УДК 004.652:658

ТЕНЗОРНАЯ МОДЕЛЬ ИНТЕГРИРОВАННОГО ХРАНИЛИЩА ДАННЫХ ИНФОРМАЦИОННОЙ СИСТЕМЫ ПРЕДПРИЯТИЯ

Е. Е. Бизянов, Н. Н. Кононенко
Донбасский государственный технический университет,
г. Алчевск, Луганская народная республика
bpeelecs@gmail.com, nikolay.kononenko.1986@gmail.com

В статье описана тензорная модель интегрированного хранилища данных предприятия. Проведен анализ проблемы интеграции баз данных, представленных в различных форматах. Рассматриваются известные на сегодняшний день тензорные модели баз данных. Предложен принцип построения тензорной модели интегрированной реляционной базы данных в виде многомерной матрицы, причем в качестве инварианта предложено рассматривать отношение. Рассмотрен пример тестирования предложенной модели построения интегрированной реляционной базы данных. Предложен способ хранения многомерных и двумерных матриц в реляционной базе данных, который предполагает умножение многомерной матрицы на двумерную матрицу с использованием SQL-запроса.

Сравнительный анализ скорости для предлагаемого метода для выбора метаданных интегрированного хранилища реляционных данных и метода, основанного на анализе системных таблиц, выполнен с помощью тестовых наборов случайных данных.

Использование тензорного представления баз данных информационной системы позволяет получить компактное математическое описание, а двоичное кодирование сущностей реляционной модели – упростить программный доступ к интегрированному хранилищу данных. Экспериментально подтверждено, что когда выборка содержит более 15 000 записей, тензорное представление хранилища данных позволяет сократить время выполнения запросов от 25 до 11 500 %.

Ключевые слова: базы данных, тензорные модели, интегрированная реляционная база данных.

TENSOR MODEL OF INTEGRATED DATA STORAGE OF ENTERPRISE INFORMATION SYSTEMS

E. E. Bizyanov, N. N. Kononenko
Donbas State Technical University, Alchevsk, Luhansk People's Republic
bpeelecs@gmail.com, nikolay.kononenko.1986@gmail.com

The paper presents a tensor model of the enterprise's integrated data storage. Problem analysis of databases integration represented in different formats is conducted. Tensor models of databases currently known are considered. The principle of constructing a tensor model of an integrated relational database in the form of a multidimensional matrix is proposed, where the relation is considered as an invariant. A testing example of the proposed model for building an integrated relational database is described. A method for storing multidimensional and two-dimensional matrices in the relational database is proposed, where the SQL query is used for the multiplication of a multidimensional matrix by a two-dimensional matrix.

A comparative analysis of the speed for the proposed method and the method based on the analysis of system tables is carried out using test sets of random data.

The usage of the tensor representation of the information system's databases allows obtaining a compact mathematical description, and the binary coding of the relational model entities al-

lows simplifying the programmatic access to the integrated data storage. Experimentally confirmed that the sample contains more than 15 000 records, the tensor representation of the data storage allows shortening the time of execution from 25 % to 11 500 %.

Keywords: databases, tensor models, integrated relational database.

Введение. Крупные промышленные предприятия, научно-исследовательские центры и государственные организации начали создавать собственные информационные системы (далее – ИС) еще в 70-х годах прошлого века. Появление в 80-х годах персональных компьютеров, развитие локальных и глобальных информационных сетей, а также новых языков программирования привело к ускорению этого процесса. Развитие микроэлектроники позволило существенно снизить цену на изделия вычислительной техники, что способствовало приобщению к процессу информатизации общества небольших фирм и учебных заведений.

Развитие ИС сопровождалось развитием специального программного обеспечения – систем управления базами данных (далее – СУБД), которые способствовали переводу документооборота в электронную форму, упрощению получения справочной информации, а также решению различных задач управления предприятиями и организациями. На действующих предприятиях локальные базы данных (далее – БД) для различных предметных областей и подразделений создавались в течение длительного времени, а для доступа к ним формировались центры актуализации информации. Одной из проблем использования БД, созданных в разные годы, является использование различных технологий доступа к данным. Перенос же существующих БД в среду более современных СУБД связан со значительными затратами ресурсов, он может привести к потере и нарушению целостности хранимых данных. Кроме того, использование разнородных БД предполагает наличие специалистов, способных их использовать и обслуживать, что также сопряжено с затратами [1].

С точки зрения программирования приложений для работы с базами данных удобнее было бы получать доступ к различным БД, имеющимся в ИС предприятия, с использованием одного механизма. В настоящее время интеграция БД, работающих под управлением разных СУБД, решается за счет применения моделей доступа к данным (ADO, ODBC, OLEDB, JDBC), обеспечивающих унифицированный механизм доступа к базам данных различных производителей, а также путем использования оболочек/адаптеров, выполняющих преобразование запросов в форму, приемлемую для конкретных СУБД [2–5].

Одним из подходов к интеграции данных предприятия является использование моделей распределенных баз данных. Так, в работе [6] предлагается рассматривать совокупность разнородных БД как одну распределенную базу данных, что позволяет применять модели распределенных БД для выполнения запросов, репликации данных и управления транзакциями. Одним из развивающихся направлений является использование тензорного анализа для получения информации, хранящейся во всех базах данных информационной системы [7–9].

Таким образом, интеграция баз данных, работающих под управлением различных СУБД и входящих в информационную систему предприятия с целью представления их как единого хранилища, до сих пор является актуальной проблемой.

Тензорные модели в теории баз данных

Единообразный механизм доступа к информации, хранящейся в матричной форме, который используется в реляционной модели данных [6] и в тензорном анализе [10], способствует развитию исследований на стыке этих областей. Обычно тензоры описывают с помощью матриц, и, так как отношения реляционной модели – тоже матрицы, это позволяет применить аппарат тензорного анализа для описания баз данных и манипулирования данными,содержащихся в них.

Использовать тензоры при проектировании баз данных информационных систем предлагалось в [8] с применением следующей аналогии с реляционной моделью: отноше-

ние – это матрица, кортеж – вектор, атрибут – скользящий индекс и т. д. Как инвариант, в [7] предлагается рассматривать «траекторию некоторого объекта».

В статье [7] предложено применить тензорную методологию для построения модели данных информационной системы путем привязки к сущностям предметной области. В качестве инвариантов в статье [7] предлагается использовать: объем данных (как константу) для описания одного и того же набора данных в различных предметных областях, а также объем данных (как константу), хранимых в таблицах мер для различных предметных областей.

В [9] рассмотрена тензорная модель базы данных, реализованная в СУБД SciDB для выполнения плана запросов. В этой модели база данных представлена многомерной матрицей, разделенной на фрагменты, а выборка производится путем указания в операторе SELECT объекта для выборки (отношения реляционной базы данных) и индексов измерений, выбранных для просмотра. Как ранг тензора в [9] принято количество атрибутов в отношениях реляционной базы данных.

Принципы построения тензорной модели интегрированной реляционной базы данных

Главной задачей тензорного анализа является поиск инвариантов [10], в качестве которых могут выступать константы [7, 10], выражения, массивы и т. п. [10]. Реляционная модель данных обладает уникальным свойством: любая операция над любым количеством отношений всегда дает отношение [6]. Следовательно, при построении тензорной модели в качестве инварианта в первую очередь следует выбрать отношение (таблицу) как объект, имеющий постоянную структуру: атрибуты определяют столбцы, кортежи – строки. При этом количество атрибутов и кортежей с точки зрения структуры модели несущественно.

На сегодняшний день известно несколько разновидностей тензоров, но мы будем рассматривать только тензоры преобразования.

Пусть все базы данных ИС предприятия используют реляционную модель и объединены во множество:

$$DB = \{DB_1, DB_2, \ldots, DB_n\}, n = \overline{1, N}, \quad (1)$$

где DB – множество баз данных, входящих в информационную систему;

DB_n – отдельно взятая БД;

N – количество баз данных в информационной системе.

Реляционная база данных представляет собой множество отношений (таблиц), каждое из которых содержит множество атрибутов, определенных на множестве доменов, при этом значения атрибутов собраны в кортежи, а отношения объединены между собой множеством связей [6]. Следовательно, реляционную базу данных можно представить в виде множества:

$$DB = \{T, A, D, VA, C, R\}, \quad (2)$$

где $T = \{T_1, T_2, \ldots, T_{nt}\}, nt = \overline{1, Nt}$ – множество отношений (таблиц), входящих в базу данных; Nt – количество отношений в БД;

$A = \{A_1, A_2, \ldots, A_{na}\}, na = \overline{1, Na}$ – множество атрибутов, входящих во все отношения базы данных, Na – количество атрибутов в отношениях БД;

$D = \{D_1, D_2, \ldots, D_{nd}\}, nd = \overline{1, Nd}$ – множество доменов, на которых определены атрибуты базы данных, Nd – количество доменов, используемых в БД;

$VA = \{VA_1, VA_2, \ldots, VA_{nva}\}, nva = \overline{1, Nva}$ – множество значений атрибутов всех отношений базы данных, Nva – количество значений атрибутов;

$C = \{C_1, C_2, \ldots, C_{nc}\}$, $nc = \overline{1, Nc}$ – множество кортежей в отношениях базы данных, Nc – количество кортежей;

$R = \{R_1, R_2, \ldots, R_{nr}\}$, $nr = \overline{1, Nr}$ – множество связей между отношениями базы данных, Nr – количество связей между отношениями в БД.

Обычно количество доменов во множестве D не превышает количества атрибутов во множестве A, т. е. $Nd \le Na$ [6].

Каждый атрибут $a \in A$, использующийся в отношении базы данных, имеет имя и определен на некотором домене $d \in D$:

$$D \to A = \{\forall a \in A \exists d \in D\}. \quad (3)$$

Каждый кортеж отношения есть отображение множества атрибутов A на множество значений атрибутов VA:

$$A \to VA = \{\forall a \in A \exists va \in VA\}. \quad (4)$$

Множество всех отношений, входящих в интегрированную базу данных, представим следующим образом:

$$T = \bigcup((A \to VA) \times (D \to A)). \quad (5)$$

Добавим еще одно измерение для связей между отношениями, определяющее связи между отношениями, и получим описание отдельной базы данных, входящей в состав интегрированной базы данных как четырехмерной матрицы:

$$DB_{\alpha\beta\gamma\delta} = \bigcup((A \to VA) \times (D \to A) \times R), \quad (6)$$

где α – скользящий индекс, отражающий атрибуты, использующиеся в отношениях базы данных;

β – скользящий индекс, отражающий домены, на которых определены атрибуты;

γ – скользящий индекс, отражающий отношения базы данных;

δ – скользящий индекс, отражающий связи между отношениями базы данных.

Для различных структурных подразделений предприятия информация может быть востребована в разрезе связей между отношениями различных баз данных. Обозначив множество связей между таблицами всех баз данных, входящих в состав ИС, как IR, представим множество отображения данных следующим образом:

$$L_{\alpha\beta\gamma\delta\varepsilon} = \bigcup((A \to VA) \times (D \to A) \times R) \times IR, \quad (7)$$

где L – множество отображений данных;

DB – множество БД информационной системы;

IR – множество связей между таблицами различных БД;

ε – скользящий индекс, отражающий связи между отношениями всех баз данных информационной системы.

Для извлечения данных из множества L и преобразования их в форму, приемлемую для структурного подразделения, потребуется выполнить операцию их отображения:

$$L \to V = \{\forall l \in L \exists v \in V\}, \quad (8)$$

где V – множество, содержащее информацию, актуальную с точки зрения конкретного структурного подразделения предприятия.

Количество элементов множества V будет равно количеству структурных подразделений предприятия.

Если рассматривать множество IR как подмножество множества R, это позволит уменьшить количество измерений модели до четырех. Тогда операцию отображения множества L можно заменить операцией отображения каждого элемента множества DB:

$$\mathrm{DB} \to \mathrm{V} = \{\forall \mathrm{db} \in \mathrm{DB}\, \exists\, \mathrm{v} \in \mathrm{V}\}. \tag{9}$$

Рассмотрим гипотетический пример. Пусть в ИС предприятия входят три базы данных $\mathrm{DB} = \{\mathrm{DB}_1, \mathrm{DB}_2, \mathrm{DB}_3\}$, содержащих по четыре отношения $\mathrm{T} = \{\mathrm{T}_1, \mathrm{T}_2, \mathrm{T}_3, \mathrm{T}_4\}$, каждое из которых содержит четыре атрибута $\mathrm{A} = \{\mathrm{A}_1, \mathrm{A}_2, \mathrm{A}_3, \mathrm{A}_4\}$, причем все атрибуты отношений определены на четырех доменах $\mathrm{D} = \{\mathrm{D}_1, \mathrm{D}_2, \mathrm{D}_3, \mathrm{D}_4\}$.

Используя формулы (3), (4) и (5), представим трехмерную структуру интегрированной базы данных следующими матрицами:

$$\mathrm{DB1} = \alpha \downarrow \overset{\beta}{\underset{}{\rightarrow}} \begin{bmatrix} \mathrm{D1} & & \mathrm{D4} & \mathrm{D2} \\ & \mathrm{D3} & \mathrm{D4} & \\ \mathrm{D1} & & & \mathrm{D2} \\ \mathrm{D1} & & \mathrm{D4} & \mathrm{D2} \end{bmatrix}$$

$$\mathrm{DB2} = \alpha \downarrow \overset{\beta}{\underset{}{\rightarrow}} \begin{bmatrix} & & \mathrm{D4} & \mathrm{D2} \\ & \mathrm{D3} & \mathrm{D4} & \\ & & & \mathrm{D2} \\ \mathrm{D1} & & \mathrm{D4} & \end{bmatrix}$$

$$\mathrm{DB3} = \alpha \downarrow \overset{\beta}{\underset{}{\rightarrow}} \begin{bmatrix} \mathrm{D1} & & & \\ & \mathrm{D2} & & \\ & & \mathrm{D3} & \\ & & & \mathrm{D4} \end{bmatrix}.$$

Каждая матрица отображает структуру многомерной БД, а именно отношения, их атрибуты и домены. Исходя из (1), интегрированную базу данных представим в виде многомерной матрицы:

$$\mathrm{DB} = \left\| \begin{array}{cccc|cccc|cccc} \mathrm{D1} & 0 & \mathrm{D4} & \mathrm{D2} & 0 & 0 & \mathrm{D4} & \mathrm{D2} & \mathrm{D1} & 0 & 0 & 0 \\ 0 & \mathrm{D3} & \mathrm{D4} & 0 & 0 & \mathrm{D3} & \mathrm{D4} & 0 & 0 & \mathrm{D2} & 0 & 0 \\ \mathrm{D1} & 0 & 0 & \mathrm{D2} & 0 & 0 & 0 & \mathrm{D2} & 0 & 0 & \mathrm{D3} & 0 \\ \mathrm{D1} & 0 & \mathrm{D4} & \mathrm{D2} & \mathrm{D1} & 0 & \mathrm{D4} & 0 & 0 & 0 & 0 & \mathrm{D4} \end{array} \right\|.$$

Такая матрица удобна для восприятия человеком, но не подходит для анализа машинными средствами, поэтому произведем ее двоичное кодирование. Например, при количестве баз данных, равном 3, достаточно под код БД выделить 2 бита числа: 0 – DB1, 1 – DB2, 2 – DB3. При количестве отношений, равном 4, достаточно выделить 3 бита для кодирования таблиц и столько же для кодирования атрибутов и т. д.

В качестве примера на рис. 1 приведены карты битов для однобайтного (рис. 1а) и четырехбайтного (рис. 1б) кода. Использование четырехбайтного кода позволяет закодировать объекты для 64 баз данных, 1024 отношений (таблиц) и 65536 атрибутов в них, что вполне достаточно для большинства практических случаев.

Биты							
7	6	5	4	3	2	1	0
БД		Таблица			Атрибут		

а

Биты		
0-15	16-25	26-31
Атрибут	Таблица	БД

б

Рис. 1. Карта битов для: а – однобайтного кода; б – четырехбайтного кода

Пример кодирования для рассматриваемого нами случая (Ndb = 3, Nt = 4, Na = 4) и использования четырехбайтного кода приведен в табл. 1.

Таблица 1

Пример кодирования атрибутов, таблиц и баз данных, входящих в состав гипотетического интегрированного хранилища данных

Двоичный код											Десятичный код
БД			Таблица				Атрибут				
DB1	0	0	T1	0	0	1	A1	0	0	1	9
DB1	0	0	T1	0	0	1	A3	0	1	1	11
DB1	0	0	T1	0	0	1	A4	1	0	0	12
DB1	0	0	T2	0	1	0	A2	0	1	0	18
DB1	0	0	T2	0	1	0	A3	0	1	1	19
DB1	0	0	T3	0	1	1	A1	0	0	1	25
DB1	0	0	T3	0	1	1	A4	1	0	0	28
DB1	0	0	T4	1	0	0	A1	0	0	1	33
DB1	0	0	T4	1	0	0	A3	0	1	1	35
DB1	0	0	T4	1	0	0	A4	1	0	0	36
DB2	0	1	T1	0	0	1	A3	0	1	1	75
DB2	0	1	T1	0	0	1	A4	1	0	0	76
DB2	0	1	T2	0	1	0	A2	0	1	0	82
DB2	0	1	T2	0	1	0	A3	0	1	1	83
DB2	0	1	T3	0	1	1	A4	1	0	0	92
DB2	0	1	T4	1	0	0	A1	0	0	1	97
DB2	0	1	T4	1	0	0	A3	0	1	1	99
DB3	1	0	T1	0	0	1	A1	0	0	1	137
DB3	1	0	T2	0	1	0	A2	0	1	0	146
DB3	1	0	T3	0	1	1	A3	0	1	1	155
DB3	1	0	T4	1	0	0	A4	1	0	0	164

Представим полученные в табл. 1 десятичные коды в виде многомерной матрицы, описывающей структуру хранилища данных:

$$DB = \left\| \begin{array}{cccc|cccc|cccc} 9 & 0 & 11 & 12 & 0 & 0 & 75 & 76 & 137 & 0 & 0 & 0 \\ 0 & 18 & 19 & 0 & 0 & 82 & 83 & 0 & 0 & 146 & 0 & 0 \\ 25 & 0 & 0 & 28 & 0 & 0 & 0 & 92 & 0 & 0 & 155 & 0 \\ 33 & 0 & 35 & 36 & 97 & 0 & 99 & 0 & 0 & 0 & 0 & 164 \end{array} \right\|.$$

Полученное представление интегрированного хранилища обеспечивает хранение сведений об интегрированной базе данных в компактной и удобной для последующего использования форме. Так, например, для получения перечня таблиц БД DB1, использующих за-

данный атрибут, необходимо выполнить операцию умножения матрицы, описывающей DB1 на вектор, задающий атрибут A1:

$$T = A \cdot B = \begin{vmatrix} 9 & 0 & 11 & 12 \\ 0 & 18 & 19 & 0 \\ 25 & 0 & 0 & 28 \\ 33 & 0 & 35 & 36 \end{vmatrix} \cdot \begin{vmatrix} 1 \\ 0 \\ 0 \\ 0 \end{vmatrix} = \begin{vmatrix} 9 \\ 0 \\ 25 \\ 33 \end{vmatrix},$$

где A – матрица, описывающая базу данных DB1;

B – тензор преобразования.

В результирующем векторе мы получили ненулевой результат в строках, соответствующих отношениям, в которых встречается атрибут A1, в данном случае это таблицы T1, T3, T4.

Используя формулу умножения трехмерной матрицы n-го порядка на двухмерную матрицу того же порядка из [11], получим следующий результат:

$$T = A \cdot B = \left\| \begin{array}{cccc|cccc|cccc} 9 & 0 & 11 & 12 & 0 & 0 & 75 & 76 & 137 & 0 & 0 & 0 \\ 0 & 18 & 19 & 0 & 0 & 82 & 83 & 0 & 0 & 146 & 0 & 0 \\ 25 & 0 & 0 & 28 & 0 & 0 & 0 & 92 & 0 & 0 & 155 & 0 \\ 33 & 0 & 35 & 36 & 97 & 0 & 99 & 0 & 0 & 0 & 0 & 164 \end{array} \right\| \times$$

$$\times \left\| \begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right\| = \left\| \begin{array}{cccc|cccc|cccc} 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 137 & 0 & 0 & 0 \\ 0 & 18 & 0 & 0 & 0 & 82 & 0 & 0 & 0 & 146 & 0 & 0 \\ 25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 33 & 0 & 0 & 0 & 97 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right\|.$$

При интеграции разнородных БД в единое информационное пространство важным элементом являются метаданные [4]. В работах [3–5, 12] для хранения метаданных для разнородных источников предлагается использовать отдельное хранилище метаданных, однако при реализации подсистемы извлечения метаданных часто возникают проблемы, обусловленные гетерогенной природой разнородных БД [4]: коммуникационной неоднородностью, неоднородностью языков запросов, различием в типах данных, неоднородностью программной среды, распределенным характером организации хранилища данных.

Сведения об элементе многомерной матрицы будем хранить в виде кортежа <PK, VALUE>, где PK – первичный ключ таблицы базы данных; VALUE – элемент многомерной матрицы. Двумерную матрицу будем хранить в кортеже вида: <A, VALUE>, где A – код атрибута (первичный ключ), VALUE – значение элемента двумерной матрицы (0 или 1).

Длина записи при этом составляет 8 байт: 4 байта – ключевое поле, 4 байта – данные об элементе многомерной матрицы.

Перед проведением анализа метаданных из разнородных источников необходимо провести кодирование в моменты наименьшей загрузки узлов запросами от пользователей.

Для проведения операций с многомерной матрицей необходимо наличие информации о координатах каждого ее элемента. Координаты измерения (БД, таблица, атрибут) могут быть вычислены в процессе выполнения запроса из данных, хранящихся в поле VALUE с применением следующих формул:

$$DB = (VALUE \,\&\, 2080374784) / 33554432, \tag{10}$$

где 2080374784 – битовая маска, используемая для выделения битов БД;

33554432 – число, используемое для сдвига полученного результата вправо на 25 бит (рис. 1).

$$T = (VALUE \,\&\, 67043328) / 32768, \tag{11}$$

где 67043328– битовая маска, используемая для выделения битов таблицы;

32768 – число, используемое для сдвига полученного результата вправо на 15 бит (рис. 1).

$$A = VALUE \,\&\, 65535, \tag{12}$$

где 65535– битовая маска, используемая для выделения битов атрибута.

Для проведения сравнительного анализа предлагаемого метода и метода, основанного на анализе системных таблиц, с целью получения информации о структуре БД были подготовлены тестовые наборы случайных данных объемом от 1 500 до 1 010 000 записей. Каждый из запросов оперировал эквивалентными объемами данных, что позволило произвести оценку предлагаемого подхода в сравнении с вариантом анализа системных таблиц конкретной БД с целью получения информации о структуре БД.

Операцию умножения многомерной матрицы на двумерную матрицу, задающей атрибуты, выполним посредством следующего запроса:

```
SELECT
  s.T, s.A, s.DB, VALUE = sum(s.VALUE * v.VALUE)
FROM
  dbo.meta AS s
  INNER JOIN vect AS v ON s.A = v.A
GROUP BY
  s.T, s.A, s.DB
ORDER BY
  s.DB, s.T, s.A
```

Подобная информация также может быть получена путем анализа системных таблиц конкретной БД, например:

```
SELECT
  c.TABLE_NAME, c.COLUMN_NAME, c.DATA_TYPE
FROM
  INFORMATION_SCHEMA.COLUMNS c
  INNER JOIN dbo.cols cc on
    c.COLUMN_NAME = cc.COLUMN_NAME
    AND c.DATA_TYPE = cc.DATA_TYPE
GROUP BY
  c.TABLE_NAME, c.COLUMN_NAME, c.DATA_TYPE
```

Результаты выполнения тестовых запросов приведены в табл. 2.

Таблица 2

Время выполнения запросов

№ теста	Количество записей в выборке	Время выполнения запроса, мс (анализ системных таблиц)	Время выполнения запроса, мс (предлагаемый подход)	Прирост производительности, %
1	1500	156,0	124,2	25,60
2	2000	192,0	134,0	43,28
3	11000	238,8	290,0	–17,66
4	12000	477,8	223,4	113,88
5	13000	422,0	236,4	78,51
6	14000	493,0	248,0	98,79
7	15000	851,2	214,0	297,76
8	16000	971,4	220,4	340,74
9	17000	1081,2	360,8	199,67
10	18000	1229,2	281,2	337,13
11	19000	1293,2	242,0	434,38
12	20000	1538,2	230,0	568,78
13	110000	6841,2	1085,2	530,41

Окончание табл. 2

№ теста	Количество записей в выборке	Время выполнения запроса, мс (анализ системных таблиц)	Время выполнения запроса, мс (предлагаемый подход)	Прирост производительности, %
14	116000	10249,2	1113,2	820,70
15	122000	13603,6	1120,8	1 113,74
16	128000	17624,3	1129,6	1 460,22
17	134000	21025,7	1154,6	1 721,04
18	140000	42196,0	1197,6	11 595,12
19	146000	48744,0	1264,8	3 753,89
20	152000	56006,0	1235,6	4 432,70
21	158000	61670,0	1284,4	4 701,46
22	164000	68300,0	1421,4	4 705,12
23	1010000	71024,0	8926,7	695,64

Для наглядного представления результатов эксперимента по данным табл. 2 построим графики (рис. 2). Здесь кривая f1 – зависимость времени выполнения запроса с использованием предлагаемого метода; кривая f2 – зависимость времени выполнения запроса на основе анализа системных таблиц БД. Для сжатия графики представлены в логарифмическом масштабе. По оси абсцисс отложен десятичный логарифм количества записей в выборке, по оси ординат – десятичный логарифм времени выполнения запроса.

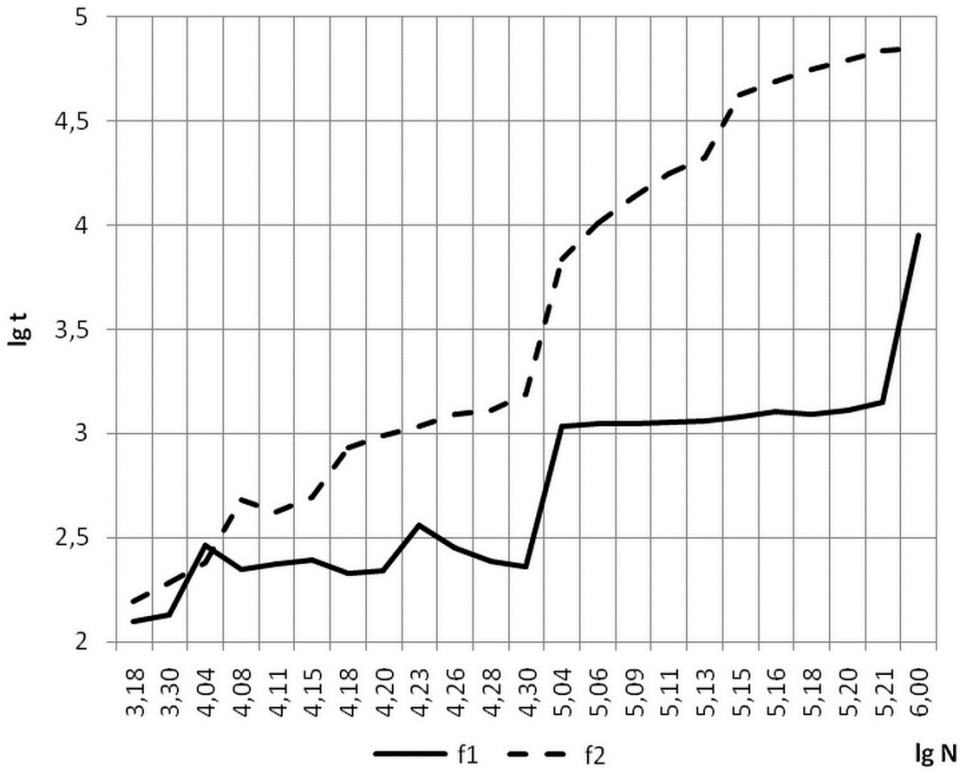

Рис. 2. Зависимость времени выполнения запроса от количества элементов данных

Как видно из полученных графиков, при возрастании количества обрабатываемых записей предлагаемый метод в сравнении с методом, использующим анализ системных таблиц БД, начинает давать выигрыш по времени выборки (график f1) уже при количестве записей более 15 000.

Выводы. Использование тензорного представления баз данных информационной системы как единого интегрированного хранилища данных позволяет получить компактное

математическое описание, а двоичное кодирование сущностей реляционной модели позволяет упростить программный доступ к интегрированному хранилищу данных.

Экспериментально подтверждено, что при выборках, включающих более 15 000 записей, тензорное представление интегрированного хранилища данных позволяет снизить время на выполнение запросов от 25 до 11 500 %, причем прирост выигрыша по времени увеличивается по мере роста количества записей в отношениях баз данных.

Литература

1. Заблодский Н. Н., Бизянов Е. Е., Зайцев И. С. Современная концепция построения информационной системы управления крупным металлургическим предприятием // Сб. науч. тр. Донбасс. гос. тех. ун-та. 2008. № 27. С. 186–191.
2. Плеханов С. В. Интеграция разнородных баз данных на основе многомерных моделей данных: на примере интеграции геоинформационной системы с информационными системами предприятия : дис. ... канд. тех. наук. Уфа, 2006. 199 с.
3. Кресов А. А., Уваров В. В. Принципы интеграции данных в сфере недропользования // Вестник кибернетики. 2011. № 10. С. 83–89.
4. Подробий А. Н. Проблемы интеграции данных на современном предприятии // Вестн. Ульянов. гос. тех. ун-та. 2012. № 1. С. 64–66.
5. White C. Data Integration: Using ETL, EAI, and EII Tools to Create an Integrated Enterprise. BI Research TDWI Webcast, 2005. 34 p.
6. Дейт К. Дж. Введение в системы баз данных. 8-е изд. М. : ИД Вильямс, 2005. 1328 с.
7. Попова Н. А. Построение модели данных с применением тензорной методологии // Современные проблемы науки и образования. 2013. № 5. URL: http://www.science-education.ru (дата обращения: 07.06.2018).
8. Арменский А. Е. Тензорные методы построения информационных систем. М. : Наука, 1989. 152 с.
9. Kim M.. TensorDB and Tensor-Relational Model (TRM) for Efficient Tensor-Relational Operations : A Diss. Presented in Partial Fulfillment of the Requirement for the Degree Doctor of Philosophy. Arizona State University. 2014. 221 p.
10. Крон Г. Тензорный анализ сетей / пер. с англ. ; под ред. Л. Т. Кузина, П. Г. Кузнецова. М. : Сов. радио, 1978. 720 с.
11. Соколов Н. П. Пространственные матрицы и их приложен. М. : Гос. изд. физ.-мат. лит-ры, 1960. 300 с.
12. Кепещук Д. Б. Разработка модели обмена данными в гетерогенных распределенных базах данных // Вестник Тюмен. гос. ун-та. 2006. № 7. С. 81–86.