Федеральное агентство по образованию Российской Федерации (РФ)

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра Электронных приборов (ЭП)

УТВЕРЖДАЮ
Заведующий кафедрой ЭП
_____ С.М. Шандаров

Системы управления базами данных

Учебно-методическое пособие
Методические указания к лабораторным работам

Разработчик:
ст. преподаватель каф. ЭП
_____ Е.С. Шандаров

2007

Содержание

Введение

Данный курс лабораторных работ посвящен теме «Системы управления базами данных» и включает в себя описание 6 лабораторных работ.

Лабораторные работы по курсу проводятся с использованием программного обеспечения бесплатно распространяемого пакета OpenOffice.org, который включает в себя текстовый процессор Writer, электронную таблицу Calc, программу для подготовки презентаций Impress и настольную СУБД Base.

В рамках данного курса студенты осваивают продукт OpenOffice Base, создают и модифицируют учебную базу данных, осваивают построение запросов на языке SQL.

Инструменты необходимые для выполнения лабораторных работ

Как было отмечено выше, для выполнения лабораторных работ Вам понадобится пакет OpenOffice.org версии 2.0 или выше.

OpenOffice.org ведет свое происхождение от офисного пакета StarOffice, разработанного немецкой фирмой StarDivision в середине 90-х годов. Осенью 1999 года корпорация Sun купила StarDivision. В июне 2000 года, уже под торговой маркой SUN вышел StarOffice 5.2 для MS Windows, Linux и Solaris.

13 октября 2000 года были открыты исходные тексты StarOffice (за исключение кода некоторых модулей, разработанных третьими фирмами), и этот день официально считается днем рождения OpenOffice.org.

В настоящее время над кодом OpenOffice.org работают как добровольцы со всего света, так и программисты корпорации SUN. Sun Microsystems, Inc в основном и финансирует деятельность проекта OpenOffice.org.

В состав OpenOffice.org входят следующие компоненты:

- Writer (текстовый процессор и редактор HTML).
- Calc (электронные таблицы).
- Draw (графический редактор).
- Impress (система презентаций).
- Base (база данных)
- Редактор формул.

По своим возможностям OpenOffice.org вполне сопоставим с известным MS Office.

Компонент Base является одним из нововведений в версии 2.0; он позволяет без проблем получить доступ к БД в OpenOffice.org 2.0. Base позволяет получить доступ к различным БД или создать БД, используя HSQL.

С помощью доступных в Base Мастеров возможно создать простую базу данных (БД) с формами и отчётами.

Описание учебной базы данных

В данном лабораторном практикуме мы будем работать с базой данных «Студенты кафедры ЭП».

БД «Студенты кафедры ЭП» является реляционной и состоит из совокупности связанных между собой плоских таблиц.

Все таблицы данной БД (как и все таблицы любой реляционной базы данных) состоят из строки заголовков столбцов и одной или более строк значений данных под этими заголовками. Эти столбцы и строки должны иметь следующие свойства:

- всякому столбцу таблицы присвоено имя, которое должно быть уникальным для этой таблицы;
- столбцы таблицы упорядочиваются слева направо, т.е. столбец 1, столбец 2, ..., столбец n. С математической точки зрения это утверждение некорректно, потому что в реляционной системе столбцы не упорядочены. Однако с точки зрения пользователя, порядок, в котором определены имена столбцов, становится порядком, в котором должны вводиться в них данные, если не предварять при вводе каждое значение именем соответствующего столбца;
- строки таблицы не упорядочены (их последовательность определяется лишь последовательностью ввода в таблицу);
- в поле на пересечении строки и столбца любой таблицы всегда имеется только одно значение данных и никогда не должно быть множества значений (правда, это "атомарное" значение может быть достаточно объемным, например, таким, как описание учебного курса);
- всем строкам таблицы соответствует одно и то же множество столбцов, хотя в определенных столбцах любая строка может содержать пустые значения (NULL-значения), т.е. может не иметь значений для этих столбцов;
- все строки таблицы обязательно отличаются друг от друга хотя бы

единственным значением, что позволяет однозначно идентифицировать любую строку такой таблицы;

- при выполнении операций с таблицей ее строки и столбцы можно обрабатывать в любом порядке безотносительно к их информационному содержанию.

Почему же база данных, составленная из таких таблиц, называется реляционной? А потому, что отношение - relation - просто математический термин для обозначения неупорядоченной совокупности однотипных записей или таблиц определенного специфического вида.

Данная БД состоит из следующих таблиц:

- Таблица групп - **groups** - содержит информацию о группах, обучающихся на кафедре ЭП
- Таблица студентов - **students** - содержит информацию о студентах, обучающихся на кафедре ЭП
- Таблица преподавателей кафедры – **prepods** – содержит информацию о преподавателях кафедры ЭП
- Таблица **subjects** – содержит информацию об учебных курсах преподаваемых на кафедре ЭП
- Таблица **marks** – содержит информацию об оценках студента по прослушанным учебным курсам

База данных «Студенты кафедры ЭП» предназначена для заполнения, хранения и выборки информации об учебном процессе на кафедре ЭП. Основными сущностями данной БД являются студенты, учебные группы, преподаватели, учебные курсы и оценки студентов по итогам изучения учебных курсов.

Данная БД должна включать в себя средства по внесению информации

посредством экранных форм и ручного заполнения таблиц, подготовки отчетности посредством выполнения операторов языка SQL.

Структура таблиц учебной базы данных

Таблица groups

- **id** - идентификатор группы, уникальное значение, тип INTEGER, первичный ключ, служит для связи между таблицей групп (groups) и студентов (students)
- **number** - номер группы, тип VARCHAR (строковый), соответствует типу нумерации, принятой в ТУСУР
- **curator_id** – внешний ключ куратора группы, тип INTEGER (ссылается на первичный ключ в таблице prepods)
- **year** - год поступления в ТУСУР, тип INTEGER

Таблица students

- **id** - идентификатор студента, уникальное значение, тип INT, первичный ключ
- **lastname** - фамилия студента, тип VARCHAR
- **firstname** - имя студента, тип VARCHAR
- **secondname** – отчество студента, тип VARCHAR
- **group_id** - идентификатор группы, тип INT, ссылается на поле id в таблице groups

Таблица prepods

- **id** – идентификатор преподавателя, уникальное значение, тип INTEGER, первичный ключ
- **lastname** – фамилия преподавателя, тип VARCHAR
- **firstname** – имя преподавателя, тип VARCHAR

- **secondname** – отчество преподавателя, тип VARCHAR

Таблица subjects

- **id** – идентификатор учебного курса, уникальное значение, тип INTEGER, первичный ключ
- **title** – название учебного курса, тип VARCHAR
- **prepod_id** – идентификатор преподавателя по данному учебному курсу, внешний ключ, ссылается на поле id в таблице prepods

Таблица marks

- student_id – идентификатор студента, компонент составного первичного ключа, ссылается на id в таблице students
- subject_id – идентификатор учебного курса, компонент составного первичного ключа, ссылается на id в таблице subjects
- mark – оценка данного студента по данному курсу, тип VARCHAR (таким образом, в данном поле оценка должна проставляться в текстовом виде, например: «отлично», «хорошо», «зачет»)

Структурированный язык запросов SQL

Все языки манипулирования данными (ЯМД), созданные до появления реляционных баз данных и разработанные для многих систем управления базами данных (СУБД) персональных компьютеров, были ориентированы на операции с данными, представленными в виде логических записей файлов. Это требовало от пользователей детального знания организации хранения данных и достаточных усилий для указания не только того, какие данные нужны, но и того, где они размещены и как шаг за шагом получить их.

Рассматриваемый же ниже непроцедурный язык SQL (Structured Query Language - структурированный язык запросов) ориентирован на операции с данными, представленными в виде логически взаимосвязанных совокупностей таблиц. Особенность предложений этого языка состоит в том, что они ориентированы в большей степени на конечный результат обработки данных, чем на процедуру этой обработки. SQL сам определяет, где находятся данные, какие индексы и даже наиболее эффективные последовательности операций следует использовать для их получения: не надо указывать эти детали в запросе к базе данных.

Реализация в SQL концепции операций, ориентированных на табличное представление данных, позволило создать компактный язык с небольшим (менее 30) набором предложений. SQL может использоваться как интерактивный (для выполнения запросов) и как встроенный (для построения прикладных программ). В нем существуют:

- предложения определения данных (определение баз данных, а также определение и уничтожение таблиц и индексов);
- запросы на выбор данных (предложение SELECT);
- предложения модификации данных (добавление, удаление и изменение данных);
- предложения управления данными (предоставление и отмена привилегий

на доступ к данным, управление транзакциями и другие). Кроме того, он предоставляет возможность выполнять в этих предложениях:

- арифметические вычисления (включая разнообразные функциональные преобразования), обработку текстовых строк и выполнение операций сравнения значений арифметических выражений и текстов;
- упорядочение строк и (или) столбцов при выводе содержимого таблиц на печать или экран дисплея;
- создание представлений (виртуальных таблиц), позволяющих пользователям иметь свой взгляд на данные без увеличения их объема в базе данных;
- запоминание выводимого по запросу содержимого таблицы, нескольких таблиц или представления в другой таблице (реляционная операция присваивания).
- агрегатирование данных: группирование данных и применение к этим группам таких операций, как среднее, сумма, максимум, минимум, число элементов и т.п.

В SQL используются следующие основные типы данных, форматы которых могут несколько различаться для разных СУБД:

INTEGER - целое число (обычно до 10 значащих цифр и знак);
SMALLINT - "короткое целое" (обычно до 5 значащих цифр и знак);
DECIMAL(p,q) - десятичное число, имеющее p цифр ($0 < p < 16$) и знак; с помощью q задается число цифр справа от десятичной точки ($q < p$, если $q = 0$, оно может быть опущено);
FLOAT - вещественное число с 15 значащими цифрами и целочисленным порядком, определяемым типом СУБД;
CHAR(n) - символьная строка фиксированной длины из n символов ($0 < n < 256$);

VARCHAR(n) - символьная строка переменной длины, не превышающей n символов (n > 0 и разное в разных СУБД, но не меньше 4096);
DATE - дата в формате, определяемом специальной командой (по умолчанию mm/dd/yy); поля даты могут содержать только реальные даты, начинающиеся за несколько тысячелетий до н.э. и ограниченные пятым-десятым тысячелетием н.э.;
TIME - время в формате, определяемом специальной командой, (по умолчанию hh.mm.ss);
DATETIME - комбинация даты и времени;
MONEY - деньги в формате, определяющем символ денежной единицы ($, руб, ...) и его расположение (суффикс или префикс), точность дробной части и условие для показа денежного значения.

В некоторых СУБД еще существует тип данных LOGICAL, DOUBLE и ряд других. Тем не менее, в целях обеспечения переносимости баз данных между различными в том числе и вычислительными платформами, целесообразно использовать ограниченный набор так называемых «канонических» типов данных. В данном лабораторном курсе мы будем ограничивать этот набор типами INTEGER, VARCHAR, DATE и FLOAT.

Ориентированный на работу с таблицами SQL не имеет достаточных средств для создания сложных прикладных программ. Поэтому в разных СУБД он либо используется вместе с языками программирования высокого уровня (например, такими как Си или Паскаль), либо включен в состав команд специально разработанного языка СУБД (язык систем dBASE, R:BASE и т.п.). Унификация полных языков современных профессиональных СУБД достигается за счет внедрения объектно-ориентированного языка четвертого поколения 4GL. Последний позволяет организовывать циклы, условные предложения, меню, экранные формы, сложные запросы к базам данных с интерфейсом, ориентированным как на алфавитно-цифровые терминалы, так и на оконный графический интерфейс (X-Windows, MS-Windows).

Все запросы на получение практически любого количества данных из одной или нескольких таблиц выполняются с помощью единственного предложения SELECT. В общем случае результатом реализации предложения SELECT является другая таблица. К этой новой (рабочей) таблице может быть снова применена операция SELECT и т.д., то есть такие операции могут быть вложены друг в друга. Представляет исторический интерес тот факт, что именно возможность включения одного предложения SELECT внутрь другого послужила мотивировкой использования прилагательного "структурированный" в названии языка SQL.

Предложение SELECT может использоваться как:

- самостоятельная команда на получение и вывод строк таблицы, сформированной из столбцов и строк одной или нескольких таблиц (представлений);
- элемент WHERE- или HAVING-условия (сокращенный вариант предложения, называемый "вложенный запрос");
- фраза выбора в командах CREAT VIEW, DECLARE CURSOR или INSERT;
- средство присвоения глобальным переменным значений из строк сформированной таблицы (INTO-фраза).

Нами будут рассмотрены только две первые функции предложения SELECT, а здесь – его синтаксис, ограниченный конструкциями, используемыми при реализации этих функций. Здесь в синтаксических конструкциях используются следующие обозначения:

- звездочка (*) для обозначения "все" - употребляется в обычном для программирования смысле, то есть "все случаи, удовлетворяющие определению";

- квадратные скобки ([]) – означают, что конструкции, заключенные в эти скобки, являются необязательными (т.е. могут быть опущены);
- фигурные скобки ({}) – означают, что конструкции, заключенные в эти скобки, должны рассматриваться как целые синтаксические единицы, т.е. они позволяют уточнить порядок разбора синтаксических конструкций, заменяя обычные скобки, используемые в синтаксисе SQL;
- многоточие (...) – указывает на то, что непосредственно предшествующая ему синтаксическая единица факультативно может повторяться один или более раз;
- прямая черта (|) – означает наличие выбора из двух или более возможностей. Например обозначение ASC|DESC указывает, можно выбрать один из терминов ASC или DESC; когда же один из элементов выбора заключен в квадратные скобки, то это означает, что он выбирается по умолчанию (так, [ASC]|DESC означает, что отсутствие всей этой конструкции будет восприниматься как выбор ASC);
- точка с запятой (;) – завершающий элемент предложений SQL;
- запятая (,) – используется для разделения элементов списков;
- пробелы ( ) – могут вводиться для повышения наглядности между любыми синтаксическими конструкциями предложений SQL;
- прописные жирные латинские буквы и символы – используются для написания конструкций языка SQL и должны (если это специально не оговорено) записываться в точности так, как показано;
- строчные буквы – используются для написания конструкций, которые должны заменяться конкретными значениями, выбранными пользователем, причем для определенности отдельные слова этих конструкций связываются между собой символом подчеркивания (_);
- термины таблица, столбец, ... – заменяют (с целью сокращения текста синтаксических конструкций) термины имя_таблицы, имя_столбца, ..., соответственно;
- термин таблица – используется для обобщения таких видов таблиц, как

базовая_таблица, представление или псевдоним; здесь псевдоним служит для временного (на момент выполнения запроса) переименования и (или) создания рабочей копии базовой_таблицы (представления).

Предложение SELECT имеет следующий формат:

```
SELECT [DISTINCT | ALL]{*
| [<выражение для столбца> [[AS] <псевдоним>]] [,…]}
FROM <имя таблицы> [[AS] <псевдоним>] [,…]
[WHERE <предикат>]
[[GROUP BY <список столбцов>]
[HAVING <условие на агрегатные значения>] ]
[ORDER BY <список столбцов>]
```

Рассмотрим вышеприведенный формат подробнее.
SELECT – ключевое слово, обозначающее начало запроса языка SQL.
Следом за ключевым словом SELECT должен быть приведен список столбцов таблицы или таблиц участвующих в запросе, которые должны присутствовать в результирующей таблице.
DISTINCT – ключевое слово, позволяющее исключить попадание в выборку результатов-дубликатов.
* - ключевой элемент показывающий что в выборку попадут все столбцы таблицы или таблиц.
AS – ключевое слово, позволяющее определить псевдоним для имени столбца. Это весьма полезно, например, в тех случаях когда в запросе участвуют таблицы с одинаковыми именами столбцов.
FROM – ключевое слово которое позволяет определить список таблиц, участвующих в запросе. Здесь также можно использовать псевдонимы с помощью ключевого слова AS.
WHERE – позволяет наложить условие по которому будет произведена выборка

необходимых данных. Условие представляет собой предикат имеющий значение true. В данное выражение может входить любое число различных условий для этого необходимо использовать ключевые слова AND, OR, NOT и другие.
GROUP BY – ключевое слово после которого следует выражение на основе которого будет производиться группировка результатов выполнения запроса. Используется вместе с агрегатными функциями.
HAVING – позволяет наложить условия на результат выполнения агрегатных функций.
ORDER BY – осуществляет упорядочивание результатов выборки на основе последующего списка столбцов.

Приведем ряд простых запросов к учебной базе данных.

Получить все записи таблицы students:

```
SELECT * FROM students
```

Получить все записи таблицы groups:

```
SELECT * FROM groups
```

Получить список всех фамилий студентов:

```
SELECT lastname FROM students
```

Получить список всех студентов отсортированный по фамилии:

```
SELECT lastname, firstname, secondname FROM students
ORDER BY lastname ASC
```

Здесь ключевое слово ASC обозначает сортировку по возрастанию, если необходим обратный порядок сортировки, используйте DESC.

Получить список преподавателей с именем Александр

```
SELECT * FROM prepods WHERE firstname='Александр'
```

Результатом данной выборки будет таблица, в которую попадут только те преподаватели кафедры имя которых Александр.

Получить список студентов фамилия которых начинается на букву «А»:

```
SELECT * FROM students WHERE lastname LIKE 'A%'
```

Здесь ключевое слово LIKE позволяет осуществить выборку по подстроке. Знак процента % показывает что здесь могут быть любые другие данные.

Для обеспечения связи двух таблиц в результирующей выборке можно воспользоваться условием в разделе WHERE:

```
SELECT students.lastname, students.firstname,
groups.number FROM students, groups WHERE
students.group_id = groups.id
```

Результатом выполнения данного запроса будет выборка состоящая из фамилии, имени и номера группы студента.

Продемонстрируем пример с использованием псевдонимов:

```
SELECT s.lastname AS 'Фамилия', s.firstname AS 'Имя' FROM
students s WHERE s.firstname LIKE 'O%'
```

Результатом данной выборки будет таблица с двумя столбцами: «Фамилия» и «Имя», строками таблицы будут студенты имя которых начинается с буквы «О».

Агрегирование данных

SQL-функции

В SQL существует ряд специальных стандартных функций (SQL-функций). Кроме специального случая COUNT(*) каждая из этих функций оперирует совокупностью значений столбца некоторой таблицы и создает единственное значение, определяемое так:

- COUNT - число значений в столбце,
- SUM - сумма значений в столбце,
- AVG - среднее значение в столбце,
- MAX - самое большое значение в столбце,
- MIN - самое малое значение в столбце.

Для функций SUM и AVG рассматриваемый столбец должен содержать числовые значения.

Следует отметить, что здесь столбец - это столбец виртуальной таблицы, в которой могут содержаться данные не только из столбца базовой таблицы, но и данные, полученные путем функционального преобразования и (или) связывания символами арифметических операций значений из одного или нескольких столбцов. При этом выражение, определяющее столбец такой таблицы, может быть сколь угодно сложным, но не должно содержать SQL-функций (вложенность SQL-функций не допускается). Однако из SQL-функций можно составлять любые выражения.

Аргументу всех функций, кроме COUNT(*), может предшествовать ключевое слово DISTINCT (различный), указывающее, что избыточные дублирующие значения должны быть исключены перед тем, как будет применяться функция. Специальная же функция COUNT(*) служит для

подсчета всех без исключения строк в таблице (включая дубликаты).

Если не используется фраза GROUP BY, то в перечень элементов_SELECT можно включать лишь SQL-функции или выражения, содержащие такие функции. Другими словами, нельзя иметь в списке столбцы, не являющихся аргументами SQL-функций.

Фраза GROUP BY (группировать по) инициирует перекомпоновку указанной во FROM таблицы по группам, каждая из которых имеет одинаковые значения в столбце, указанном в GROUP BY.

Таким образом строки таблицы группируются так, что в одной группе содержатся все строки для определенного значения в столбце по которому производится группировка. Далее к каждой группе применяется фраза SELECT. Каждое выражение в этой фразе должно принимать единственное значение для группы, т.е. оно может быть либо значением столбца, указанного в GROUP BY, либо арифметическим выражением, включающим это значение, либо константой, либо одной из SQL-функций, которая оперирует всеми значениями столбца в группе и сводит эти значения к единственному значению (например, к сумме).

Фраза HAVING играет такую же роль для групп, что и фраза WHERE для строк: она используется для исключения групп, точно так же, как WHERE используется для исключения строк. Эта фраза включается в предложение лишь при наличии фразы GROUP BY, а выражение в HAVING должно принимать единственное значение для группы.

Лабораторная работа №1. Создание структуры базы данных и заполнение таблиц

Цель работы

Знакомство с программным продуктом OpenOffice.org Base. Создание структуры учебной базы данных. Заполнение таблиц актуальными данными.

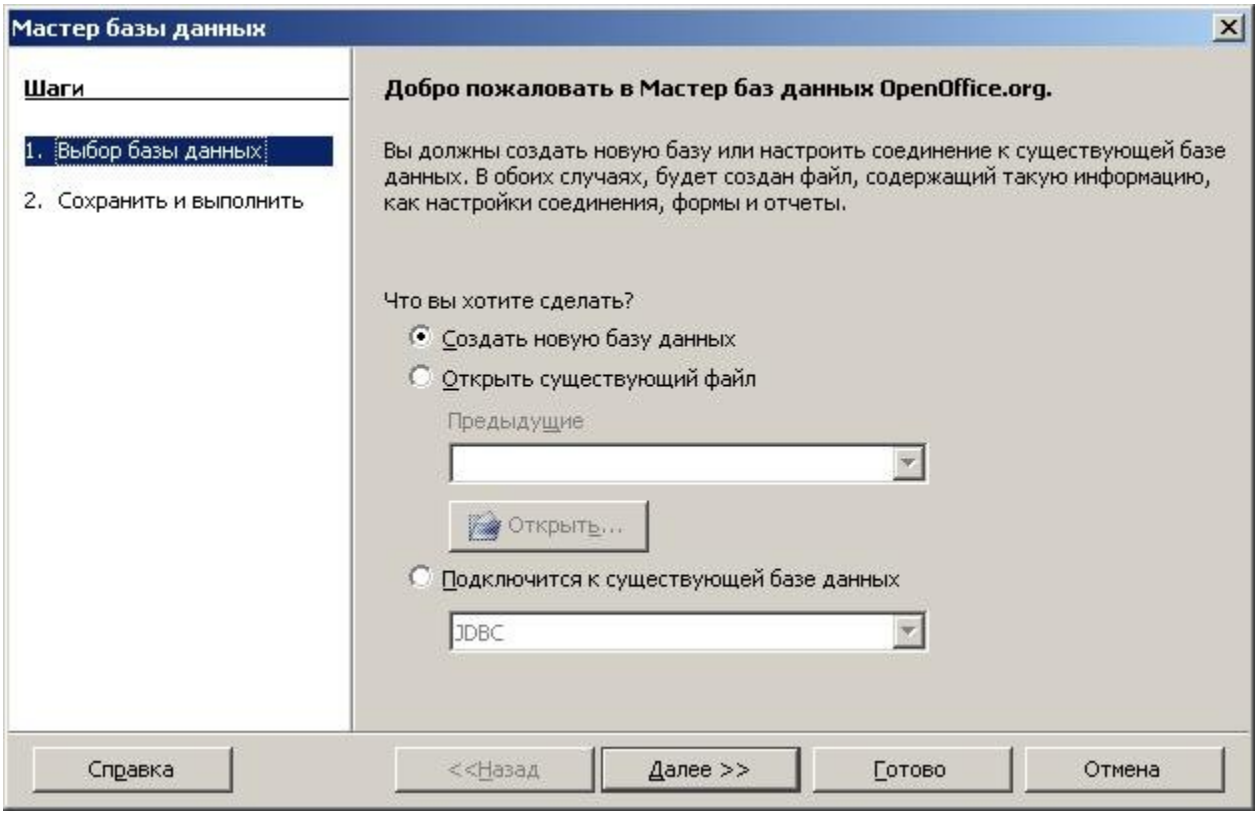
Задание на лабораторную работу

Создайте таблицы students, groups, prepods описанные выше.

Порядок выполнения работы

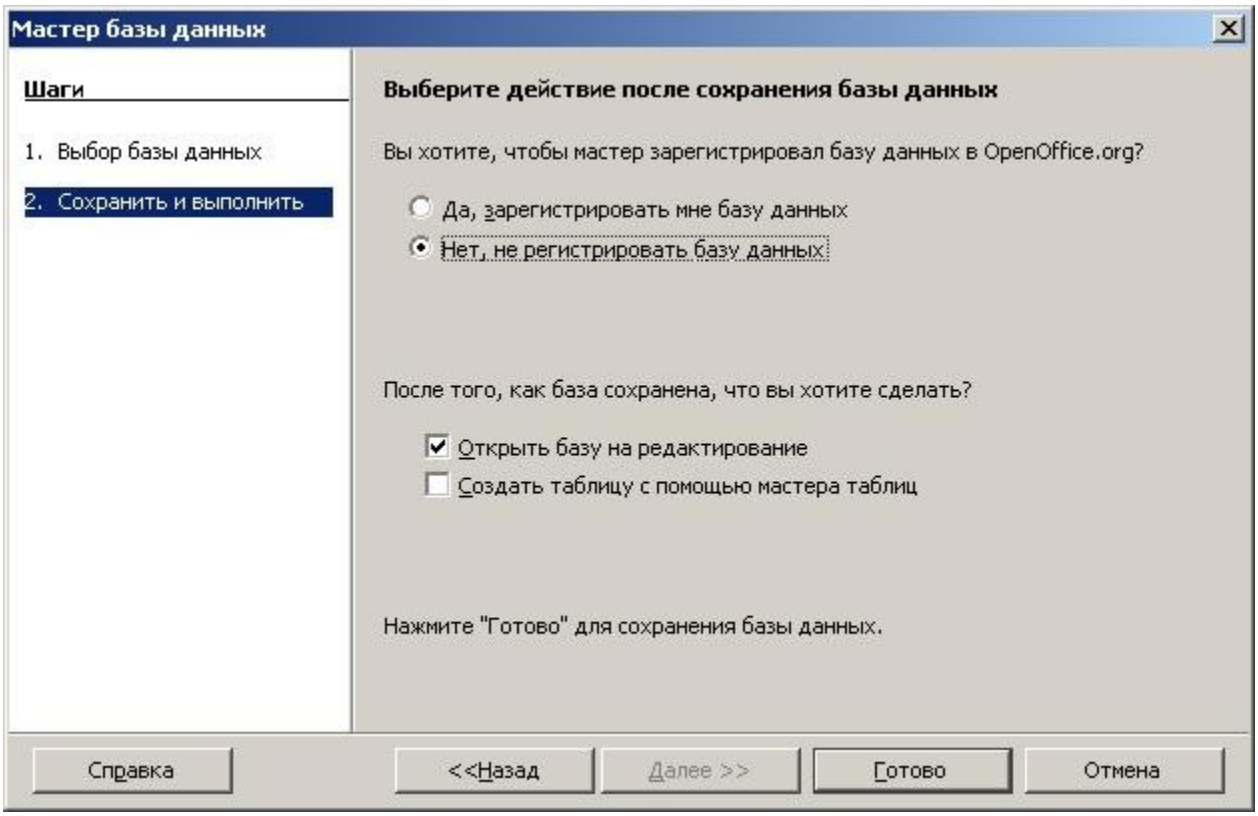
Запустите программу OpenOffice.org Base. Вы увидите диалоговое окно в котором Мастер базы данных предложит Вам создать новую или открыть существующую базу данных.

Выберите пункт «Создать новую базу данных» и нажмите кнопку «Далее»



Очевидно, что в следующий раз, когда БД будет уже создана, Вы должны выбирать пункт «Открыть существующий файл». Кроме работы с собственным форматом БД, Base умеет общаться и с иными СУБД. Для этого используются высокоуровневые интерфейсы JDBC и ODBC.

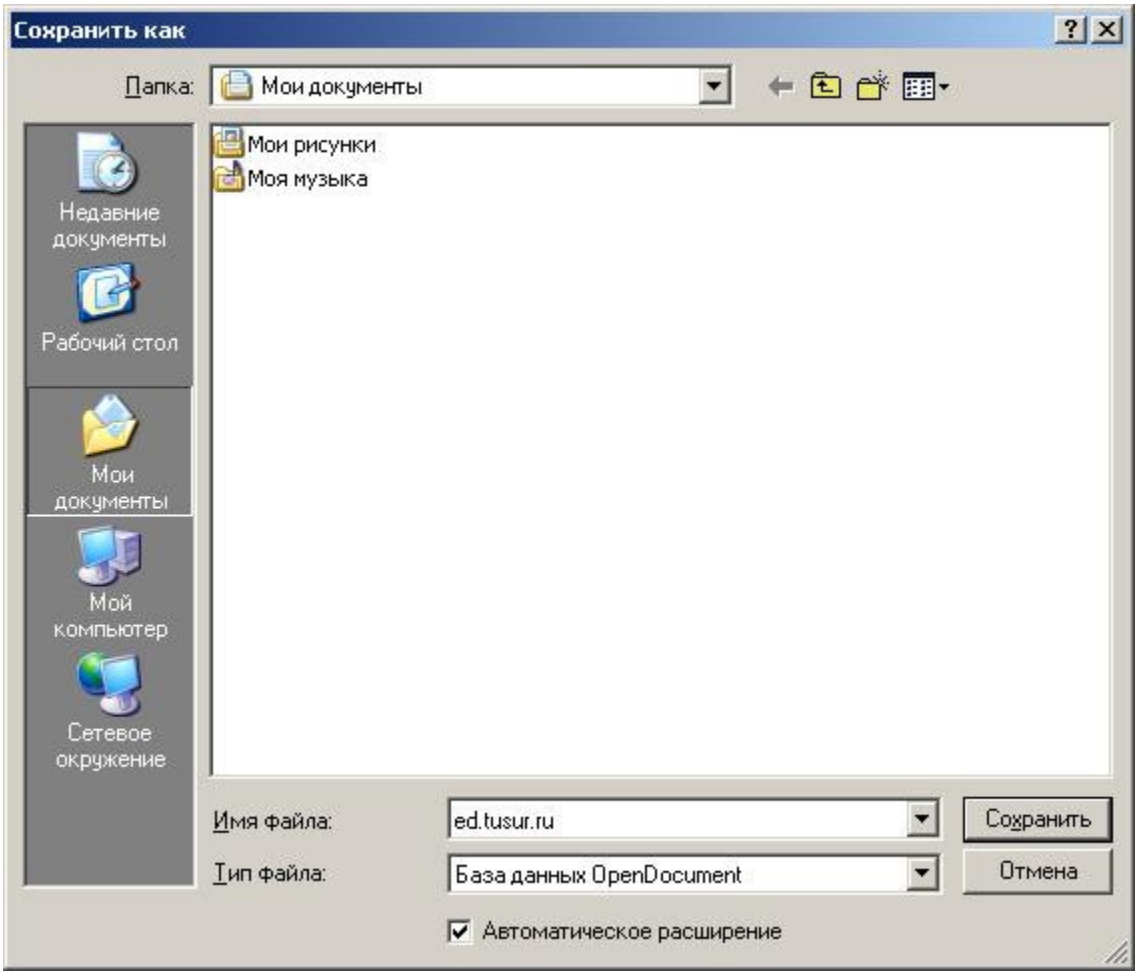
Перед Вами появится следующее диалоговое окно:



Не забудьте поставить галочку около пункта «Открыть базу на редактирование» и нажмите кнопку «Готово».

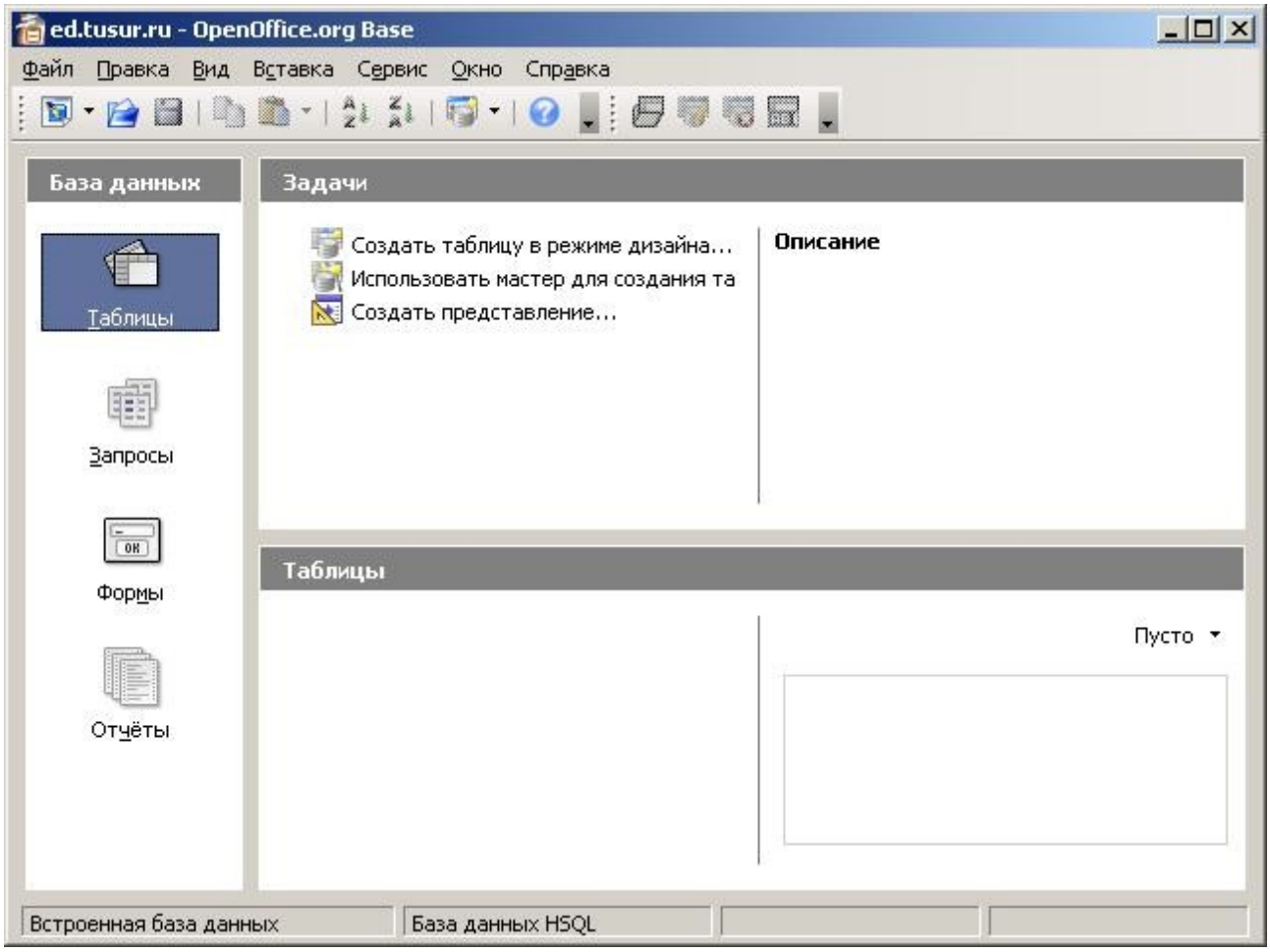
Вообще говоря, Base может помочь Вам в создании необходимых таблиц с помощью мастера, но в рамках данного учебного курса мы будем создавать таблицы самостоятельно, «руками».

Система предложит Вам выбрать для Вашей БД имя и сохранить ее на диске для дальнейшей работы. Введите в качестве имени БД «ed.tusur.ru».



Сохраните вновь созданную БД в Вашу рабочую папку. Это совсем не обязательно должна быть папка «Мои документ». Для корректного сохранения данных, проконсультируйтесь с системным администратором.

В конце концов перед Вами появится открытое окно программы Base. Выглядит оно примерно так:



Как видно из вышеприведенного рисунка Base предлагает нам несколько основных функций работы с СУБД. Сгруппированы функции следующим образом:

- **Таблицы** – в данном разделе пользователь может производить различные операции с таблицами, входящими в БД. В том числе создавать, изменять структуру, вносить новые данные, править старые и удалять ненужные записи.

  Важное замечание. К вопросу удаления записей необходимо подходить очень ответственно. Вообще говоря, идеально спроектированная БД (под

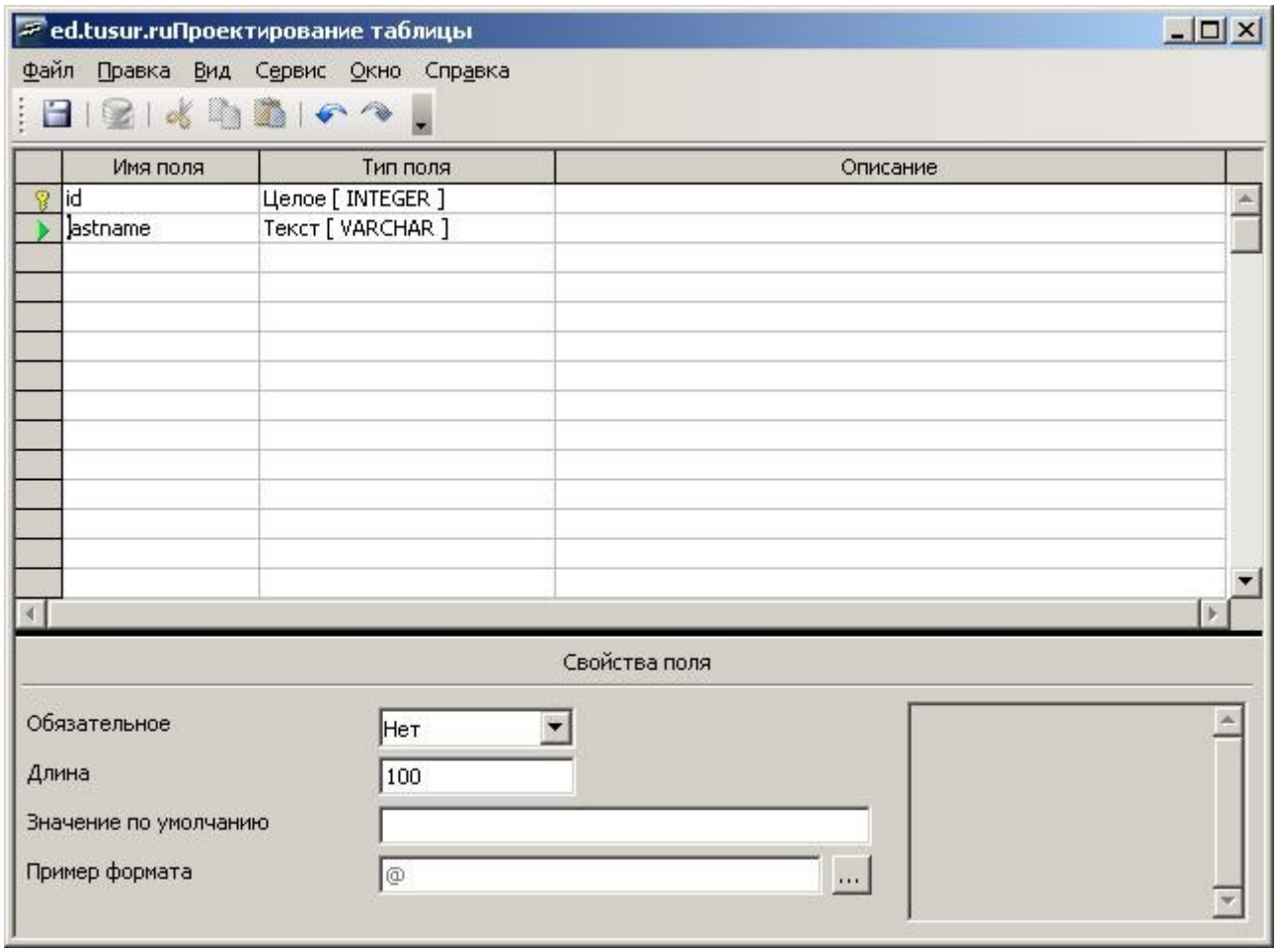
проектом понимается и структура и набор прикладных программ) дает возможность удаления записей только в исключительных случаях.

- **Запросы** – данный раздел предназначен для создания, изменения, выполнения и удаления различных запросов к нашей БД. Запросы могут быть оформлены как в канонической SQL- так и в QBE-форме (QBE – Query By Example – запрос по примеру). В рамках наших занятий мы будем иметь дело только с SQL-запросами.
- **Формы** – раздел предназначен для работы с формами БД. Формы являются необходимым элементом для эффективной работы с любой реляционной БД, поскольку позволяют отображать данные из нескольких таблиц связанных между собой посредством реляционных отношений. Формы позволяют пользователям просматривать данные в удобной форме а также предназначены для заполнения таблиц БД.
- **Отчеты** – схожий с формами элемент для работы с БД. Основным отличием от форм является то, что отчеты не позволяют изменять данные, но зато дают дополнительные возможности по представлению данных при выводе на экран или на печатающее устройство вывода.

Фактически, на первой лабораторной работе мы будем иметь дело только с разделом «Таблицы».

Для создания таблицы необходимо выбрать пункт «Создать таблицу в режиме дизайна» в блоке «Задачи» окна Base.

Перед Вами появится примерно такое окно:

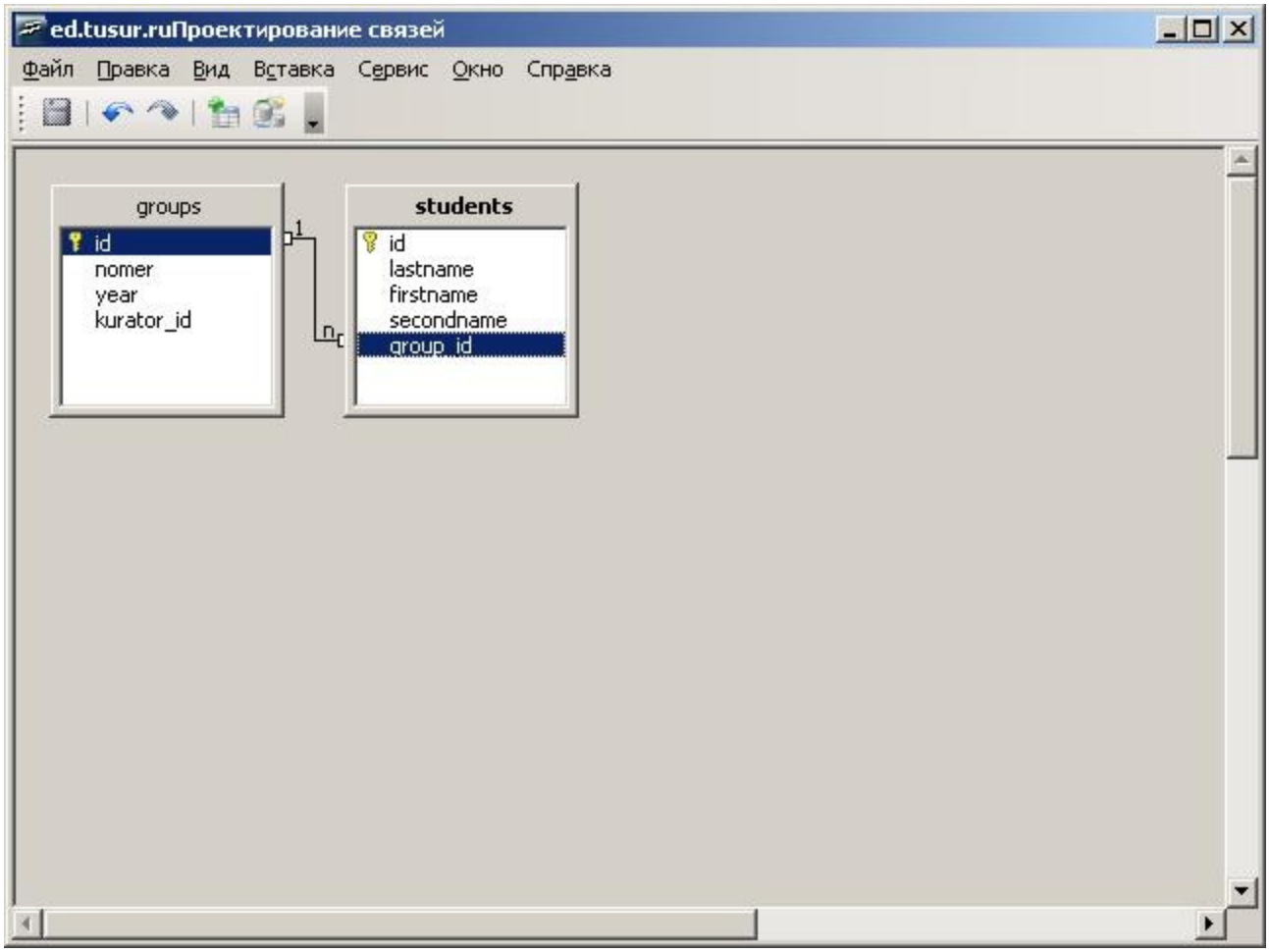


Создайте последовательно таблицы groups, students, prepods и subjects. Не забудьте определить для таблиц первичные ключи. Особое внимание обратите на соответствие типов в полях, ссылающихся на первичные ключи других таблиц.

Как было сказано выше, реляционная БД – это совокупность связанных таблиц. Связь, как было отмечено, производится с помощью ссылок на первичные ключи. В программе Base для определения связей служит пункт меню «Сервис» - «Связи».

При выборе данного пункта меню появляется окно «Проектирование связей».

Следующий пример демонстрирует создание связи между таблицами students и groups:



Создайте по примеру все необходимые связи между таблицами.

После завершения процесса создания всех таблиц и определения связей между ними приступайте к заполнению их актуальными данными. Целесообразно начинать заполнение данных с таблицы groups поскольку ее поля не ссылаются на первичные ключи других таблиц.

По результатам работы необходимо подготовить отчет в котором привести структуру таблиц, изображение окна проектирования связей и содержимое всех таблиц.

Лабораторная работа №2. Создание форм

Цель работы

Создание необходимых форм для учебной БД «Студенты кафедры ЭП» в среде OpenOffice.org Base. Произведение базовых операций над данными.

Задание на лабораторную работу

Создайте формы «Группы», «Студенты», «Преподаватели» и «Учебные курсы».

Порядок выполнения работы

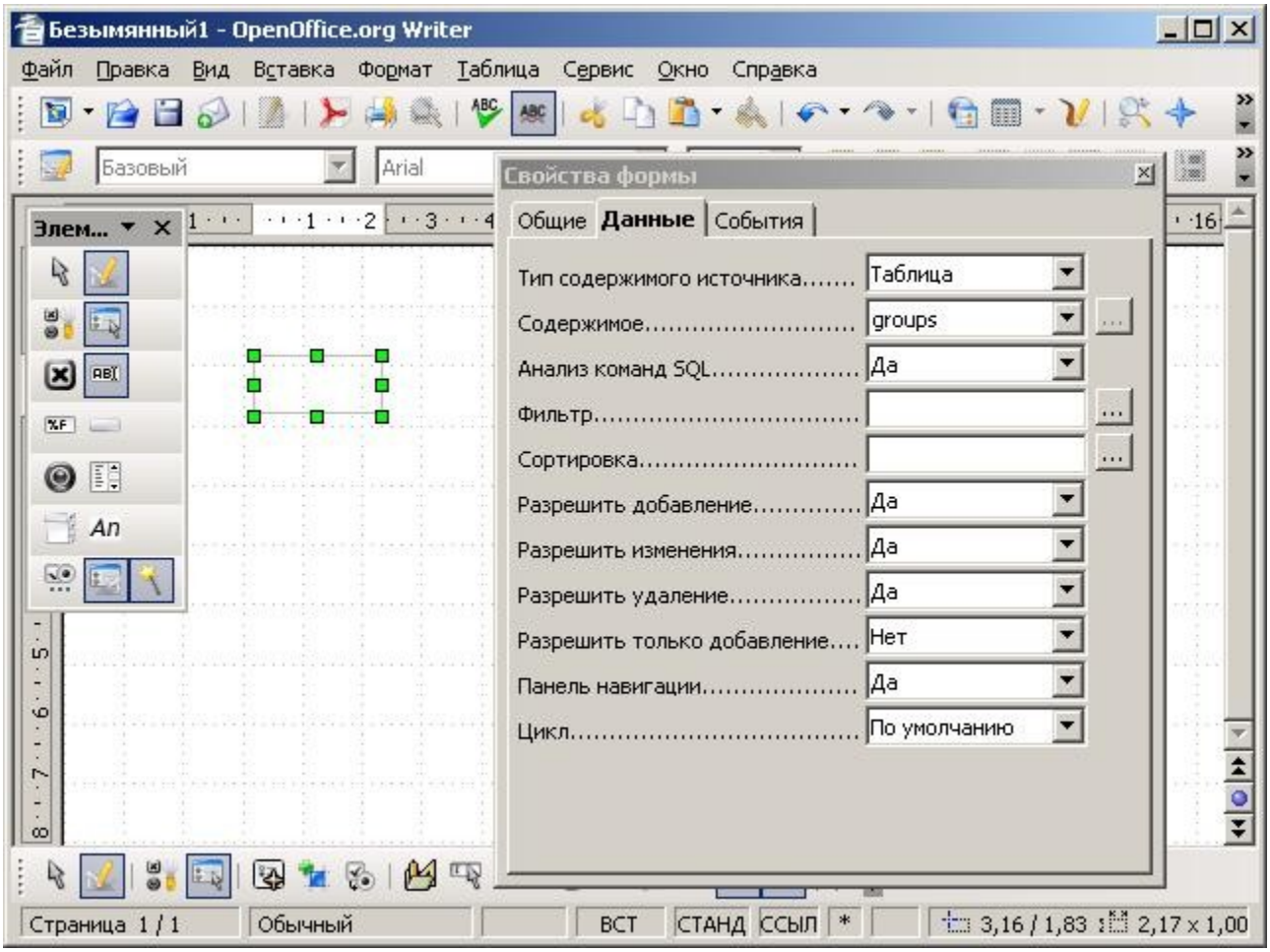
Как уже отмечалось выше, для работы с реляционными базами данных наилучшим средством являются формы и отчеты. Они позволяют визуализировать связанные данные и, таким образом, обеспечить прозрачный доступ к данным таблиц.

Для создания формы воспользуйтесь пунктом “Создание формы в режиме дизайна”.

Для создания поля ввода в форме воспользуйтесь инструментом (слева) “Текстовое поле”.

Для определения на базе какой таблицы создается форма (откуда будут браться данные и куда будет производится запись новых данных) воспользуйтесь инструментом “Свойства формы”.

Чтобы связать поле ввода данных с полем таблицы воспользуйтесь инструментом “Элемент управления”.



Замечание. Важно не забывать о наличии в форме тех полей таблицы, которые являются первичными ключами. Поскольку мы не создаем никакой управляющей программы для нашей системы, задача обеспечения уникальности первичных ключей лежит полностью на нас.

Понятно, что в случае создания прикладной программы, мы могли бы переложить функцию генерации первичных ключей на нее.

Также следует ни в коем случае не использовать встроенные в СУБД функции автоматической генерации значений первичного ключа. Это конечно сделано для упрощения жизни пользователя, но в реальной практике таких механизмов следует старательно избегать. Происходит это по многим причинам, о чем лучше поговорить на лекции.

Создавать формы, указанные в задании следует в том порядке в котором они перечислены.

При сохранении формы целесообразно не заниматься творчеством, а использовать в качестве имен те, которые перечислены в задании.

Форма “Группы” должна включать в себя 3 текстовых поля для ввода id, nomer и year и выбор куратора группы по фамилии из выпадающего списка. Список должен быть сформирован на основе данных из таблицы prepods.

Аналогичным образом необходимо создать и остальные формы. Для формы “Студенты” выбор группы должен осуществляться с помощью выпадающего списка, для формы “Преподаватели” выпадающий список не нужен поскольку таблица prepods не содержит внешних ключей. Форма “Учебные курсы” должна содержать выпадающий список для выбора преподавателя (по фамилии).

После завершения процесса создания форм необходимо внести с их помощью нескольк записей во все таблицы.

В отчете необходимо привести внешний вид всех получившихся форм.

Лабораторная работа №3. Выполнение простых запросов к БД

Цель работы

Отработка навыков и умений при использовании языка запросов SQL. Формирование простых запросов к учебной базе данных.

Задание на лабораторную работу

Создать и выполнить простые запросы к базе данных по нижеприведенному списку.

Порядок выполнения работы

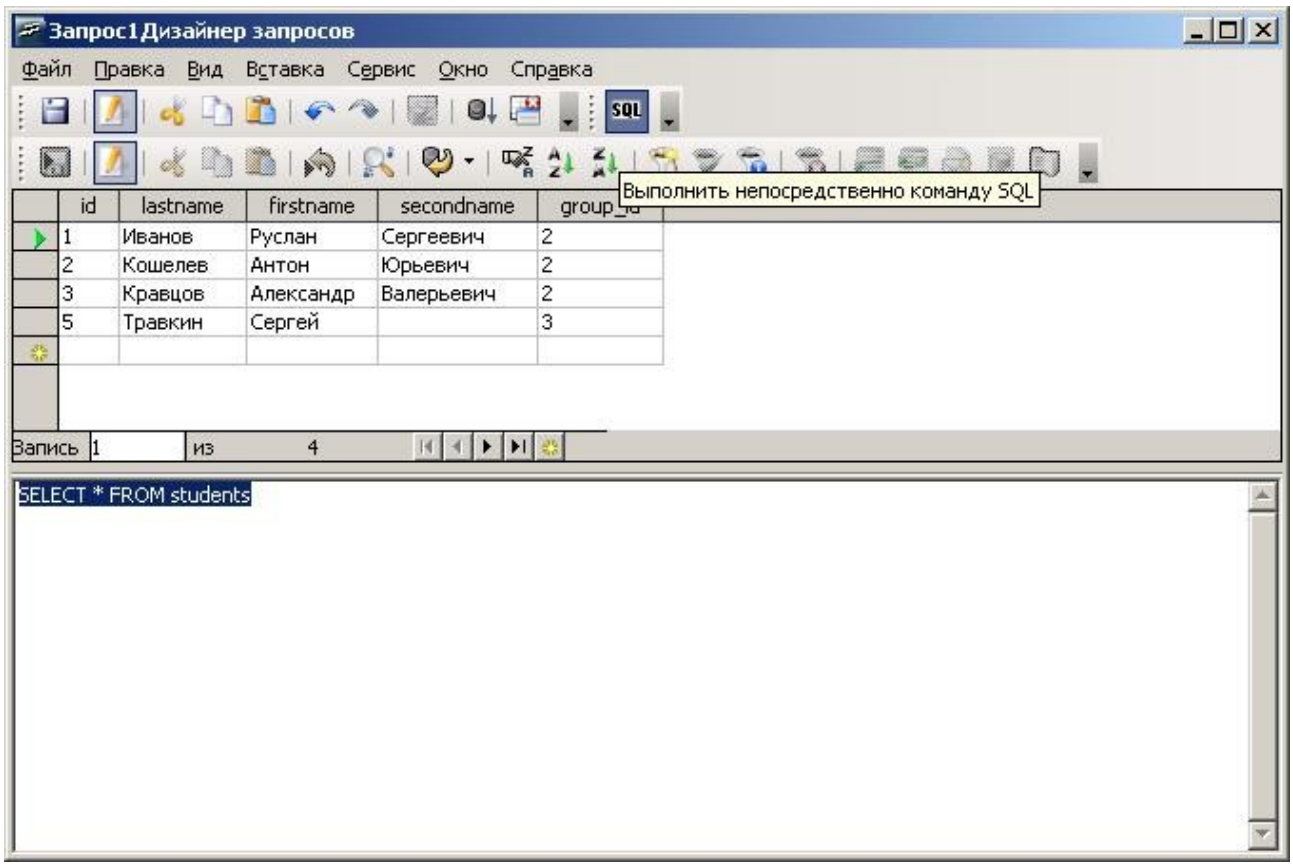
Для подготовки и выполнения запросов на языке SQL в программном продукте Base предназначен функциональный раздел “Запросы”.

Выберите раздел “Запросы”.

Выберите в данном разделе пункт “Создать запрос в SQL представлении”

В появившемся текстовом редакторе наберите текст запроса на языке SQL. Для выполнения запроса нажмите кнопку со значком SQL. Проконтролируйте результат.

Созданные запросы необходимо сохранить в разделе БД “Запросы”



Список запросов к выполению.

Получить список всех студентов из таблицы students

Результат должен иметь следующий заголовок:

Фамилия	Имя	Отчество

Получить список всех групп из таблицы groups

Результат должен иметь следующий заголовок:

Номер группы	Год поступления

Получить список всех студентов из таблицы students с именем Ольга

Результат должен иметь следующий заголовок:

Фамилия	Имя	Отчество

Получить список всех студентов из таблицы students с со значением поля group_id равным 3 или 2

Результат должен иметь следующий заголовок:

Фамилия	Имя	Отчество	group_id

Получить список всех студентов разбитых (отсортированных) по группам из таблиц students и groups (связать две таблицы)

Результат должен иметь следующий заголовок:

Фамилия	Имя	Отчество	Номер группы

Получить список всех студентов группы 351-2 (связать две таблицы)

Результат должен иметь следующий заголовок:

Фамилия	Имя	Отчество	Номер группы

Получить список тудентов группы 351-2 с именем Александр

Результат должен иметь следующий заголовок:

Имя	Фамилия	Номер группы

По итогам выполения лабораторный работы необходимо подготовить отчет. В отчете привести тексты всех выполеннных запросов и результирующие таблицы.

Лабораторная работа №4. Расширение возможностей учебной базы данных

Цель работы

Расширение возможностей учебной базы данных. Закрепление навыков работы с продуктом.

Задание на лабораторную работу

Дополнить учебную БД таблицами subjects и marks, создать формы необходимые для внесения информации об учебных курсах и оценках. Заполнить вновь созданные таблицы с помощью форм.

Порядок выполнения работы

Создать структуру таблицы subjects. Данная таблица предназначена для хранения информации об учебных курсах кафедры ЭП. Поскольку одним из полей таблицы subjects является prepod_id, ссылающееся на соответствующий первичный ключ таблицы преподавателей prepods, заполнение целесообразно производить с помощью формы. В данной форме необходимо сделать выпадающий список, ссылающийся на поле Фамилия в таблице преподавателей. Таким образом, заполнение поля prepod_id таблицы subjects будет производится путем выбора из списка фамилии преподавателя, который ведет данный курс.

Заполнить таблицу учебных курсов subjects актуальной информацией. Поскольку солздаваемая нами БД является учебной целесообразно ограничить количество курсов числом 6-7.

Создать структуру таблицы оценок студентов marks. Данная таблица имеет два поля, ссылающихся на соответствующие первичные ключи таблиц subjects и students. Таким образом, корректное заполнение данной таблицы возможно только с помощью соответствующей формы. Форма должна

содержать в себе два выпадающих списка и одно текстовое поле “Оценка”. Поле “Оценка” может принимать текстовые значения “Отлично”, “Хорошо”, “Удовлетворительно”, “Плохо”, “Зачтено”.

Подготовка отчета по лабораторной работе

В отчете по лабораторной работе необходимо привести внешний вид получившихся форм и заполненных таблиц.

Лабораторная работа №5. Выполнение усложненных запросов к учебной базе данных

Цель работы

Отработка навыков подготовки и выполнения усложненных запросов к БД, в том числе запросов с использование агрегирующих функций.

Задание на лабораторную работу

Создать и выполнить простые запросы к базе данных по нижеприведенному списку.

Порядок выполнения работы

Для подготовки и выполнения запросов на языке SQL в программном продукте Base предназначен функциональный раздел “Запросы”.

Выберите раздел “Запросы”.

Выберите в данном разделе пункт “Создать запрос в SQL представлении”

В появившемся текстовом редакторе наберите текст запроса на языке SQL. Для выполнения запроса нажмите кнопку со значком SQL. Проконтролируйте результат.

Созданные запросы необходимо сохранить в разделе БД “Запросы”

Список запросов к выполению.

Получить количество студентов в каждой группе

Результат должен иметь следующий заголовок:

Номер группы	**Количество студентов**
352-2	12

Получить количество студентов с одинаковыми именами

Результат должен иметь следующий заголовок:

Имя	**Количество**

Андрей	6

Получить количество студентов с именем Андрей

Результат должен иметь следующий заголовок:

Имя	**Количество**

Подготовка отчета по лабораторной работе

В отчете по лабораторной работе необходимо привести текст подготовленных запросов и результаты их выполнения.

Лабораторная работа №6. Создание отчетов

Приложение 1

Федеральное агентство по образованию

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра ЭЛЕКТРОННЫХ ПРИБОРОВ (ЭП)

дисциплина «Системы управления базами данных»

ОТЧЕТ

по лабораторной работе

«_____»

Выполнил студент

гр. 356-1

XXXXXXXXXXX

Проверил преподаватель

_____

_____

2006