ПРИДНЕСТРОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМ. Т.Г. ШЕВЧЕНКО
Инженерно-технический институт
Кафедра информационных технологий и автоматизированного управления производственными процессами

КУРСОВАЯ РАБОТА
ПО УПРАВЛЕНИЮ ДАННЫМИ

Методические указания

Тирасполь, 2015

УДК 004.655.3
ББК 32.973.2

Составители:
В.Г. Труханова, преподаватель
В.С. Попукайло, преподаватель

Рецензенты:
Ю.А. Стояренко, канд. тех. наук, доцент, зав. каф. информационных технологий и автоматизированного управления производственными процессами
М.В.Нижегородова, канд. пед. наук, доцент каф. программного обеспечения вычислительной техники и автоматизированных систем

Курсовая работа по управлению данными: Методические указания /Сост:В.Г. Труханова, В.С. Попукайло. – Тирасполь, 2015.–51 с., 3п.л.

Методические указания содержат требования к содержанию и оформлению пояснительной записки, методические указания к выполнению и требования к технической документации в виде приложений, а также требования к процедуре защиты курсовой работы.

Предназначены для студентов 2 курса направления 09.03.02 «Информационные системы и технологии», выполняющих курсовые работы по дисциплине «Управление данными».

УДК 004.655.3
ББК 32.973.2

Рекомендовано Научно-методическим Советом ПГУ им. Т.Г. Шевченко

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Выполнение курсовой работы является составной частью изучения дисциплины «Управление данными» для студентов направления 09.03.02 «Информационные системы и технологии» в соответствии с учебным планом. Предметом курсовой работы является освоение методов проектирования (концептуального, логического и физического) реляционных баз данных и работы с базами данных с помощью *SQL* (*StructuredQueryLanguage*), изучаемых в теоретическом курсе лекций и лабораторных занятиях.

Результатом курсовой работы является спроектированная и реализованная база данных, так же клиентская часть на языке *VisualC#.*

Курсовой проект является аттестационной работой по курсу «Управление данными». При выполнении проекта студент должен показать качество освоения материала курса и практические навыки:

– по применению методик проектирования моделей баз данных;

– по разработке баз данных на основе СУБД клиент-серверной архитектуры;

– по использованию средств серверной обработки;

– по разработке клиентской части с использованием современных технологий разработки интерфейса и доступа к данным.

Выполнение курсовой работы способствует лучшему усвоению студентами теоретических вопросов и приобретению практических навыков работы с базами данных.

1. ОРГАНИЗАЦИЯ ПРОВЕДЕНИЯ КУРСОВОЙ РАБОТЫ

1.1 Цель курсовой работы

Целью курсовой работы является освоение методов проектирования баз данных и работы с базами данных с помощью *SQL (StructuredQueryLanguage)*.

1.2 Задание на курсовую работу

Задание содержит название конкретной предметной области, для которой необходимо:

- сформулировать цель проектирования базы данных;
- определить круг запросов и задач, которые предполагается решать с использованием созданной базы данных;
- построить концептуальную, логическую модель;
- сформулировать требования к базе данных и реализовать ее;
- построить схему базы данных;
- реализовать запросы на добавление, редактирование, удаление и обновление;
- реализовать процедуры и их программный вызов;
- реализовать клиентскую часть с возможностью добавления, удаление, редактирования, обновления, фильтрации данных;
- реализация возможности вывода результатов в виде отчетов.

1.3 Этапы выполнения курсовой работы

В ходе выполнения курсовой работы рекомендуется придерживаться календарного плана, приведённого в таблице 1.

Таблица 1 – Календарный план

Содержание этапа	Продолжительность этапа
1. Выбор темы и утверждение технического задания.	2 недели
2. Разработка концептуальной и логической модели.	3 недели
3. Создание базы данных.	3 недели
4. Разработка приложения.	6 недель
5. Оформление пояснительной записки.	1 неделя
6. Сдача курсовой работы на проверку..	1 неделя
7. Защита курсовой работы.	1 неделя

По итогам всех этапов проводится защита курсовой работы с применением мультимедиа средств.

Результатом выполнения курсовой работы должны быть законченное программное приложение и отчет, оформленный по всем предъявленным к нему требованиям.

Разрабатываемое программное приложение должно:

1)заносить информацию в созданную базу данных;

2)выполнять необходимые действия по редактированию, добавлению, фильтрации и удалению информации в базе данных;

3)содержать достаточное количество данных, позволяющих показать результаты выполнения запросов;

4)выполнять процедуры из текста технического задания;

5)Выводить результаты в виде отчетов.

1.4 Содержание отчета по курсовой работе

Представляемый отчет должен содержать:

1. Техническое задание;
2. Содержание;
3. Введение;
4. Описание предметной области;
5. Концептуальную модель (ER-модели), в том числе:

– необходимый набор сущностей, отражающих предметную область и информационные потребности пользователей,

необходимый набор атрибутов каждой сущности, идентифицирующие атрибуты;
- определение множественности и условности связей;
- классификацию связей (1:1, 1:M, M:N);
- ER-диаграмму модели базы данных;
- описание модели базы данных на языке инфологического проектирования.

6. Логическую модель предметной области, в том числе:
   - описание состава отношений базы данных и набора атрибутов каждого отношения;
   - первичные и внешние ключи отношений;
   - шаги по нормализации полученных отношений с приведением модели базы данных к третьей нормальной форме;
   - необходимые ограничения целостности;
7. Этапы создание базы данных и таблиц;
8. Реляционную модель базы данных предметной области;
9. Реализация запросови процедур;
10. Создание форм редактирования, обновления и фильтрации данных, а так же добавления, удаления и обновления данных;
11. Реализация отчетов;
12. Заключение;
13. Руководство пользователя;
14. Листинг программы.

2.МЕТОДИЧЕСКИЕ УКАЗАНИЯ

2.1 Описание предметной области

Этап описания предметной области является одним из первых и наиболее важным из этапов. Модель предметной области - это наши знания о предметной области. Знания могут быть как в виде неформальных знаний в мозгу эксперта, так и выражены формально при помощи каких-либо средств. В качестве таких средств могут выступать текстовые описания предметной области, наборы должностных инструкций, правила ведения дел в компании и т.п. Рассмотрим предметную область «Банк».

Описание предметной области БАНК. Банк имеет филиалы. Менеджеры управляют филиалами. Клиенты имеют счета в филиалах. Филиалы обрабатывают счета.

1. Об объектах (сущностях) данной предметной области известно следующее:

- ФИЛИАЛАХ: номер филиала, адрес филиала;
- МЕНЕДЖЕРАХ: номер менеджера, стаж работы по специальности;
- СЧЕТАХ: номер счета, тип счета, дата открытия счета, капитализация (Да/Нет), остаток на счете;
- КЛИЕНТАХ: номер клиента, ФИО клиента, адрес клиента, подпись клиента.

2. О связях между сущностями известно, что:

- менеджер управляет *одним* филиалом. Филиалом управляет *один* менеджер;
- филиал обрабатывает *несколько* счетов. Счет обрабатывается в *одном* филиале;
- клиент может иметь *несколько* счетов. Счет принадлежит *одному* клиенту.

3. Дополнительная информация о связях:

- *не каждыйменеджер* банка управляет филиалом. *Каждый-филиал* управляется менеджером;
- *каждый филиал* обрабатывает счета. *Каждый счет* обрабатывается в филиале;
- *каждый клиент* имеет счет. *Каждый счет* принадлежит клиенту.

2.2 Построение концептуальной модели (*ER*-модель)

Этап инфологического проектирования базы данных связан с анализом и описанием разнообразных информационных требований пользователей.

Результатом этапа инфологического проектирования является представление информационных требований в виде целостной концептуальной инфологической модели предметной области.

Инфологическую модель представить в виде диаграммы, используя средства моделирования "сущность-связь". Рассмотрим пример создания концептуальной модели предметной области «Банк».

Из описания предметной области БАНК следует, что в ней можно выделить четыре сущности – *менеджер*, *филиал*, *счет*, *клиент*.

Между ними в действительности существуют три связи:

1) менеджер – *управляет* – филиал;

2) филиал – *обрабатывает* – счет;

3) клиент – *имеет* – счет.

На основании пункта 2 описания определяются типы этих связей: связь 1 – типа *1:1*, связь 2 и 3 – типа *1:M*.

Исходя из пункта 3 описания, устанавливается КП сущностей: сущности менеджер – *необязательный*, сущностей филиал, счет, клиент – *обязательный*.

Установленной информации о сущностях, их связях, типах связей, КП сущностей и знания их графического представления на ER-модели достаточно, чтобы изобразить ER-модель предметной области БАНК. Она выглядит, как на рисунке 1.

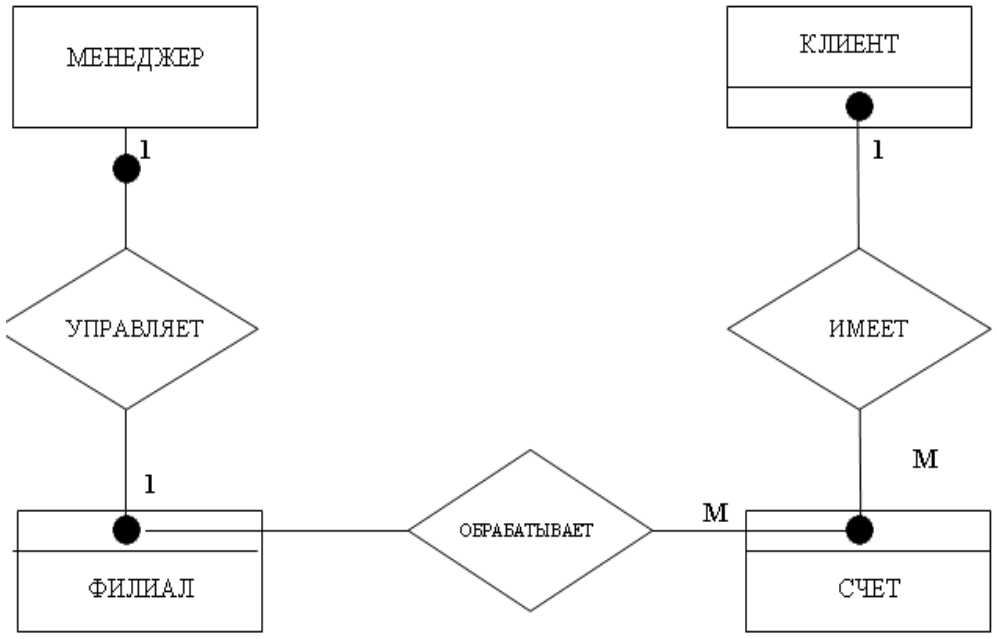


Рисунок 1 -ER-модель предметной области БАНК

Исходя из пункта 1 описания предметной области БАНК, для каждой сущности определяется набор атрибутов. Наборы атрибутов и их имена представлены в таблице.

Таблица 2 – Наборы атрибутов и их имена

МЕНЕДЖЕР		ФИЛИАЛ
Номер менеджера (НМ)		**Номер филиала (НФ)**
Стаж работы по специальности (СТАЖ)		Адрес филиала (АДР_Ф)
СЧЕТ		КЛИЕНТ
Номер счета (НС)		**Номер клиента (НК)**
Тип счета (ТИП)		Ф.И.О. клиента (ФИО_К)
Дата открытия счета (ДАТА)		Адрес клиента (АДР_К)
Капитализация (Да/Нет) (КАП)		Подпись клиента (ПОДП)
Остаток на счете, руб. (ОСТ)		

Руководствуясь определением ключа сущности, для сущностей устанавливают ключи. В таблице они показаны жирным шрифтом.

ER-модель (см. рисунок 1), дополненная наборами атрибутов сущностей (см. таблица 1), – это *концептуальная модель* предметной области БАНК.

2.3 Разработка логической модели предметной области

Логическая модель данных является начальным прототипом будущей базы данных. Логическая модель строится в терминах информационных единиц, но без привязки к конкретной системе управления базами данных. Более того, логическая модель данных необязательно должна быть выражена средствами именно реляционной модели данных.

Одну и ту же *ER*-модель можно преобразовать как в реляционную модель данных, так и в модель данных для иерархических и сетевых СУБД, или в постреляционную модель данных. Однако, т.к. мы рассматриваем именно реляционные СУБД, то можно считать, что логическая модель данных для нас формулируется в терминах реляционной модели данных. Рассмотрим пример.

Разработать логическую модель предметной области БАНК**,** предположив, что ее БД будет создаваться по реляционной модели данных.

Созданную ER-модель предметной области БАНК необходимо преобразовать в реляционную модель, то есть от ER-диаграмм перейти к таблицам. Существует шесть правил генерации таблиц из ER-диаграмм.

ER-диаграмма для связи 1 (см. рисунок 1) преобразуется по следующему правилу:

Если связь типа 1:1 и класс принадлежности (КП) одной сущности является обязательным, а другой – необязательным, то необходимо построить таблицу для каждой сущности. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Первичный ключ сущности, для которой КП является необязательным, добавляется как атрибут в таблицу для сущности с обязательным КП.

Примечание. Первичный ключ сущности – ключ сущности, значения которого уникальные и непустые.

Тогда для связи 1 будем иметь следующие две таблицы, которые можно связать по ключевому атрибуту «Номер менеджера»:

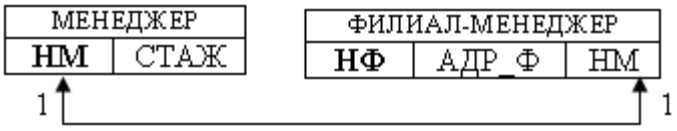


Рисунок 2 – Связь 1

ER-диаграмма для связи 2, 3 преобразуется по следующему правилу:

Если связь типа 1:М и КП сущности на стороне М является обязательным, то необходимо построить таблицу для каждой сущности.

Первичный ключ сущности должен быть первичным ключом соответствующей таблицы.

Первичный ключ сущности на стороне 1 добавляется как атрибут в таблицу для сущности на стороне М.

Тогда для связи 2 будем иметь следующие две таблицы, которые можно связать по ключевому атрибуту «Номер филиала»:

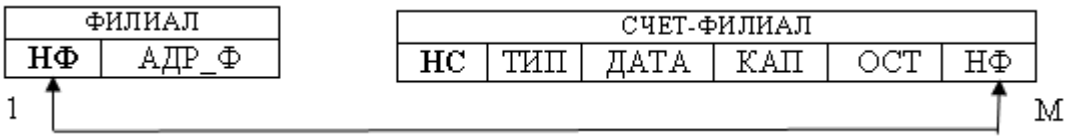


Рисунок 3 – Связь 2

Для связи 3 будем иметь следующие две таблицы, которые можно связать по ключевому атрибуту «Номер клиента»:

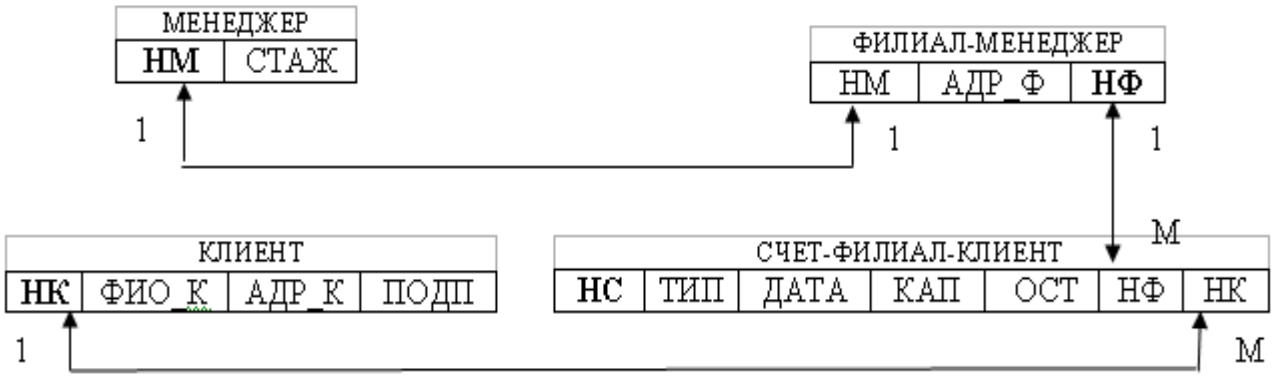


Рисунок 4 – Связь 3

Проанализируем полученные таблицы на предмет дублирования атрибутов. Таблица ФИЛИАЛ входит составной частью в таблицу ФИЛИАЛ-МЕНЕДЖЕР и поэтому ее можно исключить. В таблице СЧЕТ-ФИЛИАЛ и СЧЕТ-КЛИЕНТ дублируются атрибуты сущности СЧЕТ. Во избежание этого атрибут НК из таблицы

СЧЕТ-КЛИЕНТ добавим в таблицу СЧЕТ-ФИЛИАЛ, а таблицу СЧЕТ-КЛИЕНТ исключим.

Оставшиеся таблицы свяжем попарно по ключевым атрибутам. Тогда полученную реляционную модель БД (логическую модель) предметной области БАНК можно представить, как на рисунок 5.

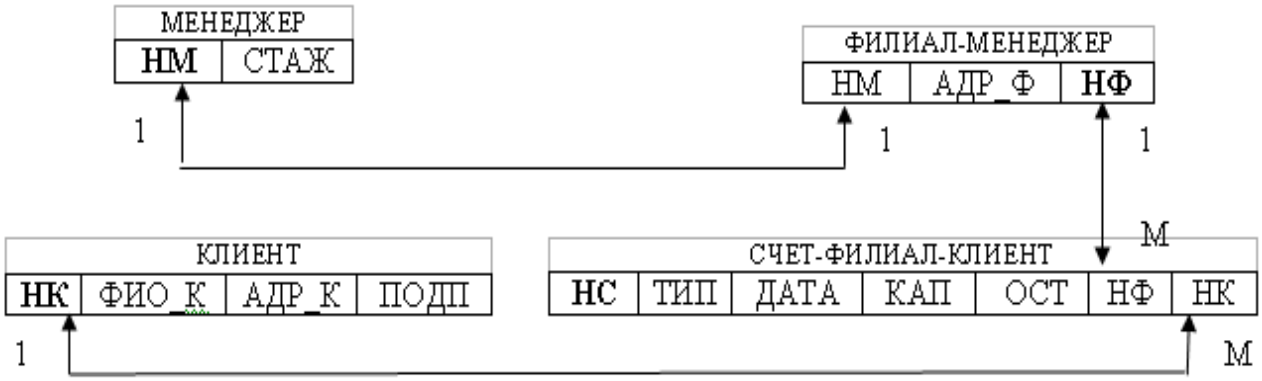


Рисунок 5 –Логическая модель БД предметной области БАНК

Отметим, что все четыре таблицы удовлетворяют требованиям первой, второй и третьей нормальной формы.

Зачастую логическую модель БД дополняют атрибутами полей, типом данных, описанием.

Таблица 4 –Структура отношения Кассир (*Kassiri*)

Имя атрибута	**Имя поля**	**Тип данных**	**Описание**
Номер клиента	НК	Числовой	Содержит номер клиента. Данный атрибут является идентифицирующим и уникальным для каждой из записей.
Ф.И.О. клиента	ФИО_К	Текстовый	Атрибут содержит фамилию клиента. Атрибут не уникален.
Адрес клиента	АДР_К	Текстовый	Атрибут содержит адрес. Атрибут не уникален.
Подпись клиента	ПОДП	Текстовый	Атрибут содержит подпись клиента. Данный атрибут является идентифицирующим и уникальным для каждой из записей.

2.4 Создание базы данных и таблиц

Создание базы данных в среде *Microsoft SQL Server Management Studio*. В разделе «Базы данных» правой кнопкой выбираем «Создать базу данных…». Назовем базу данных по индексу группы – *mbs21*. Владельцем базы данных назначим пользователя, под именем которого был произведен вход – *studentMBS21*. В разделе «Параметры» выбираем тип сортировки *Cyrillic_General_BIN* (для примера), нажимаем ОК.

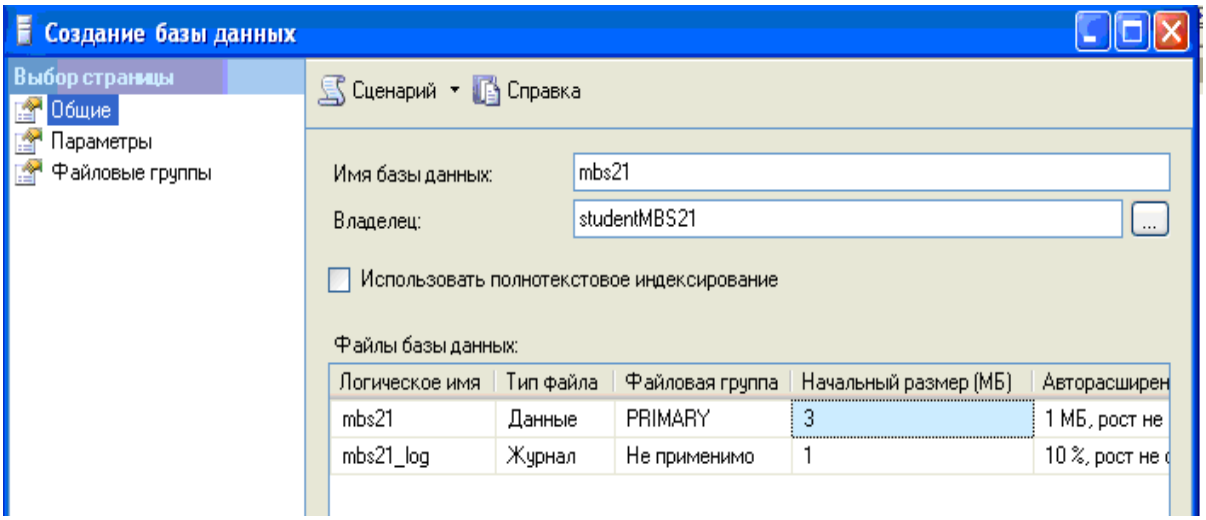


Рисунок 6 - Создание базы данных

В разделе «Базы данных» Обозревателя объектов появилась вновь созданная*mbs21*:

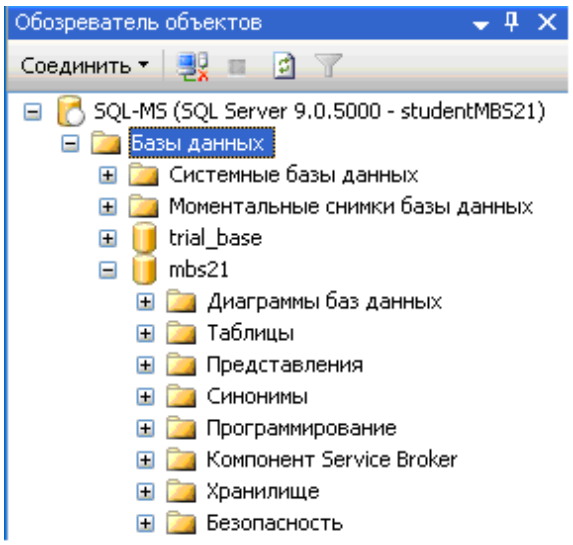


Рисунок 7 - Обозреватель объектов

Создание таблиц базы данных в среде *Microsoft SQL Server Management Studio*. Начнем с создания таблицы *Speciality*. Структура таблицы приведена ниже:

Таблица 5 – структура таблицы БД

Имя поля (столбца)	**Содержание**	**Тип данных**	**Возможность содержать NULL**
Num	Первичный ключ	int	нет
Name	Название специальности	varchar(60)	нет

В реляционных базах данных первичный ключ используется как уникальный идентификатор записи. Это поле является обязательным, оно используется для связи таблиц по внешним ключам (примеры такого связывания будут рассмотрены далее). Первичный ключ должен иметь целочисленный тип (в данном случае - *int*).

Во втором поле будет храниться название специальности - некоторая строка, поэтому мы выбираем для этого поля тип *varchar(60)*. Число в скобках означает максимальное число символов в строке.

Детальную информацию об этих типах можно посмотреть в справке.

Простейшим образом можно создавать таблицы средствами *MS SQL Server Management Studio* (правая кнопка мыши на заголовке «Таблицы» > Создать таблицу.). Получаем следующее:

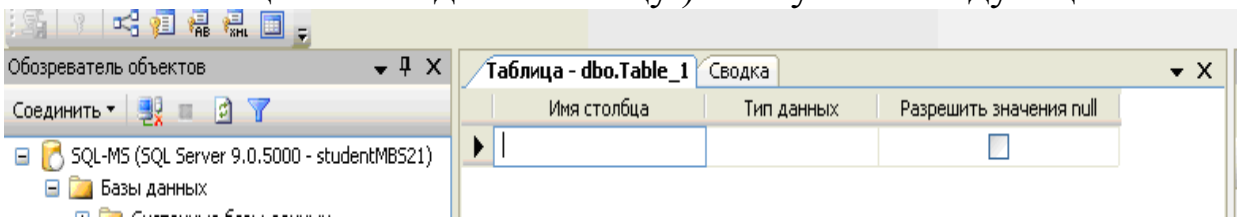


Рисунок 8 - Создание таблицы

Вводим имя первого столбца *Num* (первичный ключ – в том столбце хранится номер записи), выбираем из выпадающего списка тип данных *int*. Первичный ключ не может быть пустым, поэтому и оставляем неотмеченным поле «Разрешить значения *null*».

Затем аналогичным образом вводим имя второго столбца, задаем тип, запрещаем полю иметь значение *null*. Таблица принимает следующий вид:

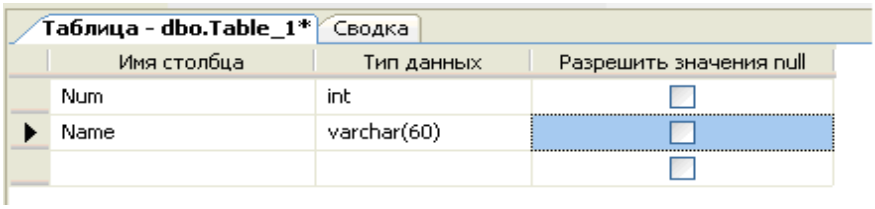


Рисунок 9 – Запрет значения *null*

Теперь необходимо указать, что поле *Num* будет являться первичным ключом. Правой кнопкой мыши щелкаем по этому полю и выбираем «Задать первичный ключ»:

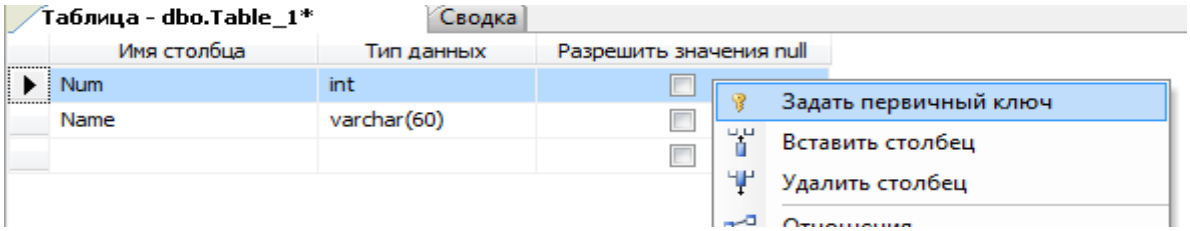


Рисунок 10 - Задание первичного ключа

Сохраняем таблицу под именем *Speciality*(после этого таблица должна появиться в обозревателе объектов). Теперь можно перейти к заполнению этой таблицы (для этого нужно в обозревателе объектов выбрать эту таблицу и в контекстном меню нажать «Открыть таблицу»):

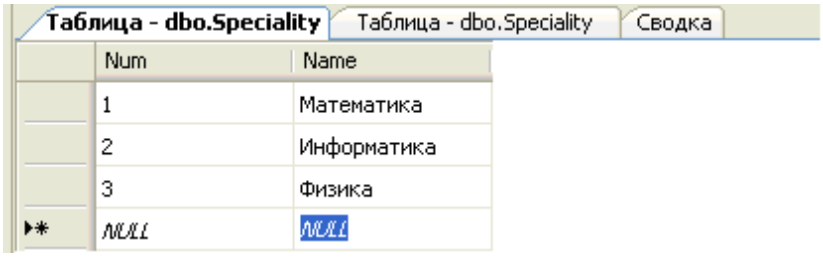


Рисунок 11 – Заполнение таблицы

При заполнении вы обнаружите, что каждый раз приходится вводить не только полезную информацию (название специальности), но и номер записи. Чтобы вводить номер записи автоматически, нужно задать спецификацию идентифицирующего столбца. Для этого необходимо в свойствах столбца указать, что данный столбец является идентифицирующим (рисунок 12):

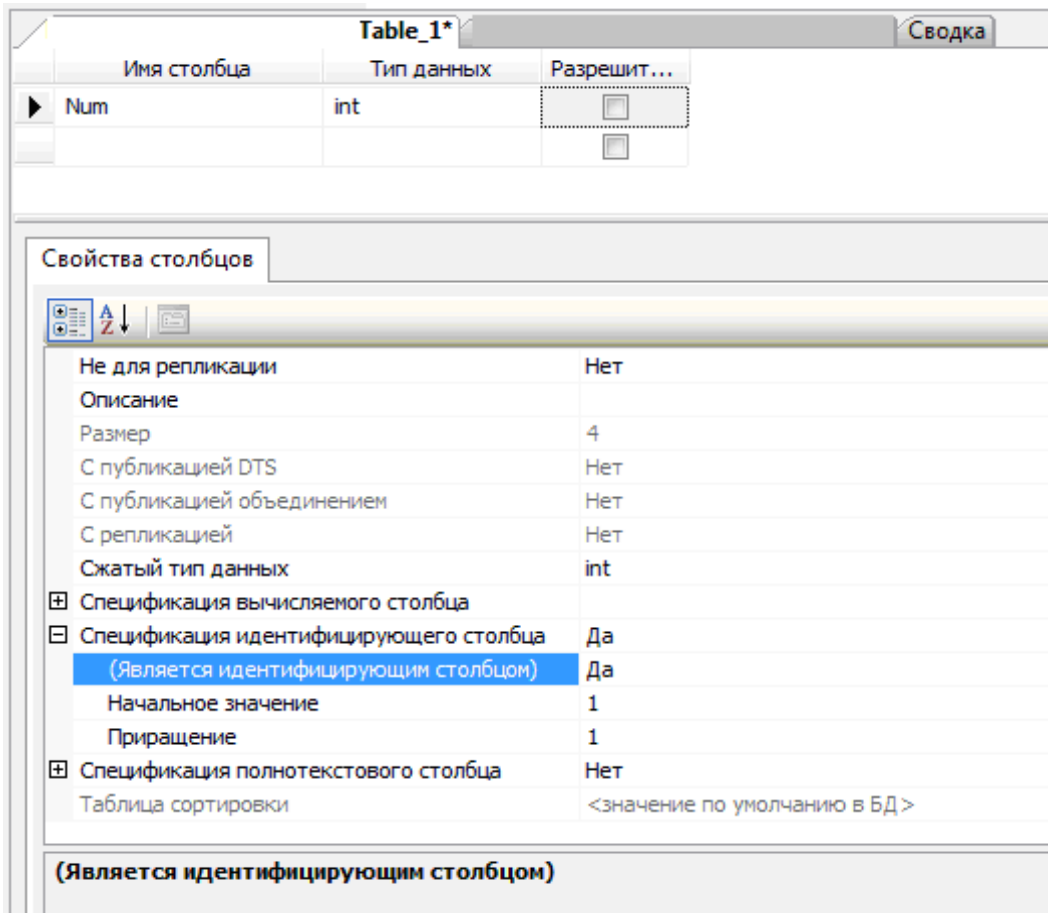


Рисунок 12 - Определение свойств идентифицирующего столбца

Создание таблиц базы данных с помощью *SQL*-запроса. Создание таблиц в графическом режиме, безусловно, удобно, однако не универсально. При использовании других средств разработки баз данных (например, *IBM DB2*) придется привыкать к новым приемам работы. Использование конструкций языка *SQL* позволяет работать с базами данных, исходя из единого подхода, в любой среде управления базами данных.

Выберите на панели инструментов «Создать запрос»:

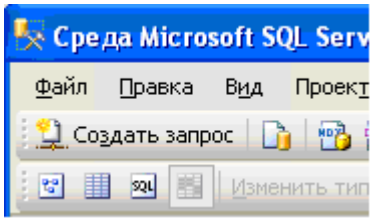


Рисунок 13 – Создание запроса

Создадим новую базу данных запросом.

Напишем

CREATE DATABASE mbs21_query

и нажмем F5. В обозревателе объектов должна появиться новая база (если сразу не появилась, то надо выделить мышью раздел «Базы данных» и в контекстном меню выбрать «Обновить»).

Теперь создадим таблицу *Speciality*. Упрощенный синтаксис создания таблиц следующий:

CREATE TABLE <имя таблицы>

(

<имя столбца 1><тип данных> *[NOT NULL] [DEFAULT*<значение по умолчанию>],

<имя столбца 2><тип данных> *[NOT NULL] [DEFAULT*<значение по умолчанию>],

...)

Введем новый запрос:

/* создание таблицы Специальность*/

USE mbs21_query -- определяем базу данных, в которую входит таблица

CREATE TABLE Speciality(

*Num INT IDENTITY(1,1) PRIMARY KEY NOT NUL*L, -- первичный ключ

NameSpec VARCHAR(60) -- название специальности

)

В обозревателе объектов видим, что таблица действительно создана. Файл с *SQL*-запросом сохраняем в своей папке (в конце работы необходимо показать запросы, которые были выполнены, преподавателю). Слово *IDENTITY(1,1)* добавлено, чтобы поле первичного ключа *Num* автоматически нумеровалось начиная с единицы (фактически, эта конструкция определяет спецификацию идентифицирующего столбца).

2.5 Схема базы данных предметной области

Для начала следует задать внешние ключи для таблиц. Выбираем в «Обозревателе объектов» таблицу *Purchase*, в раскрывающемся списке — «Ключи», в контектстном меню — «Создать первичный ключ…» .

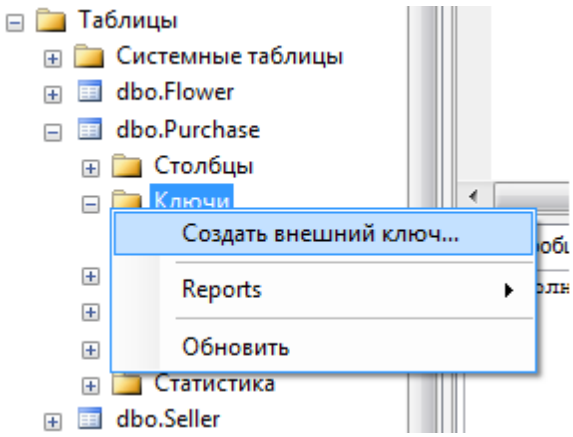


Рисунок 14 – Задание внешнего ключа

Создадим ссылку по внешнему ключу из таблицы *Purchase* (поле *Code_flower*) на таблицу *Flower* (поле *Code_flower*). Для этого в раскрывающемся окне (рисунок 15) в ячейке «Идентификация (Имя)» записываем *FK_Purchase_Flower*, затем открываем для редактирования пункт «Идентификация таблиц и столбцов» (после щелчка мышью по этому полю справа появляется маленький квадратик с точками; нажав на него, переходим на окно).

В качестве таблицы первичного ключа выбираем *Flower*, поле *code_flower*, в качестве таблицы внешнего ключа уже выбрана таблица *Purchase*, надо также здесь выбрать поле *code_flower*. Не забываем нажать ОК.

Теперь после обновления (возможно, потребуется отключиться и заново подключиться к базе данных) среди ключей таблицы *Purchase* должен появиться только что созданный *FK_Purchase_Flower.*

Аналогичным образом создадим ссылку по внешнему ключу из таблицы *Purchase* (поле *Code_seller*) на таблицу *Seller* (поле *Code_seller*).

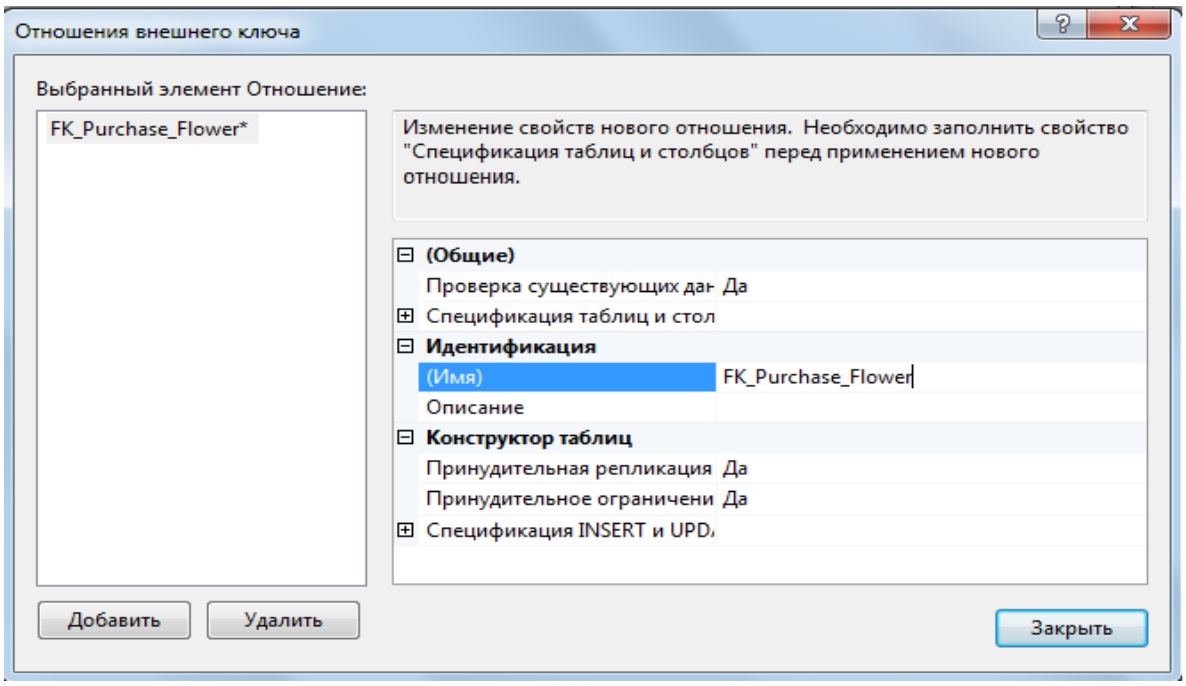


Рисунок 15– Ссылка по внешнему ключу

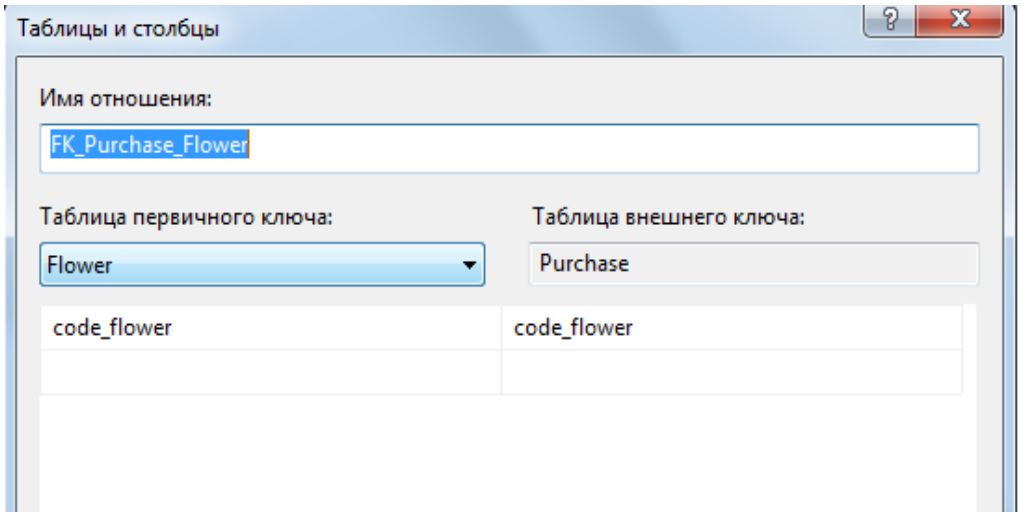


Рисунок 16 – Создание ссылки по внешнему ключу

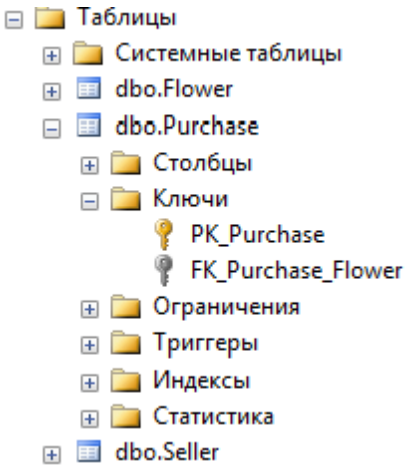


Рисунок 17– Внешние ключи

После того, как ссылки по внешним ключам созданы, создадим схему базы данных (в «Обозревателе объектов» выбираем нашу базу данных, ниже «Диаграммы баз данных», и в контекстном меню - «Создать диаграмму базы данных»; в открывшемся окне включаем в диаграмму все таблицы). Должна получиться диаграмма (схема данных), подобная изображенной на рисунке 18.

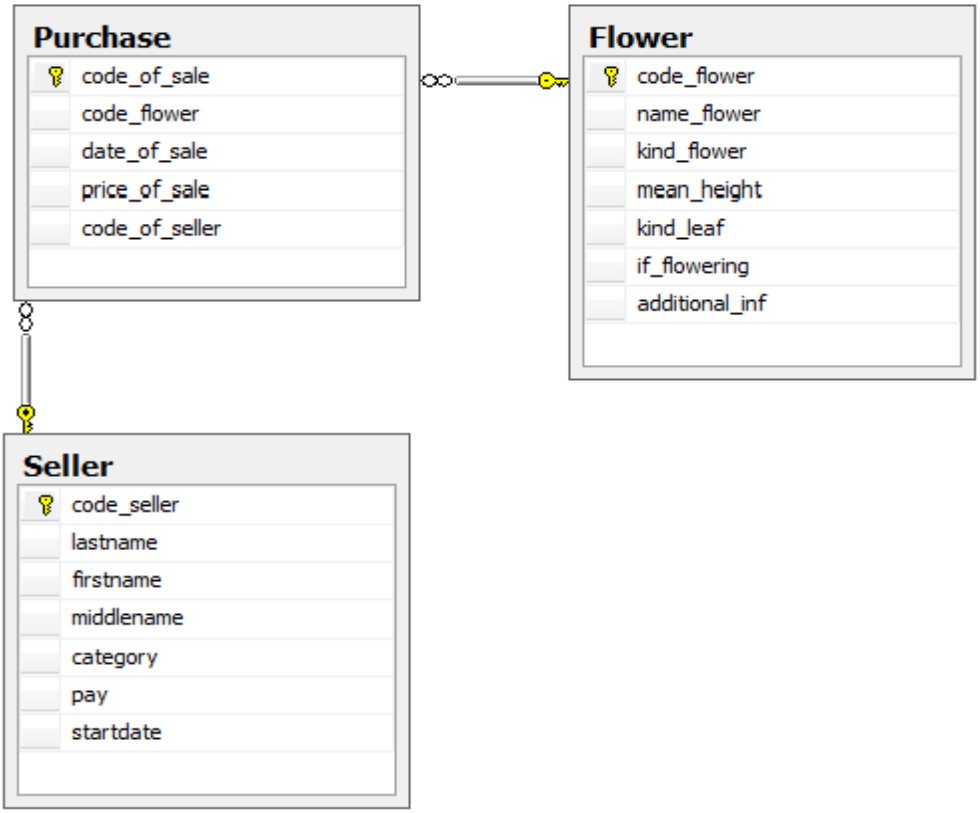


Рисунок 18 – Схема базы данных

2.6 Реализация запросов

В утилите *SQL Server Management Studio* с помощью кнопки «Создать запрос» создать отдельные программы по каждому запросу для каждой таблицы базы данных.

Значения могут быть помещены и удалены из полей тремя командами языка *DML* (Язык Манипулирования Данными):

- *insert* (вставить);
- *update* (изменить);
- *select* (выбрать);
- *delete* (удалить).

1. Команда *insert* имеет свои особенности.

1) При указании значений конкретных полей вместо использования каких-либо значений можно применить ключевое слово *default.*

2) Вставка пустой строки приводит к добавлению пробела, а не значения*null.*

3) Строки и даты задаются в апострофах.

4) Можно задавать *null* явно, а можно задавать *default.*

Добавление одной записи:

insert into ClientInfo (FirstName, LastName, Address, Phone) values('Petr','Petrov','Chehova 1371234567');

Добавление нескольких записей:

insert into ClientInfo (FirstName, LastName, Address, Phone) values('Ivan','Ivanov','Naberejnaya 13','1234568');

insert into ClientInfo (FirstName, LastName, Address, Phone) values(null,'Sidorov','Naberejnaya 25','1234569');

2. Команда *update* позволяет изменять заданные значения записей.

updateClientInfo set FirstName = 'Andrey' where FirstName = 'Petr';

В этом случае в первой записи поля *FirstName* значение *Petr*изменится на *Andrey.*

3. Для выбора некоторых данных из таблицы – команда *select*.

Выбор всех данных таблицы:

select *

from<имя таблицы>

Если необходимо выбрать только определенные столбцы, то нужно их непосредственно указать после слова *select.*

Выбрать сведения о количестве страниц из таблицы *Books* (поле *Pages*).

select Pages

from Books

Выбрать все данные из таблицы *Books.*

*select **

from Books

Для упорядочения данных по какому-то полю необходимо выполнить команду *order by*<имя поля>

Записи можно упорядочивать в восходящем (параметр сортировки asc) или в нисходящем (параметр сортировки *desc*) порядке. Параметр сортировки asc используется по умолчанию.

Выбрать все сведения о книгах из таблицы *Books* и отсортировать результат по коду книги (поле *Code_book*).

*select **

from Books

order by Code_book

Изменение порядка следования полей.

Выбрать все поля из таблицы *Deliveries* таким образом, чтобы в результате порядок столбцов был следующим: *Name_delivery, INN, Phone, Address*, *Code_delivery.*

selectName_delivery, Phone, INN, Address_del, Code_delivery

fromDeliveries

order by inn desc

Выбор некоторых полей из двух таблиц.

Когда нужно выбрать данные, находящиеся в разных таблицах, применяют объединение таблиц. Пусть, например, необходимо выбрать данные, находящиеся в таблицах *Books* и *Authors.*

Простейший шаблон объединения двух таблиц выглядит следующим образом:

*select **

from<таблица1><тип объединения><таблица2>

on<таблица1>.<столбец1> = <таблица2>.<столбец2>

Выбор строк с указанием критериев поиска (*where*).

Предложение *where* применяется для выбора записей, соответствующих определенным критериям. Критерий поиска состоит из одного или нескольких предикатов. Предикат задает проверку, выполняемую для каждой записи таблицы. Результат проверки может принимать одно из трех значений: *«true», «false»,*

«unknown». Команда извлекает для вывода только те строки из таблицы, для которых результат проверки равен *«true».*

Точное совпадение значений одного из полей.

Вывести список названий издательств (поле *Title_book*) из таблицы *Books*, которые находятся в(поле *Publish*) из таблицы *Publishing_house*.

SELECT books.title_book, publishing_house.publish
FROM books INNER JOIN publishing_house
ON books.code_publish = publishing_house.code_publish
WHERE publishing_house.publish='Лань'

Использование вложенного запроса.

Найти название книг, автором которых является Толстой

SELECT title_book
FROM books
WHERE code_author= (SELECTcode_author
FROM authors
WHERE name_author= 'Толстой')

Этот запрос работает если вложенный подзапрос выдает одно число. Если в базе данных есть несколько записей, соответствующих фамилия писателя «Толстой» то запрос завершится с ошибкой в этом случае лучше использовать конструкцию *In*:

SELECT title_book
FROM books
WHERE code_author IN (SELECTcode_author
FROM authors
WHERE name_author= 'Пушкин')

Точное несовпадение значений одного из полей.

Вывести список названий издательств (поле *Publish*) из таблицы Publishing_house, которые не находятся в городе ‘Москва’ (условие по полю City).

SELECT Publish
FROM Publishing_house
WHERE City<> 'Москва'

Выбор записей по диапазону значений (*Between*).

Пример. Вывести фамилии, имена, отчества авторов (поле *Name_author*) из таблицы *Authors*, у которых дата рождения (поле *Birthday*) находится в диапазоне 01.01.1840 – 01.06.1860.

SELECT name_author
FROM Authors

WHERE Birthday BETWEEN '01.01.1940' AND '01.01.1960'

Выбор записей по диапазону значений (*In*).

Вывести список названий книг (поле *Title_book* из таблицы *Books*)и количество (поле *Amount* из таблицы *Purchases*), которые были поставлены поставщиками с кодами 3, 7, 9, 11 (условие по полю *Code_delivery* из таблицы *Purchases*).

SELECT Books.Title_book, Purchases.Amount
FROM Books INNER JOIN Purchases
ON Books.Code_book = Purchases.Code_book
WHERE Purchases.Code_delivery IN (3,7,9,11)

Условие неточного совпадения. Выбор записей с использованием *Like*.

Выбрать из справочника поставщиков (таблица *Deliveries*) названия компаний, телефоны и ИНН (поля *Name_company, Phone*и *INN*), у которых название компании (поле *Name_company*) начинается с «ОАО».

SELECT Name_company, Phone, INN
FROM Deliveries
WHERE Name_company LIKE 'ОАО%'

Пример запроса DELETE.

Команда *delete* удаляет записи из таблицы.

delete from ClientInfo where LastName like 'Petrov';

Результатом этого запроса будет удаление первой записи из таблицы *ClientInfo*. Если не задавать условие, определяющее данные, которые необходимо удалить, то будут удалены все данные таблицы.

Запросы с командами *insert, update* и *delete* могут содержать в себе все прочие конструкции языка *SQL*.

Запросы для оконного приложения.

Для демонстрации запросов добавления, удаления и изменения записей в приложении создадим новое *Windows*-приложение и разместим на нем компоненты пользовательского интерфейса. Форма в режиме дизайна будет иметь следующий вид.

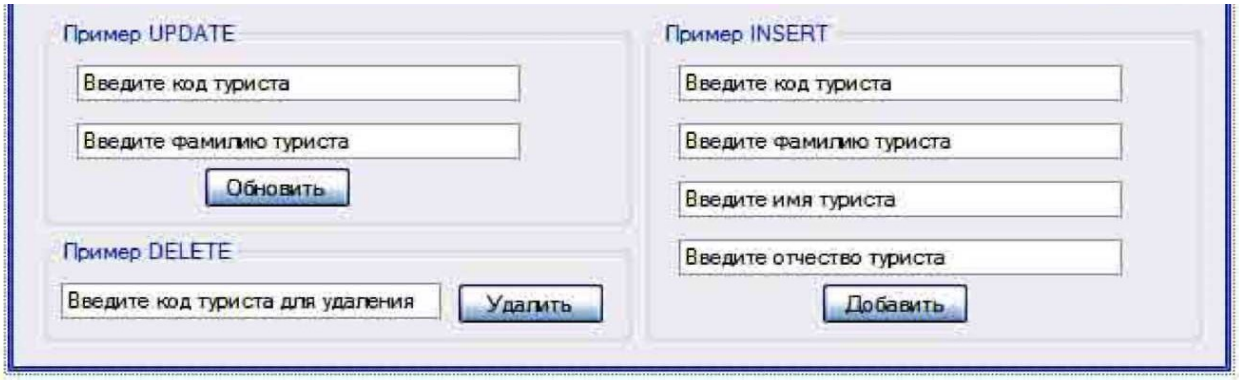


Рисунок 19 – Форма запросов

Подключаем пространство имен для работы с базой данных:

using System.Data.SqlClient;

Реализация запроса на изменение.

В классе формы создаем экземпляр *conn*:

SqlConnectionconn = null;

Обработчик кнопки btnUpdatebудет иметь следующий вид:

private void btnUpdate_Click(object sender, System.EventArgs e)

{try

{string Family = Convert.ToString(this.txtFamilyUpdate.Text);

intTouristID = int. Parse(this.txtTouristIDUpdate.Text);

conn = new SqlConnection();

conn.ConnectionString = @"Data Source=.\SQLEXPRESS;AttachDbFilename=" + @"D:\ВМИ\For ADO\BDTurJirmSQL2.mdf + ";

Integrated Security=True;

Connect Timeout=30;

User Instance=True";

conn.Open();

SqlCommandmyCommand = conn.CreateCommand();

myCommand.CommandText ="UPDATE Туристы SET Фамилия = @Family WHERE [Кодтуриста] = @TouristID";

myCommand.Parameters.Add("@Family", SqlDbType.NVarChar, 50);

myCommand.Parameters["@Family"].Value = Family;

myCommand.Parameters.Add("@TouristID", SqlDbType.Int, 4); myCommand.Parameters["@TouristID"].Value = TouristID;

intUspeshnoeIzmenenie = myCommand.ExecuteNonQuery();

if (UspeshnoeIzmenenie !=0)

{

```
MessageBox.Show("Изменения внесены", "Изменение запи-
си");}
else
{MessageBox.Show("Не удалось внести изменения", "Измене-
ние записи");}
conn.Close();}
catch(Exception ex) {
MessageBox.Show(ex.ToString());}
finally {
conn.Close();} }
```

Обратим внимание на то, что в блоке *finally* происходит закрытие соединения, это нужно сделать в любом случае, независимо от результата выполнения команды. Значения, введенные пользователем в текстовые поля *txtFamilyUpdate* и *txtTouristIDUpdate*, помещаются в переменные *Family*и *TouristID*. В запросе к базе данных используются два параметра - *@Family* и *@TouristID*. Они добавляются в коллекцию объекта *Command* с помощью метода *Add* свойства *Parameters*, а затем значения параметров устанавливаются равными переменным *Family* и *TouristID*.

Реализация запроса на добавление.

Добавим обработчик кнопки *btnInsert*, для добавления записи.

```
private void btnInsert_Click(object sender, System.EventArgs e)
{try
{
intTouristID = int.Parse(this.txtTouristIDInsert.Text);
string Family = Convert.ToString(this.txtFamilyInsert.Text);
string FirstName = Con-
vert.ToString(this.txtFirstNameInsert.Text);
string MiddleName = Con-
vert.ToString(this.txtMiddleNameInsert.Text);
conn = new SqlConnection();
conn.ConnectionString = @"Data
Source=.\SQLEXPRESS;AttachDbFilename=" + @"D:\ВМИ\For
ADO\BDTur_firmSQL2.mdf" + ";
Integrated Security=True;
Connect Timeout=30;
```

```
User Instance=True";
conn.Open();
SqlCommandmyCommand = conn.CreateCommand();
myCommand.CommandText = "INSERT INTO " +"Туристы
([Кодтуриста], Фамилия, Имя, Отчество) " +"VALUES
(@TouristID, @Family, @FirstName, @MiddleName)";
myCommand.Parameters.Add("@TouristID", SqlDbType.Int, 4);
myCommand.Parameters["@TouristID"].Value = TouristID;
myCommand.Parameters.Add("@Family",
SqlDbType.NVarChar, 50);
myCommand.Parameters["@Family"].Value = Family;
myCommand.Parameters.Add("@FirstName",
SqlDbType.NVarChar, 50);
myCommand.Parameters["@FirstName"].Value = FirstName;
myCommand.Parameters.Add("@MiddleName",
SqlDbType.NVarChar, 50);
myCommand.Parameters["@MiddleName"].Value = Middle-
Name;
intUspeshnoeIzmenenie = myCommand.ExecuteNonQuery();
if (UspeshnoeIzmenenie !=0)
{
MessageBox.Show("Изменения внесены", "Изменение запи-
си");}
else
{
MessageBox.Show("Не удалось внести изменения", "Измене-
ние записи");}}
catch(Exception ex) {
MessageBox.Show(ex.ToString());}
finally {
conn.Close();
}}
```

В запросе используются четыре параметра: *@TouristID, @Family, @FirstName, @MiddleName.*Тип данных создаваемых параметров соответствует типу данных полей таблицы «Туристы» в базе.

Реализация запроса на удаление.

Добавим обработчик кнопки *btnDelete*.

```
private void btnDelete_Click(object sender, System.EventArgs e)
```

```
{try
{intTouristID = int.Parse(this.txtTouristlDDelete.Text);
conn = new SqlConnection();
conn.ConnectionString = @"Data
Source=.\SQLEXPRESS;AttachDbFilename=" + @"D:\ВМИ\For
ADO\BDTur_firmSQL2.mdf" + ";
Integrated Security=True;
Connect Timeout=30;
User Instance=True";
conn.Open();
SqlCommandmyCommand = conn.CreateCommand();
myCommand.CommandText = "DELETE FROM Туристы"
+"WHERE [Кодтуриста] = @TouristID";
myCommand.Parameters.Add("@TouristID", SqlDbType.Int, 4);
myCommand.Parameters["@TouristID"].Value = TouristID;
intUspeshnoeIzmenenie = myCommand.ExecuteNonQuery();
if (UspeshnoeIzmenenie !=0)
{
MessageBox.Show("Изменения внесены", "Изменение запи-
си");}
else
{MessageBox.Show("Не удалось внести изменения", "Измене-
ние записи");}
catch(Exception ex)
{MessageBox.Show(ex.ToString());
finally
conn.Close();
}}
```

Запускаем приложение. В каждой из групп заполняем поля, затем нажимаем на кнопки. Проверять результат можно, запуская *Management Studio* и просматривая каждый раз содержимое таблицы.

2.7 Реализация процедур

Теперь вернемся к работе с хранимыми процедурами, содержимое которых представляло собой, по сути, элементарные запросы на выборку в *Windows*-приложениях.

Применение хранимых процедур с параметрами, как правило, связано с интерфейсом приложения - пользователь имеет возмож-

ность вводить значение и затем на основании его получать результат.

Для демонстрации программной работы с хранимыми процедурами создадим новое *Windows*-приложение.

В классе формы создаем строку подключения:

string connectionString = @"Data Source=.\SQLEXPRESS;
Attach DbFilename=" + @"D:\ВМИ\For ADO\BDTur_firmSQL2.mdf" + "integrated Security=True;Connect Timeout=30;User Instance=True";

В каждом из обработчиков кнопок создаем объекты *Connection* и *Command*, определяем их свойства, для последнего добавляем нужные параметры в набор *Parameters*:

private void btnRun_p1_Click(object sender, System.EventArgs e)
{ SqlConnection conn = new SqlConnection();
conn.ConnectionString = connectionString;
SqlCommandmyCommand = conn.CreateCommand();
myCommand.CommandType = CommandType.StoredProcedure;
myCommand.CommandText = "[proc_p1]";
stringFamilyParameter = Convert.ToString(txtFamily_p1.Text);
myCommand.Parameters.Add("@Фамилия", SqlDbType.NVarChar, 50);
myCommand.Parameters["@Фамилия"].Value = FamilyParameter; conn.Open();
SqlDataReaderdataReader = myCommand.ExecuteReader();
while (dataReader.Read())
{
// Создаем переменные, получаем для них значения из объекта *dataReader*, //используя метод *Get* ТипДанных
intTouristID = dataReader.GetInt32(0);
string Family = dataReader.GetString(1);
stringFirstName = dataReader.GetString(2);
stringMiddleName = dataReader.GetString(3); //Выводим данные в элемент*lbResult_p1*
lbResult_p1.Items.Add("Кодтуриста: " + TouristID+ " Фамилия: " + Family + " Имя: "+ FirstName + " Отчество: " + MiddleName);
}
conn.Close(); } // btnRun_p1_Click
private void btnRun_p5_Click(object sender, System.EventArgs e)

```
        {
        SqlConnection conn = new SqlConnection();
        conn.ConnectionString = connectionString;
        SqlCommandmyCommand = conn.CreateCommand();
        myCommand.CommandType = CommandType.StoredProcedure;
        myCommand.CommandText = "[proc_p5]";
        stringNameTourParameter = Con-
vert.ToString(txtNameTour_p5.Text);
        doubleKursParameter = double.Parse(this.txtKurs_p5.Text);
        myCommand.Parameters.Add("@nameTour",
SqlDbType.NVarChar, 50);
        myCommand.Parameters["@nameTour"].Value = Name-
TourParameter;
        myCommand.Parameters.Add("@Курс", SqlDbType.Float, 8);
        myCommand.Parameters["@Курс"].Value = KursParameter;
        conn.Open();
        intUspeshnoeIzmenenie = myCommand.ExecuteNonQuery();
        if (UspeshnoeIzmenenie !=0)
        {
        MessageBox.Show("Измененияявнесены", "Измененизаписи");}
        else
        {
MessageBox.Show("Не удалось внести изменения", "Изменение за-
писи");}
        conn.Close();}
        private void btnRun_proc6_Click(object sender, Sys-
tem.EventArgs e)
        {
        SqlConnection conn = new SqlConnection();
        conn.ConnectionString = connectionString;
        SqlCommandmyCommand = conn.CreateCommand();
        myCommand.CommandType = CommandType.StoredProcedure;
        myCommand.CommandText = "[proc6]"; conn.Open();
        stringMaxPrice = Con-
vert.ToString(myCommand.ExecuteScalar());
        lblPrice_proc6. Text = MaxPrice;
        conn.Close();}
```

На практике наиболее часто используются хранимые процедуры с входными и выходными параметрами.

В классе формы создаем объект *Connection: SqlConnection-conn = null;*

Обработчик кнопки *btnRun* принимает следующий вид:

```
private void btnRun_Click(object sender, System.EventArgs e)
{try
{conn = new SqlConnection();
conn.ConnectionString = @"Data
Source=.\SQLEXPRESS;AttachDbFilename=" + @"D:\ВМИ\For
ADO\BDTur_firmSQL2.mdf" + "integrated Security=True;
Connect Timeout=30;User Instance=True";
SqlCommandmyCommand = conn.CreateCommand();
myCommand.CommandType = CommandType.StoredProcedure;
myCommand.CommandText = "[proc_po1]";
intTouristID = int.Parse(this.txtTouristID.Text);
myCommand.Parameters.Add("@TouristID", SqlDbType.Int, 4);
myCommand.Parameters["@TouristID"].Value = TouristID;
//Необязательная строка, т.к. совпадает со значением по умолчанию.
//myCommand.Parameters["@TouristID"].Direction = Parame-
terDirection.Input;
myCommand.Parameters.Add("@LastName",
SqlDbType.NVarChar, 60);
myCommand.Parameters["@LastName"].Direction = Parame-
terDirection.Output;
conn.Open();
myCommand.ExecuteScalar();
lblFamily.Text = Con-
vert.ToString(myCommand.Parameters["@LastName"].Value);
}
catch (Exception ex)
{   MessageBox.Show(ex.ToString());
} finally
{   conn.Close(); }
```

2.8 Создание форм редактирования, обновления и фильтрации данных

Для создания приложения *WindowsForms*: Файл → Создать → Проект → Приложение *WindowsForms* .В проекте выбираем меню: Сервис → Подключиться к базе данных. В открывшемся окне введите имя сервера и базы данных, нажмите *OK*.

Слева в окне 'Обозреватель серверов' можно увидеть подключенную базу данных.

Создать форму при помощи меню: Проект → добавить форму *Windows*.На каждую форму добавить по компоненту типа *DataGridView.*

Для отображения данных таблицы берем источник данных, для это во вкладке нажать: Выберите источник данных → Добавить источник данных проекта → Базы данных → Набор данных → Далее → Выбрать необходимые таблицы.

Для вызова каждой формы из стартовой, переносят элемент типа *Button* и в соответствующих методах *Click* вызывают формы с помощью кода:

Пример вызова формы по кнопке:

```
для FormAuthors:
FormAuthors myForm3 = new FormAuthors();
myForm3.Show();
```

Для связи с созданными таблицами на форму добавить компонент типа *BindingNavigator.*

Настроить *у BindingNavigator* свойство *BindingSource* для связи с созданной таблицей(значение должно совпадать со значением свойства элемента *DataGridView*). Добавить компонент типа *BindingNavigator*на остальные формы.

Для обновления данных по кнопке используют метод *Click.*

Пример вызова обновления по кнопке:

```
this. Validate();
this.booksBindingSource.EndEdit ();
this.booksTableAdapter.Update (this.dB_BOOKSDataSet.Books);
```

Для применения фильтра данных по определенному полю в методе *Click* кнопки прописывают код:

Пример фильтра по текущему издательству;

```
intbb = dataGridView1.CurrentCell.RowIndex;
booksBindingSource.Filter = "Code_Publish = " +
dataGridView1[4,bb].Value;
```

2.9 Реализация отчетов

Отчеты во многом похожи на формы и тоже позволяют получить результаты работы запросов в наглядной форме, но только не на экране, а в виде распечатки на принтере. Таким образом, в результате работы отчета создается бумажный документ.

На *VisualC#* есть несколько способов создания отчетов. Один из способов создание отчетов это использование генератора отчета *FastReport*. Генератор можно скачать с официального сайта компании. После установки генератора необходимо перезапустить *VisualC#.* Затем необходимо добавить компоненты *FastReport.*

Для создания отчета необходимо поместить компонент *Report* на главную форму. После этого двойным щелчком нажать на компонент и выбрать данные, которые нам нужны для составления отчета. После этого откроется сам редактор отчетов. Для сохранения изменений нужно просто сохранить файл отчета в любом месте.

Отчеты состоят из разделов или секций (*Bands)*, а разделы могут содержать элементы управления. Для настройки разделов надо нажать на рабочей области на кнопку «Настроить бэнды».

1. Структура отчета состоит из следующих разделов: заголовок отчета, подвал отчета, заголовок страницы, подвал страницы, область данных, заголовок колонки, подвал колонки, фоновый.

2. Раздел заголовок отчета служит для печати общего заголовка отчета.

3. Раздел заголовок страницы можно использовать для печати подзаголовков, если отчет имеет сложную структуру и занимает много страниц. Здесь можно также помещать и номера страниц, если это не сделано в нижнем колонтитуле.

4. В области данных размещают элементы управления, связанные с содержимым полей таблиц базы. В эти элементы управления выдаются данные из таблиц для печати на принтере. Эти разделы будут на печати воспроизводиться столько раз, сколько записей присутствует в привязанном запросе или таблице.

5. Раздел подвал страницы используют для тех же целей, что и раздел заголовок страницы. Можно использовать для подстановки полей для подписей должностных лиц, если есть необходимость подписывать отчет на каждой странице.

6. Разделы колонок используют для размещения дополнительной информации или итоговой информации по всем данным отчета. Печатается сверху или снизу области данных.

7. Для предварительного просмотра отчета в том виде, как он будет расположен на бумаге, необходимо вызвать метод *Show* компонента *Report*(на главной форме в меню добавить раздел и в методе *Click*написать этот метод, например *Report1.Show()).*

Пример отчета в режиме «Конструктор» представлен на рисунке.

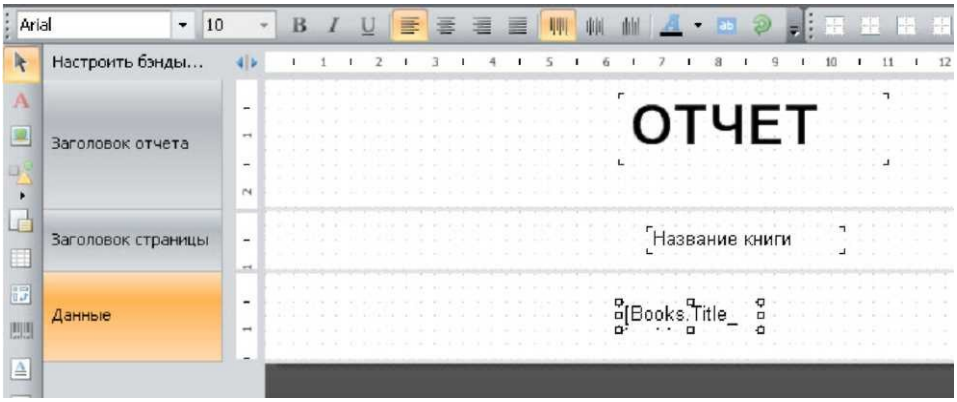


Рисунок 20 - Пример отчета в режиме «Конструктор»

3. ОФОРМЛЕНИЕ ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ

3.1 Оформление текстового материала

Пояснительную записку работы выполняют на листах бумаги формата А4 без рамки. Повреждения листов, помарки текста или графики не допускаются. Правила оформления приведены в следующей таблице.

Таблица 4 – Оформление текстового материала

Наименование элементов	Требования к оформлению
1 Заголовок раздела	
Новая страница	да
Шрифт *Times New Roman* (*pt*)	14 (полужирный)
Интервал до заголовка раздела (*pt*)	0
Интервал после заголовка раздела (*pt*)	20
Выравнивание	по центру
Межстрочное расстояние	1,5
Перенос слов	нет
2 Заголовок подраздела	
Новая страница	нет
Шрифт*Times New Roman* (*pt*)	14 (полужирный)
Интервал до (*pt*)	12
Интервал после (*pt*)	8
Выравнивание	по центру
Межстрочное расстояние	1,5
Перенос слов	Нет
3 Основной текст	
Шрифт *Times New Roman* (*pt*)	14
Абзацный отступ (см)	1,25
Выравнивание	по ширине
Межстрочное расстояние	1,5
Перенос слов	Да
4 Подписи к рисункам и заголовки таблиц	
Шрифт *Times New Roman* (*pt*)	12
Перенос слов	Нет
5 Параметры документа	
Размер бумаги	А4
Верхнее поле	20 мм
Нижнее поле	20 мм
Правое поле	10 мм
Левое поле	30 мм

Текст работы должен быть набран на компьютере по всей ширине страницы с обязательным переносом слов.

Все используемые наименования на иностранных языках должны быть выделены курсивом, например, *Internet*.

Все листы работы нумеруются, начиная с титульного листа. Номер страницы на титульном листе и задании не проставляют. Нумерация страниц текста и приложений, входящих в состав работы, должна быть сквозная. Номер страницы проставляется внизу, справа.

Каждый абзац должен начинаться с красной строки. Каждый абзац должен содержать законченную мысль и состоять, как правило, из 4–5 предложений. Слишком крупный абзац затрудняет восприятие смысла и свидетельствует о неумении четко излагать мысль.

При печати работы необходимо установить запрет "висячих строк", то есть не допускается перенос на новую страницу или оставление на предыдущей странице одной строки абзаца, состоящего из нескольких строк. Следует избегать также оставления на последней строке абзаца части слова или даже одного целого слова.

В этом случае лучше изменить формулировку предложения так, чтобы на последней строке абзаца оставалось не менее трех–четырех слов, либо использовать уплотненный текст, но не более, чем на 0,3 *pt*.

Каждая глава работы должна начинаться с новой страницы. Параграфы следуют друг за другом без вынесения нового параграфа на новую страницу. Не допускается начинать новый параграф внизу страницы, если после заголовка параграфа на странице остается одна–две строки основного текста. В этом случае параграф необходимо начать с новой страницы.

Заголовки глав, а также заголовки введения, заключения, содержания и списка литературы должны быть напечатаны прописными буквами и располагаться посередине строки. Если заголовок состоит из нескольких строк, то интервал между ними должен быть одинарным. Заголовки параграфов начинаются с прописной буквы, последующие буквы – строчные. Точка в конце заголовка не ставится. Точка не ставится и после последней цифры нумерации заголовка подраздела.

Не допускается использование подчеркивания в заголовках. Не допускается также использование двух и более типов выделения в за-

головках (например, курсив и жирный шрифт, курсив и другой цвет, отличный от основного текста). Не допускается также перенос слов в заголовках глав и параграфов.

Размер символов в математических выражениях не должен превышать размер символов основного текста.

Главы, параграфы, пункты и подпункты (кроме введения, заключения, списка использованных источников и приложений) нумеруют арабскими цифрами. Предельная нумерация для подразделов – трехзначная. Внутри пунктов или подпунктов могут быть приведены перечисления, которые отделяют друг от друга точкой с запятой. Перед каждой позицией перечисления следует ставить только дефис, например

– модифицированный метод случайного баланса;

– метод наименьших квадратов с предварительной ортогонализацией факторов;

– метод точечных распределений.

Пример перечислений с вложениями:

а) выполнение – состояние процесса, во время которого процесс обладает всеми необходимыми ресурсами и непосредственно выполняется процессором;

б) ожидание – процесс заблокирован:

1) он не может выполняться по своим внутренним причинам;

2) он ждет осуществления некоторого события, например, завершения операции ввода–вывода;

Слово "глава" не пишется. Заголовки должны четко и кратко отражать содержание раздела. Нельзя использовать аббревиатуры (сокращения) без первого полного упоминания в тексте. Числовые значения величин с обозначением единиц физических величин и единиц счета следует писать цифрами, а числа без обозначения от единицы до девяти – словами, например, 3 км, но – три программы. Если приводится ряд или диапазон числовых значений, выраженных в одной и той же единице физической величины, то ее указывают только после последнего числового значения, например; 1,50; 1,75; 2,00 В; от 10 до 100 Ом. Недопустимо отделять единицу физической величины от числового значения (переносить их на разные строки или страницы).

Количество уровней заголовков в работе определяется тематикой работы и научным руководителем.

3.2 Оформление графического материала

Рисунки. Иллюстрации (чертежи, графики, схемы, диаграммы, фотоснимки, рисунки) следует располагать в работе непосредственно после текста, в котором они упоминаются впервые, или на следующей странице, если в указанном месте они не помещаются. На все иллюстрации должны быть даны ссылки в работе. Иллюстрации должны иметь названия, которые помещают под иллюстрацией. Иллюстрации следует нумеровать арабскими цифрами порядковой нумерацией в пределах глав, например, «Рисунок 2.1 – Блок-схема» (первый рисунок второго раздела).

Если в работе только одна иллюстрация, ее нумеровать не следует и слово "Рисунок" под ней не пишут. Количество рисунков в пояснительной записке должно быть достаточным для того, чтобы ее текст можно было читать с минимальным обращением к документам графической части проекта.

При ссылках на иллюстрации следует писать «... в соответствии с рисунком 1.2». Иллюстрации каждого приложения обозначают отдельной нумерацией арабскими цифрами с добавлением перед цифрой обозначения приложения, например: «Рисунок А.3 – Вид окна ввода».

Если в приложениях большое количество рисунков, то наименования рисунков можно не употреблять, а только его нумеровать. Для схем алгоритмов и программ, располагаемых на нескольких листах пояснительной записки, на первом листе указывается «Рисунок 1.2 – Функциональная схема», на последующих листах – «Продолжение рисунка 1.2».

Если наименование рисунка занимает более одной строки, то межстрочный интервал должен быть одинарным. Точка по окончании наименования рисунка не ставится.

Между рисунком и основным тексом работы (до рисунка и после наименования рисунка) следует оставить пустую строку одинарного интервала 12 *pt*.

Диаграммы. При изображении диаграмм оси координат шкал следует выполнять сплошной толстой линией, а координатную сетку сплошной тонкой линией. Допускается линии сетки выполнять в местах, соответствующих кратным графическим интерва-

лам, или делать засечки вместо линий. Функциональные зависимости нужно выполнять сплошной линией.

При изображении двух и более функциональных зависимостей на одной диаграмме допускается использовать линии различных типов (сплошную, штриховую и т. д.). При этом для каждой функциональной зависимости может быть использована своя шкала.

Единицы измерения на диаграммах наносятся вместе с обозначением переменной величины, после запятой.

Пересечения надписей и линий на диаграмме не допускаются. При недостатке места следует прерывать линию. Подпись диаграммы выполняется по правилу подписей рисунков, например, «Рисунок 2.5 – Диаграмма классов».

Таблицы. Цифровой материал рекомендуется помещать в работе в виде таблиц. Таблицу следует располагать в работе непосредственно после текста, в котором она упоминается впервые, или на следующей странице, а при необходимости - в приложении. Таблицы располагаются по центру документа. До названия таблицы и после самой таблицы должна быть пустая строка интервалом 1,0 размером шрифта 12 *pt*. Между названием таблицы и самой таблицей – интервал 6 *pt*. В шапке таблицы переносы слов запрещены. По возможности в ячейках таблицы переносы не используются.

На все таблицы должны быть ссылки в тексте. При ссылке следует писать слово «таблица» с указанием номера, например, «таблица 2.1». Таблицы следует нумеровать арабскими цифрами порядковой нумерацией в пределах глав. Если в работе одна таблица, ее не нумеруют и слово «Таблица» не пишут.

Слово "Таблица" и наименование таблицы начинаются с прописной буквы, точка в конце заголовка не ставится. Заголовки граф таблицы должны начинаться с прописных букв, подзаголовки – со строчных, если последние подчиняются заголовку. Заголовки граф таблиц должны быть выровнены по центру относительно ячейки.

Допускается располагать таблицу в альбомном формате.

Каждая таблица должна иметь номер и наименование, которые располагаются в центре страницы над таблицей, например,

Таблица 5 – Вероятность появления угроз

Вероятность	**Средняя частота появления**
0	данный вид атаки отсутствует
1	реже, чем 1 раз в год
2	около 1 раза в год
3	около 1 раза в месяц
4	около 1 раза в неделю
5	практически ежедневно

Таблицы слева, справа и снизу, как правило, ограничивают линиями. Если в конце страницы таблица прерывается, то линию, ограничивающую таблицу снизу, не проводят. Высота строк таблицы должна быть не менее 12*pt* . В одной графе должно быть соблюдено, как правило, одинаковое количество десятичных знаков для всех значений величин. При отсутствии отдельных данных в таблице следует ставить прочерк (тире).

Шрифт в таблицах используется *Times New Roman*, 12 *pt*, одинарный интервал. Заголовки граф указываются в единственном числе. Таблицу следует размещать так, чтобы читать ее без поворота работы, если такое размещение невозможно, таблицу располагают так, чтобы ее можно было читать, поворачивая работу по часовой стрелке. При переносе таблицы на другую страницу название столбцов таблицы следует повторить, и над ней по центру размещают слова "Продолжение таблицы" с указанием ее номера. Если шапка таблицы велика, допускается ее не повторять: в этом случае следует пронумеровать графы и повторить их нумерацию на следующей странице. Наименование таблицы не повторяют. Разделять заголовки и подзаголовки боковика и граф диагональными линиями не допускается.

Заменять кавычками повторяющиеся в таблице цифры, математические знаки, знаки процента, обозначения марок материала, обозначения нормативных документов не допускается.

Формулы и уравнения. В формулах в качестве символов следует применять обозначения, установленные соответствующими стандартами. Пояснения символов и числовых коэффициентов, входящих в формулу, если они не пояснены ранее, должны быть приведены непосредственно под формулой. Пояснения каждого символа следует давать с новой строки в той последовательности, в которой они приведены в формуле. Первая строка пояснения должна начинаться со

слова «где», без двоеточия после него, без абзацного отступа. Набор формул необходимо осуществлять в соответствующем редакторе.

Формулы и математические уравнения рекомендуется набирать в редакторе формул *Microsoft Equation* 3.0 по требованиям ГОСТ 7.32-2001.

Требования к размерам: обычный символ 14 *pt*; крупный индекс 10*pt*; мелкий индекс 8*pt*; крупный символ 20*pt*; мелкий символ 14*pt*.

Уравнения и формулы следует выделять из текста в отдельную строку. Выше и ниже каждой формулы или уравнения должно быть оставлено не менее одной свободной строки одинарного интервала, размером шрифта 12 *pt*без отступов. Если уравнение не умещается в одну строку, оно должно быть перенесено после знака равенства (=) или после знака (+), или после других математических знаков с их обязательным повторением в новой строке. Например,

$$\sum_{g=1}^{N} x_{g_0} \cdot \widetilde{x}_{g_i}^2 = \sum_{g=1}^{n} x_{g_0} \left(x_i^2 - \frac{\sum_{g=1}^{N} x_{g_i}^2}{N} \right) = \sum_{g=1}^{N} x_{g_0} x_i^2 - \sum_{g=1}^{N} x_{g_0} \left(\frac{\sum_{g=1}^{N} x_{g_i}^2}{N} \right) =$$

$$= \sum_{g=1}^{N} x_{g_i}^2 - N \cdot \frac{\sum_{g=1}^{N} x_{g_i}^2}{N} = 0, \qquad (3.1)$$

Формулы и уравнения в работе следует нумеровать порядковой нумерацией в пределах главы в круглых скобках в крайне правом положении напротив формулы, например,

$$\frac{\sum_{i=1}^{k} S_{i\bullet\bullet}^2}{k} = \frac{\sum_{i=1}^{k}\sum_{j=1}^{m} (\bar{Y}_{ij\bullet} - \bar{Y}_{i\bullet\bullet})^2}{k(m-1)} \approx \frac{\sigma_{H.II}^2}{n} + \sigma_{C.II}^2, \qquad (3.2)$$

где $\bar{Y}_{i\bullet\bullet} = \frac{1}{m}\sum_{j=1}^{m} \bar{Y}_{ij\bullet}$ для всех $i = \overline{1,k}$;

n– объем выборки.

Если в работе только одна формула, то их не нумеруют.

Если в работе используются русские буквы для обозначения переменных, к ним применяются те же правила оформления.

Ссылки. При ссылке на учебник или пособие после напоминания о нем в тексте работы проставляют в квадратных скобках номер, под которым оно значится в библиографическом списке. В необходимых случаях (обычно при использовании цифровых данных или цитаты) указываются и страницы, на которых помещается используемый источник, например, [9, с. 4-5]. Ссылки на таблицы, рисунки, приложения берутся в круглые скобки. При ссылках следует писать: "в соответствии с данными в таблице 5.1" или (таблица 5.1), "по данным рисунка 3.1" или (рисунок 3.1), "в соответствии с приложением А" или (приложение А).

3.3 Оформление списка использованных источников

Список должен содержать перечень источников, использованных при выполнении курсовой работы.

В список литературы включают все источники в алфавитном порядке авторов. Сначала оформляются монографии, учебники, затем справочники, затем периодические издания (журналы), затем патентные документы, ГОСТы, СНиПы и др., в заключении Интернет-ссылки. Выполнение списка и ссылки на него в тексте - по ГОСТ 7.1-84 и ГОСТ 7.32-2001.

Сведения о книгах (монографии, учебники, справочники и т. д.) должны включать: фамилию и инициалы автора, заглавие книги, место издания, издательство, год издания, количество страниц в книге. Допускается сокращение названий городов - М. (Москва), Л. (Ленинград), К. (Киев), Мн. (Минск), СПб. (Санкт-Петербург). Например,

1) Шляндин В.М. Цифровые измерительные устройства. – М.: Высшая школа, 1991. – 335с.

Сведения о статье из периодического издания должны включать фамилию и инициалы автора, заглавие статьи, наименование серии (если есть), год выпуска, том (при необходимости), номер издания (журнала), страницы, на которых помещается статья. Например,

1)Пестов Е.Н., Мокренко П.В. Прецизионный квантовый преобразователь тока. //Приборы и системы управления, 1988. – №9. –с. 25-28.

Сведения о патентных документах должны включать: характер документа, его номер, страну, выдавшую документ, название, инициалы и фамилию автора, страну, из которой данный автор, когда и где опубликован документ. Например,

3) А. с. 436350 СССР. Двоичный сумматор / К.Н. Корнеев (СССР).–Заявл. 12.01.82; Опубл. 30.03.84, Бюл. №26.

Пример оформления статей из иностранного журнала:

5) GanagisowaT.,KawashimaI. Activegurutor // Electronicletters. – 1988. – Vol.3, № 3. – p. 5-8.

Сведения о стандартах и технических условиях выполняются следующим образом:

4) ГОСТ 7.32–2001. Отчет о научно-исследовательской работе. Структура и правила оформления. – М.: Изд-во стандартов, – 2001.– 18с.

Пример оформления списка использованной литературы в приложении Д.

В тексте курсовой работы обязательно должны присутствовать ссылки на использованную литературу. Ссылка должна размещаться в конце текста, взятого из источника, в квадратных скобках с указанием номера данного источника в списке литературы.

3.4 Оформление приложений

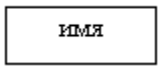
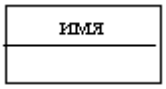
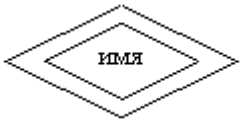
Приложения следует оформлять как продолжение курсовой работы на её последующих страницах после списка использованных источников. Каждое приложение должно начинаться с новой страницыи в графе с названием приложения следует писать слово «ПРИЛОЖЕНИЕ» с соответствующей нумерацией. Если приложений более одного, то они обозначаются прописными буквами русского алфавита, начиная с буквы А, исключая буквы Ё, З, И, И, О, Ч, Щ, Ь, Ы, Ъ, после буквы Я приложения обозначаются арабскими цифрами.

Листы работы, имеющие формат более А4помещаются в качестве приложений и складываются по формату листов работы. Если приложений много, они оформляются отдельной книгой, на

титульном листе которой должно быть написано прописными буквами слово «ПРИЛОЖЕНИЯ». Нумерация страниц, на которых расположены приложения, должна производиться только в содержании.

3.6 Оформление элементов нотаций, используемых при построении диаграмм "сущность-связь"

Таблица 6 –Нотация Чена

Элемент диаграммы	**Обозначает**
	независимая сущность
	зависимая сущность
	родительская сущность в иерархической связи
	Связь
	идентифицирующая связь
	Атрибут
	первичный ключ
	внешний ключ (понятие внешнего ключа вводится в реляционной модели данных)
	многозначный атрибут
	получаемый (наследуемый) атрибут в иерархических связях

Связь соединяется с ассоциируемыми сущностями линиями. Возле каждой сущности на линии, соединяющей ее со связью, цифрами указывается класс принадлежности.

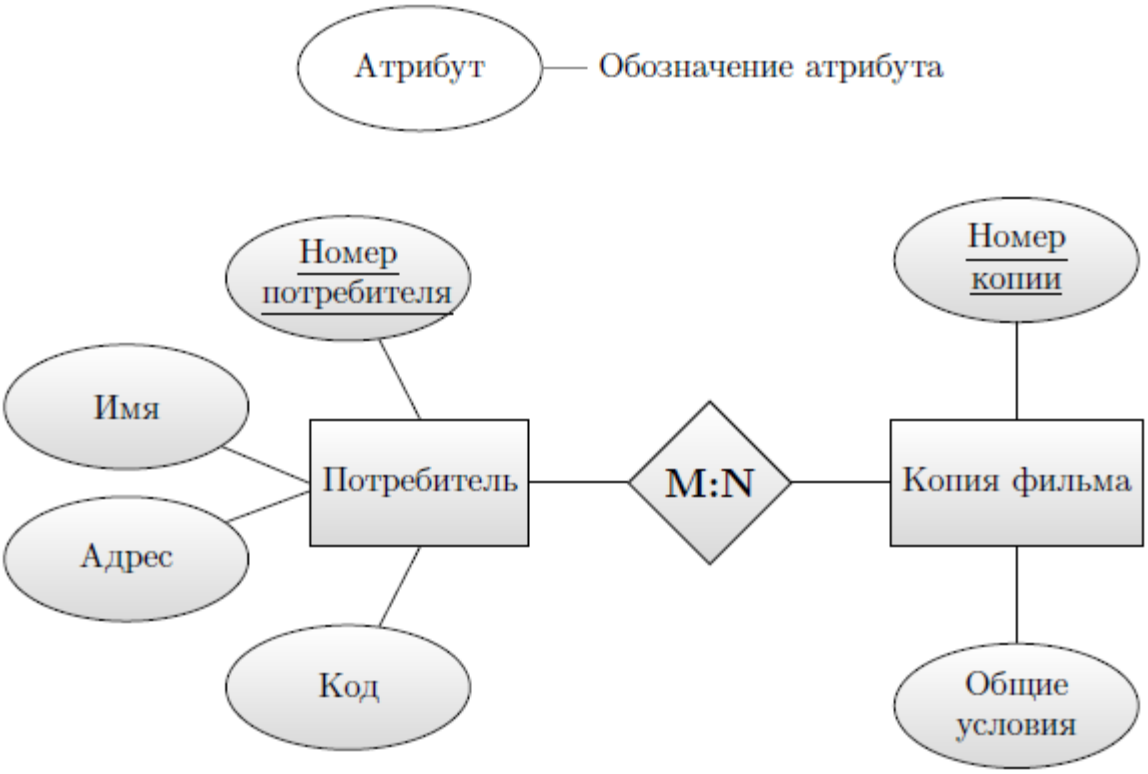


Рисунок 21– Пример диаграммы «сущность - связь» в нотации Чена

Рисунок 22– Пример диаграммы «сущность - связь» в нотации Чена

ЗАКЛЮЧЕНИЕ

В современном мире в основе любой информационной системы лежит база данных, а точнее СУБД. И выбор той или иной СУБД существенно влияет на функциональные возможности информационной системы и проектные решения. Предложение на рынке СУБД огромно, и перед разработчиком встает сложный выбор, какую СУБД использовать. Ситуация усугубляется при необходимости обеспечить поддержку различных источников данных, причем каждый из таких источников данных может хранить и обрабатывать данные по- своему. Кроме того, в различных языках программирования различна поддержка работы с той или иной СУБД. То есть, еще возникает проблема несоответствия обработки информации большинством СУБД и способам обработки информации различными языками программирования.

Решение выдвинутых проблем предлагается при помощи технологии *ADO .NET*, разработанной компанией *Microsoft*, и включенной в их новую платформу *.NET Framework. ADO .NET*, как часть *Microsoft .NET Framework*, представляет собой набор средств и слоев, позволяющих приложению легко управлять и взаимодействовать со своим файловым или серверным хранилищем данных.

Технология *ADO.NET* в полной мере способна предоставить механизм для доступа к любому источнику данных, тем самым давая разработчику мощный механизм взаимодействия с базами данных, способный в полной мере реализовать все потребности, возникающие при проектировании ИС.

ЛИТЕРАТУРА

1. Астахова И.Ф. , SQL в примерах и задачах; Учеб. пособие /– Мн.: Новое знание, 2002. – 176 с.
2. Мамаев Е. В. MicrosoftSQLServer 2000.−СПб.:БХВ–Петербург,2005.−1280с.
3. О. Н. Евсеева, А. Б. Шамшев, Работа с базами данных на языке C#. Технология ADO.NET: Учебное пособие/ - Ульяновск: УлГТУ, 2009. - 170 с.
4. ADO.NET: Обзор технологии [Электронный ресурс]. - Режим доступа: http://www.cyberguru.ru/dotnet/ado-net/adonet-overview.html, свободный. - Загл. с экрана.
5. C# 2005 для профессионалов / К. Нейгел, Б. Ивьен, Д. Глинн, К. Уотсон, М. Скиннер, А. Джонс. - Москва; Санкт-Петербург; Киев: «Диалектика», 2007.−180с.
6. Воройский, Ф. С. Информатика. Новый систематизированный толковый словарь-справочник / ФИЗМАТЛИТ, 2003. −158 с.
7. Кариев, Ч. А. Разработка Windows-приложений на основе Visual C# / Ч. А. Кариев. - М. : БИНОМ. Лаборатория знаний, Интернет-университет информационных технологий - ИНТУИТ.ру, 2007. −328 с.
8. Классы, интерфейсы и делегаты в. C# 2005 : учебное пособие / сост. О. Н. Евсеева, А. Б. Шамшев. - Ульяновск : УлГТУ, 2008.
9. Лабор, В. В. Си Шарп создание приложений для Windows / В. В. Лабор. - Минск, Харвест, 2003. −172 с.
10. Петцольд, Ч. Программирование для MicrosoftWindows на C# / Ч. Пет- цольд. В 2 т. : пер. с англ. - М. : Издательско-торговый дом «Русская Редакция», 2002. −183 с.

КУРСОВАЯ РАБОТА ПО УПРАВЛЕНИЮ ДАННЫМИ

Методические указания