МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Кафедра информатики и математики

КУРСОВАЯ РАБОТА

РАЗРАБОТКА БАЗЫ ДАННЫХ ВЕДЕНИЯ РЕЕСТРА ИМУЩЕСТВА СТУДЕНЧЕСКОГО ГОРОДКА

Работу выполнил_____ Е.И. Голубов
(подпись, дата) (инициалы, фамилия)
Филиал ФГБОУ ВО «КубГУ» в г. Новороссийске курс 2 ОФО
Направление 38.03.05 «Бизнес-информатика»

Научный руководитель
Канд.физ.-мат.наук, доцент _____ С.В. Дьяченко
(подпись, дата) (инициалы, фамилия)

Нормоконтролер
Специалист по УМР_____О.П. Гринчишина
(подпись, дата) (инициалы, фамилия)

Краснодар 2017

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель данной курсовой работы является приобретение практических навыков в процессе изучения и разработки базы данных.

В современном мире существует огромное множество разнообразных баз данных, без которых век информационных технологий не носил бы подобного названия и не смог бы прогрессивно развиваться. Современный мир не может обойтись без структурированной и отсортированной информации, а базы данных помогают сэкономить время в этом непростом занятии. Базы данных необходимы для многих областей деятельности человека, будь то банковское дело, продовольственный магазин или же учет домашних расходов.

Базы данных встречаются на каждом шагу. Практически любая система это хорошо построенная база данных, например всем известный «Консультант +» или же учет единого налога в министерстве налогов и сборов.

Задачи курсовой работы состоят в следующем:

1) Создать базу данных.

2) Разработать пользовательский интерфейс.

В настоящее время многие современные языки программирования, поддерживают программирование баз данных, с помощью таких языков можно создать необходимую базу данных, и неважно, будет ли она простой или очень сложной.

Базы данных особенно актуальны в современном мире, ведь хранение и обработка информации оказывает влияние на эффективность принятия решений в производственной деятельности, экономике и политике. Поскольку в современном обществе информационные технологии являются его неотъемлемой частью, базы данных необходимы в качестве прочной основы для обеспечения структурированной информацией.

1 Средство разработки базы данных Microsoft Access

1.1 Основная характеристика Microsoft Access

Microsoft Access – это система управления базами данных (СУБД), относящейся к реляционному типу. Пожалуй, главным достоинством данной средой разработки баз данных является простой, интуитивно понятный графический интерфейс [5].

Как и принято среди реляционных СУБД, весь набор данных в Access распределяется по разным таблицам, но, при этом, хранится в одном файле, что является существенным отличием Access от его аналогов. Речь идёт не только об информации, которая содержится в таблицах, но и о других объектах баз данных, которые описываются далее.

Почти все основные операции Access могут быть выполнены при минимальных затратах человеческих сил за счёт, так называемых, «Мастеров» («Wizards»): их внушительное количество обеспечивает ощутимую экономию времени всех, без исключения, пользователей, особенно – незнакомых с программированием.

В рамках сети с файловым сервером, а также в локальной одноранговой сети возможна разработка многопользовательской базы данных Access, в которой доступ к ней могут получить одновременно несколько пользователей. Сама сеть отвечает за поддержку аппаратного характера и программными средствами обеспечивает обмен данными между компьютерами [11]. Access контролирует границы доступа различных пользователей, тем самым обеспечивая защищённость данных. Присутствуют определённые ограничения по осуществлению многопользовательской работы с Access по причине того, что данная СУБД не является серверной. Как правило, на файловый сервер загружается файл Access, с присущим ему расширением *.mdb, чтобы другие пользователи могли удалённо получить доступ к данным, используя сеть. Однако обработка данных происходит непосредственно там, где запущено

приложение, то есть на основном клиенте, что резко ограничивает круг пользователей до двадцати человек, а при особенно больших объёмах данных, содержащихся в таблицах – и вовсе до пятнадцати, поскольку нагрузка на сеть значительно возрастает.

Если рассматривать Access со стороны поддержания целостности данных, то он соответствует моделям базы данных средней сложности. Отсутствие таких средств как, например, триггеры и хранимые процедуры вынуждает разработчиков перекладывать ответственность за поддержание бизнес логики на клиентское приложение.

Касательно разграничения доступа к информации и её защиты в целом – Access не обладает достаточно надёжными средствами для осуществления данных функций на высоком уровне. Присутствует лишь стандартная защита с применением пароля, устанавливаемого пользователем, но данный барьер не способен остановить соответствующего специалиста.

При этом, несмотря на все перечисленные недостатки у Microsoft Access имеется ряд неоспоримых преимуществ по сравнению подобными СУБД.

Доступность Access – безусловно, одно из ключевых достоинств, которым данный продукт обязан своей принадлежности компании Microsoft. Microsoft, в свою очередь, предоставляет операционную систему и программное обеспечение внушительной части пользователей персональных компьютеров. Полная совместимость с операционной системой Windows, регулярные обновления и поддержка со стороны производителя, возможность импорта и экспорта данных в формат Excel, – другой популярной программы от Microsoft, – а также поддержка множества языков являются слагаемыми популярности Access.

У Access нет чёткого ориентира на определённую категорию пользователей: люди с совершенно разной профессиональной подготовкой могут справиться с поставленной задачей, благодаря наличию ранее упомянутых «Мастеров», развитой системе справок и доступному для человека с базовым

уровнем владения персональным компьютером интерфейсу. Всё это ощутимо ускоряет процесс проектирования и создания базы данных, а также облегчает выборку данных из неё.

Microsoft Access предоставляет возможность обойтись без разработки запросов на языке SQL и не прибегать к языку VBA для программирования модулей или макросов, предлагая взамен самые разные диалоговые средства [17].

В целом, в Microsoft Access достаточно сильно развиты встроенные средства разработки приложений. Основная масса приложений, которые распространяются среди пользователей, в том или ином объёме содержит код VBA (Visual Basic for Applications). Построение команд SQL в процессе работы программы, использование Windows API, работа с переменными, обработка ошибок – все эти и многие другие стандартные задачи выполняются в Access при помощи VBA.

Язык макрокоманд – одно из средств программирования в Access. Макросы это программы, созданные на данном языке. Они используются для построения связей между определёнными действиями, которые реализуются при помощи форм, запросов и отчётов. Пользователь может управлять макросами при диалоговой работе с данными через формы, своими действиями вызывая системные события, которые и влияют на макросы [24].

Таким образом, обладая чертами, присущими СУБД, Access не ограничивается её возможностями. Он позволяет разрабатывать приложения, работающие с базами данных, не отменяя, при этом, того факта, что сам Access – простая и в освоении, и в использовании система управления базами данных.

1.2 Функциональные возможности Microsoft Access

Познакомимся с базовыми функциями Microsoft Access, чтобы прояснить аспекты, связанные с его возможностями.

Итак, в Access 2010 содержатся следующие типы объектов: таблица, запрос, форма, отчёт, макрос, модуль. Данный пример представлен на рисунке 1.

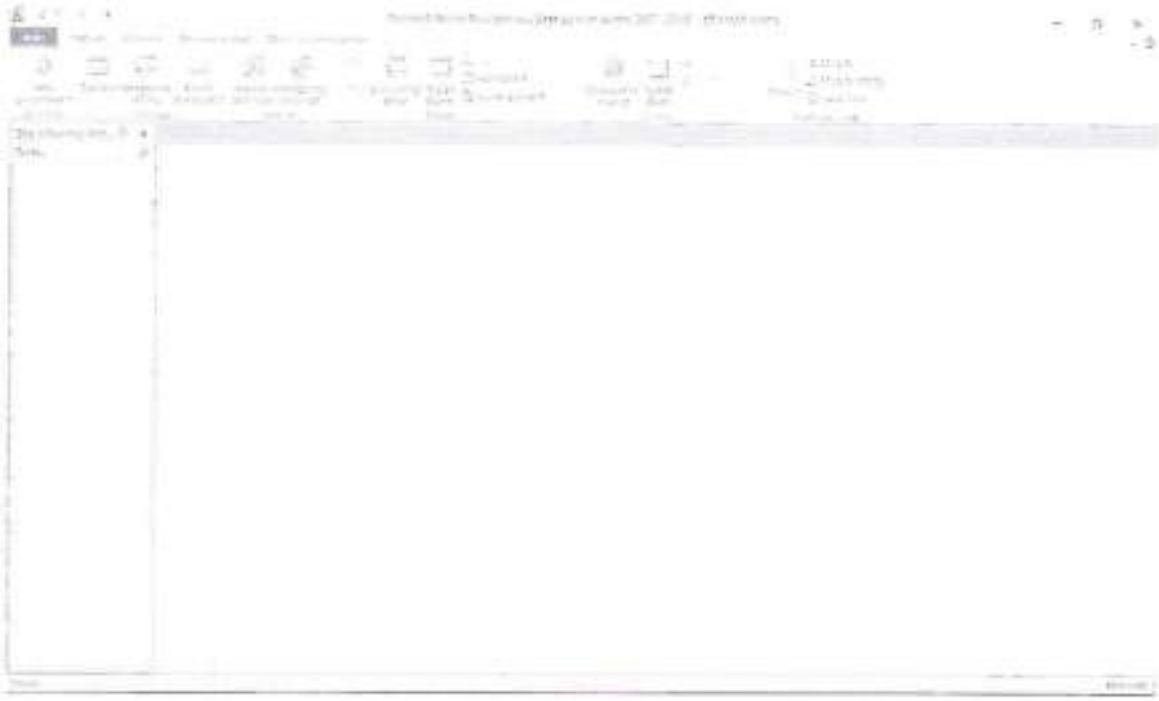

Рисунок 1 – Окно объектов базы данных

Работа Access может осуществляться лишь с одной базой данных одновременно [23]. При этом одна база данных Access способна содержать в себе множество таблиц, запросов, отчётов, форм, макросов, а также модулей. Все эти данные сохраняются и держатся в одном файле расширения mdb.

Таблица является объектом базы данных, где данные представлены в виде записей, – то есть, строк, – а также полей, – то есть, столбцов. Это главный структурный элемент в системе управления реляционной базой данных [7]. Для ускорения доступа к данным в каждой таблице задаётся первичный ключ, который задаётся по нажатию на кнопку «Ключевое поле», представленную на рисунке 2.

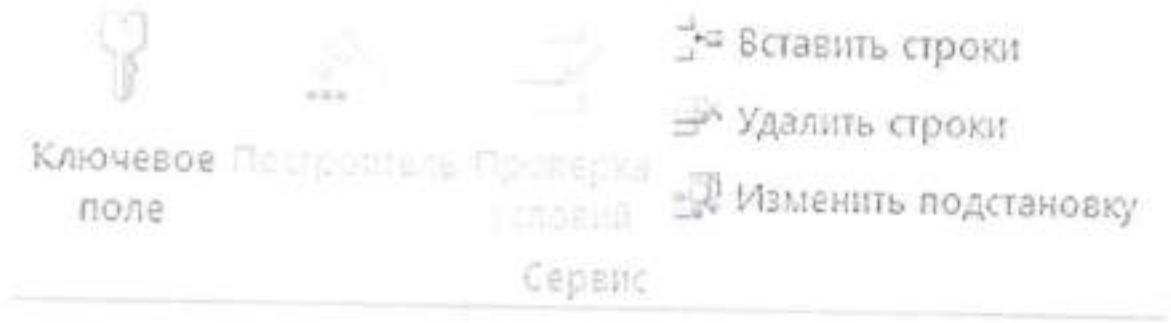

Рисунок 2 – Ключевое поле

Но для того, чтобы его задать, требуется для начала создать структуру самой таблицы. Сделать это можно тремя способами: войдя в режим конструктора, выбрав один из предложенных шаблонов или просто введя данные в пустую таблицу [16]. Выбор в пользу того или иного способа напрямую связан с целью и уровнем подготовки пользователя, а также с планами по использованию данных. Рассмотрим подробнее каждый из имеющихся вариантов:

1) Режим конструктора. Для того чтобы войти в режим «Конструктора», достаточно перейти на вкладку «Создание» и в группе «Таблицы» выбрать «Конструктор таблиц». В открывшемся окне можно задать имя поля под соответствующей надписью и присвоить ему определённый тип данных [9]. Полный инструментарий режима «Конструктора» представлен на рисунке 3.

Рисунок 3 – Конструктор таблиц

2) Шаблоны. С целью экономии времени пользователей, слабо владеющих персональным компьютером, существует обширный набор шаблонов готовых таблиц с полями, готовых для заполнения [8]. Чтобы ими воспользоваться, необходимо перейти в меню «Файл» и выбрать пункт «Создать», после чего откроется выбор всех доступных шаблонов. Это продемонстрировано на рисунке 4.

Рисунок 4 – Шаблоны

3) Пустая таблица. Чтобы создать пустую таблицу, нужно лишь во вкладке «Создание» выбрать «Таблица» в соответствующей группе. Всё, что далее требуется от пользователя, это ввести названия полей и перейти к их заполнению, а Access сам определит тип данных в зависимости от содержимого [20], что наглядно продемонстрировано на рисунке 5.

Рисунок 5 – Работа с «пустой таблицей»

Говоря о типах данных, необходимо раскрыть суть основных из них [13]:

1) Текстовый. В поле содержится текст с количеством символов до 255.

2) Поле Memo. Предназначено для содержания длинных блоков текста, например, описания чего-либо. В нём может помещаться до 64000 символов.

3) Числовой. Поле, содержащее числовое значение.

4) Дата/ время. В данном поле содержится дата и время, начиная с 100, и заканчивая 9999 годом.

5) Денежный. Поле, в котором обозначаются валюты. Можно записать 15 знаков до и 4 знака после запятой.

6) Счётчик. Производится автоматическая вставка последовательных или случайных чисел по мере добавления новых записей.

7) Логический. Здесь содержатся значения «Да», «Нет», а также прочие поля, содержащие лишь одно истинное из двух возможных значений.

8) Поле объекта OLE. Позволяет хранить данные разных типов, причём одновременно, что устраняет главное ограничение реляционных баз данных – невозможность хранить данные сразу нескольких типов в одном поле.

9) Гиперссылка. В нём может храниться как текст, так и его комбинация с числами, сохранённая как текст. Используется как контейнер для хранения пути к какому-либо объекту.

Предусмотрена возможность просмотра, редактирования, удаления старых и добавления новых записей. Также можно сортировать данные и видоизменять таблицы. Представить все имеющиеся связи в графическом виде позволяет средство под названием «Схема данных», который можно увидеть на рисунке 6.

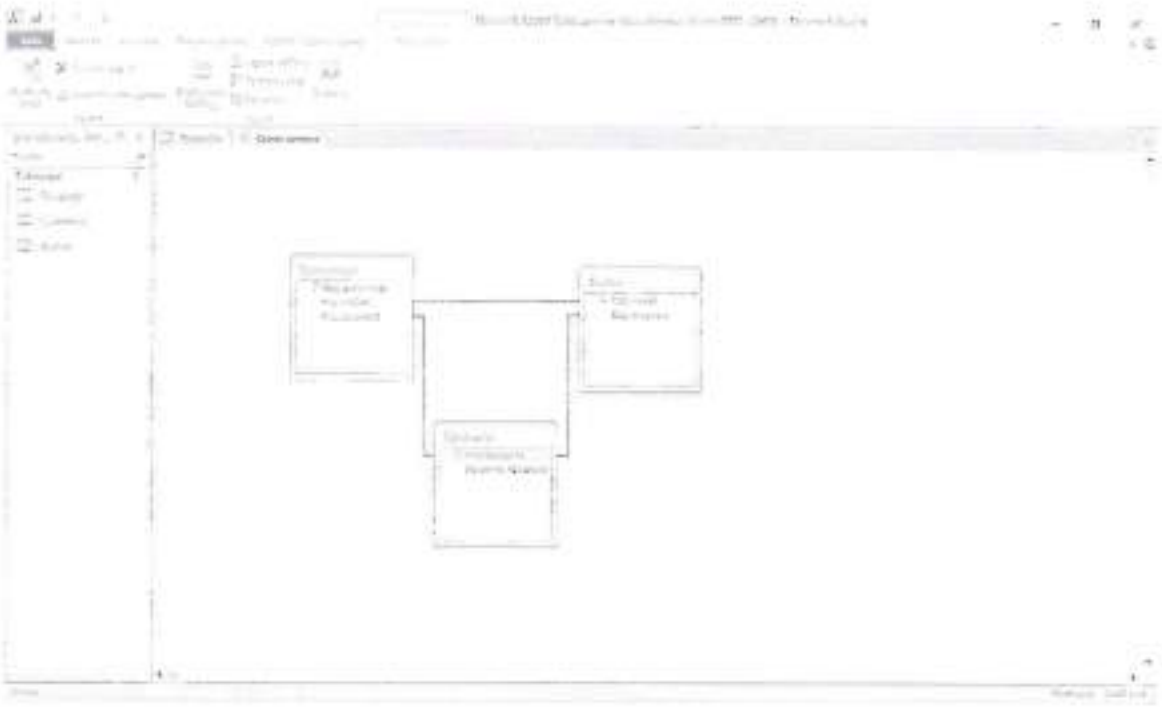

Рисунок 6 – Схема данных

Запрос это объект, в котором хранится текст SQL-запроса, содержащий индивидуальное имя в определённой базе данных. Создать запрос, воспользовавшись «Мастером запросов» или «Конструктором запросов» – решение целиком и полностью зависящее от пользователя. Если сделан выбор в пользу первого варианта, то после нажатия на соответствующую иконку пользователю будут предложены несколько типов запросов: простой, перекрёстный, на повторяющиеся записи, а также на записи без подчинённых. И если с первым запросом всё понятно, то оставшимся трём требуется дать определение. В перекрёстных запросах исходные данные обрабатываются статистическим образом, после чего выводятся результаты в виде таблицы, которая, по сути, являет собой сводную таблицу данных из Excel [15]. Во избежание вывода повторяющихся записей можно произвести поиск дубликатов, чему и способствует запрос на повторяющиеся записи. Вывести запрос, который покажет, например, всех клиентов, оставшихся без заказа, поможет запрос на записи без подчинённых. Во всех случаях от пользователя требуется лишь выбрать таблицу и поле для выборки данных в ней. Поскольку интерфейс во

всех представленных случаях почти одинаков, в качестве примера показано создание только простого запроса в режиме «Мастер запросов» на рисунке 7.

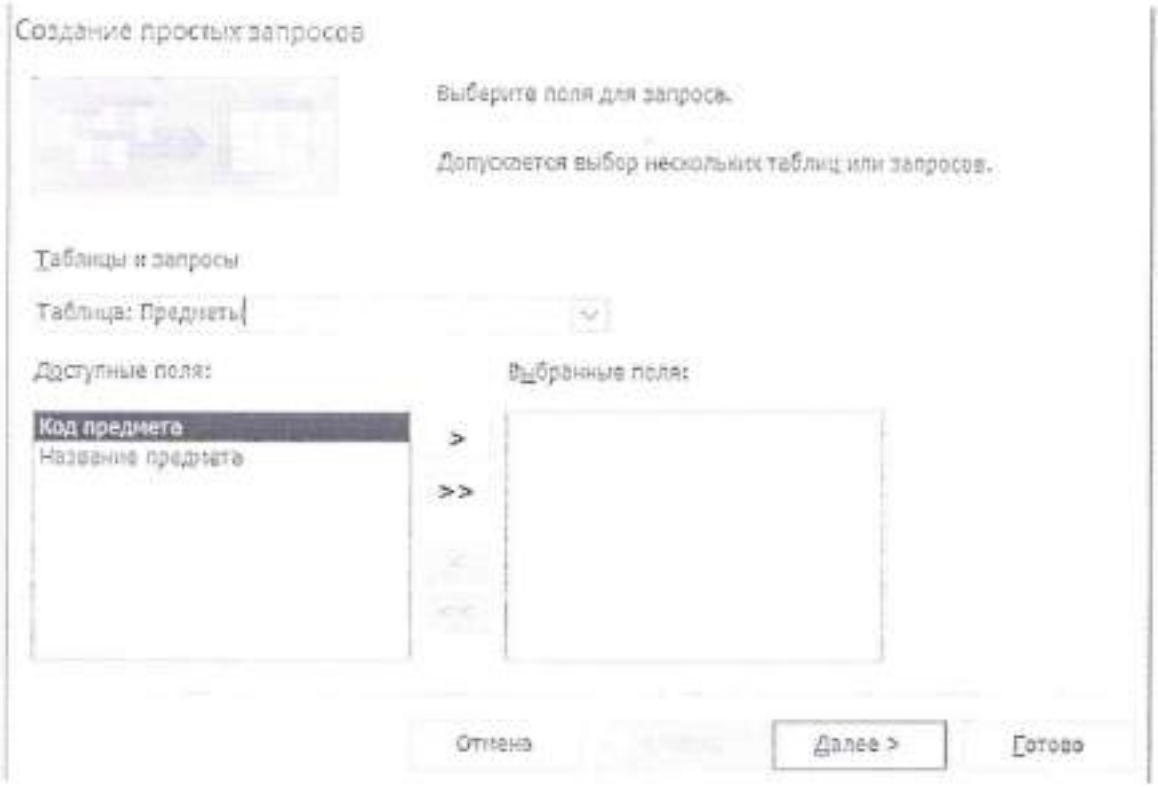

Рисунок 7 – Режим «Мастера запросов»

Режим «Конструктора запросов» позволяет пользователю не только выбрать таблицы, для удобства связать их графическим способом и определить поля, по которым будут выбираться данные, но и задать дополнительные условия и параметры сортировки для каждого из этих полей [12]. Все эти возможности отражены на рисунке 8.

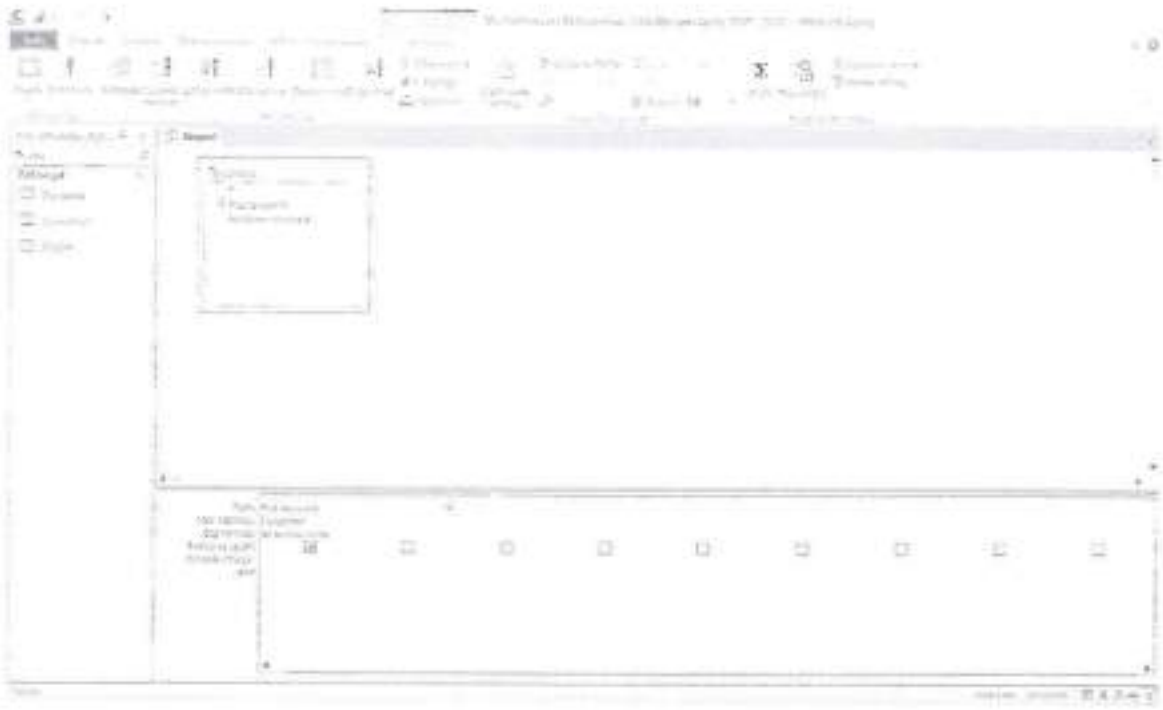

Рисунок 8 – Режим «Конструктора запросов»

Форма – особый контейнер для таких компонентов, связанных с интерфейсом, как поле для ввода данных, кнопки, поле для отображения данных из таблиц и прочее [1]. В зависимости от разрабатываемого приложения, разработчик волен расположить на форме компоненты, позволяющие просматривать, вводить, исправлять и группировать данные. Пример работы с формой представлен на рисунке 9.

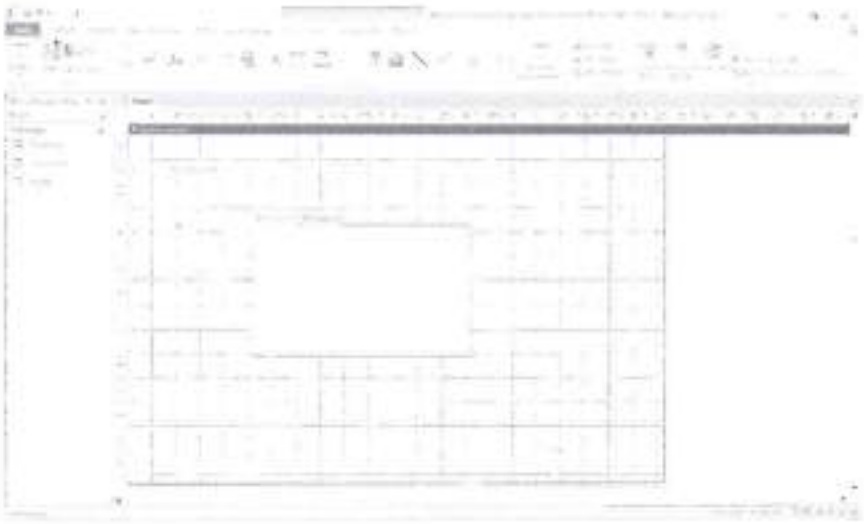

Рисунок 9 – Конструктор форм

Отчёт – объект, который применяется в случае, когда требуется произвести выборку из базы данных в структурированном виде, то есть оформить как таблицу или список, а затем создать из этого документ, который можно было бы распечатать или включить в документ другого приложения.

Страницы – инструмент для публикации данных в интернете или же в локальной сети. Создание страницы аналогично созданию формы, отличаются только используемые компоненты. Основное отличие при работе заключается в редактировании данных в базе, помимо возможности их просматривать. После сохранения страницы в качестве объекта базы данных возможен её экспорт в виде файла формата HTML, что позволяет получить доступ к данным, используя интернет-браузер.

Макрос – объект, который представляет собой упорядоченный набор макрокоманд с целью автоматизировать те действия, которые выполняются очень часто при работе с базой [18]. Макрокоманду можно выбрать в предложенном списке, в то время как параметры задаёт разработчик.

Модуль – контейнер программного кода, написанного на языке VBA. Редактор Visual Basic применяется для просмотра и редактирования кода, который находится в наборе модулей.

2 Delphi – как язык программирования и средство для работы с базами данных

2.1 Основные понятия, связанные с языками программирования

В данном пункте будут рассмотрены основные языки программирования, применяемые в клиентском приложении:

1) Изначально, у истоков создания языка программирования Delphi не была поставлена цель по обеспечению наиболее высокой производительности исполняемого кода для того чтобы экономить место ОЗУ (оперативного запоминающего устройства). С самого начала своего существования, язык ставил во главу стройность и высокую читабельность, поскольку, одной из задач данного языка являлось обучение программированию. Эта изначально заданная стройность, позволила языку в дальнейшем, при росте аппаратных мощностей, упростила расширение языка новыми функциями и конструкциями.

Delphi это язык программирования, используемый в среде с таким же названием, и является сочетанием некоторых важнейших технологий [2]:

– Простое быстрое и визуальное создание программ из программных прототипов.

– Компилятор, который имеет высокую производительность, в машинный код.

– Различные средства для создания и построения баз данных.

Первоначально язык программирования Delphi назывался Object Pascal. Далее начиная со среды Delphi 7.0, в различных официальных документах Borland начала использование названия Delphi для обозначения языка Object Pascal.

2) SQL (структурированный язык запросов) – это язык, предназначенный для управления базами данных для реляционных баз данных [4]. Как таковой SQL-язык не является Тьюринг-полным языком программирования,

но, однако, его стандарт дает возможность создавать для него различные процедурные расширения, которые помогают расширять его функциональность до полноценного и самостоятельного языка программирования.

2.2 База данных в Delphi

После того как база данных будет закончена в Microsoft Office Access, можно приступать к созданию клиентского приложения в Delphi.

Среда Delphi обладает большим количеством всевозможных функций, процедур, команд и инструментов которые упростят создание клиентского приложения. Самые основные инструменты, которые были применены при создании клиентского приложения, представлены ниже [10]:

1) TADOConnection. Предназначен для подключения базы данных к приложению.

2) TADOTable. Вступает в контакт с указанной пользователем таблицей базы данных.

3) TADOQuery. Предназначен для создания запросов.

4) TDataSource. Создан для связи сетки отображения данных, с самой базой данных.

5) TDBGrid. Предназначен для вывода информации из базы данных, в приложение.

Чтобы связать базу данных с приложением необходимо в компоненте TADOConnection выбрать свойство ConnectionString. После того как необходимый компонент будет выбран, откроется окно, которое показано на рисунке 10.

Рисунок 10 – Параметр ConnectionString

Для того чтобы связать базу данных, необходимо указать путь к базе данных, созданной в Microsoft Office Access. Для этого необходимо нажать на клавишу «Build...» [22], после чего откроет окно, представленное на рисунке 11.

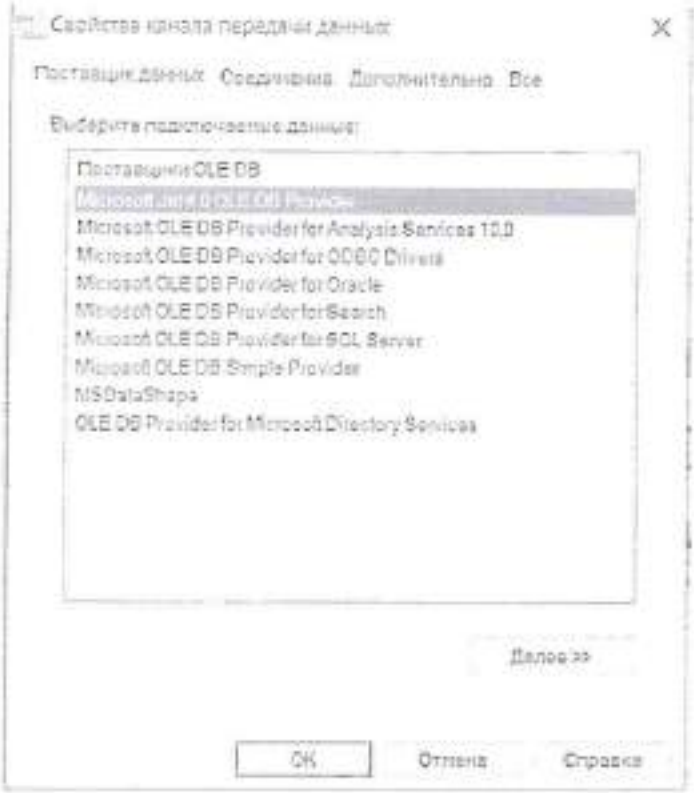

Рисунок 11 – Поставщики данных

В текущем окне необходимо выбрать свойство Microsoft Jet 4.0 OLE DB PROVIDER. После того как необходимое свойство выбрано, нужно установить соединение с заранее созданной базой данных. Это делает в следующем окне, представленном на рисунке 12.

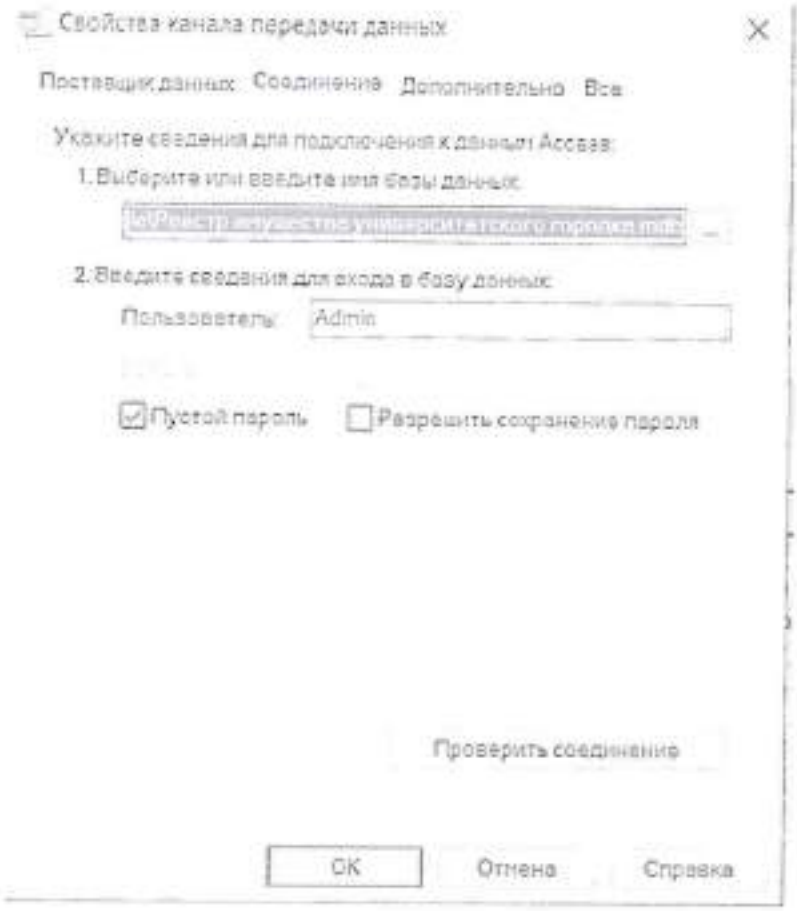

Рисунок 12 – Настройка соединения базы данных

При нажатии на клавишу «Проверить соединение» будет проведена проверка соединения приложения с выбранной базой данных. Если подключение выполнено верно, программа выдаст окно с надписью: «Проверка подключения выполнена».

После переноса всех необходимых компонентов на формы можно приступать к созданию, изменению, редактированию базы данных.

Компонент TADOQuery – аналог компонента TQuery, который используется при работе с BGE. TADOQuery применяется для создания и выполне-

ния SQL запросов. Основное свойство этого компонента – SQL, которое содержит методы выполнения запроса и, непосредственно, сам запрос [25]. TADOQuery точно так же, как и TQuery имеет свойство DataSource, которое, позволяет ему передать параметры запроса от одного компонента другому.

2.3 Программирование в Delphi

Embarcadero Delphi являет собой, прежде всего, инструмент для написания программ под операционную систему Windows. Сам по себе Delphi – очень простое и мощное средство, предназначенное для разработки программ, в основе которых лежит графический интерфейс, а также консольных приложений без графического интерфейса [19].

С помощью Delphi приложения можно разрабатывать не только для Windows, но и для Linux, для чего существует Borland Kylix. Это дает больше возможностей новые, а также увеличивает потенциально возможную отдачу от усилий, которые были вложены в изучение Delphi. В Delphi применяется кроссплатформенная библиотека компонентов CLX и визуальные дизайнеры для создания высокопроизводительных приложений для Windows, которые повторной компиляций можно легко превратить в приложения для Linux [21].

Delphi признан одним из ведущих языков программирования, которому присуща простая и понятная среда для написания различных программ и приложений. Delphi помогает преодолеть барьеры между сложными языками, находящимися на более высоком уровне, и языками, на более низком уровне.

При воссоздании графического интерфейса программ Delphi, у есть все возможности и преимущества языка программирования Object Pascal, "завернутого" в среду RAD (от английского rapid application development — быстрая разработка приложений) [3]. Такие компоненты окна графического пользовательского интерфейса, как формы, списки объектов, кнопки, уже вклю-

чены в состав приложения Delphi. Все это значительно упрощает работу по созданию программ и приложений. Не нужно писать никакой код, чтобы добавить их на форму. Требуется просто перенести их на Форму, как в простом графическом редакторе. Можно также внести на Форму элементы управления ActiveX, для создания в кратчайшие сроки специализированных приложений, например, таких, как веб-браузеры [14]. Всего несколько нажатий кнопки мыши отделяют разработчиков дизайна приложений, использующих Delphi, от внесения правок в интерфейс программы.

Delphi может предстать в большом количестве комбинаций из самых разнообразных конфигураций, в зависимости от версии, чтобы в полной мере отвечать потребностям различных пользователей.

Интерфейс Delphi состоит из большого количества составных частей. Из этого множества частей можно выделить самые основные:

1) Окно Редактора Исходного Текста.

2) Дизайнер Форм.

3) Инспектор Объектов.

4) Палитра Компонентов.

5) Справочник.

Конечно же, это не все составляющие Delphi. Есть и другие важные составляющие, например, системное меню, линейка инструментов и так далее.

Разберем каждую из этих частей более подробно:

1) Окно редактора исходного текста. Окно редактора объектов – безусловно, то место, в котором разработчики, использующие Delphi, проводят основную массу времени. Пример окна редактора исходного текста представлен на рисунке 13.

Рисунок 13 – Окно редактора исходного текста

Здесь программисты и пишут код.

2) Дизайнер форм. У дизайнера форм весьма простой принцип использования его возможностей, за счёт интуитивно понятной конструкции. Пример Дизайнера Форм показан на рисунке 14.

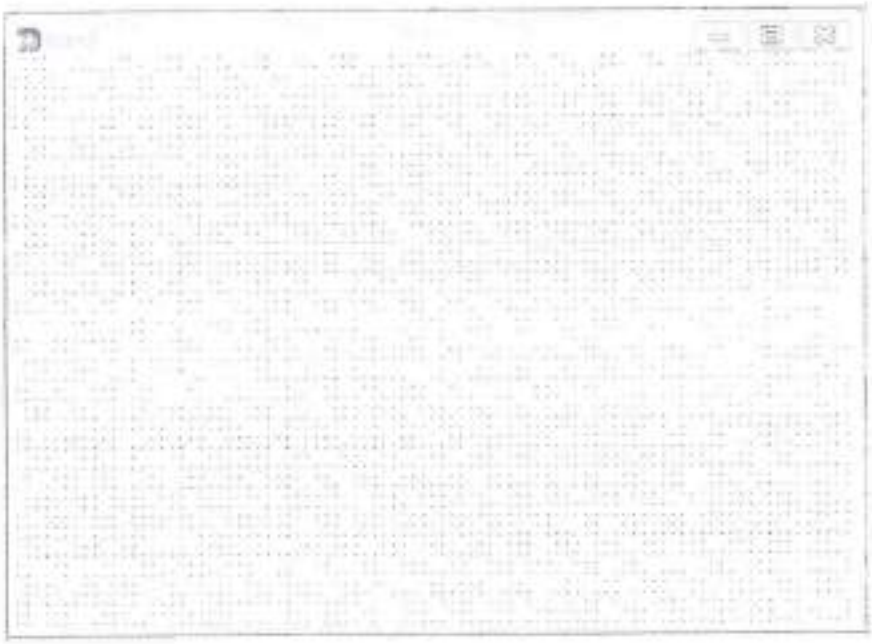

Рисунок 14 – Создание формы

Дизайнер Форм изначально состоит из одного пустого окна, которое можно заполнить большим количеством всевозможных объектов, которые можно выбрать на Палитре Компонентов.

3) Инспектор Объектов. Инспектор Объектов состоит из двух разделов. Каждый из этих разделов возможно применять для определения поведения этого компонента. Первая страница этого раздела - это список свойств, вторая страница — это список событий. Если необходимо редактировать что-либо, связанное с определенным компонентом, то это обычно делается через Инспектор Объектов [6]. Например, нужно изменить размер и имя компонента TLabel изменяя свойства Caption, Left, Top, Height, и Width и так далее. Пример Инспектора Объектов представлен на рисунке 15.

Рисунок 15 – Инспектор Объектов

Вторая страница — это страница событий. Она напрямую связана с Редактором; если дважды щелкнуть мышкой на правую сторону какого-нибудь подпункта, то код, который соответствует данному событию автоматически, запишется в Редактор, в этот момент, Редактор получит фокус, и сразу же появится возможность добавить код обработчика данного события.

4) Палитра Компонентов. «Палитра Компонентов» дает возможность программисту выбрать необходимые объекты для размещения их на Дизайнере Форм. Для использования «Палитры Компонентов» нужно лишь выбрать необходимый элемент для работы и перенести его на Дизайнер Форм или же просто первый раз щелкнуть мышкой на него и потом второй раз - на Дизайнере Форм. Выбранный объект появится на проектируемом окне и его можно изменять и редактировать с помощью мыши. Пример Палитры Компонентов представлен на рисунке 16.

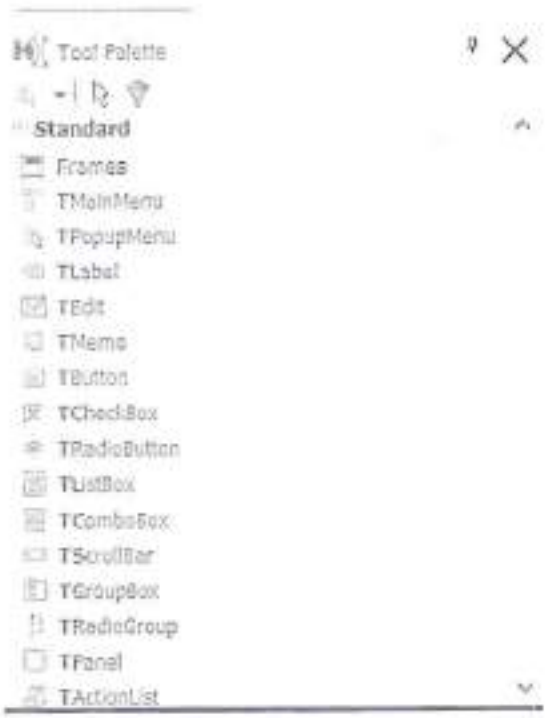

Рисунок 16 – Палитра компонентов

Палитра Компонентов использует постраничную группировку объектов. Внизу Палитры находится множество наборов закладок - Standard, Additional, Dialogs и т.д.

5) Справочник. Последняя, наиболее важная часть среды Delphi – это Справочник. Для того, чтобы получить доступ к данному инструменту необходимо выбрать в системном меню пункт Help и затем Contents. Тогда на экране появится необходимый справочник. Пример справочника показан на рисунке 17.

Рисунок 17 – Справочник

Справочник контекстно-зависим. То есть по нажатию клавиши F1 выводится справка, подсказывающая о текущем состоянии программы. Например, если находясь в Инспекторе Объектов, выбрать какое-либо свойство и нажать F1, то появится справка о назначении данного свойства.

3 Реестр имущества студенческого городка

3.1 Разработка базы данных в Microsoft Access

Чтобы спроектировать базу данных для регистрации имущества университетского городка, нужно для начала определить, какие данные будут в ней содержаться. Контейнерами для исходных данных служат таблицы, название которых даёт характеристику находящимся внутри данным. Таким образом, в базе данных имеются следующие таблицы:

1) Здания.

2) Идентификаторы.

3) Комнаты и аудитории.

4) Предметы.

5) Студенты.

6) Факультеты.

Кратко раскроем суть каждой из них. В таблице «Здания» присутствуют всего три поля: код здания, название и адрес. Стоит отметить, что код для какого-либо наименования, то есть первое поле в любой таблице, является, как правило, ключевым. Именно с ним будет в дальнейшем устанавливаться связь. В «Названии» определяется тип здания, в данном случае их всего два: университет и общежитие. Столбец «Адрес» содержит название города, улицы, а также номер дома. Попутно необходимо рассмотреть тип данных, устанавливаемых для каждого поля, что наглядно демонстрируется в таблице 1.

Таблица 1 – Типы полей таблицы «Здания»

Название поля	Тип данных
Код_здания	Счётчик
Название	Текстовый
Адрес	Текстовый

Далее рассмотрим таблицу «Комнаты и аудитории». В ней присутствуют опять же три поля: код комнаты, номер комнаты и код здания. Значение первого поля было уже описано выше, но теперь будет лучше понятно его применение. Итак, первые четыре строки соответствуют порядковым номерам комнат в общежитии, а последние три – аудиториям «101», «102» и «103» соответственно. Во избежание путаницы и существует поле с кодом здания, чтобы можно было легко определить, в каком здании находится определённая комната. Типы данных для этой таблице указаны на таблице 2.

Таблица 2 – Типы полей таблицы «Комнаты_и_аудитории»

Название поля	Тип данных
Код_комнаты	Счётчик
Номер_комнаты	Текстовый
Код_здания	Числовой

Следующая таблица – «Предметы». Здесь всё так же три поля: код предмета, название предмета и количество. Первое поле пригодится в таблице идентификаторов, о которой – чуть дальше по тексту. Во втором поле содержатся наименования всех имеющихся предметов: стол, стул, шкаф, кровать, тумбочка, парта, стул, компьютер, монитор, мышь, клавиатура и проектор. В третьем столбце, напротив каждого наименования, выводится количество определённого вида предмета. Типы данных таблицы «Предметы» продемонстрированы в таблице 3.

Таблица 3 – Типы полей таблицы «Предметы»

Название поля	Тип данных
Код_предмета	Счётчик
Название_предмета	Текстовый
Количество	Числовой

Таблица «Факультеты» представляет собой список имеющихся факультетов высшего учебного заведения. Имеется три поля: код факультета, наименование факультета и количество учащихся. Всего в представленной таблице три факультета: прикладная математика, экономика и юриспруденция. Типы данных обозначены в таблице 4.

Таблица 4 – Типы полей таблицы «Факультеты»

Название поля	Тип данных
Код_факультета	Счётчик
Наименование_факультета	Текстовый
Количество_учащихся	Числовой

В таблице «Студенты» содержится пять полей: код студента, полное имя студента, номер телефона, код факультета и код комнаты. Поле с номером телефона нужно, чтобы до любого студента, при необходимости, можно было дозвониться, поле с кодом факультета – чтобы всегда знать, на каком факультете студент проходит обучение, а поле с кодом комнаты – чтобы знать, где он проживает. Типы данных указаны в таблице 5.

Таблица 5 – Типы полей таблицы «Студенты»

Название поля	Тип данных
Код_студента	Счётчик
ФИО_студента	Текстовый
Номер_телефона	Текстовый
Код_факультета	Числовой
Код_комнаты	Числовой

Таблица «Идентификаторы» является связующим звеном между всеми таблицами. В ней содержится четыре поля: идентификатор, код предмета,

код здания и код комнаты. Поле с идентификаторами обозначает инвентарный номер предмета и позволяет вести учёт каждого вида предмета. Типы данных представлены в таблице 6.

Таблица 6 – Типы полей таблицы «Идентификаторы»

Название поля	Тип данных
Идентификатор	Счётчик
Код_предмета	Числовой
Код_здания	Числовой
Код_комнаты	Числовой

Более ясному пониманию взаимосвязи всех таблиц способствует «Схема данных», представленная на рисунке 18.

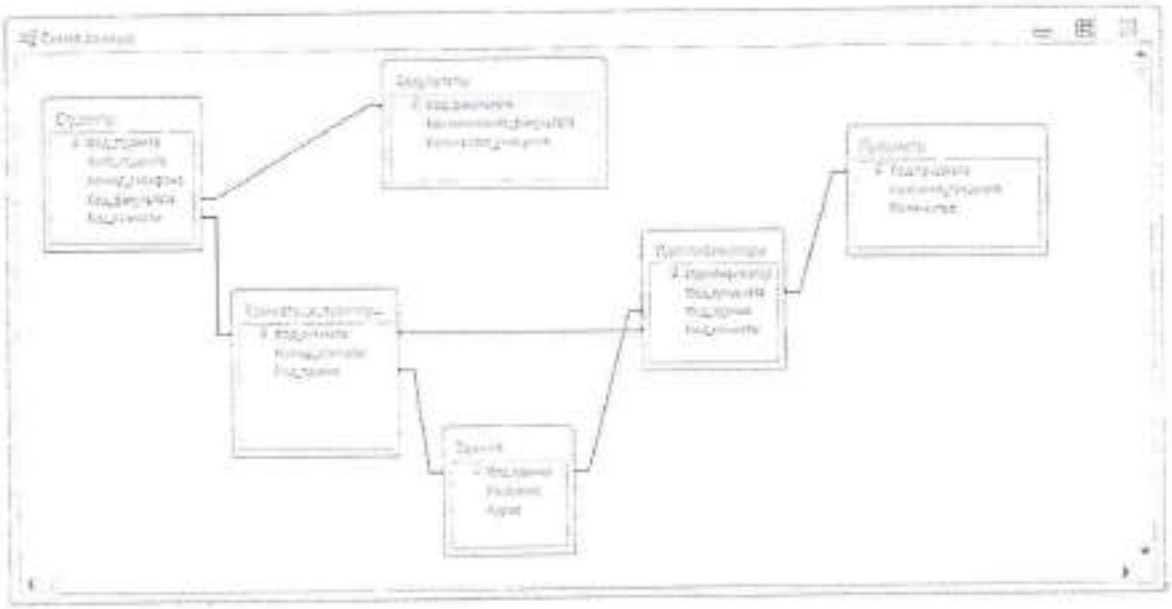

Рисунок 18 – Схема связи таблиц

Данная схема наглядно демонстрирует все связи, а также помогает при создании запросов. Особо выделяется таблица «Идентификаторы», в которой присутствует самое большое, по сравнению с другими таблицами, количество связей.

3.2 Создание клиентского приложения в Delphi

Прежде чем приступить к разработке самого приложения, нужно указать путь к базе данных. Требуется создать DataModule, на котором будут размещены шесть объектов TADOTable, к ним – ещё шесть TDataSource (для имеющихся шести таблиц из базы данных), один объект TADOQuery вместе с ещё одним TDataSource (для создания запросов), а также TADOConnection, в котором и будет указан путь к базе данных. В итоге, DataModule выглядит как на рисунке 19.

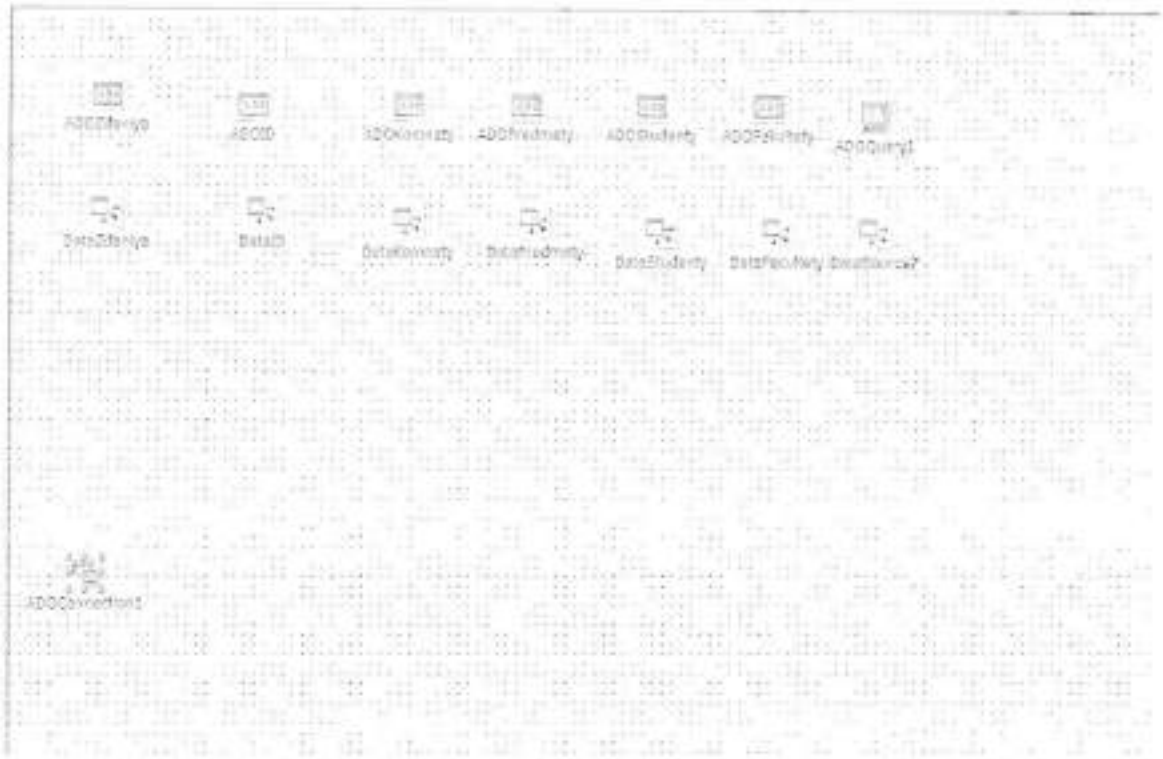

Рисунок 19 – Все объекты в DataModule

О том, как указать путь к базе данных, достаточно подробно написано в предыдущей главе. После этого нужно выделить один из ADOTable и в «Инспекторе объектов» в разделе «Linkage» выбрать ADOConnection в свойстве «Connection», а затем в разделе «Database», нажав на пустое поле «TableName», выбрать в появившемся списке одну из таблиц из базы данных. Для удобства следует заменить имя «ADOTable1» на более удобное, в дан-

ном случае на «ADOZdaniya», что можно сделать, отредактировав в разделе «Miscellaneous» свойство «Name». Затем требуется выделить один из DataSource, и нажав на поле «DataSet» в разделе «Database», выбрать в появившемся списке «ADOZdaniya». Далее требуется проделать аналогичные действия для оставшихся пяти ADOTable и всех, кроме последнего, DataSource. После этого нужно выделить ADOQuery, нажать на свойство «SQL» в разделе «Database» и написать в появившемся окне «select from * предметы». В оставшемся DataSource нужно выбрать «ADOQuery1», точно так же, как ранее это было проделано с ADOTable.

Чтобы пользователи могли свободно получить доступ к нужной информации, необходимо сделать главное меню. Для этого нужно добавить на главную форму элемент под названием «TMainMenu». В нём нужно создать четыре подменю: справочник, запросы, помощь и выход. Через «Справочник» будет реализован доступ ко всем таблицам, а о запросах будет сказано далее. Результат продемонстрирован на рисунке 20.

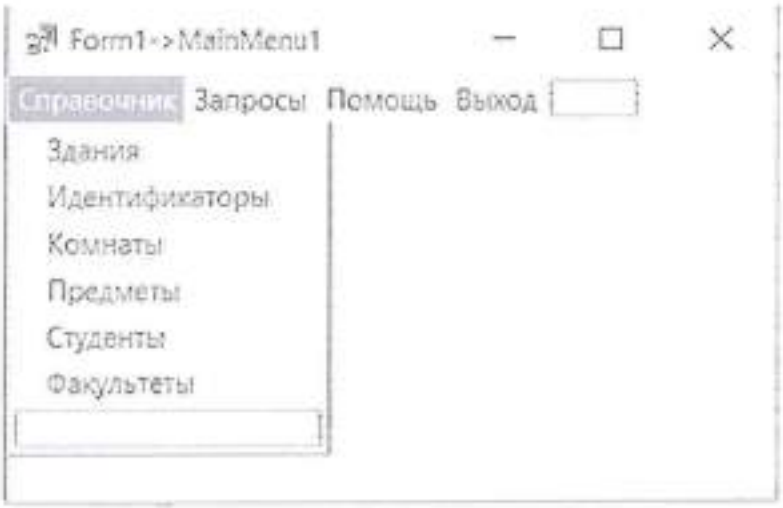

Рисунок 20 – Главное меню

Для отображения таблиц требуется создать к ним формы. На форме располагается элемент под названием «TDBGrid». В разделе «Linkage» в поле «DataSource» нужно выбрать в развернувшемся списке «DataStudenty».

Остальные формы делаются по такому же принципу. Как будет выглядеть получившаяся форма, показано на рисунке 21.

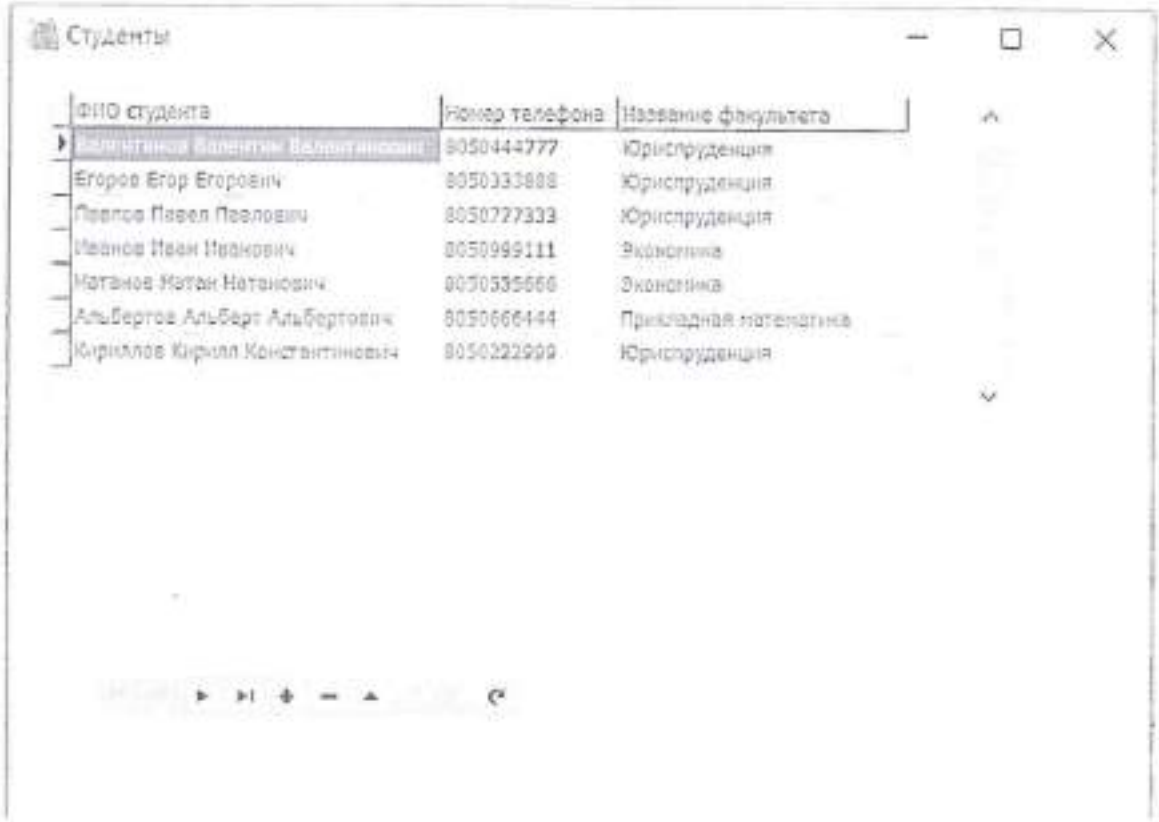

Рисунок 21 – Форма «Студенты»

Запросы помогают лучше ориентироваться в информации и быстрее получать необходимые сведения. Первый запрос под названием «Студент-Комната» помогает понять, в какой комнате проживает тот или иной студент. Запрос представлен на рисунке 22.

Рисунок 22 – Запрос «Студент-Комната»

Код запроса:

```
dbgrid1.Visible:=true;
datamodule2.adoquery1.close;
datamodule2.adoquery1.sql.clear;
datamodule2.adoquery1.sql.Add('Select Студенты.ФИО_студента as [ФИО студента], Комнаты_и_аудитории.Номер_комнаты as [Номер комнаты] from Студенты, Комнаты_и_аудитории where Студенты.Код_комнаты=Комнаты_и_аудитории.Код_комнаты');
 datamodule2.adoquery1.open;
form9.dbgrid1.Columns[0].title.alignment:=tacenter;
form9.dbgrid1.Columns[0].alignment:=tacenter;
form9.dbgrid1.Columns[0].Width:=245;
form9.dbgrid1.Columns[1].title.alignment:=tacenter;
form9.dbgrid1.Columns[1].alignment:=tacenter;
form9.dbgrid1.Columns[1].Width:=120;
form9.DBGrid1.Width:=400;
form9.dbgrid1.Height:=230;
```

Следующий запрос помогает посмотреть, сколько и какие именно студенты учатся на интересующем пользователя факультете. Чтобы его создать, понадобятся элементы «TDBGrid» и «TComboBox». В последнем пользователь выбирает наименование факультета, и в таблице ниже он видит всех студентов, которые числятся на данном факультете. Называется запрос «Факультет-Студент» и он представлен на рисунке 23.

Рисунок 23 – Запрос «Факультет-Студент»

Код запроса:

```
dbgrid1.Visible:=true;
datamodule2.adoquery1.close;
datamodule2.adoquery1.sql.clear;
str1:=combobox1.items.strings[combobox1.itemindex];
datamodule2.adoquery1.sql.Add('Select
Факультеты.Наименование_факультета, Студенты.ФИО_студента as [ФИО
студента] from Факультеты, Студенты where
Факультеты.Код_факультета=Студенты.Код_факультета and
Факультеты.Наименование_факультета=:p2');
datamodule2.adoquery1.parameters.paramvalues['p2']:=str1;
```

```
datamodule2.adoquery1.open;
dbgrid1.Columns[0].title.alignment:=tacenter;
dbgrid1.Columns[0].alignment:=tacenter;
dbgrid1.Columns[0].width:=0;
dbgrid1.Columns[1].title.alignment:=tacenter;
dbgrid1.Columns[1].alignment:=tacenter;
dbgrid1.Columns[1].Width:=250;
```

Следующий запрос обладает тем же набором элементов, что и предыдущий. Называется он «Студент-Предмет», и его суть сводится к закреплению ответственности студента за определёнными предметами. Пользователь выбирает в списке нужного ему студента, а в таблице ниже появляются все предметы, за которые студент несёт ответственность, причём с указанием инвентарного номера. Пример работы данного запроса продемонстрирован на рисунке 24.

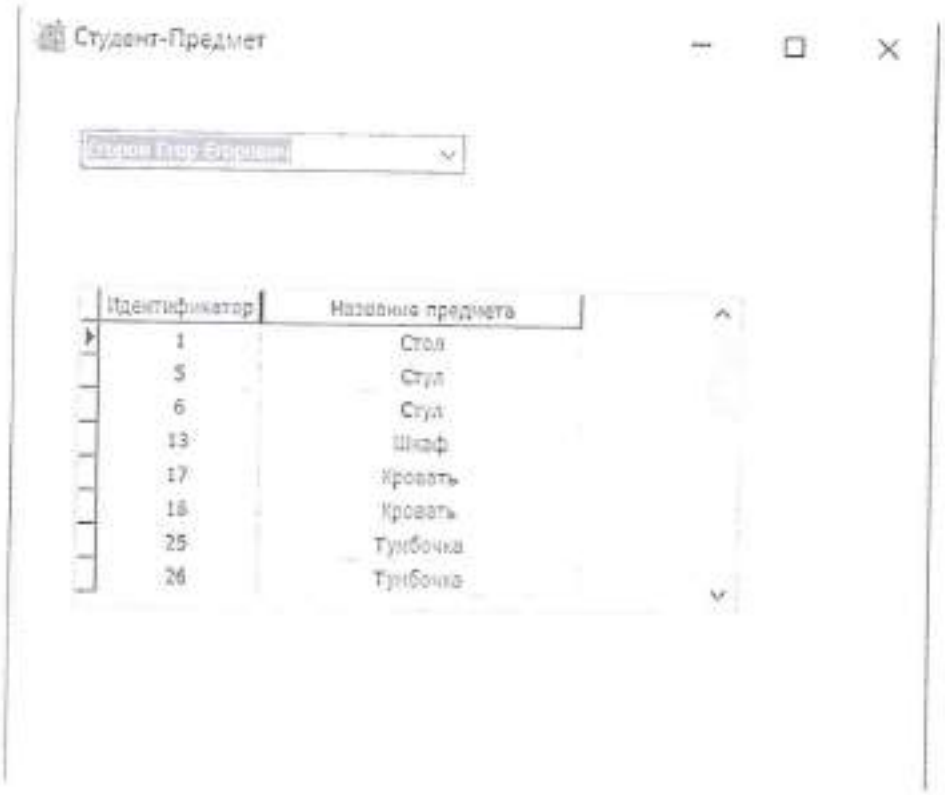

Рисунок 24 – Запрос «Студент-Предмет»

Код запроса:

```
dbgrid1.Visible:=true;
datamodule2.adoquery1.close;
datamodule2.adoquery1.sql.clear;
str1:=combobox1.items.strings[combobox1.itemindex];
datamodule2.adoquery1.sql.Add('Select Идентификаторы.Идентификатор, Студенты.ФИО_студента, Предметы.Название_предмета as [Название предмета]'#13
+#10'from Идентификаторы, Студенты, Предметы where Идентификаторы.Код_предмета=Предметы.Код_предмета and Студенты.Код_комнаты=Идентификаторы.Код_комнаты and Студенты.ФИО_Студента=:p3');
datamodule2.adoquery1.parameters.paramvalues['p3']:=str1;
datamodule2.adoquery1.open;
dbgrid1.Columns[0].title.alignment:=tacenter;
dbgrid1.Columns[0].alignment:=tacenter;
dbgrid1.Columns[0].Width:=100;
dbgrid1.Columns[1].Width:=0;
dbgrid1.Columns[2].title.alignment:=tacenter;
dbgrid1.Columns[2].alignment:=tacenter;
dbgrid1.Columns[2].width:=200;
```

Последний запрос представляет собой количество предметов каждого вида, отсортированное в порядке по убыванию. Для этого запроса потребуется только «TDBGrid». Запрос показан на рисунке 25.

Рисунок 25 – Запрос «Количество предметов»

Код запроса:

```
Datamodule2.adoquery1.close;
datamodule2.adoquery1.sql.clear;
datamodule2.adoquery1.sql.Add('SELECT Название_предмета as [Название предмета], Количество FROM Предметы ORDER BY Количество desc');
Datamodule2.adoquery1.open;
form12.dbgrid1.Columns[0].Width:=140;
form12.dbgrid1.Columns[0].title.alignment:=tacenter;
form12.dbgrid1.Columns[0].alignment:=tacenter;
form12.dbgrid1.Columns[1].Width:=100;
form12.dbgrid1.Columns[1].alignment:=tacenter;
form12.dbgrid1.Columns[1].title.alignment:=tacenter;
form12.DBGrid1.Width:=300;
form12.DBGrid1.height:=300;
```

Все вышеописанные действия помогли создать клиентское приложение, которое облегчит использование базы данных, созданной в Microsoft Access.

ЗАКЛЮЧЕНИЕ

Цели курсовой работы выполнены полностью. Была разработана база данных для реестра имущества университетского городка. Курсовая работа создана в программе Microsoft Access. Был реализован удобный пользовательский интерфейс. В данной работе также выполнены задачи, которые ставились в начале курсовой работы, все задачи решены, и их решение можно просмотреть непосредственно в курсовой работе.

В процессе разработки базы данных, были сделаны выводы, что система управления базами данных Microsoft Access – мощный и удобный инструмент для их создания. Графический многооконный интерфейс, который дает возможность в диалоговом режиме создавать таблицы, формы, запросы, отчеты. Специальные приспособления, которые автоматизируют работу во время создания и ведения базы данных («Мастера», «Конструкторы» и прочее) заметно упрощают весь процесс и делают программу доступной в изучении каждому.

В современном мире непрерывно развиваются технологии, а значит, появляются всё более надёжные и громоздкие базы данных. Однако, далеко не все предприятия и заведения могут себе позволить самые новые разработки. Но в данной курсовой работе и не были поставлены такие задачи. Суть кроется в том, что не всем предприятиям требуются базы данных, спроектированные по последнему слову техники, а следовательно, Microsoft Access в комбинации с Delphi способен к качественному обслуживанию многих клиентов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1) Абросимова, М.А. Базы данных: проектирование и создание программного приложения в СУБД MS Access: практикум / М.А. Абросимова. – М, 2014. – 6 с.

2) Аникеев, С.В. Разработка приложений баз данных в Delphi: самоучитель / С.В. Аникеев, А.В. Маркин. – М, 2013. – 36 с.

3) Баженова, И.Ю. Основы проектирования приложений баз данных / И.Ю. Баженова. – М, 2016. – 34 с.

4) Баженова, И.Ю. SQL и процедурно-ориентированные языки / И.Ю. Баженова. – М, 2016. – 23 с.

5) Братченко, Н.Ю. Распределенные базы данных: учебное пособие / Н.Ю. Братченко. – Ставрополь, 2015. – 19 с.

6) Букатов, А.А. Методы и средства интеграции независимых баз данных в распределенных телекоммуникационных сетях: монография / А.А. Букатов, А.В. Пыхалков. – Ростов-на-Дону, 2013. – 23 с.

7) Васюков, О.Г. Управление данными: учебно-методическое пособие / О.Г. Васюков. – Самара, 2014. – 33 с.

8) Громов, Ю.Ю. Управление данными: учебник / Ю.Ю. Громов, О.Г. Иванова, А.В. Яковлев. – Тамбов, 2015. – 115 с.

9) Грошев, А.С. Информатика: учебник для вузов / А.С. Грошев. – М, 2015. – 382 с.

10) Гущин, А.Н. Базы данных: учебник / А.Н. Гущин. – М, 2014. – 115 с.

11) Жданов, С.А. Информационные системы: учебник / С.А. Жданов. – М, 2015. – 80 с.

12) Карпова, Т.С. Базы данных: модели, разработка, реализация: учебное пособие / Т.С. Карпова. – М, 2016. – 343 с.

13) Колокольникова, А.И. Информатика: учебное пособие / А.И. Колокольникова, Е.В. Прокопенко, Л.С. Таганов. – М, 2013. – 91 с.

14) Королев, В.Т. Технология ведения баз данных: учебное пособие / В.Т. Королев, Е.А. Контарёв, А.М. Черных. – М, 2015. – 38 с.

15) Кузнецов, С.Д. Введение в модель данных SQL: курс / С.Д. Кузнецов. – М, 2016. – 117 с.

16) Кузнецов, С.Д. Введение в реляционные базы данных / С.Д. Кузнецов. – М, 2016. – 32 с.

17) Медведкова, И.Е. Базы данных / И.Е. Медведкова, Ю.В. Бугаев, С.В. Чикунов. – Воронеж, 2014. – 76 с.

18) Микляев, И.А. Универсальные объектно-ориентированные базы данных на реляционной платформе: Монография / И.А. Микляев. – Архангельск, 2014. – 17 с.

19) Платонов, Ю.М. Информатика: учебное пособие / Ю.М. Платонов, Ю.Г. Уткин, М.И. Иванов. – М, 2014. – 157 с.

20) Сирант, О.В. Работа с базами данных / О.В. Сирант, Т.А. Коваленко. – М, 2016. – 83 с.

21) Черячукин, В.В. Право интеллектуальной собственности на программы для ЭВМ и базы данных в Российской Федерации и зарубежных странах: учебное пособие / В.В. Черячукин. – М, 2015. – 35 с.

22) Чеснокова, О.В. Информационные технологии в юридической деятельности: учебное пособие / О.В. Чеснокова. – М, 2014. – 167 с.

23) Чурбанова, О.В. Базы данных и знаний. Проектирование баз данных в Microsoft Access: учебно-методическое пособие / О.В. Чурбанова, А.Л Чурбанов. – Архангельск, 2015. – 78 с.

24) Щелоков, С.А. Базы данных: учебное пособие / С.А. Щелоков. – Оренбург, 2014. – 124 с.

25) Щелоков, С.А. Разработка и создание баз данных средствами СУБД Access и SQL Server / С.А. Щелоков. – Оренбург, 2014. – 21 с.