Московский государственный университет
имени М.В. Ломоносова

На правах рукописи

Кудрявцев Юрий Александрович

Алгоритмы эффективной обработки MOLAP-кубов

05.13.11 – математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Автореферат
диссертации на соискание учёной степени
кандидата физико-математических наук

Москва — 2009

Работа выполнена на кафедре системного программирования факультета вычислительной математики и кибернетики Московского государственного университета имени М.В. Ломоносова.

Научный руководитель:	доктор физико-математических наук, профессор Марков А. С.
Официальные оппоненты:	доктор физико-математических наук, профессор Машечкин И.В.
	доктор физико-математических наук, профессор Кузнецов С.О.
Ведущая организация:	Кафедра системного программирования – Южно-Уральский государственный университет

Защита состоится 18 декабря 2009 в 11 ч. 00 мин. на заседании диссертационного совета Д 501.001.44 Московского государственного университета имени М.В. Ломоносова по адресу: 119991, ГСП-1, Москва, Ленинские горы, МГУ, 2-й учебный корпус, факультет вычислительной математики и кибернетики, ауд. 685.

С диссертацией можно ознакомиться в библиотеке факультета ВМиК МГУ.

С текстом автореферата можно ознакомиться на официальном сайте ВМиК МГУ http://cs.msu.ru в разделе "Наука"–"Работа диссертационных советов" – "Д 501.001.44"

Автореферат разослан "18" ноября 2009.

Учёный секретарь
диссертационного совета
профессор Н.П. Трифонов

Общая характеристика работы.

Актуальность темы.

Термин OLAP (Online Analytical Processing) был введен в 1993 Эдгаром Коддом ([Cod93]). Цель OLAP систем – предоставление пользователю возможности анализа больших объемов данных в интуитивно понятной форме. Кодд сформулировал 12 признаков OLAP–данных, и большинство современных OLAP средств отвечают этим постулатам. Однако 12 признаков в дальнейшем трансформировались в 4 ключевых определения, сформулированные Найджелом Пендзом (см. [Pen05]), на которые теперь ссылаются при определении OLAP-систем.

FASMI-тест. OLAP-система должна быть:

- Fast – быстрой, обеспечивать почти мгновенный отклик на большинство запросов
- Shared – многопользовательской, должен существовать механизм контроля доступа к данным, а также возможность одновременной работы многих пользователей
- Multidimensional – многомерной, данные должны представляться в виде многомерных кубов.
- Information – данные должны быть полны с точки зрения аналитика, т.е. содержать всю необходимую информацию.

Окончательную формулировку термина предложила в 1995 группа исследователей во главе с Джимом Греем ([GBLP95]), проанализировав создававшиеся тогда пользовательские приложения баз данных, и предложив расширение языка SQL – оператор CUBE. Этот оператор отвечает в SQL за создание многомерных кубов. Концепция многомерного представления данных является, наряду с моделью транзакций, одной из самых известных идей Кодда. В этой работе исследователи указали ряд эвристических рекомендаций по реализации новой структуры данных.

CUBE представляет собой обобщение операторов GROUP BY по всем возможным комбинациям измерений с разными уровнями агрегации дан-

ных. Каждая сгруппированная таблица относится к группе ячеек, описываемых кортежами из измерений, по которым формируется куб. Оператор, расширяющий SQL, называется CUBE BY (синтаксис такой же, как и у GROUP BY).

Дальнейшее развитие OLAP-операций в SQL привело к тому, что в стандарт SQL'99 был включен набор операторов для работы с OLAP-данными (запросы grouping set, rollup by, cube by, window by, rank, rownum и пр.).

В реализации OLAP-приложений возникает ряд проблем, прежде всего связанных с экспоненциальным увеличением объема данных, которые необходимо хранить или рассчитывать.

OLAP-приложения делятся прежде всего по методу хранения данных на:

- ROLAP, Relational OLAP – все данные куба (фактическая таблица + агрегаты) хранятся в реляционной таблице
- MOLAP, Multidimensional OLAP – и фактические данные, и агрегаты хранятся в многомерной бд
- HOLAP, Hybrid OLAP – фактические данные хранятся в реляционной таблице, агрегаты – в виде многомерных структур (многомерной БД)

Дальнейшее разделение происходит по принципу частичной или полной материализации агрегатов, т.о. OLAP-системой с полной материализаций агрегатов называется система, в которой все возможные агрегирующие представлениея посчитаны на этапе вычисления куба. В системе с частичной материализацией выбор агрегирующих представлений для расчета определяется на основе какого-либо алгоритма для достижения максимальной скорости ответов на запросы пользователей.

В 1994 Майкл Стоунбрейкер и Сунита Сараваги ([GBLP95]) предложили модель оценки выбора агрегированных представлений с учетом стоимости материализации и изменения скорости работы запросов. Пять лет спустя, Карлофф и Михаил ([GBLP95]) показали сводимость этой задачи к

NP-трудной задаче упаковки, доказав тем самомым отсутствие оптимальных алгоритмов решения этой задачи.

Исходя из вышеуказанного, данная работа посвящена разработке оптимальных алгоритмов постреления MOLAP-систем с полной материализацией.

Цель работы. Целью диссертационной работы является развитие новых алгоритмов расчета MOLAP-кубов для систем с полной материализацией, улучшение существующих результатов и оценка применимости параллельных вычислений к задаче расчета OLAP-кубов.

В соответствии с целью исследования были поставлены следующие задачи:

1. Выявить и описать методы построения OLAP-кубов не вошедшие в современные научные обзоры других исследователей, классифицировать методы;

2. Создать математическую модель, на базе которой оценить оптимальность методов

3. Использовать полученные математические результаты для создания более оптимального метода обработки OLAP-данных

4. Оценить применимость современной парадигмы параллельных вычислений над большими объемами данных Map/Reduce для задачи создания OLAP-кубов

Методы исследования. В работе используются методы теории груп и решеток, теория реляционных баз данных и теория параллельных вычислений.

Научная новизна работы. Следующие особенности работы определяют её научную новизну:

1. проведен анализ существующих алгоритмов обработки MOLAP-кубов, выделены следующие группы алгоритмов - синтаксические, семантические и аппроксимирующие. Для каждой из групп приведены примеры наилучших, на момент написания работы, алгоритмов

и указаны условия, в рамках которых применение этих алгоритмов является оптимальным (см главу 3). На основе результатов анализа разработана формальная математическая модель OLAP данных на базе теории решеток. Для предложенной модели с учётом классов эквивалентности доказана оптимальность представления OLAP-кубов замкнутыми решетками по введенному отношению покрытия, показана эквивалентность представления кубов замкнутыми решетками и разбиения решетки куба на классы эквивалентности по отношению покрытия, предложенному в алгоритме Quotient Cube. При этом доказана оптимальность алгоритма Quotient Cube (см главу 4).

2. для задачи создания замкнутых решеток OLAP-кубов или же Quotient Cube-решеток предложен алгоритм, использующий парадигму Map/Reduce. Предложенный алгоритм, в отличие от уже существующих линейных алгоритмов, может эффективно использовать многомашинные кластеры и многоядерные серверы, что позволяет существенно увеличить объем обрабатываемых данных и уменьшить время расчетов, см главу 5.

3. создан прототип алгоритма на технологиях Apache Hadoop (многомашинный кластер, см [KMSB08]) и Standord Phoenix (использование map/reduce для многопроцессорных серверов, см [RRP+07]), проведены эксперименты, показывающие преимущества данного алгоритма по отношению к уже существующим, см главу 5.

Теоретическая и практическая ценность. Разработана математическая модель представления OLAP-кубов, использующая формализм теории решеток. На базе этой модели доказана оптимальность представления куба замкнутыми решетками. Приведен новый параллельных алгоритм вычисления замкнутых кубов на базе парадигмы Map/Reduce вычислений. Полученный алгоритм апробирован на экспериментальных данных, показана эффективность метода.

Апробация работы. Результаты диссертации докладывались и обсуждались на следующих конференциях и семинарах:

1. Трижды на семинаре московской секции ACM SIGMOD (2005, 2007, 2008)
2. Международной конференции 'Бизнес-Информатика', Звенигород, 2007
3. Дважды на конференции 'Корпоративные Базы Данных', Москва, 2007, 2009
4. Конференции 'Бизнес-Аналитика на современном предприятии', Москва, 2008
5. Конференции 'Advances in Databases, Knowledge, and Data Applications', Канкун, Мексика, 2009
6. Конференции Syrcodis, Москва, 2006
7. На семинаре 'Проблемы современных информационно-вычислительных систем' под управлением Васенина В.А. 2008
8. Неоднократно на семинаре 'Технологии баз данных' под управлением Кузнецова С.Д. и Маркова А.С., Москва, 2004-2008
9. На семинаре 'Математические модели информационных технологий' под управлением Кузнецова С.О., ГУ-ВШЭ, 2008

Публикации. Содержание диссертации изложено в работах [1 – 5].

Объем и структура работы. Диссертация содержит 142 страницы текста, состоит из введения, четырех глав, библиографии.

Содержание работы.

Во **введении** обоснована актуальность исследуемой проблемы, сформулирована цель и задачи диссертационной работы, введены необходимые определения, перечислены полученные в диссертации новые результаты и описана структура диссертации.

Рассматривается следующая постановка задачи.

Для базовой (фактической) таблицы r, на основе которой будет строиться OLAP-куб, множество атрибутов r разделяем на 2 группы:

1. Набор измерений (категорий, локаторов), которые служат критериями для анализа и определяют многомерное пространство OlAP-куба. За счет фиксации значений измерений получаются срезы (гиперплоскости) куба. Каждый срез представляет собой запрос к данным, включающий агрегации.

2. Набор мер – функции, которые каждой точке пространства ставят в соответствие данные.

Из атрибутов r создаются измерения, содержащие проекцию r по атрибуту, с введенной иерархией (например, для таблицы, в которой хранятся фактические данные по продажам магазина, возможно наличие измерение под названием "Время содержащего иерархию вида "Год-Месяц-Неделя-День"). Куб представляет собой декартово произведение измерений, где для каждого элемента произведения назначен набор мер. В кубе введены отношения обобщения и специализации (roll-up/drill-down) по иерархиям измерений. Ячейка высокого уровня иерархии может "спускаться"(drill-down) к ячейке низкого уровня (для примера (R1,ALL,весна) может "спуститься"к ячейке (R1,книги,весна)) и наоборот, "подняться"(roll-up) (от (R1,книги,весна) к (R1,ALL,весна) по измерению "продукты").

Рассмотрим пример, на который будем ссылаться в дальнейшем.

Таблица 1. Фактические данные для примера

Регион	Продукт	Время года	Продажи
R1	книги	Весна	9
R1	Еда	Осень	3
R2	книги	Осень	6

Размер куба данных определяется по формуле $\prod_{i=1}^{d}(c_i + 1)$, где d - количество измерений ("столбцов"), c_i - размерность измерения, т.е. количество различных значений кортежей по этому измерению. $+1$ отвечает за "значение"ALL, агрегирующее все возможные значения измерения.

В **главе 2** дано более подробное описание задачи. Анализ различных алгоритмов построения MOLAP-кубов приведен **в третьей главе**. Подробно рассмотрен семантический алгоритм Quotient Cube, оптимальность представления которого доказывается в **главе 4**.

Таблица 2. Куб для таблицы 1. Агрегирующая функция - AVG.

Регион	Продукт	Время года	AVG(Продажи)
R1	книги	Весна	9
R1	Еда	Осень	3
R2	книги	Осень	6
R1	книги	ALL	9
R1	ALL	Весна	9
ALL	книги	Весна	9
...	...	...	...
R2	ALL	ALL	6
ALL	Еда	ALL	3
ALL	ALL	Весна	9
ALL	ALL	ALL	6

Для доказательства оптимальности Quotient-решетки **в четвертой главе** сформулировано описание OLAP-куба в терминах теории решеток. При этом вводятся следующие определения:

Определение 1. Многомерное пространство *Space(r)* $= \{\times_{A \in D}(r[A] \cup ALL) \cup \{0, 0, \ldots, 0\}\}$, *где* $\times$ - *декартово произведение,* $\{0, 0, \ldots, 0\}$ - *нулевой кортеж.*

$\forall s \in Space(r), s$ – многомерный кортеж. Для куба из примера (см. таблицу 1) получаем следующее многомерное пространство:

{(R1,книги,весна), (R1,еда,осень), (R2,книги,осень),(R1,еда,весна),. . . , (ALL,еда,осень), (R1,ALL,весна),. . . ,
(ALL,ALL,ALL)} – всего 28 кортежей.

Введем отношение обобщения/специализации на $Space(r)$ обозначаемое $\geq_g$.

Определение 2. Отношение порядка $\geq_g$ $u, v \in Space(r)$

$$u \geq_g v = \begin{cases} \forall A \in D, v[A] \subseteq u[A] \\ \textit{в противном случае } v = \{0, 0, \ldots, 0\} \end{cases}$$

Если $u, v \in Space(r), u \geq_g v$ тогда u обобщает v в $Space(r)$.

Для куба из примера (см. таблицу 1): (ALL,еда,осень)$\geq_g$(R1,еда,осень), (ALL,ALL,ALL)$\geq_g$(ALL,еда,осень).

Операторы, создающие новые кортежи: сумма (+) и произведение ($\bullet$).

Определение 3. *Сумма двух кортежей – наименьший кортеж, обобщающий оба операнда.*

$u, v \in Space(r)$

$$t = u + v \Leftrightarrow \forall A \in D, t[A] = \begin{cases} u[A], \textit{если } u[A] = v[A] \\ ALL \text{ , } \textit{в противном случае} \end{cases}$$

Для примера из таблицы 1, (R2,еда,осень) + (R1,ALL,осень) = (ALL,ALL,осень)

Определение 4. *Произведение двух кортежей – наибольший кортеж, уточняющий оба операнда.*

Пусть $\forall A \in D,\ z[A] = u[A] \wedge v[A]$. *Тогда* $u, v \in Space(r)$

$$t = u \bullet v \Leftrightarrow \begin{cases} t = z \textit{ если } \neg\exists A \in D | z[A] = \{0\} \\ \{0, 0, \ldots, 0\} \text{ , } \textit{в противном случае} \end{cases}$$

Например, (ALL,еда,осень) • (R1,ALL,осень) = (R1,еда,осень)

Введя определения $Space(r)$, $\geq_g$ и операторы + и • мы можем дать определение *решетки куба.*

Определение 5. Решетка куба

Пусть r *– отношение базы данных над* $D \cup M$. *Чум* $CL(r) = \{Space(r), \geq_g\}$ *– решетка, в которой пересечение и объединение вводятся следующим образом:*

$$\forall T \subseteq CL(r),\ \wedge T = +_{t \subseteq T} t$$

$$\forall T \subseteq CL(r),\ \vee T = \bullet_{t \subseteq T} t$$

$\{CL(r), \geq_g, \wedge, \vee\}$ *– решетка куба.*

В заданных терминах, решетка для алгоритма Quotient Cubes (см описание в главе 3) определяется следующим образом:

Определение 6. Quotient решетка куба

Пусть CL(r) – решетка куба(см. определение 5), и $\equiv_{cov}$ *– отношение эквивалентности. Quotient решетка куба* $QCLR(r) = (CL(r)/\ \equiv_{cov}, \preceq)$. *Для двух классов эквивалентности* $A, B \in QCLR(r), A \succeq B$ *если* $\exists a \in A, \exists b \in B | a \geq_g b$.

Минимальной решеткой, с точки зрения классов эквивалентности является замкнутая решетка. В **четвертой главе** приведено доказательство эквивалентности Quotient решетки и замкнутой решетки. Таким образом, Quotient Cubes алгоритм является оптимальным по количеству хранимых замкнутых классов алгоритмом хранения OLAP-кубов.

Следующей задачей является разработка эффективного алгоритма расчета Quotient кубов на кластере много процессорных серверов.

Алгоритм создания замкнутого MOLAP-куба

С учетом стремительного роста объема данных, которые необходимо обрабатывать, распараллеливание программ остается единственно возможным вариантом реализации вычислительно сложных задач, например создания графа web-страниц и ранжирования их. Учитывая этот факт, группа инженеров в Google, создала новую библиотеку Map-Reduce (см [DG06] и [Lam08]), которая сейчас широко используется для новых приложений, создаваемых в компании, вместо старых подходов к написанию параллельных программ на "чистом"C++.

Map/Reduce парадигма базируется на представлении необходимых вычислений в виде последовательности 2х функций: map и reduce. При этом функция map применяется ко всем входным данным и каждому входному значению сопоставляет пару (ключ, f_{map}(элемент исходных данных)). Функция reduce применяется к множеству пар содержащим одинаковый ключ и возвращает пару (ключ, f_{reduce}(множество результатов f_{map} с таким же ключом)). Подобные операции были введены в языке LISP, где map применялся ко всем элементам списка, а reduce (или в некоторых других функциональных языках, fold) "сворачивает"список, применяя агрегирующую функцию над всеми элементами.

В **пятой главе** приведено подробное описание Map/Reduce подхода и кратко описаны основные реализации.

Основной задачей алгоритма создания Quotient куба является вычисление замыкания (или покрытия) каждой ячейки куба и вычисление таким образом классов эквивалнентности. Мы используем Map-Reduce функции для обхода решетки куба поочередно сверху-вниз и снизу-вверх, что позволяет рано обнаруживать некоторые типы замкнутых классов и избегать

таким образом ненужных вычислений. Промежуточные результаты (текущие нижние грани) мы храним в распределенном кэше.

Описание в псевдокоде приведено в 1.

Псевдокод № 1. Вычисление замкнутого куба

```
1 compute_closed_cube(data d, distributed cache dc, number of dimensions D):
2 for i in [1..D/2]:
3 { if map(d, D–i, dc) <> 0 {reduce(d, D–i, dc)}
4   else cube is computed
5   if map(d, i, dc) <> 0 {reduce(d, i, dc)}
6   else cube is computed}
7   closed classes from dc are transformed into QC–Tree for query processing.
```

Если в результате map фазы не генерируются новые ячейки – значит все замкнутые классы уже вычислены и вычисления можно завершить.

Шаги алгоритма:

1. 1ый шаг вниз, все верхние грани с $supp = 1$ записываются как паттерны замкнутых классов, т.е. если $(a_1, *, *)$ имеет $supp = 1$, то записывается класс (a_1, b_1, c_1), $(a_1, *, *)$ и ячейки, содержащие a_1 в дальнейших шагах не порождаются

2. 1ый шаг вверх, все верхние грани с $supp >= 2$ записываются в промежуточное хранилище

3. 2ой шаг вниз, ищутся грани с $supp >= 2$, совпадающие с записанными на шаге 2, такие классы с только что порожденной верхней гранью записываем (и они снова будут использоваться как паттерн, который порождать не надо), все промежуточные классы, полученные на шаге 2, удаляем

4. 2ой шаг вверх, аналогично 2, только $supp >= 2$. Дублирующихся верхних граней ячеек с supp=2 порождено не будет, они попадут в паттерны, полученные на шаге 3

5. 3ий шаг вниз, аналогично 3 и т.д.

Листинги map и reduce функций в псевдокоде представлены в 2 и 3.

Псевдокод № 2. Map функция

```
1  map(data d, level L, distributed cache dc):
2  for each cell c in d
3  {
4  c_aggr = generate_aggregate_cells(c, L)
5  for each aggr_cell in c_aggr:
6    {if for each ubound in dc[classes_upper_bounds]
7      (aggrc * ubound == {0,0,...,0}) then output (aggrc,1)}
8  }
```

Функция generate_aggregate_cells генерирует все возможные агрегаты заданного уровня для указанной ячейки. Например, для первого уровня, она порождает набор ячеек, в которых одна из координат заменена значением ALL.

Шаги 6-7 map функции гарантируют что ячейки, входящие в уже вычисленные замкнутые классы не будут вычисляться, уменьшая таким образом объем вычислений и результатов map фазы.

Псевдокод № 3. Reduce функция

```
1  reduce(map_output, level L, distributed cache dc):
2  for each  (cell,support) in map_output
3  {resulting_support = sum of map emited supports
4  if (resulting_support = (D - L + 1))
5  {if exists lbound,lbound_support in dc[closed_classes_lowerbounds]
6    (cell >= lbound) and (lbound_support = resulting_support)
7    {dc[closed_classes] = (cell,ubound)}
8    else
9    {dc[closed_classes_upperbounds]+= (cell,resulting_support)}
10 }
11 else if resulting_support > 1
12 {if exists ubound,ubound_support in dc[closed_classes_upperbounds]
13      (ubound >= cell) and (ubound_support = resulting_support)
14      {dc[closed_classes] = (cell,ubound)}
15   else
16      {dc[closed_classes_lowerbounds]+= (cell,resulting_support)}}
17 }
```

Во время обхода ячеек решетки мы записываем потенциальные нижние грани и доказанные верхние грани, так что мы определяем замкнутый класс либо по верхней грани (если идем сверху-вниз, шаги 5-9) или по нижней грани (шаги 12-14).

Рассмотрим работу алгоритма на примере для данных таблицы 1.

Шаги алгоритма:

1. Первый шаг сверху-вниз порождает ячейки 3его уровня (с двумя координатами, равными All). При этом все верхние грани с $support = 1$ записываются в dc[closed_classes_upperbounds]. Таким образом записываются (R2,All, All), (all,еда,all) и (all,all,весна), т.к. их support равен 1.(all,all,осень),(all,книги,all) и (R1,all,all) записываются как потенциальные нижние грани, вместе с их значением support.

2. Первый шаг снизу вверх, порождающий агрегаты первого уровня. Верхние грани, порожденные на предыдущем шаге ассоциируются с фактическими ячейками и, поскольку агрегатов с support = 2 не порождается, нижние грани, порожденные на предыдущем шаге формируют замкнутые классы. Вычисление замкнутого куба завершено.

Описанный алгоритм реализован на Apache Hadoop и запускаемых в виде Map задач на серверах копий M/R библиотеки Stanford Phoenix для создания локальных кубов.

В этой конфигурации общий алгоритм построения куба состоит из следующих фаз (см рисунок 1):

1. Распределение данных на Hadoop File System

2. Запуск локальных Phoenix в Hadoop Streaming как map задачи

3. Получение наборов классов эквивалентности Quotient Cube

4. Формирование локального QC-Tree на каждом Hadoop-сервере

Обработка запросов состоит из (см схему 2):

1. Запуска запроса в виде map задачи на Hadoop – обращение к каждому локальному Phoenix кубу

2. Агрегации полученных результатов в reduce фазе

Основные результаты работы:

- проведена классификация существующих алгоритмов, выделены следующие группы алгоритмов *синтаксические, семантические и аппроксимирующие*, для каждой группы алгоритмов приведены примеры наилучших на момент написания работы алгоритмов, указаны условия применимости и оптимальные задачи для каждого из алгоритмов

- разработана формальная математическая модель OLAP данных на базе теории решеток, на введенной модели доказана оптимальность (с точки зрения классов эквивалентности) представления OLAP-кубов замкнутыми решетками по введенному отношению покрытия

- для задачи создания замкнутых решеток OLAP-кубов предложен алгоритм, использующий парадигму Map/Reduce.

- создан прототип алгоритма на технологиях Apache Hadoop (многомашинный кластер) и Stanford Phoenix (использование map/reduce для многопроцессорных серверов), проведены эксперименты, показывающие преимущества данного алгоритма по отношению к уже существующим.

Рис. 1. Схема обработки кубов

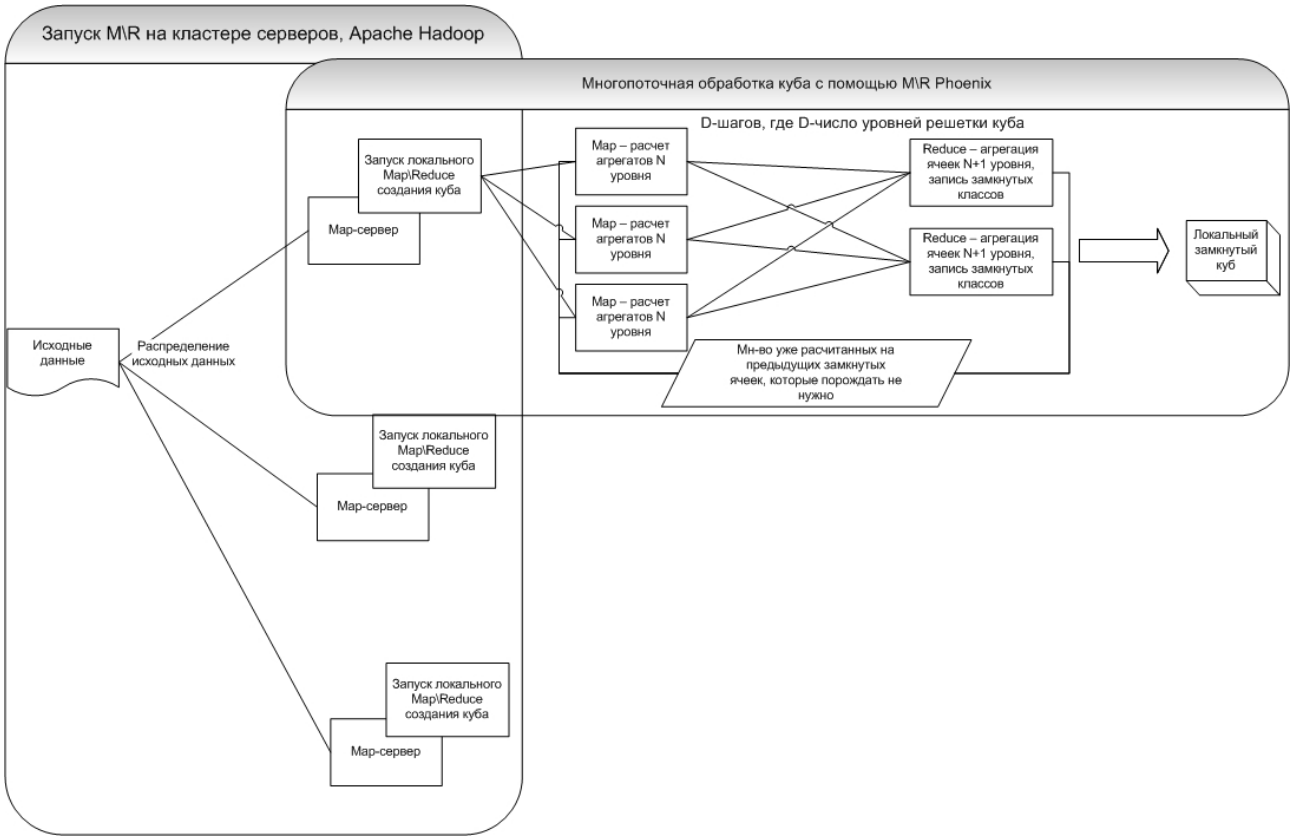
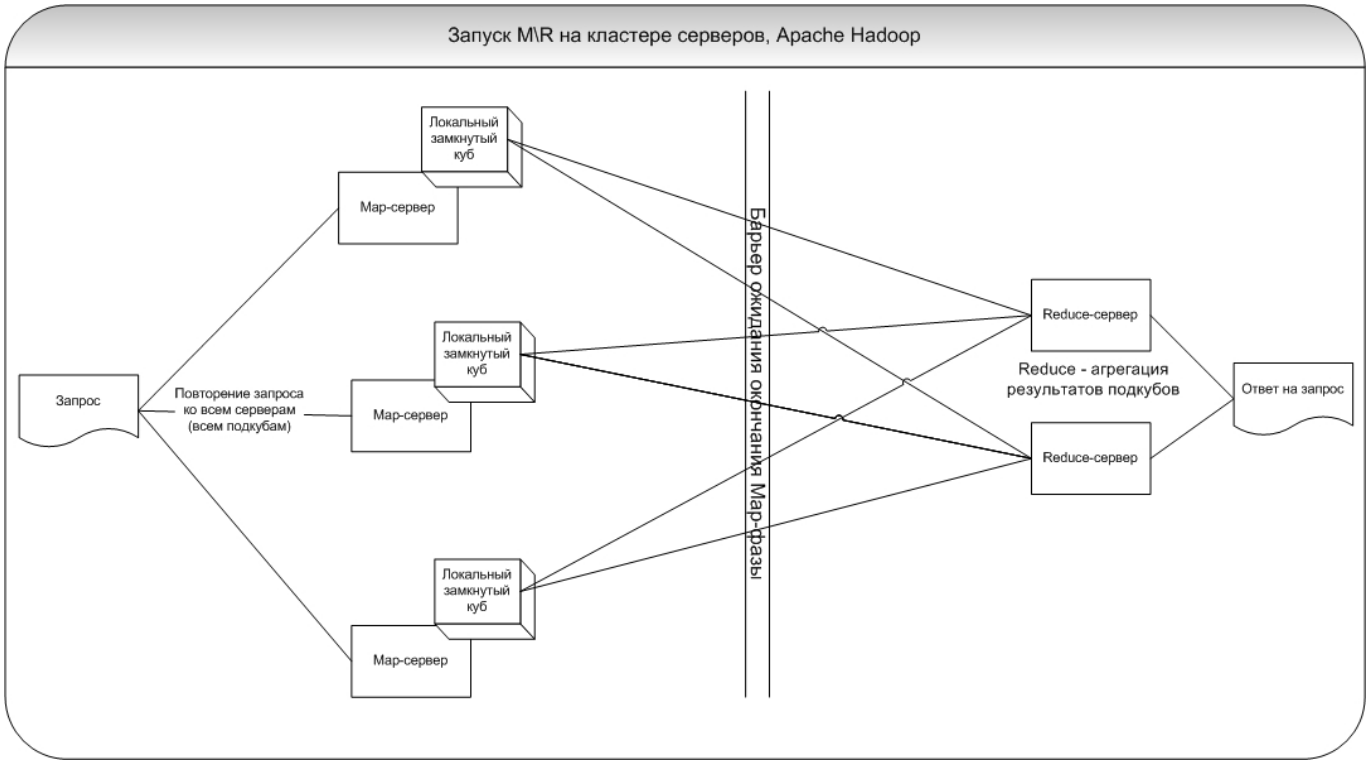


Рис. 2. Схема обработки запросов к кубам



Список публикаций по теме диссертации

1. 'Applying Map-Reduce Paradigm for Parallel Closed Cube Computation', Kuznetsov Sergei, Kudryavcev Yury, Advances in Databases, Knowledge, and Data Applications, DBKDA '09, pages 62 - 67

2. 'Математическая модель OLAP-кубов', Кузнецов С.Д., Кудрявцев Ю.А.,'Программирование', №5 2009

3. 'OLAP-технологии: обзор решаемых задач и исследований', Ю.Кудрявцев, Бизнес-Информатика, апрель 2008, Междисциплинарный научно-практический журнал, Государственный Университет — Высшая Школа Экономики, страницы 66-79, апрель 2008

4. 'Efficient algorithms for MOLAP data storage and query processing', Yuri Kudryavcev, Syrcodis, 2006 Сборнике тезисов конференции Syrcodis 2006

5. Сборнике работ молодых ученых факультета ВМиК МГУ 2005 (работа награждена дипломом второй степени)