**ВВЕДЕНИЕ**

Высшее учебное заведение сегодня – это сложный учебно-хозяйственный комплекс с многочисленными внешними и внутренними связями. И управление Вузом, его информационными потоками, учебным процессом, документооборотом и прочими процессами представляет собой сложную систему, мелкие и крупные задачи которой тесно связаны между собой.

Система управления ВУЗом должна представлять собой гибкую, динамичную систему, позволяющую эффективно рассматривать намеченные планы, оперативно учитывать все изменения и новые тенденции. Большие возможности для совершенствования управления каждым ВУЗом предоставляет использование вычислительной техники и средств связи. В настоящее время накоплен определённый опыт разработки и внедрения автоматизированных систем управления ВУЗом.

В организационной системе наиболее трудоемкими являются процессы, связанные с обработкой информации – сбор, накопление, преобразование, отображение, хранение, передача и вывод. Ускорить эти процессы и облегчить труд персонала Вуза позволяет АСУ.

Практически все современные виды деятельности человека немыслимы без использования современных информационных средств, компьютерных технологий. В Таразском Государственном Университете на кафедре «Компьютерные системы» готовят специалистов по самым современным, перспективным и престижным специальностям, таким как вычислительная техника и программное обеспечение, информационные системы, компьютерные системы управления. Деятельность кафедры связана с учетом большого количества информации, которую необходимо систематизировать.

Целью данного проекта является создание ИС для автоматизации учета и анализа информации (данных) об успеваемости студентов кафедры «Компьютерные системы». Необходимо не только упорядочить информацию, но и упростить процессы ее анализа и принятия необходимых управленческих решений.

Задачами данного проекта являются:

* выявить необходимость автоматизации информационных потоков процессов деятельности кафедры;
* проектирование базы данных в ERWin и генерация ее в Interbase;
* создание клиент-приложения в объектно-ориентированной среде программирования Delphi7.

Целью всех выполняемых действий являются:

* автоматизировать процесс регистрации, учета информации;
* сократить время поиска необходимых данных;
* оптимизировать хранение информации, необходимой для деятельности кафедры.

**1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

**1.1. Постановка задачи**

В настоящий момент на кафедре «Компьютерные системы» студенты обучаются по направлениям 050602 «Информатика», 050703 «Информационные системы», 050704 «Вычислительная техника и программное обеспечение».

В Таразском Государственном Университете с 2004 года обучение производится по кредитной технологии. Занятия организуются и проводятся в соответствии с учебными планами направлений и специальностей и рабочими программами дисциплин, входящих в учебные планы. Разработка учебных планов, составление разнообразных отчетов и т.п. – вся эта работа проводится вручную, занимая значительную часть времени работы заведующего кафедрой. В течение семестра студенты проходят две промежуточные аттестации баллы, за которые вносятся в первую и вторую промежуточную ведомость. Выставление баллов происходит по определенным критериям: лекционное занятие, практическое занятие и СРС, (которое включает в себя баллы за курсовую работу, если такая имеется). В конце семестра (по истечении 15 недель) студенты сдают экзамены по данным дисциплинам, полученная оценка вносится в экзаменационную ведомость. Затем на основе трех ведомостей: двух промежуточных (накопительных) и экзаменационной формируется итоговая ведомость, которая включает в себя итоговую оценку по дисциплине в цифровом и буквенном выражении, которая затем идет в транскрипт студента. В итоге полученные данные анализируются: производится анализ баллов полученных студентами по итогам семестра, сдача курсовых работ, а также баллов по ним; просматриваются и редактируются при необходимости учебные планы. По данной информации формируются отчеты для заведующего кафедрой, а также в деканат. Деканат, получает информацию для контроля организации учебного процесса по специальностям и кафедрам факультета, набор документов по текущей успеваемости студентов факультета и рейтинговые оценки за экзаменационные сессии. Каждый заведующий кафедрой получает актуальный рабочий учебный план специальности и список дисциплин кафедры, промежуточные и экзаменационные ведомости по дисциплинам кафедры для внесения рейтинговых баллов, имеет доступ к справочнику и пополняет фонд кафедры рабочих учебных программ.

Разработка и корректировка учебных планов специальностей ведется сотрудниками кафедр. На этапе получения стандартов специальностей решаются следующие задачи:

* ввод данных об обязательных циклах дисциплин;
* корректировка контрольных показателей.

На этапе разработки учебных планов:

* ввод данных о дисциплинах, назначенных кафедрой, дисциплинах специализации, дисциплинах по выбору студента;
* ввод данных о распределении занятий и курсовых работ по семестрам.

Обработать вручную такой объем информации довольно таки сложно. Именно поэтому было принято решение создать автоматизированную систему для учета и анализа успеваемости студентов, а также контроля учебных планов.

С точки зрения реализации проектируемая система должна удовлетворять следующим требованиям:

* системность и информационная совместимость подсистем и элементов подсистемы, т.е. создание во всей информационной системе взаимоувязанной совокупности форм обмена информацией;
* методическое единство, т.е. разработка различных подсистем на основе единых принципов, и обеспечение взаимосвязи различных подсистем, входящих в состав системы (показатели, формы документов);

Следовательно, результатом данной работы должен стать проект, удовлетворяющий требованиям заведующего кафедрой «Компьютерные системы» по учету хранимой информации об успеваемости студентов кафедры.

**1.2. Входные/выходные данные**

Исходными данными являются:

* данные о студенте (ФИО, группа);
* данные о преподавателе (ФИО, должность);
* данные о группах (наименование, староста, эдвайзер, год поступления);
* сведения о специальностях (наименование, год введения);
* данные о дисциплинах (наименования, кредиты, наличие курсовой работы).

На основе исходных данных формируется:

* рабочий учебный план, который носит справочный характер;
* ведомость успеваемости студентов.

Выходные документы/данные в деятельности кафедры:

* отчет «Рабочий учебный план»;
* отчет Дисциплин, по которым есть курсовые работы;
* отчет по дисциплинам с учетом итогового балла и балла полученного по курсовой работе, если такая имеется;
* отчет по итоговым баллам студентов по дисциплинам.

# 

# 2. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

# 2. 1. Анализ известных подходов к решению проблемы

На сегодняшний день существуют два основных подхода к разработке ПО ИС, один из которых называется *функционально-модульный* или *структурный*. В его основу положен принцип функциональной декомпозиции, при которой структура системы описывается в терминах иерархии ее функций и передачи информации между отдельными функциональными элементами. Второй, *объектно-ориентированный* использует объектную декомпозицию. Структура системы описывается в терминах объектов и связей между ними, а поведение системы описывается в терминах обмена сообщениями между объектами. Каждый объект системы обладает своим собственным поведением, моделирующим поведение объекта реального мира.

При разработке ПО ИС с использованием структурного подхода система разбивается на функциональные подсистемы, которые в свою очередь, делятся на подфункции, те – на задачи и так далее. Процесс разбиения продолжается вплоть до конкретных процедур. При этом автоматизируемая система сохраняет целостное представление, в котором все составляющие компоненты взаимосвязаны. При разработке системы "снизу-вверх", от отдельных задач ко всей системе, целостность теряется, возникают проблемы при описании информационного взаимодействия отдельных компонентов.

Важным качеством объектно-ориентированного подхода является согласованность моделей деятельности организации и моделей проектируемой системы от стадии формирования требований до стадии реализации. Объектно-ориентированный подход обладает рядом преимуществ, таких как:

* объектная декомпозиция дает возможность создавать ПС меньшего размера путем использования общих механизмов. Использование объектно-ориентированного подхода существенно повышает уровень унификации разработки и пригодность для повторного использования программ, а также проектов;
* объектная декомпозиция уменьшает риск создания сложных систем ПО, т.к. она предполагает эволюционный путь развития системы на базе относительно небольших подсистем;
* объектная модель позволяет в полной мере использовать выразительные возможности объектных и объектно-ориентированных языков программирования.

Главным же достоинством объектно-ориентированного подхода является то, что объектно-ориентированные системы более открыты и легче поддаются внесению изменений, т.к. их конструкция базируется на устойчивых формах. Переход на новую технологию связан с высокими начальными затратами, а также с преодолением психологических трудностей. Это относится к недостаткам объектно-ориентированного подхода. Но все-таки объектно-ориентированная модель наиболее адекватно отражает реальный мир, представляющий собой совокупность взаимодействующих объектов. Объектно-ориентированный подход не дает немедленной отдачи. Эффект от его применения начинает сказываться после разработки двух-трех проектов [6].

В настоящее время на практике количество CASE-средств, поддерживающих объектно-ориентированный подход, невелико по сравнению с поддерживающими структурный подход. В том числе, диаграммы, отражающие специфику объектно-ориентированного подхода (диаграммы классов, объектов и т.д.), гораздо менее наглядны и плохо понимаемы непрофессионалами. Примерами объектно-ориентированных CASE-средств являются: Rational Rose фирмы Rational Software и Paradigm Plus фирмы Computer Associates, предназначенные для автоматизации этапов анализа и проектирования ПО.

Современные CASE-средства, поддерживающих структурный подход, охватывают обширную область поддержки многочисленных технологий проектирования ИС: от простых средств анализа и документирования до полномасштабных средств автоматизации, покрывающих весь жизненный цикл ПО. На сегодняшний день российский рынок программного обеспечения располагает следующими наиболее развитыми CASE-средствами: Vantage Team Builder (Westmount I-CASE); Designer/2000; Silverrun; ERwin+BPwin; S-Designor; CASE.Аналитик.

ERwin — современное средство проектирования баз данных.Отличительной чертой ERwin/ERX является высокая степень обеспечения согласованного взаимодействия между средствами создания баз данных и средствами разработки приложений. ERwin поддерживает все наиболее популярные реляционные СУБД, включая Oracle; Sybase; Informix; Microsoft SQL Server; FoxPro; InterBase; Paradox; Access и др. Одна и та же модель может быть использована для создания нескольких баз данных или для переноса приложения с платформы одной СУБД на другую. ERwin можно использовать совместно со многими популярными средствами разработки приложений: Delphi, PowerBuilder, Visual Basic, Oracle Designer/2000 и др. [6]

Основные характеристики:

* поддержка стандартной нотации IDEF1X для ER диаграмм моделей данных, нотации IE и специальной нотации, предназначенной для проектирования хранилищ данных – DIMENSIONAL;
* возможность импорта/экспорта данных из BРwin, Oracle Designer;
* поддержка триггеров, хранимых процедур и шаблонов;
* развитые средства проверки корректности моделей данных;
* автоматическая генерация SQL DDL для создания баз данных;
* глубокая интеграция с продуктами Oracle, Sybase, Centura, Microsoft на базе единого репозитория и эффективного обмена проектами; импорт/экспорт с Rational Rose;
* автоматическая генерация экранных форм приложений для PowerBuilder, Delphi, Visual Basic, созданных на основе спроектированной модели данных.

2.2. Проектирование базы данных в ERwin

Для создания ER – диаграммы в проекте используется программное средство ERwin.

В базе данных использованы домены**.** Домен - это потенциальное множество значений простого типа данных, он имеет сходство с подтипом данных в некоторых языках программирования. Домен определяется двумя элементами - типом данных и логическим выражением, которое применяется к данным. Если результат этого выражения равен значению «истина», то экземпляр данных принадлежит домену [6].

Таблица 1- Домены, используемые в диаграмме.

|  |  |  |
| --- | --- | --- |
| Логическое имя домена | Физическое имя домена | Родительский домен |
| *examen* | *t\_examen* | Number |
| *itog* | *t\_itog* | Number |
| *kol\_vo\_kreditov* | *t\_ko\_lvo\_kreditov* | Number |
| *lc1* | *t\_lc1* | Number |
| *lc2* | *t\_lc2* | Number |
| *pr1* | *t\_pr1* | Number |
| *pr2* | *t\_pr2* | Number |
| *srs1* | *t\_srs1* | Number |
| *srs2* | *t\_srs2* | Number |
| *Disciplina\_s* | *t\_Disciplina\_s* | String |
| *Disciplina\_p* | *t\_Disciplina\_p* | String |
| *Specialnost\_s* | *t\_Specialnost\_s* | String |
| *Specialnost\_p* | *t\_Specialnost\_p* | String |
| *Doljnost* | *t\_Doljnost* | String |
| *FIO* | *t\_FIO* | String |
| *gruppa* | *t\_gruppa* | String |
| *Semester* | *T\_semestr* | number |

В диаграмму входят такие сущности:

Сущность *Student* включает:

* *FIO;*
* *Gruppa.*

Сущность *Gruppa* включает:

* *Gruppa;*
* *God postuplenia;*
* *Edvaizer.*

Сущность *Specialnost* включает:

* *Shifr\_specialnosti;*
* *Specialnost\_p;*
* *Specialnost\_s;*
* *God\_vvedenia.*

Сущность *Disciplina* включает:

* *Kod\_disciplina;*
* *Disciplina\_p;*
* *Disciplina\_s.*

Сущность *Prepodavatel* включает:

* *Kod\_disciplina;*
* *Shifr\_specialnost;*
* *Semester;*
* *Kol\_vo\_kreditov.*

*СущностьVedomost* включает:

* *Kod\_prepod;*
* *Kod\_disciplina;*
* *Shifr\_specialnost;*
* *Kod\_student;*
* *Lc1;*
* *Lc2;*
* *Pr1;*
* *Pr2;*
* *Srs1;*
* *Srs2;*
* *Examen;*
* *Itog.*

На физическом уровне модель, напротив, зависит от конкретной базы данных, выбираемой пользователем. Таким образом, одной логической модели может соответствовать несколько физических моделей. Кроме того, так как в физической модели речь идет уже о реально существующих в БД физических объектах, то обязательно должны быть определены типы данных и атрибутов.

На **логическом уровне** данные представляются так, как они выглядят в реальном мире. Объектами логического уровня являются сущности и атрибуты. Модель логического уровня является универсальной и не связана с конкретнойбазой данных [6.

Затем генерируем SQL код первоначально установив программу выбрав Server/Target Interbase, затем выбрав Server Tasks/Schema Generation, установив необходимые параметры нажимаем кнопку Preview для просмотра сгенерированного SQL кода. Итак, получив SQL код, мы можем перейти к следующему этапу проектирования.

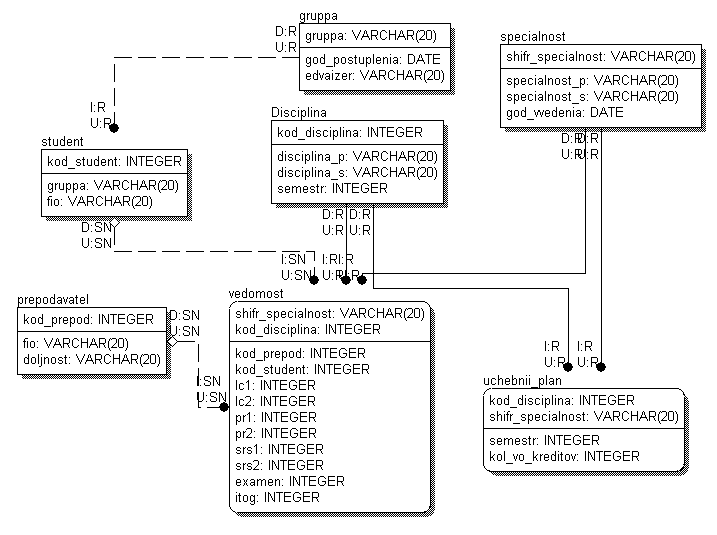


Рисунок 1- Диаграмма на физическом уровне

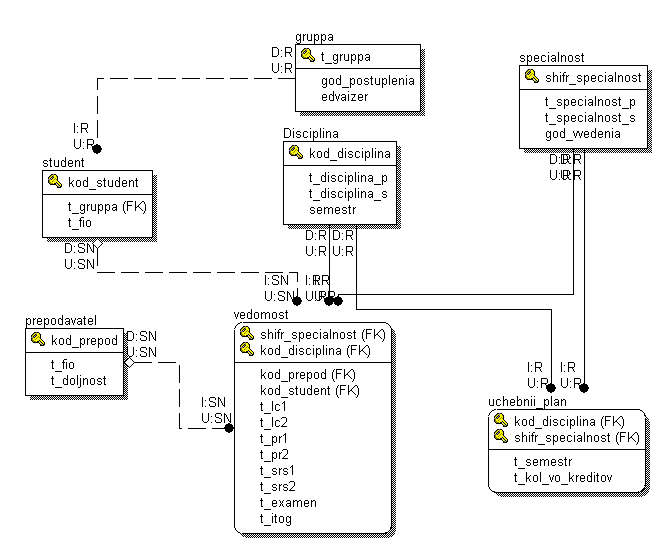


Рисунок 2 - Диаграмма на логическом уровне.

2.3. Создание базы данных в INTERBASE

**2.3.1. INTERBASE и область его применения**

InterBase представляет собой полнофункциональный SQL- сервер. Сервер баз данных- это программный процесс, который выполняется на узле сети, где расположен главный компьютер или физически расположена сама база данных. Процесс сервера- единственный процесс на любом узле, который может исполнять прямые операции ввода-вывода для файлов базы данных.

Клиенты посылают запросы серверному процессу, чтобы выполнить различные действия в базе данных, включая:

* Поиск в базе данных по заданным условиям;
* Сравнение, сортировку и представление данных в табличном виде;
* Изменение хранимых данных;
* Добавление новых данных в базу;
* Удаление данных из базы данных;
* Создание новых базы данных и структур данных;
* Выполнение программного кода на сервере;
* Передачу сообщения другим клиентам, подключенным в данный момент к серверу.

Серверный процесс является полностью сетевым, он поддерживает запросы на подключение от других узлов сети и тот же самый протокол InterBase прикладной программы, что и клиентские процессы.

Несколько клиентов могут быть связаны с многопоточным процессом сервера одновременно.

Сервер регулирует доступ к отдельным записям данных в пределах базы данных и обеспечивают монопольный доступ к записям, когда клиенты выдают запросы на изменение данных в записях.

Отличительными качествами InterBase являются:

* Высокая производительность и надежность сервера при минимальных требованиях к техническим средствам.
* Поддержка стандарта SQL-92, обеспечивающая переносимость приложений.
* Относительно низкая стоимость продукта.
* Простота установки и поддержки сервера. Удобный и не требующий специальной подготовки механизм администрирования базой данных.

Все это делает InterBase прекрасным выбором для реализации корпоративных систем малого и среднего масштаба (с количеством пользователей в несколько десятков). При реализации очень крупных проектов (с сотнями или более пользователей) стоит, наверное, рассмотреть более мощные серверы – типа Oracle или Informix [1].

**2.3.2. Уровни реализации языка SQL**

По своему существу SQL предназначен только для описания взаимодействия с базой данных. SQL не содержит средств, необходимых для разработки законченных программ и может использоваться только как их часть. В зависимости от назначения SQL может использовать один из трех уровней реализации.

1. Интерактивный или автономный SQL(ISQL) дает возможность пользователям непосредственно извлекать информацию из базы данных или записывать ее в базу данных. Этот уровень доступен посредством соответствующих утилит или специальных стандартных средств для работы с базами данных. В InterBase это реализуется такими утилитами, как Isql.exe и Wisql32.exe. С их помощью информация может быть загружена в базу данных или извлечена из нее и выведена на экран, печать или файл.
2. Статический SQL (SQL)- фиксированный (исполнимый), т.е. заранее подготовленный и записанный , а не формируемый в процессе выполнения программы SQL код, обычно используемый в приложениях. Существует две версии статического SQL. Встроенный или внедренный SQL – это код SQL, включенный в исходный код программы, написанной на том или ином алгоритмическом языке. Используемый язык при этом называют базовым. InterBase поддерживает внедренный SQL. Другое использование статического SQL- модульный язык. В этом случае модули SQL присоединяются к приложению на этапе компоновки.
3. Динамический SQL (DSQL) – код SQL, генерируемый приложением во время его выполнения. Динамический SQL заменяет статический в тех случаях, когда необходимый код не может быть определен во время написания приложения, так как он сам зависит от действий пользователя во время выполнения приложения.

В InterBase используются все 3 уровня реализации SQL. Необходимо помнить, что на разных уровнях доступны не все команды языка. Кроме того, синтаксис команд разных уровней может отличаться. При описании синтаксиса соответствующих команд оговаривается уровень реализации, на который ориентирован это синтаксис. В большинстве случаев, правда этот синтаксис является единым [8].

**2.3.3. Скорость доступа к данным**

Скорость доступа к данным определяется объемом просматриваемой в процессе выборки информации. При отсутствии условий для выборки (фильтрации) и упорядочения просмотр осуществляется всегда в физическом порядке, обеспечивая максимальную скорость поиска. Когда задаются некоторые условия, то скорость поиска может быть существенно увеличена, если данные, участвующие в условиях выборки тем или иным способом упорядочены. В этом случае оказывается возможным сократить объем просматриваемых данных, сразу исключив те, которые не удовлетворяют условиям выборки.

В InterBase упорядочение данных достигается заданием индексов по интересующему полю или группе полей. Индекс представляет собой список значений индексных полей, в котором каждому набору значений сопоставляется указатель на соответствующие строки таблиц. Сами индексы организуются, как правило, в виде бинарных деревьев или списков, что позволяет вести достаточно быстрый поиск по ним. При этом объем памяти, занимаемый индексом, относительно невелик.

Следует помнить, что при всяком обновлении данных должны обновляться и индексы. Таким образом, платой за быстрый поиск является увеличение затрат времени на обновление данных. Кроме того, сами индексы после большого числа обновлений становятся несбалансированными, вследствие чего время поиска по ним возрастает. Их можно перестроить, но это также требует затрат времени.

В этих условиях при проектировании базы данных необходимо находить разумный компромисс между требованиями по ускорению поиска данных и требования по скорости их обновления. В частности, использование индексов для небольших по объему таблиц вообще не оправдано. Если имеется индекс по группе полей, то поиск по первому из полей групп может прямо использовать этот индекс, следовательно, нет смысла по нему делать отдельный индекс. Если поиск по каким-либо полям редок, то построение по ним индекса тоже не эффективно. К сожалению, более конкретные рекомендации по определению состава индексов едва ли возможны. Оптимальный выбор зависит и от структуры базы и от характера ее использования [8].

2.4. Этапы создания базы данных в INTERBASE

Первым этапом является создание базы данных:

Базы данных InterBase-сервера хранятся в виде файлов, имеющих расширение .*gdb*. для создания новой БД в окне программы *IBConsole* выберем команду *Database* *→ Create Database.*

В появившемся диалоговом окне создания базы данных необходимо ввести следующие данные:

* Введите в поле *Alias* название базы данных: date\_as;
* В первой строке таблицы *Files* укажите полное имя базы данных: C:\DATE\date.gdb и ее размер в страницах: 230;

Далее укажем в таблице *Options*  дополнительные сведения о базе данных:

* *Page size* (размер страницы) - 4096. Это значение проставляется по умолчанию.
* *Default character set* (набор символов по умолчанию) выберем из списка WIN1251. Это необходимо для того, чтобы в нашей базе данных можно было сохранять строковые данные, содержащие символы кириллицы.
* *SQL Dialect* - вводим 3 [5].

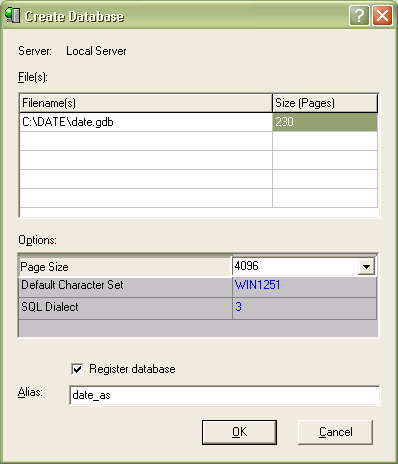


Рисунок 3 - Диалоговое окно создания базы данных

После всего нажимаем **ОК**.

Программа IBConsole создает только файл базы данных, а каталог, в котором создается база данных, должен уже существовать на жестком диске. Например, в нашем случае, на диске **С:\** должна существовать папка **DATE,** в противном случае база данных создана не будет, а на экране появится сообщение об ошибке.

Второй этап- создание объектов базы данных. Подключившись к базе данных, дважды щелкнув по ее значку, увидим все компоненты входящие в базу данных. Для работы с базами данных в составе IBConsole имеется дополнительный инструмент - программа для интерактивной работы с SQL-запросами «Interactive SQL». Для ее запуска следует выбрать в меню пункт Tools→Interactive SQL. Эта программа позволяет набирать SQL-запросы и выполнить, сразу же видя результат. Поэтому окно программы делится на две области: верхнюю - область SQL-запросов и нижнюю - область результатов.

С помощью уже имеющегося кода мы создадим в нашей базе данных таблицы, индексы, триггеры ссылочной целостности.

**Создание доменов**

Столбцы в различных таблицах базы данных могут содержать однотипные данные. Кроме того, виды контроля этих данных также могут быть одинаковыми. В этих условиях описания данных и методов их контроля целесообразно выполнить один раз, тогда при описании таблиц достаточно указывать только имя соответствующего описания данных. Для этих целей и служат описания доменов. Перед тем как создавать столбцы, которые ссылаются на домены, необходимо задать описания доменов. Для этих целей существует команда CREAT DOMAIN. В результате ее выполнения создается шаблон, на который можно ссылаться в командах создания и модификации таблиц.

Наиболее полезно использование доменов, когда большое число таблиц содержит идентичные типы данных.

Столбцы, базирующиеся на доменах, наследуют характеристики домена, причем часть из них может быть предопределена в локальных описаниях столбцов. Прежде чем создавать домены, необходимо выполнить проектирование базы данных, включающее проектирование системы таблиц и их нормализацию [3].

SQL коды для создания доменов:

CREATE DOMAIN t\_disciplina\_p VARCHAR(80);

CREATE DOMAIN t\_disciplina\_s VARCHAR(50);

CREATE DOMAIN t\_doljnost VARCHAR(50);

CREATE DOMAIN t\_examen INTEGER;

CREATE DOMAIN t\_fio VARCHAR(50);

CREATE DOMAIN t\_gruppa VARCHAR(20);

CREATE DOMAIN t\_itog INTEGER;

CREATE DOMAIN t\_kol\_vo\_kreditov INTEGER;

CREATE DOMAIN t\_lc1 INTEGER;

CREATE DOMAIN t\_lc2 INTEGER;

CREATE DOMAIN t\_pr1 INTEGER;

CREATE DOMAIN t\_pr2 INTEGER;

CREATE DOMAIN t\_semestr INTEGER;

CREATE DOMAIN t\_specialnost\_p VARCHAR(80);

CREATE DOMAIN t\_specialnost\_s VARCHAR(20);

CREATE DOMAIN t\_srs1 INTEGER;

CREATE DOMAIN t\_srs2 INTEGER;

Выполнив SQL коды получим:

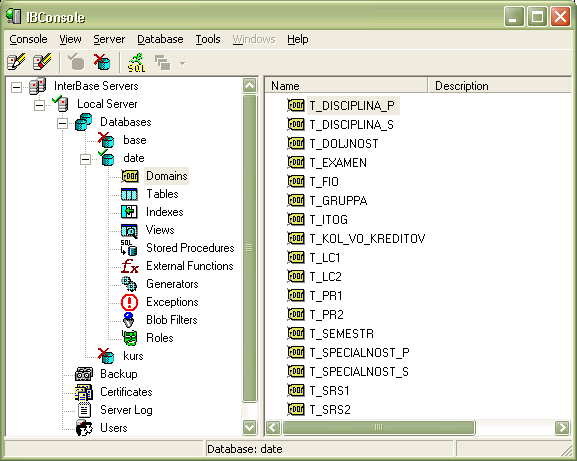


Рисунок 4 - Окно IBConsole после создания доменов

**Создание таблиц и индексов**

Перед тем как перейти к созданию таблиц, необходимо выполнить проектирование базы данных и нормализацию таблиц.

Далее следует определить, какие таблицы и с какими столбцами (полями) подлежат созданию. Создать комплект необходимых доменов и только после этого можно перейти к физическому вводу описаний таблиц.

При создании таблицы мы должны задать, как минимум, ее имя и перечень полей с их атрибутами и контрольными ограничениями. Кроме того, при создании таблицы можно задать ее первичный ключ, внешние ключи, задающие требования по поддержанию логической целостности, дополнительные виды контроля на уровне записей [4].

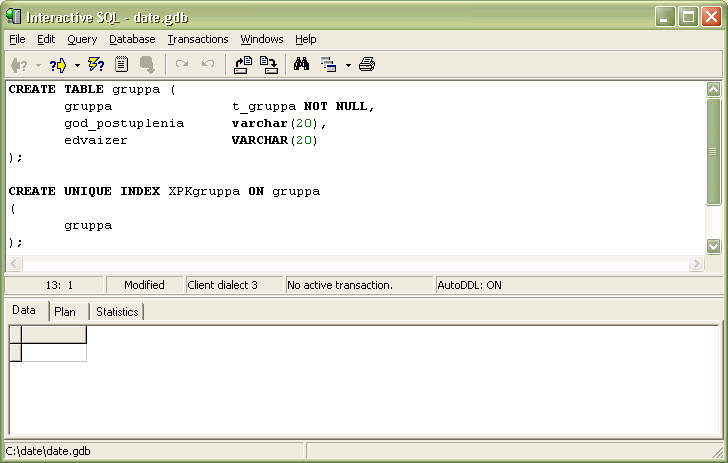


Рисунок 5 - Окно Interactive SQL при создании таблицы группа и уникального индекса для данной таблицы.

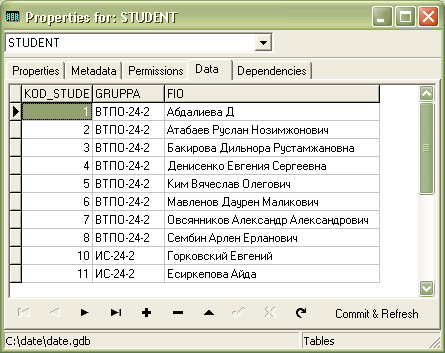


Рисунок 6 - Таблица *Студент*

Выполнив данный код нажав комбинацию клавиш Ctrl+Enter мы получим таблицу, в которую затем мы можем внести записи. Однако стоит заметить т.к. между таблицей Student и Gruppa существует связь по полю Gruppa, то перед заполнением таблицы Student, необходимо заполнить соответствующие значения в таблицу Gruppa, иначе на экране появится сообщение об ошибке.

SQL коды для создания таблиц и индексов:

*CREATE TABLE gruppa (*

*gruppa t\_gruppa NOT NULL,*

*god\_postuplenia varchar(20),*

*edvaizer VARCHAR(20)*

*);*

*CREATE UNIQUE INDEX XPKgruppa ON gruppa*

*(*

*gruppa*

*);*

*ALTER TABLE gruppa*

*ADD PRIMARY KEY (gruppa);*

*CREATE TABLE student (*

*kod\_student INTEGER NOT NULL,*

*gruppa t\_gruppa NOT NULL,*

*fio t\_fio*

*);*

*CREATE UNIQUE INDEX XPKstudent ON student*

*(*

*kod\_student*

*);*

*ALTER TABLE student*

*ADD PRIMARY KEY (kod\_student);*

*CREATE TABLE Disciplina (*

*kod\_disciplina INTEGER NOT NULL,*

*disciplina\_p t\_disciplina\_p,*

*disciplina\_s t\_disciplina\_s,*

*semestr INTEGER*

*);*

*CREATE UNIQUE INDEX XPKDisciplina ON Disciplina*

*(*

*kod\_disciplina*

*);*

*ALTER TABLE Disciplina*

*ADD PRIMARY KEY (kod\_disciplina);*

*CREATE TABLE specialnost (*

*shifr\_specialnost VARCHAR(20) NOT NULL,*

*specialnost\_p t\_specialnost\_p,*

*specialnost\_s t\_specialnost\_s,*

*god\_vvedenia varchar(20)*

*);*

*CREATE UNIQUE INDEX XPKspecialnost ON specialnost*

*(*

*shifr\_specialnost*

*);*

*ALTER TABLE specialnost*

*ADD PRIMARY KEY (shifr\_specialnost);*

*CREATE TABLE prepodavatel (*

*kod\_prepod INTEGER NOT NULL,*

*fio t\_fio NOT NULL,*

*doljnost t\_doljnost NOT NULL*

*);*

*CREATE UNIQUE INDEX XPKprepodavatel ON prepodavatel*

*(*

*kod\_prepod*

*);*

*ALTER TABLE prepodavatel*

*ADD PRIMARY KEY (kod\_prepod);*

*CREATE TABLE vedomost (*

*shifr\_specialnost VARCHAR(20) NOT NULL,*

*kod\_prepod INTEGER,*

*kod\_disciplina INTEGER NOT NULL,*

*kod\_student INTEGER,*

*lc1 t\_lc1,*

*lc2 t\_lc2,*

*pr1 t\_pr1,*

*pr2 t\_pr2,*

*srs1 t\_srs1,*

*srs2 t\_srs2,*

*examen t\_examen,*

*itog t\_itog*

*);*

*CREATE UNIQUE INDEX XPKvedomost ON vedomost*

*(*

*shifr\_specialnost,*

*kod\_disciplina*

*);*

*ALTER TABLE vedomost*

*ADD PRIMARY KEY (shifr\_specialnost, kod\_disciplina);*

*CREATE TABLE uchebnii\_plan (*

*kod\_disciplina INTEGER NOT NULL,*

*shifr\_specialnost VARCHAR(20) NOT NULL,*

*semestr t\_semestr,*

*kol\_vo\_kreditov t\_kol\_vo\_kreditov*

*);*

*CREATE UNIQUE INDEX XPKuchebnii\_plan ON uchebnii\_plan*

*(*

*kod\_disciplina,*

*shifr\_specialnost*

*);*

*ALTER TABLE uchebnii\_plan*

*ADD PRIMARY KEY (kod\_disciplina, shifr\_specialnost);*

*ALTER TABLE student*

*ADD FOREIGN KEY (gruppa)*

*REFERENCES gruppa;*

*ALTER TABLE vedomost*

*ADD FOREIGN KEY (kod\_prepod)*

*REFERENCES prepodavatel;*

*ALTER TABLE vedomost*

*ADD FOREIGN KEY (kod\_student)*

*REFERENCES student;*

*ALTER TABLE vedomost*

*ADD FOREIGN KEY (shifr\_specialnost)*

*REFERENCES specialnost;*

*ALTER TABLE vedomost*

*ADD FOREIGN KEY (kod\_disciplina)*

*REFERENCES Disciplina;*

*ALTER TABLE uchebnii\_plan*

*ADD FOREIGN KEY (shifr\_specialnost)*

*REFERENCES specialnost;*

*ALTER TABLE uchebnii\_plan*

*ADD FOREIGN KEY (kod\_disciplina)*

*REFERENCES Disciplina;*

Третий этап. Создав все объекты мы сможем увидеть их на дереве нашей базы (рис.7).

Затем необходимо автоматизировать уникальные значения таблиц, т.к. при работе с глобальной базой данных указать уникальное значение не всегда возможно. Так как вставкой записей могут заниматься одновременно несколько пользователей. Угадать последнее вставленное значение уникального значения практически невозможно, так что необходимо как-то автоматизировать этот процесс. С этой целью в базе данных предусмотрены специальные объекты – генераторы.

*Генератор* - это именованная числовая переменная, значение которой можно увеличивать с помощью специальной функции gen\_id. Для того, чтобы можно было пользоваться генератором, его необходимо создать и инициализировать. Для создания генератора используется оператор (запрос).

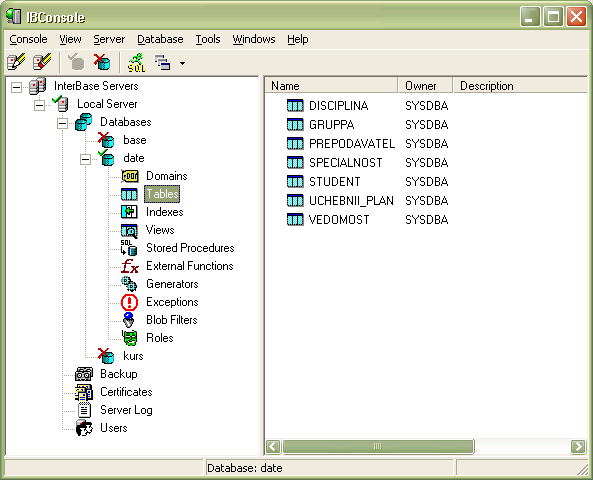


Рисунок 7 - Окно IBConsole после создания таблиц

Рассмотрим установку генератора для таблицы Student(Студент), где уникальным является kod\_student (номер студента). Создадим генератор, возвращающий уникальное значение для поля kod\_student в таблице STUDENT и установим стартовое значение равное 1. Для этого введите в окно запросов ISQL следующее:

CREATE GENERATOR kod\_student\_gen

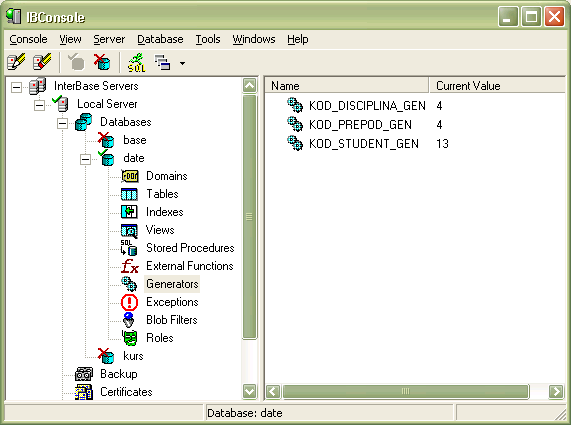


Рисунок 8 - Окно IBConsole после создания генераторов.

Теперь присвоим нашему генератору начальное значение. Так как в таблице STUDENT уже есть записи, то нужно инициализировать генератор наибольшим имеющимся в базе значением поля kod\_student. Посмотреть его можно, набрав в окне ISQL запрос

select MAX(kod\_student) from STUDENT

В окне результатов появится примерно следующее:

MAX

===

4

Значение поля равно 4 так как мы вручную внесли уже четыре записи. По умолчанию у созданного генератора значение равно 0. Наберите и выполните запрос:

SET GENERATOR kod\_student\_gen TO 4

Теперь текущее значение генератора равно 4. Функция gen\_id() увеличивает значение генератора на заданное число и возвращает полученное значение.

С помощью генератора kod\_student\_gen автоматизировано получение уникального числового значения для ключевого поля kod\_student, однако при каждой вставке новой записи приходится вызывать функцию gen\_id(kod\_student\_gen,1). Сервер позволяет автоматизировать и эту задачу c помощью механизма, который называется *триггером*.

Триггер - это некоторая процедура, обрабатываемая автоматически SQL-сервером при возникновении определенного события в базе данных *- вставки, удаления и изменения записи.* Непосредственно из программы к триггерам обратиться нельзя. Нельзя и передавать им входные параметры и получать от них значения выходных параметров. Триггеры всегда реализуют *действие*. Триггер связан с определенной таблицей базы и никогда не вызывается пользователем напрямую. Вместо этого он срабатывает (еще говорят - «стреляет») при попытке выполнить действие, заданное в его определении.

По событию изменения таблицы БД триггеры различаются на вызываемые при:

* добавлении новой записи
* изменении существующей записи
* удалении записи

По отношению к событию, влекущему их вызов, триггеры различаются на:

* Выполняемые до наступления события
* Выполняемые после наступления события
* Автоматическое обеспечение каскадных воздействий в дочерних таблицах при изменении, удалении записи в родительской таблице выполняется на сервере. Пользователю нет необходимости заботиться о программной реализации каскадных воздействий, поскольку каскадные воздействия выполняет сервер, что снижает загрузку сетевого трафика.
* Изменения в триггерах не влекут необходимости изменения программного кода в клиентских приложениях и не требует распространения новых версий клиентских приложений у пользователей [8].

При операциях вставки, удаления и изменения записи в теле триггера можно обращаться к значениям полей текущей записи при помощи специальных переменных NEW и OLD, например, узнать значение поля first\_name при вставке записи можно, воспользовавшись выражением NEW.kod\_student. Создадим для таблицы STUDENT триггер, который будет подставлять в поле kod\_student очередное уникальное значение. Введите в окне ISQL и выполните следующий запрос:

*SET TERM !! ;*

*CREATE TRIGGER kod\_student\_gen FOR STUDENT*

*BEFORE INSERT AS*

*BEGIN*

*new.kod\_student=gen\_id(kod\_student\_gen,1);*

*END !!*

*SET TERM ; !!*

Этот триггер будет срабатывать перед вставкой записи в таблицу STUDENT, присваивая полю kod\_student уникальное значение, полученное от генератора   
kod\_student\_gen:

*new.kod\_student=gen\_id(kod\_student\_gen,1);*

Затем получим еще два генератора и два триггера для таблиц PREPODAVATEL и DISCIPLINA, чтобы автоматизировать вставку новых записей. Ниже представлены SQL коды для реализации этих задач:

Для таблицы PREPODAVATEL

*CREATE GENERATOR kod\_PREPOD\_gen*

*select MAX(kod\_PREPOD) from prepodavatel*

*SET GENERATOR kod\_prepod\_gen TO 1*

*SET TERM !! ;*

*CREATE TRIGGER kod\_prepod\_gen FOR PREPODAVATEL*

*BEFORE INSERT AS*

*BEGIN*

*new.kod\_prepod=gen\_id(kod\_prepod\_gen,1);*

*END !!*

*SET TERM ; !!*

Для таблицы DISCIPLINA

*CREATE GENERATOR kod\_disciplina\_gen*

*select MAX(kod\_disciplina) from disciplina*

*SET GENERATOR kod\_disciplina\_gen TO 1*

*SET TERM !! ;*

*CREATE TRIGGER kod\_disciplina\_gen FOR DISCIPLINA*

*BEFORE INSERT AS*

*BEGIN*

*new.kod\_disciplina=gen\_id(kod\_disciplina\_gen,1);*

*END !!*

*SET TERM ; !!*

**3.** **РЕАЛИЗАЦИЯ БД В СРЕДЕ ПРОГРАММИРОВАНИЯ DELPHI**

**3.1. Формирование справочной системы**

База данных включает в себя 7 таблиц: *Student, Prepodavatel, Gruppa, Disciplina, Specialnost, Uchebnii\_plan* и *Vedomost.* Причем *Student, Prepodavatel, Gruppa, Disciplina, Specialnost* являются справочными для таблиц *Uchebnii\_plan* и *Vedomost*. Поэтому первоначально рассмотрим автоматизацию работы данных таблиц. Данные таблицы содержатся на одной форме на компоненте PageControl1, на которой мы создаем 5 страниц, выбрав в контекстном меню PageControl1/New page. В итоге мы получаем TabSheet1.Caption=Список студентов, TabSheet2. Caption =Список групп, TabSheet3. Caption =Список дисциплин, TabSheet4. Caption =Список специальностей, TabSheet5. Caption =Список преподавателей. Для соединения с базой данных поставим на форму DataBase1, установив его свойства AliasName=date\_as, DatabaseName=date, Connected=true.

**3.1.1. Формирование системы Gruppa**

Перейдя на страницу Список групп расположим на ней Table1, Table2, 2 DataSource, 2 DbGrid, 3 кнопки Button, 4 текстовых поля Edit, 4 Label и 1 GroupBox. Затем установим для них свойства (см. рисунок 9):

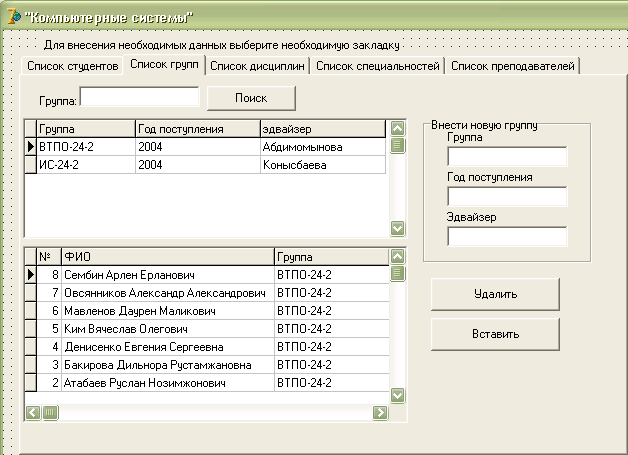


Рисунок 9 - Страница «Список групп»

Соединяем Table1, Table2 с базой данных устанавливая свойство DatabaseName=date. А также для таблицы Table1 устанавливаем имя таблицы, из которой мы берем записи Tablename=Student, Datasource6 устанавливаем Table1. Для таблицы Table2 устанавливаем имя таблицы, из которой мы берем записи Tablename=Gruppa, Datasource2 устанавливаем Table2. Для того чтобы связать таблицы Table1 и Table2, таким образом чтобы установив в таблице Gruppa курсор на ту или иную группу в таблице Student выводились только те записи, которые соответствуют данной группе. Для этого установим у Table1 Student свойства: Mastersource=Datasource2, MasterFields=gruppa (смотреть рисунок), IndexFieldName=gruppa.

Для того чтобы внести новую группу необходимо внести значения Группу, Год поступления и Эдвайзера в соответствующие Edit4, Edit5, Edit6, затем нажать кнопку «Вставить», ниже представлен обработчик OnClick:

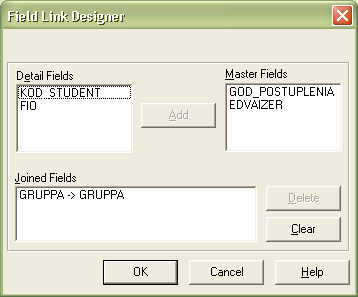


Рисунок 10 - Процесс соединения MasterFields для таблиц

*procedure TForm1.Button8Click(Sender: TObject);*

*begin*

*table2.Insert;*

*Table2.Fields[0].AsString := Edit4.Text;*

*Table2.Fields[1].AsString := edit5.Text;*

*Table2.Fields[2].AsString := Edit6.Text;*

*Table2.Post;*

*edit4.Clear;*

*edit5.Clear;*

*edit6.clear;*

*end;*

Для удаления записи из таблицы, воспользуемся кнопкой «Удалить», вставив в обработчик события OnClick следующую строчку:

table2.Delete;, которая удалит сведения о группе, на которой стоит курсор.

Также возможен поиск по группе. Для этого нам необходимо ввести название группы которую необходимо найти и нажать кнопку «Поиск». Код данной процедуры:

*procedure TForm1.Button13Click(Sender: TObject);*

*begin*

*Table2.SetKey;*

*Table2.Fields[0].AsString := Edit7.Text;*

*Table2.GotoKey;*

*edit7.Clear;*

*end;*

Итак мы автоматизировали работу с записями о группах, которая позволяет нам работать с сведениями о группах, а также просматривать список студентов выбранной группы.

**3.1.2. Формирование системы Student**

Перейдя на страницу Список студентов расположим на ней Query4, DataSource1, DbGrid, 2 кнопки Button,2 текстовых поля Edit, 2 Label и 4 SpeedButton. Затем установим для них свойства (см. рисунок 11).

Соединить Query с базой данных, установив DatabaseName=date и записав в SQL: select fio, gruppa from student. Установить у DataSource1 Dataset=Query4, чтобы он хранил набор данных Query, а для Dbrid установить DataSource1, чтобы в таблице мы могли увидеть результат запроса. 4 SpeedButton необходимы нам для перемещения по записям в таблице переходя: на следующую запись, первую, последнюю и предыдущую. Для этого мы запишем в обработчик событий для каждой из кнопок соответствующую строчку:

query4.Next;//следующая;

query4.First;//первая;

query4.Last;//последняя;

query4.Prior;//предыдущая.

Для того чтобы внести в таблицу соответствующее значение запишем для кнопки «Вставить» соответствующий код:

*procedure TForm1.Button1Click(Sender: TObject);*

*begin*

*query4.Insert;*

*query4.FieldByName('FIO').AsString:= Edit2.Text;*

*query4.FieldByName('GRUPPA').AsString:= Edit3.Text;*

*query4.Post;*

*edit2.Clear;*

*edit3.Clear;*

*end;*

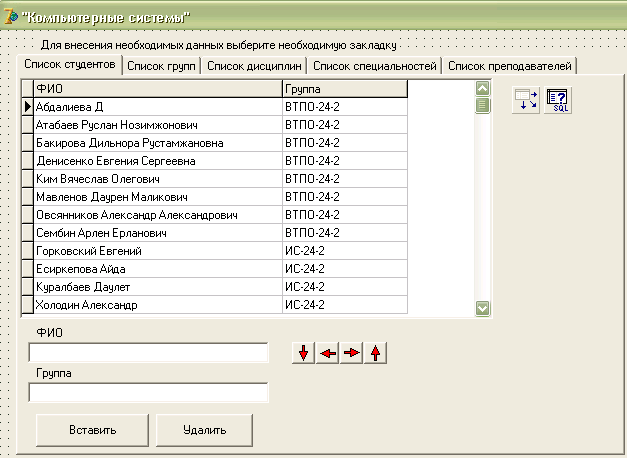


Рисунок 11 - Страница «Список студентов»

Для удаления записи из таблицы, воспользуемся кнопкой «Удалить», вставив в обработчик события OnClick следующую строчку:

*query4.Delete;*

Т.к. удаление записи является безвозвратной, то необходимо для query4 записать событие:

*procedure TForm1.Query4BeforeDelete(DataSet: TDataSet);*

*begin*

*if not (MessageDlg('Вы действительно хотите удалить запись?',*

*mtError, [mbYes, mbNo], 0)=mrYes) then*

*begin*

*Abort;*

*end;*

*end;*

Итак мы автоматизировали работу с таблицей Student, которая позволяет работать с записями. На рисунке представлена форма в откомпилированном виде:



Рисунок 12 - Процесс удаления студента из списка

**3.1.3. Формирование системы Disciplina**

Перейдя на страницу Список дисциплин расположим на ней Query1, DataSource 3, 1 DbGrid, 2 кнопки Button, 2 текстовых поля Edit, 2 Label. Затем установим для них свойства (см. рисунок 13):

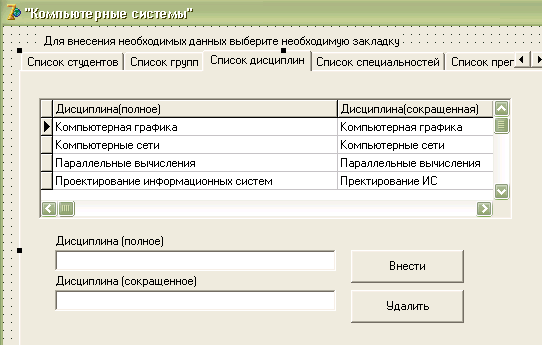


Рисунок 13 - Страница «Список дисциплин»

Установим соединение Query1, DataSource 3, DbGrid (также как и в предыдущем параграфе). Установим для Query1 свойство SQL равным:

select disciplina\_p, disciplina\_s from Disciplina

Order by disciplina\_p;

Обработчики событий для «Внести»:

*procedure TForm1.Button15Click(Sender: TObject);*

*begin*

*query1.Insert;*

*query1.FieldByName('disciplina\_p').AsString:=edit8.Text;*

*query1.FieldByName('disciplina\_s').AsString:=edit9.Text;*

*query1.Post; edit8.Clear; edit9.Clear;*

*end;*

Обработчики событий для «Удалить»:

*procedure TForm1.Button16Click(Sender: TObject);*

*begin*

*query1.Delete;*

*end;*

Т.к. удаление записи является безвозвратной, то необходимо для query1 записать событие:

*procedure TForm1.Query1BeforeDelete(DataSet: TDataSet);*

*begin*

*if not (MessageDlg('Вы действительно хотите удалить запись ?',*

*mtError, [mbYes, mbNo], 0)=mrYes) then*

*begin*

*Abort;*

*end;*

*end;*

**3.1.4. Формирование системы Specialnost**

Перейдя на страницу Список специальностей расположим на ней Query2, DataSource 4, 1 DbGrid, 2 кнопки Button, 4 текстовых поля Edit, 4 Label. Затем установим для них свойства (см. рисунок 14).

Установим для Query2 свойство SQL равным:

select \* from specialnost;

Обработчики событий для «Внести»:

*procedure TForm1.Button20Click(Sender: TObject);*

*begin*

*query2.Insert;*

*query2.FieldByName('shifr\_specialnost').AsString:=edit1.Text;*

*query2.FieldByName('specialnost\_p').AsString:=edit11.Text;*

*query2.FieldByName('specialnost\_s').AsString:=edit12.Text;*

*query2.FieldByName('god\_vvedenia').AsString:=edit13.Text;*

*query2.Post;*

*edit1.Clear;edit11.Clear;*

*edit12.Clear;edit13.Clear;*

*end;*

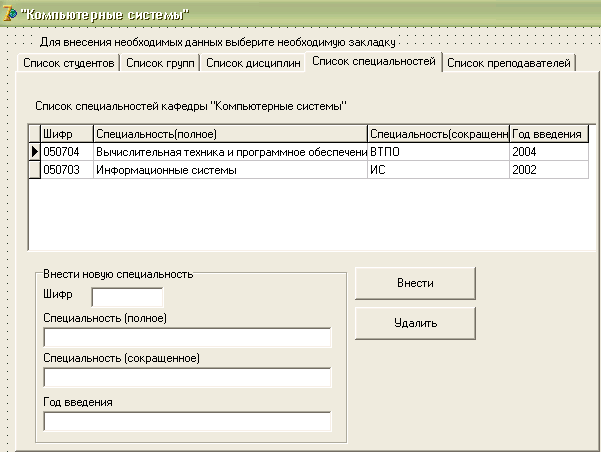


Рисунок 14 - Страница «Список специальностей»

Обработчики событий для «Удалить»:

*query2.Delete;*

Т.к. удаление записи является безвозвратной, то необходимо для query2 записать событие:

*procedure TForm1.Query2BeforeDelete(DataSet: TDataSet);*

*begin*

*if not (MessageDlg(' Вы действительно хотите удалить запись?',*

*mtError, [mbYes, mbNo], 0)=mrYes) then*

*begin*

*Abort;*

*end;*

*end;*

**3.1.5. Формирование системы Prepodavatel**

Перейдя на страницу Список преподавателей расположим на ней Query3, DataSource 5, 1 DbGrid, 2 кнопки Button, 2 текстовых поля Edit, 2 Label. Затем установим для них свойства (см. рисунок 15).

Установим для Query3 свойство SQL равным:

select fio, doljnost from prepodavatel

order by fio;

Обработчики событий для «Вставить»:

*procedure TForm1.Button17Click(Sender: TObject);*

*begin*

*query3.Insert;*

*query3.FieldByName('fio').AsString:=edit10.Text;*

*query3.FieldByName('doljnost').AsString:=edit14.Text;*

*query3.Post;edit10.Clear;edit14.Clear;*

*end;*

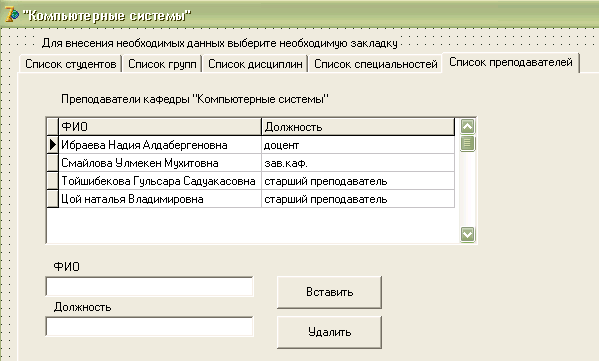


Рисунок 15 - Страница «Список преподавателей»

Обработчики событий для «Удалить»:

Query3.Delete;

Т.к. удаление записи является безвозвратной, то необходимо для query3 записать событие:

*procedure TForm1.Query3BeforeDelete(DataSet: TDataSet);*

*begin*

*if not (MessageDlg(' Вы действительно хотите удалить запись?',*

*mtError, [mbYes, mbNo], 0)=mrYes) then*

*begin*

*Abort;*

*end;*

*end;*

**3.2. Автоматизация работы системы Vedomost**

Для автоматизации работы с таблицей Vedomost разместим на форме 2Query,1DataSource, 1DBgrid, 5Combobox, 1GroupBox, 3Checkbox, 2Button, 5Label. Установим значения компонентов, как на рисунке 16.

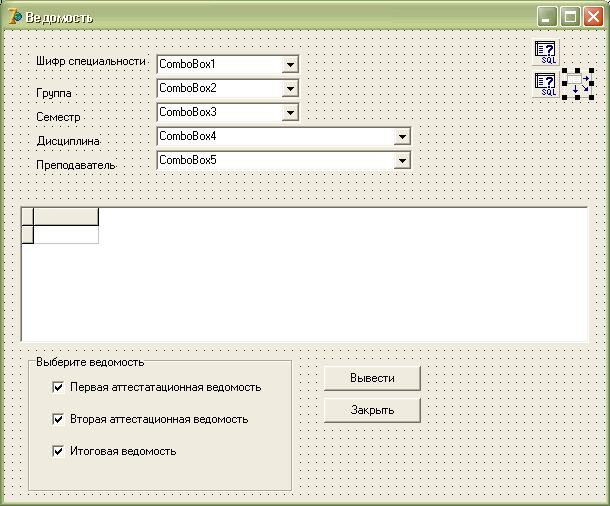


Рисунок 16 – Ведомость

Соединим Query1 и Query2 с базой данных. Установим значение DataSource1, DataSource=Query2. Работа формы заключается в том, что мы выбираем при помощи Combobox значения шифра специальности, группы, семестра, дисциплины, преподавателя и нажав кнопку «Вывести» в DBgrid мы увидим выставленные оценки студентов. Здесь мы применяем уже не статический, а динамический SQL, т.е. SQL-код для выборки стоит уже не в свойствах компонента Query, а применяется программно.

Для заполнения Combobox необходимыми значениями полей из разных таблиц мы объявляем в переменные:

sp\_shifr: TstringList;//список шифров специальностей из таблицы Specialnost

sp\_gr: TstringList;//список групп из таблицы Gruppa

sp\_sem: TstringList;//список семестров из таблицы Disciplina

sp\_disc: TstringList;//список дисциплин из таблицы Disciplina

sp\_prep: TstringList; //список преподавателей из таблицы Prepodavarel

А для того, чтобы в Combobox появились значения необходимо написать обработчик события FormCreate для Form2

*procedure TForm2.FormCreate(Sender: TObject);*

*begin*

*combobox1.Clear;*

*combobox2.Clear;*

*combobox3.Clear;*

*combobox4.Clear;*

*combobox5.Clear;*

*//shifr\_specialnost*

*sp\_shifr:=TstringList.Create;*

*query1.Close;*

*with query1 do*

*begin*

*close;*

*sql.Clear;*

*sql.Add('select shifr\_specialnost');*

*sql.Add('from specialnost');*

*sql.Add('order by shifr\_specialnost');*

*open;*

*while not eof do*

*begin*

*combobox1.Items.Add(fields[0].asstring);*

*sp\_shifr.Add(fields[0].asstring);*

*next;*

*end;*

*end;*

*//gruppa*

*sp\_gr:=TstringList.Create;*

*query1.Close;*

*with query1 do*

*begin*

*close;*

*sql.Clear;*

*sql.Add('select gruppa');*

*sql.Add('from gruppa');*

*sql.Add('order by gruppa');*

*open;*

*while not eof do*

*begin*

*combobox2.Items.Add(fields[0].asstring);*

*sp\_gr.Add(fields[0].asstring);*

*next;*

*end;*

*end;*

*//semestr*

*sp\_sem:=TstringList.Create;*

*query1.Close;*

*with query1 do*

*begin*

*close;*

*sql.Clear;*

*sql.Add('select semestr');*

*sql.Add('from disciplina');*

*sql.Add('group by semestr');*

*sql.Add('order by semestr');*

*open;*

*while not eof do*

*begin*

*combobox3.Items.Add(fields[0].asstring);*

*sp\_sem.Add(fields[0].asstring);*

*next;*

*end;*

*end;*

*//disciplina*

*sp\_disc:=TstringList.Create;*

*query1.Close;*

*with query1 do*

*begin*

*close;*

*sql.Clear;*

*sql.Add('select disciplina\_p');*

*sql.Add('from disciplina');*

*sql.Add('order by disciplina\_p');*

*open;*

*while not eof do*

*begin*

*combobox4.Items.Add(fields[0].asstring);*

*sp\_disc.Add(fields[0].asstring);*

*next;*

*end;*

*end;*

*//prepodavatel*

*sp\_prep:=TstringList.Create;*

*query1.Close;*

*with query1 do*

*begin*

*close;*

*sql.Clear;*

*sql.Add('select fio');*

*sql.Add('from prepodavatel');*

*sql.Add('order by fio');*

*open;*

*while not eof do*

*begin*

*combobox5.Items.Add(fields[0].asstring);*

*sp\_prep.Add(fields[0].asstring);*

*next;*

*end;*

*end;*

Далее для того, чтобы для указанных условий выборки в таблице DBgrid вышли соответствующие им данные необходимо выполнить следующий динамический запрос, который будет выполняться при нажатии кнопки «Вывести». Ниже представлен код данной процедуры:

*procedure TForm2.Button3Click(Sender: TObject);*

*begin*

*with query2 do begin*

*query2.Close;*

*sql.Clear;*

*sql.add('select st.fio, v.lc1, v.pr1, v.srs1, v.lc2, v.pr2, v.srs2, v.examen, v.itog');*

*sql.Add('from vedomost v inner join specialnost s on (v.shifr\_specialnost=s.shifr\_specialnost)');*

*sql.Add('inner join student st on (v.kod\_student=st.kod\_student)');*

*sql.Add('inner join gruppa g on (st.gruppa=g.gruppa)');*

*sql.Add('inner join disciplina d on (v.kod\_disciplina=d.kod\_disciplina)');*

*sql.Add('inner join prepodavatel p on (v.kod\_prepod=p.kod\_prepod)');*

*sql.Add('where 1=1');*

*sql.Add('and s.shifr\_specialnost='''+sp\_shifr[combobox1.itemindex]+'''');*

*sql.Add('and g.gruppa='''+sp\_gr[combobox2.itemindex]+'''');*

*sql.Add('and d.disciplina\_p='''+sp\_disc[combobox4.itemindex]+'''');*

*sql.Add('and p.fio='''+sp\_prep[combobox5.itemindex]+'''');*

*open;*

*end;*

*end;*

В результате при установлении всех условий выборки и нажатии кнопки «Вывести» мы получим данные об успеваемости за весь семестр. Но если нам необходимо узнать данные только первой аттестационной ведомости, второй аттестационной ведомости или итоговой ведомости, мы можем выбрать с помощью одного из трех Checkbox, которые сгруппированы в GroupBox.

Т.е. при нажатии Checkbox1 мы увидим данные первой аттестационной ведомости

*procedure TForm2.CheckBox1Click(Sender: TObject);*

*begin*

*query2.FieldByName('lc2').Visible:=checkbox1.Checked;*

*query2.FieldByName('pr2').Visible:=checkbox1.Checked;*

*query2.FieldByName('srs2').Visible:=checkbox1.Checked;*

*query2.FieldByName('examen').Visible:=checkbox1.Checked;*

*query2.FieldByName('itog').Visible:=checkbox1.Checked;*

*end;*

*,Checkbox2 – второй*

*procedure TForm2.CheckBox2Click(Sender: TObject);*

*begin*

*query2.FieldByName('lc1').Visible:=checkbox2.Checked;*

*query2.FieldByName('pr1').Visible:=checkbox2.Checked;*

*query2.FieldByName('srs1').Visible:=checkbox2.Checked;*

*query2.FieldByName('examen').Visible:=checkbox2.Checked;*

*query2.FieldByName('itog').Visible:=checkbox2.Checked;*

*end;*

*,а Checkbox3 – итоговая ведомость*

*procedure TForm2.CheckBox3Click(Sender: TObject);*

*begin*

*query2.FieldByName('lc1').Visible:=checkbox3.Checked;*

*query2.FieldByName('pr1').Visible:=checkbox3.Checked;*

*query2.FieldByName('srs1').Visible:=checkbox3.Checked;*

*query2.FieldByName('lc2').Visible:=checkbox3.Checked;*

*query2.FieldByName('pr2').Visible:=checkbox3.Checked;*

*query2.FieldByName('srs2').Visible:=checkbox3.Checked;*

*end;*

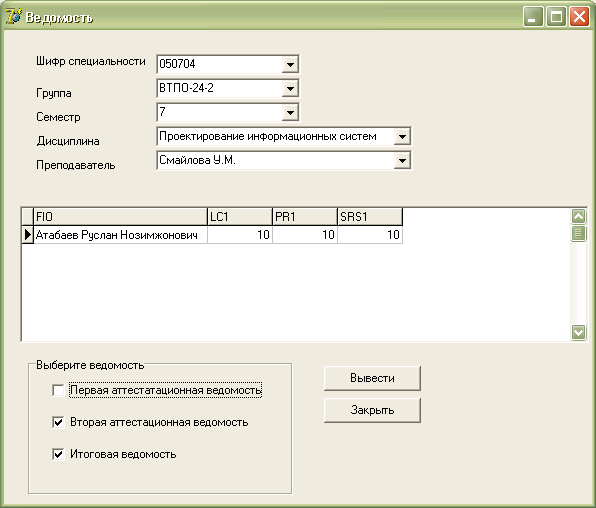


Рисунок17 - Ведомость при выполнениях условий выборки

**3.3. Автоматизация работы системы Uchebnii\_plan**

Для автоматизации работы с таблицей Uchebnii\_plan разместим на форме 3Query, 2DataSource, 1DBgrid, 2Combobox, 2Checkbox, 3Button, 4Label. Установим значения компонентов, как на рисунке 18.

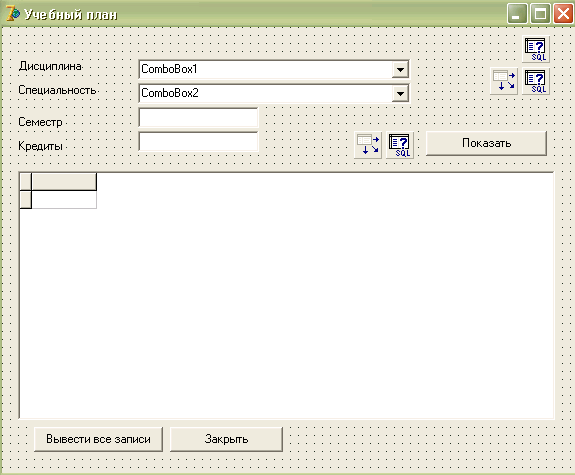


Рисунок 18 - Учебный план

Соединим Query1 и Query2, Query3 с базой данных. Установим значение DataSource1 в DataSource=Query2, DataSource2 в DataSource=Query2. Работа формы заключается в том, что мы выбираем при помощи Combobox значения дисциплины, шифра специальности и нажав кнопку «Показать» в Edit1, Edit2 мы увидим семестр и количество кредитов, а также просмотр всего учебного плана. Здесь мы применяем уже не статический, а динамический SQL, т.е. SQL-код для выборки стоит уже не в свойствах компонента Query, а применяется программно.

Для заполнения Combobox необходимыми значениями полей из разных таблиц мы объявляем в переменные:

sp\_shifr: TstringList;//список шифров специальностей из таблицы Specialnost

sp\_disc: TstringList;//список дисциплин из таблицы Disciplina

А для того, чтобы в Combobox появились значения необходимо написать обработчик события FormCreate для Form3:

*procedure TForm3.FormCreate(Sender: TObject);*

*begin*

*combobox1.Clear; combobox2.Clear;*

*//shifr specialnost*

*sp\_shifr:=TstringList.Create;*

*query1.Close;*

*with query1 do*

*begin*

*close;*

*sql.Clear;*

*sql.Add('select shifr\_specialnost');*

*sql.Add('from specialnost');*

*sql.Add('order by shifr\_specialnost');*

*open;*

*while not eof do*

*begin*

*combobox2.Items.Add(fields[0].asstring);*

*sp\_shifr.Add(fields[0].asstring);*

*next;*

*end;*

*end;*

*//disciplina*

*sp\_disc:=TstringList.Create;*

*query1.Close;*

*with query1 do*

*begin*

*close;*

*sql.Clear;*

*sql.Add('select disciplina\_p');*

*sql.Add('from disciplina');*

*sql.Add('order by disciplina\_p');*

*open;*

*while not eof do*

*begin*

*combobox1.Items.Add(fields[0].asstring);*

*sp\_disc.Add(fields[0].asstring);*

*next;*

*end;*

*end;*

Далее для того, чтобы для указанных условий выборки в таблице DBgrid вышли соответствующие им данные необходимо выполнить следующий динамический запрос, который будет выполняться при нажатии кнопки «Вывести».

Весь список входящий в учебный план мы увидим нажав кнопку «Вывести все записи». Ниже представлен код данной процедуры:

*procedure TForm3.Button2Click(Sender: TObject);*

*begin*

*with query3 do begin*

*query3.Close; sql.Clear;*

*sql.Add('select d.disciplina\_p, s.shifr\_specialnost, d.semestr, u.kol\_vo\_kreditov');*

*sql.Add('from uchebnii\_plan u, specialnost s, disciplina d ');*

*sql.Add('where (u.kod\_disciplina=d.kod\_disciplina)');*

*sql.Add('and (u.shifr\_specialnost=s.shifr\_specialnost)');*

*//sql.Add('u.semestr=d.semestr'); open;*

*end;*

*end;*

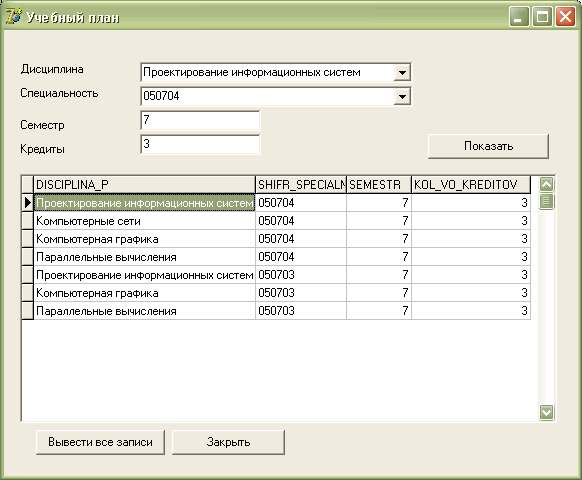


Рисунок 19 - Учебный план в откомпилированном виде.

**ЗАКЛЮЧЕНИЕ**

Автоматизация технологических и управленческих процессов, без которой не мыслимо эффективное решение задач управления промышленным или торговым предприятием, банком или учебным заведением, государственной структурой, основывается на переработке больших объемов информации.

Эффективность автоматизированных информационных управляющих систем в значительной мере зависит от того, насколько обеспечивается скорость доступа к данным, их полнота, достоверность, непротиворечивость. И практически везде информационная система представляет собой интегрированную систему, ядро которой представляет база данных. Основным направлением в разработке автоматизированных информационных систем в настоящее время является ориентация на использование СУБД, базирующихся на SQL– серверах [7].

Данная самостоятельнаяработа посвящена проектированию системы учета и анализа успеваемости студентов кафедры «Компьютерные системы». Целью проектирования является повышение эффективности работы кафедры «Компьютерные системы», снижение трудоемкости процессов обработки информации.

На сегодняшний день снижение времени принятия управленческих решений является актуальным для руководителей организаций, так как это позволит наиболее эффективно работать организации.

Для физической реализации базы данных в качестве целевой СУБД использована Delphi7. Данная система позволяет хранить и обрабатывать информацию о студентах кафедры, группах, специальностях, дисциплинах и преподавателях, а также позволяет создавать рабочий учебный план и производить учет успеваемости студентов по результатам аттестаций и экзамена. В качестве сервера для администрирования базы данных был выбран Interbase, т.к. InterBase представляет собой полнофункциональный SQL- сервер. Языком администрирования InterBase является интерактивный SQL, который используется в специальных утилитах (типа WISQL или DBD), позволяющих в интерактивном режиме вводить запросы с использованием команд SQL, посылать их для выполнения на сервер и получать результаты в предназначенном для этого окне. Встроенный SQL используется в прикладных программах, позволяя им посылать запросы к серверу и обрабатывать полученные результаты, в том числе комбинируя set-ориентированный и record-ориентированный подходы.

**СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

1. Архангельский А.Я. Delphi5. Спр-ное пособие. – М.: ЗАО Изд. БИНОМ,2001
2. Астахова И.Ф. и др. SQL в примерах и задачах. Из-во Питер, 2001
3. Бобровский С. Delphi5. Учеб. курс. Изд. Дом ПИТЕР, 2001
4. Бойко В.В., Савинков В.М. Проектирование баз данных информационных систем. – М.: Финансы и статистика, 1989
5. Бондарь А. InterBase и Firebird. Прак. рук-во для умных польз. и нач. разработчиков. «БХВ-Петербург», 2007
6. Боровиков В.В. MS ACCESS 2002. Программирование и разработка БД и приложении. М.Фин. и стат-ка,2002
7. Дарахвилидзе П., Марков Е. Программирование в Delphi7. Санкт-Петербург, «БХВ-Петербург», 2003
8. Дейв Энсон, Йен Стивенсон. Проектирование баз данных. Oracle. /Пер. с англ. –К.:Издательская группа BHV, 1999
9. Диго С.М. Проектирование и использование баз данных. М.: ФС, 1995
10. Епанешников . А.М. Епанешников В.А. Delphi. Проектирование СУБД. М., Диалог МИФИ 2001
11. Калянов Г.М. CASE-технологии. Консалтинг и автоматизации бизнес-процессов. 2-е изд. –М.:Горячая линия⎯Телеком, 2000
12. Кандзюба С.П., Громов В.Н. Delphi6/7. Базы данных и приложения. Лекции и упражнения. -К.:Издательство «ДиаСофт ЮП», 2002
13. Кириллов В.В. Структурированный язык запросов SQL. – СПб.: ИТМО, 1994
14. Культин Н.Б. Delphi6. Программирование на Object Pascal. Изд-во.: БХВ Санкт-Петербург, 2002
15. Маклаков С.В. BPWin и ERWin. CASE -средства разработки информационных систем. М. ДИАЛОГ-МИФИ, 2000
16. Маклаков С.В. Моделирование бизнес-процессов с BPWin4.0. М. ДИАЛОГ-МИФИ, 2002
17. Петров В.Н. Информационные системы. СПб.: Питр,2002
18. Томас Конноли, Каролин Бегг, Анна Срачан. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. М., СПб., К., 2000 2- изд
19. Фаронов В.В. Delphi 3. Учебный курс. М.:”Ноллидж”, 1998
20. Федоров А., Елманова Н. Базы данных для всех. – М.: КомпьПресс, 2001
21. Фролов А.В., Фролов Г.В. БД в Интернете: Прак РУК. по созд Web-приложений с БД. М.: Изд-во-ТД “Русс. Ред”, 2000
22. Хомоненко А.Д., Цыганков В.М., Мальцев М.Г. Базы данных. . ⎯ СПб.:КОРОНА принт, 2000
23. Шейкер Т.Д. Разработка приложений в системе Delphi. Владивосток: Изд-во ДВГТУ, 2006.
24. Шумаков П.В. Delphi3 и разработка приложений БД. М.Изд-во «Нолидж», 1998
25. Чекалов А.П. Базы данных: от проектирования до разработки приложения. – СПб.: БХВ-Петербург, 2003
26. Чекалов А.П. Базы данных: Delphi, SQL сервера. – СПб.: БХВ-Петербург, 2003

**ПРИЛОЖЕНИЯ**

**ТЕМЫ САМОСТОЯТЕЛЬНЫХ****РАБОТ**

**по дисциплине *Проектирование баз данных***

**для студентов специальностей Информатика, ИС, ВТиПО**

1. Проектирование базы данных контроля экспортно-импортных валютных операций банка
2. Проектирование базы данных учета материальных ценностей
3. Проектирование базы данных распределения и использования водных ресурсов
4. Проектирование базы данных расчета сдельной заработной платы
5. Проектирование базы данных ссудного портфеля
6. Проектирование базы данных расчетно-кредитного отдела банка
7. Проектирование БД формирования себестоимости готовой продукции
8. Проектирование базы данных расчетов с потребителями
9. Проектирование базы данных сбора страховых платежей и их возмещения
10. База данных расчетов с абонентами за телекоммуникационные услуги
11. Проектирование базы данных выдачи и погашения кредитов
12. Проектирование базы данных учета материальных запасов на оборотном складе
13. Проектирование базы данных налогообложения физических лиц
14. Проектирование базы данных накопительного пенсионного фонда
15. Проектирование базы данных клиентов кредитного отдела
16. Проектирование базы данных пассажирского автотранспорта
17. Проектирование базы данных регистрации кассовых аппаратов
18. Проектирование базы данных регистрации индивидуальных налогоплательщиков (ИНН)
19. Проектирование базы данных Железнодорожных перевозок
20. Проектирование базы данных кассы автовокзала
21. Проектирование базы данных формирования библиотечного фонда
22. Базы данных диссертационного фонда научно-технической библиотеки
23. Базы данных регистрации и учета архивных документов
24. Формирование информационного фонда адресного бюро
25. Формирование информационного фонда агентства по продаже, покупке и обмену жилищного фонда
26. Формирование информационного фонда справочного бюро абонентов телекоммуникационных услуг
27. Базы данных учета документов отдела кадров
28. Формирование каталога информационно-педагогического модуля
29. Проектирование базы данных по продаже автомобилей
30. Проектирование базы данных по продаже коммерческой недвижимости
31. Проектирование базы данных строительного оборудования и материалов
32. Проектирование базы данных по продаже комплектующих персональных компьютеров
33. Проектирование базы поисковых каталогов юридических услуг
34. Проектирование базы данных бюро по трудоустройству населения
35. Проектирование государственного земельного кадастра
36. Проектирование базы картографической информации для автовладельцев
37. Проектирование базы данных сбыта готовой продукции
38. Проектирование базы данных учета основных средств.
39. Проектирование базы данных складского учета
40. Автоматизация учета труда и его оплаты
41. Проектирование базы данных оперативного учета производства.
42. Проектирование базы данных расчета с абонентами энергокомбината.
43. Проектирование базы данных налогообложения юридических лиц.
44. Проектирование базы данных "Депозиты"
45. Проектирование базы данных учета потребления воды населением
46. Проектирование базы данных отдела кадров

**Примерная структура и трудоемкость выполнения самостоятельной****работы**

**Структура и трудоемкость выполнения СРС**

|  |  |  |
| --- | --- | --- |
| № | Наименование разделов | Планируемая трудоемкость  (в % от общего объема) |
|  | **Титульный лист** |  |
|  | **Бланк задания** |  |
|  | **Содержание работы** |  |
|  | **Техническое задание** | 5% |
|  | **Введение** | 5% |
| 1 | **Проектирование модели данных предметной области с использованием CASE-средств** | 40% |
| 1.1 | Постановка задач. Анализ и моделирование предметной области. Выделение объектных областей. |  |
| 1.2 | Проектирование ERD (Entity-Relationship Diagrams) модели данных объектных областей с помощью программы ERwin. |  |
| 1.3 | Определение ограничений БД, установление ссылочной целостности, проектирование индексов |  |
| 1.4 | Проектирование глобальной инфологической модели данных предметной области |  |
| **2** | **Реализация БД в SQL-сервере InterBase/ Firebird** | 30% |
| 2.1 | Определение доменов, таблиц БД, представлении |  |
| 2.2 | Описание ключей: первичных, альтернативных, индексных. |  |
| 2.3 | Проектирование генераторов, триггеров и хранимых процедур |  |
| 2.4 | Организация защиты БД. |  |
| **3** | **Организация доступа к данным. Программа IBExpert** | 15% |
|  | **Заключение** | 3% |
|  | Список использованной литературы |  |
|  | Приложения:  А: Исходные данные для создания БД  В: Описание объектов БД |  |
|  | Защита самостоятельной работы студента | 2% |