па Queue позволяет эффективно управлять последовательностью записей фиксированной длины. Он использует механизмы блокировки и протоколирования, оптимизированные для работы в условиях интенсивного многопользовательского доступа к голове и хвосту очереди.

Метод Recno позволяет обрабатывать данные, хранимые в виде записей фиксированной и переменной длины. Он автоматически генерирует номера записей, начиная с единицы. Имеется возможность задать системе автоматическую перенумерацию записей после добавления или удаления записей с меньшими номерами, что позволяет поддерживать плотный список номеров и вставлять данные между существующими записями.

Таким образом, в современных СУБД используются методы доступа, организованные преимущественно на основе B- или B^+-деревьев. Использование деревьев обусловлено предположением равновероятного доступа к любым данным и незначительным ростом высоты дерева при увеличении количества данных. Как известно, в B-деревьях время доступа к данным и сложность их обработки имеют логарифмическую зависимость, что дает им преимущество по сравнению с линейными структурами данных. Однако часто такое предположение неверно. Поэтому в механизмах хранения применяются комбинированные методы доступа, использующие преимущества других структур данных: множеств, последовательностей, хеш-таблиц и т. д. Для достижения высокой производительности необходимо явным образом задавать (настраивать) параметры конфигурации. АБД может изменять размеры страниц, файлов данных, буфера (кэша) и др. параметры, специфичные для каждой ФСД. Замена одних структур другими возможна, но она также выполняется человеком. Для этого создается новая ФСД, в которую копируются данные из исходной структуры. После чего исходная структура удаляется.

СПИСОК ЛИТЕРАТУРЫ

1. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. М.: Мир, 1979. 536 с.
2. Кнут Д. Искусство программирования на ЭВМ: сортировка и поиск. Т. 3. М.: Мир, 1978. 844 с.
3. Мартин Дж. Организация баз данных в вычислительных системах. М: Мир, 1980. 662 с.
4. Дрождин В. В. Организация адаптивного индекса // Математика и информатика: Межвузовский сборник. Пенза: ПГПУ, 1996. С. 80–85.
5. Ордынцев В. М. Системы автоматизации экспериментальных научных исследований. М.: Машиностроение, 1984. 328 с.
6. MySQL. Руководство администратора. М.: Издательский дом «Вильямс», 2005. 624 с.

УДК 681.3

МОДИФИКАЦИЯ СИСТЕМЫ SQL-ЗАПРОСОВ ПРИ ИЗМЕНЕНИИ ПОЛЬЗОВАТЕЛЬСКОГО ОБЪЕКТНОГО ПРЕДСТАВЛЕНИЯ ПРЕДМЕТНОЙ ОБЛАСТИ

В. В. ДРОЖДИН, Р. Е. ЗИНЧЕНКО
Пензенский государственный педагогический университет имени В. Г. Белинского
кафедра прикладной математики и информатики

Рассмотрена проблема изменения системы базовых SQL-запросов, поддерживающих объектное представление предметной области, при изменении пользовательских представлений и предложен алгоритм автоматической генерации базовых запросов.

При создании автоматизированных информационных систем (АИС), способных самостоятельно поддерживать внешние (пользовательские) представления предметной области (ПО) необходимо дать системе метод (алгоритм), с помощью которого она сможет автоматически приводить в корректное состояние структуру базы данных (БД) и систему базовых SQL-запросов, отображающих пользовательские представления объектов ПО в БД.

В качестве примера рассмотрим предметную область “Университет”, включающую специальности, которым обучаются в университетах. Предположим, что абитуриент решил поступить на математическую специальность некоторого вуза. Поэтому для данного абитуриента имеет ценность информация о вузах и предлагаемых ими математических специальностях. Внешнее представление пользователя состоит из двух сложных понятий “вуз” и “специальность”, содержащих элементарные (простые) понятия: название, адрес, ректор, код, квалификация, и выглядит следующим образом:

вуз (название, адрес, ректор)

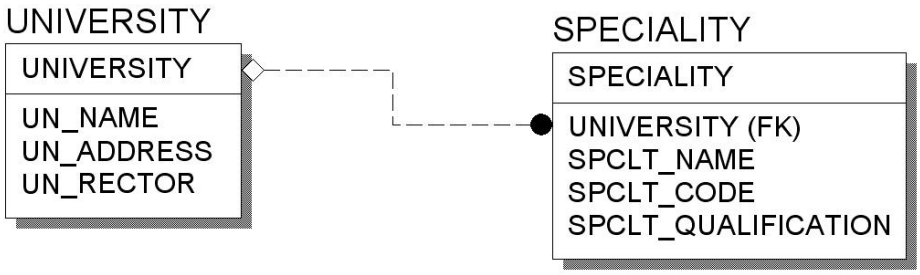
специальность (вуз, название, код, квалификация)

При этом “название” в объекте “вуз” является название вуза, а “название” в объекте “специальность” – название специальности. Свойство (атрибут) “вуз” в объекте “специальность” является ссылкой на объект “вуз” и указывает, что все свойства объекта “вуз” включаются в объект “специальность”.

Построим информационную модель ПО в форме ER-модели.

Примем соглашения: 1) название первичного ключа отношения должно совпадать с названием

отношения; 2) название внешнего ключа отношения должно совпадать с атрибутом, на который ссылается данный внешний ключ; 3) все атрибуты отношений, имеющие различную семантику, должны иметь уникальные названия на уровне всей схемы.



*Примечание: схема не нормализована.

Чтобы получить информацию о всех математических специальностях всех университетов, необходимо произвести операцию естественного соединения таблиц UNIVERSITY (Университеты) и SPECIALITY (Специальности) и наложить условие на название специальности:

UNIVERSITY **JOIN** SPECIALITY

AND lower(SPECIALITY.SPCLT_NAME) **LIKE** "%матем%"

В результате выполнения данной операции получим таблицу вида:

Таблица 1

Университет. Название	**Университет. Адрес**	**Университет. Ректор**	**Специальность. Название**	**Специальность. Код**	**Специальность. Квалификация**
ПГПУ	Пенза	Казаков	«Математика и информатика»	0100253	Учитель математики и информатики
ПГПУ	Пенза	Казаков	«Математическое обеспечение и администрирование локальных сетей»	03297	математик-программист
ПГПУ	Пенза	Казаков	«Математика и физика»	072413	Учитель математики и физики
...	...	...	...	...	...
МГУ	Москва	Садовничий	«Математика и информатика»	0100253	бакалавр
...	...	...	...	...	...

Проанализировав результирующую таблицу, абитуриент видит, что, например, в ПГПУ существует несколько математических специальностей. Поэтому для того, чтобы сделать однозначный выбор специальности для абитуриента недостаточно понятий в существующем пользовательском представлении «университет-специальность». Данное представление перестало являться для абитуриента адекватным с точки зрения выбора им специальности. Поэтому абитуриент, логично рассудив, что университет состоит из факультетов, а на факультетах производится обучение соответствующим специальностям, меняет собственное представление, добавив к существующим понятиям новое - «факультет». Теперь представление пользователя состоит из трех понятий и выглядит следующим образом:

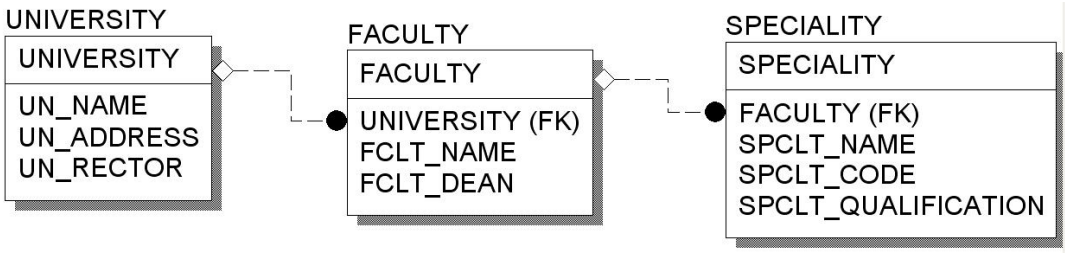
Университет (название, адрес, ректор)

Факультет (университет, название, декан)

Специальность (факультет, название, код, квалификация)

Построим информационную модель предметной области.

Чтобы получить информацию обо всех математических специальностях всех университетов, необходимо произвести операции естественного соединения таблиц UNIVERSITY (Университеты), FACULTY (Факультеты) и SPECIALITY (Специальности) и наложить условие на название специальности:



*Примечание: схема не нормализована.

UNIVERSITY **JOIN** FACULTY AND FACULTY **JOIN** SPECIALITY
AND lower(SPECIALITY.SPCLT_NAME) LIKE "%матем%"

Учитывая принятые выше соглашения, запишем операцию естественного соединения в краткой форме:

UNIVERSITY **JOIN** FACULTY **JOIN** SPECIALITY
AND lower(SPECIALITY.SPCLT_NAME) LIKE "%матем%"

В результате получим таблицу 2.

Таблица 2

Университет. Название	**Университет. Адрес**	**Университет. Ректор**	**Специальность. Название**	**Специальность. Код**	**Специальность. Квалификация**	**Факультет. Название**
ПГПУ	Пенза	Казаков	«Математика и информатика»	010253	Учитель математики и информатики	ФМФ
ПГПУ	Пенза	Казаков	«Математическое обеспечение и администрирование локальных сетей»	03297	математик-программист	ФЭМИ
ПГПУ	Пенза	Казаков	«Математика и физика»	072413	Учитель математики и физики	ФМФ
...	...	...	...	...	...	...
МГУ	Москва	Садовничий	«Математика и информатика»	010253	бакалавр	Физико-мат. фак-т
...	...	...	...	...	...	...

Проанализировав результирующую таблицу, абитуриент видит, что, например, в ПГПУ обучение математическим специальностям ведется на двух факультетах: ФМФ (две специальности) и ФЭМИ (одна специальность).

Приведенный пример достаточно наглядно демонстрирует основную проблему существующих автоматизированных информационных систем: информационная система в определенный момент времени перестает отвечать требованиям пользователей, поскольку со временем происходит неизбежное изменение реального мира, а вместе с ним и представления пользователей о ПО. Однако в АИС модель ПО жестко заложена в структуру БД. Поэтому для дальнейшего использования АИС необходимо перепроектировать схему БД, изменить систему SQL-запросов и т. д.

С технической точки зрения, ключевыми являются два аспекта:

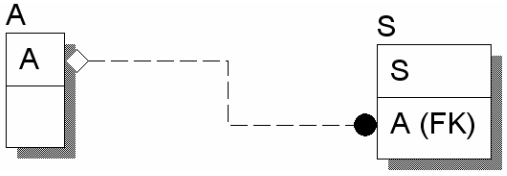
1) изменение схемы БД в соответствии с новым внешним представлением пользователя;

2) изменение системы SQL-запросов – приведение системы SQL-запросов в соответствие с новой схемой БД.

Создание АИС нового типа, способных самостоятельно решать указанные задачи позволит преодолеть недостатки существующих информационных систем.

Использование операции естественного соединения отношений с учетом приведенных выше соглашений позволяет эффективно переформулировать SQL-запросы при модификации схемы БД.

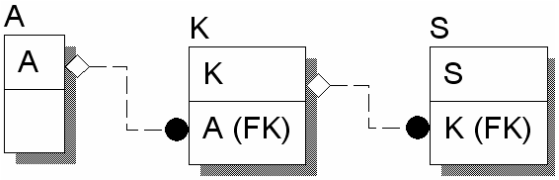
Например, пусть имеется схема БД:



Для того чтобы выбрать все S с учетом данных из A, необходимо выполнить операцию

A **JOIN** S.

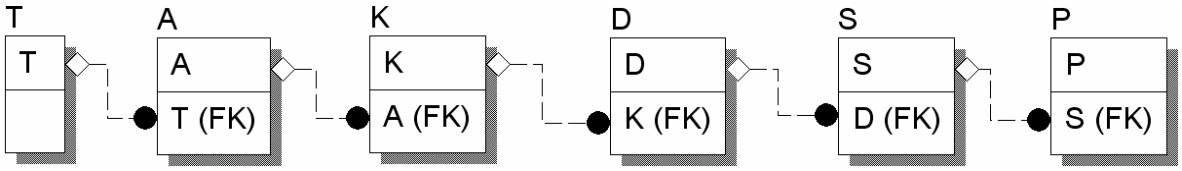
Допустим, схема претерпевает изменение вида:



Тогда чтобы выбрать все S с учетом данных из A, необходимо выполнить операцию

A **JOIN** K **JOIN** S.

Допустим, схема БД претерпевает дальнейшие изменения:



Тогда,

1) выборка информации о K с учетом данных из T задается запросом:

T **JOIN** A **JOIN** K

2) выборка информации о P с учетом данных из T задается запросом:

T **JOIN** A **JOIN** K **JOIN** D **JOIN** S **JOIN** P

И так далее.

На языке SQL данные запросы выглядят следующим образом:

1) Select *
From T, A, K
Where T.T = A.T
AND A.A = K.A

2) Select *
From T, A, K, D, S, P
Where T.T = A.T
AND A.A = K.A
AND K.K = D.K
AND D.D = S.D
AND S.S = P.S

В общем случае получаем следующую схему базового SQL-запроса:

Select *
From X1, X2, X3, ... , Xn
Where X1.X1 = X2.X1
AND X2.X2 = X3.X2
.....
AND Xn-1.Xn-1 = Xn.Xn-1

Однако не все системы управления базами данных (СУБД) позволяют задавать имена атрибутов, совпадающие с именами отношений. В этом случае в качестве идентификатора отношения можно использовать атрибут с именем ID, а атрибут, ссылающийся на внешний ключ задавать в виде ТАБЛИЦА_ID. При этом базовый SQL-запрос будет иметь вид:

Select *
From **X1, X2, X3, ... , Xn**
Where X1.ID = X2.X1_ID
AND X2.ID = X3.X2_ID
.....
AND Xn-1.ID = Xn.Xn-1_ID

Приведенные SQL-запросы на выборку данных допускают рекурсивную форму:

1) SELECT *
FROM T FULL JOIN A ON T.ID = A.T_ID

Обозначим соединение «T FULL JOIN A ON T.ID = A.T_ID» как **JOIN#1**.

Тогда:

2) SELECT *
FROM **JOIN#1** FULL JOIN K ON A.ID = K.A_ID

Обозначим соединение «**JOIN#1** FULL JOIN K ON A.ID = K.A_ID» как **JOIN#2**. Тогда:

3) SELECT *
FROM **JOIN#2** FULL JOIN D ON K.ID = D.K_ID

и так далее.

При изменении пользовательского представления о ПО, кроме изменения запросов на выборку данных, требуются запросы на создание таблиц (отношений) для представления новых объектов в БД. SQL-запросы для создания таблиц задаются следующим образом:

CREATE **TABLE T**
(
ID INTEGER CONSTRAINT TID PRIMARY KEY,
...
)

CREATE TABLE A
(
ID INTEGER CONSTRAINT AID PRIMARY KEY,

```
T_ID INTEGER CONSTRAINT A_T_ID REFERENCES T,
...
)
```

или в общем виде:

```
CREATE TABLE X1
(
ID INTEGER CONSTRAINT X1ID PRIMARY KEY,
...
)
CREATE TABLE X2
(
ID INTEGER CONSTRAINT X2ID PRIMARY KEY,
X2_ID INTEGER CONSTRAINT X2_X1_ID REFERENCES T,
...
)
```

Приведем алгоритм построения системы базовых SQL-запросов для известного пользовательского представления ПО и заданной схемы БД. Исходными данными для алгоритма являются:

- объектное представление ПО;
- схема БД;
- набор таблиц БД для каждого объекта ПО.

Алгоритм построения системы базовых SQL-запросов имеет следующий вид:

1) выбрать из множества объектов ПО минимальный сложный объект, которому соответствует сложное понятие, содержащее в своем составе только простые понятия (такой объект хотя бы один обязательно есть в любой ПО);

2) если выбранный объект представляется одной таблицей в БД, то создать базовый SQL-запрос на выборку данных из одной таблицы и перейти к п. 6, иначе перейти к п. 3;

3) среди таблиц, представляющих объект ПО, выбрать таблицу, не содержащую внешних ключей;

4) найти таблицы, ссылающиеся на таблицу из п. 3 и построить для каждой из них соединение типа **JOIN#1**;

5) найти таблицы, ссылающиеся на таблицы уже вошедшие в базовый SQL-запрос и построить для каждой из них соединение типа **JOIN#2**. П. 5 выполнять до тех пор, пока все таблицы, представляющие объект ПО, не будут включены в SQL-запрос;

6) выбрать из оставшегося множества объектов ПО минимальный объект, которому соответствует сложное понятие, содержащее в своем составе, как простые понятия, так и понятия с уже построенными базовыми SQL-запросами;

7) для выбранного объекта выполнить п. 2-5 (при выполнении п. 3 считать допустимыми таблицы, имеющими внешние ключи на таблицы уже вошедшие в базовые SQL-запросы);

8) п. 6-7 выполняются до тех пор, пока оставшееся множество объектов ПО не станет пустым.

Приведенный алгоритм формирует систему базовых SQL-запросов для всех объектов пользовательского представления ПО.

Предложенный метод создания отношений БД и модификации системы базовых SQL-запросов при изменении пользовательского представления ПО предоставляет эффективный алгоритм, позволяющий АИС автоматически поддерживать реализацию пользовательских представлений в БД.

УДК 681.3

ЭСКИЗ САМООРГАНИЗУЮЩЕЙСЯ СИСТЕМЫ

М. В. ЖУКОВ
Пензенский государственный педагогический университет имени В. Г. Белинского
кафедра прикладной математики и информатики

Система - это совокупность элементов и отношений между ними, образующая единое целое.

В зависимости от стабильности существования во времени системы делятся:

1) статические системы существуют в неизменном состоянии сколь угодно долго;

2) динамические системы изменяют свое состояние в процессе существования, причем переход системы из одного состояния в другое не может совершаться мгновенно, а происходит в результате переходного процесса. Важной особенностью динамических систем является их предсказуемость, базирующаяся на причинно-следственных связях, т.е. зная начальное состояние системы и закон перехода системы из состояния в состояние можно определить в каком состоянии будет находиться система в любой другой момент времени.

Среди динамических высокоорганизованных систем различают адаптивные, обучаемые, самообучаемые, самоорганизующиеся и развивающиеся системы:

1) адаптивной является система способная приспосабливаться к изменениям внешней среды и внутренней организации путем настройки своих параметров и изменения структуры. Адаптивная система обязательно накапливает информацию в процессе своего существования.

2) обучаемой является система способная получать информацию о закономерностях внешней среды и собственного поведения от внешнего источника (учителя) и использовать ее в дальнейшем функционировании;

3) самообучаемой является система способная самостоятельно выявлять закономерности поведения