УДК 004.047:681.3.01

С.С. Танянский

ХНУРЭ, г. Харьков, Украина, tanyansky_ss@yahoo.com

ФОРМАЛИЗАЦИЯ СРЕДСТВ ДОСТУПА К БАЗАМ ДАННЫХ ПРОИЗВОЛЬНОЙ СТРУКТУРЫ

Рассмотрен основанный на логике подход к формированию запросов в среде неоднородных баз данных. Определен алфавит с расширенной сигнатурой, позволяющий выполнять операции над данными нереляционной структуры. Сформулированы условия, обеспечивающие целостность данных при переходе состояний базы данных. Предложен алгоритм проверки соответствия текущего состояния данных заданным ограничениям.

БАЗА ДАННЫХ, ИНФОРМАЦИОННЫЙ ОБЪЕКТ, СЕМАНТИКА ДАННЫХ, ЛОГИЧЕСКОЕ ПРАВИЛО, ЗАПРОС

Введение

Языки запросов к базам данных (БД) можно разбить на три класса: алгебраические языки, позволяющие выражать запросы средствами специализированных операторов; языки исчисления предикатов, где запросы описывают требуемое множество значений путем спецификации предикатов, которому должны удовлетворять эти значения; логические языки, в которых запрос реализует построение цепочек логических выводов, комбинирующих правила и фактическую информацию для доказательства истинности или отрицания справедливости первоначального утверждения [1, 2].

В системах БД запрос, выраженный с помощью специализированного языка манипулирования данными, должен обрабатываться при использовании наиболее эффективных путей доступа к большим объемам данных для извлечения релевантной информации. Более того, представление запросов и ограничений целостности необходимо представлять в одном формализме и их проверку осуществлять одними и теми же механизмами, что должно позволить обрабатывать более сложные структуры данных.

Запросы на обновления данных могут задаваться либо явно, с помощью соответствующей инструкции языка SQL или средствами алгоритмических алгебр, обобщающих SQL [3], либо неявно посредством формул некоторого логического языка [4]. При этом наличие ограничений целостности, изменения состояния БД может отличаться от изменений, задаваемых самим запросом.

Принципиальный характер этих проблем отмечался различными коллективами исследователей и разработчиков. По мере развития информационных систем (ИС) их качество повышается в силу высокоуровневых языковых средств, используемых системами управления базой данных (СУБД) и разнотипностью используемых в них моделей данных, что отмечено в работах Л.В. Городней [5] и И. Грэхема [6]. С другой стороны, чрезмерный рост перепрограммирования имеющихся прикладных программ вызывается их жесткой зависимостью от типа применяемой СУБД. Таким проблемам посвящены работы Р. Стивенса [7] и Б. Карвина [8].

На практике большая часть данных, совместную работу с которыми необходимо автоматизировать, использует реляционные СУБД. При этом внешняя организация табличного документа не всегда соответствует реляционному отношению. Тогда необходимо использовать дополнительные программные средства для обеспечения возможности корректного доступа к данным из независимых приложений.

Таким образом, определение запросов в виде выражения исчисления с переменными на доменах, а также определение их свойств и интерпретации является актуальной в задачах совместной обработки распределенных данных.

Основной проблемой при формировании запросов неоднородных данных является обобщенное представление объектов БД. В статье используется синтаксис правил логического существования информационных объектов предметной области (ПрО) [9], на основе которых формируется запрос и рассмотрены средства декларативного описания данных.

Целью работы является формализованное представление средств описания языка БД и определение семантики данных, используя введенные языковые конструкции, основанные на логических выражениях предикатов первого порядка. Для обеспечения корректности выполнения запросов предложен алгоритм, реализующий вычисление состояния БД, удовлетворяющего заданному множеству ограничений.

1. Расширения языка исчисления с переменными на доменах

Используя результаты исследования литературных источников [10, 11] и реализованных практических задач организации доступа к данным средствами логического программирования, выражения исчисления предикатов, положенные в основу языка запросов, определим следующим образом.

Пусть $O = \{o_1,...,o_n, o_{n+1},...\}$ – произвольное множество символов информационных объектов

(в терминах исчисления – символов). Необходимо отметить, что согласно обозначениям, введенным в [9], элементы множества O будем записывать в виде $\{l_{o_1},...,l_{o_n}, l_{o_{n+1}},...\}$ или $\{l_{o_1,...,o_n,o_{n+1},...}\}$. Зададим алфавит $\mathcal{A}$ как объединение множеств вида:

$$\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3 \cup \mathcal{A}_4 \cup \mathcal{A}_5 \cup \mathcal{A}_6, \qquad (1)$$

где $\mathcal{A}_1 = \{l^i_{o_1},...,l^i_{o_n}, l^i_{o_{n+1}},...\}$ – предметные переменные (информационные объекты рассматриваемой ПрО) и $\{i \in I \mid I = \overline{1,\infty}\}$; $\mathcal{A}_2 = \{a^i_1, b^i_1, c^i_1, a^i_2, b^i_2, c^i_2,...\}$ – предметные константы (параметры поиска данных) и $\{i \in I \mid I = \overline{1,\infty}\}$; $\mathcal{A}_3 = \{\mathcal{P}^{1,2,...,n}_j\}$ – n-местные предикатные символы и $\{i_1, i_2,..., i_{n_j} \in I, j \in J \mid J = \overline{1,\infty}\}$; $\mathcal{A}_4 = \{$‘=’, ‘≠’, ‘<’, ‘>’, ‘≤’, ‘≥’$\}$ – символы сравнения; $\mathcal{A}_5 = \{$‘∧’, ‘∨’, ‘¬’, ‘∃’, ‘∀’, ‘↵’, ‘F’$\}$ – логические символы; $\mathcal{A}_6 = \{$‘,’, ‘(’, ‘)’$\}$ – вспомогательные символы.

В дальнейших выкладках некоторые индексы будут опущены, если в них не будет явной необходимости. Объединение первых трех множеств, в обозначении $\mathcal{G} = \mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3$, будем называть сигнатурой формального языка на схеме БД. Используя заданную сигнатуру, определим атомы формул в виде многочлена следующим образом:

1. Выражения вида $l^i_o \; \Theta \; a_i$ или $l^i_o \; \Theta \; l^k_o$, где Θ – символ сравнения, есть условие.

2. Атомом является условие или многочлен, взятый в скобки.

3. Атом α является многочленом.

4. Если α есть многочлен, а α_1 является атомом, то выражения $\alpha \wedge \alpha_1$, $\alpha \vee \alpha_1$, $\alpha \wedge \neg\alpha_1$, $\alpha \vee \neg\alpha_1$ также являются многочленами.

Все входящие в многочлен переменные считаются свободными (здесь следует отметить, что термин “свободные” взят из теории исчисления и не эквивалентен такому же термину, используемому в разделах 3 и 4, для обозначения информационных объектов из множества O). Формулы определим индуктивно следующим образом:

1. Предикат $\mathcal{P}^{1,2,...,n}_j(l^1_{o_1}, l^1_{o_2},..., l^1_{o_n})$ есть формула, при этом все переменные в ней свободные.

2. Если β - формула со свободными переменными, а α – многочлен, свободные переменные которого входят в множество свободных переменных β, то $\alpha \wedge \beta$ также формула с теме же свободными переменными, что и β.

3. Если β_1 и β_2 две формулы, имеющие одинаковые свободные переменные, то выражения $(\beta_1 \vee \beta_2)$ и $(\beta_1 \wedge \neg\beta_2)$ также формулы с тем же множеством свободных переменных.

4. Если β_1 и β_2 две формулы, не имеющие свободных переменных или со свободными переменными, пересечение которых не пусто, то выражения $(\beta_1 \wedge \beta_2)$ также формулы со свободными переменными, соответствующими β_1 и β_2.

5. Пусть $\varepsilon = l^1_{o_1}, l^1_{o_2},..., l^1_{o_n}$ – свободные переменные формул β_1 и β_2, причем в β_1 других переменных нет, а в β_2 могут присутствовать и другие свободные переменные, отличные от ε. Тогда выражения вида $\exists\varepsilon(\beta_1)$, $\exists\varepsilon(\beta_2)$, $\forall\varepsilon(\beta_1 \leftarrow\!\!\shortmid\, \beta_2)$ и $\forall\varepsilon(\beta_2 \leftarrow\!\!\shortmid\, \beta_1)$ есть формулы, где переменные из ε называются связными, а остальные переменные, которые входят как связно, так и свободно в β_1 и β_2, остаются такими же во вновь образованных формулах.

6. Пусть α – многочлен, β – формула, а $\varepsilon = l^1_{o_1}, l^1_{o_2},..., l^1_{o_n}$ – свободные переменные из α, причем элементы из ε включаются в множество свободных переменных β. Исходя из этого, выражения вида $\forall\varepsilon(\alpha \leftarrow\!\!\shortmid\, \beta)$ и $\forall\varepsilon(\beta \leftarrow\!\!\shortmid\, \alpha)$ есть формулы, у которых переменные из ε связные, а остальные свободные или связные переменные из β остаются такими же во вновь образованных формулах.

7. Если $l^i_{o_j}$ – свободная переменная формулы β, a^i_j – константа, Θ – символ сравнения, то выражение вида $\mathcal{F}(l^i_{o_j}(\beta)) \; \Theta \; a^i$ является формулой, у которой переменная $l^i_{o_j}$ связная, а остальные свободные или связные переменные из β остаются такими же во вновь образованной формуле.

Содержательная интерпретация формул исчисления предикатов при интеграции БД определяет сигнатуру $\mathcal{G}$ следующим образом:

1. Множество $\mathcal{A}_1$ включает все информационные объекты $\{l^i_{o_j}\}$ рассматриваемой ПрО.

2. В $\mathcal{A}_2$ входят множества значений из множества O, которые используются как параметры поиска значений, представленных в БД.

3. Каждому значению от 1 до n множества информационных объектов O ставится в соответствие $\sum_{i=1}^{n} C^i_n = 2^{n-1}$ (символ “C” – обозначает сочетания из n по i) символов предикатов. При этом, если данные не имеют реляционной структуры, то каждому информационному объекту ставится в соответствие $\sum_{j=1}^{m}\left(2^{n_j-1}\right)$ информационных объектов, где m – число различных последовательностей вывода возможных информационных объектов из O, n_j – число информационных объектов, принадлежащих некоторой последовательности вывода.

Множество всех правильных формул, которые можно построить с помощью алфавита $\mathcal{A}$, обозначим $\beta_{\mathcal{A}}$. Формулы из $\beta_{\mathcal{A}}$ будем разделять на замкнутые и открытые. Замкнутыми будем называть формулы, не содержащие свободных переменных, а открытыми – формулы у которых не все переменные связаны кванторами ‘∃’, ‘∀’, ‘$\mathcal{F}$’ ($\mathcal{F}$ – функциональный квантор, определяющий какую-либо агрегатную функцию). Замкнутые формулы представляют собой высказывания, которые являются истинными или ложными на множестве O. Открытым формулам соответствует множество истинности формулы в обозначении $\mathcal{L}(\beta)$.

При введенной интерпретации смысл правильных формул можно представить следующим образом:

1. Если β_1 и β_2 – формулы с одинаковыми свободными переменными, тогда выполняются тождества:

a. $\mathcal{L}(\beta_1 \vee \beta_2) = \mathcal{L}(\beta_1) \cup \mathcal{L}(\beta_2)$;

b. $\mathcal{L}(\beta_1 \wedge \neg\beta_2) = \mathcal{L}(\beta_1) - \mathcal{L}(\beta_2)$;

c. $\mathcal{L}(\beta_1 \wedge \beta_2) = \mathcal{L}(\beta_1) \cap \mathcal{L}(\beta_2)$.

2. Если β_1 – формула, содержащая свободные переменные $\mathcal{V}_1$, β_1 – формула, содержащая свободные переменные $\mathcal{V}_2$ и $\mathcal{V}_1 \subseteq \mathcal{V}_2$, то $\mathcal{L}(\beta_1 \wedge \beta_2) = \mathcal{L}(\beta_1) \times \{ l_o^{i_j} \} \cap \mathcal{L}(\beta_2)$, где $i \in I \mid I = \overline{1,\infty}, j = \overline{1,k}$ и $\{ l_o^{i_j} \} = V_2 - V_1$. Этот факт говорит о том, что $\mathcal{L}(\beta_1 \wedge \beta_2)$ будет содержать информационные объекты из $\mathcal{L}(\beta_1)$, которые также присутствуют и в $\mathcal{L}(\beta_2)$.

3. Если β – некоторая формула, α – многочлен, в котором свободные переменные $\mathcal{V}$ входят в β, тогда $\mathcal{L}(\beta \wedge \alpha)$ включает те информационные объекты из $\mathcal{L}(\beta) \in \mathcal{V}$, для которых многочлен α принимает значение истинно.

4. Множество истинных формул вида $\exists\varepsilon(\beta)$, использующих квантор существования, является проекцией множества истинности β на несвязанные квантором $\exists$ предметные переменные.

5. Квантор всеобщности $\forall$ в формулах может иметь двоякий смысл:

a. Ограниченный квантор всеобщности вида $\forall\varepsilon \in \beta_1(\beta_2)$ в эквивалентной записи $\forall\varepsilon(\beta_2 \lrcorner \beta_1)$. Истинностью таких формул является подмножество значений $v = \mathcal{L}(\beta_2)$ множества несвязанных квантором свободных переменных β_2. Аналогичный смысл имеют также формулы вида $\forall\varepsilon(\beta \lrcorner \alpha)$.

b. Ограниченный квантор всеобщности в формулах с квантором существования вида $\forall\varepsilon \in \exists v(\beta_2)(\beta_1 \lrcorner \beta_2)$ или $\forall\varepsilon(\beta_1 \lrcorner \beta_2)$. Истинностью таких формул является подмножество значений $\mathcal{L}(\beta_2)$ множества v несвязанных квантором свободных переменных β_2. Аналогичный смысл имеют также формулы вида $\forall\varepsilon(\alpha \lrcorner \beta)$.

Использование ограниченного квантора всеобщности обеспечивает определение соответствующего формуле множества истинности.

6. В качестве функционального квантора $\mathcal{F}$ используются какие-либо функции (как стандартные, например min, max, sum и др., так и сформулированные для индивидуальных вычислений). Функциональный квантор связывает аргумент функции $\mathcal{F}$ в формулах $\mathcal{F}(l_{o_j}^i(\beta)) \Theta a^i$. Если β содержит более одной свободной переменной, то множество истинности с функциональным квантором представляет собой подмножество $\mathcal{L}(\beta)$ множества несвязанных квантором $\mathcal{F}$ переменных.

7. Множество истинных формул вида $\mathcal{F}(l_{o_j}^i(\alpha \lrcorner \beta))$, использующих функциональный квантор. Истинностью таких формул является новый многочлен α, состоящий из несвязанных квантором свободных переменных в формуле β).

Использование исчисления предикатов позволяет создать гибкую схему формирования сложных запросов к БД, обеспечивающую построение посредством логических связок, кванторов и скобок многоуровневых условий, в которых формулы выражают условия идентификации данных интегрированной БД. Особенностью таких запросов является возможность проведения формального контроля правильности запроса, основанного на соответствии используемых в формулах переменных и констант определенным свойствам схемы БД.

Определенные выше описания формул могут быть применены не только к данным реляционной структуры, но для работы с данными иерархической или сетевой структуры необходимо ввести дополнительные атомы и фиктивные переменные, связные иерархическими зависимостями.

При обработке данных нереляционной структуры необходимо обеспечить их целостность. Очевидно, что традиционными средствами реляционной модели обеспечить корректность таких данных не представляется возможным. С целью поддержки целостности и согласованности интегрированных БД введем дополнительные ограничения на семантику данных.

2. Поддержка логической целостности в таблицах иерархической и сетевой структур

Определим характеристики компонентов ограничений целостности для формирования формул исчисления. Ограничения, которые определяют допустимые расширения БД и объектов БД, называются статическими. Ограничения, которые регламентируют доступные переходы БД между допустимыми состояниями, называются динамическими [10]. Динамические спецификации, как правило, операционно-ориентированные, то есть обосновывают выполнение той или иной конкретной операции.

Простым является ограничение, которое строится с использованием одноместных предикатов арифметического сравнения:

$$\mathcal{P}(l_{o_1}^i, ..., l_{o_n}^i) \supset l_{o_1}^i \; \Theta \; \alpha_1 \wedge ... \wedge l_{o_n}^i \; \Theta \; \alpha_n, \qquad (2)$$

где $\mathcal{P}(l_{o_1}^i, ..., l_{o_n}^i)$ – n-местный предикат; $l_{o_i}^i$ – предметные переменные; a_i – предметные константы; $\Theta = \{\text{‘=’}, \text{‘}\neq\text{’}, \text{‘<’}, \text{‘>’}, \text{‘}\leq\text{’}, \text{‘}\geq\text{’}\}$ – символы сравнения, символ ‘$\supset$’ – является знаком импликации.

Более сложные виды ограничений могут быть выражены обязательной поддержкой ключей, первичного и внешнего. Первичный ключ определим как предикат $\mathcal{PK}(R, l_{o_i}^i)$, где R – имя таблицы, уникально определяющее множество предметных переменных $l_{o_1}^i, ..., l_{o_n}^i$, соответствующих атрибутам R. Выражение, определяющее первичный ключ, имеет вид:

$$\begin{aligned} \mathcal{PK}(R, l_{o_i}^i) \equiv \forall l_{o_i}^1 \, \forall l_{o_i}^2 \, \forall l_{o_i}^3 \, (R(l_{o_1}^1, l_{o_2}^1, ..., l_{o_n}^1, \\ l_{o_1}^2, l_{o_2}^2, ..., l_{o_n}^2) \wedge (l_{o_1}^1, l_{o_2}^1, ..., l_{o_n}^1, l_{o_1}^3, l_{o_2}^3, ..., \\ l_{o_n}^3) \supset (l_{o_1}^2 = l_{o_1}^3) \wedge (l_{o_2}^2 = l_{o_2}^3) \wedge ... \wedge \\ \wedge (l_{o_n}^2 = l_{o_n}^3) \wedge (l_{o_1}^1 \neq a_1) \wedge (l_{o_2}^1 \neq a_2) \wedge ... \wedge (l_{o_n}^1 \neq a_n)). \end{aligned} \qquad (3)$$

Ограничение на внешний ключ определяется как совпадение значений связных атрибутов двух отношений. Представим внешний ключ в виде предиката $FK(R_1, R_2, l^i_{o_i})$, где переменные $l^i_{o_1}, l^i_{o_2}, ..., l^i_{o_n}$ таблицы R_1 ссылаются на атрибуты отношения R_2, то есть совокупность этих переменных в отношении R_2 является внешним ключом. Выражение для описания внешнего ключа имеет вид:

$$FK(R_1, R_2, l^i_{o_i}) \equiv \forall l^1_{o_i} \exists l^2_{o_i} \ (R_1(l^1_{o_1}, l^1_{o_2}, ..., l^1_{o_n}) \supset$$
$$\supset((R_2(l^2_{o_1}, l^2_{o_2}, ..., l^2_{o_n}) \wedge (l^1_{o_1} = l^2_{o_1}) \wedge (l^1_{o_2} = l^2_{o_2}) \wedge$$
$$\wedge ... \wedge (l^1_{o_n} = l^2_{o_n}) \wedge PK(R_2, l^2_{o_1}, l^2_{o_2}, ..., l^2_{o_n}))) \vee \quad (4)$$
$$\vee ((l^1_{o_1} = a_1) \wedge (l^1_{o_2} = a_2) \wedge ... \wedge (l^1_{o_n} = a_n)).$$

Рассмотренные ограничения являются структурными, то есть определяемыми при проектировании БД, и зависят от множества заданных зависимостей (для реляционной модели – функциональных и многозначных зависимостей). Под явными ограничениями целостности будем понимать ограничения, заданные пользователем. Среди таких ограничений можно выделить ограничение на домен (явно неподдерживаемое в реляционной модели данных). К ним относится, например ограничение, которое устанавливает значение, определенное по умолчанию, в случае если ни одно из вводимых значений не соответствует ограничению. Соответствующее выражение имеет вид:

$$CH(R, l^i_{o_i}, a_j) \equiv \forall l^i_{o_i} \ (R(l^i_{o_i}) \wedge a_i \notin D_i \supset$$
$$\supset R(< l^i_{o_i}, D_i(a_j) >)), \quad (5)$$

где a_i – предметная константа (вносимое значение); $< l^i_{o_i}, D_i(a_j) >$ – значение $a_j \in D_i$ информационного объекта $l^i_{o_i}$, устанавливаемое по умолчанию, заметим, что $O = \bigcup_{i=1}^{n} D_i$ и $l^i_{o_i} \in O$.

Необходимо отметить, что частным случаем такого ограничения является внесение пустого значения в $l^i_{o_i}$ (такой прием относится к пользовательским допущениям).

Ограничения идентификаторов объектов формулируются в терминах зависимостей атрибутов. В терминах введенных информационных объектов ограничение на зависимость обладает следующим свойством: для информационного объекта $l^1_{o_i}$ не существует информационного объекта $l^2_{o_i}$, такого что $< l^1_{o_i}, D_1(a_i) > = < l^1_{o_i}, D_2(b_i) >$, $< l^2_{o_i}, D_1(a_i) > \neq < l^2_{o_i}, D_2(b_i) >$.

Предикат для описания зависимости обозначим как $FD(R, l^1_{o_i}, l^2_{o_i})$, для которого выражение имеет вид:

$$FD(R, l^1_{o_i}, l^2_{o_i}) \equiv \forall l^1_{o_i} \ (R(l^1_{o_i}) \supset \neg \exists l^2_{o_i} \ (R(l^2_{o_i}) \wedge$$
$$\wedge < l^1_{o_i}, D_1(a_i) > = < l^1_{o_i}, D_2(b_i) > \wedge < l^2_{o_i}, \quad (6)$$
$$D_1(a_i) > \neq < l^2_{o_i}, D_2(b_i) >)).$$

Рассмотрим два основных типа преобразования атрибутов: 1:1 и 1:*m*. Кроме этого можно рассматривать типы *m*:1 и *m*:*m* [12]. Преобразование 1:1 имеет место, когда атрибуту целевой таблицы строго соответствует атрибут таблицы источника. Преобразование 1:*m*, *m*:1 возникает тогда, когда существует функциональная зависимость между атрибутом целевой таблицы и несколькими атрибутами таблицы источника или нескольких атрибутов целевой таблицы и атрибутом таблицы источника соответственно. Преобразование *m*:*m* возникает при необходимости одновременного получения нескольких атрибутов целевой таблицы из нескольких атрибутов таблицы источника.

Использование информации об ограничениях, которые накладываются на отношения, позволяет получить дополнительную информацию для преобразования БД. Информация о первичном ключе дает возможность определить перечень полей для определения операций добавления, удаления или модификации целевой таблицы. Информация об ограничениях ссылочной целостности позволяет корректно реализовывать соответствие между несколькими таблицами-источниками и целевой таблицей. Использование ограничений по умолчанию дает возможность определить несформированные значения атрибутов первичного ключа.

Определим предикат $IN(R_1, R_2, l^i_{o_i})$ как последовательность ссылочных ограничений между таблицами вида:

$$IN(R_1, R_2, l^i_{o_i}) \equiv \exists R_1, \exists R_2, \forall l^i_{o_i} \ (R_1,$$
$$FK(R_1, R_2, l^i_{o_i}), R_2) \supset IN(R_1, R_2, l^i_{o_i}), \quad (7)$$

где R_1 и R_2 – таблицы; $l^i_{o_i}$ – связный информационный объект.

Пусть необходимо преобразовать схему БД, то есть построить переход типа $R_1 \leftarrow R_2$, представленный на рис. 1(а) в схему, представленную на рис. 1(б).

R_1

$l^1_{o_1}$	$l^1_{o_2}$	$l^1_{o_3}$
a^1_1	a^2_1	a^3_1
a^1_2	a^2_2	a^3_2

а) таблица реляционной структуры

R_2

$l^2_{o_1}$	$l^2_{o_4}$	$l^2_{o_5}$	$l^2_{o_6}$	…
a^1_1	a^3_1			…
a^1_2		a^3_2		…

б) таблица нереляционной структуры

Рис. 1. Схемы БД, для которых необходимо построить переход $R_1 \leftarrow R_2$

Отметим, что выделены атрибуты первичного ключа. Для перехода $R_1 \leftarrow R_2$ необходимо построить соответствие "информационный объект" ← "значение" и "значение" ← "информационный объект" $l^1_{o_3} \leftarrow a^3_1$, $l^1_{o_3} \leftarrow a^3_2$ и соответствия атрибутов $a^2_1 \leftarrow l^2_{o_4}$, $a^2_2 \leftarrow l^2_{o_5}$. Для обеспечения целостности при таком переходе необходимо воспользоваться свойством предиката PK,

Таким образом, предикатное выражение для преобразования схем R_2 в R_1 будет иметь вид:

$$\exists R_1 \exists R_2 \forall l^2_{o_i} (R_1(l^1_{o_i}, l^2_{o_i}) \supset R_2(l^1_{o_i}) \wedge (l^1_{o_2} = l^2_{o_4}) \wedge \wedge (l^1_{o_2} = l^2_{o_5}) \wedge (l^1_{o_2} = l^2_{o_6}) \wedge ... \wedge (l^1_{o_2} = l^2_{o_n}) \wedge \wedge (a^2_q \neq a^2_j)); i = \overline{1,n}; q, j = \overline{1,m}. \quad (8)$$

При формировании целевой таблицы выполняется объединение переменных, определяющих атрибуты таблиц - источников. С другой стороны, при формировании значений ключевых атрибутов целевой таблицы необходимо использовать дополнительную информацию, которая извлекается из ограничений, накладываемых на таблицы – источник. В рассматриваемом примере атому $(a^2_q \neq a^2_j)$ всегда будет соответствовать значение *True*. Это определяется тем, что атрибуты в одной таблице должны иметь различные имена и, следовательно, при преобразовании $a^2_1 \leftarrow l^2_{o_4}$ и $a^2_2 \leftarrow l^2_{o_5}$ пара информационных объектов $l^2_{o_4}$, $l^2_{o_5}$ в качестве значений информационного объекта $l^1_{o_2}$ в таблице R_1 повторяться не будет.

Решение задачи реализации перехода $R_1 \leftarrow R_2$ можно осуществить с помощью написания запроса на каком-либо языке программирования либо записать в терминах реляционного исчисления. Второй подход является более эффективным, так как носит описательно-формализованный характер и гарантирует корректность результата запросов.

Для примера, представленного на рис. 1, сформулируем запрос на выборку для каждой таблицы средствами реляционного исчисления с переменными на доменах. Для таблицы R_1 выражение имеет вид (9), а для таблицы R_2 – вид (10):

$$\{ p^{l_{o_i}} \mid (\exists l^1_{o_1}, l^1_{o_2}, l^1_{o_3})(R_1(l^1_{o_1}, l^1_{o_2}, l^1_{o_3}) \wedge \wedge p^{l_{o_1}} = l^1_{o_1} \wedge p^{l_{o_2}} = l^1_{o_2} \wedge p^{l_{o_3}} = l^1_{o_3})\}, \quad (9)$$

$$\{ p^{l_{o_i}} \mid (\exists l^2_{o_1}, l^2_{o_4}, l^2_{o_5}, l^2_{o_6}, ..., l^2_{o_n}) (R_2(l^2_{o_1}, l^2_{o_4}, l^2_{o_5}, l^2_{o_6}, ..., l^2_{o_n}) \wedge p^{l_{o_1}} = l^2_{o_1} \wedge p^{l_{o_2}} = = l^2_{o_4} \wedge p^{l_{o_3}} = l^2_{o_5} \wedge p^{l_{o_4}} = l^2_{o_6} \wedge ... \wedge p^{l_{o_n}} = l^2_{o_n})\}, \quad (10)$$

где $p^{l_{o_i}}$ – свободная переменная, определяющая результат запроса.

Для одновременной обработки данных объединим формулы, описывающие доступ к данным таблиц R_1 и R_2, с учетом свойств предиката (9) и правила преобразования (10). Получаем выражение:

$$\{ p^{l_{o_i}} \mid (\exists l^1_{o_1}, l^1_{o_2}, l^1_{o_3}) ((R_1(\exists l^1_{o_1}, l^1_{o_2}, l^1_{o_3}) \wedge \wedge (\exists l^2_{o_1}, l^2_{o_4}, l^2_{o_5}, l^2_{o_6}, ..., l^2_{o_n}) (R_2(l^2_{o_1}, l^2_{o_4}, l^2_{o_5}, l^2_{o_6}, ..., l^2_{o_n}) \wedge \wedge (p^{l_{o_1}} = l^1_{o_1} \wedge p^{l^1_{o_2}} = l^1_{o_2} \wedge p^{l_{o_3}} = l^1_{o_3}) \wedge (p^{l_{o_1}} = l^2_{o_1} \wedge p^{l^1_{o_2}} = l^2_{o_4} \wedge p^{l_{o_3}} = l^2_{o_5} \wedge p^{l_{o_4}} = l^2_{o_6} \wedge ... \wedge p^{l_{o_n}} = l^2_{o_n}) \wedge \wedge (l^1_{o_2} = l^2_{o_4}) \wedge (l^1_{o_2} = l^2_{o_5}) \wedge (l^1_{o_2} = l^2_{o_6}) \wedge ... \wedge \wedge (l^1_{o_2} = l^2_{o_n}) \wedge (a^2_q \neq a^2_j))\}; i = \overline{1,n}; q, j = \overline{1,m}. \quad (11)$$

Необходимо отметить, что в R_2 могут присутствовать неопределенные значения. Для поддержки корректного преобразования данных воспользуемся предикатом $\mathcal{CH}$ (5).

В рассматриваемом примере модифицируем предикат, исключив кортежи с неопределенными значениями.

Поддержка ограничений целостности на зависимости, как отмечалось выше, заключается в интерпретации предиката $\mathcal{FD}$, то есть в вычислении его истинного значения. Пусть $R = (Sch, F)$ – таблица БД с количеством строк m, где Sch – схема таблицы и $Sch \subseteq O$, F – множество функциональных зависимостей вида $F = \{ l^1_{o_i} \rightarrow l^2_{o_i} \}$, $i, j = \overline{1,n}$, $l^1_{o_i}$, $l^2_{o_j} \subseteq O$. Необходимо проверить, удовлетворяет ли состояние R некоторой зависимости $l^1_{o_k} \rightarrow l^2_{o_k} \in F, k = \overline{1,p}$.

Таким образом, выражение (11) примет вид:

$$p^{l_{o_i}} \mid (\exists l^1_{o_1}, l^1_{o_2}, l^1_{o_3})((R_1(\exists l^1_{o_1}, l^1_{o_2}, l^1_{o_3}) \wedge \wedge (\exists l^2_{o_1}, l^2_{o_4}, l^2_{o_5}, l^2_{o_6}, ..., l^2_{o_n}) (R_2(l^2_{o_1}, l^2_{o_4}, l^2_{o_5}, l^2_{o_6}, ..., l^2_{o_n}) \wedge (p^{l_{o_1}} = l^1_{o_1} \wedge p^{l^1_{o_2}} = l^1_{o_2} \wedge p^{l_{o_3}} = l^1_{o_3}) \wedge (p^{l_{o_1}} = = l^2_{o_1} \wedge p^{l^1_{o_2}} = l^2_{o_4} \wedge p^{l_{o_3}} = l^2_{o_5} \wedge p^{l_{o_4}} = l^2_{o_6} \wedge ... \wedge p^{l_{o_n}} = = l^2_{o_n}) \wedge \wedge (l^1_{o_2} = l^2_{o_4}) \wedge (l^1_{o_2} = l^2_{o_5}) \wedge (l^1_{o_2} = l^2_{o_6}) \wedge ... \wedge \wedge (l^1_{o_2} = l^2_{o_n}) \wedge (a^2_q \neq a^2_j) \wedge l^1_{o_2} \neq \varnothing)\}; i = \overline{1,n}; q, j = \overline{1,m}. \quad (12)$$

АЛГОРИТМ. Проверка состояния БД.

ВХОД. R – таблица базы данных, функциональная зависимость $l^1_{o_k} \rightarrow l^2_{o_k} \in F$.

ВЫХОД. Множество строк $\{Str_i\}$ $(i = \overline{1,n} \mid n \leq m)$, удовлетворяющих $l^1_{o_k} \rightarrow l^2_{o_k}$ (при этом значение логической переменной α соответствует *True*).

МЕТОД.

ШАГ 1. Устанавливаем значение логической переменой $\alpha = True$ и просматриваем строки $\{Str_i\}$, $i = \overline{1,m}$.

ШАГ 2. Для каждого значения левого элемента $Str^1_i(l^1_{o_k})$ просматриваем значения правого элемента строки $Str_j(l^2_{o_k}), j = \overline{1,m-1}$.

ШАГ 3. Если $Str_i(l^2_{o_k}) = Str_j(l^2_{o_k})$, то выполняется проверка $Str_i(l^2_{o_k}) = Str_j(l^2_{o_k})$.

ШАГ 4. Если условие шага 3 выполняется, то удаляем Str_j из области перебора и продолжаем просмотр.

ШАГ 5. Если условие шага 3 не выполняется, то выходное значение логической переменной определяется как $\alpha = False$ и шаг 7.

ШАГ 6. При успешном окончании просмотра Str_i, то есть когда $i = m$, выходное значение устанавливается как $\alpha = True$ и шаг 7.

ШАГ 7. Алгоритм закончен.

Как видно, в наихудшем случае придется выполнить $m+(m-1)+...+1 = m(m+1)/2$ обращений. Для проверки всех зависимостей из F алгоритм необходимо выполнить p раз. Таким образом, вычислительную сложность поддержки множества

зависимостей, в среднем, можно оценить как $p \cdot m \cdot (m+1)/2$ обращений.

Таким образом, наиболее эффективным путем разрешения проблем преобразования данных является оснащение систем управления БД специальными языковыми средствами поддержки ограничений, способными реализовать требования пользователей, определенные в концептуальной схеме БД.

Выводы

При использовании различных источников информации одни и те же сущности реального мира могут моделироваться с помощью различных структур и типов данных. При попытке интеграции разнородных источников необходимо либо приводить данные к общей структуре, либо в результате интеграции данные будут иметь неправильную структуру. Приведение к общей структуре большого количества источников может быть невозможным, или обобщенная структура будет очень сложной, а ее использование неэффективным.

На практике большая часть данных, совместную работу с которыми необходимо автоматизировать, использует реляционные СУБД. При этом внешняя организация табличного документа не всегда соответствует реляционному отношению. Тогда необходимо использовать дополнительные средства для обеспечения возможности корректного доступа к данным из независимых приложений.

Средства, определяющие корректность модификации данных выражаются в виде ограничений, накладываемых на значения, допустимые в некотором состоянии БД. При модификации структуры данных необходимо сохранить начальные ограничения или так их изменить, чтобы не противоречить семантике данных, то есть ограничения исходного и модифицированного состояния должны быть эквивалентны.

Исследован класс ограничений целостности, который определяет допустимые транзакции, то есть переход из одного состояния БД в другое. К классу рассматриваемых модификаций добавлены операторы замены одних фактов другими, не сводимые непосредственно к операторам добавления и удаления.

Формально определен язык запросов как выражения исчисления с переменными на доменах. Определены свойства и интерпретация формул языка. Показано, что предложенная формулировка языка может быть применена не только к данным реляционной структуры.

Определены характеристики компонентов ограничений целостности для формирования формул исчисления. Заданы ограничения на допустимые значения и ключи, а также введены дополнительные ограничения, явно не поддерживаемые в реляционных моделях, ограничения по ссылке и ограничение на домен. Разработан алгоритм проверки соответствия состояния БД заданным ограничениям при модификации данных.

Исследована практическая значимость полученных результатов, определены критерии использования неоднородных (не соответствующих реляционным требованиям) отношений локальных БД. Использование рассмотренных способов построения запроса позволяет средствами традиционных реляционных СУБД организовать доступ к неоднородным данным.

Дальнейшее исследование в области неоднородных БД следует направить на определения ограничений для корректного перехода состояний БД при интеграции логически независимых информационных систем.

Список литературы: 1. *Ульман Дж.* Основы систем баз данных [Текст] / Дж. Ульман. – М. : Финансы и статистика, 1983. – 334 с. **2.** *Черри С.* Логическое программирование и базы данных [Текст] / С. Черри, Г. Готлоб, Л. Танка. – М. : Мир, 1992. – 352 с. **3.** *Abiteboul S.* Updates a new Frontier [Text] / S. Abiteboul S. // Proc. of the Second International Conference on the Theory of Databases. – ICDT'88. – 1988. – № 326. – P.1 – 18. **4.** *Katsuno H.* Propositional knowledge base revision and minimal change [Text] / H. Katsuno, A.O. Mendelson. // Artifscsal Intelligence. - 1991. – V. 52. – P. 253 – 294. **5.** *Городняя Л.В.* Основы функционального программирования. [Текст] / Л.В. Городняя - М. : Изд-во "Интернет - университет информационных технологий – INTUIT.ru", 2004. – 280 с. **6.** *Грэхем И.* Объектно-ориентированные методы. Принципы и практика = Object–Oriented Methods: Principles & Practice. [Текст] / И. Грэхем – 3-е изд. – М. : «Вильямс», 2004. – 880 с. **7.** *Стивенс Р.* Программирование баз данных [Текст] / Р. Стивенс. – Изд-во «Бином-Пресс», 2003. – 384 с. **8.** *Карвин Б.* Программирование баз данных SQL. Типичные ошибки и их устранение [Текст] / Б. Карвин. – Изд-во «Рид Групп», 2012. – 336 с. **9.** *Танянский, С.С.* Характеристические свойства объектов информационных систем [Текст] / С.С. Танянский. // "Штучній інтелект" науковий журнал – 2007. – № 1. – С. 78-89. **10.** *Калиниченко Л.А.* Методы и средства интеграции неоднородных баз данных [Текст] / Л.А. Калиниченко. – М. : Наука, 1983. – 424 с. **11.** *Маркова Л.И.* Метод построения блочно-ациклической схемы реляционной базы данных [Текст] / Л.И. Маркова, С.С. Танянский, Д.А. Руденко // Глобальные информационные системы. Проблемы и тенденции развития: материалы I междунар. научн. конф. – Харьков – Туапсе, ХНУРЭ, 2006. – С.101 – 102. **12.** *Дейт К.Дж.* Введение в системы баз данных [Текст] / К.Дж. Дейт. – М. : Диалектика, 1998. – 784 с.

Поступила в редколлегию 20.11.2012

УДК 004.047:681.3.01

Формалізація засобів доступу до баз даних довільної структури / С.С. Танянський // Біоніка інтелекту: наук.-техн. журнал. – 2013. – № 1 (80). – С. 82-87.

Досліджено й запропоновано формальні засоби формування запитів до даних нереляційної структури. Визначено основні властивості і характеристики станів бази даних, що інтегруються. Запропоновано алгоритм перевірки відповідності бази даних заданим обмеженням.

Іл. 1. Бібліогр.: 12 найм.

UDK 004.047:681.3.01

Formalization means of access to the database of any structure / S. Tanyansky // Bionics of Intelligense: Sci. Mag. – 2013. – № 1 (80). – P. 82-87.

Explore and propose formal means querying relational data, not structure. The basic properties and characteristics of the states integrable database. The algorithm for checking the conformity database defined constraints.

Fig. 1. Ref.: 12 items.