Математические структуры и моделирование
2003, вып. 11, с. 175–183

УДК 681.3

АВТОМАТИЗАЦИЯ ПОСТРОЕНИЯ ЗАПРОСОВ К РЕЛЯЦИОННЫМ БАЗАМ ДАННЫХ

А.В. Угаров

In this article the new method of construction of queries to relation databases. The method is based on join property verification and let automatic construction queries with correct outer joins.

Введение

По мере распространения сетевых технологий все больше и больше информации можно получить в Интернет. Причем информация улучшается не только количественно, но и качественно. Все чаще web-странички наполняются динамическим содержанием и, более того, строятся на основе данных, хранящихся в базах данных в структурированном виде. Еще одним двигающим фактором структуризации данных в Интернет является неудовлетворительность поиска данных в неформализованных источниках [1]. Таким образом, наблюдается тенденция к хранению информации во всемирной сети в структурированных источниках данных, которая, по-видимому, в ближайшие годы еще более укрепит свои позиции.

С другой стороны, для доступа к сильно структурированным данным необходим соответствующий инструментарий. Причем наиболее интересна возможность работы с различными СУБД и произвольными базами данных. Подобные механизмы реализованы в современных профессиональных приложениях для работы с базами данных, таких как Business Objects, Oracle Forms, Microsoft Access, Delphi (утилита SQL Builder) и других. Однако все они предназначены в основном для разработчиков программного обеспечения, к тому же ориентированы на традиционные базы данных.

Одна из основных целей подобного инструментария – это предоставление наиболее прозрачно для пользователя, внутренней структуры базы данных и максимальная простота при работе с ней. То есть необходим инструментарий для организации интерфейса между пользовательским представлением базы данных, в терминах которого будет формироваться запрос, и внутренним представлением данных в базе данных. Запрос, сформированный на основе пользовательского представления данных, должен быть корректно преобразован в

E-mail: uav@opsb.ru

ОФ Института математики им. С.Л. Соболева СО РАН

запрос на основе внутреннего представления [2–4]. Под корректностью здесь понимается однозначное отображение состояний пользовательского представления данных в состояния внутреннего представления. Вместе с возрастающим количеством информационных ресурсов полная или частичная автоматизация построения соответствующего отображения приобретает все большую актуальность.

Соответствующее отображение можно построить, определив механизм преобразования произвольного состояния пользовательского представления в соответствующее ему состояние внутреннего представления. Для второго представления наиболее удобно использовать реляционную модель данных [3]. Соответственно, исходный запрос пользователя следует преобразовать в универсальный реляционный запрос (УРЗ) [5, 6]. В данной статье рассматривается проблема автоматизации построения УРЗ. Также в работе сформулированы требования к внутреннему представлению данных, приведен алгоритм построения множества отношений для УРЗ.

1. Теоретические основы межмодельных отображений

При построении схемы реляционной базы данных традиционно используется процесс нормализации исходного отношения. В теории нормализации центральным понятием является декомпозиция. Будем использовать следующее устоявшееся определение декомпозиции [5, 6]:

Определение 1. Пусть задана схема отношения R на множестве атрибутов $U = \{A_1, A_2, \ldots, A_n\}$ и множество $\mathbb{F}$ всех функциональных зависимостей на R. Декомпозицией схемы отношения R называется совокупность схем отношений $\rho = \{R_1, R_2, \ldots, R_k\}$ таких, что $\bigcup_{j=1}^{k} R_j = R$, то есть $\bigcup_{j=1}^{k} U_j = U$, где $U_j = \{A_{j_1}, A_{j_2}, \ldots, A_{j_m}\}$ – множество атрибутов отношения R_j. При этом не требуется, чтобы R_j были непересекающимися.

На практике декомпозиция – это разбиение первичного отношения на несколько отношений, при котором получается «лучшая», по тем или иным причинам, схема базы данных. За примером построения декомпозиции можно обратиться к [7].

Осуществление декомпозиции некоторого первоначального множества схем отношений может снять часть проблем, возникающих при проектировании реляционной базы данных (избыточность, аномалии обновления, включения, удаления) [5–8]. Вместе с тем при «неправильной» декомпозиции могут возникнуть проблемы при восстановлении исходного отношения. Таким образом, при построении декомпозиции необходимо учитывать дополнительные ограничения. В частности, декомпозиция должна удовлетворять свойству соединения без потерь информации (ССП).

Определение 2. Пусть R – схема отношения, в результате декомпозиции которой получены схемы $R_1, R_2, \ldots, R_k$ и $\mathbb{F}$ - множество функциональных зависимостей на R. Говорят, что декомпозиция $\rho = \{R_1, R_2, \ldots, R_k\}$ обладает

свойством соединения без потерь (относительно $\mathbb{F}$), если любое отношение r со схемой R, удовлетворяющее $\mathbb{F}$, может быть представлено в виде:

$$r = \pi_{R_1}(r) \bowtie \pi_{R_2}(r) \bowtie \ldots \bowtie \pi_{R_k}(r),$$

то есть отношение r является естественным соединением своих проекций на все R_j, $j = \overline{1,k}$.

Если декомпозиция обладает свойством соединения без потерь, то любое отношение может быть восстановлено из его проекций.

В более узком случае, когда нужно восстановить подмножество исходного отношения, используется понятие локального свойства соединения без потерь (локальное ССП).

Определение 3. Пусть R – схема отношения, в результате декомпозиции которой получены схемы $R_1, R_2, \ldots, R_k$ и $\mathbb{F}$ - множество функциональных зависимостей на R. Говорят, что схемы $R_1, R_2, \ldots, R_l$ $(l < k)$ удовлетворяют локальному свойству соединения без потерь (относительно $\mathbb{F}$), если для любого отношения r со схемой R, удовлетворяющему $\mathbb{F}$, выполнено:

$$\pi_{R'}(r) = \pi_{R_1}(r) \bowtie \pi_{R_2}(r) \bowtie \ldots \bowtie \pi_{R_l}(r), \qquad \text{где } R' = \bigcup_{i=1}^{l} R_i.$$

Выполнение локального ССП является необходимым условием для корректного восстановления подмножества исходного отношения. Если некоторое множество отношений не удовлетворяет локальному ССП, то запрос к базе данных, построенный на их основе, не является адекватным представлением пользовательского запроса. Другими словами, исходное отношение не может быть корректно восстановлено на основании информации из полученного при декомпозиции множества отношений.

Второе требование при осуществлении декомпозиции связано с понятием функциональных зависимостей [5,6,9]. Зависимости в $\mathbb{F}$ можно рассматривать как ограничения целостности для отношения R, поэтому при построении декомпозиции необходимо требовать сохранение функциональных зависимостей.

Определение 4. Пусть задана схема отношения R на множестве атрибутов $U = \{A_1, A_2, \ldots, A_n\}$ и $\mathbb{F}$ - множество всех функциональных зависимостей на R. Проекцией $\mathbb{F}$ на множество атрибутов $Z \subseteq U$, обозначаемой $\pi_z(\mathbb{F})$, называется множество зависимостей $X \to Y$, в $\mathbb{F}^+$, таких, что $X \cup Y \subseteq Z$.

Наиболее интересны проекции множества всех функциональных зависимостей, существующих на первичном отношении, на множество атрибутов отношений, получаемых в результате декомпозиции исходного отношения.

Определение 5. Декомпозиция $\rho = \{R_1, R_2, \ldots, R_k\}$ сохраняет множество зависимостей $\mathbb{F}$ на R, если из множества функциональных зависимостей $\bigcup_{j=1}^{k} \pi_{R_j}(\mathbb{F})$ логически следуют все зависимости, принадлежащие $\mathbb{F}$.

Свойство декомпозиции сохранять функциональные зависимости является важным критерием правильности построения структуры базы данных. Нарушение этого свойства приводит к возможности ввода противоречивых данных в базу данных.

Свойство сохранения функциональных зависимостей (СФЗ) и свойство соединения без потерь никак не связаны. Декомпозиция ρ может удовлетворять свойству соединения без потерь, не сохраняя множества функциональных зависимостей, и наоборот. Поэтому, помимо проверки ССП, при построении декомпозиции необходимо проверять также и СФЗ. В литературе [6] можно найти алгоритмы для проверки СФЗ.

И, наконец, третье требование: декомпозиция должна находиться в 3НФ. Это условие определяет отсутствие нежелательных (т.е. не от ключа) функциональных зависимостей атрибутов отношения, не входящих в первичный ключ.

Замечание 1. В общем случае полезно выполнение требований более высоких нормальных форм (4НФ, 5НФ). Это позволило бы избежать ошибок, связанных с многозначными зависимостями. Однако этот вопрос не рассматривается в рамках данной статьи.

Подытожим сказанное выше. При построении отображения пользовательского представления данных во внутреннее представление будем требовать выполнения следующих ограничений:

1. Схема базы данных находится в 3НФ.
2. Множество отношений схемы базы данных удовлетворяет ССП.
3. Схема базы данных сохраняет функциональные зависимости.

2. Алгоритм проверки свойства соединения без потерь

Существует несколько алгоритмов проверки ССП [5,6]. Как правило, в литературе такие алгоритмы приведены в виде, используемом для доказательства его корректности. Однако для наибольшей эффективности имеет смысл несколько изменить структуру алгоритма. Предлагается некоторая модификация алгоритма, предложенного Дж. Ульманом.

Алгоритм 1. *Пусть задана схема отношения $\mathbb{R}$ на множестве атрибутов $U = \{A_1, A_2, \ldots, A_n\}$, $\mathbb{F}$ – множество всех функциональных зависимостей на $\mathbb{R}$ и декомпозиция $\rho = \{R_1, R_2, \ldots, R_k\}$.*

Подготовительный этап. Строим таблицу с n столбцами и k строками. Столбец j соответствует атрибуту A_j, а строка i – схеме отношения R_i. На пересечении строки i и столбца j поместим символ «+», если $A_j \in R_i$. В противном случае поместим туда символ «−».

Рабочий цикл. Последовательно «рассматриваем» каждую из зависимостей $\{X \to Y\} \in \mathbb{F}$. Для зависимости $X \to Y$ отыскиваем все строки в таблице, где во всех столбцах, соответствующих атрибутам из множества X, находятся символы «+». Для всех найденных строк во всех столбцах, соответствующих атрибутам из множества Y, проставляем символы «+».

Этап проверки. Если после проверки всех зависимостей из $\mathbb{F}$ в таблице появилась строка, полностью состоящая из одних «+», тогда декомпозиция ρ удовлетворяет свойству соединения без потерь. Если за весь рабочий цикл в таблице не модифицировано ни одного значения, то декомпозиция не обладает

таким свойством. Если же строки, состоящей их одних «+» нет, но таблица была изменена, то необходимо повторить рабочий цикл. ■

Существует доказательство того, что алгоритм корректно проверяет свойство соединения без потерь [6, с.167].

Нетрудно показать, что алгоритм выдает верный результат и для проверки локального свойства соединения без потерь.

Данный алгоритм основан на построении замыкания множества атрибутов. Алгоритм полиномиально зависит от мощности множества функциональных зависимостей (ФЗ) и количества атрибутов. Интересны также алгоритмы проверки ССП на основе построения замыканий ФЗ. Несмотря на то что вычислительная трудность таких алгоритмов экспоненциальная, для построения локального замыкания ФЗ они могут быть более эффективны (в силу меньшего количества зависимостей).

3. Алгоритм построения запроса

Как уже было сказано выше, схема базы данных должна удовлетворять ССП. Однако, даже если в целом для схемы базы данных ССП выполнено, то для некоторого подмножества отношений локальное ССП выполнено не всегда. То есть не всегда возможно восстановление всех отношений из их проекций.

Тем не менее именно выполнение локального ССП является основным критерием для формирования запроса в терминах внутреннего представления базы данных. Запрос к реляционной базе можно представить в общем виде [5, 6]. Универсальный реляционный запрос (УРЗ):

$$\pi_X(\sigma_F(R_1 \bowtie R_2 \bowtie \ldots \bowtie R_k)), \tag{1}$$

где X – некоторое подмножество атрибутов отношений $R_j, j = \overline{1,k}$, F – логическое выражение над атрибутами.

Наиболее интересен механизм выбора множества отношений $\{R_1, R_2 \ldots R_k\}$. Очевидно, что это множество должно содержать все отношения, атрибуты которых имеют соответствие в пользовательском запросе. Однако если соответствующий набор отношений не удовлетворяют локальному ССП, то их естественное соединение может содержать лишние кортежи, то есть кортежи, отсутствующие в предметной области. Следовательно, УРЗ не будет эквивалентен запросу в терминах пользовательского представления. Чтобы избежать подобной проблемы, необходимо определить минимальное множество отношений, включающее все R_j, $j = \overline{1,k}$ и удовлетворяющее локальному ССП.

Возможны два подхода для построения такого множества. Первый заключается в последовательном исключении «лишних» отношений из схемы до минимального числа. Второй подход, наоборот, последовательно добавляет по некоторым правилам к уже существующему множеству отношений новое, «необходимое» отношение до тех пор, пока локальное ССП не будет выполнено. Второй подход более предпочтителен, поскольку на основе имеющихся данных «лучшее» отношение определить проще, чем «худшее». Однако возможен также и комбинированный подход.

Для построения множества отношений, удовлетворяющих локальному ССП (результирующего множества), предлагается следующий алгоритм:

Алгоритм 2. *Пусть* $\mathbb{M}$ *– множество всех отношений,* $\mathbb{X} \subset \mathbb{M}$ *– множество отношений, для которых проверяется выполнение локального ССП. Для любого отношения* $A \in \mathbb{M}$ *введем обозначения:* $U(A)$ *– множество всех атрибутов* A, $PK(A)$ *– множество атрибутов из отношения* A*, входящих в первичный ключ.*

Шаг 1. Строим множество $\mathbb{Y} := \{A \in \mathbb{M} \setminus \mathbb{X} \mid PK(A) \subset \bigcup_{B \in \mathbb{X}} U(B)\}$ *(множество всех отношений, не принадлежащих* $\mathbb{X}$*, первичный ключ которых содержится в объединении всех атрибутов всех отношений из* $\mathbb{X}$*). Если* $\mathbb{Y} \neq \emptyset$*, то переходим на шаг 2, иначе – на шаг 5.*

Шаг 2. Строим множество $\mathbb{Z} := \{A \in \mathbb{Y} \mid \exists B \in \mathbb{X} : PK(B) \subset U(A)\}$ *(множество тех отношений из* $\mathbb{Y}$*, множество атрибутов которых содержит первичный ключ хотя бы одного отношения из множества* $\mathbb{X}$*. Если* $\mathbb{Z} \neq \emptyset$*, то переходим на шаг 3, иначе - на шаг 4.*

Шаг 3. Добавляем любое отношение из $\mathbb{Z}$ *в* $\mathbb{X}$ *и удаляем это отношение из* $\mathbb{Z}$*. Если ССП выполнено для нового множества отношений* $\mathbb{X}'$*, то требуемое множество построено, ВЫХОД. Иначе повторяем шаг 3, до тех пор пока* $\mathbb{Z} \neq \emptyset$*. Если из* $\mathbb{Z}$ *удалены все отношения, но ССП не выполнено, то переходим на шаг 1.*

Шаг 4. Добавляем любое отношение из $\mathbb{Y}$ *в* $\mathbb{X}$*. Если при этом ССП выполнено для нового множества отношений* $\mathbb{X}'$*, то требуемое множество построено, ВЫХОД. Иначе переходим на шаг 1.*

Шаг 5. Строим множество $\mathbb{Y} := \{A \in \mathbb{M} \setminus \mathbb{X} \mid U(A) \cap \bigcup_{B \in \mathbb{X}} U(B) \neq \emptyset\}$ *(множество отношений, не принадлежащих* $\mathbb{X}$*, хотя бы один атрибут которых содержится во множестве всех атрибутов всех отношений из* $\mathbb{X}$*). Если* $\mathbb{Y} = \emptyset$*, то все отношения схемы базы данных не удовлетворяют ССП, декомпозиция построена неправильно, поэтому ОШИБКА, ВЫХОД. Иначе переходим на шаг 6.*

Шаг 6. Строим множество $\mathbb{Z} := \{A \in \mathbb{Y} \mid \exists B \in \mathbb{X} : PK(B) \subset U(A) \setminus PK(A)\}$ *(множество отношений из* $\mathbb{Y}$*, в неключевые атрибуты которых включается первичный ключ хотя бы одного отношения из* $\mathbb{X}$*). Если* $\mathbb{Z} = \emptyset$*, то добавляем в* $\mathbb{X}$ *любое отношение из* $\mathbb{Y}$*, иначе – добавляем любое отношение из* $\mathbb{Z}$*. Проверяем ССП для нового множества отношений* $\mathbb{X}$*. Если ССП выполнено, то требуемое множество построено, ВЫХОД. Иначе – переход на шаг 1.* ■

Структурно данный алгоритм может быть записан так:

while (не выполнено ССП(X)) do
  $\mathbb{Y} := \{A \in \mathbb{M} \setminus \mathbb{X} \mid PK(A) \subset \bigcup_{B \in \mathbb{X}} U(B)\}$
  if $\mathbb{Y} \neq \emptyset$ *then*
    $\mathbb{Z} := \{A \in \mathbb{Y} \mid \exists B \in \mathbb{X} : PK(B) \subset U(A)\}$
    if $\mathbb{Z} \neq \emptyset$ *then*

while (не выполнено ССП($\mathbb{X}$*) &* $\mathbb{Z} \neq \emptyset$ *) do*
$\mathbb{X} := \mathbb{X} \cup \{A\}, A \in \mathbb{Z}$
$\mathbb{Z} := \mathbb{Z} \backslash \{A\}$
end while
else
$\mathbb{X} := \mathbb{X} \cup \{A\}, A \in \mathbb{Y}$
end if
else
$\mathbb{Y} := \{A \in \mathbb{M} \backslash \mathbb{X} \mid U(A) \cap \bigcup_{B \in \mathbb{X}} U(B) \neq \emptyset\}$
if $\mathbb{Y} \neq \emptyset$ *then*
$\mathbb{Z} := \{A \in \mathbb{Y} \mid \exists B \in \mathbb{X} : PK(B) \subset U(A) \backslash PK(A)\}$
if $\mathbb{Z} \neq \emptyset$ *then*
$\mathbb{X} := \mathbb{X} \cup \{A\}, A \in \mathbb{Z}$
else
$\mathbb{X} := \mathbb{X} \cup \{A\}, A \in \mathbb{Y}$
end if
else
ОШИБКА (ССП не выполнено для всех отношений)
end if
end if
end while

Замечание 1. Термин «любой» здесь означает выбор произвольного отношения (например первого по порядку). Тем не менее осмысленный выбор более «удачного» отношения может значительно уменьшить время поиска.

Замечание 2. Для проверки локального ССП на каждом шаге алгоритма может быть использован алгоритм 2.

Замечание 3. На шаге 1 алгоритма идет отбор тех отношений, добавление которых не приведет к потере информации. Потери информации не может быть, поскольку добавляются условия соединения с отношениями, атрибуты первичного ключа которых уже содержатся во множестве выбранных атрибутов.

Замечание 4. На шаге 2 на выбранные отношения накладывается дополнительное условие, позволяющее в некоторых случаях избежать аномалий проецирования соединения, когда множество всех отношений не удовлетворяет пятой нормальной форме (5НФ). Следствием аномалий проецирования соединений является появление ложных записей в результате запроса, которых нет в предметной области. За примером можно обратиться к [5]. Информационные модели, в которых имеются нарушения 4НФ и 5НФ, встречаются относительно редко, поэтому обычно проектировщики заканчивают процесс нормализации, дойдя до 3НФ. Следовательно, в реальной базе данных возможно появление подобных аномалий, и желательно избегать их.

Замечание 5. На шаге 5 алгоритма выбираются только те отношения, добавление которых к $\mathbb{X}$ может повлиять на выполнение ССП. Добавление отношений, не удовлетворяющих этому условию, не поможет выполнению ССП (это хорошо видно при рассмотрении алгоритма проверки ССП).

Замечание 6. На шаге 6 отбрасываются те отношения, добавление которых может привести к потери информации при соединении отношений. Это отношения, первичный ключ которых заведомо не включается во множество атрибутов отношений множества $\mathbb{X}$ (иначе мы бы не были на шаге 6), но содержит первичный ключ одного из отношений в $\mathbb{X}$. Такие отношения представляют собой разрешение связи «многие ко многим» в реляционной модели и содержат информацию о связи двух объектов в предметной области. При этом если об одном из объектов нет информации в базе данных, то при соединении этих отношений происходит потеря информации.

Множество отношений, полученных в результате работы алгоритма, удовлетворяет локальному ССП, тем самым решает вышеописанную проблему при построении УРЗ. Для завершения формирования отображения необходимо определить правила построения множества X и логического выражения F из (1). В данной работе были рассмотрены только простейшие случаи, когда атрибуты пользовательской модели данных имеют однозначное соответствие в модели данных базы данных. В этом случае построение X и F на основе пользовательского запроса достаточно очевидно. Все атрибуты пользовательской модели однозначно преобразуются в атрибуты модели базы данных, таким образом формируется множество X. Аналогично формируется логическое выражение. Рассмотрение более сложных случаев имеет важное значение, однако это выходит за рамки данной статьи. Для более сложного примера можно обратиться к [3].

На основе разработанных алгоритмов был создан инструментарий, представляющий интерфейс пользователя для доступа к сетевой базе данных. Была протестирована корректная работа алгоритма преобразования пользовательского запроса. Для тестирования программного обеспечения использовалась схема базы данных «Аэропорт» [7], в которой представлена информация о действующих рейсах, обслуживающих их самолетах, пассажирах, а также сведений о регистрации и продажи билетов. Тестирование проводилось для платформ Borland BDE (Paradox 7.0) и Oracle8i.

Заключение

В данной работе был рассмотрен механизм автоматического преобразования запросов, сформулированных в рамках пользовательского представления данных, в реальные запросы к базам данных. Данный механизм является начальным этапом формализации операций преобразования моделей данных. Также разработанный алгоритм, позволяя облегчить реализацию пользовательских интерфейсов к реляционным базам данных, имеет ценное практическое применение.

Для дальнейшего развития средств автоматического преобразования моде-

лей данных важна формализация исходной (внутренней) и целевой (пользовательской) моделей данных. Такая формализация должна включать в себя описание схемы модели в наиболее общем виде, определение ограничений целостности и области допустимых значений. Дальнейшим этапом в развитии является описание и классификация операций преобразования моделей, их строгое обоснование.

Литература

1. Жигалов Влад. *Как нам обустроить поиск в Сети?* // Открытые Системы. 2000. №12. http://www.osp.ru/os/2000/12/053.htm
2. Калиниченко Л.А. *Методы и средства интеграции неоднородных баз данных.* М.: Наука, 1983. С.423.
3. Зыкин С.В. *Формирование пользовательского представления реляционной базы данных с помощью отображений.* // Программирование. 1999. №3, С.70–80.
4. Зыкин С.В. *Межмодельные отображения в базах данных.* Омск: ОмГУ, 2000. http://newasp.omskreg.ru/db/index.html
5. Мейер Д. *Теория реляционных баз данных.* М. : Мир, 1987. С.608
6. Ульман Дж. *Основы систем баз данных.* М. : Финансы и статистика, 1983. С.334.
7. Зыкин С.В. *Базы данных.* // Методические указания и практические задания для студентов математического факультета. Омск : ОмГУ, 1999. С.23.
8. Дейт К. *Введение в системы баз данных, 6-е издание.* К.; М.; СПб.: Издательский дом «Вильямс», 2000. С.782
9. Кузнецов С.Д. *Введение в системы управления базами данных* // СУБД. 1995. №1,2,3,4, 1996. №1,2,3,4,5