ПЕРЕЧЕНЬ ССЫЛОК

1. GRUMMP – Generation and Refinement of Unstructured, Mixed-Element Meshes in Parallel [Электронный ресурс] – Режим доступа : http://tetra.mech.ubc.ca/GRUMMP/.
2. *Останин А. Н.* Применение математических методов и ЭВМ. Вычислительные методы проектирования оптимальных конструкций: Учеб. пособие для вузов / Под общ. ред. А. Н. Останина. – Мн.: Выш. шк., 1989. – 279 с.
3. *Куприков М. Ю.* Твердотельное моделирование – новый подход к вопросам проектирования и подготовки технической документации [Электронный ресурс] / М. Ю. Куприков. – Режим доступа : http://nit.miem.edu.ru/cgi-bin/article?id=76.
4. *Голованов Н. Н.* Геометрическое моделирование / Н. Н. Голованов. – М.: Издательство физико-математической литературы, 2002. – 472 с.
5. *Рвачев В. Л.* Теория R-функций и некоторые ее приложения / В. Л. Рвачев. – К. : Наук. думка, 1982. – 552 с.
6. *Толок В. А.* Метод конечных элементов: теория, алгоритмы, реализация / В. А. Толок, В. В. Киричевский, С. И. Гоменюк, С. Н. Гребенюк, Д. П. Бувайло. – К. : Наук. думка, 2003. – 316 с.
7. *Скворцов А. В.* Эффективные алгоритмы построения триангуляции Делоне / А. В. Скворцов, Ю. Л. Костюк // Геоинформатика. Теория и практика. – Томск : Изд-во Томского ун-та, 1998. – Вып. 1. – С. 22–47.

Надійшла 4.08.2008

В статті розглянуто задачу побудови геометричних моделей інженерних конструкції нестандартної форми. Виконано огляд основних методів представлення геометричних моделей у сучасних САПР. Описано застосування апарату функцій В. Л. Рвачеві для формального опису та наступної дискретизації плоских геометричних об'єктів складної форми.

The problem of plotting geometrical models for engineering structures of non-standard form is considered in the paper. The review of the basic methods to describe geometrical models in modern CAD is done. Application of V. I. Rvachev's function apparatus for formal description and subsequent discretization of plane geometrical objects of complex form is discussed.

УДК 004.4'24

М. Ю. Терновой

ПІДХІД ДО ПРЕДСТАВЛЕННЯ ІНФОРМАЦІЇ В ТЕРМІНАХ ПРЕДМЕТНОЇ ОБЛАСТІ

Запропоновано підхід до отримання інформації з бази даних та її представлення в термінах предметної області. До складу цього підходу входять метод прив'язки до предметної області та метод побудови запиту до бази даних в термінах предметної області.

ВСТУП

Для ефективного управління будь-якою організаційною структурою керівнику необхідна різноманітна інформація, причому сутність цієї інформації та форма її подання будуть змінюватися в залежності від задачі, що вирішується. Від оперативності отримання та зручності подання необхідної інформації залежить своєчасність управлінських рішень [1]. На сьогоднішній день інформаційні технології складають основу інформаційно-аналітичної діяльності будь-якої організації. В сучасних інформаційних системах (ІС), що функціонують в організаціях, дані зберігаються в базах даних (БД), які найчастіше є реляційними [2]. Тому описана вище задача подання інформації зводиться до отримання даних з бази даних та їх представленні у формі, що вимагається.

Як правило, у впроваджених в організаціях ІС закладена можливість отримання звітів на основі поточних даних, але кількість різних за формою звітів обмежена. Створення нових звітів вимагає вдосконалення або переробки ІС за участі розробників.

З іншого боку на ринку програмного забезпечення існує багато систем, які надають можливість формувати різноманітні звіти, так звані генератори звітів (ГЗ). Генератори звітів існують як у вигляді самостійних програм, так і у вигляді генераторів звітів, що вбудовані в системи керування базами даних (СКБД) або середовища розробки. Серед останніх можна виділити [3–6]: MS SQL Server, MS Access, 1С. Аналогічні інструменти є в Delphi та інших середовищах розробки. Однак, користуватися вбудованими ГЗ можна лише за умови, що і вся система побудована на цій СКБД, або за допомогою цього середовища розробки.

Серед ГЗ, які є самостійними програмами, можна виділити [7–10]: MS SQL Server Reporting Services, Seagate Crystal Reports, VSREPORT, Report Sharp-Shooter та інші.

До загальних недоліків використання існуючих ГЗ можна віднести те, що для їх використання необхідні спеціальні знання в області інформаційних технологій та теорії баз даних. І хоча людина, що приймає рішення, є достатньо високо кваліфікованою в своїй предметній області, в області роботи з базами даних вона може не мати потрібних знань. Необхідність таких знань пов'язана з тим, що для отримання інформації з бази даних необхідно побудувати SQL-запит на вибірку для чого потрібно знати не тільки

саму мову SQL, а також необхідно знати структуру бази даних. Тому за відсутності таких знань очевидною стає необхідність звертатися до IT-спеціалістів, що підвищує вартість та знижує оперативність.

Виходячи з вищенаведеного актуальною є задача розробки ГЗ, який би дозволив працювати з даними з БД в термінах предметної області. Для цього, перш за все, необхідно розробити теоретичні основи такого ГЗ. Тому для вирішення цієї задачі нижче запропоновано підхід до отримання інформації з бази даних та її представлення, використовуючи лише терміни предметної області. Під терміном предметної області розуміється словесний опис тієї інформації, яка зберігається в базі даних та може бути отримана як результат виконання SQL-запиту.

1 ПРОБЛЕМА ПОБУДОВИ SQL-ЗАПИТУ НА ВИБІРКУ

В мові SQL запит на вибірку визначається оператором SELECT. Тоді можна сказати, що проблема полягає в автоматичній побудові SQL-запитів SELECT, при умові що користувач задає лише терміни предметної області та відповідні обмеження на них. Рішення цієї проблеми полягає в розробці підходу до формування такої службової інформації, та способу побудови запиту на основі цієї інформації, використовуючи які ГЗ зможе автоматично будувати запити на вибірку.

Запишемо загальний формат SQL-запиту на вибірку [2], дещо спростивши його та отримаємо SQL-запит наступного вигляду (1):

SELECT *column_expression*
FROM *table_name*
[WHERE *condition* [AND *column_condition*]] (1)

При побудові багатотабличного запиту необхідно в операторі FROM вказати всі таблиці, що використовуються в запиті, а також проміжні таблиці, через які проходить зв'язок між таблицями, що не пов'язані напряму, а в операторі WHERE необхідно вказати як саме зв'язані таблиці *condition*, а також умови на значення полів *column_condition*, які входять в *column_expression*. При виникненні помилок в побудові запиту можуть бути отримані помилкові дані. Наприклад, у випадку необхідності виведення даних з полів двох незв'язаних таблиць буде отримано декартовий добуток всіх значень поля з однієї таблиці на всі значення поля другої таблиці.

2 ПРИВ'ЯЗКА ДО БАЗИ ДАНИХ

Для надання можливості використання термінів предметної області при проведенні обробки даних пропонується робити одноразову прив'язку ГЗ до бази даних. Для цього заповнюються таблиця прив'язки (ТПр) термінів предметної області (ПО) до БД (табл. 1) та таблиця зв'язків (ТЗв).

Таблиця 1 – Таблиця прив'язки термінів ПО до бази даних

Термін предметної області	Таблиця, в якій зберігаються дані	Назва поля в таблиці, яке відповідає критерію

В таблиці ТЗв перший стовпчик та перший рядок – це назви таблиць, що є в базі даних та будуть використовуватися в запитах, а на перетині стовпчика з рядком, у випадку, коли є прямий зв'язок між таблицями, записуються поля таблиці, що знаходиться у верхньому рядку, та які пов'язані з відповідними полями таблиці, що знаходиться в першому стовпчику. Поля таблиць організуються у вигляді масивів. У випадку коли зв'язку немає, поле залишається порожнім. На діагоналі знаходяться порожні поля.

Під прив'язкою ГЗ до БД буде розумітись заповнення ТПр та ТЗв. Повною прив'язкою назвемо таке заповнення таблиці ТПр та ТЗв, виходячи з якого можна побудувати вірний SQL-запит на вибірку.

Множиною імен таблиць буде називатись впорядкована множина потужністю N, елементами якої є назви таблиць в базі даних $TN = \{TableName_1, TableName_2, ..., TableName_N\}$.

Під відображенням зв'язності буде розумітись $C: TN \times \times TN \to \{0, 1\}$, де $C(TableName_i, TableName_j) = 1$ у випадку коли таблиця i зв'язана з таблицею j, та $C(TableName_i, TableName_j) = 0$ у протилежному випадку.

Матрицею зв'язності $\hat{C}$ буде називатись $N \times N$ розмірна матриця, яка відповідає відображенню C, та кожен елемент якої вказує на те, чи існує прямий зв'язок між таблицями. Ця матриця будується на основі таблиці зв'язків заміною непустого поля на одиницю, а пустого на нуль.

Головною вимогою до прив'язки є її повнота, тобто прив'язка повинна гарантувати, що буде отримано вірний запит.

Схему даних БД можна представити у вигляді зваженого графу $G = (V, E)$, де замість таблиць слід поставити вершини, а замість зв'язків між таблицями – ребра з одиничною вагою. Множина вершин V графу G буде дорівнювати множині назв таблиць TN, а матриця суміжності A графу G буде відповідати матриці зв'язків C (2).

$$V = TN, \; A = C. \qquad (2)$$

Тоді пошук зв'язку між таблицями БД можна інтерпретувати як знаходження мінімального підграфу H графу G, який буде містити всі вершини, що відповідають таблицям, які входять до запиту. Така задача носить назву задачі про мінімальне зв'язування [11].

Для отримання вірного та однозначного запиту необхідно існування однозначного зв'язку між таблицями. Адже, як було описано вище результатом виконання SQL-запиту, який побудовано по таблицях, що не зв'язані напряму, та між якими не вказано зв'язок через інші таблиці, буде декартовий добуток значень одного поля на значення другого. До того ж, якщо зв'язок є неоднозначним, тобто може реалізовуватись через різні таблиці, то з теорії побудови реляційних баз даних та реляційної алгебри буде слідувати можливість побудови різних запитів, використовуючи які будуть отримані різні результуючі дані з БД. Якщо розглядати відповідний граф G, то це буде означати існування всього одного ланцюгу між кожними двома вершинами та, як наслідок, одиничність підграфу H.

З теорії графів відомо, що необхідною та достатньою умовою для існування між кожними двома вершинами тільки одного ланцюгу є зв'язність відповідного графу, а також відсутність циклів, тобто відповідний граф повинен бути деревом [11, 12].

Виходячи з вищесказаного, слід відмітити, що для доведення повноти прив'язки слід визначити, чи є граф G, який відповідає структурі БД та визначається матрицею суміжності $A = C$, деревом. Перевірка цього факту проводиться описаним нижче методом.

Метод перевірки того, що прив'язка є повною, можна розділити на три частини.

По-перше, виходячи з ТПр (табл. 1) та ТЗв, слід побудувати матрицю зв'язності $\hat{C}$, метод побудови якої описано вище, та визначити граф G, який буде відповідати структурі даних БД предметної області. Як було показано раніше, матриця суміжності A буде дорівнювати матриці зв'язності $A = \hat{C}$.

По-друге, перевірити, чи є граф G зв'язним. Перевірка зв'язності графу G проводиться виходячи з теореми [12], яка говорить, що граф G, який визначається матрицею суміжності A, зв'язний тоді і тільки тоді, коли $\hat{A}^I_{ij} = 1$ для всіх його вершин i і j. Де $\hat{A}^I = I \vee \hat{A}$, I – мультиплікативна одинична матриця.

$$A = A \vee A^{\oplus 2} \vee \ldots \vee A^{\oplus n}, \quad A^{\otimes k} = A^{\otimes k-1} \otimes A,$$
$$A^{\otimes 1} = A, \quad k = \overline{1, n}.$$

$B = C \vee D$ означає $B_{ij} = C_{ij} \vee D_{ij}$, де B, C, D – матриці розмірністю $m \times s$.

$F = C \otimes E$ означає $F_{ij} = (C_{i1} \wedge E_{1j}) \vee (C_{i2} \wedge E_{2j}) \vee \vee \ldots \vee (C_{is} \wedge E_{sj})$, $i = \overline{1, m}$, $j = \overline{1, p}$, де F, C, E – матриці, які мають розмірність $m \times p$, $m \times s$, $s \times p$ відповідно.

Тоді, перевірка зв'язності буде здійснюватись таким чином:

1. Знайти матрицю A, використавши алгоритм Уоршолла [12];

2. Визначити матрицю $\hat{A}^I = I \vee \hat{A}$;

3. Перевірити необхідну та достатню умову зв'язності $\hat{A}^I_{ij} = 1$, $i = \overline{1, n}$, $j = \overline{1, n}$.

По-третє, у випадку зв'язності графу перевірити, чи є граф G деревом. Для цього слід скористатися теоремою, яка говорить про те, що зв'язний граф є деревом у випадку, коли кількість його вершин більше на одиницю, ніж кількість його ребер, тобто $|V| = |E| + 1$.

Та якщо граф G є деревом, то робиться висновок про те, що прив'язка є повною.

В графі G напряму не задані ребра, зв'язки між вершинами графу визначаються через матрицю зв'язності A. Для того, щоб скористатися згаданою вище теоремою, слід визначити кількість ребер. Вона буде дорівнювати сумі елементів матриці A, які знаходяться вище головної діагоналі:

$$|E| = \sum_{i=2}^{N-1} \sum_{j=i+1}^{N} A_{ij}.$$

3 ПОБУДОВА ЗАПИТУ ДО БАЗИ ДАНИХ

Як було зазначено вище, основною проблемою при побудові SQL-запиту є знаходження зв'язку між таблицями бази даних, що не зв'язані між собою напряму, та, виходячи із знайденого зв'язку, формування аргументів операторів FROM та WHERE.

Постановка цієї задачі може бути записана наступним чином. Базуючись на ТПр, ТЗв, $\hat{C}$ та $TN = (TableName1, TableName2, \ldots, TableNameN)$ та виходячи з вхідних даних $D_{I_F} = \{(T_{ik_l}, cond_{ik_l}) | l = \overline{1, n_F}\}$, якими є обрані користувачем терміни предметної області T_{ik_l} та задані обмеження на них $cond_{ik_l}$, побудувати SQL-запит на вибірку.

Метод рішення задачі динамічної побудови запиту полягає в послідовному розв'язку двох більш дрібних задач. По-перше, визначення полів таблиць, що відповідають термінам предметної області, та формування *column_expression* і завдання умов на значення полів *column_condition* в (1). По-друге, визначення зв'язку між таблицями та формування *table_name* і *condition* в (1). Далі розглянуто більш докладно постановки та методи розв'язання цих задач.

Постановка першої задачі полягає в тому, що виходячи з ТПр та $D_{I_F} = \{(T_{ik_l}, cond_{ik_l}) | l = \overline{1, n_F}\}$, необхідно визначити *column_expression* і *column_condition.*

Розв'язок цієї задачі полягає в співставленні кожному терміну ПО T_{ik_l} з $\hat{D}_{I_F}$ відповідної назви поля $ColumnName_{t_l r_l}$ та таблиці $TableName_{t_l}$ з БД, які знаходяться в ТПр. Після цього *column_expression* формується простим переліком через кому назв відповідних полів, а *column_condition* переліком умов на значення визначених раніше полів, які записуються визначеним в мові SQL способом (3).

$$column\_expression = \\ = TableName_{t_i}.ColumnName_{t_1 r_1}, \\ TableName_{t_2}.ColumnName_{t_2 r_2}, \ldots, \\ TableName_{t_{n_F}}.ColumnName_{t_{n_F} r_{n_F}};$$

$$\text{column\_condition} = \\ = cond_{ik_1}(TableName_{t_1}.ColumnName_{t_1 r_1}) AND \\ cond_{ik2}(TableName_{t_2} ColumnName_{t_2 r_2}) AND \ldots \\ AND\ cond_{ik_{n_F}}(TableName_{t_{n_F}}.ColumnName_{t_{n_F} r_{n_F}}). \quad (3)$$

Наступним кроком формується множина назв таблиць $Table\_N = \{TableName_{t_l} | l = \overline{1, n_F}\}$, поля яких відповідають обраним користувачем термінам ПО. *Table_N* буде використовуватись при розв'язку наступної задачі.

Постановку другої задачі можна записати наступним чином. Виходячи з *Table_N*, $\hat{C}$ та *TN* визначити зв'язок між таблицями та сформувати *table_name* і *condition* в (1).

Як відмічалось раніше, пошук зв'язку між таблицями БД можна інтерпретувати як розв'язок задачі про мінімальне зв'язування в теорії графів. В цьому випадку постановку другої задачі можна переписати наступним чином: для графу $G = (V, E)$ необхідно визначити частковий підграф $H = (V', E')$, який буде містити вказані в деякій множині $Q = Table\_N$ вершини з найменшою сумарною вагою ребер. Та, використовуючи підграф *H*, *TN* та ТЗв, визначити *table_name* і *condition* в (1). Слід відмітити, що $|Q| > 1$.

В загальному випадку першу частину цієї задачі пропонується розв'язувати шляхом вирішення задачі про найкоротший ланцюг для вершин множини *Q*, вказуючи як джерело по черзі кожну вершину $v_Q \in Q$ [11]. Тоді для кожної вершини $v_V \in V$ будуть обраховані $|Q|$ чисел – відстаней від $v_Q \in Q$. Просумувавши їх для всіх вершин, визначається вершина з найменшою сумою. Через неї вершини з *Q* з'єднуються найкоротшими шляхами, що й дає розв'язок поставленої задачі.

Однак, оскільки граф *G* є деревом можна запропонувати простіший підхід для визначення підграфу *H*. Ідея такого підходу полягає в обранні довільної вершини $v_{i_1} \in Q$ в якості кореня, та введення відображення нумерації *pred* вершин графу *G* таким чином, що кожній не кореневій вершині ставиться у відповідність інформація про номер вершини, яка знаходиться вище, та з якою вона пов'язана (4). Таку вершину будемо називати батьківською. Оскільки *G* дерево, то задане таким чином відображення *pred* однозначне:

$$pred: V \setminus v_{i_1} \to [1;N],\ \forall v_j \in V \setminus v_{i_1}, \\ \exists! pred(v_j) = i : (v_i, v_j) \in E. \quad (4)$$

Процес перенумерації, тобто визначення відображення *pred*, проводиться по вершинах графу *G* до тих пір, доки не перенумеровані всі вершини з множини *Q*. Тобто множина вершин, для яких задана нумерація V_{num}, та на яких буде визначено відображення *pred,* в загальному випадку є підмножиною множини вершин *V* графу *G* (5). Множину номерів вершин, які складають множину припустимих значень відображення *pred* позначимо через *U*:

$$V_{\text{num}} \subseteq V,\ pred: V_{\text{num}} \to U,\ \forall v_j \in V_{\text{num}}, \\ \exists! pred(v_j) = i : (v_i, v_j) \in E. \quad (5)$$

Далі вводиться відображення розмітки вершин *mark*, що визначено на множині вершин $V_{\text{num}} \subseteq V$ та дає одиницю, якщо вершина $v_s \in V_{\text{num}}$ є частиною ланцюгу $P_{i_k i_1}$, який з'єднує вершину $v_{i_k} \in Q$ та кореневу вершину v_{i_1}, та нуль в протилежному випадку (6). У випадку коли $mark(v_s) = 1$ будемо казати, що вершина $v_s \in V_{\text{num}}$ позначена, в протилежному випадку – не позначена.

$$mark: V_{\text{num}} \to \{0;1\},\ \begin{cases} mark(v_s) = 1,\ v_s \in P_{i_k i_1}; \\ mark(v_s) = 0,\ v_s \notin P_{i_k i_1}. \end{cases} \quad (6)$$

Процес визначення відображення розмітки проводиться по всіх вершинах $v_{i_k} \in Q$, які ще не були позначені, та проводиться вверх до першої на шляху позначеної вершини. Всі інші вершини вважаються не позначеними. Тоді граф, який складається з всіх позначених вершин графу *G,* і буде шуканим підграфом *H*.

Описаний спосіб реалізується двома послідовно виконуваними алгоритмами: алгоритмом нумерації графу *G* (рис. 1) та алгоритмом розмітки графу *G* і знаходження *H*. Після цього, виходячи з *H*, *TN* та ТЗв можна сформувати *table_name* і *condition*. Алгоритм формування *table_name* і *condition* буде містити кроки, які аналогічні тим, що використовуються

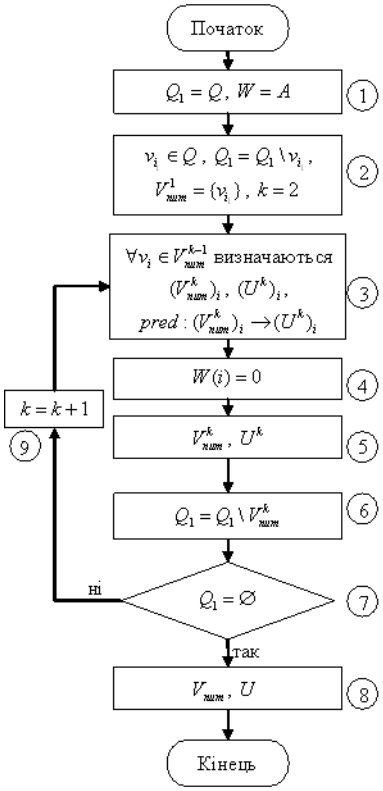

Рисунок 1 – Блок-схема алгоритму нумерації

в алгоритмі розмітки, тому доречним буде об'єднати ці алгоритм в один. Цей алгоритм буде мати назву алгоритм розмітки та формування *table_name* і *condition* (рис. 2). Слід зазначити, що оскільки граф G відповідає всій схемі даних БД, то підграф H відповідає частині схеми даних БД. Це означає, що оскільки $TN = V$, а $V' \subseteq V$, то $V' \subseteq TN$. Позначимо $TN' = V'$ множину імен таблиць БД, через які необхідно побудувати зв'язок в SQL-запиті, для його вірної реалізації. Множина ребер E' буде відповідати множині зв'язків між таблицями з TN'.

Додатково до введених позначень в цих алгоритмах будуть використовуватись наступні. Матриця W, яка перед початком роботи алгоритму дорівнює матриці суміжності A графу G, $W(i)$ буде означати i-ту строку матриці W, а w_{ij} – елемент матриці W, який знаходиться в i-му рядку та j-му стовпчику. Множина Q_1, яка буде складатись з тих вершин множини Q, які на даному кроці проведення перенумерації та розмітки не були помічені. Змінні *temp* та *temp*1, які будуть використовуватись для тимчасового зберігання номеру вершини. Множина вершин V^k_{num} з області визначення відображення *pred*, які були визначені на k-й ітерації, а множина припустимих значень – U^k. Тоді область визначення та множина припустимих

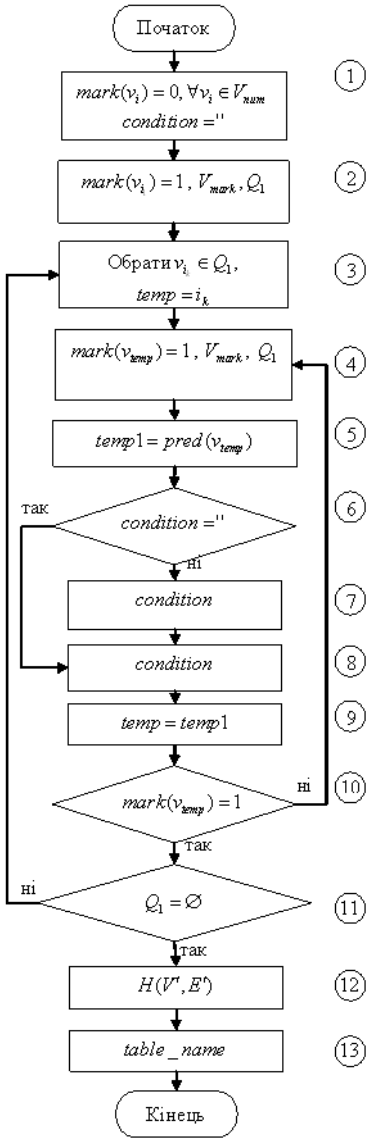

Рисунок 2 – Блок-схема алгоритму розмітки та формування table_name і condition

значень відображення *pred* визначається через $V_{\text{num}} = \bigcup_k V^k_{\text{num}},\ U = \bigcup_k U^k$.

Використавши отримані в (3) *column_expression* і *column_condition*, а також сформовані в результаті виконання алгоритму *table_name* і *condition* можемо записати SQL-запит у вигляді (1).

ВИСНОВКИ

Запропонований підхід до отримання інформації з бази даних та її представлення в термінах предметної області має наступні переваги:

1. При коректному формуванні таблиць прив'язки та зв'язків можна обійти недоліки, які були зроблені при проектуванні бази даних;

2. Даний підхід гарантує вірність побудови складних запитів та підвищує оперативність отримання необхідної інформації;

3. Виключає необхідність для користувача володіння спеціальними інформаційно-технічними знаннями та дозволяє зосередитись на вирішенні задач управління.

ПЕРЕЛІК ПОСИЛАНЬ

1. *Голышев Л. К.* Сложные системы с развитой функцией информационно-аналитической поддержки управления. Элементы теории, методологии, практики / Л. К. Голышев. – К. : 2001. – 253 с.
2. *Конноли Т.* Базы данных: проектирование, реализация и сопровождение. Теория и практика / Т. Конноли, К. Бегг, А. Страчан. – М. : «Вильямс», 2000. – 1120 с.
3. Microsoft SQL Server 2008 [Електронний ресурс]. Електрон. текстові дані. – Режим доступу: www.microsoft.com/SQL/default.mspx. – Wednesday, 15 April, 2009.
4. Microsoft Office Access 2007 [Електронний ресурс]. Електрон. текстові дані. – Режим доступу: http://office.microsoft.com/ru-ru/access/default.aspx. – Wednesday, 15 April, 2009.
5. *Праг К. Н.* Microsoft Access 2000. Библия пользователя / К. Н. Праг, М. Р. Ирвин. – М. : Диалектика, 2001. – 1039 с.
6. 1С: Предприятие [Електронний ресурс]. Електрон. текстові дані. – Режим доступу: http://www.1c.ru/. – Wednesday, 15 April, 2009.
7. Microsoft SQL Server 2008: Reporting Services [Електронний ресурс]. Електрон. текстові дані. – Режим доступу: http://www.microsoft.com/sqlserver/2008/en/us/reporting.aspx. – Wednesday, 15 April, 2009.
8. SAP BusinessObjects. Crystal Reports [Електронний ресурс]. Електрон. текстові дані. – Режим доступу: http://www.sap.com/solutions/sapbusinessobjects/sme/reporting/crystalreports/index.epx. – Wednesday, 15 April, 2009.
9. The VSReport Designer. [Електронний ресурс]. Електрон. текстові дані. – Режим доступа: http://help.globalscape.com/help/eft5/arm/the_vsreport_designer.htm. – Wednesday, 15 April, 2009.
10. Flexible reporting to leverage your Business Intelligence capabilities [Електронний ресурс]. Електрон. текстові дані. – Режим доступу: http://www.perpetuumsoft.com/Product.aspx?lang=en&pid=21. – Wednesday, 15 April, 2009.
11. *Касьянов В. Н.* Графы в программировании: обработка, визуализация и применение / В. Н. Касьянов, В. А. Евстигнеев. – СПб. : БХВ-Петербург, 2003. – 1104 с.
12. *Андерсон Дж. А.* Дискретная математика и комбинаторика: Пер. с англ. / Дж. А. Андерсон – М. : Вильямс, 2003. – 960 с.

Надійшла 23.02.2009
Після доробки 11.05.2009

Предложен подход к получению информации из базы данных и ее представлению в терминах предметной области. В состав этого подхода входят метод привязки к предметной области и метод построения запроса к базе данных в терминах предметной области.

The approach to information obtaining from database and its representation in the terms of data domain was proposed. This approach is represented by the method of data domain binding and the method of query construction in data domain terms.

УДК 681.326

А. С. Шкиль, Е. Е. Сыревич, Д. Е. Кучеренко, Самер Альмадхоун

МЕТОД ОБРАТНОГО ПРОСЛЕЖИВАНИЯ ДЛЯ ПОИСКА ОШИБОК ПРОЕКТИРОВАНИЯ В HDL-КОДЕ

В данной статье были рассмотрены методы поиска ошибок проектирования в неструктурированном HDL-коде. Разработан метод обратного прослеживания. Проведен эксперимент над HDL-моделью цифрового устройства с использованием данного метода.

1 ПОСТАНОВКА ЗАДАЧИ

Верификация цифровых проектов, то есть аппаратных или встроенных аппаратно-программных систем, описанных на языке описания аппаратуры (Hardware Description Language – HDL), является важной задачей в процессе проектирования цифровых устройств. Часто более 70 % времени разработки затрачивается на поиск и исправление ошибок в проекте. *Целью* данной работы является разработка методов поиска дефектов/ошибок проектирования в неструктурированном HDL-коде, позволяющих сократить время проведения диагностического эксперимента и уменьшить длину диагноза. Исходя из вышесказанного, необходимо решить *задачу* адаптации метода обратного прослеживания при верификации HDL-модели и провести диагностический эксперимент по поиску ошибок проектирования в рамках верификации [1].

2 АНАЛИЗ МЕТОДОВ ПОИСКА ДЕФЕКТОВ ПРИМЕНИТЕЛЬНО К МОДЕЛЯМ ЦИФРОВЫХ УСТРОЙСТВ

В данной работе аналогично понятию дефект используется понятие ошибки проектирования. Диагно-