УДК 004.047:681.3.01

С.С. Танянский

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

ЯЗЫКОВЫЕ СРЕДСТВА ДЕКЛАРАТИВНОГО ОПИСАНИЯ НЕОДНОРОДНЫХ БАЗ ДАННЫХ

Рассмотрен основанный на логике подход к описанию неоднородных баз данных. Сформулированы свойства логических правил, при которых вывод результата будет достигаться за конечное число итераций, то есть запрос буде выполнен корректно. Предложены алгоритмы корректного вычисления результата запроса по заданному экстенсионалу базы данных.

БАЗА ДАННЫХ, ИНФОРМАЦИОННЫЙ ОБЪЕКТ, СЕМАНТИКА ДАННЫХ, ЛОГИЧЕСКОЕ ПРАВИЛО

Введение

Современные системы управления базами данных (СУБД), обладая мощным арсеналом средств решения типичных задач, таких, как задачи загрузки и обновления БД, простой поиск по условиям, задачи прямого счета (суммирование, подсчет количества, средних значений и т.п.), имеют в то же время ограниченные возможности по решению сложных задач (в частности, в неоднородных структурах). Для реализации последних в рамках традиционных технологий пользователь вынужден, во-первых, осуществлять глубокую проработку задачи, связанную не только с описанием структуры и правил функционирования системы, но и с явным описанием процедур поиска решения; во-вторых — эти процедуры, описанные на каком-либо формальном языке, необходимо “перевести” на входной язык СУБД, что само по себе является трудоемкой задачей. При этом, сопровождение полученного таким образом программного продукта требует серьезной специальной подготовки пользователя.

В то же время процесс перепроектирования задач оказывается столь же трудоемким и длительным. Таким образом, применение технологий обработки неоднородной информации, основанных на использовании традиционных языков программирования баз данных (например, языка SQL), оказывается неэффективным.

С другой стороны, базу данных (БД) можно определить как комплекс алгебраических и логических средств, ориентированный на разработку прикладных программ, независимых от СУБД при одновременном взаимодействии с другими, возможно, неоднородными БД. Следует отметить, что интеграция должна строиться таким образом, чтобы обеспечивалась возможность эволюционного развития системы, выраженного в допущении автономного использования, включая модификацию интегрируемых БД в рамках локальных прикладных программ одновременно с их использованием в составе единого информационного пространства.

Исследования в этой области ведутся с момента практического использования БД в распределенных и крупномасштабных системах. Оригинальные подходы были рассмотрены в [1], а также при построении испытательной распределенной базы данных [2]. Подходы к управлению распределенными ресурсами представлены в работах Sheth A.P., Larson J.A. [3], Garcia-Solaso M., Saltor F., Castellanos M. [4], в которых рассматриваются задачи, возникающие при достижении локальной автономности. Среди переведенных источников можно выделить работу группы авторов [5], которая отличается широтой и глубиной охвата материала по вопросам проектирования и использования современных систем БД.

Основным выводом из рассмотренного материала является то, что описание данных в неоднородных информационных системах является актуальной задачей и при этом требует умения разумно использовать сочетание технологических и архитектурных решений.

Таким образом, основной проблемой при решении задачи описания неоднородных данных является обобщенное представление объектов БД. В статье используется определение правил логического существования информационных объектов предметной области (ПрО) на основе требований информационной системы, и рассмотрены средства декларативного описания данных.

Целью работы является представление средств описания БД и формальное определение семантики данных, используя введенные языковые конструкции, основанные на логическом программировании. Для обеспечения универсальности подхода предлагаются алгоритмы, реализующие вычисление результатов запроса по заданному экстенсионалу БД (т.е. набору правил, описывающих свойства структуры данных).

1. Определение свойств правил декларативного языка описания данных

Определим основные характеристики декларативного языка и проведем параллель между исчислением предикатов первого порядка (в частности, реляционным исчислением) и правилами

логического программирования, которые коррелируют с *L*-правилами, описывающими существование и свойства информационных объектов ПрО [6].

Логический язык, рассматриваемый далее, является прототипом языка DataLog и состоит из импликативных правил типа «если..., то...». В этих правилах выражаются следующие факты: из определенной комбинации кортежей в определенном отношении можно вывести, что некоторый другой кортеж входит в другое отношение, или получить ответ на запрос.

Так как *L* - правила описывают ПрО на уровне отношений между информационными объектами БД, то использование логического программирования может быть применено при реализации запросов к неоднородным структурам данных. Таким образом, декларативное решение задачи обработки интегрированных данных подразумевает наличие логического языка с развитыми выразительными средствами и метод вычисления, который на основе аксиоматического представления ПрО с помощью правил вывода и унификации производит поиск информации в интегрированной среде [7]. Прикладным инструментом для реализации таких задач является компилятор дедуктивной базы данных (ДБД) как средство трансляции логического языка обработки данных. Модель ДБД представляет собой тройку:

$$M^{DDB} = <EDB, IDB, I>, \qquad (1)$$

где *EDB* — экстенсиональная база данных (ЭБД): соответствующие отношения хранятся в БД, *IDB* — интенсиональная (вычисляемая) база данных (ИБД): отношения вычисляемые посредством применения одного или нескольких правил, *I* — множество ограничений целостности.

Отношения в контексте DataLog представляется с помощью предикатов. Каждый предикат обладает фиксированным количеством аргументов. Предикат с аргументом будем называть атомом. Синтаксис атома представляет собой *n*-арную функцию типа $\mathcal{P}(x_1,..., x_n)$, возвращающую значение булевого типа. Если *R* – это таблица с *n* атрибутами, перечисленными в некотором порядке, то *R* можно трактовать как предикат, соответствующий таблице, при этом атом $R(l_{o_1},..., l_{o_n})$ будет иметь значение *True*, если список $(l_{o_1},..., l_{o_n})$ является строкой таблицы *R*, в противном случае атому соответствует значение *False* (здесь l_{o_i} является информационным объектом, который может быть семантически неделимым, то есть он не может быть интерпретирован по-другому, и многозначным, то есть его можно заменить несколькими объектами, имеющими в совокупности тот же смысл).

В качестве аргументов атомы способны воспринимать не только константы, но и переменные. Если аргументами атома являются одна или несколько переменных, атом представляет собой функцию, которая получает значения этих переменных и возвращает величины *True* или *False*.

Операторы, подобные операторам традиционной реляционной алгебры, в DataLog описываются с помощью правил. Согласование реляционных терминов и терминов логического программирования можно представить следующими пунктами:

– реляционный атом соответствует заголовку правила;

– символ "←" можно трактовать как условие "если";

– тело правила, которое может состоять из одного или нескольких атомов, трактуется как подцель правила, которые могут быть как реляционными, так и арифметическими атомами;

– в случае нескольких подцелей, они объединяются оператором *AND* ("∧"), и при необходимости каждой подцели может предшествовать оператор логического отрицания *NOT* ("¬").

Параллель между реляционными и логическими языками запросов можно показать на следующем примере. Пусть дана схема БД $R(l_{o_1}, l_{o_2}, l_{o_3}, l_{o_4}, l_{o_5})$ и пусть необходимо найти множество значений из l_{o_1} и l_{o_2}, для которых l_{o_3} <= N (где N = const), тогда правило будет иметь вид:

$$\mathcal{P}(x_1, x_2) \leftarrow R(x_1, x_2, x_3, x_4, x_5) \wedge x_3 \le N, \qquad (2)$$

где $(x_1, x_2, x_3, x_4, x_5)$ – предикатные переменные, соответствующие информационным объектам (l_{o_1}, $l_{o_2}, l_{o_3}, l_{o_4}, l_{o_5}$).

Аналогичный запрос, выраженный в терминах реляционной алгебры, будет иметь вид (3):

$$R' = \pi_{l_{o_1}, l_{o_2}} (\sigma_{l_{o_3} < N} (R)). \qquad (3)$$

В общем виде запрос в DataLog – это набор из одного или нескольких правил. Если в заголовке правила ссылка только на одно отношение, то содержимое этого отношения является ответом на запрос. Если в заголовке содержится ссылка на несколько отношений, то только одно из них будет интерпретироваться как итог выполнения запроса, а все остальные – как вспомогательные данные, используемые для получения результата.

Имена, используемые для переменных, особого смысла не имеют. Если переменная присутствует в списке аргументов только одного атома, ее заменяет символ подчеркивания ("_"). Если все экземпляры символа подчеркивания использовать для обозначения различных переменных, тогда правило (2) можно переписать в виде:

$$\mathcal{P}(x_1, x_2) \leftarrow R(x_1, x_2, x_3, \_, \_) \wedge x_3 \le N. \qquad (4)$$

Поскольку каждая из переменных x_4 и x_5 упоминается только один раз, то их можно заменить символом "_", в отношении остальных переменных это сделать нельзя, так как они присутствуют в выражении правила несколько раз.

2. Вычисление результата запроса средствами логического программирования

Воспользуемся работами [8, 9] и определим дополнительную терминологию для формулирования запросов средствами логического программирования.

Терм представляет собой константу или переменную. Терм называется основным тогда и только тогда, когда он не содержит переменных. В контексте DataLog это означает, что терм является основным тогда и только тогда, когда он является константой. Множество констант всех основных термов называется универсумом Эрбрана. Заметим, что в контексте логического программирования термом может быть сложная структура, построенная из функциональных символов, переменных и констант.

Основным атомом называется атом, все аргументы которого являются константами. Литералом называется атом $\mathcal{P}(x_1,..., x_n)$ или отрицание атома $\neg\mathcal{P}(x_1,..., x_n)$. Соответственно основным литералом называется основной атом или отрицание основного атома. Например, $\mathcal{P}_1(x_1, x_2)$, $\neg\mathcal{P}_2(x_3, x_4)$, $\mathrm{P}_3(x_5, X)$. Здесь два первых литерала – основные, а последний – нет, поскольку он содержит переменную. Литералы, являющиеся атомами, называются положительными, а литералы, являющиеся атомами с отрицанием, называются отрицательными.

Дизъюнктом (клаузой) называется конечный список атомов. Дизъюнкты определяются как множества и представляются в множественной нотации, например, $\{\mathcal{P}_1(x_1, x_2), \neg\mathcal{P}_2(x_3, x_4)\}$. Кроме того, в процессе манипулирования дизъюнктами с помощью алгоритмов важен порядок литералов в дизъюнктах (в то время, как с точки зрения семантики порядок литералов в дизъюнкте не имеет значения). Поэтому дизъюнкт будем считать списком или упорядоченным множеством. Определенные таким образом Хорновские дизъюнкты могут также быть представлены в нотации DataLog.

Дизъюнкты, состоящие из единственного атома, называются простыми. Дизъюнкты, содержащие только отрицательные атомы, называются отрицательными, а дизъюнкты, содержащие только положительные атомы, называются положительными.

Существуют некоторые ограничения, касающиеся способов использования переменных, предполагающие, что результатом вычисления правила должно служить конечное отношение и правила с использованием арифметических атомов подцелей с отрицанием (то есть атомов с приставкой "¬"). Эти ограничения выражаются посредством условия безопасности, которое определяет, что любая переменная, присутствующая в правиле, должна использоваться и в контексте некоторой реляционной подцели этого правила без операции отрицания.

Например, правило вида $\mathcal{P}(x_1, x_2) \leftarrow \mathcal{P}_1(x_1, x_3) \wedge \neg\mathcal{P}_2(x_4, x_1, x_3) \wedge x_1 \le x_2$ содержит три нарушения условия безопасности. Во-первых, переменная x_2 присутствует в заголовке, но ее нет в подцели $\mathcal{P}_1$. Необходимо отметить, что факт наличия x_2 в арифметической подцели $x_1 \le x_2$ не ограничивает возможные значения x_2 конечным множеством. Во-вторых, переменная x_4 присутствует только в подцели $\mathcal{P}_2$, являющейся аргументом оператора отрицания, и ссылок на нее больше нет. В-третьих, ссылка на переменную x_2 содержится в арифметической подцели, но в подцели без отрицания $\mathcal{P}_1$ ссылок на x_2 нет.

Рассмотрим правило $L = \{L_0 \leftarrow L_1,..., L_m\}$ и список основных фактов $(F_1,...., F_n)$. Если существует подстановка Ξ такая, что для всех $i = \overline{1,n}$ $\Xi(L_i) = F_i$, то из правила L и из фактов $(F_1,...., F_n)$ можно вывести за один шаг факт $\Xi(L_0)$. Выведенный факт может быть новым фактом или уже известным.

Например, рассмотрим правило $L = \{\mathcal{P}(X, Z) \leftarrow \mathcal{P}(X, Y), \mathcal{P}(Y, Z)\}$ и основные факты $F = \{\mathcal{P}(x_1, x_2)\}$ и $\{\mathcal{P}(x_2, x_3)\}$. Тогда можно вывести за один шаг факт $\{\mathcal{P}(x_1, x_3)\}$, используя подстановку $\Xi = \left\{\frac{X}{x_1}, \frac{Y}{x_2}, \frac{Z}{x_3}\right\}$. Далее рассмотрим правило $L = \{\mathcal{P}(X, Y) \leftarrow \mathcal{P}(Y, X)\}$ и факт $F = \{\mathcal{P}(x_1, x_1)\}$. Очевидно, нельзя вывести ничего нового, кроме $F' = \{\mathcal{P}(x_1, x_1)\}$, то есть самого факта.

Для вычисления множества всех фактов, которые могут быть выведены за один шаг, рассмотрим алгоритм 1.

АЛГОРИТМ 1. Вычисление новых фактов.

ВХОД. Множество правил L, исходное множество фактов F.

ВЫХОД. Множество фактов F'.

МЕТОД.

Шаг 1. Устанавливается логическая переменная $\alpha = \mathit{False}$ и обнуляется результирующее множество $F' = 0$.

Шаг 2. Перебирается все множество фактов F, $i = \overline{1,n}$.

Шаг 3. Проверяется выводимость за один шаг, используя подстановку Ξ.

Шаг 4. Если вывод возможен, то новый факт присоединяется к результату $F' := F' \cup P$ и $\alpha = \mathit{True}$. Если все факты проанализированы, то шаг 5.

Шаг 5. Алгоритм закончен.

Правильность алгоритма 1 легко доказать.

УТВЕРЖДЕНИЕ 1. Алгоритм 1 вычисляет новые факты для любого множества фактов F с помощью правил из множества L.

ДОКАЗАТЕЛЬСТВО. Пусть Ξ_i обозначает подстановку, порожденную выполнением алгоритма 1 для каждого $i = \overline{1,n}$. Если алгоритм завершается успешно, то есть с результатом $\alpha = \mathit{True}$, то подстановка $\Xi_1,..., \Xi_n$ реализует подстановку Ξ вывода факта за один шаг. С другой стороны, нетрудно

видеть, что в случаях, когда алгоритм 1 завершается неуспешно, то есть с результатом $\alpha = False$, то не может быть реализовано получение факта за один шаг.

Доказательство закончено.

Например, пусть дано правило $L = \{\mathcal{P}(X, Z) \leftarrow \mathcal{P}(X, Y), \mathcal{P}(Y, Z)\}$ и факты $F = \{\mathcal{P}(x_1, x_2); \mathcal{P}(x_2, x_3)\}$. Имеем $n = 2$ (количество шагов перебора) и $F_1 = \{\mathcal{P}(x_1, x_2)\}$, $F_2 = \{\mathcal{P}(x_2, x_3)\}$, $L_0 = \{\mathcal{P}(X, Z)\}$, $L_1 = \{\mathcal{P}(X, Y)\}, L_2 = \{\mathcal{P}(Y, Z)\}$. На первом шаге литералы L копируются в $F' := \mathcal{P}(X, Z) \cup \mathcal{P}(X, Y) \cup \mathcal{P}(Y, Z)$. Применяя подстановку $\Xi(L_1, F_1) = \left\{\frac{X}{x_1}, \frac{Y}{x_2}\right\}$ к L - литералам, получаем $F'_0 = \mathcal{P}(x_1, Z)$, $F'_1 = \mathcal{P}(x_1, x_2)$, $F'_2 = \mathcal{P}(x_2, Z)$. На втором шаге применяем подстановку $\Xi(L_2, F_2) = \left\{\frac{Z}{x_3}\right\}$ к L - литералам, получаем $F'_0 = \mathcal{P}(x_1, x_3)$, $F'_1 = \mathcal{P}(x_1, x_2)$, $F'_2 = \mathcal{P}(x_2, x_3)$. После завершения алгоритма возвращается факт $F'_0 = \mathcal{P}(x_1, x_3)$.

Для получения полного множества фактов, которые могут быть выведены за один шаг, рассмотрим следующий алгоритм.

АЛГОРИТМ 2. Вычисление полного множества фактов.

ВХОД. Конечное множество дизъюнктов L.

ВЫХОД. Множество фактов F''.

МЕТОД.

Шаг 1. Устанавливается логическая переменная $\alpha = False$ и обнуляется результирующее множество $F'' = 0$.

Шаг 2. Перебираются все множества правил L, $i = \overline{1,m}$.

Шаг 3. Последовательно для всех фактов $i = \overline{1,n}$ применяем алгоритм 1.

Шаг 4. Если после выполнения алгоритма 1 $\alpha = True$, то $F'' = F'' \cup F'$. Если все дизъюнкты проанализированы, то шаг 5.

Шаг 5. Алгоритм закончен.

Очевидно, что приведенный алгоритм эффективно вычисляет все факты, которые могут быть выведены за один шаг из L, поскольку для каждого правила из L он рассматривает каждую возможную комбинацию фактов и проверяет, совпадают ли эти факты с литералами в правой части правила.

Например, пусть дано множество дизъюнктов $L = \{L_1{:}\mathcal{P}(X, Z) \leftarrow \mathcal{P}(X, Y), \mathrm{P}(Y, Z); L_2{:}\mathcal{P}(X, Y) \leftarrow \mathcal{P}(Y, X)\}$ и множество фактов $F = \{\mathcal{P}(x_1, x_2); \mathcal{P}(x_2, x_3); \mathcal{P}(x_3, x_4); \mathcal{P}(x_4, x_5)\}$. Применяя алгоритм 2 к множеству L, получим факты $F'' = \{\mathcal{P}(x_1, x_3); \mathcal{P}(x_2, x_4); \mathcal{P}(x_3, x_5); \mathcal{P}(x_2, x_1); \mathcal{P}(x_3, x_2); \mathcal{P}(x_4, x_3); \mathcal{P}(x_5, x_4)\}$.

В общем виде процедуру вывода можно описать следующим образом. Пусть L — множество дизъюнктов, факт F' может быть выведен из L, в обозначении $L \models\!\Rightarrow F'$, если выполняются условия [8]:

– $L \models\!\Rightarrow F'$, если $F' \in L$;

– $L \models\!\Rightarrow F'$, если некоторое правило $L_i \in L$ и существуют факты $(F_1, \ldots, F_n)$, такие, что $L \models\!\Rightarrow F_i$ $(i = \overline{1,n})$ и F' может быть выведено за один шаг из $(F_1, \ldots, F_n)$;

– во всех других случаях $L \models\!\Rightarrow F'$ не выполняется.

Например, пусть как в предыдущем примере дано множество дизъюнктов $L = \{L_1{:}\mathcal{P}(X, Z) \leftarrow \mathcal{P}(X, Y), \mathcal{P}(Y, Z); L_2{:}\mathcal{P}(X, Y) \leftarrow \mathcal{P}(Y, X)\}$ и множество фактов $F = \{\mathcal{P}(x_1, x_2); \mathcal{P}(x_2, x_3); \mathcal{P}(x_3, x_4); \mathcal{P}(x_4, x_5)\}$. Так как $\mathcal{P}(x_1, x_2) \in L$, то выполняется $L \models\!\Rightarrow \{\mathcal{P}(x_1, x_2)\}$. С другой стороны, факт $\mathcal{P}(x_2, x_1)$ также можно вывести за один шаг, то есть $L \models\!\Rightarrow \{\mathcal{P}(x_2, x_1)\}$. Из L_1 можно получить $L_1 \models\!\Rightarrow \{\mathcal{P}(x_1, x_1)\}$, следовательно, имеем $L \models\!\Rightarrow \{\mathcal{P}(x_1, x_1)\}$.

Как и для правил логического существования L - правил, любой вывод факта может быть представлен с помощью дерева вывода. В таком дереве используются вершины двух видов: одни из них помечены дизъюнктами из L, а другие помечены фактами, выводимыми из L. Первый уровень состоит из дизъюнктов L, следующий уровень состоит из дизъюнктов L или дизъюнктов, которые выведены за один шаг из L и т.д. Каждая вершина дерева соединена не более чем с одной вершиной ближайшего более высокого уровня. Если дизъюнкт используется в нескольких выводах, то в дереве появляются несколько копий этого дизъюнкта.

Например, дерево вывода для поиска факта $\{\mathcal{P}(x_1, x_1)\}$ из предыдущего примера может быть представлено деревом, как на рис. 1.

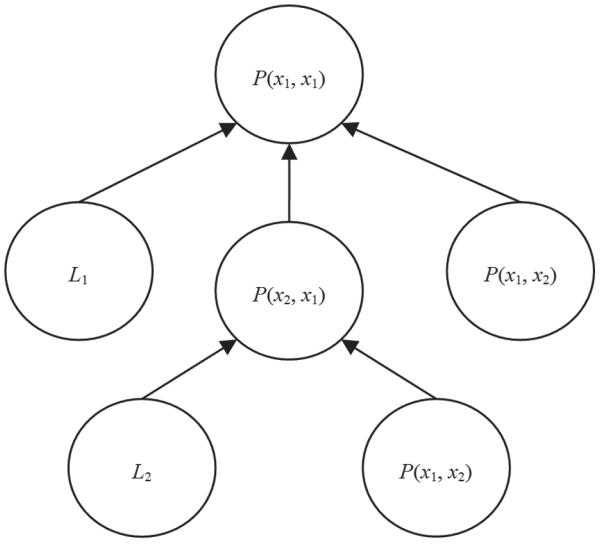

Рис. 1. Дерево вывода факта

Рассмотренные выкладки и алгоритмы определяют теоретико-доказательную схему, позволяющую выводить новые факты из первоначального множества дизъюнктов L. Доказательства корректности и полноты схемы вывода можно найти, например в [8].

Выводы

В статье исследованы и предложены методы и алгоритмы формирования результата запроса, представленного множеством информационных объектов *O*. Для обеспечения поддержки неоднородности свойства БД и ограничения целостности описываются логическими выражениями *I*.

Таким образом, автором определены основные свойства и характеристики декларативного описания данных. Показано, что использование синтаксических конструкций дедуктивных БД хорошо сопрягаются с логическими правилами. Для выведения новых фактов предложен соответствующий алгоритм, который вычисляет новые факты для любого множества фактов с помощью логических правил существования. Для обобщения полученных результатов построен алгоритм нахождения полного множества фактов из заданного набора информационных объектов. Рассмотрены методы использования правил языка дедуктивных БД DataLog для формирования запросов к таблицам, поддерживаемым реляционной СУБД.

Практическое использование полученных результатов заключается в осуществлении контроля модификаций структуры БД. Нарушения структуры таблиц может привести к несогласованности данных, что естественны образом отразится на целостности данных. Появление информационных объектов, противоположных по смыслу, на практике может иметь место при приведении различных моделей данных к единому виду, что также влияет на семантику данных.

Следует отметить, что дальнейшее исследование рассмотренного материала следует направить на определения минимальных требований к семантике БД, при которой допустимы изменения ее структуры.

Список литературы: 1. *Карденас, А.Ф.* Управление неоднородными распределенными базами данных [Текст] / А.Ф. Карденас // ТИИЭР. – 1987. – Т. 75, № 5. – с. 72-86. **2.** *Дуайер, П.А.*, Ларсон Дж.А. Опыт работы с испытательной распределенной базой данных [Текст] / П.А. Дуайер, Дж.А. Ларсон // ТИИЭР. – 1987. – Т. 75, № 5. – с. 126-138. **3.** *Sheth, A.P* Federated database for managing distributed, heterogeneous, and autonomous databases [Text] / A.P. Sheth, J.A. Larson // Computing Surveys. – 1990. – № 22(3). – p. 183-236. **4.** *Garcia-Solaso, M.* Semantic heterogeneity in maltidatabase system [Text] / F. Saltor, M. Castellanos // In Bukhres and Elmagarmid. – 1996. – p. 129-195. **5.** *Гарсиа-Молина, Г.* Системы баз данных. Полный курс [Текст] / Г. Гарсиа-Молина, Дж. Ульман, Дж. Уидом. Пер. с англ. – М.: Издательский дом "Вильямс". – 2003. – 1088 с.: ил. – 3000 экз. – ISBN 5-8459-0384-X. **6.** *Танянский, С.С.* Характеристические свойства объектов информационных систем [Текст] / С.С. Танянский. // "Штучній інтелект" науковий журнал – 2007. – № 1. – С. 78-89. **7.** *Пономаренко, Л.А.* Интеграция информационных систем при частичном отображении моделей данных [Текст] / Л.А. Пономаренко, С.С. Танянский, В.А. Филатов // Проблемы системного подхода в экономике. – 2008. – № 26. – С. 33-44. **8.** *Черри С.* Логическое программирование и базы данных [Текст] / С. Черри, Г. Готлоб, Л. Танка. – М.: Мир, 1992. – 352 с. **9.** *Ульман Дж.* Введение в системы баз данных [Текст] / Дж. Ульман, Дж. Уидом. // М.: Лори, 2006. – 379 с.

Поступила в редколлегию 30.04.2012

УДК 004.047:681.3.01

Мовні засоби декларативного опису неоднорідних баз даних / С.С. Танянський // Біоніка інтелекту: наук.-техн. журнал. – 2012. – № 2 (79). – С. 58–62.

Досліджено й запропоновано методи та алгоритми формування результату запиту. Визначено основні властивості і характеристики декларативного опису даних. Запропоновано алгоритми використання правил мови дедуктивних баз даних для формування запитів до таблиць реляційної структури.

Іл. 1. Бібліогр.: 9 найм.

UDK 004.047:681.3.01

Declarative language means heterogeneous databases / S. Tanyansky // Bionics of Intelligense: Sci. Mag. – 2012. – № 2 (79). – P. 58–62.

Investigated and proposed methods and algorithms form the query result. Defines the main features and characteristics of the declarative data. The algorithms use language rules of deductive databases for querying a relational table structure.

Fig. 1. Ref.: 9 items.