ИЗВЕСТИЯ
ПЕНЗЕНСКОГО ГОСУДАРСТВЕННОГО
ПЕДАГОГИЧЕСКОГО УНИВЕРСИТЕТА
имени В. Г. БЕЛИНСКОГО
ФИЗИКО-МАТЕМАТИЧЕСКИЕ И ТЕХНИЧЕСКИЕ НАУКИ
№ 18 (22) 2010

IZVESTIA
PENZENSKOGO GOSUDARSTVENNOGO
PEDAGOGICHESKOGO UNIVERSITETA
imeni V. G. BELINSKOGO
PHYSICAL, MATHEMATICAL AND TECHNICAL SCIENCES
№ 18 (22) 2010

УДК 004.657

ОБРАБОТКА ДАННЫХ В ИНФОРМАЦИОННОЙ СИСТЕМЕ С ДИНАМИЧЕСКИМ СООТВЕТСТВИЕМ МОДЕЛИ ПРЕДМЕТНОЙ ОБЛАСТИ И СХЕМЫ БАЗЫ ДАННЫХ

© В. В. ДРОЖДИН, Р. Е. ЗИНЧЕНКО
Пензенский государственный педагогический университет им. В. Г. Белинского,
кафедра прикладной математики и информатики
e-mail: drozhdin@spu-penza.ru

Дрождин В. В., Зинченко Р. Е. – Обработка данных в информационной системе с динамическим соответствием модели предметной области и схемы базы данных // Известия ПГПУ им. В. Г. Белинского. 2010. № 18 (22). С. 145–150. – *Для информационных систем, поддерживающих модель системно-изоморфного динамического соответствия концептуальной модели предметной области и схемы базы данных, предложен механизм изменения объемов понятий в форме операций поиска, добавления, модификации и удаления данных. Реализация операций обработки данных в системе осуществляется на основе системы базовых SQL-запросов. Для конкретной модели предметной области приведены примеры формирования базовых SQL-запросов и их использования в операциях обработки данных.*

Ключевые слова: информационная система, модель предметной области, база данных, системный изоморфизм, обработка данных.

Drozhdin V. V., Zinchenko R. E. – Data processing in the information system with a dynamic correspondence of data domain model and database scheme // Izv. Penz. gos. pedagog. univ. im.i V. G. Belinskogo. 2010. № 18 (22). P. 145–150. – *For the information systems supporting the model of system – isomorphic dynamic correspondence of conceptual data domain model and database scheme, the article suggests the mechanism of changing the entities' volumes in the form of search, addition, modification and deletion of data. The realization of data processing in the system is made on basic SQL-queries. The examples of forming the basic SQL-queries and their use in the operations of data processing for the concrete model of data domain are given.*

Keywords: information system, data domain model, database, system isomorphism, data processing.

Автоматизированная информационная система (АИС) с системно-изоморфным динамическим соответствием концептуальной модели предметной области (ПО) и схемы базы данных (БД) не требует проектирования и разработки, так как формируется автоматически на основе концептуальной модели ПО, задаваемой пользователями системы. Для создания полнофункциональной АИС, кроме реализации механизма соответствия модели ПО и схемы БД, необходимо дать пользователю возможность производить обработку данных, т.е. требуется реализация операций изменения объемов понятий.

Данную задачу целесообразно решать на основе системы базовых SQL-запросов. В работах [1–4] рассмотрено формирование системы базовых SQL-запросов, реализующих механизм соответствия концептуальной модели ПО и схемы БД и используемых для организации сложных вычислений и поиска данных. Поэтому разработаем подобную систему базовых SQL-запросов, позволяющих реализовать операции изменения объемов понятий в форме ввода, модификации и удаления данных.

В АИС концептуальная модель ПО представляется системой понятий, а БД формируется в рамках реляционной модели данных. Для отображения понятий в БД и обработки данных для каждого понятия концептуальной модели ПО целесообразно создавать следующие SQL-запросы:

1) для понятия-агрегата, понятия-компонента, родового понятия, видового понятия, понятия-класса и конкретного понятия создаются базовые SQL-запросы типа select, insert, update и delete;

2) для понятия-образа создается базовый SQL-запрос только типа select. Базовые SQL-запросы других типов для понятия-образа не создаются, т.к. понятие-образ является представлением данных с высокой степенью абстрагирования и обобщения, а объекты его объема формируется как интегральные объекты из объектов конкретного понятия. Вследствие этого операции типа insert, update и delete не применимы к объему понятия-образа.

3) для понятия-подкласса создается базовый SQL-запрос только типа select. Базовые SQL-запросы других типов для понятия-подкласса не создаются, т.к. для обработки данных, соответствующих объему понятия-подкласса, используются базовые SQL-запросы типа insert, update и delete понятия-класса. При этом АИС контролирует, чтобы модифицируемые, добавляемые или удаляемые данные не выходили за рамки предиката, формирующего объем понятия-подкласса.

Приведем примеры базовых SQL-запросов. Для иллюстрации разработанного метода будем использовать фрагмент модели ПО «Учебный процесс вуза», приведенный на рис. 1.

Схема БД для фрагмента модели ПО приведена на рис. 2.

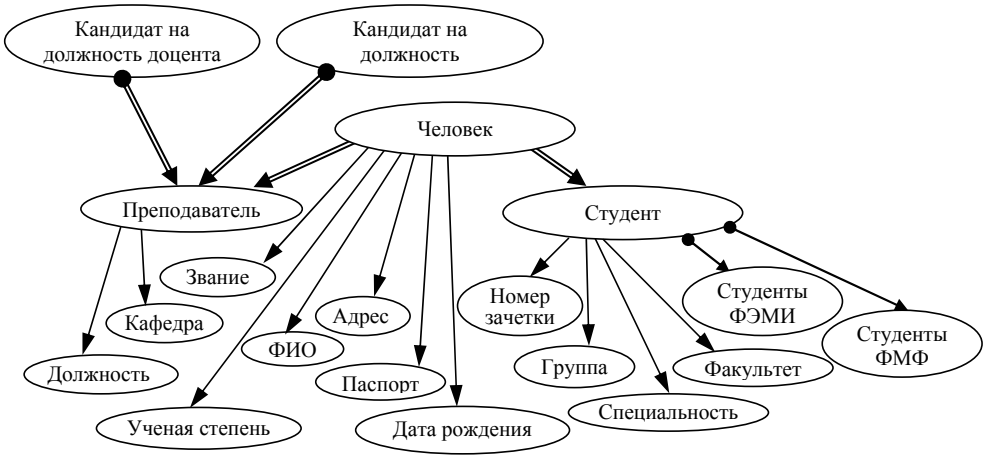

Рис. 1. Фрагмент модели ПО «Учебный процесс ВУЗа».
Стрелки на рисунке 1 имеют следующую интерпретацию [5]:
- отношение агрегации: ⟶ ;
- отношение классификации: ⊷ ;
- отношение обобщения: ⟹ ;
- отношение абстрагирования: ⊷ .

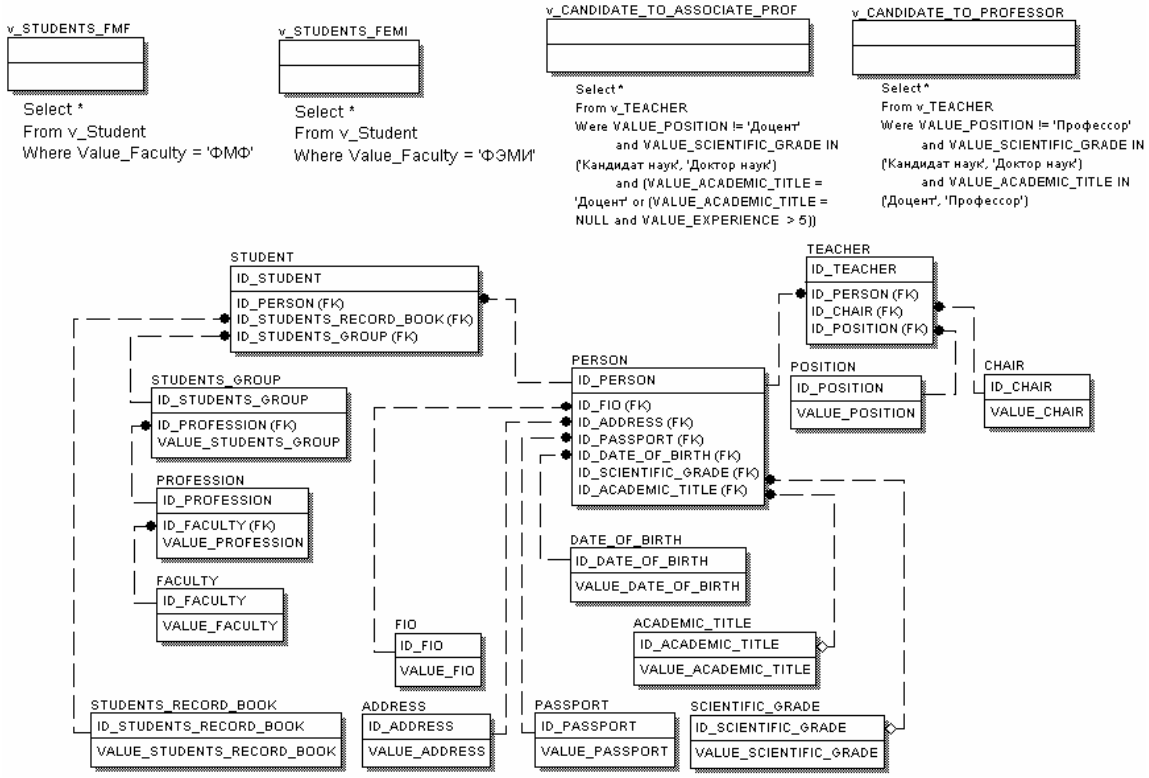

Рис. 2. Схема БД для фрагмента ПО «Учебный процесс ВУЗа».

Приведем пример наборов базовых SQL-запросов для различных понятий модели ПО.

Для простого понятия “ФИО”, не включающего в свой состав других понятий, будет сформирован следующий набор базовых SQL-запросов:

1. запрос на выборку данных (select):

```
CREATE OR REPLACE VIEW v_ФИО AS
Select *
FROM ФИО
```

2. запрос на добавление данных (insert):

```
INSERT INTO ФИО values (ID_ФИО, Value_ФИО)
```

3. запрос на модификацию данных (update):

```
UPDATE ФИО
SET Value_ФИО = new_Value_ФИО
Where ID_ФИО = current_ID_ФИО
```

4. запрос на удаление данных (delete):

```
DELETE FROM ФИО Where ID_ФИО = current_
ID_ФИО
```

Для сложного понятия “Человек” (понятие-агрегат), связанного отношением агрегации с понятиями “ФИО”, “Паспорт”, “Адрес”, “Дата рождения”, “Ученая степень”, “Звание” (понятия-компоненты), будет сформирован следующий набор базовых SQL-запросов:

1. запрос на выборку данных (select):

```
CREATE OR REPLACE VIEW v_Человек AS
SELECT *
FROM  Человек
    NATURAL JOIN v_ФИО
    NATURAL JOIN v_Адрес
    NATURAL JOIN v_Паспорт
    NATURAL JOIN v_Дата_рождения
    NATURAL JOIN v_Ученая_степень
    NATURAL JOIN v_Звание
```

где v_V - базовый SQL-запрос, формирующий объем понятия V.

2. запрос на добавление данных (insert):

```
INSERT into Человек
Values (ID_Человек,
          ID_ФИО,
          ID_Адрес,
          ID_Паспорт,
          ID_Дата_рождения,
          ID_Ученая_степень,
          ID_Звание)
```

3. запрос на модификацию данных (update):

```
UPDATE Человек
SET ID_ФИО = new_ID_ФИО,
      ID_Адрес = new_ID_Адрес,
      ID_Паспорт = new_ID_Паспорт,
      ID_Дата_рождения = new_ID_Дата_рождения,
      ID_Ученая_степень = new_ID_Ученая_степень,
      ID_Звание = new_ID_Звание
WHERE ID_Человек = current_ID_Человек
```

4. запрос на удаление данных (delete):

```
DELETE FROM Человек
WHERE ID_Человек = current_ID_Человек
```

У остальных понятий модели ПО (за исключением указанных выше случаев) также создаются аналогичные наборы базовых SQL-запросов.

Имея систему базовых SQL-запросов для концептуальной модели ПО, представляется возможным эффективно решать задачи, связанные с выборкой, вводом, модификацией и удалением данных в системе.

Как видно из приведенных наборов базовых SQL-запросов, реализация операций изменения объема простого понятия будет осуществляться путем выполнения соответствующего SQL-запроса. Поэтому далее рассматривается обработка данных на примере реализации операций изменения объема сложного понятия.

В общем случае обработка данных, составляющих объем сложного понятия, осуществляется следующим образом:

- выборка данных об объектах определенного понятия модели ПО осуществляется с помощью базового SQL-запроса типа select для этого понятия [3, 4];

- включение нового объекта в объем некоторого понятия модели ПО реализуется путем последовательного выполнения базовых SQL-запросов типа select и insert для всех понятий, входящих в состав этого понятия, и выполнения базового SQL-запроса типа insert для данного понятия;

- модификация информации об объекте какого-либо понятия модели ПО осуществляется путем последовательного выполнения базовых SQL-запросов типа select, insert и update для всех понятий, входящих в состав этого понятия, и выполнения базового SQL-запроса типа update для данного понятия;

- удаление объекта из объема понятия модели ПО производится путем выполнения базового SQL-запроса типа delete для данного понятия.

Рассмотрим подробнее реализацию операций обработки данных на основе базовых SQL-запросов в системе с поддержкой модели системно-изоморфного динамического соответствия концептуальной модели ПО и схемы БД на примере понятия “Человек”.

Ввод данных

При добавлении пользователем информации о новом человеке (например, ФИО = “Иванов Андрей Николаевич”, Адрес = “Пенза”, Паспорт = “123456”, Дата рождения = “30.06.1970”, Ученая степень = “Кандидат наук”, Звание = “Доцент”) сначала извлекаются и выполняются базовые SQL-запросы типа select для понятий “ФИО”, “Адрес”, “Паспорт”, “Дата рождения”, “Ученая степень” и “Звание”. Если объект «Иванов Андрей Николаевич» уже содержится в объеме понятия «ФИО», то будет получен его идентификатор. Аналогично будут получены идентификаторы объектов “Пенза”, “123456”, “30.06.1970”, “Кандидат наук” и “Доцент” из объемов понятий “Адрес”, “Паспорт”, “Дата рождения”, “Ученая степень” и “Звание” соответственно.

В случае отсутствия какого-либо объекта в объеме соответствующего понятия, он автоматически будет добавлен с помощью базового SQL-запроса типа insert. Механизм применения базовых SQL-запросов

типа insert заключается в следующем: текст запроса insert посредством применения операции замены (Replace) трансформируется в текст реальной DML-инструкции. Для понятия “ФИО” базовый SQL-запрос типа insert

INSERT into ФИО values (ID_ФИО, Value_ФИО)

преобразуется в DML-инструкцию

INSERT into ФИО values (1, 'Иванов Андрей Николаевич')

путем замены ID_ФИО на сгенерированное значение “1” и замены Value_ФИО на введенное пользователем значение “Иванов Андрей Николаевич”.

Для остальных понятий рассматриваемого примера базовые SQL-запросы типа insert

INSERT into Адрес values (ID_Адрес, Value_Адрес)
INSERT into Паспорт values (ID_Паспорт, Value_Паспорт)
INSERT into Дата_рождения values (ID_Дата_рождения, Value_Дата_рождения)
INSERT into Ученая_степень values (ID_Ученая_степень, Value_Ученая_степень)
INSERT into Звание values (ID_Звание, Value_Звание)

преобразуются в DML-инструкции

INSERT into Адрес values (10, 'Пенза')
INSERT into Паспорт values (20, '123456')
INSERT into Дата_рождения values (30, '30.06.1970')
INSERT into Ученая_степень values (40, 'Кандидат наук')
INSERT into Звание values (50, 'Доцент')

Идентификаторы, полученные в результате выполнения DML-инструкций, используются в базовом SQL-запросе типа insert для понятия “Человек”. При этом базовый SQL-запрос

INSERT into Человек values (ID_Человек, ID_ФИО, ID_Адрес, ID_Паспорт, ID_Дата_рождения, ID_Ученая_степень, ID_Звание)

преобразуется в DML-инструкцию

INSERT into Человек values (100, 1, 10, 20, 30, 40, 50)

где 100 – сгенерированное значение идентификатора объекта, добавляемого в объем понятия “Человек”;
1, 10,20, 30, 40, 50 – идентификаторы объектов понятий более низкого уровня, входящих в состав объекта понятия “Человек”.

Модификация данных

Рассмотрим изменение объема понятия путем модификации информации о некотором объекте. Например, необходимо отредактировать объект <Иванов Андрей Николаевич, Пенза, 123456, 30.06.1970, Кандидат наук, Доцент> для приведения его к виду <Иванцов Андрей Николаевич, Пенза, 123456, 30.06.1970, Кандидат наук, Доцент> (изменилось ФИО). В этом случае сначала извлекаются и выполняются базовые SQL-запросы типа select для понятий “ФИО”, “Адрес”, “Паспорт”, “Дата рождения”, “Ученая степень” и “Звание”. Если объект “Иванцов Андрей Николаевич” уже содержится в объеме понятия “ФИО”, то будет получен его идентификатор.

В противном случае извлекается и выполняется базовый SQL-запрос типа insert для понятия “ФИО” и возвращается идентификатор добавленного объекта “Иванцов Андрей Николаевич”. Для этого базовый SQL-запрос

INSERT into ФИО values (ID_ФИО, Value_ФИО)

преобразуется в DML-инструкцию

INSERT into ФИО values (2, 'Иванцов Андрей Николаевич')

путем замены ID_ФИО на сгенерированное значение “2” и Value_ФИО на введенное пользователем значение “Иванцов Андрей Николаевич”.

Для объектов “Пенза”, “123456”, “30.06.1970”, “Кандидат наук” и “Доцент” из объемов понятий “Адрес”, “Паспорт”, “Дата рождения”, “Ученая степень” и “Звание” соответственно будут получены их идентификаторы.

Идентификаторы объектов затем используются в базовом SQL-запросе типа update для понятия “Человек” путем применения операции замены (Replace) к тексту запроса. Для этого базовый SQL-запрос

```
UPDATE Человек
SET ID_ФИО = new_ID_ФИО,
    ID_Адрес = new_ID_Адрес,
    ID_Паспорт = new_ID_Паспорт,
    ID_Дата_рождения = new_ID_Дата_рождения,
    ID_Ученая_степень = new_ID_Ученая_степень,
    ID_Звание = new_ID_Звание
WHERE ID_Человек = current_ID_Человек
```

преобразуется в DML-инструкцию

```
UPDATE Человек
SET ID_ФИО = 2,
    ID_Адрес = 10,
    ID_Паспорт = 20,
    ID_Дата_рождения = 30,
    ID_Ученая_степень = 40,
    ID_Звание = 50
WHERE ID_Человек = 100
```

где 100 – идентификатор модифицируемого объекта понятия “Человек”;
2, 10, 20, 30, 40, 50 – идентификаторы объектов понятий более низкого уровня, входящих в состав объекта понятия “Человек”.

Удаление данных

Если пользователь производит удаление какого-либо объекта (например, объекта <Иванцов Андрей Николаевич, Пенза, 123456, 30.06.1970, Кандидат наук, Доцент>), то извлекается и выполняется базовый SQL-запрос типа delete для соответствующего понятия. Например, для понятия “Человек” базовый SQL-запрос типа delete

```
DELETE FROM Человек
    WHERE ID_Человек = current_ID_Человек
```

преобразуется в DML-инструкцию

```
DELETE FROM Человек
    WHERE ID_Человек = 100
```

путем замены current_ID_Человек на значение идентификатора удаляемого объекта (100).

Рассмотренный пример демонстрирует обработку данных в системе на основе метода базовых

SQL-запросов для понятий, связанных отношением агрегации.

Для понятий, связанных отношением обобщения, метод работает аналогично. При изменении объемов понятий путем добавления новых или изменения существующих объектов сначала извлекаются и выполняются базовые SQL-запросы типа select, insert и update для родового понятия, затем полученные идентификаторы используются в базовых SQL-запросах типа select, insert и update для видового понятия. Удаление объекта из объема понятия осуществляется по идентификатору текущего объекта понятия (видового или родового).

Приведем пример изменения объема понятия “Преподаватель”, связанного отношением обобщения с понятием “Человек” (рисунок 1). Например, пользователь добавляет преподавателя <Иванцов Андрей Николаевич, Пенза, 123456, 30.06.1970, Кандидат наук, Доцент, ПМИ, Профессор>. Для этого последовательно (как показано выше) извлекаются и выполняются базовые SQL-запросы типа select и insert для понятий “ФИО”, “Адрес”, “Паспорт”, “Дата рождения”, “Ученая степень”, “Звание”, “Кафедра”, “Должность”, “Человек”.

Полученные идентификаторы соответствующих объектов затем используются в базовом SQL-запросе типа insert для понятия “Преподаватель” путем применения операции замены (Replace) к тексту базового SQL-запроса. При этом базовый SQL-запрос

INSERT into Преподаватель values (ID_Преподаватель, ID_Человек, ID_Кафедра, ID_Должность)

преобразуется в DML-инструкцию

INSERT into Преподаватель values (200, 100, 60, 70)

где 200 – сгенерированное значение идентификатора для объекта, добавляемого в объем понятия “Преподаватель”;

100 – идентификатор объекта из объема родового понятия “Человек”, которому соответствует добавляемый объект видового понятия “Преподаватель”;

60, 70 – идентификаторы объектов понятий более низкого уровня, входящих в состав объекта понятия “Преподаватель”.

Рассмотрим изменение объемов понятий, связанных отношением классификации.

Для понятий-подклассов создаются базовые SQL-запросы только типа select, так как объем понятия-подкласса является ограничением (на основании соответствующего предиката) объема понятия-класса. Следовательно, весь объем понятия-подкласса целиком содержится в объеме соответствующего понятия-класса. Поэтому для ввода, модификации и удаления данных для понятия-подкласса используются базовые SQL-запросы типа insert, update и delete понятия-класса. При этом роль АИС заключается в контроле соответствия добавляемого объекта понятия-подкласса основанию классификации, установленному предикатом. Например, пусть в АИС содержатся сведения о факультетах, приведенных на рис. 3.

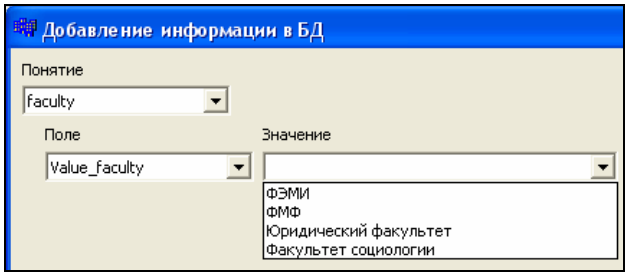

Рис. 3. Информация о факультетах.

Как показано в концептуальной модели ПО (рисунок 1) понятия “Студенты ФМФ” и “Студенты ФЭМИ” являются понятиями-подклассами для понятия-класса “Студент”.

Добавление нового объекта в объем понятия-класса осуществляется аналогично добавлению объекта в объем понятия-агрегата. При этом для выбора значения факультета для добавляемого объекта из объема понятия “Студент” доступны все факультеты, составляющие объем понятия “Факультет” (рис. 4).

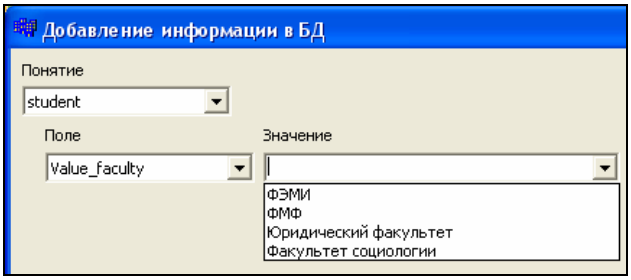

Рис. 4. Объекты из объема понятия “Факультет”, доступные для выбора при вводе нового объекта понятия “Студент”.

Однако для выбора значения факультета для объекта, добавляемого в объем понятия-подкласса “Студенты ФЭМИ”, доступны не все факультеты, имеющиеся в АИС, а только те, которые удовлетворяют предикату, формирующему объем данного понятия-подкласса (рис. 5).

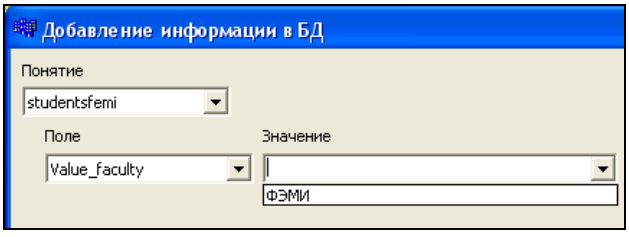

Рис. 5. Объекты из объема понятия “Факультет”, доступные для выбора при вводе нового объекта понятия “Студенты ФЭМИ”.

Таким образом, для понятий, связанных отношением классификации, обработка данных также реализуется посредством метода базовых SQL-запросов.

Рассмотрим обработку данных для понятий, связанных отношением абстрагирования. Как указано выше, для понятий-образов создаются базовые SQL-запросы только типа select. Это определяется тем, что объем понятия-образа не относится к первичным данным информационной системы. Поэтому объекты понятия-образа модифицируются автоматически при модификации объектов конкретных понятий.

Реализация предложенного метода позволяет конечным пользователям создавать АИС путем постепенного расширения и уточнения концептуальной модели ПО в процессе работы с системой, выполняя при этом обычные для них операции выборки, ввода, модификации и удаления данных. Такой подход не требует привлечения разработчиков программного обеспечения – система автоматически поддерживает системно-изоморфное динамическое соответствие концептуальной модели ПО и схемы БД, что позволяет пользователям в полном объеме использовать имеющуюся в системе информацию независимо от происходящих изменений в модели ПО (добавление новых понятий, установление новых связей между понятиями и т.д.).

Для апробации предложенного подхода организации АИС разработано экспериментальное программное обеспечение, поддерживающее системно-изоморфное динамическое соответствие концептуальной модели ПО и схемы БД с их независимой эволюцией [3], и реализующее обработку данных, изложенную в настоящей статье. По результатам апробации можно сделать следующие выводы:

1. Предложенный подход к организации и функционированию АИС является вполне работоспособным и позволяет сделать серьезный шаг к созданию самоорганизующихся информационных систем.

2. Для создания эффективных АИС с предложенной организацией в первую очередь необходима реализация активных БД, способных самостоятельно эволюционировать для обеспечения требуемого уровня эффективности обработки данных.

3. Необходима реализация всех механизмов концептуального моделирования ПО, предложенных в [5], для представления в АИС реального мира с требуемой полнотой и точностью и интенсификации взаимодействия пользователей с системой на основе этой модели.

4. Необходима реализация пользовательского интерфейса АИС, предоставляющего пользователям, не являющимся специалистами в области компьютерной техники, с одной стороны, очень широкие возможности по конструированию, настройке и контролю системы, а с другой – естественное взаимодействие пользователей с системой.

Таким образом, разработанная программная система-оболочка при всех ее недостатках может эффективно применяться для решения следующих задач:

1. Использоваться в качестве системы-прототипа, не требующего программирования, при создании больших информационных систем.

2. Использоваться как реально эксплуатируемая система для создания небольших (настольных) информационных систем.

СПИСОК ЛИТЕРАТУРЫ

1. Дрождин В.В., Зинченко Р.Е. Формирование системы SQL-запросов для отображения объектного пользовательского представления предметной области в базу данных // Проблемы информатики. 2008. № 1. С. 48–50.
2. Дрождин В.В., Зинченко Р.Е. Модификация системы SQL-запросов при изменении пользовательского объектного представления предметной области // Известия ПГПУ им. В.Г. Белинского. Серия физико-математические и технические науки. 2008. №8 (12). С. 106–110.
3. Зинченко Р.Е. Системно-изоморфное динамическое соответствие концептуальной модели предметной области и схемы базы данных // Программные продукты и системы. 2010. № 1. С. 71–75.
4. Дрождин В.В., Зинченко Р.Е., Герасимова Е.В. Обобщенная операция абстрагирования как реализация принципа открытости самоорганизующейся информационной системы // Программные продукты и системы. 2010. № 2. С. 143–151.
5. Дрождин В.В., Зинченко Р.Е. Системный подход к концептуальному моделированию предметной области в самоорганизующейся информационной системе // Программные продукты и системы. 2009. № 4. С. 73–79.