Технология формирования многомерных данных

© С.В. Зыкин © С.В. Мосин © А.Н. Полуянов
Институт математики им. С.Л. Соболева СО РАН,
Омск
szykin@mail.ru svmosin@gmail.com andrey.poluyanov@gmail.com

Аннотация

В статье рассмотрена проблема автоматизации формирования многомерного представления данных с использованием раздельного задания размерностей и мер. В качестве источника данных служит реляционная база данных. На исходную базу данных не накладываются какие-либо ограничения, например, требование иерархичности схемы, что позволяет применить эту технологию без модификации существующих баз данных и их приложений. Существенным отличием предложенной технологии является отказ от функциональной зависимости мер от размерностей. При этом усложняется проблема анализа корректности представления данных, однако, появляется возможность использования содержательных (не ключевых) атрибутов в качестве размерностей. Кроме того, в ячейке для мер может присутствовать несколько однородных значений (список), используемых при анализе данных. Для формально строгого изложения материала определена структура исходного и целевого представлений данных. Основная идея преобразований основана на последовательности межмодельных отображений. Корректность представления данных достигается за счет использования промежуточного представления, получившего название «Таблица соединений». Для анализа корректности построений на основе свойства соединения без потери информации и реализованных зависимостей вводится понятие и способ формирования контекстов приложения и ограничений на данные. Предлагаемая в данной статье технология является развитием традиционных OLAP-технологий.

1 Введение

Обработка и анализ накопленной информации является актуальной для многих исследовательских и прикладных задач. Наиболее популярным способом ее решения в настоящее время является технология оперативной аналитической обработки данных OLAP (online analytical processing) и Data Mining. Основой OLAP-технологии является построение гиперкубического (многомерного) представления данных. Проблема автоматизации анализа данных актуальна как для пользователей больших корпоративных информационных систем, так и для пользователей сравнительно небольших баз данных. Связано это с тем, что одни и те же данные приходится многократно реорганизовывать вручную, чтобы обработать их тем или иным алгоритмом поиска закономерностей.

В работах, посвященных OLAP, значительное внимание уделяется исследованию свойств моделей гиперкубов [5, 13, 15] и операциям их преобразования [8, 13] с целью получения представления, необходимого для анализа данных. Особое внимание уделяется построению иерархий в размерностях [4–6, 10, 13], что позволяет гарантировать корректность операций агрегации данных. В работах [5, 6, 10] рассматриваются нормальные формы для многомерных моделей данных, которые позволяют контролировать значения NULL в иерархиях размерностей.

В большинстве работ предполагается, что кубическое представление данных должно быть постоянно хранимым и периодически обновляемым из операционной базы данных (MOLAP). Основным аргументом в пользу такого дублирования данных выдвигается требование минимального времени отклика системы на запросы пользователя. При этом предполагается, что на сформированном представлении будут удовлетворены все потребности пользователя в анализе данных. Другой подход заключается в динамическом формировании многомерных данных, однако, при этом требуется преобразование схемы исходной операционной базы данных в «звезду» или «снежинку» (ROLAP), что нарушает принцип независимости данных, в частности независимость схемы операционной базы данных от места и способа использования данных. Общий недостаток этих двух подходов в регламентированности предполагаемых операций анализа данных.

В данной статье предполагается, что основой аналитической работы пользователя является

Труды 16-й Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» — RCDL-2014, Дубна, Россия, 13-16 октября 2014 г.

необходимость формирования новых гиперкубов из исходного реляционного представления данных. Это необходимо при выявлении скрытых закономерностей в данных и проведения анализа данных, не предусмотренного при проектировании хранилищ данных. Следовательно, основное внимание необходимо акцентировать на сокращении времени формирования схемы и представления нового гиперкуба. Формирование представления гиперкуба должно быть выполнено автоматически алгоритмами в соответствии с выбранной схемой.

Далее будем предполагать следующую технологическую последовательность обработки данных:

- Исходные данные должны быть представлены в реляционном нормализованном виде [9,14], и к ним обеспечивается доступ по технологии OLTP (online transaction processing);
- Пользовательское представление данных в виде гиперкубов, реализующее технологию OLAP, обеспечивается инструментарием, преобразующим исходные данные в необходимый на данный момент гиперкуб.
- Гиперкубическое представление данных далее используется в специализированном пакете программ для классификации, кластеризации, прогнозирования и т.д.

Существенные затраты времени для установления соответствия между реализацией базы данных (БД) и реализацией гиперкуба в данной работе предлагается сократить за счет автоматизации этого процесса с использованием свойств схемы исходной операционной базы данных.

В работе [16] для автоматизации формирования гиперкуба предлагается использовать последовательность преобразований:

$$\boldsymbol{RRD} \Rightarrow \boldsymbol{TJ} \Rightarrow \boldsymbol{GC},$$

где ***RRD*** – реляционное представление данных, ***TJ*** – таблица соединений, ***GC*** – гиперкуб (далее соответствующие модели будут формально определены). В данном случае ***RRD*** – представление исходной модели данных, ***GC*** – целевой. Наличие промежуточного представления ***TJ*** позволяет динамически управлять содержимым гиперкуба за счет определения контекстных ограничений и логических ограничений на данные.

В данной работе предусматривается возможность формирования списка однородных значений в отдельной ячейке гиперкуба без их агрегирования, поскольку списки таких значений используются во многих алгоритмах анализа данных.

Рассмотрим формализацию задачи. Пусть задана схема базы данных $\mathfrak{R} = \{R_1, R_2, ..., R_k\}$, полученная в результате нормализации отношений [9, 14]. Отношения R_i определены на множестве атрибутов $U = \{A_1, A_2, ..., A_n\}$. Пусть $[R_i]$ – схема отношения, множество атрибутов, на которых определено отношение R_i. Предположим, что схема $\mathfrak{R}$ является редуцированной [9], то есть, не существует двух отношений таких, что $[R_i] \subseteq [R_j]$, при $i \neq j$. Кортеж $t[X]$ – совокупность значений атрибутов $A_j \in X \subseteq [R_i]$, заданных в кортеже $t \in R_i$. Доопределим результат операции сравнения: неопределенное значение NULL атрибута A_j в кортеже t: $t[A_j]$=NULL, не равно любому другому значению, в том числе другому неопределенному значению (значение UNKNOWN становится эквивалентным значению FALSE).

Многомерное представление будем задавать в виде совокупности размерностей: $\{D_1, D_2, \ldots, D_d\}$, где D_l – множество расширенных имен атрибутов: $R_i.A_j$, $A_j \in [R_i]$, M – множество мер, также заданных в виде расширенных имен атрибутов Значения D_l являются значениями координат гиперкуба, значения M будут располагаться в рабочей области гиперкуба. Для каждой размерности задается ограничение в виде логической формулы F_l, которая формируется из элементарных условий, связанных операциями конъюнкции, дизъюнкции и отрицания. Элементарные условия заданы на атрибутах отношений: $R_i.A_j \theta R_{i'}.A_{j'}$ либо $R_i.A_j \theta$<const>, $\theta \in \{\geq, \leq, >, <, =, \neq\}$, <const> – константа либо неопределенное значение, задаваемое пользователем при активизации приложения. Причем, атрибуты в выражении F_l могут не принадлежать ни размерностям, ни мерам, а ограничение будет задаваться опосредованно, за счет контекстов. Способ формирования множеств отношений, из которых будут получены значения размерностей и мер гиперкуба, обсуждается далее.

В данной работе предлагается отказаться от необходимости выполнения функциональной зависимости [5, 6, 10, 16]:

$$D_1 D_2 \ldots D_d \rightarrow M, \qquad (1.1)$$

что позволит использовать содержательные (не ключевые) атрибуты в размерностях и иметь в одной ячейке гиперкуба несколько значений (список) атрибута $R_i.A_j \in M$. Списки значений используются в анализе данных, когда значения параметров не надо соотносить с объектами. Рассмотрим примеры представлений данных, для которых такое предположение является актуальным.

Пример 1. Рассмотрим фрагмент учебного плана в вузе. Задано множество атрибутов: A_1 – Специальность, A_2 – Предмет, A_3 – Семестр, A_4 – Количество часов, A_5 – Вид занятия, A_6 – Контроль успеваемости, A_7 – Номер специальности, A_8 – Номер предмета. На предложенном множестве атрибутов существуют следующие зависимости:

$$DEP = \{A_7 \rightarrow A_1, A_8 \rightarrow A_2, A_7A_8A_3A_5 \rightarrow A_4, A_7A_8A_3 \twoheadrightarrow A_5(A_6)\}$$

(три функциональных и одна многозначная). По правилам построения нормальных форм [9, 14] будет получена следующая схема базы данных: Специальности = $R_1(\underline{A_7}, A_1)$, Предметы = $R_2(\underline{A_8}, A_2)$, Учебная нагрузка = $R_3(\underline{A_7}, \underline{A_8}, \underline{A_3}, \underline{A_5}, A_4)$, Контроль = $R_4(\underline{A_7}, \underline{A_8}, \underline{A_3}, \underline{A_6})$, где подчеркнуты ключевые атрибуты отношений. Одно из возможных представлений гиперкуба приведено в таблице 1.

Таблица 1. Фрагмент учебного плана в вузе

Семестр	2							
Предмет	Математика				Физика			
Вид занятия	Лекции	Практика	Лаб. раб.	*Контроль успеваемости*	Лекции	Практика	Лаб. раб.	*Контроль успеваемости*
Специальность	*Количество часов*	*Количество часов*	*Количество часов*		*Количество часов*	*Количество часов*	*Количество часов*	
История	36	36	18	зач., экз.	18	18	18	зач.
Филология	18	18	18	зач.	36	36	18	зач., экз.
Правоведение	48	48	24	зач., экз.	48	48	24	зач., экз.

В таблице 1 атрибуты размерностей представлены жирным шрифтом, атрибуты мер – курсивом, значения атрибутов – обычным шрифтом.

Схема гиперкуба в таблице 1 может быть представлена в следующем виде:

$$\{R_1.A_1\}\times\{R_3.A_3\{R_2.A_2\{R_3.A_5(R_3.A_4)\}(R_4.A_6)\}\}, \quad (1.2)$$

где $D_1=\{R_1.A_1\}$ и $D_2=\{R_3.A_3,\ R_2.A_2,\ R_3.A_5\}$ – размерности, $M=\{R_3.A_4,\ R_4.A_6\}$ – меры. Логическое ограничение:

$F=(R_4.A_3=2)\wedge(R_2.A_2=$ 'Математика'$\vee R_2.A_2=$'Физика'$))$.

В литературе, посвященной моделированию гиперкубов [5, 6, 10, 16], меры рассматриваются отдельно от размерностей, поскольку предполагается, что иерархии в каждой размерности имеют по одному терминальному уровню. В таблице 1 размерность D_2 имеет два терминальных уровня $R_2.A_2$ и $R_3.A_5$, так как каждый из них имеет собственную сопоставленную меру, хотя между ними установлено иерархическое подчинение. Если бы меры не были сопоставлены размерностям, то возникает неоднозначность в расположении значений мер. Таким образом, схема (1.2) задает иерархию для размерности D_2, в которой терминальными уровнями фактически является меры. Кроме того, меры должны быть сопоставлены только атрибутам одной размерности, поскольку в противном случае в одной ячейке таблицы надо было бы сопоставить разнородные значения мер, либо дополнительно дробить ячейки в рабочей области гиперкуба.

Таблица 1 может иметь другую схему, например:

$$\{R_1.A_1\{R_2.A_2\{R_3.A_5(R_3.A_4)\}(R_4.A_6)\}\}\times\{R_3.A_3\}.$$

Структура таблицы при этом изменится, а содержание останется прежним.

Пример 2. Рассмотрим фрагмент базы данных лечебного заведения. Задано множество атрибутов: A_1 – № пациента, A_2 – ФИО пациента, A_3 – № показателя, A_4 – Показатель, A_5 – Значение показателя, A_6 – № дня получения показателя, A_7 – Группа пациентов. На предложенном множестве атрибутов существуют следующие зависимости: $DEP=\{A_1\rightarrow A_2A_7,\ A_3\rightarrow A_4,\ A_1A_3A_6\rightarrow A_5\}$. По правилам построения нормальных форм будет получена следующая схема базы данных: Пациенты = $R_1(\underline{A_1}, A_2, A_7)$, Перечень анализов = $R_2(\underline{A_3}, A_4)$, Результаты анализов = $R_3(\underline{A_1}, \underline{A_3}, \underline{A_6}, A_5)$. Одно из возможных представлений гиперкуба приведено в таблице 2.

Схема гиперкуба в таблице 2 имеет вид:

$$\{R_3.A_6\}\times\{R_2.A_4\{R_1.A_7(R_3.A_5)\}\}, \quad (1.3)$$

где $D_1=\{R_3.A_6\}$ и $D_2=\{R_2.A_4,\ R_1.A_7\}$ – размерности, $M=\{R_3.A_5\}$ – мера. Логическое ограничение: $F=(R_2.A_4=$'Креатинин'$\vee R_2.A_2=$'Белок'$\vee R_2.A_2=$'Билирубин'$)\wedge(R_1.A_7=2\vee R_1.A_7=3)$. Списки значений в таблице 2 удобно использовать для анализа данных с использованием критерия Уилкоксона, алгоритмами дискриминантного анализа и т.д. В таблице 2 значения показателей могут совпадать друг с другом. Это не означает, что они дублируются при формировании гиперкуба, эти значения получены у разных пациентов. Следовательно, при формировании гиперкуба необходимо различать дублированные значения мер, полученные вследствие способа формирования представления, от совпадающих значений мер, соответствующих различным объектам базы данных.

При формировании представлений в примерах 1 и 2 использовались так называемые контексты и таблица соединения, определение и способ формирования которых рассматриваются далее. В данной статье предлагается не ограничивать пользователя в выборе структуры гиперкуба, а только подсказывать ему корректные решения по формированию гиперкуба при заданных мерах и размерностях.

2 Технология формирования многомерных данных

Для автоматизации построения представления многомерных данных предлагается следующая детализация последовательности их формирования:

1. Пользователь из списка атрибутов БД формирует множества атрибутов размерностей D_1, D_2, …, D_d и меры M. Естественными являются ограничения: $D_i \cap D_j = \varnothing$, $i\neq j$, $D_i \cap M = \varnothing$, $i, j = 1, 2, \ldots, d$. Дополнительным технологическим ограничением является запрет на использование атрибута в качестве меры, если на него установлено ограничение в логическом выражении. Основанием для такого ограничения является возможность наличия значений размерностей в представлении и отсутствия соответствующих значений мер, хотя они есть в БД. Это может служить причиной для неверной интерпретации результатов.

Таблица 2. Сводная таблица анализов пациентов

Показатель	Креатинин		Белок		Билирубин	
Группа пациентов	2	3	2	3	2	3
№ дня получения показателя	*Значение показателя*	*Значение показателя*	*Значение показателя*	*Значение показателя*	*Значение показателя*	*Значение показателя*
1	61, 97, 78, 101 ...	64, 104, 69, 49 ...	82, 70, 67, 69 ...	70, 64, 80, 74 ...	14.2, 17.8, 18.84, 44.3 ...	19.5, 16.8, 8.6, 19.5 ...
2	63, 102, 83, 113 ...	71, 108, 71, 32 ...	64, 58, 68, 61 ...	55, 57, 54, 62 ...	34.7, 15.4, 96.5, 64.9 ...	36.8, 19.5, 32.4, 73.9 ...
3	59, 59, 87, 79 ...	71, 110, 75, 51 ...	68, 62, 58, 59 ...	55, 65, 70, 65 ...	19.5, 17.8, 83.78, 114.3 ...	24.9, 12.3, 15.8, 30.3 ...

2. Формирование иерархий размерностей для множеств атрибутов D_1, D_2, …, D_d. Иерархии формируются автоматически по правилам, рассмотренным в работе [17], и при желании пользователь может их модифицировать. Основная идея подхода заключается в размещении атрибутов с меньшим набором допустимых значений на более высоких уровнях иерархии, что уменьшает количество дублированных значений на нижних уровнях иерархии и, как следствие, уменьшает количество пустых ячеек для мер в рабочей области гиперкуба. При этом анализируются функциональные, многозначные зависимости между атрибутами размерности, появляющиеся циклы при построении иерархии удаляются за счет анализа полустепени исхода для каждой вершины цикла.

3. По шаблону, соответствующему дизъюнктивной нормальной форме, задаются логические ограничения на размерности F_1, F_2, …, F_d. где логическая формула F_i задана на атрибутах размерности D_i. По умолчанию каждая формула есть конъюнкция условий определенности (*IS NOT NULL*) для атрибутов размерности.

4. Формирование контекстов размерностей C_1, C_2, …, C_d. (некоторые контексты могут быть пустыми, а некоторые – псевдоконтекстами). Далее будут представлены соответствующие определения и алгоритмы формирования контекстов.

5. Формирование контекста приложения C и соответствующей реализации в виде таблицы соединений TJ. Кортежи реализации C должны содержать значения атрибутов размерностей и *сопоставленные* им значения мер, следовательно, контекст C должен содержать контексты размерностей C_1, C_2, …, C_d. С учетом сказанного логическое ограничение на кортежи $t \in TJ$ имеет следующий вид:

$$F(t)=F_1(t) \wedge F_2(t) \wedge \ldots \wedge F_d(t).$$

6. Формирование реализаций размерностей TJ_1, TJ_2, …, TJ_d с сортировкой значений в соответствии с иерархией. Если контекст размерности не пуст, то он используется для формирования TJ_i, в противном случае реализация размерности является проекцией TJ.

7. Формирование реализации (представления) композиционной таблицы (заполнение значений мер на соответствующих местах таблицы).

Пользователь вручную выполняет шаги 1 и 3 и осуществляет выбор предложенных вариантов в шагах 2, 4 и 5. Все остальные операции выполняются автоматически.

Заметим, что в предложенной последовательности шагов формирования композиционной таблицы исключается необходимость каким-либо образом модифицировать исходную операционную БД, что делает возможным реализовать все принципы проектирования БД [9, 14], в том числе самый важный – принцип независимости данных.

В работах, посвященных построению гиперкубического представления данных, в качестве промежуточных моделей используются SQL-таблицы. Интерфейс между БД и хранилищем данных программируется. Предлагаемый в данной работе подход исключает затраты на программирование.

В предлагаемой статье рассмотрена общая постановка задачи, включающая шаги от построения схемы композиционной таблицы до ее реализации. Особое внимание уделено правилам и алгоритмам формирования контекстов.

3 Контексты

3.1 Свойства контекстов

Пусть DEP – множество зависимостей (функциональных, многозначных, включения, соединения), определенных на множестве атрибутов U и множестве отношений $\mathfrak{R}$. Пусть R – отношение, определенное на множестве атрибутов U (универсальное реляционное отношение). Рассмотрим классические определения зависимостей [9,14].

Определение 3.1 (ФЗ). Пусть X и Y – некоторые подмножества из множества атрибутов U. Будем говорить, что X функционально определяет Y: $X \to Y$, если в любой реализации R не могут присутствовать два кортежа $t,u \in R$, такие что $t[X]=u[X]$ и $t[Y] \neq u[Y]$.

Пусть заданы множества атрибутов $X \subseteq U$, $Y \subseteq U$ и $X \cap Y = \varnothing$, $Z=[R] \setminus (X \cup Y)$.

Определение 3.2 (МЗ). Множество X мультиопределяет множество Y в контексте Z: $X \twoheadrightarrow Y(Z)$ (многозначная зависимость), если для

произвольной реализации R существует два кортежа $t_1,t_2 \in R$ таких, что $t_1[X]=t_2[X]$, то существует кортеж t_3, для которого выполнено:

$$t_3[X]=t_1[X],\ t_3[Y]=t_1[Y],\ t_3[Z]=t_2[Z],$$

в силу симметрии существует кортеж t_4:

$$t_4[X]=t_1[X],\ t_4[Y]=t_2[Y],\ t_4[Z]=t_1[Z].$$

Определение 3.3 (ЗС). Отношение $R(V_1, V_2, ..., V_p)$ удовлетворяет зависимости соединения $*(V_1, V_2, ..., V_p)$ тогда и только тогда, когда R удовлетворяет свойству соединения без потерь информации (СБПИ):

$$R=R[V_1]\bowtie R[V_2]\bowtie\ldots\bowtie R[V_p],$$

где $\bowtie$ – операция естественного соединения, $R[V_i]$ – проекция отношения R по атрибутам V_i.

Заметим, что многозначная зависимость является частным случаем зависимости соединения, а функциональная зависимость является частным случаем многозначной зависимости [9,14].

Формальным основанием для установления связей на схеме базы данных являются зависимости включения [2]:

Определение 3.4 (ЗВ). Пусть $R_i[A_1, \ldots, A_m]$ и $R_j[B_1, ..., B_p]$ – схемы отношений (не обязательно различные), $V \subseteq \{A_1, \ldots, A_m\}$ и $W \subseteq \{B_1, ..., B_p\}$, $|V|=|W|$, тогда объект $R_i[V] \subseteq R_j[W]$ называется зависимостью включения, если $R_i[V] \subseteq R_j[W]$, где $|V|$ – мощность множества V.

Если выполнено условие $V=W$, то такой вид ЗВ называется типизированными (typed) [7,12]. Это дополнительное ограничение вполне согласуется с общепринятым свойством связей на схеме БД: связи отражают количественное соотнесение кортежей в отношениях, и не обладают какой-либо семантикой. Необходимость связывания различных по смыслу атрибутов, скорее всего, является признаком потери какой-либо ФЗ для связываемых атрибутов, либо, как в примере 1.1 [7], оставшаяся ЗВ после удаления избыточных зависимостей установлена для атрибутов-синонимов.

Пусть $C=\{R_1, R_2, ..., R_m\}$ – произвольное подмножество отношений реляционной БД.

Определение 3.5. Зависимость $dep_j \in DEP$ будем считать реализованной на C, если операция дополнения, удаления или модификации кортежа в произвольном отношении $R_i \in C$ будет заблокирована организационно-техническими средствами, если при этом нарушается зависимость dep_j,.

Под организационными средствами подразумевается способ проектирования схемы БД с указанием ограничений целостности на данные, под техническими – возможности системы управления базами данных (СУБД) по поддержке этих ограничений целостности.

Определение 3.6. Множество C будем называть контекстом, если оно удовлетворяет свойству СБПИ на зависимостях DEP, реализованных в C.

Замечание. В основе контекста лежит операция естественного соединения, которая собирает из различных отношений БД связанные друг с другом по значению данные. Затем эти данные (кортежи) участвуют в формировании новых структур, естественным образом дополняя и ограничивая друг друга, что делает уместным использования термина «контекст» для совокупности таких значений.

В существующих OLAP-системах, в том числе в "Microsoft Analysis Services" и "ORACLE Analytic Workspace Manager", свойство СБПИ не анализируется, что является основной причиной дополнительных ограничений при формировании представлений гиперкубов: иерархическая схема БД, в которой мерами могут быть только числовые атрибуты корня, размерности формируются из атрибутов подчиненных вершин.

Алгоритм проверки свойства СБПИ [14] является полиномиальным, но все равно довольно затратным по времени и по памяти для больших схем БД. Рассмотрим вспомогательные свойства, которые позволят улучшить эти характеристики.

Теорема 3.1. Множество отношений C обладает свойством СБПИ, если существует отношение $R_i \in C$, замыкание первичного ключа которого совпадает со всем множеством атрибутов отношений множества C.

Пусть $C_m=\{R_1, R_2, ..., R_m\}$ – произвольное множество отношений и $[C_m]=[R_1]\cup[R_2]\cup ... \cup[R_m]$.

Теорема 3.2. Множество отношений $C_{m+1}=\{R_1, R_2, ..., R_m, R_{m+1}\}$ не обладает свойством СБПИ на DEP, если зависимость $Z \twoheadrightarrow X(Y)$ не выводима из DEP, где $X \subseteq [C_m]$, $Y \subseteq [R_{m+1}]$ и $[C_m] \cap [R_{m+1}] \subseteq Z$.

Определение 3.7. (Существующее соединение): Выражение $R_1 \bowtie R_2 \bowtie \ldots \bowtie R_m$ будем называть существующим соединением, если для совокупности отношений R_i, $i = 1, \ldots, m$, существует хотя бы одна перестановка $V_1, V_2, \ldots, V_m$ отношений $R_1, R_2, \ldots, R_m$ такая, что $([V_1] \cup [V_2] \cup \ldots \cup [V_j]) \cap [V_{j+1}] \neq \varnothing$, $j=1,\ldots,m-1$.

Замечание: Предложенное в определении ограничение на схемы $R_1, R_2, \ldots, R_m$ является более слабым, чем ограничение квазистягиваемости пересечений [9], которое используется для формирования полусоединений.

Теорема 3.3. Если множество отношений $C=\{R_1, R_2,\ldots, R_m\}$ не образует существующее соединение, то оно не обладает свойством СБПИ на множестве функциональных зависимостей FD.

Проверки условий, предложенных в теоремах 3.1 – 3.3 являются менее затратными по памяти и по времени (линейная зависимость количества итераций от количества зависимостей или

отношений), чем проверка свойства СБПИ по алгоритму [14] (кубическая сложность, оценка приведена ниже). Осталось разобраться со сложностью алгоритма проверки свойства существующего соединения.

Рассмотрим алгоритм проверки существования соединения и формализуем его с использованием псевдокода, что необходимо для последующего доказательства его корректности.

Вход: множество отношений $C=\{R_{mas[1]}, R_{mas[2]},\ldots, R_{mas[k]}\}$, ***mas*** – массив с номерами отношений, для которых осуществляется проверка существования соединения. Считаем, что элементы массива нумеруются с ***1***, ***k*** – число элементов массива.

Выход: ***flag_exist = 1*** – если соединение существует, ***flag_exist = 0*** – если соединение не существует.

```
gran=2
for i=1 to k-1
  for j=gran to k
    if ((R_mas[i] ∩ R_mas[j]) ≠ ∅) then
      temp=mas[gran]
      mas[gran]=mas[j]
      mas[j]=temp
      gran=gran+1
    endif
  endfor
  if gran=k+1 then
    flag_exist=1
    exit for
  endif
  if gran=i+1 then
    flag_exist=0
    exit for
  endif
endfor
```

Теорема 3.4. На выходе алгоритма ***flag_exist = 1*** тогда и только тогда, когда множество отношений $C=\{R_{mas[1]}, R_{mas[2]},\ldots, R_{mas[k]}\}$ образует существующее соединение.

Следствие: Если существует одна перестановка из определения, то существует как минимум ***k*** различных перестановок. Это следует из того, что перестановка может начинаться с любого из ***k*** отношений.

С формальной точки зрения разработанный алгоритм является аналогом алгоритма поиска в ширину на графах [1]. Однако алгоритм проверки существования соединения отношений существенно отличается от алгоритма поиска в ширину тем, что на вход алгоритма подается не полностью построенный граф, а только его вершины, при работе алгоритма строятся только те ребра, которые необходимы для проверки свойства существования соединения отношений.

Анализ рассмотренного алгоритма показал, что самая трудоемкая операция: проверка пересечений. Она в алгоритме выполняется ***k(k-1)/2*** раз. Аналогичные по сложности операции при проверке свойства СБПИ выполняются примерно ***nk(1+k)*** раз, где ***n*** – общее количество атрибутов в схеме базы данных. При этом учитывалось, что минимальное покрытие множества функциональных зависимостей [14] по мощности примерно равно количеству отношений. При проверке свойства СБПИ могут использоваться многозначные зависимости и зависимости соединений. Поскольку ***n*** не менее чем на порядок больше чем ***k*** и примерно 70 процентов комбинаций отношений на схеме БД не образуют существующее соединение, то применение рассмотренного алгоритма перед проверкой свойства СБПИ дает существенный выигрыш в результирующем количестве операций.

3.2 Формирование контекстов

Первоначальный выбор размерностей и мер гиперкуба предлагается сделать в расширенном виде: $R_i.A_j$, где R_i – наименование отношения из исходной реляционной БД, и A_j – наименование атрибута в этом отношении. Таким образом будет задано начальное множество отношений $C^0=\{R^0_1, R^0_2, \ldots, R^0_q\}$, участвующее в обязательном порядке сначала в формировании таблицы соединения, а потом – гиперкуба.

Совокупность отношений, по которым строится гиперкуб, должна удовлетворять свойству СБПИ [11], поскольку лишние кортежи в промежуточном представлении данных дают лишние значения в рабочей области гиперкуба. Следовательно, дальнейшая задача состоит в дополнении множества C^0 отношениями из $\mathfrak{R}$, чтобы результирующее множество отношений удовлетворяло свойству СБПИ на множестве зависимостей, то есть являлось контекстом. В общем случае таких вариантов дополнения существует несколько. Каждый из вариантов (контекстов) имеет свою смысловую нагрузку, поэтому окончательный выбор контекста может выполнить только пользователь. Задача алгоритма заключается в последовательной генерации контекстов без зацикливания. Для сокращения количества перебираемых вариантов при формировании контекстов, ближайших к множеству R^0, предлагается сделать этот перебор направленным.

Сформулируем критерии, которые позволят сделать перебор отношений направленным.

- Замыкание первичного ключа нового отношения R_i совпадает со всем множеством атрибутов в выбранных отношениях. Дополнение этого отношения к C^0 гарантирует выполнение свойства СБПИ по теореме 3.1 Такое отношение получает приоритет 3.

- Для отношения R_i выполнено условие существования связи, соответствующей ЗВ $R_i[X] \subseteq R_j[X]$ с уже выбранными отношениями R_j, где множество атрибутов X является первичным ключом отношения R_j. Такое отношение получает приоритет 2, поскольку высока вероятность выполнения свойства СБПИ для результирующего множества отношений.
- Если дополняемое отношение R_i не удовлетворяет условиям теорем 3.2 и 3.3, то такое отношение получает приоритет 1. Остальные отношения получают приоритет 0.
- Формируемый контекст не должен содержать лишних отношений, наличие которых обусловлено только порядком присоединения отношений к контексту в алгоритме.

Перечисленные критерии увеличивают вероятность более быстрого достижения результата.

Введем обозначения. Пусть $C^1=\{R^1_1, R^1_2, ..., R^1_p\}$ – множество отношений не входящих в исходное множество R^0: $R^1=\Re \setminus R^0$. U^0 – множество атрибутов, на котором определены отношения из R^0: $U^0=[R^0_1] \cup [R^0_2] \cup \ldots \cup [R^0_q]$. DEP^0 – множество зависимостей, реализованных на отношениях из R^0.

Рассмотрим алгоритм, удовлетворяющий сформулированным критериям. Для изложения этого и последующих алгоритмов будем ограничиваться схематическим описанием, если не требуется обоснование корректности и реализация на псевдокоде займет много места.

Алгоритм формирования контекстов.

- Подсчет весов для отношений R^1, и их упорядочение по убыванию весов.
- Формируются сочетания без повторений из отношений R^1, сначала по одному, затем по два и так далее. Сочетания начинаются с наименьших значений, и далее последовательно увеличиваются, например сочетания по два элемента: ***(1,2)***, ***(1,3)***,..., ***(2,3)***.... Текущее сочетание отношений совместно с R^0 проверяем при необходимости на выполнение свойства СБПИ. Если свойство выполнено, то полученное сочетание дополняется к множеству контекстов.
- В процессе выполнения алгоритма пользователю предлагается выбрать нужный контекст приложения.

Замечание. Такая схема алгоритма на ближайших итерациях находит дополнительные отношения с наибольшей вероятностью образующие наименьший контекст с исходным множеством отношений R^0. В примерах 1 и 2 все отношения исходных баз данных являются контекстами приложений. Не трудно убедиться, что они удовлетворяют свойству СБПИ.

Далее контекст приложения будем обозначать C_0.

В примерах 1 и 2 встречаются ячейки с несколькими значениями (списком). Возникает закономерный вопрос: если в одном списке присутствуют два и более совпадающих значения, то дублируют они друг друга или нет. Ответ следующий: если эти значения соответствуют одному и тому же объекту и одному и тому же параметру, то это дублирование. В результирующем представлении гиперкуба ***GC*** идентификаторы объектов отсутствуют, однако, они есть в промежуточном представлении данных – таблице соединений ***TJ***. Для того, чтобы в дальнейшем иметь возможность определять дублированные значения, введем понятие ключа меры.

Определение 3.8. Множество атрибутов KM_j будем называть ключом атрибута меры $R_i.A_j \in M$ в таблице соединений ***TJ***, если $KM_j \subseteq [TJ]$, зависимость $KM_j \rightarrow R_i.A_j$ выводима на множестве функциональных зависимостей, и не существует выводимой зависимости $Y \rightarrow R_i.A_j$, где $Y \subset KM_j$. Пусть $KM=KM_1 \cup KM_2 \cup \ldots \cup KM_h$, где $h=|M|$, общий ключ для всех мер гиперкуба.

После того как сформирована схема ***GC*** с установлением иерархий в размерностях и присоединением мер к атрибутам одной из размерностей (в примерах 1 и 2 меры присоединены к вертикальным размерностям), простейшим решением задачи формирования гиперкуба по контексту приложения $C_0=\{R^0_1, R^0_2, ..., R^0_m\}$ $(m \geq q)$ при $F=\varnothing$ является выполнение операции естественного соединения отношений для формирования промежуточного представления ***TJ***:

$$TJ=R^0_1[V_1] \bowtie R^0_2[V_2] \bowtie \ldots \bowtie R^0_m[V_m], \qquad (3.1)$$

где V_i – множество атрибутов $A_j \in [R^0_i]$, для которых выполнено: либо существует размерность D_l такая, что $R^0_i.A_j \in D_l$, либо $R^0_i.A_j \in M$, либо $R^0_i.A_j \in KM$, либо $R^0_i.A_j$ атрибут, который участвует в операции естественного соединения с другими отношениями контекста (определяется зависимостями соединения). Далее значения координат формируются в виде проекций по соответствующим атрибутам: $TJ[D_l]$, с необходимой сортировкой кортежей каждой размерности в соответствии с иерархией. Завершается построение ***GC*** присваиванием значений мер M в рабочей области ***GC***: для каждого кортежа $t \in TJ$ на пересечении значений координат $t[D_l]$, $l=1,\ldots,d$, ставится значение $t[A_j]$, $R^0_i.A_j \in M$ (в данном случае R^0_i будет произвольным). Во всех остальных ячейках ***GC*** ставится значение NULL. Проблема дублированных значений в ячейках гиперкуба и остальные детали реализации технологии будут далее рассмотрены.

Предложенная процедура формирования ***GC*** не решает следующие проблемы:

- Если какому-либо набору значений размерности не соответствует ни одного значения меры, то эти значения размерностей не появятся в реализации гиперкуба (по свойству операции естественного соединения). Однако отсутствие

значений мер также является предметом анализа данных.

- Ограничения на значения размерностей могут быть заданы опосредованно – через значения на связанные кортежи в отношениях, которые не входят в контекст приложения. Тогда эти отношения должны образовывать отдельный контекст с отношениями для размерностей.
- Для некоторых размерностей необходимо иметь декартово произведение исходных отношений (все возможные комбинации атрибутов одной размерности). Тогда эти отношения не должны дополняться другими отношениями для получения контекста.

3.3 Управление размерностями

Резюмируя сформулированные требования к представлению размерностей в ***GC*** получим, что в нем должны быть представлены следующие компоненты:

- контексты размерностей,
- псевдоконтексты.

Рассмотрим эти компоненты.

1. Отдельные контексты формируются для размерностей, которые должны быть связаны по значениям с множеством отношений, не совпадающим с C_0. Допустим, в примере 1 не для всех предметов заполнены сведения об учебной нагрузке, но в представлении требуется наличие всех предметов. Тогда размерность $D_1=\{R_1.A_1\}$ должна быть сформирована по отдельному контексту $\{R_1\}$. В другом случае на размерность накладывается ограничение в виде логической формулы, в которую входят атрибуты отношений, не принадлежащих контексту C_0. В этом случае совокупность отношений, атрибуты которых выбраны в качестве компонентов размерности и компонентов логической формулы, дополняется новыми отношениями до необходимого контекста по алгоритму, схема которого предложена в предыдущем разделе.

2. Псевдоконтексты размерностей формируются аналогичным способом, за исключением того, что ограничения могут задаваться только на атрибутах уже выбранных отношений и это множество отношений не дополняется другими отношениями до контекста. В примерах 1 и 2 нет размерностей, сформированных по псевдоконтекстам. Однако, в примере 1 альтернативное представление данных со схемой

$$\{R_1.A_1\{R_2.A_2\}\}\times\{R_3.A_5(R_3.A_4)\}$$

имеет размерность, сформированную по псевдоконтексту из отношений R_1 и R_2.

Если какая-либо размерность должна являться частью контекста приложения, то для этой размерности отдельный контекст не формируется. Такие контексты будем называть пустыми. В представлении ***TJ***, сформированном по формуле (3.1), все размерности являются частью контекста приложения.

Заметим, что для контекста приложения не целесообразно задавать отдельное ограничение, поскольку оно будет фактически являться ограничением на меры. Тогда в результирующем представлении будет содержаться пустая ячейка, либо ячейка с неполным списком значений, что может быть неправильно воспринято пользователем: значения размерностей присутствуют в представлении, а соответствующего значения меры нет. Тогда как это значение есть в БД, но оно "отфильтровано" ограничением.

Логическая формула для всего представления данных будет иметь вид: $F=F_1\wedge F_2\wedge\ldots\wedge F_d$, где формула F_l соответствует размерности D_l. Если для какой-либо размерности логическая формула не задана, то вместо нее в общую формулу подставляется значение *TRUE*.

Сформированное множество всех контекстов и псевдоконтекстов обозначим $C=\{C_0, C_1, \ldots, C_d\}$. Заметим, что дополнительные контексты и псевдоконтексты необходимы, прежде всего, для управления содержимым размерностей ***GC***.

4 Формирование гиперкубического представления

Рассмотрим алгоритм формирования гиперкуба ***GC***. Пусть размерности упорядочены так, что размерность D_s соответствует контексту C_s, $s=1, 2,\ldots, d$.

Шаги $s=1, 2,\ldots, d$. На каждом шаге формируется представление размерности D'_s, для которых имеются отдельные контексты:

$$TJ_s=\sigma_{Fs}(R^s_1[V^s_1]\bowtie R^s_2[V^s_2]\bowtie\ldots\bowtie R^s_{m(s)}[V^s_{m(s)}])[D_s],$$

где $C_s=\{R^s_1, R^s_2,\ldots, R^s_{m(s)}\}$, атрибуты V^s_i отношений R^s_i выбираются по правилам, аналогичным формуле (3.1), $m(s)$ – элемент массива, содержащего количество отношений в каждом контексте.

Замечание. Каждое представление TJ_s целесообразно сразу преобразовать в иерархическое представление в соответствии с заданной схемой. В каждом узле иерархии будут содержаться значения атрибута текущего уровня, имеющие одно общее значение атрибута предыдущего уровня. Значения внутри узла иерархии сортируются и каждое из них, кроме листьев, имеет указатели на узлы – потомки (потомков у одного значения может быть несколько). Значения атрибутов в листьях снабжаются пустыми указателями переменной длины, которые позднее будут заполнены ссылками на ячейки со списками значений атрибутов мер. При формировании иерархии размерности неопределенные значения атрибутов будем считать совпадающими, Это позволит объединить значения атрибутов мер, если соответствующие им значения размерностей являются неопределенными. Например, в иерархии адресов {Страна{Область{Город{Район{Населенный пункт}}}}}

для жителей города атрибут 'Населенный пункт' будет неопределен, а для сельских жителей неопределенным будет атрибут 'Город'. Если считать неопределенные значения атрибутов различными, то значения атрибутов мер для жителей одного района города будут распределены по различным ячейкам, аналогичная ситуация будет и для жителей одного населенного пункта в сельской местности. Обе эти ситуации являются ошибкой.

Шаг ***d+1.*** На этом шаге формируется представление контекста приложения:

$$TJ=\sigma_F(R^0_1[V^0_1]\bowtie R^0_2[V^0_2]\bowtie\ldots\bowtie R^0_{m(0)}[V^0_{m(0)}]),$$

где $C_0=\{R^0_1, R^0_2,\ldots, R^0_{m(0)}\}$, атрибуты V^0_i отношений R^0_i выбираются по правилам, аналогичным формуле (3.1), $m(0)$ –количество отношений в контексте приложения ($m(0)=m$).

Замечание. Выражения F_l в формуле F, в которых есть неопределенные элементарные условия, заменяются значением *TRUE*. Следовательно, формула F задает более слабые ограничения, чем на предыдущих шагах, и в представлении TJ будет больше кортежей, чем это необходимо для сопоставления со значениями сформированных размерностей. Если $d=0$, то в представлении TJ нет лишних кортежей.

Шаг ***d+2.*** Для каждого кортежа $t\in TJ$ проверяется наличие соответствующих значений в иерархиях размерностей (идентифицирующий путь), что аналогично условию: существует кортеж $u\in TJ_s$ такой, что $t[D'_s]=u[D'_s]$, $s=1, 2,\ldots, d$. Если это условие выполнено, то кортеж t используется для формирования иерархий размерностей, для которых отсутствуют отдельные контексты. Если соответствующий путь отсутствует в иерархии, то он формируются по правилам, аналогичным предыдущим шагам.

Каждое значение атрибута $A_j\in M$ текущего кортежа t дополняется к соответствующей ячейке со списком, если оно не дублировано. Ячейка существует, если все идентифицированные пути в иерархиях имеют совпадающий указатель на нее. В противном случае создается новая ячейка, в нее дополняется значение атрибута A_j и каждому идентифицированному пути в иерархиях дополняется указатель на эту ячейку.

Конец алгоритма.

Для завершения обсуждения алгоритма необходимо определить признак дублирования значения атрибута меры в ячейке. Отражением семантики сформированного контекста приложения является следующее определение.

Определение 4.1. Значение атрибута меры $t[A_j]$, где $A_j\in M$, для текущего кортежа $t\in TJ$ дублирует значение $t'[A_j]$, $t'\in TJ$, если:

1) $t[A_j]=t'[A_j]$,

2) $t[D_s]=t'[D_s]$, $s=1, 2,\ldots, d$,

3) $t[KM_j]=t'[KM_j]$.

Замечание 1. Условия 1 и 2 определения 4.1 проверяются непосредственно по текущим значениям кортежа t, размерностей D_s и значениям атрибута A_j, уже записанных в текущую ячейку. Для проверки условия 3 необходимо знать значения ключевых атрибутов KM_j уже просмотренного кортежа t'. Наиболее экономичным и технологичным решением данной проблемы является временное хранение номера кортежа t' совместно со значением атрибута A_j в ячейке. Тогда условие 3 можно проверить, сравнивая два кортежа в представлении TJ. Альтернативным решением этой задачи является хранение номера кортежа исходного отношения, который использовался для формирования TJ, однако стандартный доступ к данным с использованием языка SQL такой возможности не предоставляет.

Замечание 2. Если в контексте приложения для атрибута меры A_j отсутствует ключ, то по аксиоме рефлексивности [14] ключом будет сам атрибут. Следовательно, в ячейках для этого атрибута не может быть совпадающих значений.

5 Заключение

Рассмотренная в данной работе технология является развитием технологии, предложенной в работе [3]. Предложенная модель многомерных данных является обобщением известных моделей, прежде всего за счет снятия ограничения (1.1). Технология ориентирована на работу аналитика, где не требуется быстрая (за доли секунд) реакция системы на запросы, поскольку в большинстве случаев аналитик должен вдумчиво выполнить различные виды анализа над различными представлениями данных с участием ИТ-специалиста. Технология предназначена, прежде всего, для ИТ-специалиста, чтобы он мог за приемлемое время подготовить для обработки необходимое представление данных. Хотя, после некоторого обучения, аналитик самостоятельно может освоить данную технологию.

Первая версия данной технологии была реализована в качестве приложения в среде Delphi с использованием библиотеки ADOdb (разработчик – Редреев П.Г., руководитель – Зыкин С.В.) и использовалась для обработки данных пациентов кардиологического диспансера. Существенным недостатком разработки было длительное время формирования итоговых таблиц. Причины оказались технологическими (неверный выбор среды разработки) и методологическими (использование алгоритмов полного перебора при формировании представлений). В результате была существенно переработана технология формирования многомерных данных и предполагается ее реализация с использованием промежуточных представлений.

Основным методологическим принципом в данной стать является то, что операционная база данных должна удовлетворять принципам независимости, неизбыточности, непротиворечивости и т.д. Эта база данных является ядром приложений для множества пользователей, а не только отдельно взятого аналитика.

Литература

[1] A.V. Aho, J.E. Hopcroft, J.D. Ullman. Data Structures and Algorithms // Addison-Wesley, 1983. – 427 p.

[2] M. Casanova, R. Fagin, C. Papadimitriou. Inclusion Dependencies and Their Interaction with Functional Dependencies // Journal of Computer and System Sciences. 1984. No. 28(1). P. 29–59.

[3] E.F. Codd, S.B. Codd, C.T. Salley. Providing OLAP (On-line Analytical Processing) to User-Analysts: An IT Mandate // Codd & Date, Inc. 1993. – 31 p.

[4] P. Giorgini, S. Rizzi, M. Garzetti. Goal-oriented requirement analysis for data warehouse design // In Proceedings of the 8th ACM international Workshop on Data Warehousing and OLAP: DOLAP '05, 2005. P. 47–56.

[5] J. Lechtenborger, G. Vossen. Multidimensional normal forms for data warehouse design // Inf. Syst., V. 28, No. 5, 2003. P. 415–434.

[6] W. Lehner, J. Albrecht, H. Wedekind. Normal forms for multidimensional databases // Proceedings of the Tenth International Conference on Scientific and Statistical Database Management, 1998. P. 63–72.

[7] M. Levene, M.W. Vincent. Justification for Inclusion Dependency Normal Form // IEEE Transactions on Knowledge and Data Engineering. 2000. Vol. 12, No. 2. P. 281–291.

[8] H.-G. Li, H. Yu, D. Agrawal, A.E. Abbadi. Progressive ranking of range aggregates // Data & Knowledge Engineering, 2007. Vol. 63, No. 1. P. 4–25.

[9] D. Maier. The theory of relational databases// Computer Science Press, 1983. – 637 p.

[10] J. Mazon, J. Trujillo, J. Lechtenborger. Reconciling requirement-driven data warehouses with data sources via multidimensional normal forms. Data Knowl. Eng. 2007. V. 63, No. 3. P. 725–751.

[11] L. Miller, S. Nila. Data Warehouse Modeler: A CASE Tool for Warehouse Design // Thirty-First Annual Hawaii International Conference on System Sciences. 1998. Vol. 6. P. 42–48.

[12] R. Missaoui, R. Godin. The Implication Problem for Inclusion Dependencies: A Graph Approach // SIGMOD Record. 1990. Vol. 19, No 1. P. 36–40.

[13] T.B. Pedersen, C.S. Jensen, C.E. Dyreson. A foundation for capturing and querying complex multidimensional data // Inf. Syst. 2001. Vol. 26, No. 5. P. 383–423.

[14] J. Ullman. Principles of database systems // Computer Science Press, 1980. – 379 p.

[15] P. Vassiliadis, T. Sellis. A survey of logical models for OLAP databases // SIGMOD Rec. 1999. Vol. 28, No 4. P. 64–69.

[16] S.V. Zykin. Formation of Hypercube Representation of Relational Database // Programming and Computer Software. 2006. Vol. 32, No. 6. P. 348–354.

[17] П. Г. Редреев. Автоматизация формирования иерархий измерений многомерных моделей данных // Известия Саратовского университета. Новая серия. Серия: Математика. Механика. Информатика. 2009. Т. 9. № 4, ч. 1. С. 84–87.

Technology of Multidimensional Data Formation

Sergey V. Zykin, Sergey V. Mosin, Andrey N. Poluyanov

In paper the automation problem of shaping of multidimensional data presentation with the use of the separate formation of dimensions and measures is considered. Data source is a relational database. For the source database any restrictions are not imposed. For example, the requirement of its schema hierarchy. The essential difference of the proposed technology is the absence of a functional dependency between the measures and dimensions.