3. Зенкевич О. Метод конечных элементов в технике / О. Зенкевич. — М. : Мир, 1975. — 541 с.
4. Хомченко А. Н. Стандартные серендиповы многочлены и линейчатые поверхности / А. Н. Хомченко, Е. И. Литвиненко, И. А. Астионенко // Міжвуз. зб. «Комп'ютерно-інтегровані технології: освіта, наука, виробництво». — Луцьк : ЛНТУ, 2011. — Вип. 6. — С. 266–269.
5. Литвиненко Е. И. Внутренние моды конечных элементов: преобразование лагранжевых моделей в серендиповы / Е. И. Литвиненко, А. Н. Хомченко // Вестник Херсонского национального технического университета. — Херсон : ХНТУ, 2012. — Вып. 2 (45). — С. 205–210.
6. Хомченко А. Н. Взвешенное усреднение базисов и наследственность в многопараметрических серендиповых аппроксимациях / А. Н. Хомченко, Е. И. Литвиненко, И. А. Астионенко // Геометричне та комп'ютерне моделювання. — Харьков : ХДУХТ, 2010. — Вип. 26. — С. 66–72.

Bicubic polynomial serendipity family received (built) by weighted averaging of the inner modes.

Key words: *bicubic polynomial, weighted averaging, inner modes.*

Отримано: 04.10.2012

UDC 004.048;004.5;004.82

I. A. Chimir*, Doctor of Technical Sciences,
Yu. O. Furtat,** Ph. D. Student

*Odessa State Economic University, Odessa
**Institute for Modelling in Energy of G. E. Pukhov
NAS of Ukraine, Kyiv, Kyiv

USING DOMAIN-INDEPENDENT DIALOG-BASED PROBLEM SOLVER TO FACILITATE DATABASE MANAGEMENT

Working with databases and forming queries to retrieve data requires knowledge of the stored data structure and access mechanisms. This can be a problem for unprofessional users. Means are to be developed to simplify this process, make it similar to the natural language use. in this article a domain-independent dialog-based problem solver is considered as one of such means.

Key words: *data manipulation language, database query, dialog, interrogative interaction, active agent, reactive agent.*

The dialog method of querying the database. Modern means of manipulating data in databases are increasingly focused on the inexperienced unprofessional user. One of the trends is the increasing in the "degree of non-procedure" in the data manipulation language (DML). In [1], a classification of a number of DML used to query the databases is given, described by the relational model. Structured query language Sequel is considered. The following information is also presented: the average time for a layman to master Sequel — 10 .. 12 hours, time required to the form a query of average complexity — 3 .. 5 minutes.

Structured query languages, such as Sequel, QUEL, and in particular, SQL (Structured Query Language) are, at present, major means of querying relative databases. In [2] it is noted that the requests are formulated in these languages, regardless of how relations are stored, and saving users from having to solve many low-level computational problems. Thus these DML are real high-level languages, and their creation is as important for the use of databases, as important it was, at one time, creating Fortran for solving numerical problems.

Below is an example query in SQL dialect used by the database management system (DBMS) DB2 [3]:

```
SELECT NAME
FROM S
WHERE 'P2' IN
(SELECT NUMBER_OF_PART
FROM SP
WHERE NUMBER_OF_SUPPLIER =
   S.NUMBER_OF_SUPPLIER);
```

The above example SQL-query in natural language can be formulated as follows: "Get the names of suppliers who supply part P2."

An analysis of the SQL-query leads to two important conclusions:

- to create a query one has to know SQL syntax;
- to create a query one has to know the logical structure of the database.

These "extra" knowledge is often an insurmountable barrier to the end user and, therefore, no SQL-type DML has the necessary end user level non-procedure.

In [4], the term "frontal program" is used to refer to the means by which one can hide SQL from the end user, in the sense that in the process of interaction with the frontal program SQL-queries are generated automatically. It is assumed that for interaction with the frontal program little or no additional knowledge is needed. An example of the frontal program is language QBE (Query By Example) [5], where to obtain the necessary records a user enters a sample entry. In addition to tools such as QBE, a series of frontal programs are based on the use of various dialects of natural forms — NFQL (Natural Forms Query Language) [6]. The degree of non-procedure NFQL-type DML is high enough. Instead of the drudgery of coding, you can concentrate on completing the form. This job requires zero knowledge of the query language, but requires familiarity with a number of forms.

Clearly, the higher the degree of non-procedure tools provided by the frontal program, the higher the efficiency of its applicability. From this point of view, are highly effective frontal programs that hide structured queries with a dialog with a fixed allocation of roles between dialog agents.

In [7] an example that illustrates this approach can be found. A request that is formulated in a natural language as: "Which workers worked overtime during the month," and in a structured form is written as:

```
FROM FILE = PAYROLL
FOR (PERIOD = 25 AND TYPE = HOURLY)
LIST PAY
```

is transformed into a piece of dialog shown below:

```
LEVEL SELECTIONS:
     1.  PAYROLL LEVEL – EMPLOYEE INFO
     2.  TIME LEVEL – PERIOD INFO                 Step 1
PLEASE SELECT A LEVEL ( )
<user response>: 1
PAYROLL LEVEL SEARCH SELECTOR:
     1.  SEARCH ON PERIOD CONTAINED IN
     2.  SEARCH ON TYPE (TIME FRAME)               Step 2
     3.  SEARCH ON PAY (DOLLARS AND CENTS)
PLEASE SELECT A SERCH ( )
<user response>: 1
PAY-PERIOD CONTAINED IN SEARCH SELECTOR
THESE PAY-PERIODS ARE POSSIBLE:
     1.  125 (WEEKLY)
     2.  243 (BI-WEEKLY)
     3.  453 (SEMY-MONTHLY)                         Step 3
     4.  895 (MONTHLY)
PLEASE SELECT A PAY-PERIOD ( )
<user response>: 4

ANY MOE SEARCH CRITERIA: (Y/N)                      Step 4
<user response>: Y

AND/OR: ( )                                         Step 5
<user response>: AND
```

The above examples of non-procedural end user interaction with database use classification methods can be used for introducing new data from the database. Methods for retrieval of data are divided into two classes:

- direct interrogative interaction;
- inverse interrogative interaction.

Direct interrogative interaction suggests that the data from the database is passed to the user in the form of response to the question. A user plays the role of an active (asking) agent.

For *inverse interrogative interaction* database query is generated automatically by the frontal program from the user's answers to the frontal program questions. User is a reactive (responding) agent.

Without conducting a detailed comparative analysis of the direct and inverse methods of obtaining data from the database, we note one important difference — the reactivity of the end user for inverse interrogative interaction. Reactivity allows to build computer systems that require from the user minimal or zero knowledge on the computer system internal organization. Therefore, the application of the inverse interrogative method to retrieve data from the database in the frontal program allows you to develop software packages that have the maximum degree in the formulation of non-procedural database queries.

Fragment of the Dialog Knowledge Base (DiKB), that automatically generates SQL-query, must consist of a chain of steps, scenes of which have the following contents in terms of the logical structure of the question-answer pair:

- scene used to define an explicit set of the question subject properties
- scene used to define a property of an explicit set-associated subject;
- scene used to specify the answer cardinality, for example – as a prerequisite index.

Inverse interrogative interaction is a universal practice in the construction of computer systems to communicate with databases, that does not require the use of knowledge of both the internal organization of the system and the structure of the database.

Ideas of the inverse interrogative interactions were tested experimentally in [8].

A dialog problem-solver is an interactive program that allows to create, store and edit the knowledge needed to support question-answering dialog process aimed at solving the problem. Both declarative and procedural knowledge, required to implement this process, is stored in an dialog knowledge base DiKB. In the system the principle of the separate storage of declarative and procedural knowledge is adopted. Portions of declarative knowledge with the logical structure of the question and answer will be called Que-chunk and Ans-chunk correspondingly.

Declarative knowledge is stored in the question memory QMem, and procedural knowledge — in the dialog access method memory DiAM. QMem and DiAM are considered in terms of method of access to information regarding the current and the next steps of dialog. From this point of view QMem is a memory with random access to the questions specifications (Que-chunk) and, therefore, to gain access to the specifications of a particular question needs an address of the corresponding specification. DiAM is a network structure, which can transmit the current response of a reactive agent (Ans-chunk) to the specifications of a relevant answer. Thus, DiAM, in fact, holds the solution method.

Fig. 1 shows the class diagram modeling the simplified structure of the domain-independent dialog problem-solver. Dialog knowledge base (DiKB) is associated with two classes: DiMC, performing a step-by-step interpretation of knowledge stored in an interactive knowledge base, and the class DiGen (Dialogue Generator), which is a DiKB full screen editor.

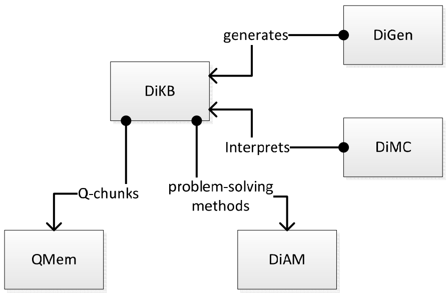

***Fig. 1.** Simplified structure of domain-independent dialog problem-solver*

Class DiMC consistently interprets every step of dialogue, and the phases of his work do not depend on the specific steps, or on the specific subject area.

Once the dialog knowledge base is developed, there is no need in the DiGen class. This class is used during the generation and editing of dialog knowledge base. Note that methods of the DiMC and DiGen classes can be distributed among the classes of the interactive knowledge base, if the latter is viewed as an object database.

Structure of the dialog problem-solver can now be synthesized. This structure is can shown in Fig. 2.

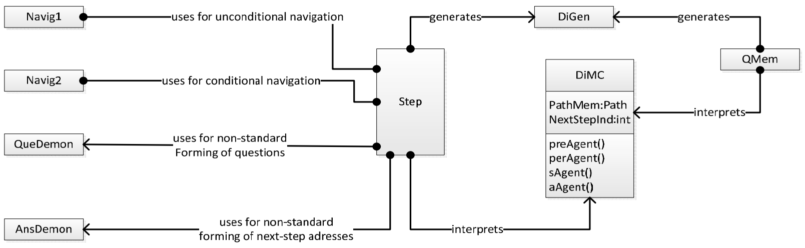
Navig1
uses for unconditional navigation
generates
DiGen
generates
QMem
Navig2
uses for conditional navigation
Step
DiMC
PathMem:Path
NextStepInd:int
interprets
QueDemon
uses for non-standard Forming of questions
preAgent()
perAgent()
sAgent()
aAgent()
AnsDemon
uses for non-standard forming of next-step adresses
interprets

***Fig. 2.** The structure of domain-independent dialog problem-solver*

Software agents that implement a unified cyclic process of the datalogical interpretation of the node are modeled as class DiMC methods: preAgent, perAgent, aAgent, sAgent (elements of the dialog modeling machine [9]).

It is assumed that the database fields are open for selected methods. In addition, fields that simulate the problem-solver "inner world" are introduced into the DiMC class.

For class DiMC a decentralized way is adopted to control method-agents call. It is assumed that the method-agents themselves manage the calls of each other.

Conclusions. The problem-solver, described in this article, allows for easier user-database interaction by introducing natural language dialog mechanics into the data management process. On the basis of created problem-solver structure an application can be developed to aid database users in different domains of knowledge and branches of production.

References

1. Дешко А. И. Непроцедурные языки манипулирования данными. Перспективы развития и классификация. Программирование / А. И. Дешко, Б. В. Игнатенко, В. И. Павловский. — 1985. — №4.
2. Грэй П. Логика, алгебра и базы данных / П. Грэй. — М. : Машиностроение, 1989.
3. Дейт К. Руководство по реляционной СУБД В2 / К. Дейт. — М. : Финансы и статистика, 1988.
4. Файнберг В. Базы данных типа «клиент-сервер» / В. Файнберг // Компьютер-Пресс. — 1990. — №7.
5. Zloof M. M. QBE/OBE; a Language for Office and Business Automation. Computer / M. M. Zloof. — 1981.
6. Embley D. W. NFQL: The Natural Forms Query Language / D. W. Embley // ACM Transactions on Database Systems. — 1998. — Vol. 14, №2. — 1989.
7. Clifton L. Barbary Database Primer on Natural Language / L. Clifton // Journal of System Management. — April, 1987.
8. Ус М. Ф. Организация системы программируемого диалога для общения с базой данных : автореферат дис. ... канд. техн. наук / М. Ф. Ус. — К., 1992.
9. Верлань А. Ф. Когнитивные основы и особенности моделирования диалогового процесса / А. Ф. Верлань, И. А. Чмырь , Д. Велев, Ю. О. Фуртат // Сборник трудов IV Международной конференции «Моделирование-2012». — К. : ИПМЭ им. Г. Е. Пухова НАН Украины, 2012. — С. 442–445.

Робота з базами даних та формування запитів на отримання даних потребують знань про структуру даних, що зберігаються та про механізми доступу. Це може бути проблемою для непрофесіоналів. Потрібно розробити засоби для спрощення цього процесу, зробити його подібним до використання природної мови. У статті в якості одного з таких засобів розглядається проблемно-незалежний вирішував проблем на основі діалогу.

Ключові слова: *мова управління даними, запит бази даних, інтеррогативна діалогова взаємодія, активний агент, реактивний агент.*

Отримано: 11.10.2012