Бондаренко Александр Викторович, Бессарабов Никита Анатольевич

МЕТОД ПРЕОБРАЗОВАНИЯ РЕЛЯЦИОННЫХ ДАННЫХ В МНОГОМЕРНЫЕ КУБЫ И ПОСТРОЕНИЯOLAP-ОТЧЕТОВ

Статья посвящена методу минимизации среднего времени обработки при проведении ROLAP-анализа за счет автоматизации этапа подготовки данных из реляционной базы. Эффективность метода проверена на случайно сгенерированной базе данных с различным числом записей, полей и шкал измерений. Показана минимизация суммарного времени от запроса к исходной базе данных до окончания вычисления куба в итеративном процессе OLAP-анализа многомерного куба с варьированием состава измерений.

Адрес статьи: www.gramota.net/materials/1/2012/6/2.html

Статья опубликована в авторской редакции и отражает точку зрения автора(ов) по рассматриваемому вопросу.

Источник

Альманах современной науки и образования

Тамбов: Грамота, 2012. № 6 (61). С. 13-17. ISSN 1993-5552.

Адрес журнала: www.gramota.net/editions/1.html

Содержание данного номера журнала: www.gramota.net/materials/1/2012/6/

© Издательство "Грамота"

Информация о возможности публикации статей в журнале размещена на Интернет сайте издательства: www.gramota.net

Вопросы, связанные с публикациями научных материалов, редакция просит направлять на адрес: almanac@gramota.net

Когнитивный стиль отражает индивидуальные особенности познавательных процессов, показывает ментальные действия, аффективно, эмоционально представленные в сознании.

Список литературы

1. **Холодная М. А.** Когнитивные стили о природе индивидуального ума. М., 2002.
2. **Gardner R. W.** Individual Differences in Figural After-Effects and Response to Reversible Figures // British Journal of Psychology. 1961. Vol. 52. Issue 3. P. 269-272.
3. **Kagan J.** Reflection-Impulsivity: the Generality and Dynamics of Conceptual Tempo // J. of Abnormal Psychology. 1966. Vol. 71.

УДК 004.08
Технические науки

Статья посвящена методу минимизации среднего времени обработки при проведении ROLAP-анализа за счет автоматизации этапа подготовки данных из реляционной базы. Эффективность метода проверена на случайно сгенерированной базе данных с различным числом записей, полей и шкал измерений. Показана минимизация суммарного времени от запроса к исходной базе данных до окончания вычисления куба в итеративном процессе OLAP-анализа многомерного куба с варьированием состава измерений.

Ключевые слова и фразы: *OLAP*-анализ; многомерный *OLAP*-куб; реляционная база данных; структура «звезда»; структура «снежинка»; *SQL*-запрос; первичный ключ базы данных; внешний ключ базы данных.

Александр Викторович Бондаренко
Никита Анатольевич Бессарабов
Кафедра управляющих и информационных систем
Московский физико-технический институт (государственный университет)
nikitabes@mail.ru

МЕТОД ПРЕОБРАЗОВАНИЯ РЕЛЯЦИОННЫХ ДАННЫХ В МНОГОМЕРНЫЕ КУБЫ И ПОСТРОЕНИЯ *OLAP*-ОТЧЕТОВ©

1. Постановка задачи

OLAP-анализ является одним из наиболее быстро развивающихся направлений информатики [1-18]. В литературе наиболее часто имеет место следующая классификация *OLAP*-систем: *ROLAP* - реляционные *OLAP*-системы, для которых структура куба данных хранится в реляционной базе данных; *MOLAP* - многомерные *OLAP*-системы, для которых структура куба хранится в специализированной многомерной базе данных; *HOLAP* - гибридные *OLAP*-системы, которые сочетают черты *ROLAP* и *MOLAP*.

В литературе встречаются еще некоторые специфические виды *OLAP*-систем: *DROLAP* (dense-region based) - *OLAP*-анализ, основанный на плотных областях; *OOLAP* (object-relational) - объектно-реляционный *OLAP*-анализ, схожий с *ROLAP*; *RTOLAP*, *R-ROLAP* (real-time) - *ROLAP* реального времени (или *Real-Time Analytical Processing - RAP*), отличающийся от *ROLAP* тем, что для хранения агрегатов не создаются дополнительные реляционные таблицы; *In-Memory OLAP* (*ROLAP* и *MOLAP*) - практически не отличающиеся от *Real-Time ROLAP*; *DOLAP* (desktop) - настольный *OLAP*-анализ с одноуровневой технологией *OLAP*; *Application* - прикладной *OLAP*-анализ, представляющий собой программу просмотра (viewer), которая улучшает обслуживание пользователей; *WOLAP* (web-based) - *OLAP*-анализ ориентированный на *Web* («тонкий клиент»); *SOLAP* (spatial) - пространственный *OLAP*-анализ, предназначенный для изучения пространственных данных; *SeOLAP* (semantic) - семантический *OLAP*-анализ, ориентированный на семантические методы поиска и извлечения данных и знаний; *Mobile* - *OLAP*-анализ для беспроводных сетей или мобильных устройств; *JOLAP* (java OLAP) - *OLAP*-анализ для создания и поддержания *OLAP* данных и метаданных на корпоративной платформе *Java*.

В сложных многошаговых процессах анализа данных, хранящихся в корпоративных длительно функционирующих информационных системах, где необходим комплексный анализ, связанный с реализацией аналитических алгоритмов расчета показателей, решением задач планирования и других, применение классических *OLAP*-решений требует повторения процедур построения многомерных кубов, что приводит к существенным временным затратам на повторную подготовку данных и снижает их эффективность.

Цель работы состояла в разработке алгоритмического и программного обеспечения, повышающих эффективность проведения *OLAP*-анализа и расширяющих возможности оперативной аналитической обработки за счет автоматизации этапа подготовки данных из реляционной базы и минимизации суммарного времени выполнения вычислений.

© Бондаренко А. В., Бессарабов Н. А., 2012

В работе использовалась инструментальная среда *Delphi 7.0* для создания программного обеспечения. Экспериментальный анализ проводился с помощью компьютерного моделирования для заполнения записями таблиц баз данных на *MS SQL Server 2000*. Для оценок сложности алгоритмов применялись методы теории структур данных и баз данных, комбинаторики, теории вероятностей и математической статистики.

Задачи, решаемые в данной работе:

1. Создать программный интерфейс подготовки данных из исходной базы данных для *OLAP*-анализа. Программа должна по заданной основной таблице прочитать все внешние ключи в исходной базе данных, выделить структуру «снежинка» или «звезда» и подготовить *SQL*-запрос, тем самым, автоматизировав эту часть работы и избавив Аналитика от необходимости составления запроса.

2. Обосновать преимущество специализированных алгоритмов построения многомерного *OLAP*-куба по сравнению с применением обычных *SQL*-запросов посредством экспериментально полученных данных о времени выполнения. Для этого случайно сгенерировать базы данных с различным числом записей, полей и шкал измерений, запрограммировать на языке *SQL* запросы для вычисления агрегирующих функций с различными сочетаниями измерений, необходимых для куба, а также разработать программу построения куба с применением основного компонента (HierCube) с фиксацией времени выполнения. Применяя разработанные программы, провести эксперименты по построению куба двумя способами с различным числом записей, полей и шкал измерений, построить соответствующие зависимости времени построения куба от числа записей, полей и шкал измерений, провести статистическую обработку результатов эксперимента.

3. Минимизировать по времени полный цикл выполнения *OLAP*-анализа на основе теоретических оценок: времени выполнения *SQL*-запросов при построении таблиц из структуры «снежинка» или «звезда»; времени построения многомерного куба; минимизации суммарного времени от запроса к исходной базе данных до окончания вычисления куба в итеративном процессе *OLAP*-анализа многомерного куба с варьированием состава измерений.

Дана реляционная база данных (БД) в виде таблицы фактов с F полями, R записями и N первичными ключами, соответствующими некоторым N таблицам, каждая i-я из которых состоит из F_i полей и R_i записей, $i=1, ..., N$.

Аналитик выполняет исследовательскую задачу с применением *OLAP*-технологии, а именно, на очередной j-й итерации своего исследования определяет набор $< F_{j_1},...,F_{j_{d_j}} >$ полей таблицы фактов и таблиц измерений из исходной БД, $d_j \le F + F_1 + ... + F_N$ получает d_j-мерный куб данных с заданным набором $< A_{j_1},...,A_{j_k} >$ показателей (агрегирующих функций) и проводит собственно исследовательскую работу, используя аналитические *OLAP*-операции: сечение гиперкуба, вращение (rolling), консолидацию, операцию спуска (drill-down), разбиение с поворотом (slicing and dicing).

Аналитик может завершить исследование на j-й итерации, либо принять решение изменить набор исследуемых полей $< F_{j_1},...,F_{j_{d_j}} >$ из исходного множества в количестве F полей, включив дополнительные поля и, быть может, исключив некоторые из них. Исследовательская работа продолжается, выполняя $j:=j+1$ итерацию с тем же срезом БД, то есть, при том же наборе R записей БД.

Задача состоит в разработке алгоритма обработки данных, при котором время подготовки данных минимально. Кроме того, требуется экспериментально подтвердить теоретический результат.

Для этого из исходной БД выбирается максимально возможный для структуры «звезда» или «снежинка» состав полей F. Предполагая, что времена обработки пропорциональны числу элементарных операций над записями БД, выводятся аналитические оценки для T_p (время на подготовку данных) и T_{cube} (время на построение куба), и показывается, что при достаточно большом числе итераций $n > n_0$ суммарное время вычислений куба меньше, чем если бы на каждой итерации выбирать только интересующий Аналитика набор полей и повторять итерации, начиная с выполнения *SQL*-запросов к исходной БД и построения дерева, то есть, выполняя весь цикл вычислений, а не только единожды выполненного среза БД. Будем называть такую стратегию анализа данных - избыточным построением многомерного куба.

2. Теоретическая оценка времени выполнения проверки структуры, создания дерева поиска и таблицы фактов при построении многомерного куба из структуры базы данных «снежинка» или «звезда»

Из исходной БД выбирается максимально возможный для структуры «звезда» или «снежинка» состав полей F. Предполагаем, что времена обработки пропорциональны числу элементарных операций над записями БД.

Пусть $f_1(N)$ - время проверки структуры БД и построения дерева поиска (структура «звезда» или «снежинка»). Для проверки структуры и построения дерева поиска в худшем случае необходимо провести $N!$ сравнений. При времени проведения одного сравнения τ_1 время проверки структуры $f_1(N) = 2(N-1)!\cdot\tau_1$.

Действительно, пусть в исходной базе данных заданы N внешних ключей $FK_1, ..., FK_N$ и N первичных ключей $PK_1, ... , PK_N$. Для построения дерева поиска типа «снежинка» или «звезда» необходимо просматривать каждый внешний ключ FK_i и добавлять первичный ключ PK_i посредством конструкции: *if FKi = FactsName then add PKi to Tree* и затем присваивать имя первичного ключа: *FactsName = PKi*. Для выполнения указанных процедур требуется перебрать следующее число вариантов:

$1(N-1)(N-2) \ldots 2\cdot1 + (N-1)(N-2) \ldots 2\cdot1 = 2(N-1)!$

Пусть $f_2(R,F,N)$ - время создания таблицы фактов, то есть выбора из БД всех нужных записей. Пусть $d_i > 1,\ i=1,\ldots,N$ - размер шкалы i-го измерения, то есть, число различных значений измерений i-го атрибута.

При среднем времени считывания одного поля по запросу < *select * from* >, равном τ_2 время создания таблицы фактов $f_2(R,F,N) = R\sum_{i=1}^{N}\log(d_i)\tau_2$.

Пусть T_{cube} - время создания многомерного куба.

Тогда суммарное время подготовки данных и построения куба T_0 равно

$$T_0 = f_1(N) + f_2(R,F,N) + T_{cube} = 2(N-1)!\,\tau_1 + R\sum_{i=1}^{N}\log(d_i)\tau_2 + T_{cube}$$

Пусть Аналитик выполняет n итераций *OLAP*-анализа, то есть, n раз выбирает состав интересующих измерений, выполняет процедуру подготовки данных и построения многомерного куба и, получив различные представления многомерного куба, проводит анализ данных.

Тогда T - суммарное время подготовки данных и построения многомерного куба за все n итераций равно

$$T = \sum_{j=1}^{n}\left(f_1(N) + f_2(R,F,N)\right) + \sum_{j=1}^{n}T_{cube}(j) = = \sum_{j=1}^{n}\left(2(N_j-1)!\,\tau_1 + R\sum_{k=1}^{N_j}\log(d_{i_k})\tau_2\right) + \sum_{j=1}^{n}T_{cube}(j)$$

где $T_{cube}(j)$ - время создания многомерного куба на j-й итерации;

N_j - число первичных ключей БД использующихся на j-й итерации;

F_j - множество первичных ключей БД, использующихся на j-й итерации;

$d_{i_k} > 1,\ k=1,\ldots,N_j$ - размер шкалы i_k-го измерения, то есть, число различных значений измерений i_k -го атрибута;

$(i_1,\ldots,i_{Nj})$ - множество индексов являющихся подмножеством индексов $(1,\ldots,N)$ измерений, то есть, на j-й итерации Аналитиком используется для анализа некоторое подмножество измерений атрибутов.

3. Минимизация суммарного времени обработки от запроса к исходной базе данных до окончания вычислений в итеративном процессе анализа многомерного куба с варьированием состава измерений

Аналитику представляется целесообразным применять следующую стратегию анализа данных: очертить широкий, возможно избыточный, круг измерений, то есть, интересующих атрибутов, который включает в себя все атрибуты, от которых когда-либо будет необходимо исследовать зависимости, а затем проводить тактику итеративных заданий запросов построения многомерного куба для некоторых наборов измерений.

Для сформулированной стратегии анализа данных суммарное время подготовки данных и построения многомерного куба за все n итераций T_{opt} равно

$$T_{opt} = f_1(N) + f_2(R,F,N) + \sum_{j=1}^{n}T_{cube}(j) = 2(N-1)!\,\tau_1 + R\sum_{i=1}^{N}\log(d_i)\tau_2 + \sum_{j=1}^{n}T_{cube}(j)$$

где $T_{cube}(j)$ - время создания многомерного куба на 1-й итерации, а для $j>1$ - время фильтрации куба на j-ой итерации *OLAP*-анализа;

N - число всех первичных ключей БД, которые будут использованы;

F - множество всех первичных ключей БД, которые будут использованы (исчерпывающий набор атрибутов);

$d_k > 1,\ k=1,\ldots,N$ - размер шкалы k-го измерения, то есть, число различных значений измерений k-го атрибута.

Покажем, что при $n > n_0$, где $n_0 = n_0(N,R,F)$ - некоторое число, выполняется неравенство $T_{opt} < T$, то есть, при достаточно большом числе итераций $n > n_0$ суммарное время вычислений куба меньше, чем, если бы на каждой итерации выбирать только интересующий Аналитика набор полей и повторять итерации, начиная с построения дерева и выполнения *SQL*-запросов к исходной БД, а именно, выполняя весь цикл вычислений, а не только единожды выполненного среза БД.

Докажем, что

$$2(N-1)!\,\tau_1 + R\sum_{i=1}^{N}\log(d_i)\tau_2 + \sum_{j=1}^{n}T_{cube}(j) < \sum_{j=1}^{n}\left(2(N_j-1)!\,\tau_1 + R\sum_{k=1}^{N_j}\log(d_{i_k})\tau_2\right) + \sum_{j=1}^{n}T_{cube}(j)$$

при $n > n_0$, где $n_0 = \dfrac{2(N-1)! + RD}{2(N-1)! + RND_0}$,

N - число первичных ключей,

R - число записей в таблице фактов,

$D = \sum_{k=1}^{N}\log(d_k)$,

$d_k > 1,\ k=1,\ldots,N$ - размер шкалы k-го измерения, то есть, число различных значений измерений k-го атрибута,

$D_0 = \log(\min(d_1,\ldots,d_N))$ - соответствует минимальному размеру шкалы различных значений измерений.

Действительно, исходное неравенство эквивалентно следующему:

$$2(N-1)!\,\tau_1 + R\sum_{k=1}^{N}\log(d_k)\tau_2 < \sum_{j=1}^{n}\left(2(N_j-1)! + R\sum_{k=1}^{N_j}\log(d_{i_k})\tau_2\right)$$

Если $D_0 = \log(\min(d_1,\ldots,d_N))$, а $N \geq N_j > 1$ то:

$2(N-1)!\tau_1 + R\,D\tau_2 < 2n(N-1)!\tau_1 + nRND_0\tau_2$

Время проведения одного сравнения τ_1 и среднее время считывания одного поля τ_2 соответствуют времени выполнения одного и того же физического процесса считывания информации всех полей одной записи. Поэтому их различие не имеет существенного значения и неравенство примет следующий вид $2(N-1)!+RD < 2n(N-1)! + n\,RND_0$.

Поэтому $n_0 = \dfrac{2(N-1)!+RD}{2(N-1)!+RND_0}$.

Таким образом, при числе итераций работы Аналитика с подготовкой данных *OLAP*-анализа более некоторого значения n_0: $n > n_0$, выполняется неравенство $T_{opt} < T$, то есть, при достаточно большом числе итераций $n > n_0$ суммарное время вычислений куба меньше, чем, если бы на каждой итерации выбирать только интересующий Аналитика набор полей и повторять итерации, начиная с построения дерева и выполнения *SQL*-запросов к исходной БД, а именно, выполняя весь цикл вычислений, а не только единожды выполненного среза БД. Это свидетельствует об эффективности предложенной в работе методики обработки данных для *OLAP*-анализа.

4. Сравнение экспериментально полученных значений времени в итеративном процессе анализа многомерного куба по обычной и избыточной методике построения

Для проверки преимущества схемы работы с избыточным построением куба по сравнению с обычной схемой были сгенерированы 10 таблиц базы данных, заполненных случайными числами.

Генерировались случайные наборы данных распределенных равномерно с 3, 5, 7, 10, 15, 20, 25, 30, 35 и 40 измерениями, каждое со шкалой (числом различных значений шкалы) 10 и объемом выборки 1000000.

Для создания записей базы данных запускался генератор псевдослучайных чисел *Randomize*, и затем в цикле использовалась стандартная функция *Random()*, возвращающая случайное число, равномерно распределенное на интервале (0; 1). Для заданного числа измерений (полей) *m_field* и заданной шкалы каждого измерения (число различных значений шкалы) *m_izm* генерировались значения содержимого полей при помощи конструкции:

```
for i_field:=1 to n_field
begin
m_izm= m_field +(i_field-1)* m_field;
y :=random();
for i:=1 to m_izm begin if y>=(i-1)/m_izm.and.y<=i/m_izm then x:=i/m_izm end;
end;
```

а также генерировалось соответствующее имя поля, и добавлялись записи в соответствующую таблицу базы данных.

Для вычислительного эксперимента был подготовлен фрагмент программы вызова схемы с построением избыточных многомерных кубов с агрегатом *COUNT* для каждой из 10 таблиц базы данных с фиксацией показаний таймера от начала и по окончании построения куба. Чтобы проверить время выполнения аналогичных операций при помощи обычной схемы, подготовлен фрагмент программы с вычислением агрегата *COUNT* для всех ячеек многомерного куба для каждой из 10 таблиц базы данных с фиксацией показаний таймера от начала и по окончании эксперимента.

Вычислительный эксперимент повторялся 30 раз для одной и той же базы данных и для каждой схемы построения многомерных кубов, вычислялось среднее время выполнения и среднеквадратичное отклонение времени. Результаты эксперимента приведены в Таблице 1.

С помощью критерия Стьюдента была проверена гипотеза о равенстве средних.

Табл. 1. *Зависимость времени построения многомерного куба двух алгоритмов от числа измерений*

Число измерений (шкала каждого измерения равна 10)	Время анализа куба по стратегии избыточного построения куба, с	Время анализа куба по обычной стратегии, с
3	6,2±1,7	6,2±1,7
5	18,3±2,1	19,1±2,4
7	26,5±3,0	29,2±3,2
10	32,8±4,3	39,0±4,9
15	41,5±6,4	54,0±6,8
20	48,2±8,6	65,2±8,8
25	57,2±9,7	75,3±9,9
30	66,0±10,4	85,8±10,7
35	72,5±11,0	90,8±11,1
40	77,2±11,1	96,2±11,6

На Рисунке 1 представлена зависимость времени построения многомерного куба двух схем работы от числа измерений.

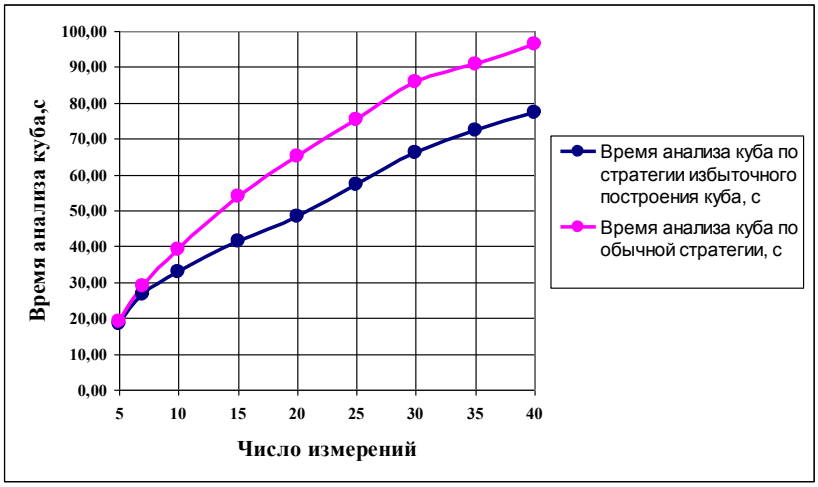

Рис. 1. *Зависимость времени построения многомерного куба двух схем работы от числа измерений*

Анализ данных, представленных в Таблице 1 и на Рисунке 1, свидетельствует о превосходстве эффективности схемы с построением избыточного куба по сравнению с обычной схемой при достаточно высокой размерности задачи (более 10 измерений) - значения времени выполнения отличаются статистически существенно с высоким уровнем значимости ($P < 0{,}001$). При размерности менее 10 обе схемы практически эквивалентны (*P-value* = 0,1).

Таким образом, при размерности задачи 10 Аналитику представляется целесообразным применять следующую стратегию анализа данных: очертить широкий, возможно избыточный, круг измерений, то есть, интересующих атрибутов, который включает в себя все атрибуты, от которых когда-либо будет необходимо исследовать зависимости, а затем проводить тактику итеративных фильтраций измерений многомерного куба. Это свидетельствует об эффективности предложенной в работе методики обработки данных для *OLAP*-анализа (схемы с избыточным построением многомерного куба).

Список литературы

1. **Бергер А.** Microsoft SQL Server 2005 Analysis Services. OLAP и многомерный анализ данных. СПб.: БХВ-Петербург, 2007.
2. **Гудков А. С.** Использование префиксных деревьев при построении систем анализа данных: автореф. дисс. … канд. физ.-мат. наук. М., 2006.
3. **Заботнев М. С.** Разработка методов и средств анализа многомерных баз данных с неполной информацией: автореф. дисс. … канд. техн. наук. М., 2006.
4. **Ишенин П. П.** Инструментальные средства построения комплексов моделей и аналитических приложений в OLAP-технологии: дисс. … канд. техн. наук. Красноярск, 2006.
5. **Кудрявцев Ю. А.** Алгоритмы эффективной обработки MOLAP-кубов: дисс. … канд. физ.-мат. наук. М., 2009.
6. **Agrawal R., Ramakrishnan S.** Fast Algorithms for Mining Association Rules // Proc. 20th Int. Conf. Very Large Data Bases. Morgan Kaufmann, 1994. P. 487-499.
7. **Beyer K., Ramakrishnan R.** Bottom-Up Computation of Sparse and Iceberg Cubes // SIGMOD. 1999.
8. **Codd E. F.** Providing OLAP for End-User Analysis: an IT Mandate // Computer World. USA, 1993.
9. **Deshpande P. M., Agarwal S., Naughton J. F., Ramakrishnan R.** Computation of Multidimensional Aggregates: Technical Report 1314. University of Wisconsin, 1996.
10. **Dong Xin, Jiawei Han, Xiaolei Li, Zheng Shao, Wah B. W.** Computing Iceberg Cubes by Top-Down and Bottom-Up Integration: the StarCubing Approach // IEEE Transactions on Knowledge and Data Engineering. 2007.
11. **Han Jiawei, Kamber M.** Data Mining: Concepts and Techniques. Second Edition. University of Illinois at Urbana-Champaign, 2009.
12. **Harinarayan V., Rajaraman A., Ulman J.** Implementing Data Cubes Efficiently // SIGMOD. 1996.
13. **Karloff H. J., Mihail M.** On the Complexity of View-Selection Problem // PODS. 1999.
14. **Nigel Pendse.** Olapreport: Database Explosion // The OLAP Report. 2005.
15. **Rafanellir M.** Multidimensional Databases: Problems and Solutions. Idea Group Publishing, 2003.
16. **Sarawagi S., Agrawal R., Gupta A.** On Computing the Data Cube: Technical Report RJ10026. IBM Almaden Research Center, 1996.
17. **Thomsen E.** OLAP Solutions: Building Multidimensional Information Systems. Second Edition. Wiley Computer Publishing; John Wiley & Sons, Inc., 2002.
18. **Xiaolei Li, Dong Xin Jiawei, Wah B. W.** Star-Cubing: Computing Iceberg Cubes by Top-Down and Bottom-Up Integration // VLDB. 2003.