УДК 355.01: 004.056

М.Ю. Ромодин, Ю.А. Лапшов, П.И. Соснин

ПСЕВДОКОДОВОЕ ПРОГРАММИРОВАНИЕ В КОНЦЕПТУАЛЬНОМ ПРОЕКТИРОВАНИИ БАЗ ДАННЫХ

Ромодин Михаил Юрьевич, *окончил Ульяновский государственный технический университет, факультет информационных систем и технологий,. Аспирант кафедры «Вычислительная техника» УлГТУ. Инженер-программист ФНПЦ ОАО «НПО «Марс». Имеет статьи в области САПР. [e-mail: romodin_mihail@mail.ru].*

Лапшов Юрий Александрович, *магистр техники и технологий, окончил УлГТУ. Аспирант кафедры «Вычислительная техника» УлГТУ. Имеет статьи в области САПР. [e-mail: y.lapshov@mail.ru].*

Соснин Петр Иванович, *заслуженный работник высшей школы РФ, доктор технических наук, профессор, окончил радиотехнический факультет Ульяновского политехнического института. Заведующий кафедрой «Вычислительная техника» УлГТУ. Имеет многочисленные труды в области концептуального проектирования автоматизированных систем. [e-mail: sosnin@ulstu.ru].*

Аннотация

В статье представлены средства псевдокодового программирования проектных решений в концептуальном проектировании баз данных автоматизированных систем (АС). Предложенный подход и средства его реализации ориентированы на экспериментирование с версиями проектных решений в контексте разрабатываемой АС.

Ключевые слова: автоматизированное проектирование, база опыта, база данных.

Mikhail Yurievich Romodin, *Post-graduate Student at the Chair 'Computer Engineering' of Ulyanovsk State Technical University; graduated from the Faculty of Information Systems and Technologies of Ulyanovsk State Technical University; author of articles in the field of CAD. e-mail: romodin_mihail@mail.ru.*

Yury Aleksandrovich Lapshov *has a Master's degree in Engineering and Technologies; graduated from Ulyanovsk State Technical University; Post-graduate Student at the Chair 'Computer Engineering' of Ulyanovsk State Technical University; author of articles in the field of CAD. e-mail: y.lapshov@mail.ru.*

Petr Ivanovich Sosnin, *Honoured Worker of the Higher School of the Russian Federation; Doctor of Engineering, Professor; graduated from the Faculty of Radioengineering of Ulyanovsk Polytechnic Institute; Head of the Chair 'Computer Engineering' of Ulyanovsk State Technical University; author of numerous papers in the field of conceptual design of computer-aided systems. e-mail: sosnin@ulstu.ru.*

Abstract

The article presents the facilities of design decision pseudocode programming in conceptual design of data bases of computer-aided systems. The proposed approach and its implementation tools are oriented at the experiment with design decision releases in the context of the exploited computer-aided systems.

Key words: computer-aided design, experience base, data base.

Введение

В разработках АС, включающих базы данных, их концептуальное проектирование осуществляют согласованно, экспериментируя с проектными решениями. Для базы данных к основным концептуальным решениям принято относить структуру отношений и их связность, спецификации атрибутики и выбор первичных ключей. Разумеется, вариативность этих решений в существенной мере зависит от применения базы данных, то есть от запросов к ее содержимому, которые на ранних этапах концептуального проектирования могут быть оценены фрагментарно с высокой степенью неопределенности [1].

Для обеспечения необходимой согласованности потенциальных запросов и эскизных решений по базе данных проектировщикам, ответственным за концептуальный проект базы данных, целесообразно предоставить возможность экспериментирования с проектными решениями. Другими словами, проектировщикам необходимы инструменты, позволяющие создавать модельные реализации запросов к имитациям фрагментов будущей базы данных.

В статье представлены такие инструменты, в основу которых положено псевдокодовое программирование баз данных и их фрагментов, с учетом программируемого доступа к тестовому содержанию, вложенному в запрограммированные модели. Разработанные инструментальные средства нацелены на прототипирование проектных решений, в которых используется взаимодействие процесса или пользователя с информационным содержимым

проектируемой базы данных. Псевдокодовая динамика, вложенная в прототипы, не ограничивается только имитацией доступа к базе данных, а позволяет проектировщикам включать в прототипы аналоги действий, используемых в таких широко известных системах прототипирования, как Balsamiq Mockups, Fireworks, Justinmind Prototyper, iRise и Lucid Spec [2].

Разработанные средства экспериментирования с запрограммированными концептуальными решениями реализованы как специализированное расширение инструментальной вопросно-ответной среды $WIQA$ (Working In Questions and Answers), обслуживающей концептуальное проектирование АС [3].

Среда псевдокодового программирования

Инструментальная среда $WIQA$ разработана как АС, архитектура которой интегрирует ряд интерпретаций $WIQA$, одной из которых является понимание и использование этого комплекса как «инструментальной среды псевдокодового программирования» проектных задач [4]. Обобщенное представление такой интерпретации приведено на рисунке 1.

На обобщенной схеме отражено, что решение каждой программируемой задачи Z из дерева задач проекта протоколируется в вопросно-ответной форме (Question-Answer protocol, QA-протокол), рафинированной версией которого является псевдокодовая программа (QA-программа). Дерево задач проекта с их программами и организационная модель коллектива, исполняющего проект, загружаются в вопросно-ответную память (QA-память) инструментария **WIQA подобно тому, как загружаются программы в оперативную память компьютера.** Проектировщик, исполняющий роль «интеллектуального процессора» (на схеме I^j и I^k-процессоры), вызывает необходимую ему QA-программу из памяти и исполняет ее в режиме интерпретации с помощью «Интерпретатора», встроенного в инструментальную среду.

Специфику и представления псевдокодовых программ и их исполнения определяет память среды, ячейки которой предназначены для хранения интерактивных объектов, ориентированных на регистрацию «вопросов» и «ответов» (задачи – частный случай вопросов). Каждая ячейка имеет богатую атрибутику (например, ее уникальное индексное имя, тип, время последней модификации, указатель на лицо, создавшее экземпляр объекта, и другие системные атрибуты ячейки) [3]. Более того, проектировщик, зарегистрировавший в протоколе (использовавший в программе) определенный объект, имеет право и возможности приписать этому объекту любое количество полезных ему атрибутов, используя механизмы объектно-реляционного преобразования [4]. Отмеченная специфика распространяется на данные (QA-данные) и операторы (QA-операторы) псевдокодовых программ, из-за чего они и получили название «вопросно-ответные программы» и обозначение QA-программы.

Язык псевдокодового программирования

Язык псевдокодового программирования [5] проектных задач встроен в вопросно-ответную моделирующую среду $WIQA$ и ориентирован на ее QA-память [6], из-за чего в последующем тексте он будет обозначаться L^{WIQA}.

Специфику языка L^{WIQA} определяют следующие его особенности:

1. Язык L^{WIQA} специально приближен к **естественному языку в его алгоритмическом употреблении**, обслуживающему взаимодействие человека с доступным опытом в деятельности. По этой причине язык ориентирован на решение задач, за которыми стоят проектные прецеденты и их связные совокупности.

2. Текст исходного псевдокода программы на языке L^{WIQA} построен из представлений его структурных единиц (предложение за предложением, оператор за оператором) с помощью QA-данных в форме QA-модели этой

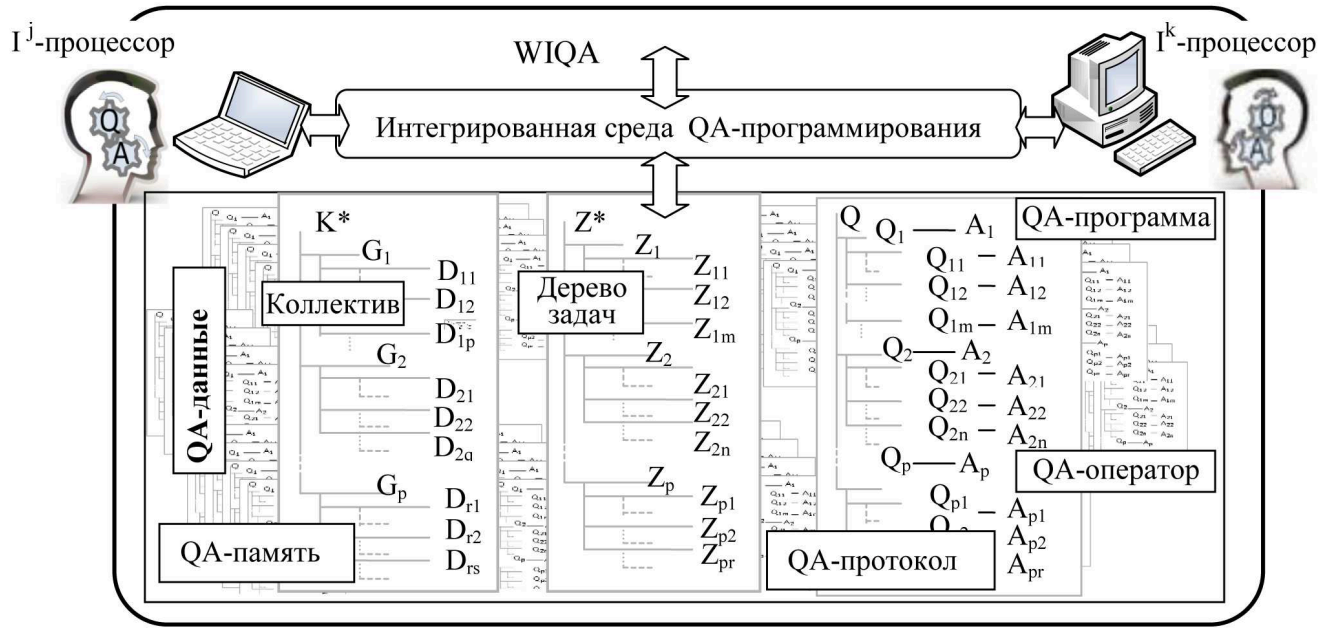


Рис. 1. $WIQA$ как среда псевдокодового программирования

задачи, что открывает для проектировщика возможности QA-моделирования в его работе с псевдокодом задачи, если в этом имеется необходимость.

3. Язык L^{WIQA} – объектно-ориентирован, поддерживает программирование задач с базами данных и программирование задач управления потоками работ в коллективе.

4. Если работа, запрограммированная на языке L^{WIQA}, может быть «передана» компьютеру, то соответствующая QA-программа компилируется в код, исполняемый компьютерным процессором (K-процессором).

Другими словами, один из классов QA-программ может исполняться как I-процессором, так и K-процессором. В работах с программами этого класса открываются следующие возможности:

1. Программы, исполнение которых будет возложено на K-процессор, создаются проектировщиком на языке L^{WIQA} в режиме интерпретации и отлаживаются с использованием роли I-процессора. Отлаженная программа компилируется в автоматически исполняемый код.

2. В процессе эксплуатации QA-программы в режиме интерпретации ее автоматизированное исполнение совершенствуется до состояния автоматического исполнения. В таком процессе «умение», используемое в процессе решения, переходит в «навык».

3. Программируя определенную работу проектировщиков, ее можно компоновать из программ, одна часть из которых исполняется I-процессором, а другая исполняется K-процессором.

4. Части одной и той же программы можно распределять между I-процессором и K-процессором, для повышения эффективности ее согласованного исполнения.

Язык L^{WIQA} разрабатывался как открытый для включения в его структуру дополнительных типов QA-данных и QA-операторов. Существующая структура языка приведена на рисунке 2.

На рисунке 2 отдельными группами показаны:

• «Структуры данных», за которыми стоит эмуляция любых традиционных типов данных (скаляры, массивы, записи, множества, стеки, очереди, ...) с помощью средств QA-данных;

• «Модели данных», используемые для спецификации табличных структур и их совокупностей, а также иерархических структур;

• «Базовые операторы» традиционных псевдокодовых языков программирования (Appoint, Input, Output, If, GOTO другие);

• «Операторы баз данных», эмулирующие основные SQL-операторы;

• «Операторы потоков работ», эмулирующие основные операторы языков имитационного моделирования;

• «Операторы визуализации», которые разрабатываются в настоящее время для просмотра по запросам проектировщиков ячеек QA-памяти по сценариям, запрограммированным предварительно или оперативно.

Детализацию структуры языка L^{WIQA} представим таблично (табл. 1), раскрыв базовые операторы и их синтакис.

Аналоги операторов, приведенные в таблице 1, применяются в традиционном псевдокодовом программировании. Отличия обусловлены тем, что QA-оператор:

• выполняется I-процессором, который, при необходимости, анализирует сложившиеся условия (предусловия) и/или проверяет результат исполнения, используя полезные для таких действий средства моделирующей среды $WIQA$;

• наследует атрибуты «ячейки» QA-памяти, в которой оператор записан, что открывает возможность использования этой атрибутики в процессах анализа предусловия и проверки результата.

В действиях по анализу предусловий и контролю результатов QA-оператор интерпретируется как «данные» (QA-данные), включающие не только базовые атрибуты, но и дополнительные, которые I-процессор имеет право скорректировать (или до выполнения оператора, или после). Такие свойства и возможности исполнения у операторов традиционного псевдо-кодового программирования отсутствуют.

Перейдем к **операторам**, **расширяющим базовый набор** для его использования в программировании задач, предполагающих **доступ** к **базам данных**.

Ситуации, в которых полезно моделировать доступ к базе данных, когда ее нет, встают перед проектировщиками:

• в процессе концептуального проектирования базы данных AC;

• в задачах прототипирования проектных решений, включающих взаимодействие с базой данных;

• в служебных задачах процесса проектирования, когда проектировщикам приходится или полезно оперативно создавать временные табличные структуры хранения.

Для псевдокодового программирования отмеченных возможностей язык L^{WIQA} был расширен набором псевдокодовых операторов, представленных в таблице 2.

По своему назначению **операторы расширения** аналогичны родственным операторам языка SQL. Отличия, как и для базовых операторов, в исполнении и атрибутике, наследуемой от QA-памяти.

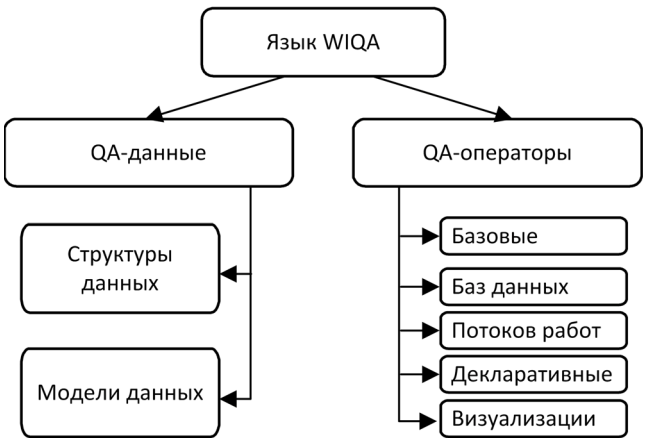


Рис. 2. Обобщенная структура языка

Таблица 1

Детализированная структура языка L^{WIQA}

Оператор	Описание
Команда на естественно-профессиональном языке	Текстовый оператор, интерпретируемый человеком-исполнителем программы
IF…THEN	Сокращенный условный оператор «Если..то»
IF..THEN..ELSE	Условный оператор «Если..то.. иначе..»
&var& := val	Присваивание переменной &var& значения val. Оператор **Appoint**.
CALL &имя_процедуры&	Вызов процедуры
GOTO &label&	Безусловный переход по метке &label&
FINISH	Окончание выполнения программы
SET &var1&, val1; &var2&, val2; … ; &varn&, valn	Установка значений множеству переменных
INTERRUPT	Прерывание выполнения программы. Нажатие человеком-исполнителем программы кнопки «Прервать выполнение»
TEST [E, NE, L, G, LE, GE], &var&, val &label&	Условный переход по метке &label& в случае [равенства, неравенства, меньше, больше, меньше или равно, больше или равно] значения переменной &var& значению val
INPUT &var&	Простой ввод человеком-исполнителем с клавиатуры значения переменной var простого типа
OUTPUT &var&	Простой вывод человеком-исполнителем на экран значения переменной var простого типа

Таблица 2

Набор операторов для работы с прототипами баз данных

Оператор	Описание
CREATE DATABASE	Создание базы данных. Методика, закодированная в виде QA-программы
CREATE TABLE	Создание таблицы. Исполняется с помощью специализированного мастера QA-данных
DROP TABLE	Удаление таблицы
SELECT	Выбор данных из таблицы
DELETE FROM	Удаление данных из таблицы
INSERT INTO	Добавление данных в таблицу
UPDATE	Изменение существующих в таблице данных

Формализация языка L^{WIQA}

Для языка L^{WIQA} разработана его формализованная модель в базисе BNF-нотаций [7]. Из-за своей грамоздскости в статье представлен фрагмент раздела этого языка в части работ с базами данных, который имеет следующий вид:

```
   <CREATE TABLE> ::= CREATE
   TABLE table _ name     ( { <
   column _ definition > | < ta-
   ble _ constraint > } [ ,...n])
< column _ definition > ::={ col-
umn _ name data _ type }
< table _ constraint > ::=[
CONSTRAINT constraint _ name ]
{ [PRIMARY KEY{ ( column [ ,...n
] ) }]| FOREIGN KEY
( column [ ,...n ] )
REFERENCES ref _ table [ ( ref _
column [ ,...n ] ) ]}

<SELECT> ::= SELECT  (*|(column [
,...n ]))
[ FROM { <table _ source> } ]
[ WHERE <search _ condition> ]
[ ORDER BY { column _ name  [
ASC | DESC ] }

<INSERT> ::= INSERT  [INTO <ta-
ble _ name> (column [ ,...n ])]
[VALUES (value, [,…n])]
<value> ::= («string»|int|float|Bool
ean|datetime)
```

Псевдокодовое программирование проектных решений

Как уже отмечалось выше, одним из приложений, для которых разрабатывался язык L^{WIQA}, является концептуальное проектирование баз данных АС в контексте запросов к их информационному содержимому. Это направление QA-программирования продемонстрируем на примере задачи «Контроль поручений», концептуальное решение которой привело к схеме, представленной на рисунке 3.

Проектные решения представим фрагментарно только для того, чтобы продемонстрировать особенности QA-программирования (рис. 4). В то же время отметим, что запрограммированная задача включена как подсистема в инструментарий $WIQA$, но она может быть развернута и использоваться независимо от системы $WIQA$.

В таблице «Субъект» есть колонка «ID группы», являющаяся Foreign Key, указывающая на таблицу «Рабочая группа». Primary Key и Foreign Key задаются как дополнительные атрибуты единиц.

На рисунке 5 изображен дополнительный атрибут Foreign Key столбца «ID группы» таблицы «Субъект». Значением атрибута является название таблицы, на которую указывает Foreign Key.

Строки данных в таблицах хранятся в ответных единицах, являющихся дочерними по отношению к вопросным, хранящим заголовки столбцов. Количество дочерних элементов в каждом столбце должно быть одинаковым.

Средства псевдокодового программирования дополняются визуальными средствами работы с базами данных, упрощающими работу пользователя с ними:

Мастер создания базы данных – представляет собой инструмент, позволяющий визуально создавать и редактировать структуру базы данных, таблицы и связи между ними.

Средство просмотра и редактирования таблиц – данный компонент позволяет просматривать структуру базы данных с возможностью просмотра и редактирования данных, содержащихся в таблице.

Средство импорта/экспорта базы данных из/в XML – данный компонент в случае необходимости позволяет сохранить всю базу данных или какую-либо ее отдельную таблицу с данными как файл и импортировать сохраненные ранее файлы.

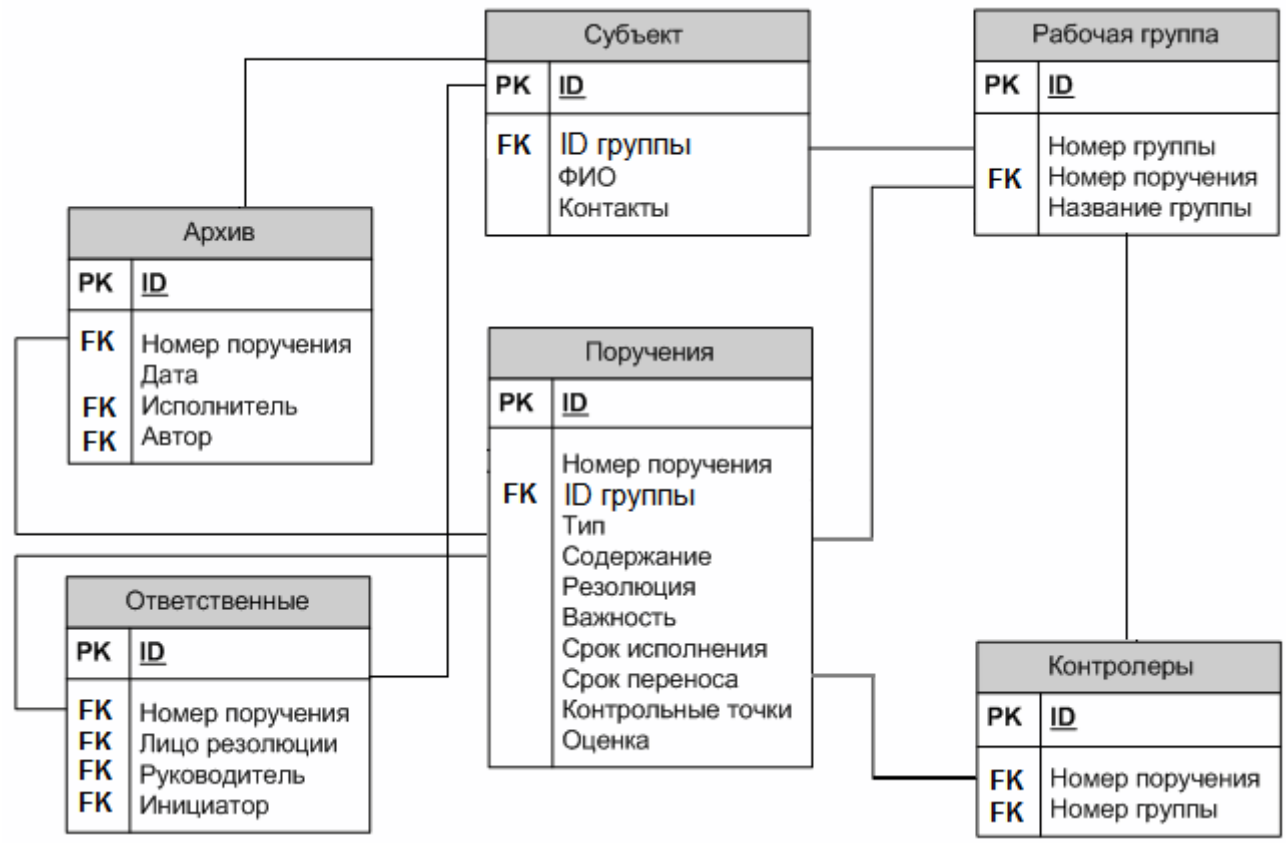


Рис. 3. Концептуальная схема базы данных для «Контроля поручений»

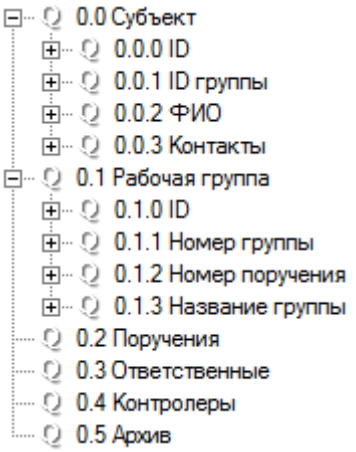
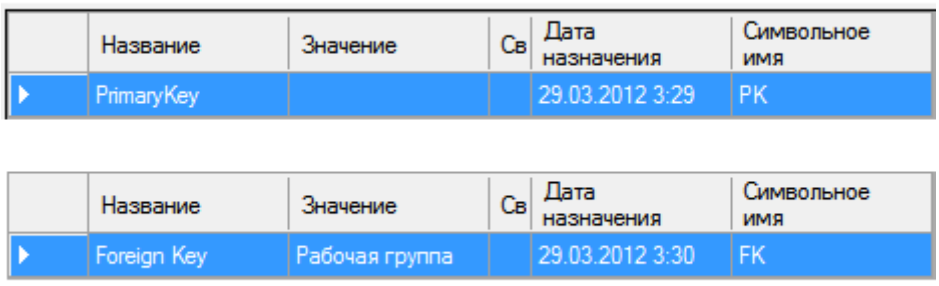


Рис. 4. Вопросно-ответная структура базы данных



	Название	Значение	Св	Дата назначения	Символьное имя
▶	PrimaryKey			29.03.2012 3:29	PK

	Название	Значение	Св	Дата назначения	Символьное имя
▶	Foreign Key	Рабочая группа		29.03.2012 3:30	FK

Рис. 5. Использование дополнительной атрибутики.

Средство помещения базы данных в вопросно-ответные шаблоны и извлечения из шаблонов в вопросно-ответный протокол.

Базы данных, как и любые другие фрагменты языка псевдокода, предполагают возможность сохранения их как вопросно-ответных шаблонов для дальнейшего неоднократного использования.

Формы хранения данных представим для отношения «Субъект» и «Рабочая группа», представление которых приведено на рисунке 6 совместно с учетом тестовой информации из таблиц 3 и 4. Строки данных в таблицах хранятся в ответных единицах, являющихся дочерними по отношению к вопросным, хранящим заголовки столбцов. Количество дочерних элементов в каждом столбце должно быть одинаковым.

Целостность данных контролируется с помощью реализованных на псевдокоде процедур добавления данных в таблицу, удаления данных и поиска.

Для демонстрации применений языка L^{WIQA} в задачах с доступом к базам данных приведем (без пояснений) фрагмент QA-программы для процедуры «Редактирование поручения»:

Q 2.3.3 PROCEDURE &Редактирование_поручений&
Q.2.3.3.1 &DbQAId& := QA_GetQAId (&current_project&, “БД поручений”)
Q.2.3.3.2 &DbId& := OpenDB(&current_project&. &DbQaId&)
Q.2.3.3.3 &groups& := ExecuteSQL(&DbId&, «SELECT ID, Название группы FROM Группа»)
Q.2.3.3.4 &people& := ExecuteSQL(&DbId&, «SELECT ID, ФИО FROM Участник»)
Q.2.3.3.5 &ToFindQuery& := AC_FindTask(&groups&, &people&)
Q.2.3.3.6 &FoundTasks& := ExecuteSQL (&DBId&, &ToFindQuery&)
Q.2.3.3.7 &TaskId& := AC_SELECTTASK(&FoundTasks&)
Q.2.3.3.8 &TaskQuery& := «SELECT * FROM Поручение WHERE ID = ''
Q.2.3.3.9 &TaskQuery& := STRCAT(&TaskQuery&, &TaskId&)
Q.2.3.3.10 &FoundTask& := ExecuteSQL (&DbId&, &TaskQuery&)
Q.2.3.3.11 AC_VIEWTASK (&FoundTask&, &Groups&, &People&)
Q 2.3.3.12 ENDPROC &Редактирование_поручений&

Для удобства взаимодействий с пользователями в псевдокодовых программах имеется возможность вызова функций из дополнительно подключаемой библиотеки TaskControlInt, содержащей методы, отображающие пользователю элементы пользовательского интерфейса – формы, содержащие поля для ввода, предназначенные для просмотра и ввода данных. Эти методы после выполнения возвращают упакованную в строковую переменную строку таблицы, которая впоследствии с помощью псевдокодовых инструкций записывается в вопросно-ответный протокол базы данных как новая или заменяя собой уже имеющуюся там запись в случае редактирования.

В библиотеке функций системы контроля поручений реализованы следующие функции (табл. 5).

На рисунке 7 представлена интерфейсная форма и приведен пример формы добавления нового поручения.

Для редактирования поручения используется та же самая форма с измененным заголовком окна. Перед отображением окна редактирования поручения псевдокодовая процедура отображает окно выбора поручения, позволяя пользователю выбрать поручение для редактирования.

Таблица 3

Таблица «Субъект»

ID (PK)	ID группы (FK)	ФИО	Контакты
0	0	Иванов Иван Иванович	ivanov@xmail.org
1	1	Алексеев Михаил Вадимович	alexeev@ymail.org
2	1	Растлер Геннадий Петрович	rastler@zmail.org

Таблица 4

Таблица «Рабочая группа»

ID (PK)	Номер группы	Номер поручения	Название группы
0	1	1	Первая группа
1	2		Вторая группа

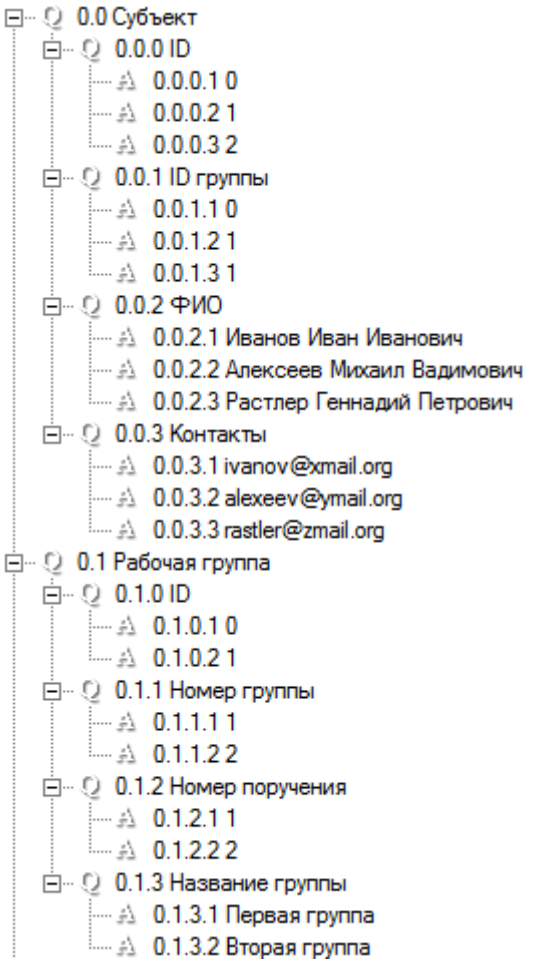


Рис. 6. Таблица с данными в виде вопросно-ответной структуры

Таблица 5

Функции системы контроля поручений

Функция	Описание
AC_CREATEPERSON	Окно добавления участника
AC_CREATEGROUP	Окно добавления рабочей группы
AC_ADDTASK	Окно добавления поручения
AC_EDITTASK	Окно редактирования поручения (окно добавления поручения с измененным заголовком окна)
AC_VIEWTASK	Окно просмотра поручения (Окно добавления поручения с измененным заголовком окна и с неизменяемыми полями)
AC_FINDTASK	Окно поиска поручения по заданным параметрам
AC_SELECTTASK	Окно выбора поручения

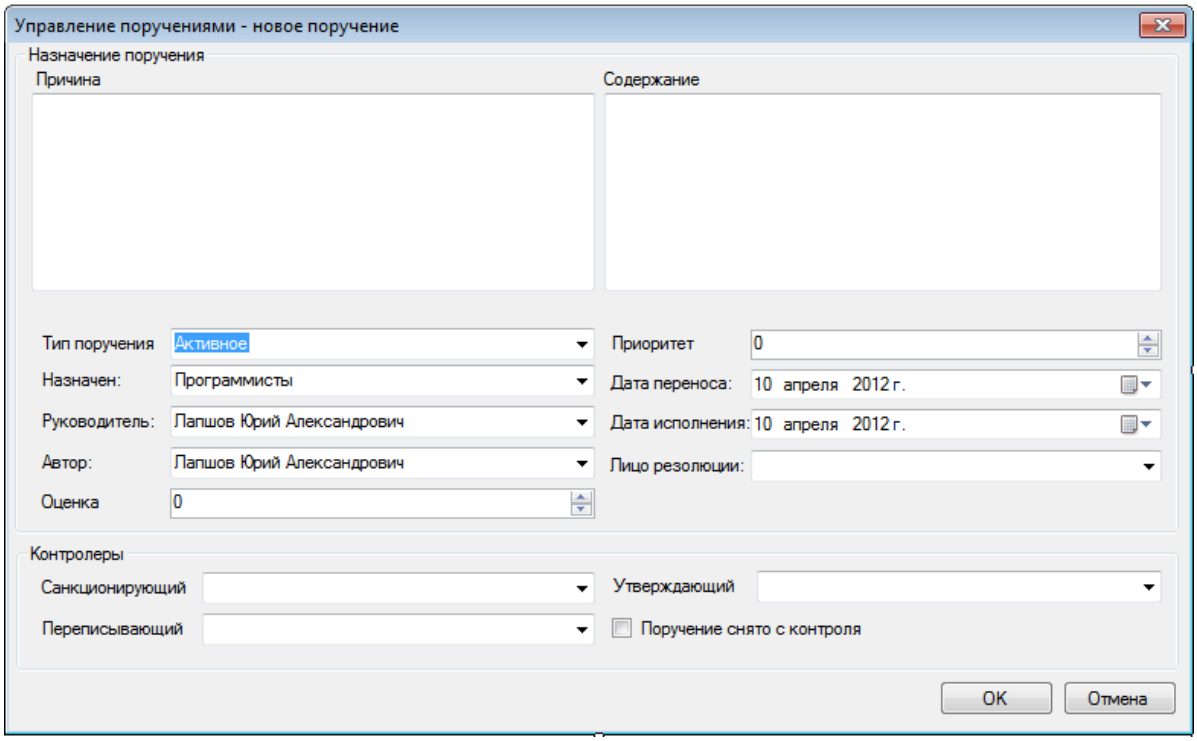


Рис. 7. Форма добавления нового поручения

Заключение

Псевдокодовое моделирование примеров проектных решений с обращениями к базам данных показало пригодность применения средств псевдокодового программирования для моделирования и реализации таких проектных решений. К основным позитивным отличиям разработанных средств прототипирования от известных относятся: возможность имитации доступа к макету любого фрагмента проектируемой базы данных; интеграция на прототипах совокупности проектных решений, затрагивающих доступ к базе данных, программные решения и действия пользователей; основным механизмом интеграции разнородных проектных рекшений в прототипах является исполняемое интерфейсное связывание.

СПИСОК ЛИТЕРАТУРЫ

1. Дейт К. Дж. Введение в системы баз данных – 8-е изд. – М.: «Вильямс», 2006. – С. 1328.

2. Reinhard Budde, Philip Bacon. Prototyping: an approach to evolutionary system development. Springer-Verlag, 2011, pp. 43–47.

3. Соснин П.И. Программирование человеко-компьютерной деятельности / Saarbruken: Lambert Academic Publishing, 2012. – С. 343 .

4. Sosnin P. Question-Answer Shell for Personal Expert System // Chapter in the book “Expert Systems for Human, Materials and Automation.” Published by Intech, 2011. – pp. 51–74.

5. S. Dasgupta, C.H. Papadimitriou, and U.V. Vazirani. Algorithms // Chapter in the book 4. McGraw-Hill, 2006.

6. Sosnin P. Question-Answer Approach to Human-Computer Interaction in Collaborative Designing. Chapter in the book “Cognitively Informed Intelligent Interfaces: Systems Design and Development” Published IGI Global, (2012). – pp. 157–176.

7. Молчанов А.Ю. Системное программное обеспечение.– СПб. : Питер, 2006. – С. 21–24.