УДК 004.658.2
Лысюк А.Н.
Научный руководитель: проф. Дереченник С.С.

РЕАЛИЗАЦИЯ КАСКАДНОГО УДАЛЕНИЯ ДАННЫХ ПРИ РАСЧЕТЕ ШТАТОВ ПРОФЕССОРСКО-ПРЕПОДАВАТЕЛЬСКОГО СОСТАВА УНИВЕРСИТЕТА

Определение штатного состава профессорско-преподавательского состава кафедр вуза выполняется на основе расчета объема их учебной работы на планируемый учебный год. Традиционно планирование базируется на исходных расчетах, выполняемых структурными подразделениями (кафедрами или факультетами). Те же факультеты и профилирующие кафедры одновременно выступают и разработчиками учебных планов соответствующих специальностей, поэтому подобные схемы допускают возможность завышения объема учебной работы в стремлении увеличить штатный состав структурных подразделений. Результатом является как возрастание годовой учебной нагрузки преподавателей, так и общей загрузки студентов различными видами учебных работ, что отнюдь не способствует повышению качества образовательного процесса.

В Брестском государственном техническом университете разработана программная система расчета штатной численности профессорско-преподавательского состава кафедр, лишенная данного недостатка благодаря централизованному и автоматизированному расчету объема учебной работы, необходимой для полной и качественной реализации базовых учебных планов по каждой специальности и форме обучения.

Для эффективной модификации исходных данных, располагающихся в базе данных (БД) типа Paradox, разработан и реализован новый метод каскадного удаления записей, основанный на оригинальном компоненте, обеспечивающем наследование существующего класса. Paradox (программный продукт фирмы Borland) - это признанный лидер на рынке систем управления базами данных. В 90-х годах Paradox фактически стал стандартом систем управления базами данных (СУБД) для персональных компьютеров.

Среди особенностей Paradox, повлиявших на выбор данной СУБД, можно выделить сочетание простоты и прозрачности с широкими возможностями функционально-завершенной системы управления данными. Результатом такого сочетания является мощная СУБД, доступная не только профессиональному программисту, но и пользователю, не имеющему глубоких познаний в программировании.

Суть каскадного удаления записей в БД Paradox, адаптированного нами для нужд разработанной системы, заключается в следующем: в одной транзакции производится удаление данной записи, а также всех других записей, ссылающихся на данную. Если на удаляемые записи также есть ссылки, то каскадное удаление продолжается дальше. Таким образом, после удаления данной записи в базе не остаётся ни одной записи, прямо или косвенно ссылающейся на неё. Если хотя бы одну из ссылающихся записей удалить не получается (либо для неё настроен запрет, либо происходит какая-либо ещё ошибка), то все удаления запрещаются.

Однако при выполнении каскадного удаления с помощью стандартного механизма BDE (Borland Database Engine) генерируется исключение (exception) с сообщением о том, что сначала необходимо удалить данные в связанной таблице. Рассмотрим данный механизм на примере связанных между собой таблиц «Кафедры» и «Дисциплины». Таблицы находятся в отношении 1:1 (рисунок 1). На рисунке жирным шрифтом выделены первичные ключи; внешним ключом является поле «код кафедры».

Предположим, что необходимо удалить некоторую кафедру, имеющую код кафедры, равный 6. При выполнении такой попытки BDE выдает сообщение о том, что сперва необходимо удалить записи в таблице «Дисциплины», ссылающиеся на данную кафедру. В результате процесс удаления разбивается на два последовательных шага:

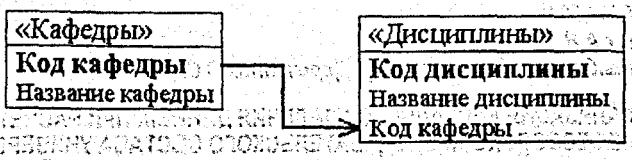

Рис. 1. Таблицы «Кафедры» и «Дисциплины» с отношением типа 1:1

1. Удаление всех дисциплин с кодом кафедры 6 из таблицы «Дисциплины», например, с помощью следующего SQL-запроса: **DELETE FROM «Дисциплины» WHERE «Код кафедры»=6**.

2. Удаление кафедры из таблицы «Кафедры» с помощью стандартной функции.

На данный момент это единственный известный способ решения проблемы. Однако он обладает существенным недостатком: перед каждым удалением записи необходимо проверять, какие таблицы связаны с данной (с учетом возможной рекурсии ссылок: в отличие от данного примера, реальные системы имеют не две, а десятки связанных таблиц). Данный вопрос не рассматривается авторами большинства руководств по СУБД, либо упоминается, что данный механизм в БД Paradox не реализован, и удаление необходимо выполнять вручную.

Суть предлагаемого метода реализации каскадного удаления в среде Borland C++ Builder, использованной при программировании разработанной системы, заключается в создании собственного специализированного компонента на базе существующего компонента TTable.

Наследование от компонента TTable выполняется следующим образом:

```
class PACKAGE MyTable : public TTable
```

Конструктор разработанного класса имеет стандартную форму

```
__fastcall MyTable(TComponent* Owner)
```

И вызывает в процессе работы конструктор родительского класса TTable:

```
__fastcall MyTable::MyTable(TComponent* Owner): TTable(Owner)
```

Регистрация компонента, необходимая для его дальнейшего использования, выполняется следующим образом (компонент включен в группу «Samples»):

```
namespace Mytable
{
  void __fastcall PACKAGE Register()
  {
    TComponentClass classes[1] = {__classid(MyTable)};
    RegisterComponents("Samples", classes, 0);
  }
}
```

Для реализации метода каскадного удаления в классе MyTable предусмотрена специальная переменная status, характеризующая состояние таблицы в конкретный момент времени. Применительно к примеру, изображенному на рис. 1, состояние может быть закодировано следующим образом:

0 – текущая таблица закрыта;

1 – текущая таблица открыта и является таблицей «Кафедры»;

2 – текущая таблица открыта и является таблицей «Дисциплины».

Эти значения могут задаваться в программе непосредственно, либо с помощью специальных методов MyOpen() и MyClose(),автоматически устанавливающих требуемое состояние таблицы. Код метода MyOpen() может выглядеть следующим образом:

```
void __fastcall MyTable::MyOpen()
{
  if (this->TableName=="Кафедры") status=1;
  if (this->TableName=="Дисциплины") status=2;
this->Open();
}
```

Соответствующий ему код метода MyClose():

```
void __fastcall MyTable::MyClose()
{
  this->Close();
  status=0;
}
```

Для создания каскадной таблицы применяется следующий метод:

```
.....
void MyTable::CreateCTable()
{
  if (ct==NULL) {
        ct=new DTable(0);
        ct->DatabaseName=this->DatabaseName;
  }
}
```

В данной функции использована переменная **ct**, ранее определенная внутри класса как **MyTable *ct**; ее проверка необходима, чтобы избежать повторного создания уже существующего объекта. Аналогичные действия должны быть предприняты при уничтожении объекта, чтобы избежать утечек памяти:

```
__fastcall MyTable::~MyTable()
{
  if (ct!=NULL) delete ct;
}
```

Собственно метод каскадного удаления реализуется следующим образом:

```
void __fastcall MyTable::MyDelete()
{
  int a,b, i, n;
  switch (status) {
        case 0: break;
        case 1: a=this->FieldByName("Код кафедры")->AsInteger;
              CreateTable();
              ct->TableName="Дисциплины";
              ct->MyOpen();
              n=ct->RecordCount;
              ct->First();
              for(i=0;i<n;i++)
              {
                    b=ct->FieldByName("Код кафедры ")->AsInteger;
                    if (a==b) ct->MyDelete();
                    else ct->Next();
              }
```

```
                ct->Close();
                this->Delete();
                break;
            case 2: this->Delete();
                break;
        }
    }
```

При удалении записи из таблицы «Кафедры» открывается каскадная таблица «Дисциплины» и из нее удаляются все записи с требуемым кодом кафедры. Далее производится удаление записи в таблице «Кафедры».

Предложенное решение сочетает в себе достоинства как ручного, так и автоматизированного метода: позволяет самостоятельно определять действия, которые будут выполняться при удалении, модификации, добавлении записей и при этом не требует детализации последовательности действий для каждого конкретного случая, как при ручном методе. Фактически программист задает необходимые транзакции, которые позволяют работать со связанными таблицами.

Разработанный шаблон позволяет реализовать не только каскадное удаление, но и добавление, а также модификацию данных. Кроме того, он пригоден для работы с несколькими внешними ключами, что достаточно актуально в современных программных продуктах.

УДК 004.514.62

Ненадовец И.В.
Научный руководитель: к.т.н., доцент Костюк Д.А.

ИСПОЛЬЗОВАНИЕ СТРУКТУРЫ «ОБЛАКО ТЕГОВ» ДЛЯ ДОСТУПА К АРХИВАМ ДОКУМЕНТОВ

По мере роста емкости накопителей и, соответственно, архивов документов, проблема поиска информации приобретает все более острый характер, а падение эффективности иерархических файловых систем как средства упорядочения становится все более заметным. На передний план выходят такие проблемы каталогизации, как необходимость дисциплинированного подхода к именованию и размещению файлов, вдумчивому составлению названий, а также значительные трудозатраты по разбору уже существующего хаоса.

Системы локального поиска по содержимому документов получили распространение на рабочих местах пользователей начиная с 2004 года. Разработчиками таких систем были опробованы два подхода: сканирование в процессе поиска и использование заранее составленных индексных таблиц. Поиск по первому варианту занимает слишком продолжительное время (а при использовании в локальной сети приводит к абсолютно неприемлемым задержкам); поэтому он не получил значительного распространения. Поиск с предварительным индексированием, занявший бóльшую долю рынка, также нельзя назвать идеальным решением проблемы, поскольку операция индексирования – ресурсоемкая процедура, которая должна постоянно выполняться в фоновом режиме для обеспечения актуальности результатов поиска, тем самым заметно повышая загрузку системы и снижая срок службы аппаратного обеспечения.

В дополнение к сказанному поиск сам по себе – отнюдь не идеальное решение. Применение вместо поиска информации визуального ориентирования в ее потоке, основанного на системе информационных тегов документов, позволяет увеличить наглядность и интуитивность интерфейса, упростить ориентирование человека в информационном пространстве.