

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Программное обеспечение средств ВТ и автоматизированных систем»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

на тему

**Клиент-серверное приложение переноса
данных между базами данных**

Студент



(подпись, дата)

Кузнецов Алексей Сергеевич

(ФИО полностью)

Руководитель



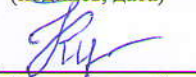
(подпись, дата)

12.06.2020

Зинкин С.А.

(фамилия, инициалы)

Нормоконтролёр



(подпись, дата)

Кучин А.В.

(фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 03.06.2020г. № 12)

Заведующий кафедрой



(подпись)

М.А. Митрохин

(фамилия, инициалы)

Работа защищена с отметкой

17.06.2020 № 9



(протокол заседания ГЭК от

Секретарь ГЭК



(подпись)

Е.И. Гурин

(фамилия, инициалы)

Пенза 2020

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
 М.А. Митрохин
«30» апреля 2020 г.

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА**

1. Студент Кузнецов Алексей Сергеевич гр.16BB2 факультета вычислительной техники

2. Тема работы: «Клиент-серверное приложение переноса данных между базами данных»

Тема утверждена приказом ПГУ № 0335а/о от " 30 " апреля 2020 г.

3. Руководитель бакалаврской работы д.т.н., профессор Зинкин С.А.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Назначение разработки – возможность открытия базы данных нескольких форматов, работа с ними и перенос данных в базы других типов.

Разработать клиентское приложение, позволяющее пользователю работать с базой данных.
Разработать серверное приложение, которое выполняет обработку полученных от приложений-клиентов запросов к базе данных.

5. Перечень подлежащих разработке вопросов.

Требуется провести анализ предметной области, выбрать систему управления базами данных, определить форматы, которые будут доступны для работы в приложении, спроектировать клиентское и серверное приложение, реализовать приложения, отладить и провести тестирование разработанных приложения.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области	10	30.04.20	
2.	Определение перечня используемых СУБД	10	04.05.20	
3.	Проектирование клиентского и серверного приложений	10	10.05.20	
4.	Создание клиентской части приложения	20	15.05.20	
5.	Создание серверной части приложения	20	20.05.20	
6.	Отладка и тестирование комплекса приложений	10	23.05.20	
7.	Оформление выпускной работы	20	31.05.20	

Дата выдачи задания " 30 " апреля 20 20 г.

Руководитель бакалаврской работы 
(подпись, дата)

Задание к исполнению принял студент 
(подпись, дата)

Работу к защите допустить
Декан факультета 
(подпись, дата)

Реферат

Пояснительная записка содержит 112 листов, 53 рисунка, 1 таблица, 8 использованных источников, 4 приложения.

КЛИЕНТ-СЕРВЕРНОЕ ПРИЛОЖЕНИЕ, БАЗА ДАННЫХ, ПРОТОКОЛ TCP/IP, SQL-ЗАПРОСЫ, СУБД, ЭКСПОРТ ДАННЫХ.

Цель работы – разработать клиент-серверное приложение, которое будет иметь доступ к различным базам данных, возможность их редактировать и экспортировать в другой тип базы данных.

Объектом разработки будет являться клиентское и серверное приложения, которые должны, взаимодействуя между собой, осуществлять работу с базами данных.

В результате проведенной работы был создан проект, удовлетворяющий поставленной задаче, а именно, создано клиентское приложение, которое имеет интуитивно понятный интерфейс для отображения и работы с данными из базы, и серверное приложение, которое принимает и направляет SQL-запросы от клиента в требуемую базу данных. Взаимодействие приложений проходит по протоколу TCP/IP.

Данный проект рекомендуется использовать при необходимости доступа к различным базам данных и переноса между ними. На предприятии, где используются базы данных различных форматов, с целью сокращения используемых СУБД до одной может быть внедрён данный проект

					ПГУ 2.090301.22.001 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.	Кузнецов А.С.				Клиент-серверное приложение переноса данных между базами данных Пояснительная записка	Лит	Лист	Листов
Провер.	Зинкин С.А.						2	112
Н. Контр.	Кучин А.В.					ФВТ, каф. ВТ гр. 16ВВ2		
Утверд.	Митрохин М.А.							

Содержание

Перечень принятых сокращений.....	5
Введение	6
1 Анализ предметной области	8
1.1 Определения перечня СУБД	8
1.2 Способы доступа к базам данных	9
1.2.1. ODBC	10
1.2.2. OLEDB	13
1.3 Архитектура приложения	15
1.4 Связь между клиентом и сервером	18
2 Проектирование приложения	20
2.1 Проектирование клиентской части	21
2.2 Проектирование серверного приложения.....	27
3 Реализация проекта	30
3.1 Процесс разработки.	30
3.1.1 Разработка клиентского приложения.....	30
3.1.2 Разработка серверного приложения.....	33
3.2 Тестирование и отладка	34
3.2.1 Тестирование возможности данных переноса между базами данных	35
3.2.2 Тестирование возможности добавления новой таблицы в базу данных	42
3.2.3 Тестирование возможности изменения данных в ячейки	45
4 Руководство пользователя	47
5 Безопасность жизнедеятельности	56
Заключение.....	58
Список используемых источников	59
Приложение А листинг программы сервера	60
Приложение А.1	60
Файл «Server.cs».....	60

Приложение А.2	79
Файл «Core.cs»	79
Приложение В листинг программы клиента	80
Приложение В.1	80
Файл «MainForms.cs».....	80
Приложение В.2	89
Файл «core.cs»	89
Приложение В.3	89
Файл «AddColumnForm.cs»	89
Приложение В.4	91
Файл «AddDataForm.cs»	91
Приложение В.5	92
Файл «AddRowForm.cs»	92
Приложение В.6	94
Файл «AddTableForm.cs».....	94
Приложение В.7	95
Файл «ConnectFom.cs».....	95
Приложение В.8	100
Файл «DeleteColumnFom.cs»	100
Приложение В.9	102
Файл «DeleteTableFom.cs».....	102
Приложение В.10	103
Файл «SQLRequistForm.cs»	103
Приложение С	105
UML-диаграммы приложения.....	105
Приложение С.1 – UML-диаграмма классов клиента.....	105
Приложение С.2 – UML-диаграмма классов клиента.....	106
Приложение D	107
Презентация	107

Перечень принятых сокращений

БД База данных

СУБД Система управления базами данных

					ПГУ 2.090301.22.001 ПЗ	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		

Введение

Изначально ЭВМ создавались для того, чтобы решать математические задачи, но развитие вычислительной техники дало новые пути использования компьютеров. Появились такие возможности, как хранение и обработка информации. Изобретение мощных устройств внешней памяти, а именно жестких дисков, сделало возможным создание первых баз данных и систем их управления (СУБД). Изначально работа с базами данных осуществлялась через консольный терминал.

В настоящий момент базы данных и СУБД подверглись большим изменениям. Из-за стремительного развития вычислительной техники базы данных стали огромными, а СУБД перешли из консоли в удобные оконные приложения с понятным интерфейсом и большим выбором инструментов для работы с данными.

Так же со временем изменился и доступ к данным. Сейчас всё чаще используются распределенные системы, в которых пользовательский интерфейс находится, непосредственно, на стороне клиента, а база данных на сервере и к ней осуществляется удаленный доступ. Причем сервер и клиент могут находиться в одной локальной сети или связываться через интернет.

Нам уже трудно представить работу какой-либо организации без использования баз данных. СУБД позволяют создавать отказоустойчивые хранилища данных. Хранение данных в электронной форме помогает заметно экономить время, так как современные системы управления базами данных позволяют быстро выполнять различные действия над данными, например: изменение данных, копирование, добавление и удаление, фильтрация и сортировка и т.д. Выполняя эти действия вручную, пользователи затрачивали гораздо больше времени.

Активное использование баз данных вызвало появления большого количества СУБД. Если по каким-либо причинам использование одной

					ПГУ 2.090301.22.001 ПЗ	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		

системы управления станет недоступным, то при переходе на новую, вероятнее всего, возникнут проблемы переноса данных в другую СУБД

В данной работе будет разработано клиент-серверное приложение, позволяющее работать с данными в различных базах данных, а также переносить данные между ними.

					ПГУ 2.090301.22.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

1 Анализ предметной области

В мире существует большое число различных СУБД, которые предназначены для работы с базами данных, а именно позволяющих создавать, редактировать, сохранять и выполнять другие действия над данными в базе. Как правило, каждая СУБД может работать только с файлами определенного формата. Для MySQL это .frm, для Microsoft SQL Server формат .mdf, а для Microsoft Access - .mdb или .accdb и т.д. Однако СУБД, которые могут работать с файлами баз данных различных форматов, не распространены.

Цель моей работы состоит в создании приложения, способного открывать файлы различных форматов, содержащие базы данных, а так же дающего возможность работы с данными, их редактирования и переноса их в файлы других форматов.

Выбор верного решения требует проведения анализа предметной области и определения некоторых ключевых моментов, на которых будет основываться разработка приложения.

1.1 Определения перечня СУБД

Для выполнения проекта требуется выбрать несколько распространенных форматов баз данных. Согласно рейтингу самых популярных СУБД за 2019 год, в топ-10 входят Microsoft SQLS erver, MySQL и Microsoft Access. Выбор именно этих вариантов обусловлен следующими моментами.

Microsoft SQL Server – это классическое коммерческое СУБД, которое позволяет оперировать базами данных довольно крупных размеров. Язык запросов Transact-SQL является основой работы этой СУБД.

MySQL – это свободно распространяемая СУБД. Для работы в ней используется процедурное расширения языка SQL – SQL/PSM (PersistentStoredModule)

					ПГУ 2.090301.22.001 ПЗ	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		

MicrosoftAccess входит в пакет MicrosoftOffice, которая широко используется в компаниях, а потому распространена в офисной среде. Для работы в ней не используется язык SQL или иных языков. Интерфейс этой СУБД интуитивно понятен и доступен любому пользователю.

Такой выбор СУБД позволит приложению удовлетворить потребности большей части пользователей.

1.2 Способы доступа к базам данных

ODBC (Open Database Connectivity) и OLEDB (Object Linking and Embedding, Database) – это драйверы для доступа к данным.

ADO (ActiveX Data Object) – это набор классов для работы с данными, который предоставляет только инструменты для удобного взаимодействия с источниками данных, а также требующий для своей работы один из драйверов, упомянутых выше, поэтому исключим его из рассмотрения и выбора подходящего для реализации проекта способа доступа к базе данных.

Возможные варианты доступа к базам данных из приложения показаны схемой на рисунке 1.

					ПГУ 2.090301.22.001 ПЗ	Лист
						9
Изм.	Лист	№ докум.	Подпись	Дата		

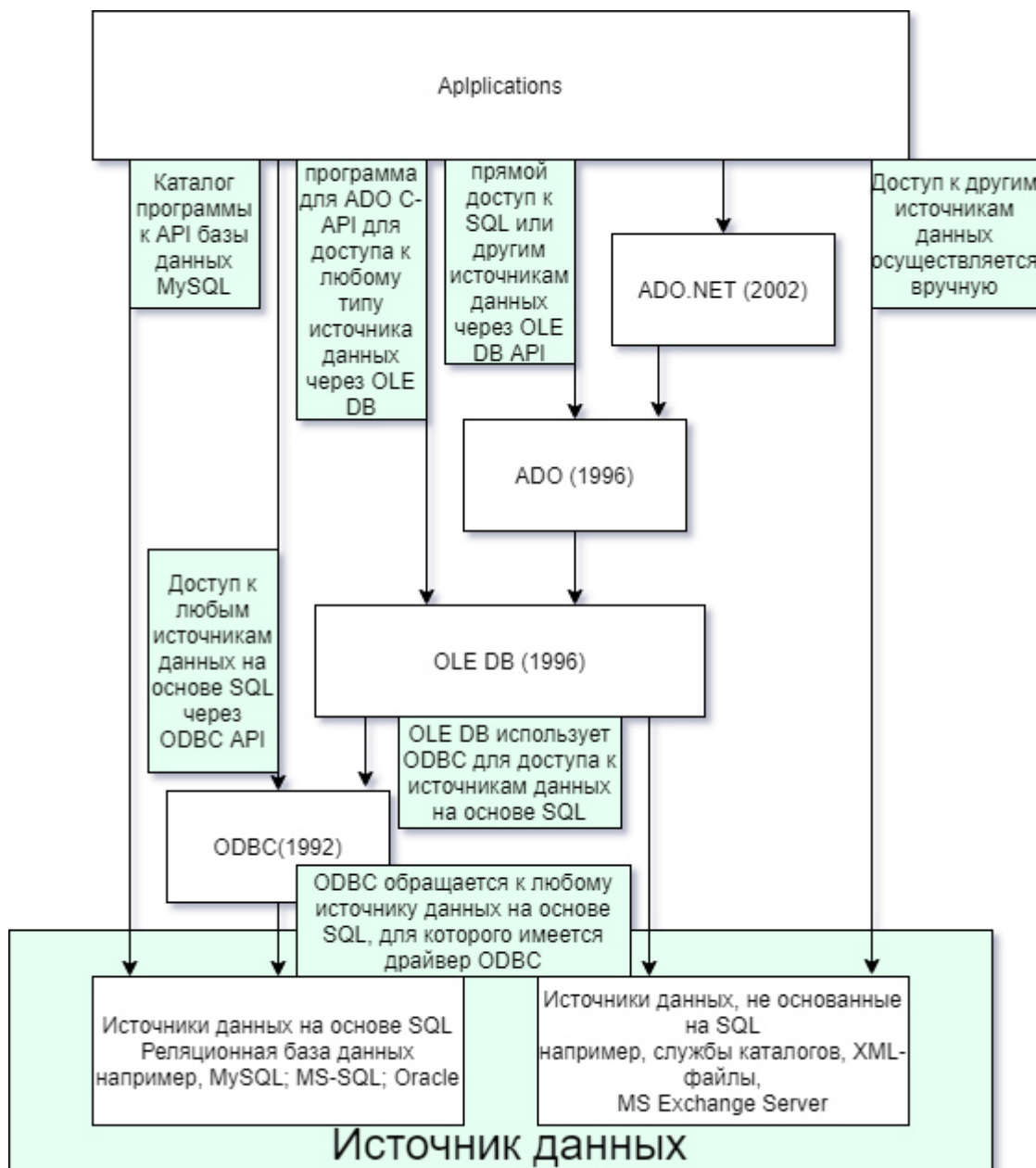


Рисунок 1 – Доступ к базе данных

1.2.1. ODBC

В 1990-х годах был создан интерфейс ODBC. Тогда существовало несколько поставщиков БД, которые имели собственный интерфейс. Для работы с несколькими источниками было необходимо писать код для каждого из них. Чтобы решить эту проблему компания Microsoft вместе с Simba Technologies создали интерфейс ODBC (Open DataBase Connectivity), что переводится как «открытая для модификаций связь с базами данных». Этот интерфейс позволил унифицировать доступ к различным источникам данных. Благодаря ему программисты смогли

создавать приложения, не обращая внимания на особенности каждого типа данных и не создавая для них отдельный программный код.

ODBC - это интерфейс уровня вызова, позволяющий приложениям иметь доступ к абсолютно любой базе данных, имеющей в ODBC драйвер. ODBC обеспечивает API, позволяющий приложению быть независимым от СУБД. Архитектура ODBC состоит из четырёх основных частей:

-Драйверы

-источники данных

-клиентское приложение (прикладная программа)

-менеджер драйверов

ODBC на физическом уровне является набором динамических библиотек DLL, обслуживающих подключение и работу с определенными типами БД. При выполнении запроса на присоединение к конкретной базе данных подключается нужная DLL. Чтобы добавить новый тип базы данных необходимо написать драйвер и зарегистрировать его. Это доказывает определение архитектуры как легко расширяемой и открытой для модификации. Но приложения, которые используют ODBC, функционируют посредством менеджера драйверов, используя SQL (рис.2).

					ПГУ 2.090301.22.001 ПЗ	Лист
						11
Изм.	Лист	№ докум.	Подпись	Дата		

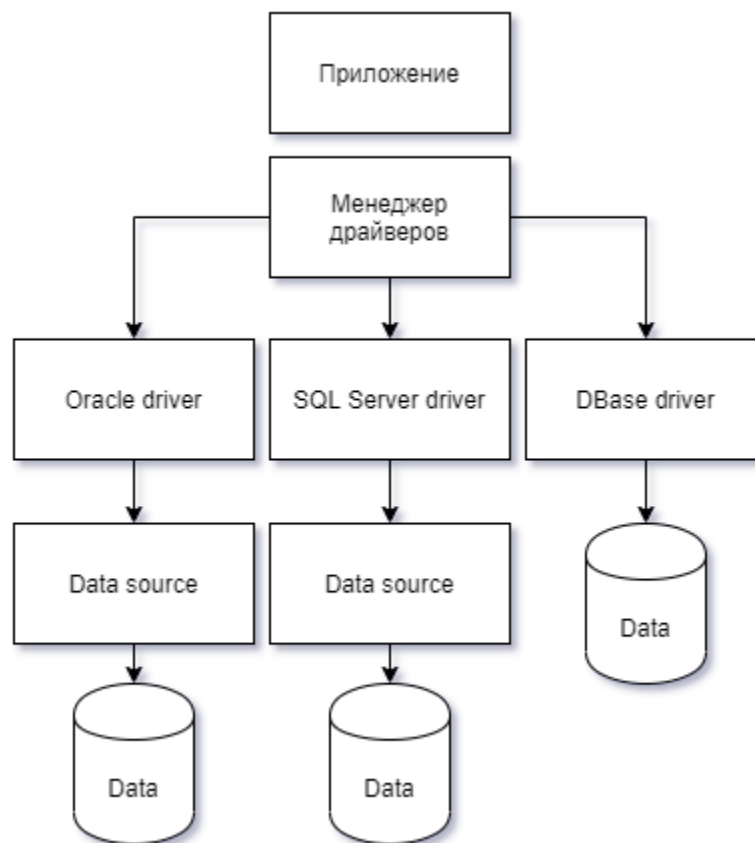


Рисунок 2 – Архитектура ODBC

Клиентское приложение обращается к БД, используя менеджер драйверов, независимо от ее типа. То есть, для пользователя практически не существует разницы между источниками данных. Следовательно, можно изменить тип базы данных, но работу над ней можно выполнять по уже имеющимся алгоритмам. Однако, следует иметь ввиду, что разные типы БД могут поддерживать расширения языка SQL, которые имеют различия в синтаксисе некоторых команд, но, в основном, подобные различия не столь значительны.

Для обеспечения этих функций работы с БД ODBC использует конструкции языка SQL для связи с разными системами управления БД:

- создание таблиц, изменение таблиц и удаление таблиц;
- обновление, отбор, удаление и вставка полей таблиц;
- создание и удаление представлений и индексов.

ODBC дает возможность работать с такими источниками данных: Microsoft SQL Server, FoxPro, Microsoft Access, Microsoft Excel и др.

1.2.2. OLEDB

Для замены ODBC компания Microsoft создала интерфейс OLEDB, который позволяет работать с:

- базами данных типа Microsoft Access, базами под управлением Microsoft SQL Server, Oracle Database и другими;
- базами данных для мейнфреймов
- индексно-последовательными файлами;
- система электронной почты типа Exchange;

Различия OLEDB от ODBC являются в возможности работать с нереляционными БД и многими другими источниками данных, а ODBC, в свою очередь, дает возможность работы только с реляционными, основанными на SQL источниками. Так же существуют и другие различия, которые представлены в таблице 1.

					ПГУ 2.090301.22.001 ПЗ	Лист
						13
Изм.	Лист	№ докум.	Подпись	Дата		

Таблица 1 - Сравнение OLEDB и ODBC

Термин	OLE DB	ODBC
Источник данных	Модуль, взаимодействующий с хранилищем информации и показывающий его содержимое в виде набора столбцов и строк	Реляционная СУБД
Команда	Текстовая строка, которую понимает источник данных	Команда SQL, интерпретируется подключенной СУБД
Таблица	Набор данных, полученный агрегированием связанных строк и однородных столбцов	Таблица в реляционной базе данных
Результирующий набор данных	Независимый автономный модуль, который показывает последовательность записей	Часть памяти клиента, содержащая последовательность записей

1.3 Архитектура приложения

Согласно заданию, проект должен иметь клиент-серверную архитектуру. Следовательно, он должен состоять из двух частей – клиентского и серверного приложения.

Клиент – является компонентом вычислительной системы (приложение, ЭВМ или же пользователь), который запрашивает и использует информацию и ресурсы у сервера, с которым установлено соединение. Сервер – это компонент вычислительной системы (приложение, ЭВМ или пользователь), который предоставляет информацию и ресурсы для использования.

У такой архитектуры есть свои плюсы и минусы. К плюсам можно отнести:

- мощность. Рабочие станции могут быть по мощности сравнимы с мэйнфреймами, но стоимость гораздо ниже;
- гибкость. При такой архитектуре можно объединять компьютерные платформы для подбора их под определенные нужды;
- открытость. Программы, аппаратуру и услуги можно приобретать у разных поставщиков;
- масштабируемость. Система легко модернизируется, если изменяются потребности;
- отказоустойчивость. При выходе одной из машин из строя, вся остальная сеть продолжает нормально функционировать.

Из минусов можно выделить:

- небольшое количество инструментальных средств обслуживания.
- Используя архитектуру клиент-серверной программы, неоднократно

					ПГУ 2.090301.22.001 ПЗ	Лист
						15
Изм.	Лист	№ докум.	Подпись	Дата		

приходится искать инструментальные средства на рынке или разрабатывать их самостоятельно;

- необходимость переобучения. Программирование для Windows или MacOS значительно отличается от программирования на языке С или COBOL;

- слабая поддержка. Некоторые части системы не всегда работают вместе правильно. Иногда довольно тяжело найти причину ошибки.

Архитектура клиент-серверной программы предусматривает, что на сервере уже установлена как СУБД, так и база данных, а на рабочих станциях находится клиентская часть приложения с алгоритмами, дающим доступ к базам. Особенностью такой архитектуры является, то, что в отличие от файл-серверной, СУБД находится не на серверной, а на клиентской машине.

Схема простейшей модели взаимодействий между клиентом и сервером показана на рисунке 3.



Рисунок 3 – Клиент-серверная архитектура

На рисунке мы видим, что клиент отправляет запросы на сервер, а сервер переправляет их на базу данных, а после отправляет ответ клиенту. Изображенная на рисунке 5 архитектура, называется двухзвенной. Так же существует так же трёхзвенная архитектура (рисунок 4)

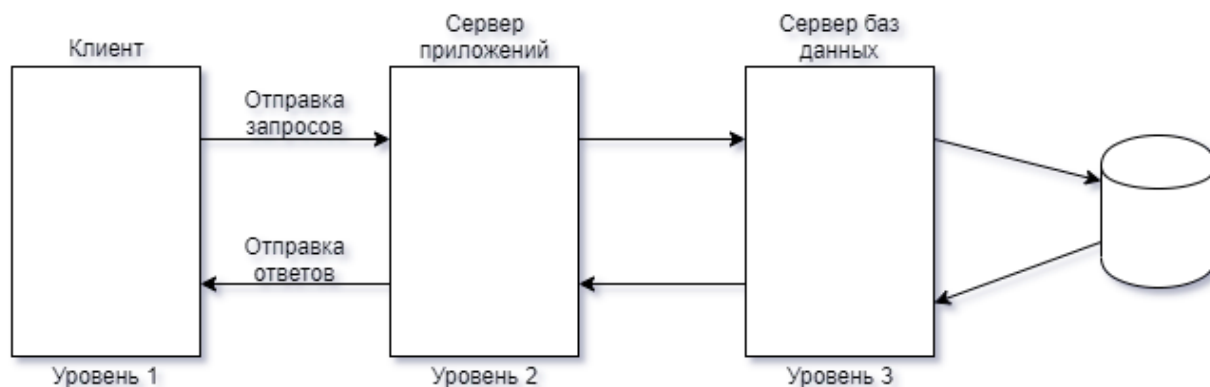


Рисунок 4 – Трёхзвенная клиент-серверная архитектура

Такой способ архитектуры предполагает разбиение сервера на сервер приложений и сервер баз данных. Это помогает уменьшить нагрузку на сервер и обеспечить большее подключение клиентов, но данный метод имеет и свои недостатки, среди которых снижение надежности и увеличение время ожидания ответа от сервера клиенту (рис. 5).



Рисунок 5 – Время ожидания ответа

Базы данных MySQL и Microsoft SQL Server физически представляют собой файл, но доступ к ним нужно осуществлять через специальный сервер. В свою очередь, базы данных Microsoft Access хранятся на компьютере виде файлов. Следовательно, можно сказать, что работа с базами данных Microsoft Access организовывается по принципу двухзвенной архитектуре, а работа в SQL базах данных – как в трёхзвенной, то есть наше серверное приложение будет являться сервером приложений

1.4 Связь между клиентом и сервером

Существует несколько различных способов расположения клиента и сервера относительно друг друга (рис 6).

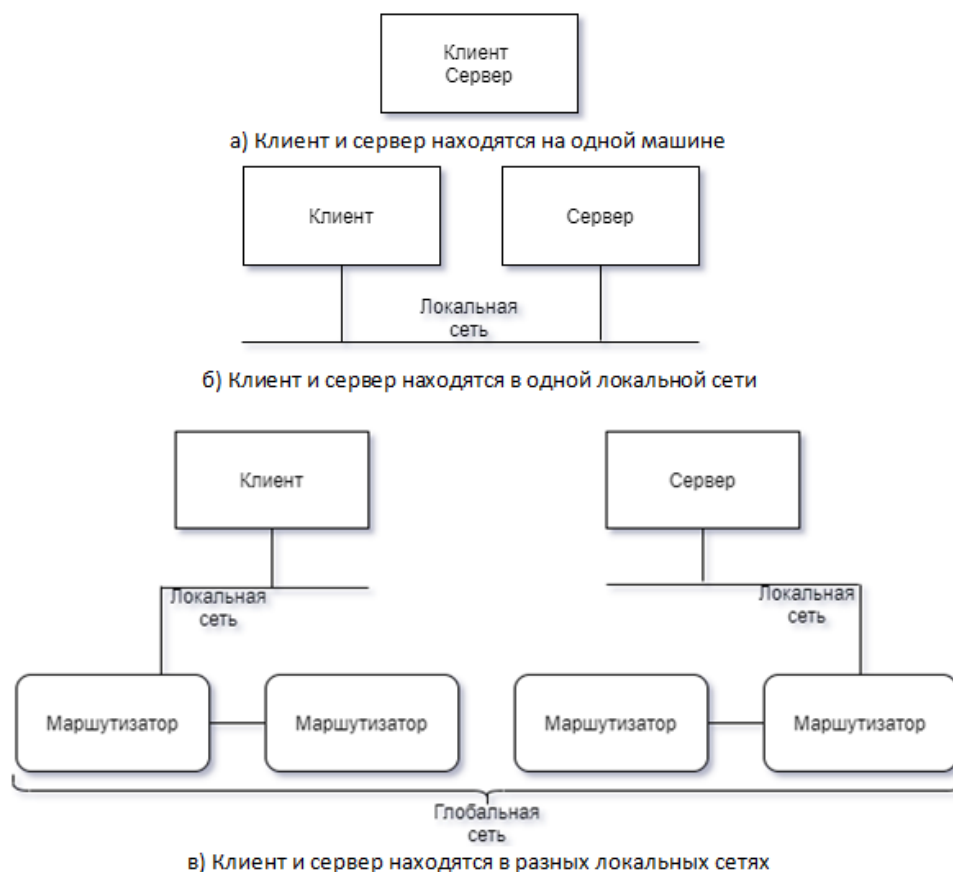


Рисунок 6 – Варианты связи клиента и сервера

На рисунке 6(а) показан самый простой способ расположения клиента и сервера, потому что в этом случае физическая связь отсутствует. Такой способ может быть использован для однопользовательских программ, но для корпоративных баз данных такой вариант не подходит.

На рисунке 6(б) пример расположения клиента и сервера в пределах одной локальной сети, но на разных станциях.

На последнем рисунке расположения клиента и сервера такое, что они находятся на разных машинах, которые связывает глобальная сеть Internet. Для двух последних вариантов необходимо организовать связь по сети. Для обеспечения этого используется стек протоколов TCP/IP.

TCP (Transfer Control Protocol) – это протокол транспортного уровня, который обеспечивает передачу данных между двумя машинами. Так же протокол контролирует подходящий размер передаваемых пакетов и, в случае сбоя, выполняет повторную передачу. Протокол не имеет ограничения на количество установленных соединений в сети, другими словами, любая станция может обмениваться данными с любым количеством станций сети. Еще одним достоинством сети с протоколами TCP/IP является то, что по нему можно объединить станции с разными архитектурами и операционными системами, такие как, Windows, MacOS, Unix и др.

Суть передачи данных по протоколу TCP/IP состоит в следующем: на сервере создается «прослушивающий» сокет – пара значений, созданная IP-адресом и портом. Клиент может подключиться к серверу, отправив данные на определенный адрес в конкретном порту. Если подключение установлено, то сервер выходит из состояния ожидания и начинает выполнять действия по обработке данных, которые получает от клиента. Из этого следует вывод, что передача данных через TCP/IP является простой в реализации и весьма надежной, а следовательно, для связи между клиентом и сервером в данном приложении разумно использовать именно этот способ организации передачи данных.

					ПГУ 2.090301.22.001 ПЗ	Лист
						19
Изм.	Лист	№ докум.	Подпись	Дата		

2 Проектирование приложения

Клиент-серверная архитектура предполагает, что доступ к базе данных будет осуществляться непосредственно через сервер, к которому, в свою очередь, клиент будет отправлять запросы. Клиент-серверное соединение должно быть организовано по протоколу TCP/IP, для обеспечения сетевой работы множества приложений.

Клиентское приложение должно иметь интуитивно понятный графический интерфейс и давать возможность редактирования данных баз данных, а именно, перенос данных в другую базу данных. Запросы, направленные к базе данных, должны создаваться на основании какого-либо действия пользователя и представлять собой команды на языке SQL, так как подавляющее число СУБД поддерживают такой вариант взаимодействия с базой. После этого сформированные запросы отправляются на сервер для последующей передачи в базу данных.

Серверное приложение может не иметь графического интерфейса, поскольку для его работы не требуется никакого ввода данных. Работа сервера состоит в создании «прослушивающего» сокета и в ожидании подключения клиента. Когда соединение установлено, осуществляется прием данных от клиента и их обработка, а именно, отправка запроса клиента в базу данных. Если не возникло никаких ошибок, то сервер должен отправить клиенту сообщение, а после снова перейти в режим ожидания.

Взаимодействие в архитектуре «Клиент-сервер» показано на рисунке 7.

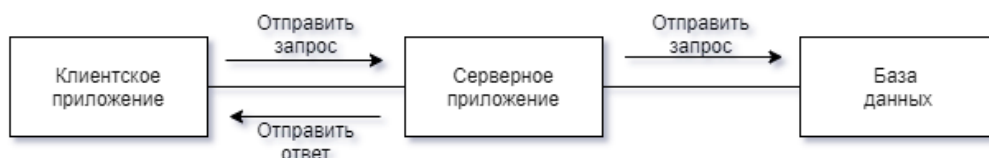


Рисунок 7 – UML-диаграмма взаимодействия клиента и сервера

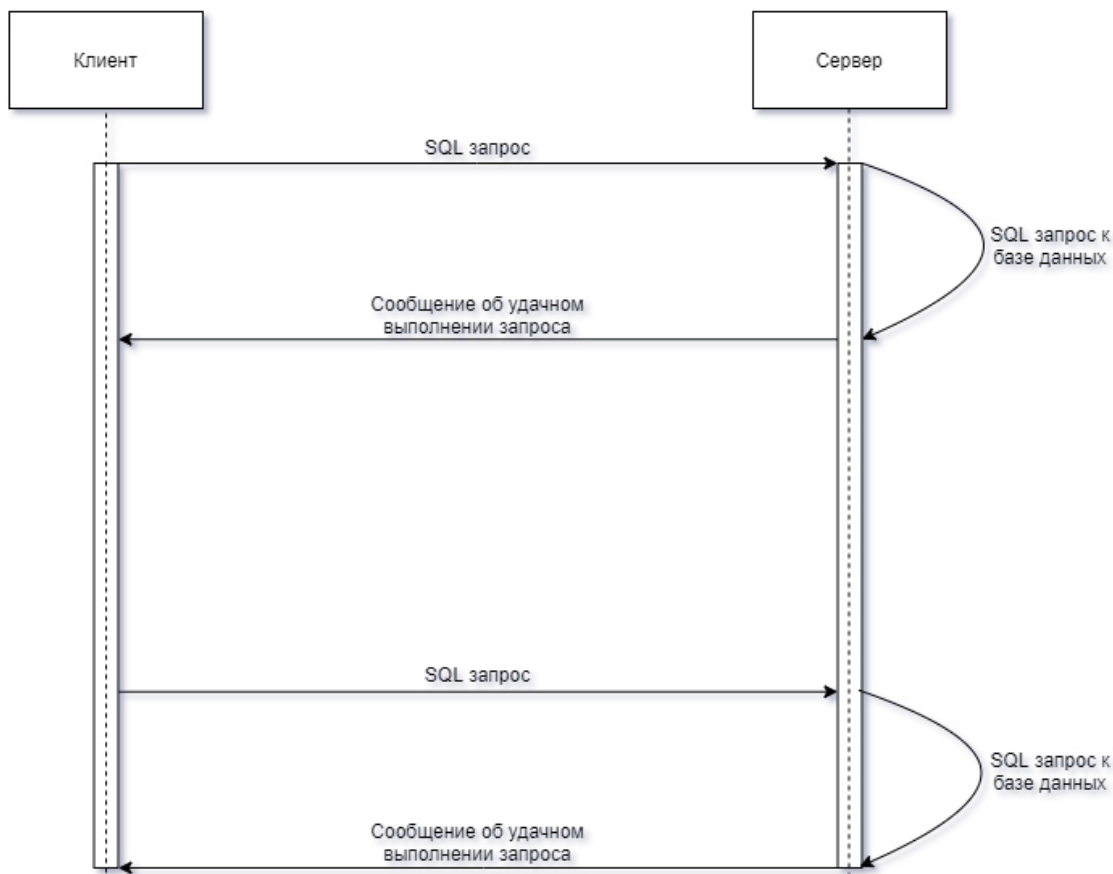


Рисунок 8 – Диаграмма последовательности

2.1 Проектирование клиентской части

Интерфейс OLEDB представляет источник данных в виде столбцов и строк. Для реализации проекта выбраны реляционные СУБД, базы данных, которые, соответственно, были созданы с их помощью. Таким образом, главным элементом в интерфейсе является таблица, содержащая данные. Кроме того, есть необходимость создать элемент, который даст возможность переключаться между всеми таблицами базы данных. Делая вывод, предположим, как пользователь будет взаимодействовать с базой данных. Все возможные варианты схематично изображены на UML-диаграмме (рис. 9).

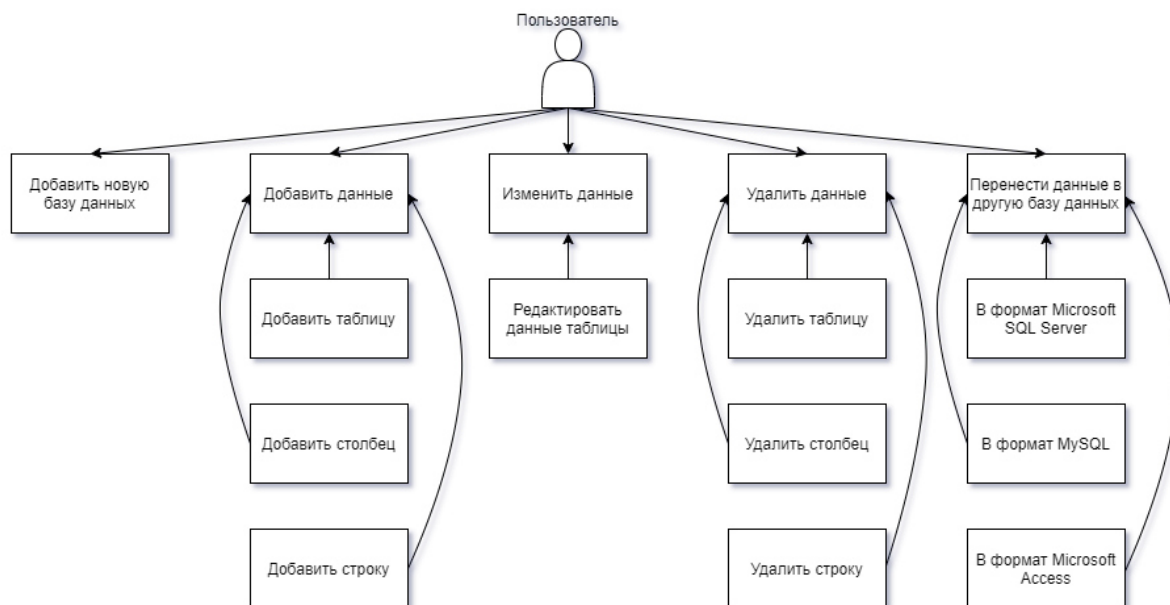


Рисунок 9 – Диаграмма вариантов использования

Для того, чтобы реализовать все эти возможности, необходимо в клиентском приложении добавить меню, вкладки или кнопки, но, при всем этом, интерфейс должен оставаться интуитивно понятен для пользователя. Они должны быть расположены так, чтобы не препятствовать работе с таблицами и данными. Некоторые элементы стоит рассмотреть более подробно.

Для добавления нового столбца от пользователя требуется ввести в одной из полей его название, а в другом поле выбрать его тип. В СУБД большое количество типов данных, мы же будем использовать самые популярные из них, а именно:

- числовой (int);
- текстовый (text);
- с плавающей точкой (float);
- десятичный или для точных значений (decimal);
- хранение информации в бинарном виде (binary)

С таким набором типов данных можно создать любую базу, так как другие типы можно подвести к одному из указанных. Например, данные

столбца базы Microsoft Access типа «Денежный» без труда переносится в столбец числового типа. Следовательно, добавление нового столбца должно давать выбор как минимум из этих пяти типов.

Добавление строки и изменение данных в ячейке должно производиться только после проверки типа введенных данных и типа столбца, в который они заносятся. Если данные не соответствуют, то следует вывести сообщение об ошибке и дать пользователю изменить данные.

Особое внимание так же необходимо уделить возможности «Открыть базу данных». Так как клиент-серверное приложение должно работать по сети, то клиент сначала должен подключиться к серверу, а после начать работу с базой данных. Так как в данной работе нет необходимости для отдельного подключения к серверу, было решено, что клиент будет подключаться автоматически при подключении к базе данных.

Для того чтобы подключиться необходимо набор полей ввода должен содержать, как минимум:

- IP адрес сервера;
- порт сервера, принимающий данные;
- тип базы данных (Microsoft SQL Server, MySQL, Microsoft Access);
- данные учетных записей для Microsoft SQL Server и MySQL;
- название базы данных, к которой хотим подключиться.

Если приложение будет иметь возможность сохранять вводимые данные в полях, то это значительно сократит время на подключение к базе и повысит эффективность работы. Для обеспечения такой возможности требуется сделать хранение данных в файле. Храниться будут последние данные, по которым удалось подключиться. После успешного подключения к базе окно должно быть закрыто и уже в клиентском приложении открыта база

					ПГУ 2.090301.22.001 ПЗ	Лист
						23
Изм.	Лист	№ докум.	Подпись	Дата		

данных, а в случае возникновения ошибки вывести сообщение об ошибке введенных данных.

Требования, перечисленные ранее, касаются исключительно интерфейса клиентского приложения и его взаимодействия с пользователем. Схема алгоритма работы клиента, связанная с подключением к базе данных, показана на рисунке 10.



Рисунок 10 – Алгоритм работы части клиентского приложения

После присоединения к базе данных, она должна открыться в главном окне приложения. Переключения между таблицами баз данных должно быть простым и интуитивно понятным, например, с помощью вкладок.

Все действия пользователя должны вызывать соответствующие обработчики. Так как в выбранных СУБД есть возможность взаимодействия с базой данных с помощью выполнения запросов на языке SQL, то в обработчиках должны конструироваться команды, в которые подставляются названия таблиц или столбцов в соответствии с синтаксисом той или иной команды. Но стоит принимать во внимание, что выбранные СУБД используют расширения языка SQL, что может стать причиной некоторой разницы в синтаксисе.

Отправка готового запроса на сервер должна быть осуществлена через создание сокета с введенными данными и последующего подключения к серверу. Кроме того, еще необходимо передавать строку подключения к базе данных, так как с её помощью будет осуществляться доступ к конкретной базе данных. Такая строка должна формироваться на этапе подключения к серверу и доступа к базе данных.

Следующим этапом является обновления интерфейса в соответствии с внесенными изменениями. Так как отрисовка элементов будет затрачивать относительно длинный промежуток времени в зависимости от размера базы данных, требуется максимально возможно сократить количество перерисовываемых элементов. То есть, если пользователь внёс изменения только в одну таблицу, то следует обновить только её, если же работа проводится над таблицами, то необходимо обновлять всю базу полностью. После добавления новой таблицы в базу данных, для удобства, стоит сразу открыть ее в приложении. Если одна из таблиц была удалена, то следует открыть любую другую таблицу из базы данных.

На рисунке 11 показана схема работы клиентского приложения.

					ПГУ 2.090301.22.001 ПЗ	Лист
						25
Изм.	Лист	№ докум.	Подпись	Дата		

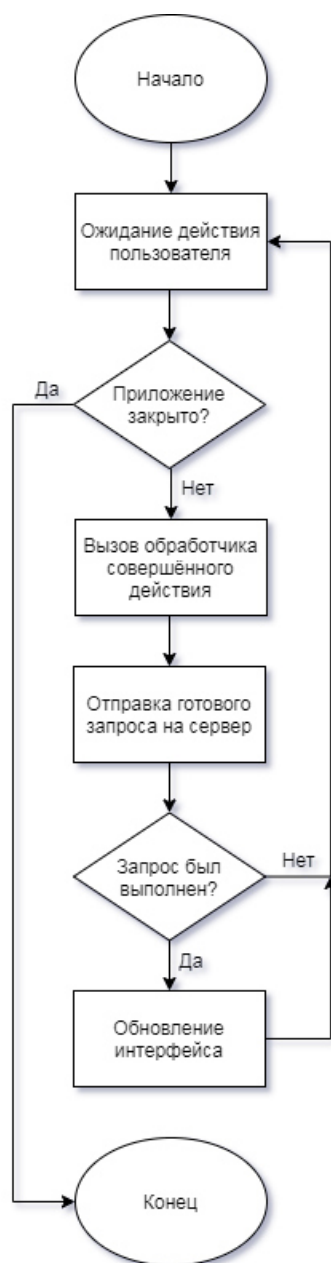


Рисунок 11 – Алгоритм работы клиентского приложения

А принимаемые клиентским приложением состояния и связи между ними можно изобразить следующей UML-диаграммой (рис. 12).



Рисунок 12 – Диаграмма состояний клиентского приложения

2.2 Проектирование серверного приложения

При запуске сервера создается сокет с портом 8080 и IP-адресом рабочей станции, на которой запущено приложение. Сокет будет переводиться в режим «прослушивания», то есть ожидания.

После установки подключения серверное приложение будет принимать запросы и строки подключения к базе посланные от клиента по установленном соединении. Затем необходимо создать подключение к базе данных с помощью указанной строки и попытаться выполнить запрос. Если запрос не будет выполнен, клиенту отправиться сообщение об ошибке. После чего сервер перейдет в режим ожидания.

Связь сервера и клиента должна быть реализована по протоколу TCP/IP. А так как для работы сервера не требуется ввода никаких данных, то его можно сделать консольным и при необходимости выводить информацию о подключенных клиентах, о выполняемых запросах или другие данные.

Схема работы серверного приложения изображена на рисунке 13.



Алгоритм работы серверного приложения

Из этого следует, что сервер может принимать состояния, показанные на рисунке 14.



Рисунок 14 – Схема состояний серверного приложения

Изм.	Лист	№ докум.	Подпись	Дата

3 Реализация проекта

Реализации проекта состоит в том, чтобы написать код описанный на предыдущий этапах проекта со всеми выделенными особенностями, в дальнейшей отладке программы и в её тестировании.

Процесс реализации следует разделить на две части: первая – это написание клиентского приложения, а вторая – создание приложения-сервера. Затем нужно провести несколько тестов, которые помогут судить о правильности работы программы.

3.1 Процесс разработки.

Для реализации проекта необходимо было выбрать наиболее подходящий интерфейс. Для написания приложений был использован язык C# и среда разработки Microsoft Visual Studio 2019.

Клиентское приложение было написано первым, чтобы создать и отладить весь необходимый функционал проекта. Программа на этом этапе взаимодействовала с базами данных без участия сервера. После проведения тестирования и получения результатов, подтверждающих правильность работы программы, требовалось создать серверное приложение.

3.1.1 Разработка клиентского приложения

Написание клиента началось с выбора интерфейса, позволяющего пользователю легко ориентироваться в функционале программы. Для этого был выбран интерфейс приложения WindowsForms, который прост в реализации и имеет широкий набор различных элементов управления, дающих возможность создать наиболее интуитивно понятный интерфейс. Все действия клиента обрабатываются механизмом событий.

Реализация клиента началось с определения вида интерфейса и создания его основной формы. В основу интерфейса лег элемент TabControl, который позволяет создать несколько вкладок, необходимых для

					ПГУ 2.090301.22.001 ПЗ	Лист
						30
Изм.	Лист	№ докум.	Подпись	Дата		

переключения между таблицами базы данных. После присоединение к базе данных в клиенте создается некоторое число вкладок, соответствующее количеству таблиц в базе. Во вкладках размещаются элементы DataGridView, в ячейки которых заносятся соответствующие данные таблицы. Для функций доступных пользователю, в верхней части окна был помещен элемент MenuStrip, который содержит несколько выпадающих меню.

На последнем этапе было добавлено окно подключения к базе данных, которое содержит поля внесения данных, которые необходимы для подключения к базе.

На UML-диаграмме (рис. 15) показан алгоритм работы реализованного приложения.

					ПГУ 2.090301.22.001 ПЗ	Лист
						31
Изм.	Лист	№ докум.	Подпись	Дата		



Рисунок 15 – Диаграмма активности клиентского приложения

На рисунке 16 показана схема обработки действий пользователя.

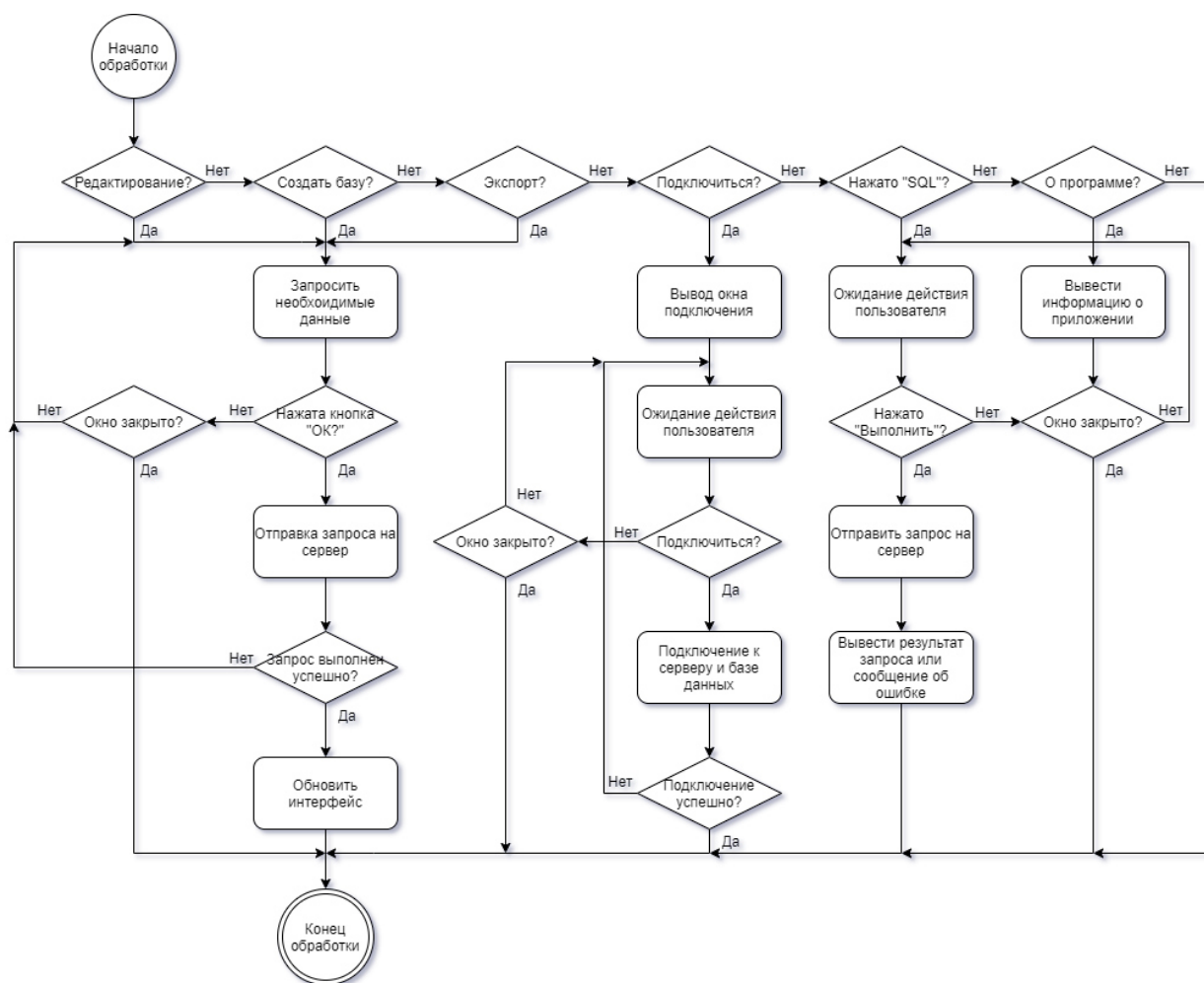


Рисунок 16 – Часть диаграммы активности клиентского приложения

3.1.2 Разработка серверного приложения

В серверном приложении нет необходимости делать интерфейс, так как там отсутствует ввод и вывод данных. Для создания сервера была выбрана консоль.

Работа серверного приложения заключается в том, что создается сокет с портом 8080 и IP-адресом станции, на которой запущено приложение. После переводится в режим ожидания подключений к нему. Когда подключение осуществляется, то сервер принимает данные, а после этого вызывается функция обработки.

Алгоритм работы серверного приложения, описанного на этапе проектирования, показан на рисунке 13.

Связи между компонентами можно показать UML-диаграммой развертывания (рис. 17).

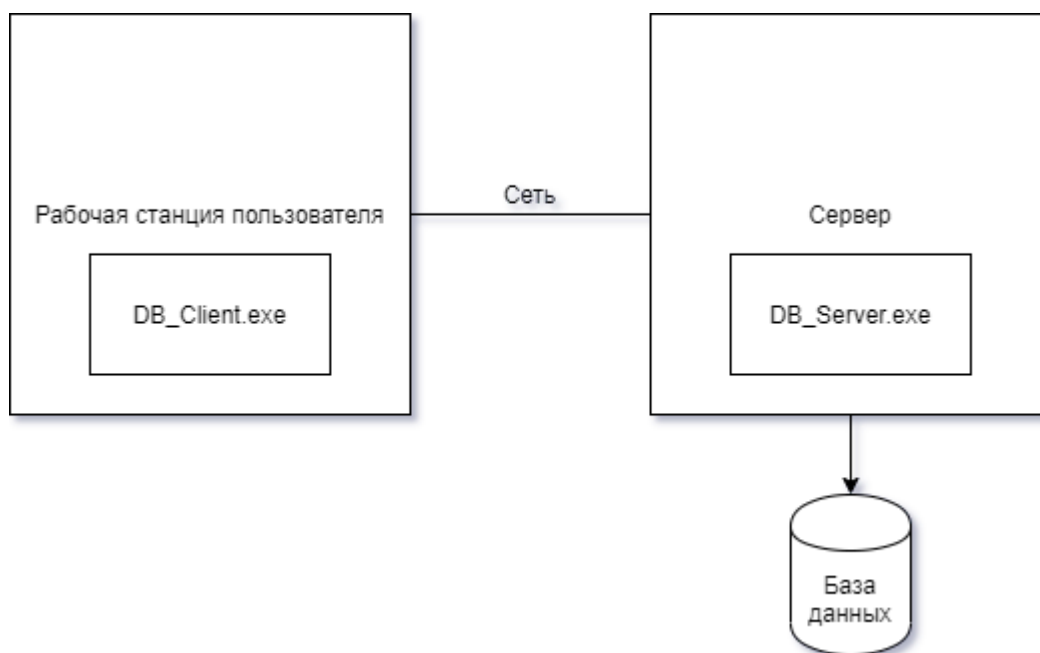


Рисунок 17 – Диаграмма развертывания

3.2 Тестирование и отладка

Приложения были написаны и отложены в среде программирования Microsoft Visual Studio 2019. Так же для тестирования клиент-серверного приложения использовались ранее обговоренные СУБД, а именно: Microsoft SQL Server 2018, MySQL Workbench 8.0 CE и Microsoft Access 2007.

Тестирование проходило, как и в процессе разработки, так и после завершения написания программы. Благодаря тестированию было выявлено и исправлено большое количество ошибок, в основном, связанных с синтаксисом языка, а так же с конструированием SQL-запросов, отправкой данных между клиентом и сервером, работы с и доступом к базам данных. Для доказательства правильности работы программы можно провести следующие тесты.

3.2.1 Тестирование возможности данных переноса между базами данных

Совершим перенос базы данных MySQL в формат Microsoft Access. В исходной базе содержится 3 таблицы: «city», «country» и «countrylanguage». Содержимое этих таблиц продемонстрировано на рисунках 18, 19 , 20 соответственно.

ID	Name	CountryCode	District	Population
1	Kabul	AFG	Kabol	1780000
2	Qandahar	AFG	Qandahar	237500
3	Herat	AFG	Herat	186800
4	Mazar-e-Sharif	AFG	Balkh	127800
5	Amsterdam	NLD	Noord-Holland	731200
6	Rotterdam	NLD	Zuid-Holland	593321
7	Haag	NLD	Zuid-Holland	440900
8	Utrecht	NLD	Utrecht	234323
9	Eindhoven	NLD	Noord-Brabant	201843
10	Tilburg	NLD	Noord-Brabant	193238
11	Groningen	NLD	Groningen	172701
12	Breda	NLD	Noord-Brabant	160398
13	Apeldoorn	NLD	Gelderland	153491
14	Nijmegen	NLD	Gelderland	152463
15	Enschede	NLD	Overijssel	149544
16	Haarlem	NLD	Noord-Holland	148772
17	Almere	NLD	Flevoland	142465
18	Arnhem	NLD	Gelderland	138020
19	Zaanstad	NLD	Noord-Holland	135621
20	Â's-Hertogen...	NLD	Noord-Brabant	129170
21	Amersfoort	NLD	Utrecht	126270
22	Maastricht	NLD	Limburg	122087
23	Dordrecht	NLD	Zuid-Holland	119811
24	Leiden	NLD	Zuid-Holland	117196
25	Haarlemmerm...	NLD	Noord-Holland	110722
26	Zoetermeer	NLD	Zuid-Holland	110214
27	Emmen	NLD	Drenthe	105853
28	Zwolle	NLD	Overijssel	105819
29	Ede	NLD	Gelderland	101574
30	Delft	NLD	Zuid-Holland	95268
31	Heerlen	NLD	Limburg	95052
32	Alkmaar	NLD	Noord-Holland	92713
33	Willemstad	ANT	Curaçao	2345
34	Tirana	ALB	Tirana	270000
35	Alger	DZA	Alger	2168000
36	Oran	DZA	Oran	609823
37	Constantine	DZA	Constantine	443727
38	Annaba	DZA	Annaba	222518

Рисунок 18 – Таблица «city»

Result Grid					
Filter Rows:					
Edit:					
Export/Import					
Code	Name	Continent	Region	SurfaceAr	
ATA	Antarctica	Antarctica	Antarctica	13120000.0	
ATF	French Southern ter...	Antarctica	Antarctica	7780.00	
BVT	Bouvet Island	Antarctica	Antarctica	59.00	
HMD	Heard Island and Mc...	Antarctica	Antarctica	359.00	
SGS	South Georgia and t...	Antarctica	Antarctica	3903.00	
AUS	Australia	Oceania	Australia and New Zealand	7741220.0	
CCK	Cocos (Keeling) Isla...	Oceania	Australia and New Zealand	14.00	
CXR	Christmas Island	Oceania	Australia and New Zealand	135.00	
NFK	Norfolk Island	Oceania	Australia and New Zealand	36.00	
NZL	New Zealand	Oceania	Australia and New Zealand	270534.00	
EST	Estonia	Europe	Baltic Countries	45227.00	
LTU	Lithuania	Europe	Baltic Countries	65301.00	
LVA	Latvia	Europe	Baltic Countries	64589.00	
GBR	United Kingdom	Europe	British Islands	242900.00	
IRL	Ireland	Europe	British Islands	70273.00	
ABW	Aruba	North America	Caribbean	193.00	
AIA	Anguilla	North America	Caribbean	96.00	
ANT	Netherlands Antilles	North America	Caribbean	800.00	
ATG	Antigua and Barbuda	North America	Caribbean	442.00	
BHS	Bahamas	North America	Caribbean	13878.00	
BRB	Barbados	North America	Caribbean	430.00	
CUB	Cuba	North America	Caribbean	110861.00	
CYM	Cayman Islands	North America	Caribbean	264.00	
DMA	Dominica	North America	Caribbean	751.00	
DOM	Dominican Republic	North America	Caribbean	48511.00	
GLP	Guadeloupe	North America	Caribbean	1705.00	
GRD	Grenada	North America	Caribbean	344.00	
HTI	Haiti	North America	Caribbean	27750.00	
JAM	Jamaica	North America	Caribbean	10990.00	
KNA	Saint Kitts and Nevis	North America	Caribbean	261.00	
LCA	Saint Lucia	North America	Caribbean	622.00	
MSR	Montserrat	North America	Caribbean	102.00	
MTQ	Martinique	North America	Caribbean	1102.00	
PRI	Puerto Rico	North America	Caribbean	8875.00	
TCA	Turks and Caicos Isl...	North America	Caribbean	430.00	
TTO	Trinidad and Tobago	North America	Caribbean	5130.00	
VCT	Saint Vincent and th...	North America	Caribbean	388.00	

Рисунок 19 – Таблица «country»

Result Grid					Filter Rows: <input type="text"/>		Edit: <input type="text"/>
	CountryCode	Language	IsOfficial	Percentage			
▶	ABW	Dutch	T	5.3			
	ABW	English	F	9.5			
	ABW	Papiamento	F	76.7			
	ABW	Spanish	F	7.4			
	AFG	Balochi	F	0.9			
	AFG	Dari	T	32.1			
	AFG	Pashto	T	52.4			
	AFG	Turkmenian	F	1.9			
	AFG	Uzbek	F	8.8			
	AGO	Ambo	F	2.4			
	AGO	Chokwe	F	4.2			
	AGO	Kongo	F	13.2			
	AGO	Luchazi	F	2.4			
	AGO	Luimbe-ng...	F	5.4			
	AGO	Luvale	F	3.6			
	AGO	Mbundu	F	21.6			
	AGO	Nyaneka-n...	F	5.4			
	AGO	Ovimbundu	F	37.2			
	AIA	English	T	0.0			
	ALB	Albaniana	T	97.9			
	ALB	Greek	F	1.8			
	ALB	Macedonian	F	0.1			
	AND	Catalan	T	32.3			
	AND	French	F	6.2			
	AND	Portuguese	F	10.8			
	AND	Spanish	F	44.6			
	ANT	Dutch	T	0.0			
	ANT	English	F	7.8			
	ANT	Papiamento	T	86.2			
	ARE	Arabic	T	42.0			
	ARE	Hindi	F	0.0			
	ARG	Indian Lan...	F	0.3			
	ARG	Italian	F	1.7			
	ARG	Spanish	T	96.8			
	ARM	Armenian	T	93.4			
	ARM	Azerbaijani	F	2.6			
	ASM	English	T	3.1			
	ASM	Samoan	T	90.6			

trylanguage 1 ×
Apply
Revert

Result Grid
Form Editor
Field Types
Query Stats
Execution Plan

Рисунок 20 – Таблица «countrylanguage»

Следует отметить, что до начала выполнения переноса в базе данных, в которую планируется совершить экспорт, отсутствуют такие таблицы (рис. 21).

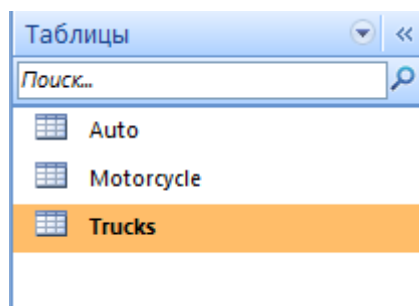


Рисунок 21 – База данных до переноса данных

Для переноса требуется открыть базу данных в разработанном приложении (рис. 22).

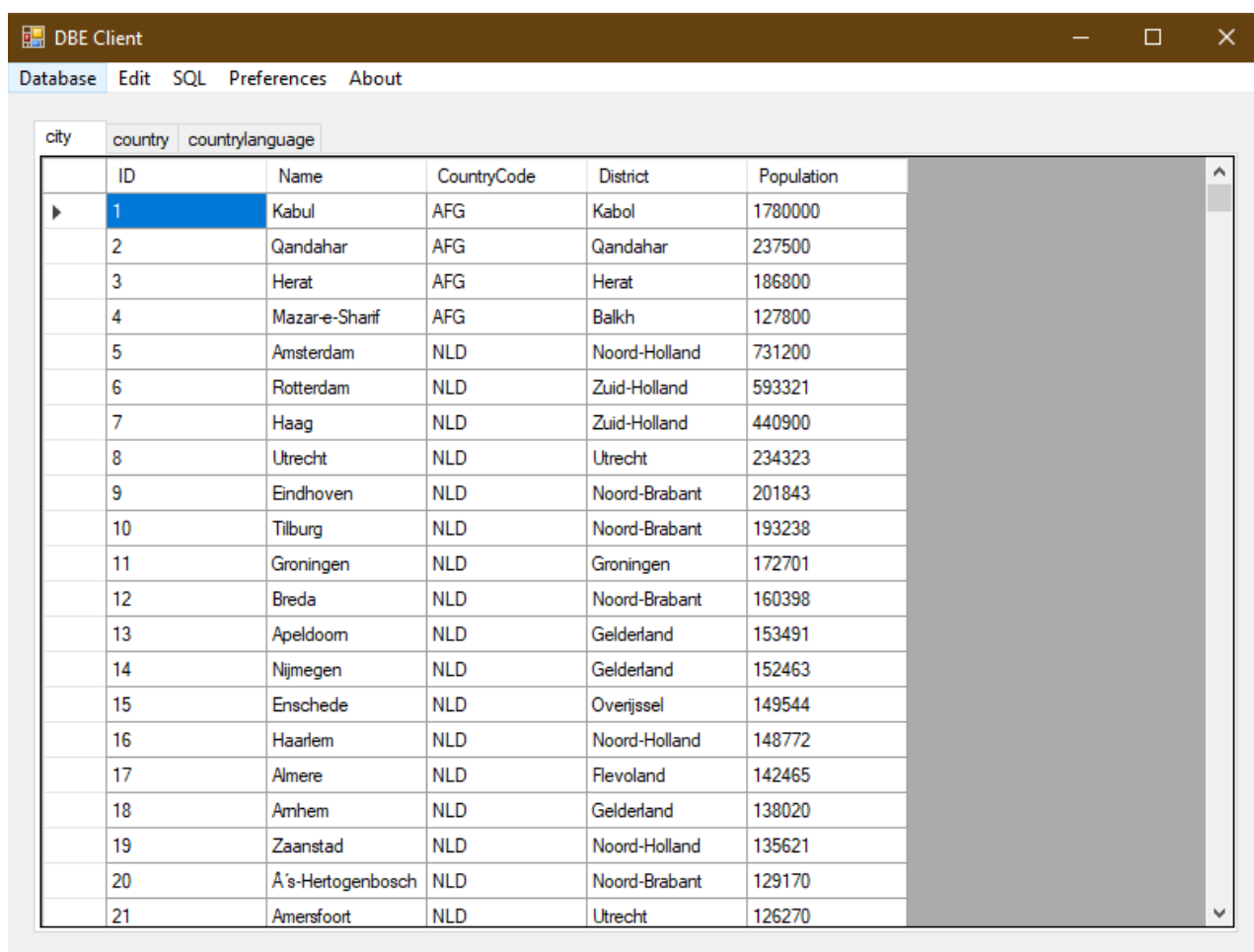


Рисунок 22 – База данных открытая в приложении

Для переноса базы в формат Microsoft Access кликнет по пункту меню «Database» и выберем «Export to...». Далее нужно выбрать в какую базу данных будет осуществлён перенос, ввести данные и нажать кнопку «Connect»(рис. 23).

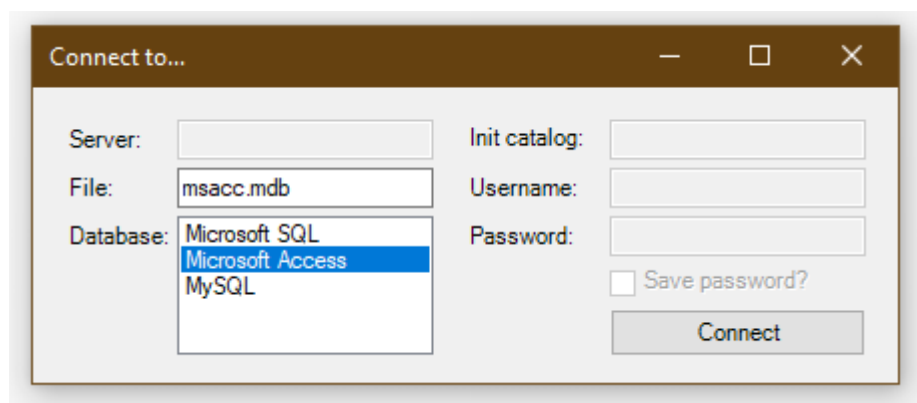


Рисунок 23 – Окно экспорта

Чтобы убедиться в правильности работы переноса откроем в СУБД Microsoft Access и в ней выберем базу данных, в которую был осуществлен перенос. В базе данных появились таблицы из базы данных MySQL (рис. 24).

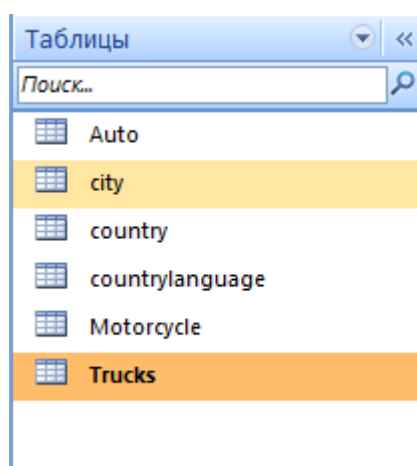


Рисунок 24 – База данных после переноса данных

Для подтверждения, что содержимое перенеслось корректно, откроем таблицы и сравним результаты (рис. 25, 26 и 27).

city				
ID	Name	CountryCod	District	Population
1	Kabul	AFG	Kabol	1780000
2	Qandahar	AFG	Qandahar	237500
3	Herat	AFG	Herat	186800
4	Mazar-e-Sharif	AFG	Balkh	127800
5	Amsterdam	NLD	Noord-Holland	731200
6	Rotterdam	NLD	Zuid-Holland	593321
7	Haag	NLD	Zuid-Holland	440900
8	Utrecht	NLD	Utrecht	234323
9	Eindhoven	NLD	Noord-Brabant	201843
10	Tilburg	NLD	Noord-Brabant	193238
11	Groningen	NLD	Groningen	172701
12	Breda	NLD	Noord-Brabant	160398
13	Apeldoorn	NLD	Gelderland	153491
14	Nijmegen	NLD	Gelderland	152463
15	Enschede	NLD	Overijssel	149544
16	Haarlem	NLD	Noord-Holland	148772
17	Almere	NLD	Flevoland	142465
18	Arnhem	NLD	Gelderland	138020
19	Zaanstad	NLD	Noord-Holland	135621
20	Assen	NLD	Noord-Brabant	129170
21	Amersfoort	NLD	Utrecht	126270
22	Maastricht	NLD	Limburg	122087
23	Dordrecht	NLD	Zuid-Holland	119811
24	Leiden	NLD	Zuid-Holland	117196
25	Haarlemmermeer	NLD	Noord-Holland	110722
26	Zoetermeer	NLD	Zuid-Holland	110214
27	Emmen	NLD	Drenthe	105853
28	Zwolle	NLD	Overijssel	105819
29	Ede	NLD	Gelderland	101574
30	Delft	NLD	Zuid-Holland	95268
31	Heerlen	NLD	Limburg	95052
32	Alkmaar	NLD	Noord-Holland	92713
33	Willemstad	ANT	Curaçao	2345

Рисунок 25 – Таблица «city»

country	Code	Name	Continent	Region	SurfaceArea	IndepYear	Population	LifeExpecta	GNP	GNPOld	LocalName	Government	HeadOfStat	Capital	Code2
ABW	Aruba	North America	Caribbean		193		103000	78,4	828	793	Aruba	Nonmetropoli	Beatrix		129 AW
AFG	Afghanistan	Asia	Southern and C		652090	1919	22720000	45,9	5976		Afghanistan/Af	Islamic Emirat	Mohammad O		1 AF
AGO	Angola	Africa	Central Africa		1246700	1975	12878000	38,3	6648	7984	Angola	Republic	JosÃ© Eduard		56 AO
AIA	Anguilla	North America	Caribbean		96		8000	76,1	63,2		Anguilla	Dependent Te	Elisabeth II		62 AI
ALB	Albania	Europe	Southern Euro		28748	1912	3401200	71,6	3205	2500	ShqipÃ«ria	Republic	Rexhep Mejd		34 AL
AND	Andorra	Europe	Southern Euro		468	1278	78000	83,5	1630		Andorra	Parliamentary			55 AD
ANT	Netherlands A	North America	Caribbean		800		217000	74,7	1941		Nederlandse A	Nonmetropoli	Beatrix		33 AN
ARE	United Arab Em	Asia	Middle East		83600	1971	2441000	74,1	37966	36846	Al-Imarat al-Ã	Emirate Feder	Zayid bin Sulta		65 AE
ARG	Argentina	South America	South America		2780400	1816	37032000	75,1	340238	323310	Argentina	Federal Repub	Fernando de la		69 AR
ARM	Armenia	Asia	Middle East		29800	1991	3520000	66,4	1813	1627	Hajastan	Republic	Robert KotÃ		126 AM
ASM	American Sam	Oceania	Polynesia		199		68000	75,1	334		Amerika Samo	US Territory	George W. Bus		54 AS
ATA	Antarctica	Antarctica	Antarctica		1,312E+07		0	0	0		Ã	Co-administra			AQ
ATF	French Southe	Antarctica	Antarctica		7780		0	0	0		Terres australe	Nonmetropoli	Jacques Chirac		TF
ATG	Antigua and B	North America	Caribbean		442	1981	68000	70,5	612	584	Antigua and B	Constitutional	Elisabeth II		63 AG
AUS	Australia	Oceania	Australia and N		7741220	1901	18886000	79,8	351182	392911	Australia	Constitutional	Elisabeth II		135 AU
AUT	Austria	Europe	Western Europ		83859	1918	8091800	77,7	211860	206025	Ãsterreich	Federal Repub	Thomas Klestil		1523 AT
AZE	Azerbaijan	Asia	Middle East		86600	1991	7734000	62,9	4127	4100	AzÃ«rbaycan	Federal Repub	HeydÃr Ãlliy		144 AZ
BDI	Burundi	Africa	Eastern Africa		27834	1962	6695000	46,2	903	982	Burundi/Uburu	Republic	Pierre Buyoya		552 BI
BEL	Belgium	Europe	Western Europ		30518	1830	10239000	77,8	249704	243948	BelgiÃ«/Belgi	Constitutional	Albert II		179 BE
BEN	Benin	Africa	Western Africa		112622	1960	6097000	50,2	2357	2141	BÃ«nin	Republic	Mathieu KÃ©		187 BJ
BFA	Burkina Faso	Africa	Western Africa		274000	1960	11937000	46,7	2425	2201	Burkina Faso	Republic	Blaise Compac		549 BF
BGD	Bangladesh	Asia	Southern and C		143998	1971	129155000	60,2	32852	31966	Bangladesh	Republic	Shahabuddin A		150 BD
BGR	Bulgaria	Europe	Eastern Europe		110994	1908	8190900	70,9	12178	10169	Balgaria	Republic	Petar Stojanov		539 BG
BHR	Bahrain	Asia	Middle East		694	1971	617000	73	6366	6097	Al-Bahrayn	Monarchy (Em	Hamad ibn Isa		149 BH
BHS	Bahamas	North America	Caribbean		13878	1973	307000	71,1	3527	3347	The Bahamas	Constitutional	Elisabeth II		148 BS
BIH	Bosnia and Her	Europe	Southern Euro		51197	1992	3972000	71,5	2841		Bosna i Herceg	Federal Repub	Ante Jelavic		201 BA
BLR	Belarus	Europe	Eastern Europe		207600	1991	10236000	68	13714		Belarus	Republic	Aljaksandr Luk		3520 BY
BLZ	Belize	North America	Central America		22696	1981	241000	70,9	630	616	Belize	Constitutional	Elisabeth II		185 BZ
BMU	Bermuda	North America	North America		53		65000	76,9	2328	2190	Bermuda	Dependent Te	Elisabeth II		191 BM
BOL	Bolivia	South America	South America		1098581	1825	8329000	63,7	8571	7967	Bolivia	Republic	Hugo BÃ¡nzer		194 BO
BRA	Brazil	South America	South America		8547403	1822	170115000	62,9	776739	804108	Brasil	Federal Repub	Fernando Henri		211 BR
BRB	Barbados	North America	Caribbean		430	1966	270000	73	2223	2186	Barbados	Constitutional	Elisabeth II		174 BB
BRN	Brunei	Asia	Southeast Asia		5765	1984	328000	73,6	11705	12460	Brunei Daruss	Monarchy (Sul	Haji Hassan al-		538 BN
BTN	Bhutan	Asia	Southern and C		47000	1910	2124000	52,4	372	383	Druk-Yul	Monarchy	Jigme Singye W		192 BT
BVT	Bouvet Island	Antarctica	Antarctica		59		0	0	0		BouvetÃya	Dependent Te	Harald V		BV
BWA	Botswana	Africa	Southern Africa		581730	1966	1622000	39,3	4834	4935	Botswana	Republic	Festus G. Mog		204 BW
CAF	Central Africa	Africa	Central Africa		622984	1960	3615000	44	1054	993	Centrafrique/T	Republic	AngÃ©-FÃ©lix F		1889 CF
CAN	Canada	North America	North America		9970610	1867	31147000	79,4	598862	625626	Canada	Constitutional	Elisabeth II		1822 CA

Рисунок 26 – Таблица «country»

countrylanguage				
CountryCod	_Language	IsOfficial	Percentage	
ABW	Dutch	T	5,3	
ABW	English	F	9,5	
ABW	Papiamentu	F	76,7	
ABW	Spanish	F	7,4	
AFG	Balochi	F	0,9	
AFG	Dari	T	32,1	
AFG	Pashto	T	52,4	
AFG	Turkmenian	F	1,9	
AFG	Uzbek	F	8,8	
AGO	Ambo	F	2,4	
AGO	Chokwe	F	4,2	
AGO	Kongo	F	13,2	
AGO	Luchazi	F	2,4	
AGO	Luimbe-ngang	F	5,4	
AGO	Luvale	F	3,6	
AGO	Mbundu	F	21,6	
AGO	Nyaneka-nkhu	F	5,4	
AGO	Ovimbundu	F	37,2	
AIA	English	T	0	
ALB	Albaniana	T	97,9	
ALB	Greek	F	1,8	
ALB	Macedonian	F	0,1	
AND	Catalan	T	32,3	
AND	French	F	6,2	
AND	Portuguese	F	10,8	
AND	Spanish	F	44,6	
ANT	Dutch	T	0	
ANT	English	F	7,8	
ANT	Papiamentu	T	86,2	
ARE	Arabic	T	42	
ARE	Hindi	F	0	
ARG	Indian Language	F	0,3	
ARG	Italian	F	1,7	

Рисунок 27 – Таблица «countrylanguage»

Исходя из результатов тестирования, можно судить о корректности работы функции переноса данных между базами данных и считать, что основная цель проекта выполнена.

3.2.2 Тестирование возможности добавления новой таблицы в базу данных

Для тестирования была использована та же база данных MySQL, что и в предыдущем тестировании. Исходная таблица выглядит следующим образом (рис. 28).

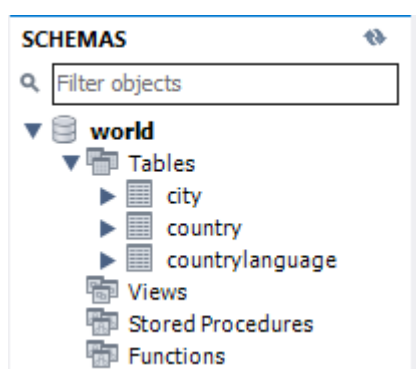


Рисунок 28 – Таблицы исходной базы данных

Откроем эту базу данных в разработанном приложении (рис. 29).

city	country	countrylanguage
ID	Name	CountryCode District Population
2	Qandahar	AFG Qandahar 237500
3	Herat	AFG Herat 186800
4	Mazare-Sharif	AFG Balkh 127800
5	Amsterdam	NLD Noord-Holland 731200
6	Rotterdam	NLD Zuid-Holland 593321
7	Haag	NLD Zuid-Holland 440900
8	Utrecht	NLD Utrecht 234323
9	Eindhoven	NLD Noord-Brabant 201843
10	Tilburg	NLD Noord-Brabant 193238
11	Groningen	NLD Groningen 172701
12	Breda	NLD Noord-Brabant 160398
13	Apeldoorn	NLD Gelderland 153491
14	Nijmegen	NLD Gelderland 152463
15	Enschede	NLD Overijssel 149544
16	Haarlem	NLD Noord-Holland 148772
17	Almere	NLD Flevoland 142465
18	Amhem	NLD Gelderland 138020
19	Zaanstad	NLD Noord-Holland 135621
20	A's-Hertogenbosch	NLD Noord-Brabant 129170
21	Amersfoort	NLD Utrecht 126270
22	Maastricht	NLD Limburg 122087

Рисунок 29 – Исходная база данных открыта в созданном приложении

Для того, чтобы добавить новую таблицу, выберем пункт меню «Edit», а далее «Add table». Откроется окно, в которое требуется ввести название новой таблицы (рис. 30).

Add table into current database

Table name:

Рисунок 30 – Добавление таблицы «population»

Далее кликнем по кнопке «Add». Окно добавления закроется, в главном окне приложения появится вкладка с новой таблицей (рис. 31).

DBE Client

Database Edit SQL Preferences About

city	country	countrylanguage	population		
	ID	Name	CountryCode	District	Population
▶	1	Kabul	AFG	Kabol	1780000
	2	Qandahar	AFG	Qandahar	237500
	3	Herat	AFG	Herat	186800
	4	Mazare-Sharif	AFG	Balkh	127800
	5	Amsterdam	NLD	Noord-Holland	731200
	6	Rotterdam	NLD	Zuid-Holland	593321
	7	Haag	NLD	Zuid-Holland	440900
	8	Utrecht	NLD	Utrecht	234323
	9	Eindhoven	NLD	Noord-Brabant	201843
	10	Tilburg	NLD	Noord-Brabant	193238
	11	Groningen	NLD	Groningen	172701
	12	Breda	NLD	Noord-Brabant	160398
	13	Apeldoorn	NLD	Gelderland	153491
	14	Nijmegen	NLD	Gelderland	152463
	15	Enschede	NLD	Overijssel	149544
	16	Haarlem	NLD	Noord-Holland	148772
	17	Almere	NLD	Flevoland	142465
	18	Amhem	NLD	Gelderland	138020
	19	Zaanstad	NLD	Noord-Holland	135621
	20	A's-Hertogenbosch	NLD	Noord-Brabant	129170
	21	Amersfoort	NLD	Utrecht	126270

Рисунок 31 – Добавлена новая вкладка с таблицей

Для подтверждения правильности работы программы откроем базу данных в СУБД MySQL (рис. 32).

SCHEMAS

Filter objects

- world
 - Tables
 - city
 - country
 - countrylanguage
 - population
 - Views
 - Stored Procedures
 - Functions

Рисунок 32 – База данных с добавленной таблицей открыта в MySQL

Новая таблица появилась в базе данных, следовательно, можно сделать вывод, что функция добавления работает правильно.

3.2.3 Тестирование возможности изменения данных в ячейки

Для проведения этого теста была выбрана база данных Microsoft Access, а конкретно база содержащая информацию о различных видах транспорта. Чтобы провести тестирование необходимо произвести редактирование какой либо ячейки. Исходная таблица «Trucks» показана на рисунке 33.

Trucks					
	id	Model	Old	Probeg	Price
	1	Volvo FH12	2010	456000	1475000
	2	DAF FT XF 105	2011	750000	9200000
	3	Scania P-340	2002	1050000	750000
	4	Ford Cargo 183	2011	431500	1265000
	5	Renault Magni	2004	360000	1150000
	6	Маз 5440	2006	510000	550000
	7	Man f2000	2000	789000	800000
	8	Iveco EuroTech	2000	560000	1500000
	9	Камаз 4308	2010	250000	890000
	10	Volvo FH13	2013	750000	3250000
*					

Рисунок 33 – Исходная таблица «Trucks»

Для изменения ячейки таблицы достаточно кликнуть по ней два раза для редактирования данных. Поменяем модель транспорта с id=1, изменив «Volvo FH12» на «Scania P400» и нажмём клавишу Enter (рис. 34).

Auto	Motorcycle	Trucks
	id	Model
...	1	Scania P400
	2	DAF FT XF 105

Рисунок 34 – Изменение данных ячейки

После изменения откроем исходную базу данных в СУБД Microsoft Access и убедимся в том, что данные ячейки действительно изменились (рис. 35).

Trucks					
	id	Model	Old	Probeg	Price
	1	Scania P400	2010	456000	1475000
	2	DAF FT XF 105	2011	750000	9200000
	3	Scania P-340	2002	1050000	750000
	4	Ford Cargo 183	2011	431500	1265000
	5	Renault Magnu	2004	360000	1150000
	6	Маз 5440	2006	510000	550000
	7	Man f2000	2000	789000	800000
	8	Iveco EuroTech	2000	560000	1500000
	9	Камаз 4308	2010	250000	890000
	10	Volvo FH13	2013	750000	3250000
*					

Рисунок 35 – Таблица "Trucks" в Microsoft Access

4 Руководство пользователя

Для работы с клиент-серверным приложением сначала необходимо запустить приложение-сервер. Как и говорилось ранее, оно является консолью и не требует ввода данных.

Запуск сервера показан на рисунке 36.

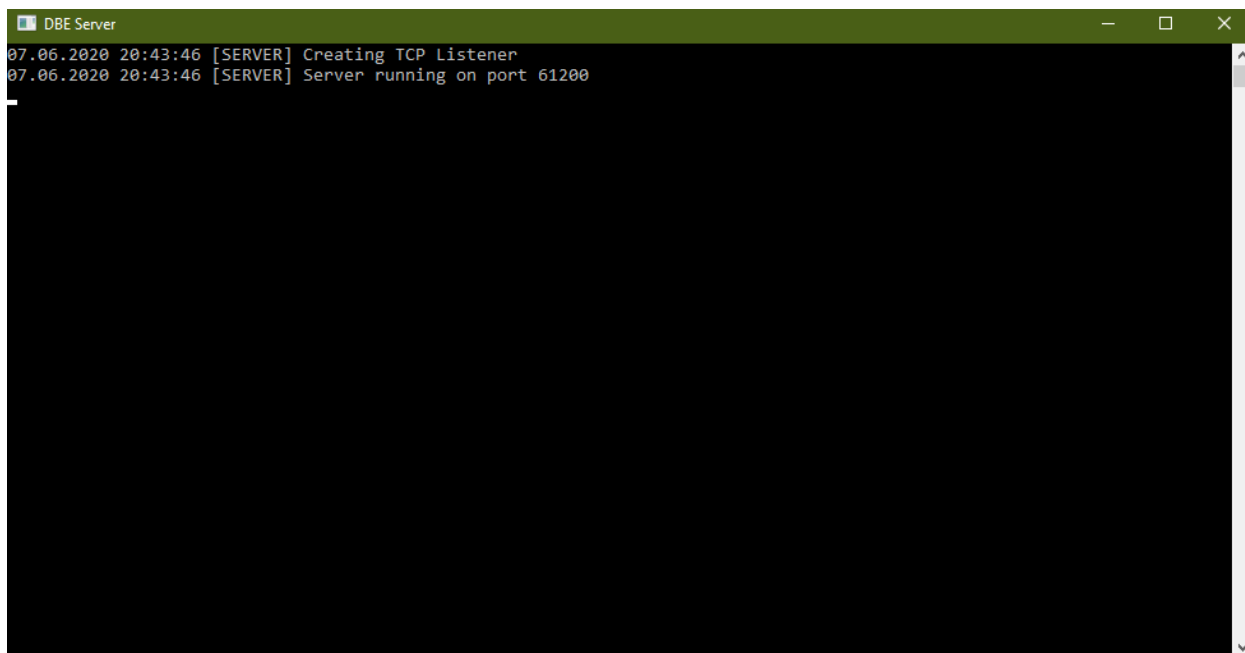


Рисунок 36 – Серверное приложение

Как только сервер будет запущен, можно запускать приложение-клиент. После его запуска открывается главное окно приложения, в котором необходимо выбрать действие, которое мы хотим сделать. Это может быть как создание базы данных, так и подключение к уже существующей (рис 37).

					ПГУ 2.090301.22.001 ПЗ	Лист
						47
Изм.	Лист	№ докум.	Подпись	Дата		

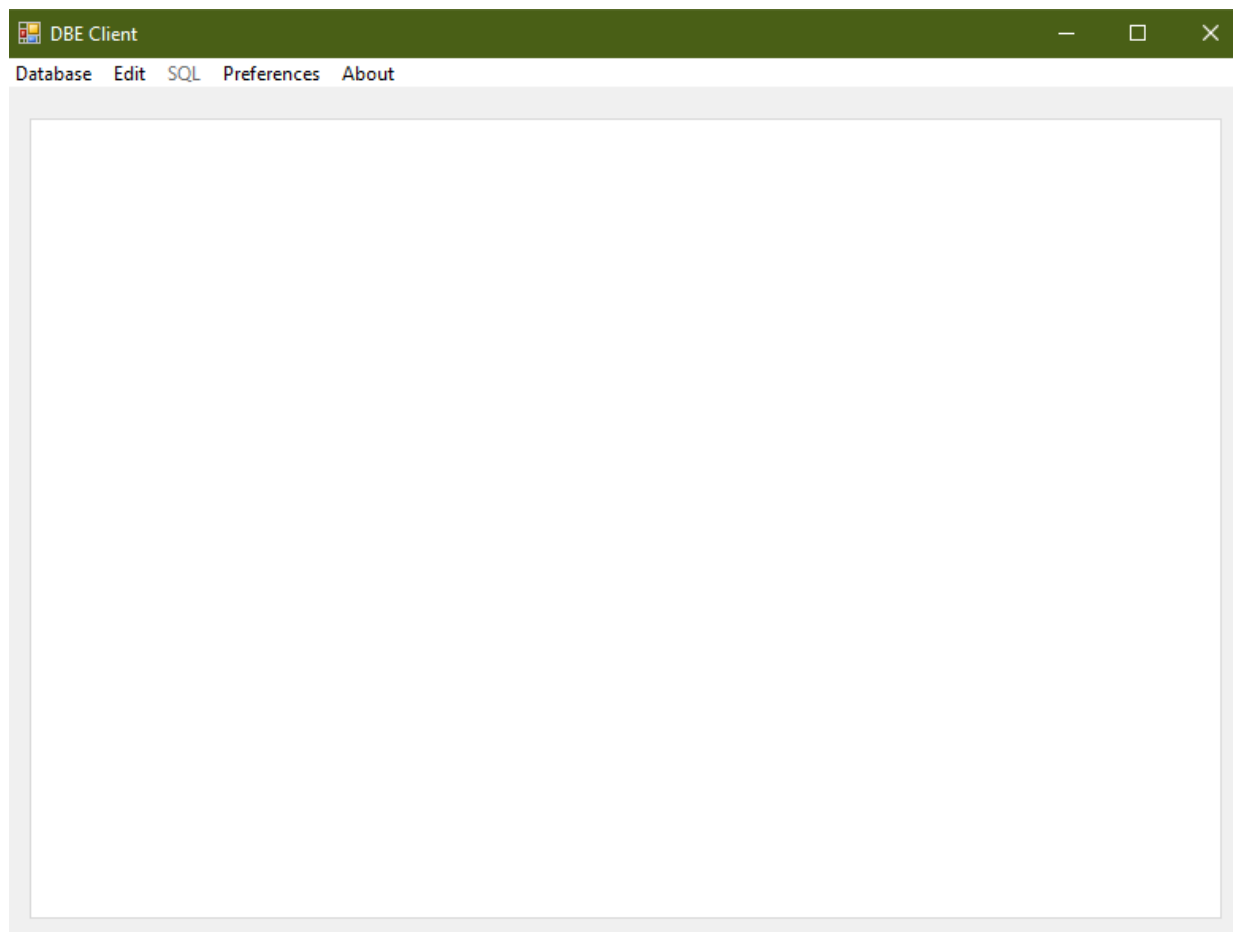


Рисунок 37 – Главное окно программы

Для подключения к базе данных необходимо перейти в меню «Database» и выбрать пункт «Connect to...» (рис. 38).

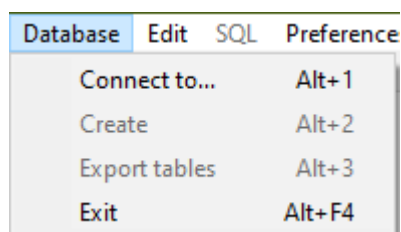


Рисунок 38 – Пункт меню «Database»

После этого откроется окно, в котором нужно указать все необходимые данные для подключения к базе данных. Чтобы выбрать типы базы данных нужно выбрать соответствующий элемент из списка поля «Database». Для подключения к базе данных Microsoft SQL Server или MySQL требуется указать IP и порт сервера, имя базы данных, имя пользователя и пароль (рис. 39).

					ПГУ 2.090301.22.001 ПЗ	Лист
						48
Изм.	Лист	№ докум.	Подпись	Дата		

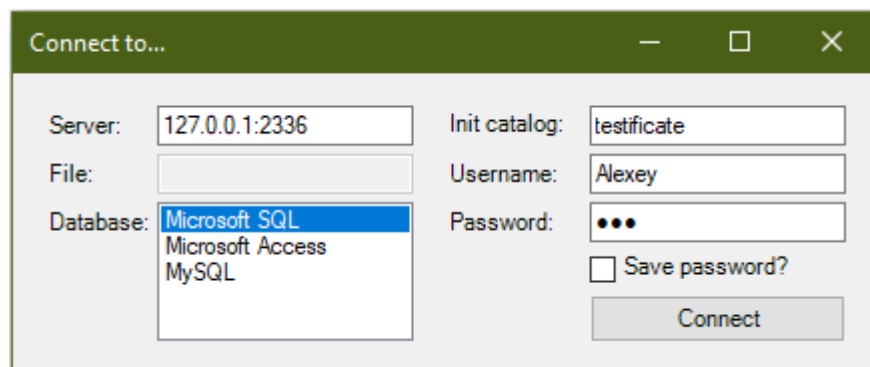


Рисунок 39 – Окно подключения к базе данных Microsoft SQL Server

Если же нужно подключиться к Microsoft Access, то ввести достаточно только имя файла, в котором хранится база данных (рис. 40).

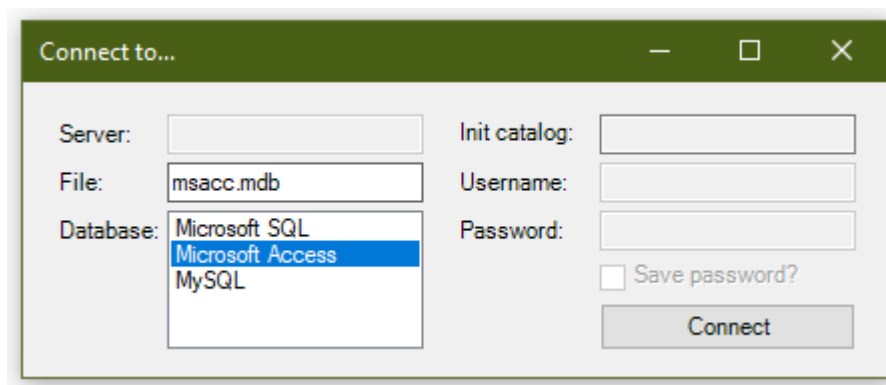


Рисунок 40 – Окно подключения к базе данных Microsoft Access

Как только будет нажата кнопка «Connect», окно подключения будет закрыто, а в основном окне приложения откроется выбранная база данных (рис. 41).

DBE Client													
Database Edit SQL Preferences About													
actor	actor_info	address	category	city	country	customer	customer_list	film	film_actor	film_category	film_list	film_text	inver
	actor_id		first_name		last_name		last_update						
▶	1		PENELOPE		GUINNESS		15.02.2006 4:34						
	2		NICK		WAHLBERG		15.02.2006 4:34						
	3		ED		CHASE		15.02.2006 4:34						
	4		JENNIFER		DAVIS		15.02.2006 4:34						
	5		JOHNNY		LOLLOBRIGIDA		15.02.2006 4:34						
	6		BETTE		NICHOLSON		15.02.2006 4:34						
	7		GRACE		MOSTEL		15.02.2006 4:34						
	8		MATTHEW		JOHANSSON		15.02.2006 4:34						
	9		JOE		SWANK		15.02.2006 4:34						
	10		CHRISTIAN		GABLE		15.02.2006 4:34						
	11		ZERO		CAGE		15.02.2006 4:34						
	12		KARL		BERRY		15.02.2006 4:34						
	13		UMA		WOOD		15.02.2006 4:34						
	14		VIVIEN		BERGEN		15.02.2006 4:34						
	15		CUBA		OLIVIER		15.02.2006 4:34						
	16		FRED		COSTNER		15.02.2006 4:34						
	17		HELEN		VOIGHT		15.02.2006 4:34						
	18		DAN		TORN		15.02.2006 4:34						
	19		BOB		FAWCETT		15.02.2006 4:34						
	20		LUCILLE		TRACY		15.02.2006 4:34						
	21		KIRSTEN		PALTROW		15.02.2006 4:34						

Рисунок 41 – Открытая база данных в приложении

Большую часть окна отводится под таблицу базы данных. Для переключения на другую таблицу достаточно нажать по вкладке с её названием. Названия вкладок соответствуют названиям таблиц.

Рассмотрим возможность редактирования базы данных и таблиц, в частности. Если необходимо добавить новую таблицу в базу данных, добавить столбец или строку в таблицу, а так же удалить, то для этого нужно перейти в меню «Edit» и выбрать желаемый элемент выпадающего меню (рис . 42).

Edit	SQL	Preferences	Abc
Add table			
Add column			
Add row			
Delete table			
Delete column			

Рисунок 42 – Пункт меню «Edit»

					ПГУ 2.090301.22.001 ПЗ	Лист
						50
Изм.	Лист	№ докум.	Подпись	Дата		

Если выбирается действие добавления новой таблицы «Add table», то открывается окно, представленное на рисунке 43, в которое вводится название будущей таблицы. После нажатия кнопки «Add» таблица добавляется в базу данных, а так же добавляется новая вкладка с этой таблицей.

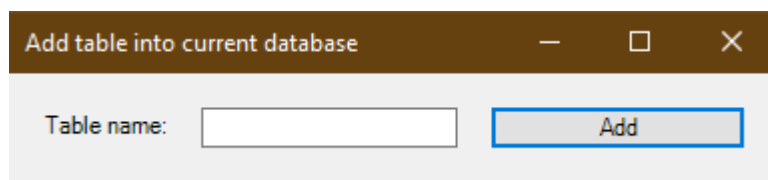


Рисунок 43 – Окно добавления новой таблицы в базу данных

Для добавления нового столбца в таблицу нужно открыть нужную вкладку, выбрать в меню «Edit» пункт «Add column» и в открытом окне задать имя столбца и его тип, а далее нажать кнопку «Add» (рис. 44).

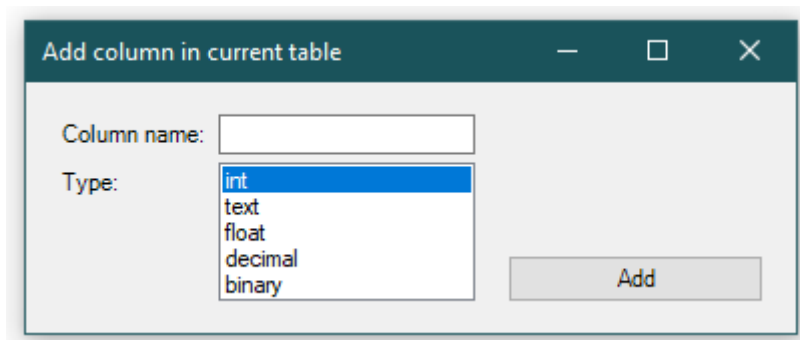


Рисунок 44 – Окно добавления нового столбца в таблицу

Чтобы удалить столбец или целую таблицу базы данных, необходимо выбрать соответствующий элемент выпадающего меню «Edit». Если пользователю требуется удалить таблицу, то необходимо открыть её, нажав на вкладку с именем таблицы, а далее выбрать «Delete table». После этого действия откроется окно подтверждения удаления (рис. 45).

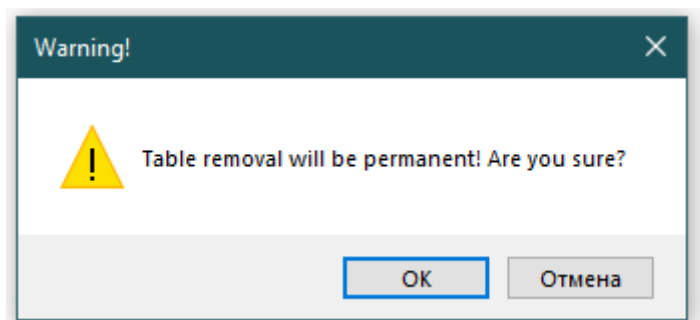


Рисунок 45 – Окно удаления таблицы из базы данных

А при удалении столбца нужно выбрать пункт «Delete column» и в открывшемся окне вписать имя столбца, которое требуется удалить в выбранной таблице (рис. 46).

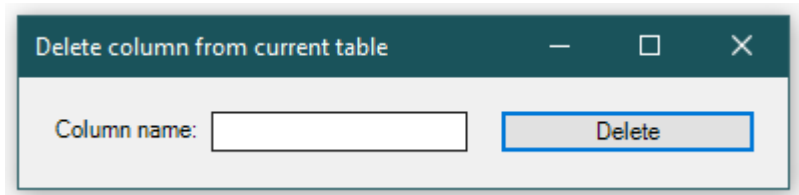


Рисунок 46 – Окно удаления столбца из таблицы

Для редактирования элемента таблицы достаточно просто кликнуть на него и начать вписывать свои данные. После завершения редактирования следует нажать клавишу Enter на клавиатуре, чтобы подтвердить ввод.

Так же есть возможность сортировки по убыванию или возрастанию. Для этого достаточно просто нажать на имя столбца и появится треугольник около названия столбца, показывающий направление сортировки. На рисунке 47 показана сортировка по одному из столбцов по убыванию.

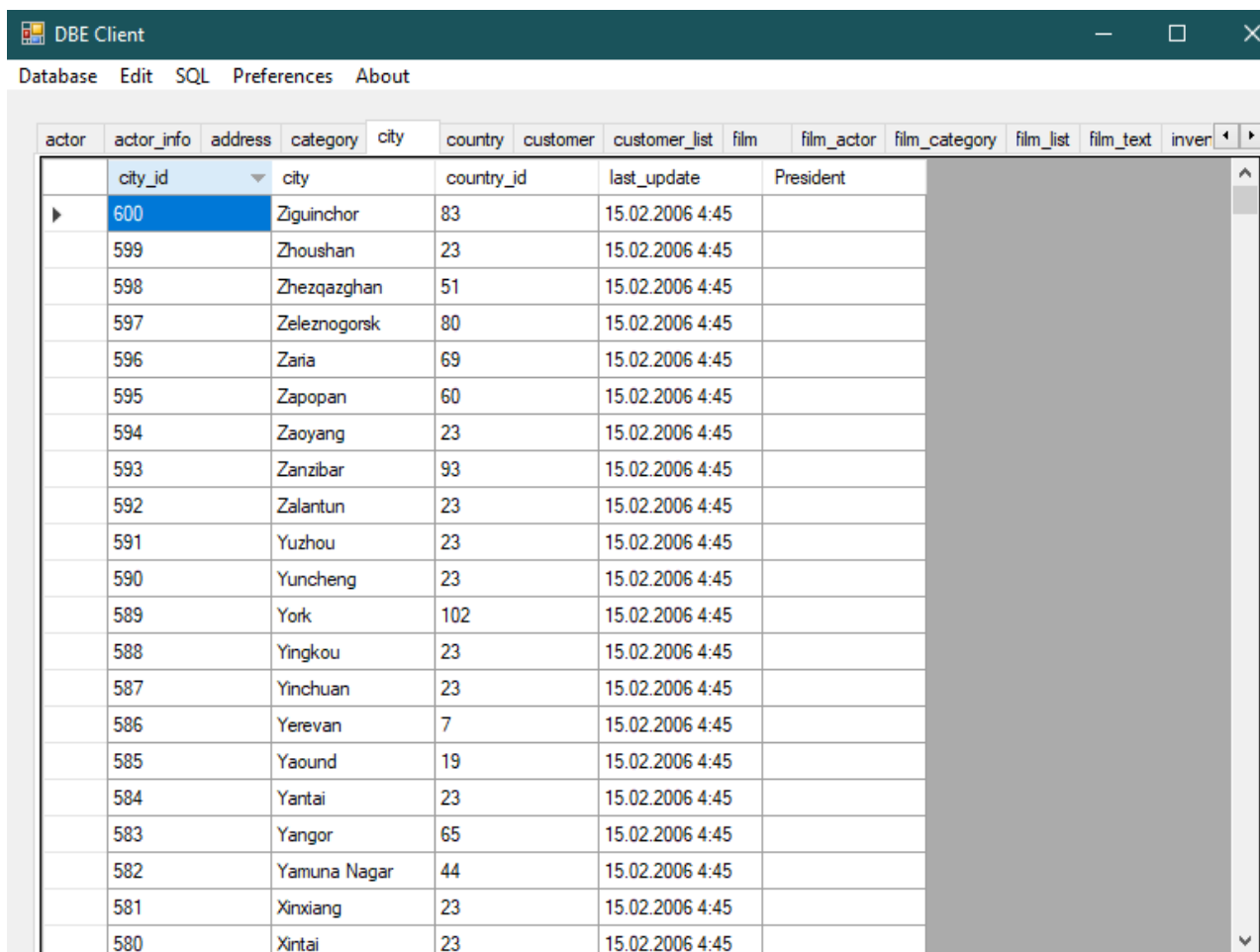


Рисунок 47 – Сортировка столбца «city_id» по убыванию

Элемент меню «SQL» вызывает окно вида, представленного на рисунке 48.

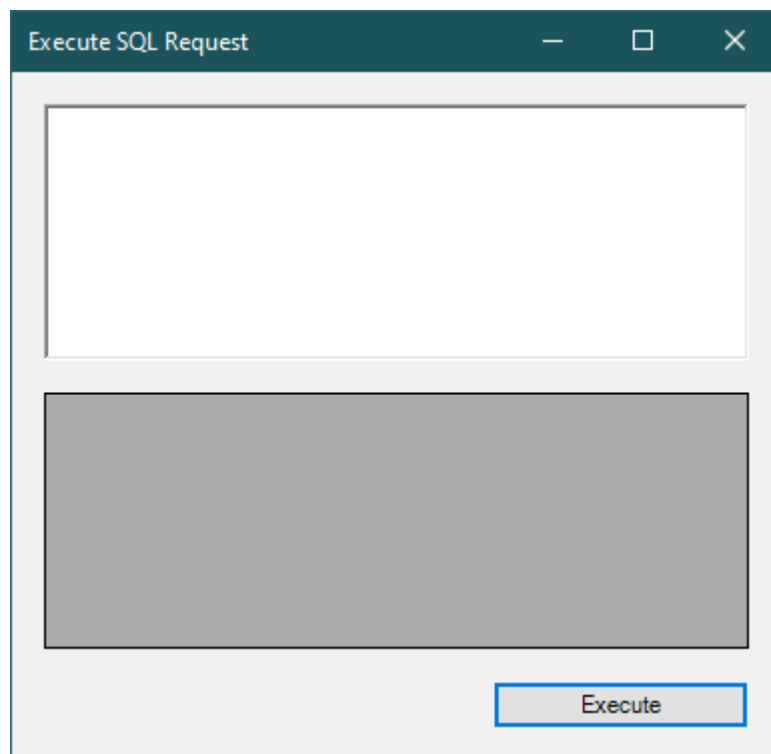


Рисунок 48 – Окно для работы с базой через SQL-запросы

Верхняя область предназначена для запроса на языке SQL, а в нижней части окна выводится запрашиваемая таблица. Эта функция предназначена для осуществления действий, не предусмотренных в других меню. Для её использования рекомендуется обладать навыками написания запросов на языке SQL. Пример запроса показан на рисунке 49.

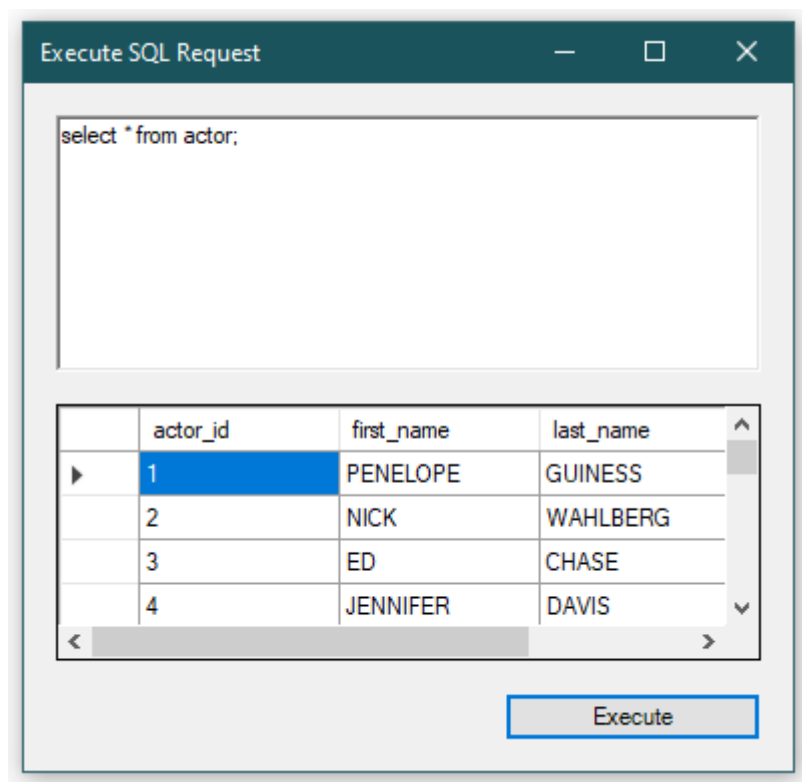


Рисунок 49 – Пример выполнения SQL-запроса в приложении

Отдельное внимание стоит уделить функции экспорта в файл, так как функция конвертирования базы данных в другой формат является основной задачей проекта. Для экспорта нужно выбрать пункт меню «Database» и выбрать элемент «Export tables». Откроется окно, идентичное окну подключения. Чтобы начать экспорт необходимо выбрать базу данных, в которую хотим совершить перенос данных, и нажать «Connect». Окно для экспорта показано на рисунке 50.

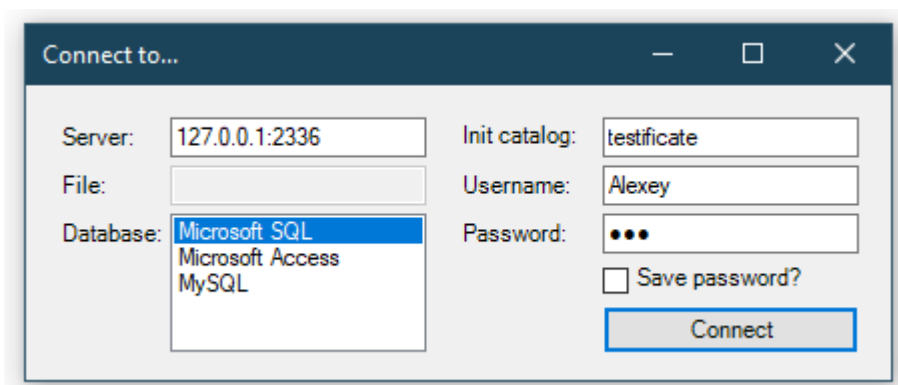


Рисунок 50 – Окно экспорта базы данных

Заключительным пунктом меню является информация о программе (рис . 51).

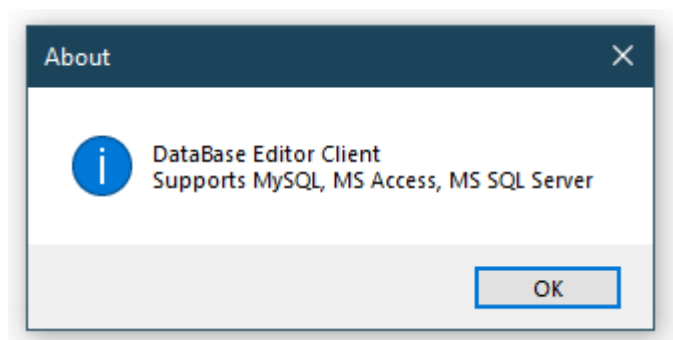


Рисунок 51 – Окно с информацией о программе

					ПГУ 2.090301.22.001 ПЗ	Лист
						55
Изм.	Лист	№ докум.	Подпись	Дата		

5 Безопасность жизнедеятельности

Дисциплина «Безопасность жизнедеятельности» считается неотъемлемой общепрофессиональной дисциплиной по всем специальностям и направлениям высшего проф. образования. Она включает в себя темы безопасного взаимодействия человека с окружающей средой, охраны труда и защиты от негативных факторов чрезвычайных ситуаций.

На протяжении всей своей истории человечество ежедневно сталкивалось с задачей обеспечения безопасности жизни. Благодаря быстрому развитию улучшилось благосостояние людей, стало лучше качество жизни и условия труда, и были достигнуты огромные успехи в области промышленности и сельского хозяйства, особенно в более развитых странах. И в связи с этим в двадцатом веке. возросло вредное воздействие человека на окружающую среду, возросли техногенные, природные и экологические катастрофы, что привело к огромным потерям среди населения и несопоставимым экономическим издержкам.

Результатом развития техносферы стало увеличения, как качества жизни людей, так и повышение уровня опасности для жизни. Антропогенные изменения в окружающей среде настолько велики, что люди становятся жертвами собственной деятельности. Смещения состояния окружающей среды в худшую сторону оказывает негативное влияние на производительность труда и отдыха, а также на здоровье.

Нередко условия труда сотрудников не соответствуют санитарно-гигиеническим нормам с точки зрения количества вредных веществ в воздухе, уровня шума, параметров микроклимата и других показателей. Опасные производственные факторы вызывают профессиональные заболевания, уровень которых в России почти удвоился за последние годы, а число людей с профессиональными заболеваниями стало самым высоким в мире.

Социально-экономические потери общества от природных и техногенных катастроф быстро растут. Без действенных мер к середине XXI

					ПГУ 2.090301.22.001 ПЗ	Лист
						56
Изм.	Лист	№ докум.	Подпись	Дата		

века сумма ущерба, нанесенного катастрофическими событиями на Земле, может превысить рост мирового валового продукта. В следствии этого одной из наиболее важных задач, влияющих на безопасность общества и его устойчивое развитие, является борьба за снижение риска природных и техногенных катастроф. Масштабы наиболее больших технологических и стихийных бедствий последних лет вполне сравнимы с чрезвычайными ситуациями военного времени.

Техногенные аварии и катастрофы, стихийные бедствия требуют как профессиональной подготовки специалистов, занимающихся предупреждением и ликвидацией их последствий, так и подготовки населения к умелым действиям в чрезвычайных ситуациях. Обучение в области противоаварийной защиты предоставляется населению, которое занимается производством и обслуживанием, учащимся образовательных учреждений начального, среднего и высшего профессионального образования, людям, не занятым производством и обслуживанием, руководителям и специалистам федеральных органов исполнительной власти, органам исполнительной власти субъектов Российской Федерации, органам местного самоуправления, учреждениям и организациям независимо от их юридической формы, а также специалистам в области противоаварийной защиты.

Компетентность людей в мире опасностей и способы защиты от них является обязательным условием безопасности жизни. Отсутствие естественных защитных механизмов требует их приобретения человеческих навыков в обнаружении опасности и использовании защитных средств. Это достижимо только в результате обучения и приобретения опыта на всех этапах образования и практической деятельности человека. Поэтому важность подготовки специалистов с высшим образованием растет.

					ПГУ 2.090301.22.001 ПЗ	Лист
						57
Изм.	Лист	№ докум.	Подпись	Дата		

Заключение

В результате работы было создано клиент-серверное приложение переноса данных между базами данных с интуитивно понятным интерфейсом. Приложение дает возможность работать с несколькими базами данных, позволяет переносить данные между ними и выполнять обработку различных данных в базах данных. Программа, полученная в результате работы, полностью соответствует поставленной задаче.

Тестирование и отладка, проводимые на нескольких баз данных, помогли обнаружить и исправить ошибки программы и получить корректные результаты её работы.

Дальнейшее улучшение программы может состоять в дополнении возможности работы с другими типами баз данных. Помимо этого, можно добавить функцию сохранения данных БД в текстовый документ, фильтрацию данных и др.

Реализация проекта помогла расширить знания и улучшить навыки по созданию подобных программ с клиент-серверной архитектурой, навыки работы с базами данных.

					ПГУ 2.090301.22.001 ПЗ	Лист
						58
Изм.	Лист	№ докум.	Подпись	Дата		

Список используемых источников

1. OLE DB или ODBC? Семь раз отметить [Электронный ресурс] - URL: <https://www.osp.ru/winitpro/2000/01/174144/>
2. Страуструп Б. «Программирование. Принципы и практика с использованием С#». 2016 г., 1328 стр.
3. OLE DB [Электронный ресурс] URL: URL: https://ru.wikipedia.org/wiki/OLE_DB
4. О модели взаимодействия клиент-сервера простыми словами. Архитектура «Клиент-сервер» с примерами. [Электронный ресурс] - URL: <http://zametkinapolyah.ru/servera-i-protokoly/o-modeli-vzaimodejstviya-klient-server-prostyimi-slovami-arxitektura-klient-server-s-primerami.html>
5. Протоколы TCP/IP простым языком. [Электронный ресурс] - URL: <https://webonto.ru/protokolyi-tcp-ip-prostyim-yazyikom/>
6. ODBC [Электронный ресурс] - URL: <https://ru.wikipedia.org/wiki/ODBC>
7. Программное управление источниками данных ODBC [Электронный ресурс] - URL: <http://www.codenet.ru/db/other/odbc/>
8. Учебное пособие БЖД [Электронный ресурс] – URL: <https://www.rea.ru/ru/org/cathedries/himfizkaf/Documents/учебное%20пособие%20БЖД.pdf>

Приложение А листинг программы сервера

Приложение А.1

Файл «Server.cs»

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.OleDb;
using System.IO;
using System.Net.Sockets;
using System.Runtime.Serialization.Formatters.Binary;
using System.Text;
using System.Threading;

namespace DB_Server
{
    class Server
    {
        private string logname = "";

        private byte[] LoadMySQLDatabase(string ip, string
port, string catalog, string user, string password)
        {
            string connection =
String.Format("Server={0};Port={1};Initial
Catalog={2};Uid={3};Pwd={4};", ip, port, catalog, user,
password);
            MySqlConnection mdb = new
MySqlConnection(connection);
            mdb.Open();
            MySqlCommand c = new MySqlCommand() { Connection
= mdb };
            c.CommandText = "show tables from " + catalog;
            MySqlDataReader odr = c.ExecuteReader();
            List<string> tables = new List<string>();
            List<DataTable> dats = new List<DataTable>();
            while (odr.Read())
            {
                tables.Add(odr[0].ToString());
            }
            odr.Close();
            for (int b = 0; b < tables.Count; b++)
            {
```

					ПГУ 2.090301.22.001 ПЗ	Лист
						60
Изм.	Лист	№ докум.	Подпись	Дата		

```

        DataTable tdt = new DataTable();
        tdt.TableName = tables[b];
        c.CommandText = "select * from " + tables[b];
        odr = c.ExecuteReader();

        for (int a = 0; a <
odr.GetSchemaTable().Rows.Count; a++)
        {

            tdt.Columns.Add(odr.GetSchemaTable().Rows[a]["ColumnName"].ToString(), (Type)odr.GetSchemaTable().Rows[a][11]);
        }
        for(int z = 0; z < int.MaxValue; z++)
        {
            if (odr.Read())
            {
                object[] bu = new
object[odr.FieldCount];
                odr.GetValues(bu);
                tdt.Rows.Add(bu);
            }
            else
            {
                break;
            }
        }
        dats.Add(tdt);
        odr.Close();
    }
    mdb.Close();
    return Serialize(dats);
} // получает данные из базы

private byte[] LoadMySQLRequest(string ip, string
port, string catalog, string user, string password, string
request) // получает данные по запросу
{
    string connection =
String.Format("Server={0};Port={1};Initial
Catalog={2};Uid={3};Pwd={4};", ip, port, catalog, user,
password);
    MySqlConnection mdb = new
MySqlConnection(connection);
    mdb.Open();

    MySqlCommand c = new MySqlCommand();
    c.Connection = mdb;

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						61
Изм.	Лист	№ докум.	Подпись	Дата		

```

        c.CommandText = request;
        MySqlDataReader odr = c.ExecuteReader();
        List<DataTable> dats = new List<DataTable>();
        DataTable tdt = new DataTable();

        try // проверяем, пустая ли таблица и
        {
            for (int a = 0; a <
odr.GetSchemaTable().Rows.Count; a++)
            {

                tdt.Columns.Add(odr.GetSchemaTable().Rows[a]["ColumnName"].ToString(), (Type)odr.GetSchemaTable().Rows[a][11]);
            }
        }
        catch
        {
            tdt.Columns.Add("empty_request"); //если
пустая (был запрос без ответа) отправляем "ПУСТО"
        }

        if (odr.HasRows)
        {

            for (int z = 0; z < int.MaxValue; z++)
            {
                if (odr.Read())
                {
                    object[] bu = new
object[odr.FieldCount];
                    odr.GetValues(bu);
                    tdt.Rows.Add(bu);
                }
                else
                {
                    break;
                }
            }
        }
        dats.Add(tdt);
        odr.Close();
        mdb.Close();
        return Serialize(dats);
    } // получает

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						62
Изм.	Лист	№ докум.	Подпись	Дата		

```

private byte[] LoadMSAccessDatabase(string filename)
{
    string connection = String.Format("Data
Source={0};Provider=Microsoft.ACE.OLEDB.12.0;Persist Security
Info=false", filename);
    OleDbConnection odb = new
OleDbConnection(connection);
    odb.Open();

    OleDbCommand c = odb.CreateCommand();
    c.Connection = odb;

    List<string> tables = new List<string>();
    List<DataTable> dats = new List<DataTable>();
    for (int b = 0; b < odb.GetSchema("Tables", new
string[] { null, null, null, "TABLE" }).Rows.Count; b++)
    {
        tables.Add(odb.GetSchema("Tables", new
string[] { null, null, null, "TABLE"
}).Rows[b][2].ToString());
    }
    for (int b = 0; b < tables.Count; b++)
    {
        DataTable tdt = new DataTable();
        tdt.TableName = tables[b];
        c.CommandText = "select * from " + tables[b];
        OleDbDataReader odr = c.ExecuteReader();

        for (int a = 0; a <
odr.GetSchemaTable().Rows.Count; a++)
        {
            tdt.Columns.Add(odr.GetSchemaTable().Rows[a]["ColumnName"].To
String(), (Type)odr.GetSchemaTable().Rows[a][5]);
        }
        for (int z = 0; z < int.MaxValue; z++)
        {
            if (odr.Read())
            {
                object[] bu = new
object[odr.FieldCount];
                odr.GetValues(bu);
                tdt.Rows.Add(bu);
            }
            else
            {

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						63
Изм.	Лист	№ докум.	Подпись	Дата		

```

        break;
    }
}
    }
    dats.Add(tdt);
    odr.Close();
}
odb.Close();
return Serialize(dats);
} // получает все данные из базы

private byte[] LoadMSAccessRequest(string filename,
string request)
{
    string connection = String.Format("Data
Source={0};Provider=Microsoft.ACE.OLEDB.12.0;Persist Security
Info=false", filename);
    OleDbConnection odb = new
OleDbConnection(connection);
    odb.Open();

    OleDbCommand c = odb.CreateCommand();
    c.Connection = odb;

    c.CommandText = request;
    OleDbDataReader odr = c.ExecuteReader();
    List<DataTable> dats = new List<DataTable>();
    DataTable tdt = new DataTable();

    try // проверяем, пустая ли таблица и
    {
        for (int a = 0; a <
odr.GetSchemaTable().Rows.Count; a++)
        {

            tdt.Columns.Add(odr.GetSchemaTable().Rows[a]["ColumnName"].ToString(), (Type)odr.GetSchemaTable().Rows[a][5]); // если
нет, отправляем таблицу
        }
    }
    catch
    {
        tdt.Columns.Add("empty_request"); //если
пустая (был запрос без ответа) отправляем "ПУСТО"
    }

    if (odr.HasRows)
    {

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						64
Изм.	Лист	№ докум.	Подпись	Дата		

```

        for (int z = 0; z < int.MaxValue; z++)
        {
            if (odr.Read())
            {
                object[] bu = new
object[odr.FieldCount];
                odr.GetValues(bu);
                tdt.Rows.Add(bu);
            }
            else
            {
                break;
            }
        }
        }
        dats.Add(tdt);
        odb.Close();
        odr.Close();
        return Serialize(dats);
    } // получает данные по запросу

    private byte[] LoadMSSQLDatabase(string ip, string
port, string catalog, string user, string password)
    {
        string connection = String.Format("Data
Source={0},{1};Provider=SQLOLEDB;Initial Catalog={2};Persist
Security Info=True;User ID={3};Password={4};", ip, port,
catalog, user, password);
        OleDbConnection odb = new
OleDbConnection(connection);
        odb.Open();

        OleDbCommand c = odb.CreateCommand();
        c.Connection = odb;

        c.CommandText = "select * from
information_schema.tables";
        OleDbDataReader odr = c.ExecuteReader();

        List<string> tables = new List<string>();
        List<DataTable> dats = new List<DataTable>();
        while (odr.Read())
        {
            tables.Add(odr[2].ToString());
        }
        odr.Close();
    }

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						65
Изм.	Лист	№ докум.	Подпись	Дата		

```

        for (int b = 0; b < tables.Count; b++)
        {
            DataTable tdt = new DataTable();
            tdt.TableName = tables[b];
            c.CommandText = "select * from " + tables[b];
            odr = c.ExecuteReader();

            for (int a = 0; a <
odr.GetSchemaTable().Rows.Count; a++)
            {

tdt.Columns.Add(odr.GetSchemaTable().Rows[a]["ColumnName"].ToString(), (Type)odr.GetSchemaTable().Rows[a][5]);
            }
            for (int z = 0; z < int.MaxValue; z++)
            {
                if (odr.Read())
                {
                    object[] bu = new
object[odr.FieldCount];
                    odr.GetValues(bu);
                    tdt.Rows.Add(bu);
                }
                else
                {
                    break;
                }
            }
            dats.Add(tdt);
            odr.Close();
        }
        odb.Close();
        return Serialize(dats);
    } // serialized, ready to use // получает все данные
из базы

private byte[] LoadMSSQLRequest(string ip, string
port, string catalog, string user, string password, string
request)
{
    string connection = String.Format("Data
Source={0},{1};Provider=SQLOLEDB;Initial Catalog={2};Persist
Security Info=True;User ID={3};Password={4};", ip, port,
catalog, user, password);
    OleDbConnection odb = new
OleDbConnection(connection);
    odb.Open();

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						66
Изм.	Лист	№ докум.	Подпись	Дата		

```

OleDbCommand c = odb.CreateCommand();
c.Connection = odb;
c.CommandText = request;
OleDbDataReader odr = c.ExecuteReader();
List<DataTable> dats = new List<DataTable>();
DataTable tdt = new DataTable();

ServerLog("Request: " + request);

try
{
    for (int a = 0; a <
odr.GetSchemaTable().Rows.Count; a++)
    {

tdt.Columns.Add(odr.GetSchemaTable().Rows[a]["ColumnName"].ToString(), (Type)odr.GetSchemaTable().Rows[a][5]);
    }
}
catch
{
    tdt.Columns.Add("empty_request");
}

if (odr.HasRows)
{
    for (int z = 0; z < int.MaxValue; z++)
    {
        if (odr.Read())
        {
            object[] bu = new
object[odr.FieldCount];
            odr.GetValues(bu);
            tdt.Rows.Add(bu);
        }
        else
        {
            break;
        }
    }
}

dats.Add(tdt);
odr.Close();
odb.Close();
return Serialize(dats);
} // получает данные по запросу

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						67
Изм.	Лист	№ докум.	Подпись	Дата		

```

        private void LoadDataInMSSQL(string ip, string port,
string catalog, string user, string password, List<DataTable>
data)
        {
            string connection = String.Format("Data
Source={0},{1};Provider=SQLOLEDB;Initial Catalog={2};Persist
Security Info=True;User ID={3};Password={4};", ip, port,
catalog, user, password);
            OleDbConnection odb = new
OleDbConnection(connection);
            odb.Open();
            OleDbCommand b = new OleDbCommand();
            b.Connection = odb;

            for (int g = 0; g < data.Count; g++)
            {
                b.CommandText = String.Format("create table
{0} ({1} {2})", data[g].TableName,
data[g].Columns[0].ColumnName,
GetSQLType(data[g].Columns[0].DataType));
                b.ExecuteNonQuery();
                for (int h = 1; h < data[g].Columns.Count;
h++)
                {
                    b.CommandText = String.Format("alter
table {0} add {1} {2}", data[g].TableName,
data[g].Columns[h].ColumnName,
GetSQLType(data[g].Columns[h].DataType));
                    b.ExecuteNonQuery();
                }
                for (int i = 0; i < data[g].Rows.Count; i++)
                {
                    string tmp = "";
                    for (int j = 0; j <
data[g].Columns.Count; j++)
                    {
                        if (data[g].Columns[j].DataType ==
typeof(Boolean))
                        {
                            tmp += String.Format("{0}",
((bool)data[g].Rows[i].ItemArray[j]) ? 1 : 0);
                        }
                        else
                        {
                            if
(data[g].Rows[i].ItemArray[j].ToString() == "")

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						68
Изм.	Лист	№ докум.	Подпись	Дата		

```

        {
            tmp += "null";
        }
        else
        {
            if
(data[g].Rows[i].ItemArray[j].ToString().Contains(","))
            {
                tmp +=
String.Format("'{0}'",
data[g].Rows[i].ItemArray[j].ToString().Replace(',', '.'));
            }
            else
            {
                if
(data[g].Rows[i].ItemArray[j].ToString().Contains("'"))
                {
                    tmp +=
String.Format("'{0}'",
data[g].Rows[i].ItemArray[j].ToString().Replace('\\', '\\'));
                }
                else
                {
                    tmp +=
String.Format("'{0}'",
data[g].Rows[i].ItemArray[j].ToString());
                }
            }
        }
        tmp += (j + 1 <
data[g].Columns.Count) ? "," : "";
    }
    b.CommandText = String.Format("insert
into {0} values ({1})", data[g].TableName, tmp);
    b.ExecuteScalar();
}

} // tested by world // загружает данные в базу

private void LoadDataInMSAccess(string filename,
List<DataTable> data)
{
    string connection = String.Format("Data
Source={0};Provider=Microsoft.ACE.OLEDB.12.0;Persist Security
Info=false", filename);

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						69
Изм.	Лист	№ докум.	Подпись	Дата		

```

        OleDbConnection odb = new
OleDbConnection(connection);
        odb.Open();
        OleDbCommand b = new OleDbCommand();
        b.Connection = odb;

        for (int g = 0; g < data.Count; g++)
        {
            b.CommandText = String.Format("create table
{0} ({1} {2})", data[g].TableName,
data[g].Columns[0].ColumnName,
GetSQLType(data[g].Columns[0].DataType));
            b.ExecuteNonQuery();
            for (int h = 1; h < data[g].Columns.Count;
h++)
            {
                try
                {
                    b.CommandText = String.Format("alter
table {0} add {1} {2}", data[g].TableName,
data[g].Columns[h].ColumnName,
GetSQLType(data[g].Columns[h].DataType));
                    b.ExecuteNonQuery();
                }
                catch
                {
                    b.CommandText = String.Format("alter
table {0} add _{1} {2}", data[g].TableName,
data[g].Columns[h].ColumnName,
GetSQLType(data[g].Columns[h].DataType));
                    b.ExecuteNonQuery();
                }
            }
            for (int i = 0; i < data[g].Rows.Count; i++)
            {
                string tmp = "";
                for (int j = 0; j <
data[g].Columns.Count; j++)
                {
                    if (data[g].Columns[j].DataType ==
typeof(Boolean))
                    {
                        tmp += String.Format("{0}",
((bool)data[g].Rows[i].ItemArray[j]) ? 1 : 0);
                    }
                    else
                    {

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						70
Изм.	Лист	№ докум.	Подпись	Дата		

```

                if
(data[g].Rows[i].ItemArray[j].ToString() == "")
                {
                    tmp += "null";
                }
                else
                {
                    if
(data[g].Rows[i].ItemArray[j].ToString().Contains("'"))
                    {
                        tmp +=
String.Format("'{0}'",
data[g].Rows[i].ItemArray[j].ToString().Replace('\'', '`'));
                    }
                    else
                    {
                        tmp +=
String.Format("'{0}'",
data[g].Rows[i].ItemArray[j].ToString());
                    }
                }
            }
            tmp += (j + 1 <
data[g].Columns.Count) ? "," : "";
        }
        b.CommandText = String.Format("insert
into {0} values ({1})", data[g].TableName, tmp);
        b.ExecuteScalar();
    }
} // tested by world // загружает данные в базу

private void LoadDataInMySQL(string ip, string port,
string catalog, string user, string password, List<DataTable>
data)
{
    string connection =
String.Format("Server={0};Port={1};Initial
Catalog={2};Uid={3};Pwd={4};", ip, port, catalog, user,
password);
    MySqlConnection mdb = new
MySqlConnection(connection);
    mdb.Open();

    MySqlCommand b = mdb.CreateCommand();
    MySqlTransaction tr;

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						71
Изм.	Лист	№ докум.	Подпись	Дата		

```

        tr = mdb.BeginTransaction();
        b.Connection = mdb;
        b.Transaction = tr;

        for (int g = 0; g < data.Count; g++)
        {
            b.CommandText = String.Format("create table
{0} ({1} {2});", data[g].TableName,
data[g].Columns[0].ColumnName,
GetSQLType(data[g].Columns[0].DataType));
            b.ExecuteNonQuery();

            for (int h = 1; h < data[g].Columns.Count;
h++)
            {
                b.CommandText = String.Format("alter
table {0} add {1} {2};", data[g].TableName,
(data[g].Columns[h].ColumnName.StartsWith("_")) ?
data[g].Columns[h].ColumnName : "_" +
data[g].Columns[h].ColumnName,
GetSQLType(data[g].Columns[h].DataType));
                b.ExecuteNonQuery();
            }
            b.CommandText = String.Format("insert into
{0} values ", data[g].TableName);
            for (int i = 0; i < data[g].Rows.Count; i++)
            {
                string tmp = "";
                for (int j = 0; j <
data[g].Columns.Count; j++)
                {
                    if (data[g].Columns[j].DataType ==
typeof(Boolean))
                    {
                        tmp += String.Format("{0}",
((bool)data[g].Rows[i].ItemArray[j]) ? 1 : 0);
                    }
                    else
                    {
                        if
(data[g].Rows[i].ItemArray[j].ToString() == "")
                        {
                            tmp += "null";
                        }
                        else
                        {

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						72
Изм.	Лист	№ докум.	Подпись	Дата		

```

                if
(data[g].Rows[i].ItemArray[j].ToString().Contains(","))
                {
                    tmp +=
String.Format("'{0}'",
data[g].Rows[i].ItemArray[j].ToString().Replace(',', ' '));
                }
                else
                {
                    if
(data[g].Rows[i].ItemArray[j].ToString().Contains("'"))
                    {
                        tmp +=
String.Format("'{0}'",
data[g].Rows[i].ItemArray[j].ToString().Replace('\'', ' `'));
                    }
                    else
                    {
                        tmp +=
String.Format("'{0}'",
data[g].Rows[i].ItemArray[j].ToString());
                    }
                }
            }
            tmp += (j + 1 <
data[g].Columns.Count) ? "," : "";
        }

        b.CommandText += String.Format("({0})",
tmp);

        b.CommandText += (i + 1 <
data[g].Rows.Count) ? "," : ";";
    }
    b.ExecuteNonQuery();
}
tr.Commit();
} // tested by world and testificate // загружает
данные в базу

private string GetSQLType(Type v)
{
    switch (v.ToString()) // types, that supports by
MS SQL, MySQL and MS Access
    {
        case "System.Int32": { return "int"; } // + +
+

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						73
Изм.	Лист	№ докум.	Подпись	Дата		

```

        case "System.String": { return "text"; } // +
+ +
        case "System.Double": { return "float"; } //
+ + +
        case "System.Boolean": { return "bit"; } // +
+ +
        case "System.Byte[]": { return "binary"; } //
+ + +
        case "System.DateTime": { return "datetime";
} // + + +
        case "System.Single": { return "real"; } // +
+ +
        default: { return "text"; }
    }
} // конвертирует .NET типы в совместимые со всеми
тремя базами SQL типы

private byte[] Serialize(object o)
{
    Stream tmp = new MemoryStream();
    BinaryFormatter format = new BinaryFormatter();
    format.Serialize(tmp, o);
    tmp.Position = 0;
    byte[] buffer = new byte[tmp.Length];
    tmp.Read(buffer, 0, (int)tmp.Length);
    tmp.Dispose();
    return buffer;
} // превращает список таблиц в массив байт

private void ServerLog(object value)
{
    try
    {
        StreamWriter k = File.AppendText(logname);
        k.WriteLine("{0} [SERVER] {1}", DateTime.Now,
value);
        Console.WriteLine("{0} [SERVER] {1}",
DateTime.Now, value);
        k.Close();
    }
    catch
    {
        ServerLog(value);
    }
} // пишем лог в консоль и в файл

private void HandleRequest(object o)

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						74
Изм.	Лист	№ докум.	Подпись	Дата		

```

        {
            TcpClient client = (TcpClient)o;

            byte[] size = new byte[4];
            client.Client.Receive(size); // получаем размер
пакета

            byte[] a = new byte[BitConverter.ToInt32(size,
0)];

            client.Client.Receive(a); // получаем сам пакет

            string[] com =
UnicodeEncoding.Unicode.GetString(a).Split(';');

            ServerLog("Request received from " +
client.Client.RemoteEndPoint);
            try
            {
                byte[] answer = new byte[1]; // string by
connection form
<db>;<ip:port>;<filename>;<username>;<password>;<catalog>;<re
quest>;

                switch (com[0])
                {
                    // все 3 ветки работают одинаково, если
длина запроса 0, то грузит всю базу данных
                    case "0": // ms sql server connection
                        {
                            ServerLog("Getting MS SQL
Data...");

                            string[] ip = com[1].Split(':');
                            answer = (com[6].Length > 0) ?
LoadMSSQLRequest(ip[0], ip[1], com[5], com[3], com[4],
com[6]) : LoadMSSQLDatabase(ip[0], ip[1], com[5], com[3],
com[4]);

                            break;
                        }
                    case "1": // ms access data
                        {
                            if (com[6].Contains("create
database")) // Ms access не умеет самостоятельно создавать
базы и это такой "костыль" !!!!!НЕ УДАЛЯТЬ template.resx!!!!
                            {
                                File.Copy("template.resx",
com[6].Split(' ')[2]);

                                answer =
UnicodeEncoding.Unicode.GetBytes("Data created");

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						75
Изм.	Лист	№ докум.	Подпись	Дата		

```

    }
    else
    {
        ServerLog("Getting MS Access
Data...");

        answer = (com[6].Length > 0)
? LoadMSAccessRequest(com[2], com[6]) :
LoadMSAccessDatabase(com[2]);
    }
    break;
}
case "2": // my sql server connection
{
    string[] ip = com[1].Split(':');
    ServerLog("Getting MySQL
Data...");

    answer = (com[6].Length > 0) ?
LoadMySQLRequest(ip[0], ip[1], com[5], com[3], com[4],
com[6]) : LoadMySQLDatabase(ip[0], ip[1], com[5], com[3],
com[4]);

    break;
}
case "3": // особый запрос - запрос на
экспорт
{
    byte[] data = new byte[0];
    // получаем инфу откуда
экспортировать и что

    switch (com[6])
    {
        case "0":
        {
            string[] ip =
com[1].Split(':');

            data =
LoadMSSQLDatabase(ip[0], ip[1], com[5], com[3], com[4]);
            ServerLog("Loading MS
SQL Data...");

            break;
        }
        case "1":
        {
            data =

LoadMSAccessDatabase(com[2]);

            ServerLog("Loading MS
Access Data...");

            break;

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						76
Изм.	Лист	№ докум.	Подпись	Дата		

```

        }
        case "2":
        {
            string[] ip =
com[1].Split(':');

            data =
LoadMySQLDatabase(ip[0], ip[1], com[5], com[3], com[4]);
ServerLog("Loading
MySQL Data...");

            break;
        }
    }
    // формируем списки таблиц
    ServerLog("Preparing data to
export...");

    Stream stream = new
MemoryStream(data);

    BinaryFormatter format = new
BinaryFormatter();

    stream.Position = 0;
    List<DataTable> exdata =
(List<DataTable>)format.Deserialize(stream);
    // раскидываем в зависимости от
данных в пакете

    switch (com[7])
    {
        case "0":
        {
            ServerLog("Exporting
to MS SQL...");

            string[] ip =
com[8].Split(':');

            LoadDataInMSSQL(ip[0], ip[1], com[12], com[10], com[11],
exdata);

            break;
        }
        case "1":
        {
            ServerLog("Exporting
to MS Access...");

            string[] ip =
com[1].Split(':');

            LoadDataInMSAccess(com[9], exdata);

            break;
        }
    }

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						77
Изм.	Лист	№ докум.	Подпись	Дата		

```

        case "2":
        {
            ServerLog("Exporting
to MySQL...");

            string[] ip =
com[8].Split(':');

LoadDataInMySQL(ip[0], ip[1], com[12], com[10], com[11],
exdata);

            break;
        }
    }
    break;
}

//отправляем список таблиц или 1, если успех

client.Client.Send(BitConverter.GetBytes(answer.Length));
Thread.Sleep(1);
client.Client.Send(answer);
}
catch(Exception e)
{
    // отправляем 0, потом размер исключения и
потом исключение клиенту
    client.Client.Send(BitConverter.GetBytes(0));

    client.Client.Send(BitConverter.GetBytes(UnicodeEncoding.Unic
ode.GetByteCount(e.Message)));
    Thread.Sleep(1);

    client.Client.Send(UnicodeEncoding.Unicode.GetBytes(e.Message
));
}
// пишем в лог, что всё закончилось хорошо
ServerLog("Request successfully handled");
} // обработка запроса

public void Start(string logname)
{
    this.logname = logname.Replace(".",
"").Replace(":", "").Replace(" ", "_") + ".log";
    int port = 61200;
    ServerLog("Creating TCP Listener");
    TcpListener tcp = TcpListener.Create(port);
    tcp.Start();
    // слушаем порт 61200

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						78
Изм.	Лист	№ докум.	Подпись	Дата		

```

        ServerLog("Server running on port " + port); //
ЛОГ
        while (true)
        {
            TcpClient client = tcp.AcceptTcpClient(); //
ждем клиента
            ServerLog("Client connecting...");
            Thread c = new Thread(new
ParameterizedThreadStart(HandleRequest)); // кидаем клиента в
НОВЫЙ ПОТОК и ждём новых
            c.Start(client);
        }
    } // страт сервера
}

```

Приложение А.2

Файл «Core.cs»

```

using System;

namespace DB_Server
{
    class Core
    {
        static void Main()
        {
            Console.Title = "DBE Server";
            Server server = new Server(); // объект класса сервера
            server.Start(DateTime.Now.ToString()); // стартуем сервер,
передаем дату и время для логов
        }
    }
}

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						79
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение В листинг программы клиента

Приложение В.1

Файл «MainForms.cs»

```
using System;
using System.Drawing;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Collections.Generic;
using System.Windows.Forms;
using System.Data;
using System.Runtime.Serialization.Formatters.Binary;
using System.IO;

namespace DB_Client
{
    class MainForm : Form
    {
        private Dictionary<string, DataGridView> table_list = new
Dictionary<string, DataGridView>();
        private MainMenu menu = new MainMenu();

        private MenuItem file = new MenuItem("Database");
        private MenuItem connect = new MenuItem("Connect to...") {
Shortcut = Shortcut.Alt1, ShowShortcut = true };
        private MenuItem create = new MenuItem("Create") { Shortcut =
Shortcut.Alt2, ShowShortcut = true };
        private MenuItem export = new MenuItem("Export tables") {
Shortcut = Shortcut.Alt3, ShowShortcut = true };
        private MenuItem exit = new MenuItem("Exit") { Shortcut =
Shortcut.AltF4, ShowShortcut = true };
        private MenuItem edit = new MenuItem("Edit");
        private MenuItem addtable = new MenuItem("Add table");
        private MenuItem addcolumn = new MenuItem("Add column");
        private MenuItem addrow = new MenuItem("Add row");
        private MenuItem deletetable = new MenuItem("Delete table");
        private MenuItem deletecolumn = new MenuItem("Delete column");
        private MenuItem prefs = new MenuItem("Preferences");
        private MenuItem sql = new MenuItem("SQL");
        private MenuItem about = new MenuItem("About");
        private ConnectForm connectform = new ConnectForm();
        private AddColumnForm columnform = new AddColumnForm();
        private AddRowForm addrowform = new AddRowForm();
        private DeleteColumnForm delcolumnform = new DeleteColumnForm();
        private AddTableForm addtableform = new AddTableForm();
        private AddDataForm adddataform = new AddDataForm();
        private PrefsForm prefsform = new PrefsForm();
    }
}
```

					ПГУ 2.090301.22.001 ПЗ	Лист
						80
Изм.	Лист	№ докум.	Подпись	Дата		

```
private TabControl tabs = new TabControl() { Size = new
Size(754, 508), Location = new Point(16, 16) };
```

```
public MainForm()
{
    Text = "DBE Client";
    Size = new Size(800, 600);
    FormBorderStyle = FormBorderStyle.FixedSingle;
    Menu = menu;

    SubscribeEvents();

    addrow.Enabled = false;
    addtable.Enabled = false;
    addcolumn.Enabled = false;
    deletetable.Enabled = false;
    deletecolumn.Enabled = false;
    create.Enabled = false;
    export.Enabled = false;
    sql.Enabled = false;

    file.MenuItems.Add(connect);
    file.MenuItems.Add(create);
    file.MenuItems.Add(export);
    file.MenuItems.Add(exit);
    edit.MenuItems.Add(addtable);
    edit.MenuItems.Add(addcolumn);
    edit.MenuItems.Add(addrow);
    edit.MenuItems.Add(deletetable);
    edit.MenuItems.Add(deletecolumn);

    Menu.MenuItems.Add(file);
    Menu.MenuItems.Add(edit);
    Menu.MenuItems.Add(sql);
    Menu.MenuItems.Add(prefs);
    Menu.MenuItems.Add(about);
    Controls.Add(tabs);
}
```

```
private void SubscribeEvents()
{
    Load += MainFormLoad;
    exit.Click += Exit;
    connect.Click += Connect;
    about.Click += About;
    addcolumn.Click += AddColumn;
    deletecolumn.Click += DeleteColumn;
    addrow.Click += AddRow;
    addtable.Click += AddTable;
    deletetable.Click += DeleteTable;
    create.Click += CreateData;
    export.Click += ExportData;
```

					ПГУ 2.090301.22.001 ПЗ	Лист
						81
Изм.	Лист	№ докум.	Подпись	Дата		

```

        sql.Click += CreateSQLRequest;
        prefs.Click += Preferences;
    }    // event list

private void Preferences(object sender, EventArgs e)
{
    prefsform.ShowDialog();
} // DBE server prefs

private void CreateSQLRequest(object sender, EventArgs e)
{
    SqlRequestForm sqlrequestform = new SqlRequestForm();
    sqlrequestform.context = this;
    sqlrequestform.Show();
} // ready to use (todo highlight)

private void ExportData(object sender, EventArgs e)
{
    string connection = "3;";
    for (int d = 0; d < 5; d++)
    {
        connection += connectform.Controls[d].Text + ";";
    }
    connection += connectform.database.SelectedIndex + ";";
    int sel = connectform.database.SelectedIndex;
    object sels = connectform.database.SelectedItem;
    string fsel = connectform.file.Text;
    if (connectform.ShowDialog() == DialogResult.OK)
    {
        if ((connectform.database.SelectedIndex == sel &&
connectform.database.SelectedItem == sels) || connectform.file.Text ==
fsel)
        {
            MessageBox.Show("Export to the same database will
cause table conflicts", "General Exception", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
        else
        {
            connection += connectform.database.SelectedIndex +
";";
            for (int d = 0; d < 5; d++)
            {
                connection += connectform.Controls[d].Text +
";";
            }
            SendToServer(connection); // send connection string
        }
    }
} // all works

private void CreateData(object sender, EventArgs e)

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						82
Изм.	Лист	№ докум.	Подпись	Дата		

```

{
    if (adddataform.ShowDialog() == DialogResult.OK)
    {
        object[] ans = LoadRequest(String.Format("create
database {0}", adddataform.name.Text));
        if ((bool)ans[1])
        {
            MessageBox.Show((string)ans[0], "Unable to add
database", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        else
        {
            if (connectform.database.SelectedIndex == 1)
            {
                connectform.Controls[1].Text =
adddataform.name.Text;
            }
            else
            {
                connectform.Controls[4].Text =
adddataform.name.Text;
            }
            ans = LoadAllTables();
            if (!(bool)ans[1])
            {
                PresentTables(GetTables(ans[0]));
            }
            else
            {
                MessageBox.Show((string)ans[0], "General
Exception", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }
} // ready to use+

private void DeleteTable(object sender, EventArgs e)
{
    if (MessageBox.Show("Table removal will be permanent! Are
you sure?", "Warning!", MessageBoxButtons.OKCancel,
MessageBoxIcon.Warning) == DialogResult.OK)
    {
        object[] ans = LoadRequest(String.Format("drop table
{0}", tabs.SelectedTab.Text));
        if ((bool)ans[1])
        {
            MessageBox.Show((string)ans[0], "Unable to delete
table", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        else
        {
            ans = LoadAllTables();
        }
    }
}

```

```

        if (!(bool)ans[1])
        {
            PresentTables(GetTables(ans[0]));
        }
        else
        {
            MessageBox.Show((string)ans[0], "General
Exception", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
} // ready to use+

private void AddTable(object sender, EventArgs e)
{
    if (addtableform.ShowDialog() == DialogResult.OK)
    {
        object[] ans = LoadRequest(String.Format("create table
{0} (id int)", addtableform.name.Text));
        if ((bool)ans[1])
        {
            MessageBox.Show((string)ans[0], "Unable to add
table", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        else
        {
            ans = LoadAllTables();
            if (!(bool)ans[1])
            {
                PresentTables(GetTables(ans[0]));
            }
            else
            {
                MessageBox.Show((string)ans[0], "General
Exception", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }
} // ready to use+

private void AddRow(object sender, EventArgs e)
{
    addrowform.Update(table_list[tabs.SelectedTab.Text]);
    if (addrowform.ShowDialog() == DialogResult.OK)
    {
        string values = "";

        for (int d = 0; d <
table_list[tabs.SelectedTab.Text].ColumnCount; d++)
        {
            values += String.Format("' {0}',",
addrowform.Controls[d].Text);

```

```

    }
    values = values.Substring(0, values.Length - 1);

    object[] ans = LoadRequest(String.Format("insert into
{0} values ({1})", tabs.SelectedTab.Text, values));
    if ((bool)ans[1])
    {
        MessageBox.Show((string)ans[0], "Unable to add row",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    else
    {
        ReloadTab(tabs.SelectedTab.Text);
    }
}
} // ready to use+

private void DeleteColumn(object sender, EventArgs e)
{
    if (delcolumnform.ShowDialog() == DialogResult.OK)
    {
        object[] ans = LoadRequest(String.Format("alter table
{0} drop column {1}", tabs.SelectedTab.Text, delcolumnform.name.Text));
        if ((bool)ans[1])
        {
            MessageBox.Show((string)ans[0], "Unable to delete
column", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        else
        {
            ReloadTab(tabs.SelectedTab.Text);
        }
    }
} // ready to use+

private void AddColumn(object sender, EventArgs e)
{
    if (columnform.ShowDialog() == DialogResult.OK)
    {
        object[] ans = LoadRequest(String.Format("alter table
{0} add {1} {2}", tabs.SelectedTab.Text, columnform.name.Text,
(string)columnform.type.SelectedItem));
        if ((bool)ans[1])
        {
            MessageBox.Show((string)ans[0], "Unable to add
column", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        else
        {
            ReloadTab(tabs.SelectedTab.Text);
        }
    }
}
}

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						85
Изм.	Лист	№ докум.	Подпись	Дата		

```

    } // ready to use+

    private void ReloadTab(string tab)
    {
        object[] ans = LoadRequest(String.Format("select * from
{0}", tab));
        if ((bool)ans[1])
        {
            MessageBox.Show((string)ans[0], "Reload Exception",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        else
        {
            table_list[tab].DataSource = GetTables(ans[0])[0];
        }
    } // ready to use+

    private void PresentTables(List<DataTable> tables)
    {
        addrow.Enabled = true;
        addtable.Enabled = true;
        addcolumn.Enabled = true;
        deletetable.Enabled = true;
        deletecolumn.Enabled = true;
        create.Enabled = true;
        export.Enabled = true;
        sql.Enabled = true;

        tabs.Controls.Clear();
        table_list.Clear();
        for (int d = 0; d < tables.Count; d++)
        {
            TabPage tp = new TabPage(tables[d].TableName);
            DataGridView tmp = new DataGridView() { Size = new
Size(746, 482), DataSource = tables[d], ScrollBars = ScrollBars.Both };
            table_list.Add(tables[d].TableName, tmp);
            tp.Controls.Add(tmp);
            tabs.Controls.Add(tp);
        }
    } // ready to use+

    public List<DataTable> GetTables(object x)
    {
        BinaryFormatter format = new BinaryFormatter();
        Stream tmp = new MemoryStream((byte[])x);
        tmp.Position = 0;
        return (List<DataTable>)format.Deserialize(tmp);
    }

    private void Connect(object sender, EventArgs e)
    {
        if(connectform.ShowDialog() == DialogResult.OK)

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						86
Изм.	Лист	№ докум.	Подпись	Дата		

```

        {
            object[] ans = LoadAllTables();
            if (!(bool)ans[1])
            {
                PresentTables(GetTables(ans[0]));
            }
            else
            {
                MessageBox.Show((string)ans[0], "General Exception",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    } // ready to use+

    private object[] LoadAllTables()
    {
        string connection =
        connectform.database.SelectedIndex.ToString() + ";";
        for (int d = 0; d < 5; d++)
        {
            connection += connectform.Controls[d].Text + ";";
        }
        return SendToServer(connection);
    } // ready to use+

    public object[] LoadRequest(string request)
    {
        string connection =
        connectform.database.SelectedIndex.ToString() + ";";
        for (int d = 0; d < 5; d++)
        {
            connection += connectform.Controls[d].Text + ";";
        }
        connection += request;
        return SendToServer(connection);
    } // ready to use+

    private object[] SendToServer(string message)
    {
        try
        {
            TcpClient client = new TcpClient();
            client.Connect(IPAddress.Parse(prefsform.server.Text),
            Convert.ToInt32(prefsform.port.Text));

            client.Client.Send(BitConverter.GetBytes(UnicodeEncoding.Unicode.GetByte
            Count(message)));

            client.Client.Send(UnicodeEncoding.Unicode.GetBytes(message));
            byte[] mess;
            byte[] size = new byte[4];
            client.Client.Receive(size);

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						87
Изм.	Лист	№ докум.	Подпись	Дата		

```

        if (BitConverter.ToInt32(size, 0) > 0)
        {
            mess = new byte[BitConverter.ToInt32(size, 0)];
            client.Client.Receive(mess);
            return new object[] { mess, false };
        }
        else
        {
            client.Client.Receive(size);
            mess = new byte[BitConverter.ToInt32(size, 0)];
            client.Client.Receive(mess);
            return new object[] {
UnicodeEncoding.Unicode.GetString(mess, 0, mess.Length), true };
        }
    }
    catch (Exception e)
    {
        return new object[] { e.Message , true };
    }

} // serializable

private void About(object sender, EventArgs e)
{
    MessageBox.Show("DataBase Editor Client\nSupports MySQL, MS
Access, MS SQL Server", "About", MessageBoxButtons.OK,
MessageBoxIcon.Information);
} // ready to use+

private void Exit(object sender, EventArgs e)
{
    Application.Exit();
} // ready to use+

private void MainFormLoad(object sender, EventArgs e)
{
    Location = new Point(Screen.PrimaryScreen.Bounds.Width / 2 -
Size.Width / 2, Screen.PrimaryScreen.Bounds.Height / 2 - Size.Height /
2);
} // ready to use+

private void InitializeComponent()
{
    this.SuspendLayout();
    //
    // MainForm
    //
    this.ClientSize = new System.Drawing.Size(284, 261);
    this.Name = "MainForm";
    this.Load += new System.EventHandler(this.MainForm_Load);
    this.ResumeLayout(false);
}

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						88
Изм.	Лист	№ докум.	Подпись	Дата		

```

    }

    private void MainForm_Load(object sender, EventArgs e)
    {

    }
}

```

Приложение В.2

Файл «core.cs»

```

using System.Windows.Forms;
using System;
namespace DB_Client
{
    class Core
    {
        [STAThreadAttribute]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.Run(new MainForm());
        }
    }
}

```

Приложение В.3

Файл «AddColumnForm.cs»

```

using System;
using System.IO;
using System.Windows.Forms;
using System.Drawing;

namespace DB_Client
{
    class AddColumnForm : Form
    {
        public TextBox name = new TextBox() { Size = new Size(128, 24),
        Location = new Point(96, 16) };
    }
}

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						89
Изм.	Лист	№ докум.	Подпись	Дата		

```

        private Label name_label = new Label() { Text = "Column
name: " , Location = new Point(16, 16), TextAlign =
ContentAlignment.MiddleLeft, Size = new Size(128, 20) };

        public ListBox type = new ListBox() { Size = new Size(128, 72),
Location = new Point(96, 40) };
        private Label type_label = new Label() { Text = "Type: ",
Location = new Point(16, 40), TextAlign = ContentAlignment.MiddleLeft,
Size = new Size(64, 20) };

        private Button add = new Button() { Text = "Add", Size = new
Size(128, 24), Location = new Point(240, 86) };

        public AddColumnForm()
        {
            FormBorderStyle = FormBorderStyle.FixedDialog;
            Size = new Size(400, 164);
            Text = "Add column in current table";

            Load += AddColumnForm_Load;
            add.Click += add_Click;

            type.Items.Add("int");
            type.Items.Add("text");
            type.Items.Add("float");
            type.Items.Add("decimal");
            type.Items.Add("binary");
            type.SelectedIndex = 0;
            type.SelectionMode = SelectionMode.One;

            Controls.Add(type);
            Controls.Add(add);
            Controls.Add(name);
            Controls.Add(name_label);
            Controls.Add(type_label);
        }

        void add_Click(object sender, EventArgs e)
        {
            DialogResult = DialogResult.OK;
        }

        void AddColumnForm_Load(object sender, EventArgs e)
        {
            Location = new Point(Screen.PrimaryScreen.Bounds.Width / 2 -
Size.Width / 2, Screen.PrimaryScreen.Bounds.Height / 2 - Size.Height /
2);
        }

        private void InitializeComponent()
        {
            this.SuspendLayout();

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						90
Изм.	Лист	№ докум.	Подпись	Дата		

```

        //
        // AddColumnForm
        //
        this.ClientSize = new System.Drawing.Size(284, 261);
        this.Name = "AddColumnForm";
        this.Load += new
System.EventHandler(this.AddColumnForm_Load_1);
        this.ResumeLayout(false);

    }

    private void AddColumnForm_Load_1(object sender, EventArgs e)
    {

    }

}
}

```

Приложение В.4

Файл «AddDataForm.cs»

```

using System;
using System.IO;
using System.Windows.Forms;
using System.Drawing;

namespace DB_Client
{
    class AddDataForm : Form
    {
        public TextBox name = new TextBox() { Size = new Size(128, 24),
Location = new Point(96, 17) };
        public Label name_label = new Label() { Text = "Data name:
", Location = new Point(16, 16), TextAlign =
ContentAlignment.MiddleLeft, Size = new Size(128, 20) };

        private Button add = new Button() { Text = "Add", Size = new
Size(128, 22), Location = new Point(240, 16) };

        public AddDataForm()
        {
            FormBorderStyle = FormBorderStyle.FixedDialog;
            Size = new Size(400, 94);
            Text = "Add database in current connection";

            Load += OnLoad;
            add.Click += AddData;

            Controls.Add(add);
            Controls.Add(name);
            Controls.Add(name_label);

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						91
Изм.	Лист	№ докум.	Подпись	Дата		

```

    }

    void AddData(object sender, EventArgs e)
    {
        DialogResult = DialogResult.OK;
    }

    void OnLoad(object sender, EventArgs e)
    {
        Location = new Point(Screen.PrimaryScreen.Bounds.Width / 2 -
Size.Width / 2, Screen.PrimaryScreen.Bounds.Height / 2 - Size.Height /
2);
    }

    private void InitializeComponent()
    {
        this.SuspendLayout();
        //
        // AddDataForm
        //
        this.ClientSize = new System.Drawing.Size(284, 261);
        this.Name = "AddDataForm";
        this.Load += new System.EventHandler(this.AddDataForm_Load);
        this.ResumeLayout(false);

    }

    private void AddDataForm_Load(object sender, EventArgs e)
    {
    }

}
}

```

Приложение В.5

Файл «AddRowForm.cs»

```

using System;
using System.Drawing;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Collections.Generic;
using System.Windows.Forms;
using System.Data;

namespace DB_Client
{
    class AddRowForm : Form
    {

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						92
Изм.	Лист	№ докум.	Подпись	Дата		

```

        private Button add = new Button() { Text = "Add", Size = new
Size(128, 22), Location = new Point(240, 16) };
        public AddRowForm()
        {
            FormBorderStyle = FormBorderStyle.FixedDialog;
            Text = "Add row";
            Size = new Size(448, 372);
            Load += AddRowForm_Load;
            VerticalScroll.Visible = true;
            VerticalScroll.Enabled = true;
            VerticalScroll.LargeChange = 1;
            add.Click += add_Click;
        }

        void add_Click(object sender, EventArgs e)
        {
            DialogResult = DialogResult.OK;
        }

        public void Update(DataGridView table)
        {
            Controls.Clear();
            int d = 0;
            for (d = 0; d < table.ColumnCount; d++)
            {
                Controls.Add(new TextBox() { Size = new Size(128, 24),
Location = new Point(128, 16 + 24 * d) });
            }
            for(d = 0; d < table.ColumnCount; d++)
            {
                Controls.Add(new Label() { Text =
table.Columns[d].HeaderText, Location = new Point(16, 16 + 24 * d),
TextAlign = ContentAlignment.MiddleLeft, Size = new Size(64, 20) });
            }
            add.Location = new Point(272, (d - 1) * 24 + 16);
            Controls.Add(add);
        }

        private void AddRowForm_Load(object sender, EventArgs e)
        {
            Location = new Point(Screen.PrimaryScreen.Bounds.Width / 2 -
Size.Width / 2, Screen.PrimaryScreen.Bounds.Height / 2 - Size.Height /
2);
        }

        private void InitializeComponent()
        {
            this.SuspendLayout();
            //
            // AddRowForm
            //
            this.ClientSize = new System.Drawing.Size(284, 261);

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						93
Изм.	Лист	№ докум.	Подпись	Дата		

```

        this.Name = "AddRowForm";
        this.Load += new
System.EventHandler(this.AddRowForm_Load_1);
        this.ResumeLayout(false);

    }

    private void AddRowForm_Load_1(object sender, EventArgs e)
    {

    }

}

```

Приложение В.6

Файл «AddTableForm.cs»

```

using System;
using System.IO;
using System.Windows.Forms;
using System.Drawing;

namespace DB_Client
{
    class AddTableForm : Form
    {
        public TextBox name = new TextBox() { Size = new Size(128, 24),
Location = new Point(96, 17) };
        private Label name_label = new Label() { Text = "Table name:
", Location = new Point(16, 16), TextAlign =
ContentAlignment.MiddleLeft, Size = new Size(128, 20) };
        private Button add = new Button() { Text = "Add", Size = new
Size(128, 22), Location = new Point(240, 16) };

        public AddTableForm()
        {
            FormBorderStyle = FormBorderStyle.FixedDialog;
            Size = new Size(400, 94);
            Text = "Add table into current database";

            Load += AddColumnForm_Load;
            add.Click += delete_Click;

            Controls.Add(add);
            Controls.Add(name);
            Controls.Add(name_label);

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						94
Изм.	Лист	№ докум.	Подпись	Дата		

```

    }

    void delete_Click(object sender, EventArgs e)
    {
        DialogResult = DialogResult.OK;
    }

    void AddColumnForm_Load(object sender, EventArgs e)
    {
        Location = new Point(Screen.PrimaryScreen.Bounds.Width / 2 -
Size.Width / 2, Screen.PrimaryScreen.Bounds.Height / 2 - Size.Height /
2);
    }

    private void InitializeComponent()
    {
        this.SuspendLayout();
        //
        // AddTableForm
        //
        this.ClientSize = new System.Drawing.Size(284, 261);
        this.Name = "AddTableForm";
        this.Load += new
System.EventHandler(this.AddTableForm_Load);
        this.ResumeLayout(false);

    }

    private void AddTableForm_Load(object sender, EventArgs e)
    {

    }
}
}

```

Приложение В.7

Файл «ConnectFom.cs»

```

using System;
using System.IO;
using System.Windows.Forms;
using System.Drawing;

namespace DB_Client
{
    class ConnectForm : Form
    {

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						95
Изм.	Лист	№ докум.	Подпись	Дата		

```

        public TextBox server = new TextBox() { Size = new Size(128,
24), Location = new Point(72, 16) };
        private Label server_label = new Label() { Text = "Server: "
, Location = new Point(16, 16), TextAlign = ContentAlignment.MiddleLeft,
Size = new Size(64, 20) };

        public TextBox file = new TextBox() { Size = new Size(128, 24),
Location = new Point(72, 40) };
        private Label file_label = new Label() { Text = "File: ",
Location = new Point(16, 40), TextAlign = ContentAlignment.MiddleLeft,
Size = new Size(64, 20) };

        public TextBox catalog = new TextBox() { Size = new Size(128,
24), Location = new Point(288, 16) };
        private Label catalog_label = new Label() { Text = "Init
catalog: ", Location = new Point(216, 16), TextAlign =
ContentAlignment.MiddleLeft, Size = new Size(64, 20) };

        public TextBox username = new TextBox() { Size = new Size(128,
24), Location = new Point(288, 40) };
        private Label username_label = new Label() { Text =
"Username: ", Location = new Point(216, 40), TextAlign =
ContentAlignment.MiddleLeft, Size = new Size(64, 20) };

        public TextBox password = new TextBox() { Size = new Size(128,
24), Location = new Point(288, 64) };
        private Label password_label = new Label() { Text =
"Password: ", Location = new Point(216, 64), TextAlign =
ContentAlignment.MiddleLeft, Size = new Size(64, 20) };

        public ListBox database = new ListBox() { Size = new Size(128,
72), Location = new Point(72, 64) };
        private Label database_label = new Label() { Text =
"Database: ", Location = new Point(16, 64), TextAlign =
ContentAlignment.MiddleLeft, Size = new Size(64, 20) };

        private CheckBox savepass = new CheckBox() { Size = new
Size(128, 24), Location = new Point(288, 86), Text = "Save password? "
};
        private Button connect = new Button() { Text = "Connect", Size =
new Size(128, 24), Location = new Point(288, 110) };

        private int lastindex = 0;
        private string confname = "connect.cfg";
        private string[] config = new string[15];

        public ConnectForm()
        {
            FormBorderStyle = FormBorderStyle.FixedDialog;
            Size = new Size(448, 186);
            Text = "Connect to...";

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						96
Изм.	Лист	№ докум.	Подпись	Дата		

```

Controls.Add(server);
Controls.Add(file);
Controls.Add(username);
Controls.Add(password);
Controls.Add(catalog);
Controls.Add(database);

Controls.Add(server_label);
Controls.Add(file_label);
Controls.Add(username_label);
Controls.Add(password_label);
Controls.Add(database_label);
Controls.Add(catalog_label);
Controls.Add(savepass);
Controls.Add(connect);

database.Items.Add("Microsoft SQL");
database.Items.Add("Microsoft Access");
database.Items.Add("MySQL");
database.SelectedIndex = 0;
database.SelectionMode = SelectionMode.One;
file.Enabled = false;

FormClosing += OnClosing;
Load += ConnectForm_Load;
connect.Click += Connect;
password.UseSystemPasswordChar = true;
database.SelectedIndexChanged += DataTypeChanged;
} // ready to use

private void ConnectForm_Load(object sender, EventArgs e)
{
    Location = new Point(Screen.PrimaryScreen.Bounds.Width / 2 -
Size.Width / 2, Screen.PrimaryScreen.Bounds.Height / 2 - Size.Height /
2);

    if (!File.Exists(confname))
    {
        File.WriteAllLines(confname, config);
    }
    else
    {
        config = File.ReadAllLines(confname);
        for (int j = 0; j < 5; j++)
        {
            if (j == 3)
            {
                Controls[j].Text = Encrypt(config[j +
database.SelectedIndex * 5], "ret");
            }
            else
            {

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						97
Изм.	Лист	№ докум.	Подпись	Дата		

```

        Controls[j].Text = config[j +
database.SelectedIndex * 5];
    }
}
connect.Select();
}

private void OnClosing(object sender, FormClosingEventArgs e)
{
    UnloadConfig();
} // ready to use

private void DataTypeChanged(object sender, EventArgs e)
{
    ListBox l = (ListBox)sender;
    switch (l.SelectedIndex)
    {
        case 0:
        {
            username.Enabled = true;
            password.Enabled = true;
            catalog.Enabled = true;
            server.Enabled = true;
            file.Enabled = false;
            savepass.Enabled = true;
            break;
        }
        case 1:
        {
            username.Enabled = false;
            password.Enabled = false;
            catalog.Enabled = false;
            server.Enabled = false;
            file.Enabled = true;
            savepass.Enabled = false;
            break;
        }
        case 2:
        {
            username.Enabled = true;
            password.Enabled = true;
            catalog.Enabled = true;
            server.Enabled = true;
            file.Enabled = false;
            savepass.Enabled = true;
            break;
        }
    }
    config = File.ReadAllLines(confname);
    for (int j = 0; j < 5; j++)
    {

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						98
Изм.	Лист	№ докум.	Подпись	Дата		

```

        if (j == 3)
        {
            if (savepass.Checked)
            {
                config[j + lastindex * 5] =
Encrypt(Controls[j].Text, "ret");
            }
            Controls[j].Text = Encrypt(config[j +
1.SelectedIndex * 5], "ret");
        }
        else
        {
            config[j + lastindex * 5] = Controls[j].Text;
            Controls[j].Text = config[j + 1.SelectedIndex * 5];
        }
    }
    File.WriteAllLines(confname, config);
    lastindex = 1.SelectedIndex;
} //ready to use

private string Encrypt(string value, string pass)
{
    string ret = "";
    char[] val = value.ToCharArray();
    char[] key = pass.ToCharArray();
    int d = 0;
    for (int z = 0; z < val.Length; z++ )
    {
        int a = (int)val[z] ^ (int)key[d];
        d += (d + 1 == key.Length) ? -d : 1;
        ret += (char)a;
    }
    return ret;
} //ready to use

private void UnloadConfig()
{
    for (int j = 0; j < 5; j++)
    {
        if (j == 3)
        {
            if (savepass.Checked)
            {
                config[j + lastindex * 5] =
Encrypt(Controls[j].Text, "ret");
            }
        }
        else
        {
            config[j + database.SelectedIndex * 5] =
Controls[j].Text;

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						99
Изм.	Лист	№ докум.	Подпись	Дата		

```

        }
    }
    File.WriteAllLines(confname, config);
} // ready to use

private void Connect(object sender, EventArgs e)
{
    UnloadConfig();
    this.DialogResult = DialogResult.OK;
}

private void InitializeComponent()
{
    this.SuspendLayout();
    //
    // ConnectForm
    //
    this.ClientSize = new System.Drawing.Size(284, 261);
    this.Name = "ConnectForm";
    this.Load += new
System.EventHandler(this.ConnectForm_Load_1);
    this.ResumeLayout(false);

}

private void ConnectForm_Load_1(object sender, EventArgs e)
{
}
}
}

```

Приложение В.8

Файл «DeleteColumnFom.cs»

```

using System;
using System.IO;
using System.Windows.Forms;
using System.Drawing;

namespace DB_Client
{
    class DeleteColumnForm : Form
    {
        public TextBox name = new TextBox() { Size = new Size(128, 24),
Location = new Point(96, 17) };
        private Label name_label = new Label() { Text = "Column
name: ", Location = new Point(16, 16), TextAlign =
ContentAlignment.MiddleLeft, Size = new Size(128, 20) };
        private Button delete = new Button() { Text = "Delete", Size =
new Size(128, 22), Location = new Point(240, 16) };

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						100
Изм.	Лист	№ докум.	Подпись	Дата		

```

public DeleteColumnForm()
{
    FormBorderStyle = FormBorderStyle.FixedDialog;
    Size = new Size(400, 94);
    Text = "Delete column from current table";

    Load += AddColumnForm_Load;
    delete.Click += delete_Click;

    Controls.Add(delete);
    Controls.Add(name);
    Controls.Add(name_label);
}

void delete_Click(object sender, EventArgs e)
{
    if(MessageBox.Show("Column removal will be permanent! Are
you sure?", "Warning!", MessageBoxButtons.OKCancel,
MessageBoxIcon.Warning) == DialogResult.OK)
    {
        DialogResult = DialogResult.OK;
    }
}

void AddColumnForm_Load(object sender, EventArgs e)
{
    Location = new Point(Screen.PrimaryScreen.Bounds.Width / 2 -
Size.Width / 2, Screen.PrimaryScreen.Bounds.Height / 2 - Size.Height /
2);
}

private void InitializeComponent()
{
    this.SuspendLayout();
    //
    // DeleteColumnForm
    //
    this.ClientSize = new System.Drawing.Size(284, 261);
    this.Name = "DeleteColumnForm";
    this.Load += new
System.EventHandler(this.DeleteColumnForm_Load);
    this.ResumeLayout(false);

}

private void DeleteColumnForm_Load(object sender, EventArgs e)
{
}

}
}

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						101
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение В.9

Файл «DeleteTableFom.cs»

```
using System;
using System.Drawing;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Collections.Generic;
using System.Windows.Forms;
using System.Data;

namespace DB_Client
{
    class PrefsForm : Form
    {
        public TextBox server = new TextBox() { Size = new Size(128,
24), Location = new Point(72, 16), Text = "127.0.0.1"};
        private Label server_label = new Label() { Text = "DBE IP:
", Location = new Point(16, 16), TextAlign =
ContentAlignment.MiddleLeft, Size = new Size(64, 20) };

        public TextBox port = new TextBox() { Size = new Size(128, 24),
Location = new Point(288, 16), Text = "61200"};
        private Label port_label = new Label() { Text = "DBE Port: ",
Location = new Point(216, 16), TextAlign = ContentAlignment.MiddleLeft,
Size = new Size(64, 20) };

        private Button apply = new Button() { Text = "Apply", Size = new
Size(128, 24), Location = new Point(288, 48) };

        public PrefsForm()
        {
            Controls.Add(server);
            Controls.Add(apply);
            Controls.Add(port);
            Controls.Add(server_label);
            Controls.Add(port_label);
            FormBorderStyle = FormBorderStyle.FixedDialog;
            Size = new Size(448, 128);
            apply.Click += Apply;
            Load += PrefForm_Load;
        }

        void Apply(object sender, EventArgs e)
        {
            DialogResult = DialogResult.OK;
        }

        private void PrefForm_Load(object sender, EventArgs e)
```

					ПГУ 2.090301.22.001 ПЗ	Лист
						102
Изм.	Лист	№ докум.	Подпись	Дата		

```

        {
            Location = new Point(Screen.PrimaryScreen.Bounds.Width / 2 -
Size.Width / 2, Screen.PrimaryScreen.Bounds.Height / 2 - Size.Height /
2);
        }

private void InitializeComponent()
{
    this.SuspendLayout();
    //
    // PrefsForm
    //
    this.ClientSize = new System.Drawing.Size(284, 261);
    this.Name = "PrefsForm";
    this.Load += new System.EventHandler(this.PrefsForm_Load);
    this.ResumeLayout(false);

}

private void PrefsForm_Load(object sender, EventArgs e)
{
}

}
}

```

Приложение В.10

Файл «SQLRequstForm.cs»

```

using System;
using System.IO;
using System.Windows.Forms;
using System.Drawing;

namespace DB_Client
{
    class SqlRequestForm : Form
    {
        public RichTextBox script = new RichTextBox() { Size = new
Size(352, 128), Location = new Point(16, 16) };
        private Button exe = new Button() { Text = "Execute", Size = new
Size(128, 24), Location = new Point(240, 304) };
        public DataGridView data = new DataGridView() { Size = new
Size(352, 128), Location = new Point(16, 160) };

        public MainForm context;

        public SqlRequestForm()
        {
            FormBorderStyle = FormBorderStyle.FixedDialog;
            Size = new Size(400, 382);

```

					ПГУ 2.090301.22.001 ПЗ	Лист
						103
Изм.	Лист	№ докум.	Подпись	Дата		

```

        Text = "Execute SQL Request";

        Load += AddColumnForm_Load;
        exe.Click += add_Click;
        Controls.Add(exe);
        Controls.Add(data);
        Controls.Add(script);
    }

    void add_Click(object sender, EventArgs e)
    {
        object[] ans = context.LoadRequest(script.Text);
        if ((bool)ans[1])
        {
            MessageBox.Show((string)ans[0], "Unable to execute
request", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        else
        {
            data.DataSource = context.GetTables(ans[0])[0];
        }
    }

    void AddColumnForm_Load(object sender, EventArgs e)
    {
        Location = new Point(Screen.PrimaryScreen.Bounds.Width / 2 -
Size.Width / 2, Screen.PrimaryScreen.Bounds.Height / 2 - Size.Height /
2);
    }

    private void InitializeComponent()
    {
        this.SuspendLayout();
        //
        // SqlRequestForm
        //
        this.ClientSize = new System.Drawing.Size(284, 261);
        this.Name = "SqlRequestForm";
        this.Load += new
System.EventHandler(this.SqlRequestForm_Load);
        this.ResumeLayout(false);
    }

    private void SqlRequestForm_Load(object sender, EventArgs e)
    {
    }
}
}

```

Приложение С

UML-диаграммы приложения

Приложение С.1 – UML-диаграмма классов клиента

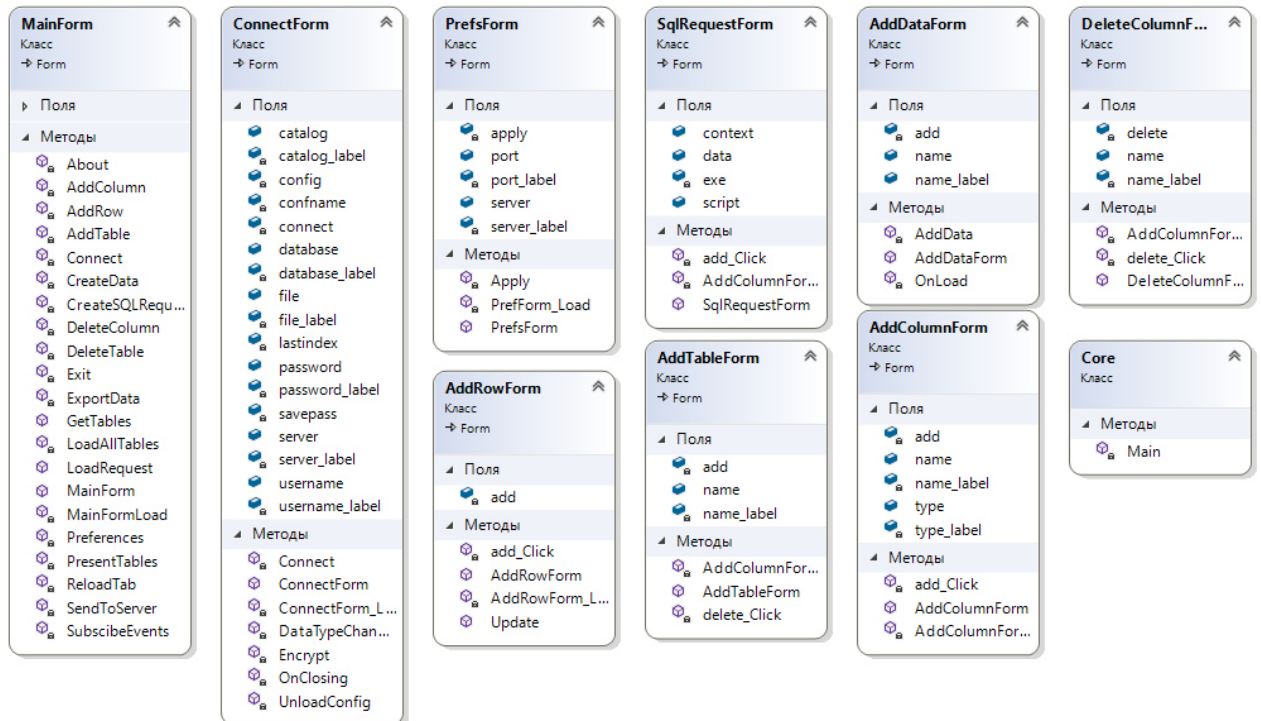


Рисунок 52 - UML-диаграмма классов клиента

Приложение С.2 – UML-диаграмма классов клиента

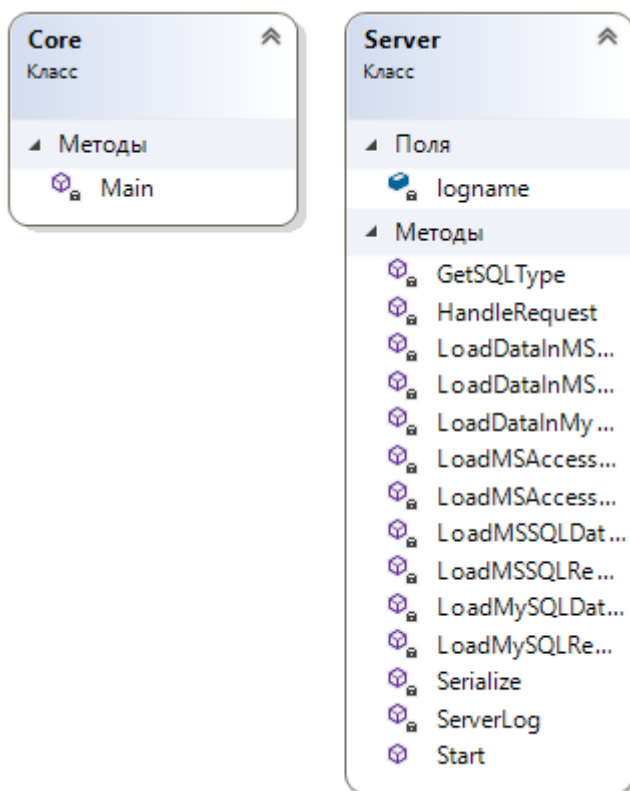


Рисунок 53 - UML-диаграмма классов сервера

Приложение D

Презентация

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

КЛИЕНТ СЕРВЕРНОЕ ПРИЛОЖЕНИЕ ПЕРЕНОСА ДАННЫХ МЕЖДУ БАЗАМИ ДАННЫХ

Выполнил: студент группы 16BB2 Кузнецов А.С.

ЦЕЛЬ РАБОТЫ

Разработать клиент-серверное приложение, имеющее доступ к различным базам данных, возможность их редактировать и экспортировать в другой тип базы данных.



АКТУАЛЬНОСТЬ ПРОЕКТА

Актуальность состоит в создании приложения, способного открывать базы данных нескольких форматов, взаимодействовать с ними и конвертировать из одного типа в другой

Кузнецов А.С.

					ПГУ 2.090301.22.001 ПЗ	Лист
						107
Изм.	Лист	№ докум.	Подпись	Дата		

НАИБОЛЕЕ РАСПРОСТРАНЕННЫЕ СУБД

В % ОТ ВСЕХ ОПРОШЕННЫХ КОМПАНИЙ

Основные используемые СУБД (% от всех опрошенных компаний)

№	СУБД	Год проведения опроса								
		2010	2011	2012	2013	2014	2015	2016	2017	2018
1	MS SQL	63%	74%	70%	66%	70%	67%	59%	61%	67%
2	MySQL	47%	40%	59%	56%	56%	54%	42%	53%	61%
3	PostgreSQL	17%	15%	26%	30%	28%	28%	33%	36%	51%
4	Oracle	49%	55%	51%	47%	45%	39%	36%	37%	40%
5	SQLite	9%	5%	12%	10%	19%	12%	10%	19%	25%
6	MS Access	19%	9%	19%	17%	18%	19%	17%	18%	16%
7	Firebird	11%	9%	10%	13%	16%	15%	11%	11%	14%
8-9	IBM DB2	13%	14%	9%	10%	12%	12%	8%	8%	7%
8-9	InterBase	9%	7%	7%	10%	6%	6%	3%	3%	7%
10-12	MSDE	7%	5%	5%	5%	7%	2%	2%	4%	6%

Кузнецов А.С.

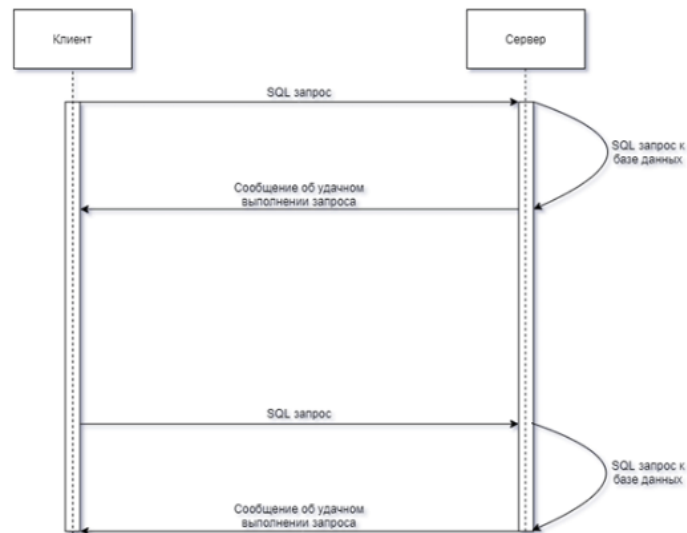


ВЗАИМОДЕЙСТВИЕ ЧАСТЕЙ ПРОГРАММЫ



Кузнецов А.С.

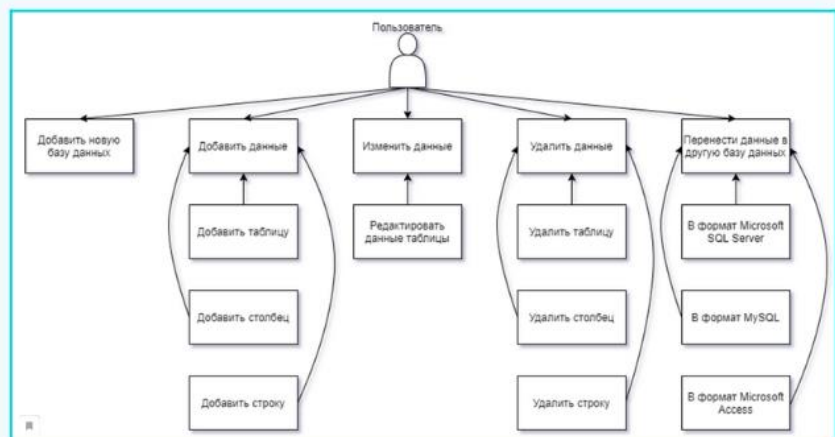
Последовательность действий программы



Кузнецов А.С.

ВЗАИМОДЕЙСТВИЕ ПОЛЬЗОВАТЕЛЯ С ПРОГРАММОЙ:

1. Открыть БД в приложении
2. Редактировать БД
3. Конвертировать БД в другой тип



Кузнецов А.С.

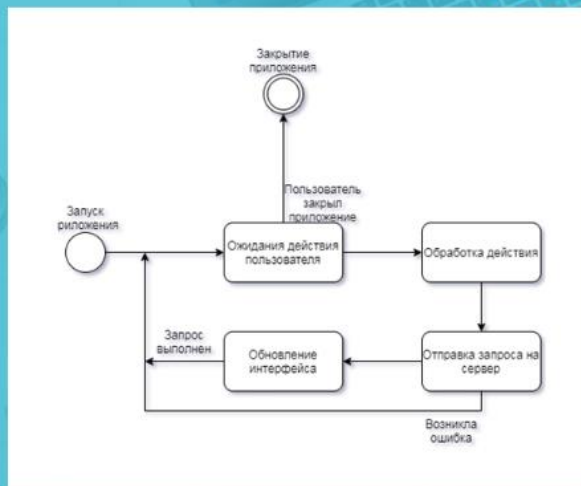
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.22.001 ПЗ

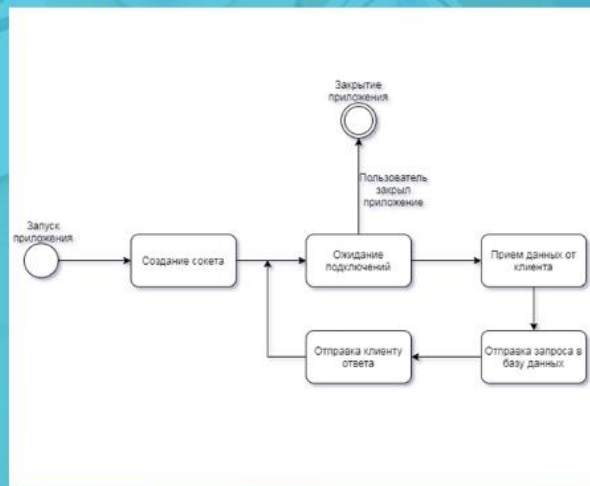
Лист

109

ПРИНИМАЕМЫЕ ПРОГРАММОЙ СОСТОЯНИЯ



Клиентское приложение



Серверное приложение

Кузнецов А.С.

Интерфейс программы

ОКНО ПОДКЛЮЧЕНИЯ

Connect to...

Server: 127.0.0.1:3306

File:

Database: Microsoft SQL, Microsoft Access, MySQL

Init catalog: world

Username: root

Password:

☐ Save password?

Connect

ОКНО ДОБАВЛЕНИЯ СТОЛБЦА

Add column in current table

Column name:

Type: int, text, float, decimal, binary

Add

ГЛАВНОЕ ОКНО ПРОГРАММЫ

ID	Name	CountryCode	District	Population
1	Kabul	AFG	Kabul	1780000
2	Gandahar	AFG	Gandahar	231500
3	Herat	AFG	Herat	186000
4	Mosere-Sherif	AFG	Balkh	127800
5	Amsterdam	NLD	Noord-Holland	731200
6	Rotterdam	NLD	Zuid-Holland	593321
7	Haag	NLD	Zuid-Holland	440900
8	Utrecht	NLD	Utrecht	234323
9	Eindhoven	NLD	Noord-Brabant	201843
10	Tilburg	NLD	Noord-Brabant	193238
11	Groningen	NLD	Groningen	172701
12	Breda	NLD	Noord-Brabant	160398
13	Apeldoorn	NLD	Gelderland	153491
14	Nijmegen	NLD	Gelderland	152463
15	Enschede	NLD	Overijssel	149544
16	Haarlem	NLD	Noord-Holland	148772
17	Almere	NLD	Flevoland	142465
18	Amsterdam	NLD	Gelderland	138020
19	Zaandam	NLD	Noord-Holland	135621
20	A's-Hertogenbosch	NLD	Noord-Brabant	129170
21	Amersfoort	NLD	Utrecht	126270

Кузнецов А.С.

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.22.001 ПЗ

Лист

110