.

24. Бадд, Т. Объектно-ориентированное программирование в действии [Текст] / Т. Бадд. – СпБ. : Питер, 1997. – 464 с.

25. Грэхем, И. Объектно-ориентированные методы: принципы и практика [Текст] / И. Грэхем. – М. : Вильямс, 2004. – 880 с.

26. Chamberlin, D. Anatomy of an object-relational database [Text] / D. Chamberlin // DB2 Magazine. – 1996. – № 1. – P. 24-37.

27. Thomsen, E. OLAP solutions: building multidimensional information systems [Text] / E. Thomsen. – NY. : John Wiley & Sons, 1997. – 696 p.

28. Pedersen, T. B. A foundation for capturing and querying complex multidimensional data [Text] / T. B. Pedersen, C. S. Jensen, C. E. Dyerson // Information Systems. – 2001. – № 5. – P. 383-423.

29. Кнут, Д. Искусство программирования для ЭВМ. Т. 1 : Основные алгоритмы [Текст] / Д. Кнут. – М. : Мир, 1976. – 735 с.

30. Кнут, Д. Искусство программирования для ЭВМ. Т. 2 : Получисленные алгоритмы [Текст] / Д. Кнут. – М. : Мир, 1977. – 724 с.

31. Кнут, Д. Искусство программирования для ЭВМ. Т. 3 : Сортировка и поиск [Текст] / Д. Кнут. – М. : Мир, 1978. – 844 с.

32. Ахо, А. Построение и анализ вычислительных алгоритмов [Текст] / А. Ахо, Дж. Хопкрофт, Дж. Ульман. – М. : Мир, 1979. – 536 с.

33. Рейнгольд, Э. Комбинаторные алгоритмы. Теория и практика [Текст] / Э. Рейнгольд, Ю. Нивергельт, Н. Део. – М. : Мир, 1980. – 476 с.

34. Кормен, Т. Алгоритмы: построение и анализ [Текст] / Т. Кормен, Ч. Лейзерсон, Р. Ривест. – М. : МНЦМО, 2000. – 960 с.

35. Чаудхари, С. Методы оптимизации запросов в реляционных системах [Текст] / С. Чаудхари // СУБД. – 1998. – № 3. – С. 22-36.

36. Кузнецов, С. Д. Методы оптимизации выполнения запросов в реляционных СУБД [Текст] / С. Д. Кузнецов // Итоги науки и техники. Вычислительные науки. Т. 1. – М. : ВИНИТИ, 1989. – С. 76-153.

37. Ghemawat, S. The Google file system [Text] / S. Ghemawat, H. Gobioff, S.-T. Lueng // Proc. of ACM Symposium on Operating Systems Principles. – 2003. – P. 29-43.

38. Dean, J. MapReduce: implified data processing on large clusters [Text] / J. Dean, S. Ghemawat // Proc. of the 6th Symposium on Operating Systems Design and Implementation. – 2004. – P. 137-150.

39. Кузнецов, С. Д. MapReduce: внутри, снаружи или сбоку от параллельных СУБД? [Текст] / С.Д. Кузнецов // Труды института системного программирования РАН. – 2010. – Т. 19. – С. 35-70.

40. Aho, A. Universality of data retrieval languages [Text] / A. Aho, J. Ullman // Proc. of the 6th ACM Symposium on Principles of Programming Languages. – 1979. – P. 110-117.

41. Abiteboul, S. A transaction language for data base update and specification [Text] / S. Abiteboul, V. Vianu // Proc. of the 6th ACM Symposium on Principles of Database Systems. – 1987. – P. 260-268.

42. Успенский, В. А. Теория алгоритмов: основные открытия и приложения [Текст] / В. А. Успенский, А. Л. Семенов. – М. : Наука, 1987. – 288 с.

43. Derougemont, M. Computability and databases [Text] / M. Derougemont // Technological Science Information. – 1987. – № 8. – P. 609-622.

44. Fagin, R. Finite model theory [Text] / R. Fagin // LNCS. – 1990. – V. 470. – P. 3-24.

45. Abiteboul, S. Datalog extensions for database queries and updates [Text] / S. Abiteboul, V. Vianu // Journal of Computer and Systems Sciences. – 1991. – № 1. – P. 62-124.

46. Kanellakis, P. Constraint query languages [Text] / K. Kanellakis, G. Cuper, P. Revesz // Journal of Computer and System Sciences. – 1995. – № 1. – P. 26-52.

47. Andreka, H. Modal languages and bounded fragments of predicate logic [Text] / H. Andreka, J. van Benthem, I. Nemeti // Journal of Philosophical Logic. – 1998. – № 2. – P. 217-274.

48. Bradfield, J. Modal logic and mu-calculi [Text] / J. Bradfield, C. Stirling. – In: Handbook of process algebra. – Elsevier, 2001. – P. 293-332.

49. Arvind, V. Expressibility of 1st order logic with a non-deterministic inductive operator [Text] / V. Arvind, S. Biswas // LNCS. – 1987. – V. 247. – P. 323-335.

50. Shmueli, O. Decidability and expressiveness aspects of logic queries [Text] / O. Shmueli // Proc. of the 6th ACM Symposium on Principles of Database Systems. – 1987. – P. 237-249.

51. Shauble, P. On the expressive power of query languages [Text] / P. Shauble, B. Wuthrich // Theory of Information Systems. – 1994. – № 1. – P. 69-91.

52. Замятин, А. П. Элементы математической теории информационных систем: выразимость и вычислимость [Текст] / А. П. Замятин, А. Б. Ливчак. – Екатеринбург : УрГУ, 1996. – 104 с.

53. Тайцлин, М. А. Сравнение выразительной силы некоторых языков запросов для баз данных [Текст] / М. А. Тайцлин // Труды математического института им. В.А. Стеклова. – 2011. – Т. 274.

– С. 297-313.

54. Gurevich, Y. Towards logic tailored for computational complexity [Text] / Y. Gurevich // Lecture Notes in Mathematics. – 1984. – V. 1104. – P. 175-216.

55. Itai, A. Unification as complexity measure for logic programming [Text] / A. Itai, J. Makowsky // Journal of Logic Programming. – 1987. – № 4. – P. 105-117.

56. Vanbenthen, J. Time, logic and computation [Text] / J. Vanbenthen // LNCS. – 1989. – V. 354. – P. 1-49.

57. Varvel, D. L. The computational completeness of extended database query languages [Text] / D. L. Varvel, L. Shapiro // IEEE Transactions on Software Engineering. – 1989. – № 1. – P. 632-638.

58. Abiteboul, S. Fixpoint logics, relational machines, and computational complexity [Text] / S. Abiteboul, M. Y. Vardi, V. Vianu // Proc. of the 7th Annual Conference on Structure in Complexity Theory. – 1992. – P. 156-158.

59. Gradel, E. Capturing complexity classes by fragments of the second-order logic [Text] / E. Gradel // Theoretical Computer Science. – 1992. – № 1. – P. 35-57.

60. Dziembowski, S. Bounded-variable fixpoint queries are PSPACE-complete [Text] / S. Dziembowski // LNCS. – 1996. – V. 1258. – P. 89-105.

61. Fagin, R. Generalized first-order spectra and polynomial-time recognizable sets [Text] / R. Fagen // SIAM-AMS Processing. – 1974. – V. 7. – P. 43-73.

62. Gurevich, Y. Algebras of feasible functions [Text] / Y. Gurevich // Proc. of the 24th Symposium on Foundations of Computer Science. – 1983. – P. 210-214.

63. Afrati, F. On Datalog vs polynomial time [Text] / F. Afrati, S. S. Cosmadakis, M. Yannakakis // Proc. of the 10th ACM Symposium on Principals of Database Systems. – 1991. – P. 13-25.

64. Hella, L. Logical hierarchies in PTIME [Text] / L. Hella // Proc. of the 7th IEEE Symposium on Logic in Computer Science. – 1992. – P. 360-368.

65. Gottlob, G. Datalog LITE: A deductive query language with linear time model checking [Text] / G. Gottlob, E. Gradel, H. Veith // ACM Transactions on Computational logic. – 2002. – № 3. – P. 42-79.

66. Arnold, A. The mu-calculus alternation-depth is strict on binary trees [Text] / A. Arnold // RAIRO Informatique Theorique at Applications. – 1999. – № 2. – P. 329-339.

67. Dahlhaus, E. Query languages for hierarchical databases [Text] / E. Dahlhaus, J. A. Makowsky // Information and Computation. – 1992. – № 1. – P. 1-32.

68. Dublish, P. Query languages which express all P-time queries for trees and unicyclic graphs [Text] / P. Dublish, S. N. Maheswari // LNCS. – 1987. – V. 452. – P. 246-253.

69. Grohe, M. Definability and descriptive complexity on databases of bounded tree-width [Text] / M. Grohe, J. Marino // LNCS. – 1999. – V. 1540. – P. 70-82.

70. Courselle, B. The monadic second order logic of graphs II: infinite graphs of bounded width [Text] / B. Courselle // Mathematical Systems Theory. – 1989. – № 1. – P. 187-221.

71. Cay, J. An optimal lower bound on the number of variables for graph identification [Text] / J. Cay // Combinatorica. – 1992. – № 12. – P. 389-410.

72. Courselle, B. The monadic second order logic of graphs IX: machines and their behaviors [Text] / B. Courselle // Theoretical Computer Science. – 1995. – V. 151. – P. 125-162.

73. Causal, D. On infinite transition graphs having a decidable monadic theory [Text] / D. Causal // LNCS. – 1996. – V. 1099. – P. 194-205.

74. Courselle, B. Monadic second order logic, graph coverings and unfolding of transition systems [Text] / B. Courselle, I. Walukiewicz // Annals of Pure and Applied Logic. – 1998. – № 1. – P. 35-62.

75. Paradaens, J. On the expressive power of relational algebra [Text] / J. Paradaens // Information Processing Letters. – 1978. – № 2. – P. 107-111.

76. Banchilhon, F. On completeness of query languages for relational databases [Text] / F. Banchilhon // LNCS. – 1978. – V. 64. – P. 119-123.

77. Chandra, A. Computable queries for relational data bases [Text] / A. Chandra, D. Harel // Journal of Computer and System Sciences. – 1980. – № 1. – P. 156-178.

78. Бениаминов, Е. М. Алгебраический подход к моделям баз данных реляционного типа [Текст] / Е. М. Бениаминов // Семиотика и информатика. – 1980. – Вып. 14. – С. 44-80.

79. Борщев, Б. В. Логический подход к описанию реляционных баз данных [Текст] / Б. В. Борщев // Семиотика и информатика. – 1981. – Вып. 16. – С. 78-122.

80. Ливчак, А. Б. Реляционные модели баз данных и полиномиальная вычислимость [Текст] / А. Б. Ливчак // НТИ. Сер. 2. Информационные процессы и системы. – 1981. – № 6. – С. 28-29.

81. Vardi, M. Complexity of relational query languages [Text] / M. Vardi // Proc. of the 14th ACM Symposium on the Theory of Computing. – 1982. – P. 137-146.

82. Chandra, A. Structure and complexity of relational queries [Text] / A. Chandra, D. Harel // Journal of Computer and System Sciences. – 1982. – № 1. – P. 99-128.

83. Ливчак, А. Б. Полиномиальные запросы к реляционным базам данных [Text] / А. Б. Ливчак // Программирование. – 1985. – № 2. – С. 66-72.

84. Immerman, N. Relational queries computable in polynomial time [Text] / N. Immerman // Information and Control. – 1986. – № 1. – P. 86-104.

85. Ливчак, А. Б. Сложностный анализ реляционных языков [Текст] /, А. Б. Ливчак // Вопросы кибернетики. – 1989. – Вып. 149. – С. 67-85.

86. Барышников, А. Ю. Языки PQL и FO+LFP эквивалентны и без наличия порядка [Текст] / А. Ю. Барышников, А. Б. Ливчак // Известия Вузов.

Математика. – 1996. – № 4. – С. 18-23.

87. Ливчак, А. Б. Недетерминированные запросы к реляционным базам данных [Текст] / А. Б. Ливчак, А. Я. Овсянников // НТИ. Сер. 2. Информационные процессы и системы. – 1994. – № 7. – С. 19-23.

88. Ливчак, А. Б. Усиление теорем о полиномиальных запросах [Текст] / А. Б. Ливчак // Программирование. – 1995. – № 3. – С. 9-16.

89. Definable relations and first-order query languages over strings [Text] / M. Benedict, L. Libkin, T. Schwentick, L. Segofin // Journal of the ACM. – 2003. – № 4. – P. 694-751.

90. Cucker, F. Logic which capture complexity classes over the reals [Text] / F. Cucker, K. Meer // Journal of Symbolic Logic. – 1999. – V. 2. – P. 363-390.

91. Гасанов, Э. Э. Теория хранения и поиска информации [Текст] / Э. Э. Гасанов, В. Б. Кудрявцев. – М. : Физматлит, 2002. – 288 с.

92. Фещук, А. А. К вопросу анализа нечетких информационных графов [Текст] / А. А. Фещук // Дискретная математика. – 2002. – № 2. – С. 65-84.

93. Гасанов, Э. Э. Теория сложности информационного поиска [Текст] / Э. Э. Гасанов. – М. : МГУ, 2005. –144 с.

94. Лапшов, И. С. Динамические базы данных, основывающиеся на хешировании методом цепочек [Текст] / И. С. Лапшов // Интеллектуальные системы. – 2005. – Т. 9. – С. 191-207.

95. Кучеренко, Н. С. Оценки сложности поиска идентичных объектов для случайных баз данных [Текст] / Н. С. Кучеренко // Материалы IX Международной конференции «Дискретная математика и ее приложения». – М. : МГУ. – 2007. – С. 329-331.

96. Кучеренко, Н. С. Сложность поиска идентичных объектов в случайных базах данных [Текст] / Н. С. Кучеренко // Интеллектуальные системы. – 2007. – Т. 11. – С. 495-516.

97. Rafanelli, M. STORM: a statistical object representation model [Text] / M. Rafanelli, A. Shoshani // Proc. of the 5th International Conference on Statistical and Scientific Database Management. – 1990. – P. 14-29.

98. Dyreson, C. E. Information retrieval from an incomplete cube [Text] / C. E. Dyreson // Proc. of the 22nd International Conference on Very Large Databases. – 1996. – P. 532-543.

99. Li, C. A data model for supporting on-line analytical processing [Text] / C. Li, X.S. Wang // Proc. of the 5th International Conference on Information and Knowledge Management. – 1996. – P. 81-88.

100. Kimbal, R. The data warehouse toolkit [Text] / R. Kimbal. – NY. : Wiley, 1996. – 600 p.

101. Agrawal, R. Modeling multi-dimensional databases [Text] / R. Agrawal, A. Gupta, S. Sarawagi // Proc. of the 13th International Conference on Data Engineering. – 1997. – P. 232-243.

102. Cabibbo, L. Querying multi-dimensional databases [Text] / L. Cabibbo, R. Torlone // Proc. of the 6th International Conference on Database Programming Languages. – 1997. – P. 319-335.

103. Datta, A. A conceptual model and algebra for on-line analytical processing in decision support databases [Text] / A. Datta, H. Thomas // Proc. of the 7th Annual Workshop on Information Technologies and Systems. – 1997. – P. 91-100.

104. Data cube: a relational aggregation operator generalizing group-by, cross-tab and sub-totals [Text] / G. Gray, S. Chaudhuri, A. Bothworth at all. // Data Mining Knowledge Discovery. – 1997. – № 1. – P. 29-54.

105. Gyssens, M. A foundation for multi-dimensional databases [Text] / M. Gyssens, L. V. S. Lakshmanan // Proc. of the 23rd International Conference on Very Large Databases. – 1997. – P. 106-115.

106. Lehner, W. Modeling large-scale scenarios [Text] / W. Lehner // Proc. of the 6th International Conference on Extending Database Technology. – 1997. – P. 153-167.

107. Vasiliadias, P. Modeling multi-dimensional databases, cubes, and cube operations [Text] / P. Vasiliadias // Proc. of the 10th International Conference on Statistical and Scientific Database Management. – 1998. –P. 53-62.

108. Jagadish, H. V. What can hierarchies do for data warehouse? [Text] / H. V. Jagadish, L. V. S. Lakshmanan, D. Srivatstava // Proc. of the 25rd International Conference on Very Large Databases. – 1999. – P. 530-541.

109. Mendelson, A. O. Temporal queries in OLAP [Text] / A. O. Mendelson, A. A. Vaismann // Proc. of the 25rd International Conference on Very Large Databases. – 2000. – P. 242-253.

110. Редько, В. Н. Основания композиционного программирования [Текст] / В. Н. Редько // Программирование. – 1979. – № 3. – С. 3-13.

111. Нікітченко, М. С. Математична логіка та теорія алгоритмів [Текст] / М. С. Нікітченко, С. С. Шкільняк. – Київ : ВПЦ "Київський університет", 2008. – 528 с.

112. Griffits, P. G. An authorization mechanism for relational database system [Text] / P. G. Griffits, B. W. Wade // ACM Transactions on Database Systems. – 1976. – № 3. – P. 242-255.

113. Stonebraker, M. Implementation of integrity constraints and views by query modification [Text] / M. Stonebraker // Proc. of the International Conference on Management of Data. – 1975. – P. 65-78.

114. Reimer, M. Solving the phantom problem by predicative optimistic concurrency control [Text] / M. Reimer // Proc. of the 9th International Conference on Very Large Data Bases. – 1983. – P. 81-88.

115. Palermo, F. P. A data base search problem [Text] / F. P. Palermo // Proc. of the 4th Symposium on Computer and Information Science. – 1972. – P. 67-101.

116. Astrahan, M. M. Implementation of structured English query language [Text] / M. M. Astrahan, D. D. Chamberlin // Communications of ACM. – 1975. – № 10. – P. 580-588.

117.Niebuchr, K. L. Algorithm for processing query by example [Text] / K. L. Niebuchr, K. W. Scholtz, S. E. Smith // IBM Technical Disclosure Bulletin. – 1976. – № 2. – P. 736-741.

118.Wong, E. Decomposition – a strategy for query processing [Text] / E. Wong, R. Youssefi // ACM Transaction on Database Systems – 1976. – № 3. – P. 223-241.

119.Schenk, K. L. An algorithm for servicing multi-relational queries [Text] / K. L. Schenk, J. R. Pinkert // Proc. of the International Conference on Management of Data. – 1977. – P. 10-20.

120.Makinouchi, A. The optimization strategy for query evaluation in RDB/V1 [Text] / A. Makinouchi, M. Tezuka, H. Kitakami, S. Adachi // Proc. of the 7th International Conference on Very Large Databases. – 1981. – P. 518-529.

121.Jarke, M. Query optimization in database systems [Text] / M. Jarke, J. Koch // Computing Surveys. – 1984. – № 2. – P. 111-152.

122.Finlelstein, S. Common expression analysis in database application [Text] / S. Finlelstein // Proc. of the International Conference on Management of Data. – 1982. – P. 235-245.

123.Park, J. Using common sub-expressions to optimize multiple queries [Text] / J. Park, A. Segev // Proc. of the 4th International Conference on Data Engineering. – 1988. – P. 311-319.

124.Sellis, T. Multiple query optimization [Text] / T. Sellis // ACM Transactions on Database Systems. – 1988. – № 1. – P. 23-52.

125.Sellis, T. On the multi query optimization problem [Text] / T. Sellis, S. Ghosh // IEEE Transactions on Knowledge and Data Engineering. – 1990. – № 6. – P. 262-266.

126.Cosar, A. Multiple query optimization with depth-first branch-and-bound and dynamic query ordering [Text] / A. Cosar, E.-P. Lim, J. Srivastava // Proc. of the International Conference on Information and Knowledge Management. – 1993. – P. 433-438.

127.Graefe, G. Extensibility and search efficiency in the Volcano optimizer generator [Text] / G. Graefe, W. G. McKenna // Technical Report CU-CS-91-563. Univ. of Colorado at Boulder, 1991.

128.Harinarayan, V. Implementing data cubes efficiently [Text] / V. Harinarayan, A. Rajaraman, J. Ulman // Proc. of the International Conference on Management of Data. – 1996. – P. 205-216.

129.Index selection for OLAP [Text] / H. Gupta, V. Harinarayan, A. Rajaraman, J. Ulman // Proc. of the International Conference on Data Engineering. – 1997. – P. 208-219.

130.Gupta, H. Selection of views to materialize in a data warehouse [Text] / H. Gupta // Proc. of the International Conference on Database Theory. – 1997. – P. 453-470.

131.Pellenkoft, A. The complexity of transformation-based join enumeration [Text] / A. Pellenkoft, C. A. Galindo-Legaria, M. Kersten // Proc. of the 23rd International Conference on Very Large Databases. – 1997. – P. 306-315.

132.Rao, J. Reusing invariants: a new strategy for correlated queries [Text] / J. Rao, D. Reiner // Proc. of the International Conference on Management of Data. – 1998. – P. 37-48.

133.Subramanian, S. N. Cost based optimization of decision support queries using transient views [Text] / S. N. Subramanian, S. Venkataraman // Proc. of the International Conference on Management of Data. – 1998. – P. 319-330.

134.Kotidis, Y. DynoMat: a dynamic view management system for data ware-houses [Text] / Y. Kotidis, N. Roussopoulos // Proc. of the International Conference on Management of Data. – 1999. – P. 371-382.

135.Efficient and extensible algorithms for multi query optimization [Text] / P. Roy, S. Seshadri, S. Sudarshan, S. Bhobe // LNCS. – 2000. – V. 1909. – P. 249-260.

136.Smith, M. Optimizing the performance of a relational algebra performance interface [Text] / M. Smith, P. Y. W. Chang // Communications for the ACM. – 1975. – № 10. – P. 568-579.

137.Aho, A. Efficient optimization of a class of relational expressions [Text] / A. Aho, Y. Sagiv, J. Ullman // ACM Transactions on Database Systems. – 1979. – № 4. – P. 435-454.

138.Aho, A. Equivalences among relational expressions [Text] / A. Aho, Y. Sagiv, J. Ullman // Journal on Computing. – 1979. – № 2. – P. 218-246.

139.Sagiv, Y. Equivalences among relational expressions with union and difference operators [Text] / Y. Sagiv // Journal of the ACM. – 1980. – № 4. – P. 633-655.

140.Krishnamurthy, R. XML-SQL query translation literature: the state of the art and open problems [Text] / R. Krishnamurthy, R. Kaushik, J. Naughton // LNCS. – 2003. – V. 2824. – P. 1-18.

141.Frasincar, F. XAL: an algebra for XML query optimization [Text] / F. Frasincar, G.-J. Houben, C. Pau // Proc. of the 13th Australian Conference. – 2002. – Vol. 5. – P. 49-56.

142.Lukichev, M. XML query algebra for cost-based optimization [Electronic resources] / M. Lukichev, D. Barashev // Proc. of the 4th Spring Young Researches Colloquium on Databases and Information Systems. – Access mode: http://ceur-ws.org/Vol-256/. – 12.04.2014.

143.Re, C. A complete and efficient algebraic compiler for Xquery [Text] / C. Re, J. Simeon, M. F. Fernandez // LNCS. – 2007. – V. 4797. – P. 81-96.

144.Пашинин, О. В. Оптимизация запросов к базам данных [Текст] / О. В. Пашинин // Математические структуры и моделирование. – 2007. – Вып. 17. – С. 100-107.

145.Mendkovich, N. New algorithms for lexical query optimization [Text] / N. Mendkovich, S. Kuznetsov // Proc. of the 31st International Conference on Information Technology Interfaces. – 2009. – P. 187-192.

146.Кузнецов, С. Д. Новые алгоритмы лексической оптимизации запросов [Текст] / С. Д. Кузнецов

// Моделирование и анализ информационных систем. – 2009. – № 4. – С. 22-33.

147.Shenoy, S. T. A system for semantic query optimization [Text] / S. T. Shenoy, Z. M. Ozsoyogly // SIGMOD Record. – 1987. – № 3. – P. 181-195.

148.Sun, W. Semantic query optimization for tree and chain queries [Text] / W. Sun, C. Yu // IEEE Transactions on Knowledge and Data Engineering. – 1994. – № 1. – P. 136-151.

149.Geng K. Survey of XML query optimization [Text] / K. Geng, G. Dobbie, Y. Meng // Proc. of the 4th International Conference on Internet Computing for Science and Engineering. – 2009. – P. 297-300.

150.Parallel algorithms for the execution of relational database operations [Text] / D. Bitton, H. Boral, D. Dewitt, W.K. Wilkinson // ACM Transactions on Database Systems. – 1983. – № 3. – P. 324-353.

151.Wong, E. Distributing a database for parallelism [Text] / E. Wong, R.H. Katz // Proc. of the SIGMOD Annual Meeting. – 1983. – P. 23-29.

152.DeWitt, D. Parallel database systems: the future of high performance database systems [Text] / D. Dewitt, J. Gray // Communications of the ACM. – 1992. – № 6. – P. 1-26.

153.Hasan, W. Coloring away communication in parallel query optimization [Text] / W. Hasan, R. Motwani // Proc. of the 21st International Conference on Very Large Databases. – 1995. – P. 239-250.

154.Соколинский, Л. Б. Организация параллельного выполнения запросов в многопроцессорной машине баз данных с иерархической структурой [Текст] / Л. Б. Соколинский // Программирование. – 2001. – № 6. – С. 13-29.

155.Григорьев, Ю. А. Теоретические основы анализа процессов доступа к распределенным базам данных [Текст] / Ю. А. Григорьев, А. Д. Плутенко. – Новосибирск : Наука, 2002. – 222 с.

156.Соколинский, Л. Б. Обзор архитектур параллельных систем баз данных [Текст] / Л. Б. Соколинский // Программирование. – 2004. – № 6. – С. 1-15.

157.Щетинин, Д. И. Оптимизация запросов в системах баз данных на параллельных структурах [Текст] / Д. И. Щетинин // Проблеми програмування. Спеціальний випуск. – 2006. – № 2-3. – С. 102–109.

158.Григорьев, Ю. А. Оценка времени выполнения запросов и выбор архитектуры параллельной системы баз данных [Текст] / Ю. А. Григорьев, В. Л. Плужников // Организация баз данных. – 2009. – № 3. – С. 3-12.

159.Fast computation of database operations using graphic processors [Text] / N. Govindaraju, B. Lloyd, W. Wang, at all // Proc. of International Conference on Management of Data. – 2004. – P. 215-226.

160.GPUTeraSort: high performance graphics coprocessor sorting for large database management [Text] / N. Govindaraju, J. Gray, R. Kumar, at all // Proc. of International Conference on Management of Data. – 2006. – P. 325-336.

161.Mars: A MapReduce framework on graphics processors [Text] / B. He, W. Feng, Q. Luo, at all // Proc. of the 17th International Conference on Parallel Architectures and Compilation Techniques. – 2008. – P. 260-269.

162.Relational query co-processing on graphic processors [Text] / B. He, M. Lu, K. Yang, at al // ACM Transactions on Database Systems. – 2009. – № 4. – P. 1-39.

163.Baccum, P. Accelerating SQL database operations on GPU with CUDA [Text] / P. Baccum, K. Skadron // Proc. of Workshop on General-Purpose Computation on Graphics Processing Units. – 2010. – P. 94-103.

164.Лыфарь, Д. А. Параллельные алгоритмы обработки реляционных баз данных [Текст] / Д.А. Лыфарь // Вестник НГУ. Серия: Информационные технологии. – 2010. – Вып. 4. – С. 72-80.

165.Zheng, Y. A quantitative performance analysis model for GPU architectures [Text] / Y. Zheng, J.D. Owens // Proc. of the 17th International Symposium on High-Performance Computer Architecture. – 2011. – P. 382-393.

166.Beyer, M. A. The importance of «Big Data»: a definition [Electronic resources] / M. A. Beyer, D. Laney. – Access mode: http://www.gartner.com/DispleyDocument?id=2057415. – 12.04.2014.

167.White, T. Hadoop: the definitive guide [Text] / T. White. – NY. : O'Reilly Media, 2010. – 624 p.

168.Strozzi, C. NoSQL: a relational database management system [Electronic resources] / C. Strozzi. – Access mode: http://www.strozzi.it/cgi-bin/CSA/tw7/I/en. – 12.04.2014.

169.Pokorny, J. NoSQL databases:a step to database scalability in web environment [Text] / J. Pokorny // Proc. of the 13th International Conference on Information Integration and Web-Based Applications and Services. – 2011. – P. 278-283.

170.Strauch, C. NoSQL databases [Electronic resources] / C. Strauch. – Access mode: http://www.christof-strauch/de/nosqldbs.pdf. – 12.04.2014.

171.Cattel, R. Scalable SQL and NoSQL data stores [Text] / R. Cattel // ACM SIGMOD Record. – 2010. – № 4. – P. 12-27.

172.Bigtable: a distributed storage system for structured data [Text] / F. Chang, J. Dean, S. Ghemawat, at all // Proc. of the 7th Symposium on Operating Systems Design and Implementation – 2006. – P. 205-218.

173.Daniel, J. A. Integrating compression and execution in column-oriented database systems [Electronic resources] / J. A. Daniel, R. M. Samuel, C. F. Miguel.
– Access mode: http://db.lcs.mit.edu/projects/cstore/abadisigmod06.pdf. – 12.04.2014.

174.C-Store: a column-oriented DBMS [Electronic resources] / M. Stonebraker, D. Abadi, A Batkin, at all. – Access mode: http://www.cs.yale.edu/homes/dna/pubs/displeypubs.cgi. – 12.04.2014.

175.Daniel, J. A. Query execution in column-

oriented database systems [Electronic resources] / J. A. Daniel. – Access mode: http://www.cs.yale.edu/homes/dna/papers/abadiphd.pdf. – 12.04.2014.

176.Григорьев, Ю. А. Оценка времени соединения таблиц в параллельной системе баз данных [Текст] / Ю. А. Григорьев, В. Л. Плужников // Информатика и системы управления. – 2011. – № 1. – С. 3-16.

177.Григорьев, Ю. А. Модель обработки запросов в параллельной колоночной системе баз данных [Текст] / Ю. А. Григорьев, Е. Ю. Ермаков // Организация баз данных. – 2012. – № 1. – С. 3-15.

178.Клеменков, П. А. Большие данные: современные подходы к хранению и обработке [Текст] / П. А. Клеменков, С. Д. Кузнецов // Труды института системного программирования РАН. – 2012. – Т. 23. – С. 143-158.

179.Лаврищева, Е. М. Методы программирования: теория, инженерия, практика. [Текст] / Е. М. Лаврищева. – Киев : Наукова думка, 2006. – 451 с.

180.Лаврищева, Е. М. Интерфейс в программировании [Текст] / Е.М. Лаврищева // Проблеми програмування. – 2007. – № 2. – С. 126–139.

181.Лаврищева, Е. М. Сборочное программирование. Основы индустрии программных продуктов [Текст] / Е. М. Лаврищева. – Киев : Наукова думка, 2009. – 371 с.

Поступила в редакцию 2.04.2014, рассмотрена на редколлегии 19.05.2014

Рецензент: д-р техн. наук, проф., заведующий кафедрой компьютерных систем и сетей В. С. Харченко, Национальный аэрокосмический университет имени Н.Е. Жуковского «ХАИ», Харьков.

МОДЕЛІ, МЕТОДИ І АЛГОРИТМИ ОПТИМІЗАЦІЇ ЗАПИТІВ У БАЗАХ ДАНИХ (ОГЛЯД)

Д. Б. Буй, В. Г. Скобелєв

В цій роботі зроблено спробу дослідити сучасний стан проблеми забезпечення ефективності функціонування СУБД з позиції моделей та методів, призначених для оптимізації запитів. На основі аналізу особливостей процесу проектування СУБД, основних моделей даних, основних понять, що використовуються при оцінці складності алгоритмів, показано, що в цьому контексті треба розглядати саме зв'язку «БД–СУБД». Охарактеризовано основні два типа математичних моделей, призначених для теоретичного аналізу ефективності функціонування зв'язки «БД–СУБД» в термінах аналізу складності операцій. По-перше, це моделі, що конструюються в термінах формальної теорії, побудованої на основі синтезу математичної логіки, теорії моделей та прикладної теорії алгоритмів. По-друге, це моделі багаторозмірних зв'язок «БД–СУБД», що будуються в термінах теорії категорій (це, зокрема, дає можливість ефективно використовувати мову схем). Розглянуто еволюцію методів «оптимізації запитів» (які є евристичними методами) в контексті розвитку моделей зв'язок «БД–СУБД».

Ключові слова: бази даних, СУБД, оптимізація запитів.

MODELS, METHODS AND ALGORITHMS FOR QUERY OPTIMIZATION IN DATABASES (A SURVEY)

D. B. Bui, V. G. Skobelev

Given paper consists some attempt to investigate the state of the art for the problem of ensuring the efficiency of BDMS from a position of models and methods intended for query optimization. On the base of analysis of features of BDMS design process, basic data models and basic notions applied for complexity analysis of algorithms it is shown that it is worth to consider exactly a bunch «DB–DBMS» in considered context. Two basic types of mathematical models intended for theoretic analysis of efficiency of performance of any bunch «DB–DBMS» in terms of analysis of complexity of operations are characterized. Firstly, these are models designed in terms of a formal theory, based on symbiosis of mathematical logic, models theory and algorithms theory. Secondly, these are models intended for many-dimensional bunches «DB–DBMS» and designed in terms of category theory (that, in particular, makes it possible to effectively use the language of schemes). In context of evolution of models of bunches «DB–DBMS» it is considered evolution of methods intended for queries optimization.

Keywords: database systems, DBMS, query optimization.

Буй Дмитрий Борисович – д-р физ.-мат. наук, профессор, профессор кафедры теории и технологии программирования факультета кибернетики, Киевский национальный университет имени Тараса Шевченко, Киев, Украина, e-mail: buy@unicyb.kiev.ua.

Скобелев Владимир Геннадиевич – д-р физ.-мат. наук, д-р техн. наук, профессор, ведущий научный сотрудник, Институт прикладной математики и механики НАН Украины, Донецк, Украина, e-mail: skbv@iamm.ac.donetsk.ua.