Знания-Онтологии-Теории (ЗОНТ-09)

Автоматизация формирования кубов данных с использованием контекстных ограничений[1]

Зыкин С.В.[2], Полуянов А.Н.[2]

[2]*Омский филиал Института математики СО РАН, ул. Певцова, д. 13, г. Омск, 644099, Россия.*

zykin@ofim.oscsbras.ru, Andrey.Poluyanov@gmail.com

Аннотация. *В статье рассмотрены проблемы автоматизации формирования интерфейса между многомерной таблицей и реляционной базой данных. Формализована постановка задачи и представлено описание существующих подходов к формированию многомерных представлений данных на примере широко распространенных CASE-инструментов. Предложено определение промежуточного представления данных в виде таблицы соединений, которая используется для обеспечения корректности формирования многомерного представления данных, и также необходима для прямого и обратного преобразования данных. Для анализа корректности построений рассматриваются реализованные зависимости (функциональные, многозначные, соединения, включения). На основе свойства соединения без потери информации и реализованных зависимостей вводится понятие и способ формирования контекстов приложения и ограничений на данные.*

Ключевые слова: online analytical processing, гиперкубическое представление данных, реляционная база данных

1 Введение

Основой технологии оперативной аналитической обработки данных OLAP (online analytical processing) [1–4] является построение гиперкубического (многомерного) представления данных. В дальнейшем это представление используется в интеллектуальном анализе данных (Data Mining) [5–6].

В работах, посвященных OLAP, значительное внимание уделяется исследованию свойств моделей гиперкубов [7–9] и операциям преобразования гиперкубов [8,10] с целью получения представления, необходимого для анализа данных.

В большинстве работ предполагается постоянное хранение гиперкубических представлений для обеспечения наибольшей скорости обработки данных. При этом, основное внимание уделяется оптимизации времени формирования гиперкубического представления данных. Тогда как схемы гиперкубов считаются неизменными. Так, в работе [4] формулируется следующий тезис: "Прежде чем выполнить любое … из оптимизирующих преобразований, следует очень тщательно разобраться в том, какой именно потребуется анализ и какие для этого необходимы отчеты и служебная информация". В данной работе предполагается, что основой аналитической работы пользователя является необходимость формирования новых гиперкубов, а не многократное формирование реализации одного и того же гиперкуба. Следовательно, основное внимание необходимо акцентировать на сокращении времени формирования схемы нового

[1] Работа выполнена по проекту РФФИ № 09-07-00059-а

гиперкуба, а формирование представления гиперкуба должно быть выполнено автоматически соответствующими алгоритмами.

В работе [11] для автоматизации формирования гиперкуба предлагается использовать последовательность преобразований:

$$RRD \Rightarrow TJ \Rightarrow ST,$$

где ***RRD*** – реляционное представление данных, ***TJ*** – таблица соединений, ***ST*** – гиперкуб "семантическая трансформация". В данном случае ***RRD*** – представление исходной модели данных, ***ST*** – целевой. Представление ***TJ*** является промежуточным.

Рассмотрим принципиальную схему формирования представлений. Пусть ***RRD*** задано совокупность отношений $\mathfrak{R}=\{R_1, R_2, ..., R_k\}$ реляционной базы данных (БД), определенных на множестве атрибутов $U=\{A_1, A_2, ..., A_n\}$. Представление (таблица) ***TJ*** формируется из результатов естественных соединений некоторого подмножества отношений: $R'=\{R'_1, R'_2, ..., R'_p\}\subseteq\mathfrak{R}$. Для каждого кортежа ***u*** каждого соединения сформируем кортеж ***t*** по следующим правилам: $t[A_j] = u[A_j]$, если атрибут A_j принадлежит соединению, и $t[A_j] = emp$ в противном случае. Каждому кортежу поставим в соответствие битовый вектор $l(t) = (l_1(t), l_2(t), ..., l_p(t))$, где $l_i(t)=1$, если реализация $R_{m(i)}$ участвует в текущем соединении, и $l_i(t)=0$ в противном случае. Схема ***TJ*** (заголовок таблицы) состоит из совокупности атрибутов $U'=\{A'_1, A'_2, ..., A'_q\}\subseteq U$, на которых определены отношения R'_i. На следующем этапе формируется гиперкуб ***ST*** размерности ***q***. Для этого дважды просматриваются кортежи таблицы ***TJ***. При первом просмотре формируются области определения атрибутов A'_j, которые после сортировки становятся значениями координат гиперкуба (измерениями). Затем создается гиперкуб с выбранными значениями координат и пустыми ячейками (значениями мер) в рабочей области гиперкуба. При повторном просмотре ***TJ*** в соответствующих ячейках гиперкуба проставляются отметки (признаки наличия соответствующего кортежа в ***TJ***). Такое представление гиперкуба является наиболее общим, но и самым разреженным. Из него можно получить другие представления в виде проекций на подпространства (некоторые измерения становятся мерами), свертки по координатам (суммирование или усреднение) и т.п. Из способа формирования гиперкуба ясно, что его содержимое полностью определяется содержимым ***TJ***. Поэтому основное внимание далее сосредоточим на формировании таблицы ***TJ***.

2 Реализованные зависимости

Рассмотрим базовые зависимости, используемые при проектировании схемы БД [12]. Пусть ***D*** – множество зависимостей (функциональных, многозначных, включения, соединения), определенных на множестве атрибутов ***U*** и множестве отношений $\mathfrak{R}$.

Определение 1 (ФЗ). Пусть ***X*** и ***Y*** – некоторые подмножества из множества атрибутов ***U***. Будем говорить, что ***X*** функционально определяет $Y\ (X\rightarrow Y)$, если в любой реализации ***r*** схемы ***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 2 (МЗ). Множество ***X*** мультиопределяет множество ***Y*** в контексте ***Z***: $X\rightarrow\rightarrow Y(Z)$ (многозначная зависимость), если для произвольной реализации ***r*** схемы ***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 (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.

Формальным основанием для установления связей являются зависимости включения [13]:

Определение 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) [14, 15]. 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 [15], оставшаяся ЗВ после удаления избыточных зависимостей установлена для атрибутов-синонимов.

Рассмотренные зависимости достаточно хорошо исследованы. Однако на практике они могут оказаться бесполезными, если существующее программное обеспечение не может их поддерживать в актуальном состоянии. Множество всех зависимостей задают допустимые состояния БД. Пусть S – совокупность допустимых состояний.

Определение 5. Зависимость $d_i \in D$ будем считать реализованной, если переход из состояния $s_j \in S$ в состояние $s_l \notin S$, противоречащий зависимости d_i, будет заблокирован организационно-техническими средствами.

Другими словами, в произвольном отношении $R_m \in \Re$ невозможно добавить, удалить или модифицировать кортеж, если после выполнения операции он будет противоречить какой-либо зависимости из D. 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.

Рассмотрим примеры таких средств. ФЗ реализуются, прежде всего, за счет первичных ключей. Пусть $PK(m)$ – первичный ключ в отношении R_m, тогда СУБД заблокирует дополнение кортежа в отношение R_m, если в нем уже есть кортеж с совпадающими значениями атрибутов первичного ключа $PK(m)$. Альтернативный первичный ключ в отношении R_m реализуется на схеме БД заданием опции "уникальный" ("без повторений") для атрибутов этого ключа. СУБД, препятствуя появлению дублированных значений таких атрибутов, автоматически реализует ФЗ, соответствующие альтернативным первичным ключам в отношении R_m. В более сложных случаях ФЗ реализуется с использованием триггеров.

Для реализации ЗВ используются связи на схеме БД. Обозначим $L_1(i,j)$ – связь **1:1** от R_i к R_j, где R_i главное отношение; $L_M(i,j)$ – связь **1:M** от R_i к R_j, где R_i главное отношение; $L(i,j)$ – связь **1:1** либо **1:M** от R_i к R_j, где R_i главное отношение.

Определение 6. Между отношениями R_i и R_j существует связь $L_1(i,j)$, если $PK(R_i)=PK(R_j)$ и для любых реализаций R_i и R_j, выполнено $R_j[V] \subseteq R_i[V]$, где $V=[R_i] \cap [R_j]$, $[R_i]$ – множество атрибутов, на которых определено отношение R_i.

Определение 7. Между отношениями R_i и R_j существует связь $L_M(i,j)$, если $PK(R_i) \neq PK(R_j)$ и $PK(R_i) \subseteq [R_j]$.

Заметим, что определения 1 и 2 соответствуют частному случаю типизированных ЗВ, которые поддерживается системами управления базами данных (СУБД) за счет создания внешних ключей (foreign key). Ограничение целостности, задаваемое связью $L_M(i,j)$, подразумевает $R_j[V] \subseteq R_i[V]$, где $V=[R_i] \cap [R_j]$. Связь устанавливается между парой отношений по однотипным атрибутам, что позволяет СУБД предотвратить удаление кортежа в главном отношении, если в подчиненном отношении имеется кортеж с совпадающими значениями связанных атрибутов. С другой стороны, в подчиненное отношение не удастся добавить кортеж, если в главном отношении нет кортежа с совпадающими значениями связанных атрибутов. Аналогично ФЗ более сложные варианты ЗВ могут быть заданы с использованием триггеров.

МЗ и ЗС в явном виде СУБД не поддерживаются. Такие зависимости будем считать реализованными, если они участвуют в декомпозиции отношений на этапе проектирования схемы БД. Например, МЗ $V \twoheadrightarrow W(Q)$ будет реализована, если существуют отношения R_i и R_j такие, что $VW \subseteq [R_i]$, $VQ \subseteq [R_j]$ и $[R_i] \cap [R_j]=V$, и зависимость $V \twoheadrightarrow W(Q)$ не является встроенной в отношение R, определенное на множестве атрибутов $[R_i] \cup [R_j]$, например, $PK(R_i) \subseteq VW$ и $PK(R_j) \subseteq VQ$. Тогда, независимое дополнение, удаление и модификация кортежей в R_i и R_j не приведет к появлению противоречивых кортежей в $R_i \bowtie R_j$. По аналогии с МЗ будем считать ЗС $*(R_{m(1)}, R_{m(2)},..., R_{m(s)})$ реализованной, если на схеме БД реализованы отношения $R'_{m(1)}$, $R'_{m(2)}$,..., $R'_{m(s)}$, где $R_{m(i)}=R'_{m(i)}[R_{m(i)}]$ для всех $i \in \{1,2,...,s\}$ и $[R'_{m(i)}] \cap [R'_{m(j)}]=[R_{m(i)}] \cap [R_{m(j)}]$ для всех $i,j \in \{1,2,...,s\}$, $i \neq j$. Кроме того, зависимость $*(R_{m(1)}, R_{m(2)},..., R_{m(s)})$ не является встроенной в отношение R, определенное на множестве атрибутов $[R'_{m(1)}] \cup [R'_{m(2)}] \cup ... \cup [R'_{m(s)}]$, например, $PK(R'_{m(i)}) \subseteq [R_{m(i)}]$ для всех $i \in \{1,2,...,s\}$. Тогда, независимое дополнение, удаление и модификация кортежей в $R'_{m(1)}$, $R'_{m(2)}$,..., $R'_{m(s)}$ не приведет к появлению противоречивых кортежей в отношении $R'_{m(1)} \bowtie R'_{m(2)} \bowtie ... \bowtie R'_{m(s)}$. Приведенные рассуждения следуют непосредственно из определения МЗ и ЗС. Однако СУБД не сможет воспрепятствовать появлению запроса пользователя, в котором отношения R_i и R_j соединяются по атрибутам $T \subset V$, и в результате дают несогласованные между собой кортежи. Аналогичная ситуация может произойти, если в запросе будет отсутствовать часть отношений ЗС. Автоматическое формирование

пользовательского представления данных с учетом указанных свойств зависимостей позволит избежать рассмотренных ситуаций.

При проверке свойства СБПИ используются ФЗ и ЗС (в частном случае МЗ [16]). Если какие-либо зависимости не реализованы, то СУБД не сможет воспрепятствовать появлению кортежа, который противоречит нереализованной зависимости. Следовательно, не гарантируется появление противоречивых кортежей. Для обоснования этого факта рассмотрим простейший пример. Пусть $U=ABC$ – множество атрибутов, множество ФЗ: $F=\{B\rightarrow C,\ AC\rightarrow B\}$. Пусть построена декомпозиция: $\rho(R_1,R_2)$, где $[R_1]=AB$ и $[R_2]=BC$ (нормальная форма Бойса-Кодда). Такая декомпозиция удовлетворяет свойству СБПИ (теорема Хеза) и пусть зависимость $AC\rightarrow B$ не реализуется. Оператор может дополнить в R_1 два кортежа: (a_1,b_1), (a_1,b_2), и в R_2 кортежи: (b_1,c_1), (b_2,c_1), что не противоречит реализованным ограничениям. Однако, выполнив операцию естественного соединения отношений получим, что нарушена ФЗ $AC\rightarrow B$. Дополнив декомпозицию ЗВ $R_2[B]\subseteq R_1[B]$ получим нормальную форму зависимостей включения [15], которая также не решает указанную проблему. Следовательно, выполнение свойства СБПИ на нереализованных зависимостях не гарантирует отсутствие противоречивых кортежей. В этом же примере можно выполнить декомпозицию $\rho(R_1)$, где $[R_1]=\underline{AB}\underline{C}$ (третья нормальная форма), где подчеркнуты ключевые атрибуты. Эта декомпозиция сохраняет зависимости F, но зависимость $B\rightarrow C$ не реализована средствами СУБД, что не помешает оператору ввести два кортежа с одинаковыми значениями атрибута B и различными значениями C. Декомпозиция $\rho(R_1,R_2)$, где $[R_1]=\underline{AB}\underline{C}$ и $[R_2]=\underline{BC}$, дополненная ЗВ $R_1[BC]\subseteq R_2[BC]$ лишена указанных противоречий, однако является избыточной.

Определение 8. ФЗ $V\rightarrow W$ реализована на множестве отношений $R'=(R'_1, R'_2, ..., R'_q)$, $q>0$ и $V\cup W\subseteq[R']$, если для любых реализаций отношений $R'_1, R'_2, ..., R'_q$ ФЗ $V\rightarrow W$ выполнена для $m_\rho(R')=R'_1\bowtie R'_2\bowtie ...\bowtie R'_q$.

Рассмотрим $R''=(R''_1, R''_2, ..., R''_p)$ множество отношений БД, F'' – множество реализованных ФЗ на отношениях из R''. Пусть W и V, где $W\subseteq[m_\rho(R'')]$ и $V\subseteq[m_\rho(R'')]$. Обозначим W^+ – замыкание множества атрибутов на множестве ФЗ [Ульман, Мейер].

Теорема 1. ФЗ $W\rightarrow V$ реализована в $m_\rho(R'')$, если $V\subseteq W^+$ на множестве F''.

Доказательство. Пусть $V\subseteq W^+$. Предположим, что зависимость $W\rightarrow V$ не реализована в $m_\rho(R'')$. Тогда отношение $m_\rho(R'')$ будет содержать два кортежа t_1, t_2, таких, что $t_1[W]=t_2[W]$ и $t_1[V]\neq t_2[V]$. Так как $V\subseteq W^+$, то существует последовательность зависимостей $W_0\rightarrow W_1$, $W_1\rightarrow W_2,...,W_p\rightarrow W_{p+1}\in F''$, $W_0=W$, и $W_{p+1}=V$. Из правила построения замыкания (к замыканию присоединяется правая часть ФЗ, если левая уже принадлежит ему) следует, что найдется зависимость $W_i\rightarrow W_{i+1}$, для которой $t_1[W_i]=t_2[W_i]$ и $t_1[W_{i+1}]\neq t_2[W_{i+1}]$. Зависимость $W_i\rightarrow W_{i+1}$ реализована отношениями $R^i=(R^i_1, R^i_2, \ldots, R^i_s)$. Поскольку $R^i\subseteq R''$, то по правилу выполнения операции естественного соединения кортежи $t_1[R^i]$ и $t_2[R^i]$ будут принадлежать $m_\rho(R^i)$, что противоречит реализованной зависимости $W_i\rightarrow W_{i+1}$.

Замечание. Использование этой теоремы совместно с теоремой 5.8 [16] дает возможность проверки выполнения свойства СБПИ без использования базового алгоритма [16]. Для этого достаточно, чтобы в совокупности отношений присутствовало отношение, замыкание ключа которого, совпадало со всем множеством атрибутов в этих отношениях.

Определение 9. ЗВ $R_j[V]\subseteq R_i[V]$ будем считать реализованной на схеме БД, если установлена связь $L(i,j)$ по множеству атрибутов W, где $V\subseteq W$.

Теорема 2. ЗВ $R_j[V]\subseteq R_i[V]$ реализована на схеме БД, если $R_j\in R^+_i[W]$, где $V\subseteq W$.

Доказательство. Поскольку $R_j\in R^+_i[W]$, то существует цепочка связей $L(m(0),m(1))$, $L(m(1),m(2))$, …, $L(m(p-1),m(p))$, где $m(0)=i$ и $m(p)=j$. Возможно, что таких последовательностей будет несколько. Наличие связи $L(m(l-1),m(l))$ гарантирует, что в отношении $R_{m(l)}$ не появится кортеж t такой, что $t[W]\notin R_{m(l-1)}[W]$, $l\in\{1,2,\ldots,p\}$. Следовательно, $R_j[W]\subseteq R_i[W]$. Поскольку $V\subseteq W$, то по аксиоме рефлексивности для ЗВ [14] имеем $R_j[V]\subseteq R_i[V]$. Что и требовалось доказать.

Замечание. ЗВ позволяют сделать направленным перебор отношений при формировании контекста. Доказанная теорема, показывает, что для этой цели возможно использование не только ЗВ, реализованные на схеме БД в виде связей, но и избыточные ЗВ, отсутствующие в явном виде на схеме БД.

3 Формирование контекстов приложения

В работе предлагается формирование схемы гиперкуба осуществлять за счет выбора атрибутов – координат (измерений гиперкуба) атрибутов – мер, чьи значения находятся в рабочей области гиперкуба, и логического выражения F на атрибутах, ограничивающего допустимые значения измерений и мер. Причем, атрибуты в выражении F могут не принадлежать ни измерениям, ни мерам, а ограничение будет задаваться опосредованно, за счет контекстов.

Пусть $R=\{R_1, R_2, ..., R_p\}$ – множество отношений реляционной БД. D – множество реализованных зависимостей на отношениях из R.

Определение 10. Множество R будем называть контекстом, если оно удовлетворяет свойству СБПИ на зависимостях D.

Не вдаваясь глубоко в семантические проблемы интерпретации результатов реляционных операций отметим, что 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.

Первоначальный выбор измерений и мер гиперкуба, а также атрибутов для формулы F, 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 $R^0=\{R^0_1, R^0_2, ..., R^0_q\}$, участвующее в обязательном порядке сначала в формировании таблицы соединения, а потом – гиперкуба.

Дальнейшая задача состоит в дополнении множества R^0 отношениями из $\mathfrak{R}$, чтобы результирующее множество удовлетворяло свойству СБПИ на множестве реализованных 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. То есть, очередным претендентом на дополнение к текущему множеству отношений будет R_i, которое позволяет сделать наибольшее количество подстановок в таблице алгоритма проверки свойства СБПИ.

Определим существующее соединение [17].

Определение 11. Выражение $R_1 \bowtie R_2 \bowtie ... \bowtie R_k$ будем называть существующим соединением, если для совокупности схем R_i, $i=1,\ldots,k$, существует хотя бы одна перестановка $V_1, V_2, ..., V_k$ схем $R_1, R_2, ..., R_k$ такая, что $(V_1 \cup V_2 \cup ... \cup V_j) \cap V_{j+1} \neq \varnothing$, $j=1,\ldots,k-1$.

Сформулируем критерии, которые позволят сделать перебор отношений направленным.

1. Дополняемые отношения должны образовывать существующее соединение с уже выбранными отношениями. В противном случае будет гарантировано невыполнение свойства СБПИ, если отсутствуют EMP-зависимости: $\varnothing \twoheadrightarrow W(Q)$ [17]. В работе [18] предложен эффективный алгоритм проверки этого свойства.
2. Замыкание первичного ключа нового отношения R_i совпадает со всем множеством атрибутов в выбранных отношениях (теорема 1). Чаще всего такое отношение имеет связи $L_M(j,i)$ с уже выбранными отношениями R_j. Воспользовавшись теоремой 2 можно выбрать такие отношения, даже если соответствующие им связи являются избыточными и удалены из схемы БД.
3. Формируемый контекст не должен содержать лишних отношений, наличие которых обусловлено только порядком присоединения отношений к контексту в алгоритме.

К сожалению, перечисленные критерии только увеличивают вероятность более быстрого достижения результата. Однако, в некоторых случаях необходимая комбинация отношений будет получена не на начальных итерациях алгоритма. Например, из перечисленных критериев следует, что отношение R_i, имеющие связи $L_M(i,j)$ с уже выбранными отношениями R_j, будет рассматриваться в последнюю очередь. Действительно, такие отношения чаще всего только дополняют новые атрибуты к таблице проверки свойства СБПИ, и реализованные на них зависимости не позволяют сделать подстановки для уже содержащихся в таблице атрибутов. Однако, следующий пример показывает, что у этого правила есть исключение. Пусть даны два исходных отношения: $R_1(\underline{E},\underline{A},B)$ и $R_2(\underline{C},\underline{B},E)$ (подчеркнуты первичные ключи отношений), которые не удовлетворяют свойству СБПИ. Дополнив отношение $R_3(\underline{C},A)$ получим, что

свойство СБПИ будет выполнено, хотя имеется только одна связь $L_M(3,2)$. Следовательно, такие отношения нельзя исключать из рассмотрения, однако анализировать их следует в последнюю очередь.

Введем обозначения. Пусть $R^1=\{R^1_1, R^1_2, ..., R^1_p\}$ – 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: $U^0=[R^0_1]\cup[R^0_2]\cup\ldots\cup[R^0_q]$. D^0 – множество зависимостей, реализованных на отношениях из R^0. T^0 – результирующая таблица алгоритма проверки свойства СБПИ для отношений из R^0 и атрибутов U^0 на множестве зависимостей D^0. Функция ***Exist(R)*** имеет значение ***TRUE***, если совокупность отношений R образует существующее соединение, и ***FALSE*** – в противном случае. Функция ***Comb(m,i,j)*** последовательно формирует сочетания без повторений из j элементов по i в изначально нулевом массиве m и выдает значение ***TRUE***, если текущее сочетание существует, и ***FALSE*** – в противном случае. Сочетания, формируемые функцией ***Comb***, начинаются с наименьших значений, и далее последовательно увеличиваются, например сочетания по два элемента: ***(1,2)***, ***(1,3)***,..., ***(2,3)***.... При $i=0$ присваивается $m(1)=1$ и выдается значение ***TRUE***, следующее значение – ***FALSE***.

Рассмотрим схему алгоритма, удовлетворяющего сформулированным критериям.

begin form_cont
for i=1 to p; {Подсчет весов отношений $R_j\in R^1$ в массиве m}
  $m(i)=|[R^1_i]\cap U^0|$; {Общее количество атрибутов текущего отношения с множеством U^0}
  for j=1 to q;
    if $PK(R^0_j)\subseteq[R^1_i]$ ***then*** $m(i)=m(i)+1$; {Проверяется наличие связи}
  endfor;
endfor;
<Сортировка $R_j\in R^1$ по убыванию веса $m(j)$. Результат: $R^2=\{R^2_1, R^2_2, ..., R^2_p\}$>
for i=0 to p;
  $m(1,\ldots,p)=0$;
  do while Comb(m,i, p);
    if $Exist(R^0\cup R^2_{m(1)}\cup R^2_{m(2)}\cup\ldots\cup R^2_{m(i)})$ ***then;***
      <Расширение таблицы T^0 отношениями $R^2_{m(1)}, R^2_{m(2)},\ldots,R^2_{m(i)}$ и их атрибутами>
      <Расширение D^0 реализованными зависимостями на $R^0\cup R^2_{m(1)}\cup R^2_{m(2)}\cup\ldots\cup R^2_{m(i)}$>
      if **<Совокупность отношений $R^0\cup R^2_{m(1)}\cup R^2_{m(2)}\cup\ldots\cup R^2_{m(i)}$ существующая>** ***then;***
        if **<Свойство СБПИ выполнено>** ***then;***
          <Сохранение контекста $R^0\cup R^2_{m(1)}\cup R^2_{m(2)}\cup\ldots\cup R^2_{m(i)}$>
          if **<Завершить генерацию контекстов>** ***then exit for;***
        endif;
      endif
    endif;
  enddo;
endfor;
<Больше контекстов нет>
end form_cont

Замечание. Рассмотренный алгоритм на ближайших итерациях находит дополнительные отношения с наибольшей вероятностью образующие наименьший контекст с исходным множеством отношений.

4 Контексты ограничений и таблица соединения

Наложение ограничений на формируемые кортежи в ***TJ*** предлагается делать двумя способами [19]:

1) Указывается совокупность отношений, которые участвуют в формировании ***TJ*** совместно, то есть остатки соединений этих отношений не должны попасть в ***TJ*** (это и есть ограничение). Например, в сводной ведомости по результатам сдачи экзаменов должны попасть только те предметы, которые изучаются на данной специальности. В этом примере участвуют отношения: "Оценки", "Предметы", "Учебная нагрузка" и т.д. Очевидно, чтобы в ***TJ*** не появились лишние строки, совокупность отношений должна быть контекстом.

2) На формируемые кортежи в ***TJ*** накладывается ограничение в виде логической формулы, определенной на совокупности атрибутов. Как в предыдущем примере, требуется выдать сводную ведомость для какой-либо одной учебной группы.

Ограничения на кортежи, участвующие в формировании таблицы соединения, удобнее всего задавать в виде дизъюнктивной нормальной формы (ДНФ): $F=F_1\vee F_2\vee\ldots\vee F_s$, где

$F_i=F_{i1}\wedge F_{i2}\wedge\ldots\wedge F_{ij}$. Совокупность отношений, на которые накладывается условие F_i, соответствуют одному контексту, который выступает как единое целое при формировании таблицы соединения (отношения из такого контекста не должны давать альтернативные кортежи, а значит и не должны участвовать в формировании таблицы соединения отдельно от своего контекста). Условия F_i и F_j задают альтернативные ограничения и, следовательно, в таблице соединения должны присутствовать альтернативные кортежи из различных контекстов, что гарантируется алгоритмом формирования таблицы соединения [19].

Способ формирования контекстов для указанных ограничений совпадает с формированием контекста приложения. Для логических ограничений начальное множество отношений задается в самой формуле F_i. Из этого множества определяются другие начальные условия алгоритма ***form_cont***. Далее для формирования контекста ограничения, соответствующего F_i, используется алгоритм ***form_cont***. Целесообразно во всех случаях увеличить веса отношений, которые уже выбраны для контекста приложения, чтобы они быстрее попали в поле зрения пользователя.

5 Заключение

Во введение рассмотрена схема формирования гиперкуба наиболее общего вида. На практике применяются различные методы сжатия информации в представлении гиперкуба (свертки, проекции и т.д.). Метод сжатия без потери информации, основанный на операции проекции, предложен в работе [11]. Этот метод позволяет сделать представление гиперкуба наглядным (в виде таблицы), отображая данные на плоскость. Для этого исходное множество атрибутов, сформированное пользователем, разбивается на три множества: X, Y и Z, где X и Y обобщенные координаты (многоуровневые заголовки строк и столбцов таблицы), а атрибуты Z становятся мерами, то есть, располагаются в рабочем поле таблицы. Каждая ячейка таблицы разбивается на количество ячеек, совпадающее с количеством атрибутов в множестве Z. Ограничением на такое преобразование является наличие реализованной ФЗ: $XY\rightarrow Z$.

Существующие промышленные разработки, реализующие технологию OLAP: ORACLE-Express, ИнфоВизор OLAP-Дизайнер, RS-DataHouse и т.д., не содержат в своем составе компонентов, предлагаемых в данной статье. Предоставляя пользователю широкий набор средств для формирования структуры гиперкубов и выполнения операций над гиперкубами, эти разработки возлагают всю ответственность за корректность предоставления результирующих данных на пользователя. Другими словами, пользователь сам должен просчитать выполнение свойств промежуточных представлений, чтобы в них не появились лишние кортежи, противоречащие свойству СБПИ [12,16], и, как следствие, лишние значения в рабочей области гиперкуба. Наилучшим решением данной проблемы является дополнение существующих промышленных разработок средствами, предлагаемыми в данной статье.

Литература

[1] Harinarayan V., Rajaraman A., Ullman J. D. Implementing Data Cubes Efficiently// *SIGMOD Conference. - Montreal, CA*. 1996. - P. 205-216.

[2] Gray J., Chaudhuri S., Bosworth A., etc. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals// *Data Mining and Knowledge Discovery*. - 1997. - № 1. - P. 29-53.

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

[4] Armstrong R. Seven Steps to Optimizing Data Warehouse Performance// *Computer*, V. 34, № 12, 2001. - P. 76-79.

[5] Parsaye K. Surveying Decision Support: New Realms of Analysis// *Database Programming and Design*. - 1996. - № 4. - P. 26-33.

[6] Parsaye K. OLAP and Data Mining: Bridging the Gap// *Database Programming and Design*. - 1997. - № 2. - P. 30-37.

[7] Vassiliadis P., Sellis T. A survey of logical models for OLAP databases// *SIGMOD Rec*., V. 28, № 4, 1999. P. 64-69.

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

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

[10]Li H.-G., Yu H., Agrawal D., Abbadi A.E. Progressive ranking of range aggregates// *Data & Knowledge Engineering*, V. 63, 2007. P. 4-25.

[11]Зыкин С.В. Формирование гиперкубического представления реляционной базы данных // *Программирование*. - 2006. - № 6. - С. 348 - 354.

[12]Мейер Д. Теория реляционных баз данных// - *М.: Мир*, 1987. - 608 с.

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

[14]Missaoui R., Godin R. The Implication Problem for Inclusion Dependencies: A Graph Approach// *SIGMOD Record*. - 1990. - V 19, - № 1. - P. 36 - 40.

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

[16]Ульман Дж. Основы систем баз данных// - *М.: Финансы и статистика*, 1983. - 334 с.

[17]Зыкин С.В. Построение отображения реляционной базы данных в списковую модель данных// *Управляющие системы и машины*. - 2001. - № 3. - С. 42 - 63.

[18]Полуянов А.Н. Алгоритм проверки существования соединения отношений при межмодельных преобразованиях данных// *Математическое моделирование и информационные технологии: управление, искусственный интеллект, прикладное программное обеспечение, технологии программирования : материалы VIII школы-семинара молодых ученых* / Ин-т динамики систем и теории управления СО РАН. - Иркутск, 2006. - С. 145-149.

[19]Зыкин С.В., Полуянов А.Н. Формирование представлений данных с контекстными ограничениями// *Омский научный вестник. Серия "Приборы, машины и технологии"*, 2008, № 1(64). - С. 141-144.