МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Югорский государственный университет» (ЮГУ)
НЕФТЯНОЙ ИНСТИТУТ
(ФИЛИАЛ) ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ «ЮГОРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(НефтИн (филиал) ФГБОУ ВО «ЮГУ»)

РАССМОТРЕНО
На заседании ПЦК МиЕНД
Протокол заседания
№ 07 от «31» августа 2022 г.
_____ Бойко Я.С

УТВЕРЖДАЮ
Зам. директора по УВР
НефтИн (филиал) ФГБОУ ВО «ЮГУ»
«31» августа 2022 г.
_____Хайбулина Р.И

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

КОМПЛЕКТ КОНТРОЛЬНО-ИЗМЕРИТЕЛЬНЫХ МАТЕРИАЛОВ ПО УЧЕБНОЙ ДИСЦИПЛИНЕ

(МЕЖДИСЦИПЛИНАРНОМУ КУРСУ)

МДК.01.02 БАЗЫ ДАННЫХ
(наименование учебной дисциплины, МДК)

программы подготовки специалистов среднего звена (ППССЗ)

по специальности СПО

10.02.05 Обеспечение информационной безопасности автоматизированных систем
(код, наименование)
программы подготовки специалистов среднего звена (ППССЗ)

базовой подготовки

г. Нижневартовск

-2022-

Комплект контрольно-измерительных материалов по учебной дисциплине МДК.01.02 Базы данных программы подготовки специалистов среднего звена (ППССЗ) разработан на основе Федерального государственного образовательного стандарта (далее – ФГОС) по специальности среднего профессионального образования (далее – СПО) 10.02.05 Обеспечение информационной безопасности автоматизированных систем, в соответствии с рабочей программой учебной дисциплины МДК.01.02 Базы данных.

Разработчик:

НефтИн (филиал) ФГБОУ ВО «ЮГУ»	преподаватель	А.Н. Тымощук
(место работы)	(занимаемая должность)	(инициалы, фамилия)

1. Паспорт комплекта контрольно-измерительных материалов

1.1. Область применения

Комплект контрольно-измерительных материалов предназначен для проверки результатов освоения учебной дисциплины (далее - УД) МДК.01.02 Базы данных программы подготовки специалистов среднего звена (ППССЗ) по специальности СПО *10.02.05 Обеспечение информационной безопасности автоматизированных систем.*

Комплект контрольно-измерительных материалов позволяет оценивать:

1.1.1. Освоение профессиональных компетенций (ПК) и общих компетенций (ОК)

Профессиональные и общие компетенции	Средства проверки (№ задания)
ПК 1.1. Производить установку и настройку компонентов автоматизированных (информационных) систем в защищенном исполнении в соответствии с требованиями эксплуатационной документации.	ПЗ 1-24, Выполнение практических работ
ОК 01. Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам	ПЗ 1-24, Выполнение практических работ
ОК 02. Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности.	ПЗ 1-24, Выполнение практических работ
ОК 03. Планировать и реализовывать собственное профессиональное и личностное развитие.	ПЗ 1-24, Выполнение практических работ
ОК 04. Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами.	ПЗ 1-24, Выполнение практических работ
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста.	ПЗ 1-24, Выполнение практических работ

ОК 06. Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных общечеловеческих ценностей, применять стандарты антикоррупционного поведения.	ПЗ 1-24, Выполнение практических работ
ОК 07. Содействовать сохранению окружающей среды, ресурсосбережению, эффективно действовать в чрезвычайных ситуациях.	ПЗ 1-24, Выполнение практических работ
ОК 08. Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности.	ПЗ 1-24, Выполнение практических работ
ОК 09. Использовать информационные технологии в профессиональной деятельности.	ПЗ 1-24, Выполнение практических работ
ОК 10. Пользоваться профессиональной документацией на государственном и иностранном языках.	ПЗ 1-24, Выполнение практических работ

1.1.2. Освоение умений и усвоение знаний

Освоенные умения, усвоенные знания	№ заданий для проверки
1	2
У1: организовывать, конфигурировать, производить монтаж, осуществлять диагностику и устранять неисправности компьютерных сетей, работать с сетевыми протоколами разных уровней	ПЗ 1-24
У2: настраивать и устранять неисправности программно-аппаратных средств защиты информации в компьютерных сетях по заданным правилам	ПЗ 1-24
У3: обеспечивать работоспособность, обнаруживать и устранять неисправности	ПЗ 1-24
З1: теоретические основы компьютерных сетей и их аппаратных компонент, сетевых моделей,	ПЗ 1-24

протоколов и принципов адресации	
З2: принципы построения, физические основы работы периферийных устройств	ПЗ 1-24
З3: принципы основных методов организации и проведения технического обслуживания вычислительной техники и других технических средств информатизации	ПЗ 1-24

(Освоенные умения и усвоенные знания перечисляются из части ФГОС соответствующей УД (МДК), при наличии здесь указываются и дополнительные умения, и знания, введенные за счёт вариативной части).
Номера заданий для проверки – номера методических указаний к выполнению ЛПЗ, номера заданий в методические указания в ЛПЗ)

1.2. Система контроля и оценки освоения программы учебной дисциплины МДК.01.02 Базы данных.

1.2.1. Формы промежуточной аттестации по ППССЗ при освоении учебной дисциплины МДК.01.02 Базы данных.

Учебная дисциплина (междисциплинарный курс)	Формы промежуточной аттестации
1	2
МДК.01.02 Базы данных 3 семестр	Дифференцированный зачет
МДК.01.02 Базы данных 5 семестр	Экзамен

(Формы промежуточной аттестации указываются в соответствии с учебным планом)

1.2.2. Организация контроля и оценки освоения программы учебной дисциплины МДК.01.02 Базы данных.

Промежуточный контроль по дисциплине осуществляется форме экзамена.

Условием положительной аттестации по дисциплине на экзамене является положительная оценка освоения всех умений, знаний, а также формируемых профессиональных компетенций по всем контролируемым показателям.

2. Комплект материалов для оценки уровня освоения умений и усвоения знаний, сформированности общих и профессиональных компетенций при изучении учебной дисциплины МДК.01.02 Базы данных.

2.1. Комплект материалов для оценки уровня освоения умений, усвоения знаний, сформированности общих и профессиональных компетенций

ТЕМЫ РЕФЕРАТОВ (ДОКЛАДОВ)

по учебной дисциплине МДК.01.02 Базы данных

1. История развития, назначение и роль баз данных.
2. Файловые системы и базы данных.
3. Структуры данных и базы данных.
4. Способы хранения информации в базах данных.
5. Способы повышения эффективности обработки данных за счет их организации.
6. Общая характеристика, назначение, возможности, состав и архитектура СУБД.
7. Классификация СУБД.
8. Информационное, лингвистическое, математическое, аппаратное, организационное, правовое обеспечения СУБД.
9. Типология баз данных. Документальные базы данных. Фактографические базы данных.
10. Типология баз данных. Гипертекстовые и мультимедийные базы данных.
11. Типология баз данных. Объектно-ориентированные базы данных.
12. Типология баз данных. Распределенные базы данных. Коммерческие базы данных.
13. Недостатки реляционных СУБД.
14. Объектные расширения реляционных СУБД.
15. Средства автоматизации проектирования баз данных.
16. Централизация логики приложения на сервере базы данных.
17. Информационные хранилища. OLAP-технология.
18. XML-серверы.
19. Принципы построения БД.
20. Проблема создания и сжатия больших информационных массивов, информационных хранилищ и складов данных.
21. Фрактальные методы в архивации.
22. Управление складами данных.
23. Средства поддержания целостности базы данных
24. Серверы баз данных.
25. Многоплатформеные СУБД. СУБД Oracle.
26. Многоплатформеные СУБД. Informix.
27. Многоплатформеные СУБД. Sybase.
28. Многоплатформеные СУБД. DB2.
29. Многоплатформеные СУБД. MySQL.
30. СУБД, ориентированные на конкретные платформы. СУБД DBManager в OS/2.
31. СУБД, ориентированные на конкретные платформы. СУБД SQL/400 в AS/400.
32. СУБД, ориентированные на конкретные платформы. СУБД Access в Microsoft Windows.

33. СУБД семейства XBase, Dbase.
34. Базы данных реального времени.
35. Жизненный цикл базы данных.
36. Циклическая база данных.
37. Сжатие без потерь в реляционных СУБД.
38. Защита информации в СУБД.
39. Нормальные формы: НФБК. 3 примера.
40. Нормальные формы: 4НФ. 3 примера.
41. Нормальные формы: 5НФ. Описание. 3 примера.
42. Хранение деревьев в реляционных базах данных.
43. Способы переноса данных с одного типа БД в другую. На примере переноса данных из MySQL в Access.
44. Способы переноса данных с одного типа БД в другую. На примере переноса данных из Access в MySQL.
45. Экспорт/импорт между базами данных различных производителей.
46. Реальные и фантастические разработки БД.
47. Физическое хранение реляционных таблиц.
48. Сериализация транзакций в БД.
49. Анализ качества баз данных.
50. Пути формирования баз данных для директ-маркетинга.
51. Архитектура и функционирование адресных баз данных.
52. Сверхбольшие базы данных.
53. Эксплуатация баз данных. Состав, порядок планирования и проведения регламентных работ.
54. Эксплуатация баз данных. Сервисные средства СУБД.
55. Эксплуатация баз данных. Задачи администратора базы данных.
56. Эксплуатация баз данных. Организация труда обслуживающего персонала.

Критерии оценки:

Оценка «5» ставится, если:

• Содержание реферата соответствует теме;
• Тема раскрыта полностью;
• Оформление реферата соответствует принятым стандартам;
• При работе над рефератом автор использовал современную литературу;
• В реферате отражена практическая работа автора по данной теме;
• В сообщении автор не допускает ошибок, но допускает оговорки по невнимательности, которые легко исправляет по требованию учителя;
• Сообщение логично, последовательно, технически грамотно;
• На дополнительные вопросы даются правильные ответы,

Оценка «4» ставится, если:

• Содержание реферата соответствует теме;
• Тема раскрыта полностью;
• Оформление реферата соответствует принятым стандартам;
• При работе над рефератом автор использовал современную литературу;

• В реферате отражена практическая работа автора по данной теме;

• В сообщении автор допускает одну ошибку или два-три недочета, допускает неполноту

ответа, которые исправляет только с помощью учителя.

Оценка «3» ставится, если:

• Содержание реферата не полностью соответствует теме;

• Тема раскрыта недостаточно полно;

• В оформлении реферата допущены ошибки;

• Литература, используемая автором, при работе над рефератом устарела;

• В реферате не отражена практическая работа автора по данной теме;

• Сообщение по теме реферата допускаются 2-3 ошибки;

• Сообщение неполно, построено несвязно, но выявляет общее понимание работы;

• При ответе на дополнительные вопросы допускаются ошибки, ответ неуверенный, требует постоянной помощи учителя.

Оценка «2» ставится, если:

• Содержание реферата не соответствует теме;.

Оценка «1» ставится, если:

• Обучающийся не представил рефератную работу соответствующую выбранной теме.

Составители _____ /А.Н. Тымощук /
подпись Ф.И.О.

__.__.2022 г.

КОМПЛЕКС ПРАКТИЧЕСКИХ ЗАНЯТИЙ
по учебной дисциплине МДК.01.02 Базы данных

ТЕМАТИКА ПРАКТИЧЕСКИХ ЗАНЯТИЙ

№	Название практического занятия	Кол-во часов
1.	Операции над отношениями	2
2.	Операции над отношениями	2
3.	Проектирование инфологической модели данных	2
4.	Проектирование структуры базы данных	2
5.	Проектирование структуры базы данных	2
6.	Проектирование базы данных с использованием CASE-средств	2
7.	Проектирование базы данных с использованием CASE-средств	2
8.	Создание базы данных средствами СУБД. Работа с таблицами: добавление, редактирование, удаление, навигация по записям.	2
9.	Создание взаимосвязей	2
10.	Сортировка, поиск и фильтрация данных	2
11.	Способы объединения таблиц	2
12.	Создание базы данных с помощью команд SQL. Редактирование, вставка и удаление данных средствами языка SQL	2
13.	Создание и использование запросов. Группировка и агрегирование данных	2
14.	Коррелированные вложенные запросы	2
15.	Создание в запросах вычисляемых полей. Использование условий	2
16.	Управление доступом к объектам базы данных	2
17.	Установка СУБД. Настройка компонентов СУБД.	2
18.	Создание форм и отчетов	2
19.	Создание меню. Генерация, запуск.	2
20.	Профилирование запросов клиентских приложений.	2
21.	Разработка хранимых процедур и триггеров	2
22.	Управление правами доступа к базам данных	2
23.	Аудит данных с помощью средств СУБД и триггеров	2
24.	Резервное копирование и восстановление баз данных	2
		48

Практическая работа № 1-2

Тема: Операции над отношениями

Цель: Получить практический опыт выполнения операций над отношениями.

Краткие теоретические сведения

В основе реляционной модели данных лежит понятие отношения, Отношение - множество элементов, называемых кортежами, элементы кортежа – атрибуты (поля). Длина кортежа (количество атрибутов) определяет арность отношения, количество кортежей – мощность отношения.

1 В соответствие с заданием создать и заполнить таблицы БД "Фирма", установить связи между ними.

2 Продемонстрировать на компьютере заполненные таблицы, схему данных.

3 Ответить на контрольные вопросы.

4 Сделать вывод о проделанной работе.

Порядок выполнения:

1. Запустите **Microsoft Access**.

2. Создайте базу данных **Фирма**. Сотрудники данной организации работают с клиентами и выполняют их заказы.

3. Создайте в **режиме Конструктор** 3 таблицы: ***Сотрудники, Клиенты и Заказы***. Если все сведения поместить в одной таблице, то она станет очень неудобной для работы. В ней начнутся повторы данных. Всякий раз, когда сотрудник Иванов будет работать с какой-либо фирмой, придется прописывать данные о сотруднике и клиенте заново, в результате чего можно допустить множество ошибок. Чтобы уменьшить число ошибок, можно исходную таблицу разбить на несколько таблиц и установить связи между ними. Это будет более рационально.

Таблица Сотрудники

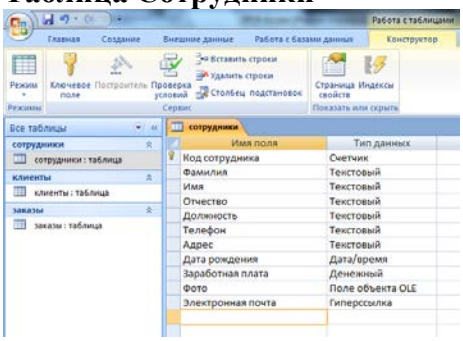


Таблица Клиенты

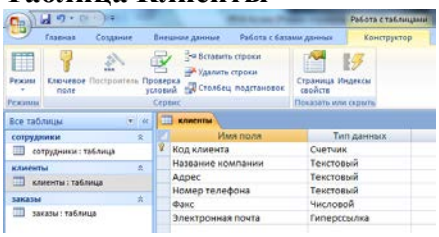
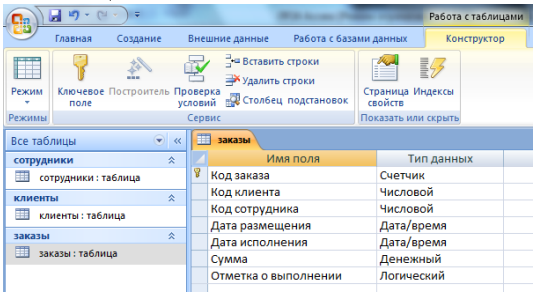


Таблица Заказы



4. Установите ключевые поля.

Отдельные таблицы, содержащие информацию по определенной теме, необходимо связать в единую структуру базы данных. Для связывания таблиц следует задать **ключевые поля**.

Ключ состоит из одного или нескольких полей, значения которых однозначно определяют каждую запись в таблице. Наиболее подходящим в качестве ключевого поля является *Счетчик*, так как значения в данном поле являются уникальными (т. е. исключают повторы).

При создании таблиц в режиме конструктора ключевое поле устанавливается автоматически. Откройте созданные Вами таблицы в **режиме Конструктор** и проверьте установленные ключевые поля:

1) в таблице *Сотрудники* ключевое поле *Код сотрудника*

сотрудники

Имя поля	Тип данных
Код сотрудника	Счетчик
Фамилия	Текстовый
Имя	Текстовый
Отчество	Текстовый
Должность	Текстовый
Телефон	Текстовый
Адрес	Текстовый
Дата рождения	Дата/время
Заработная плата	Денежный
Фото	Поле объекта OLE
Электронная почта	Гиперссылка

2) в таблице *Клиенты* ключевое поле *Код клиента*

клиенты

Имя поля	Тип данных
Код клиента	Счетчик
Название компании	Текстовый
Адрес	Текстовый
Номер телефона	Текстовый
Факс	Числовой
Электронная почта	Гиперссылка

2) в таблице *Заказы* ключевое поле *Код заказа*

3) в таблице *Заказы* ключевое поле *Код заказа*

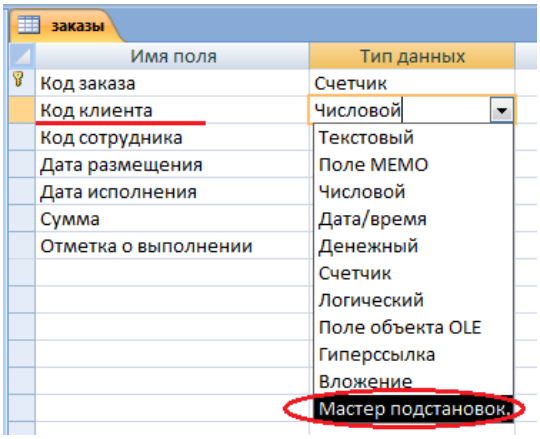
заказы

Имя поля	Тип данных
Код заказа	Счетчик
Код клиента	Числовой
Код сотрудника	Числовой
Дата размещения	Дата/время
Дата исполнения	Дата/время
Сумма	Денежный
Отметка о выполнении	Логический

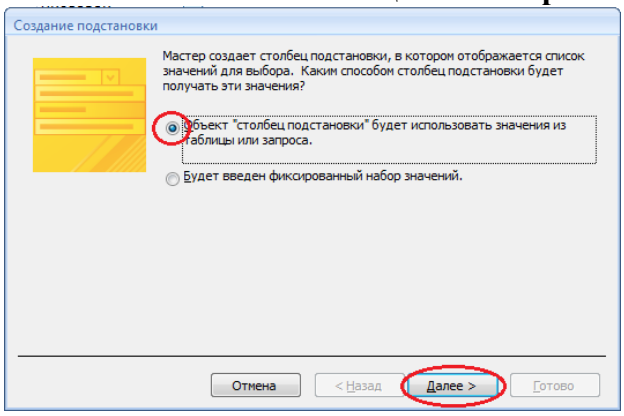
4. Создайте раскрывающиеся списки с помощью **Мастера подстановок.**

Таблица **Заказы** содержит поля **Код сотрудника** и **Код клиента**. При их заполнении могут возникнуть некоторые трудности, так как не всегда удается запомнить все предприятия, с которыми работает фирма, и всех сотрудников с номером кода. Для удобства можно создать раскрывающиеся списки с помощью **Мастера подстановок**.

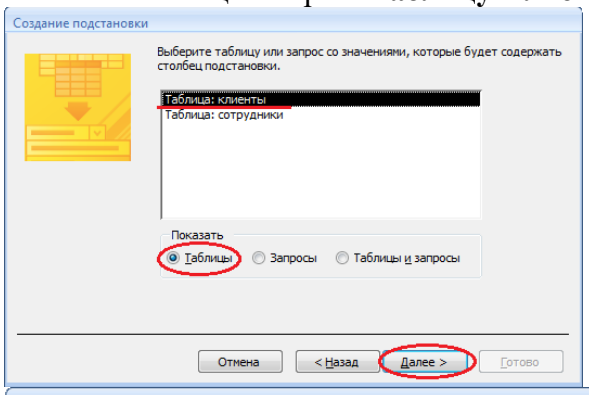
Откройте **таблицу Заказы** в режиме **Конструктора**. Для поля **Код клиента** выберите **тип данных Мастер подстановок**.

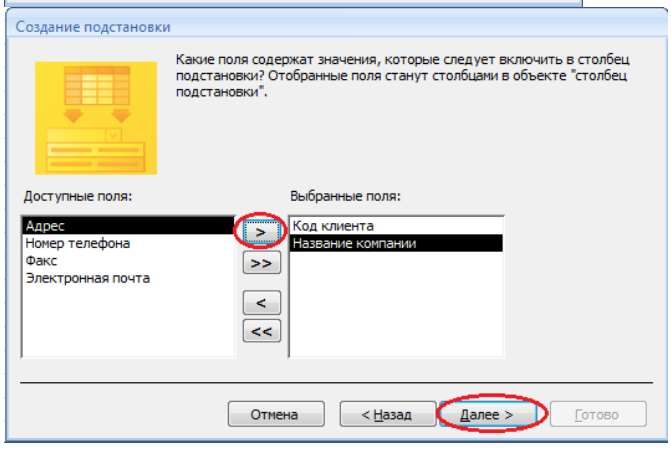


В появившемся окне выберите команду **Объект "столбец подстановки" будет использовать значения из таблицы или запроса** и щелкните на кнопке **Далее.**

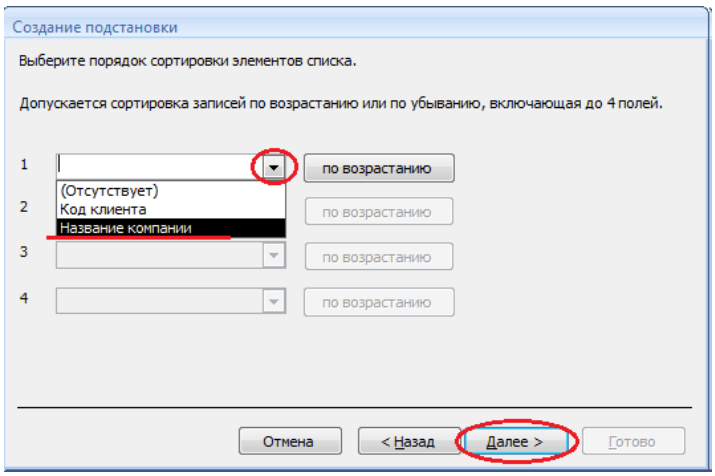


В списке таблиц выберите **таблицу Клиенты** и щелкните на кнопке **Далее**.

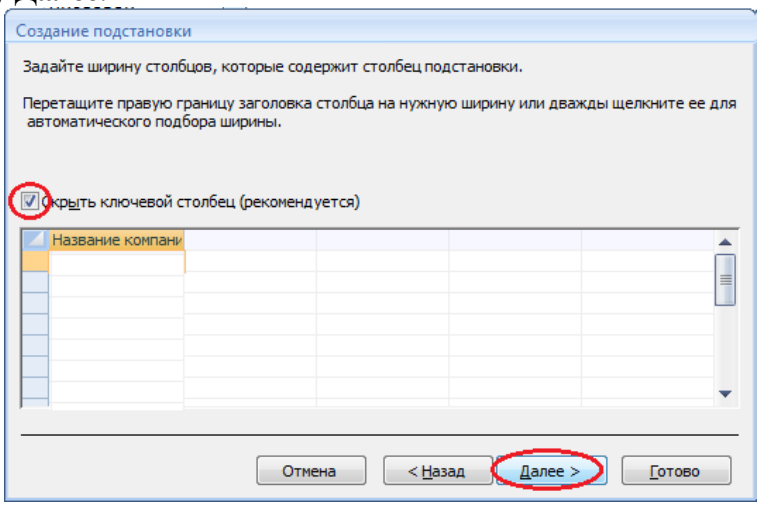




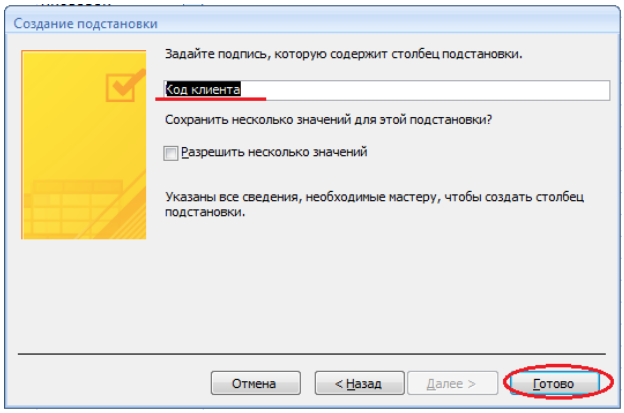
Выберите порядок сортировки списка по полю **Название компании** и нажмите кнопку **Далее**.



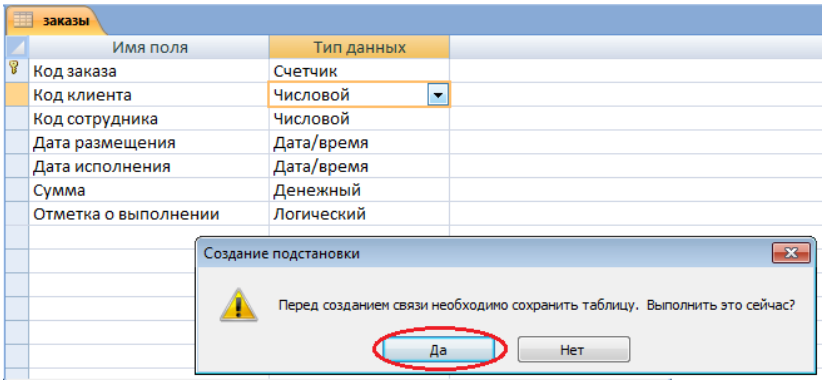
В следующем диалоговом окне задайте необходимую ширину столбцов раскрывающегося списка, установите флажок **Скрыть ключевой столбец** и нажмите кнопку **Далее**.



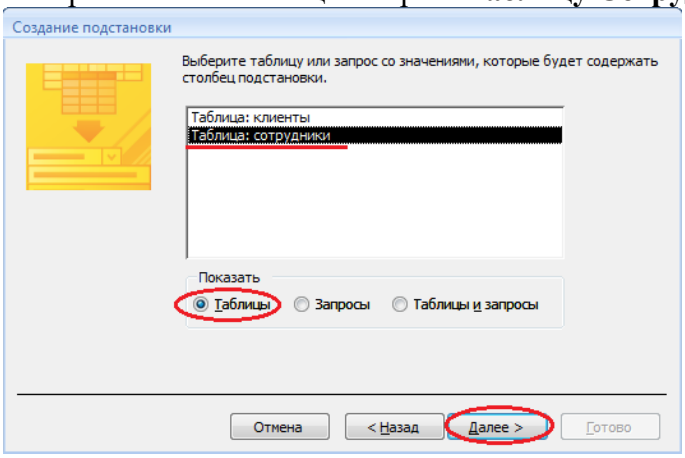
На последнем шаге **Мастера подстановок** замените при необходимости надпись для поля подстановок и щелкните на кнопке **Готово**.



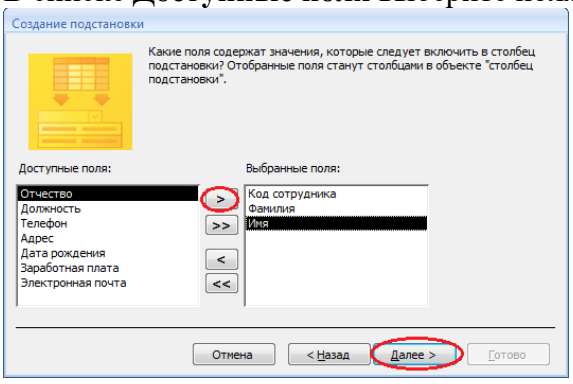
Сохраните полученный результат.



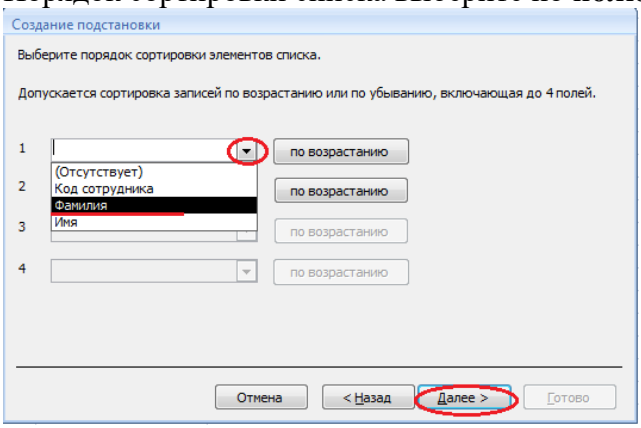
6. Аналогичным образом создайте раскрывающийся список для поля **Код сотрудника**. Теперь в списке таблиц выберите **таблицу Сотрудники**



В списке **Доступные поля** выберите поля **Код сотрудника, Фамилия, Имя**.



Порядок сортировки списка выберите по **полю Фамилия**.



Все остальные действия проводятся аналогично пункту 6.

7. Создайте связей между таблицами.

Существует несколько типов отношений между таблицами:

- *при отношении «один-к-одному»* каждой записи ключевого поля в первой таблице соответствует только одна запись в связанном поле другой таблицы, и наоборот. Отношения такого типа используются не очень часто. Иногда их можно использовать для разделения таблиц, содержащих много полей, для отделения части таблицы по соображениям безопасности;

- *при отношении «один-к-многим»* каждой записи в первой таблице соответствует несколько записей во второй, но запись во второй таблице не может иметь более одной связанной записи в первой таблице;

- *при отношении «многие-к-многим»* одной записи в первой таблице могут соответствовать несколько записей во второй таблице, а одной записи во второй таблице могут соответствовать несколько записей в первой.

Закройте все открытые таблицы, так как создавать или изменять связи между открытыми таблицами нельзя.

Выполните команду вкладки Лента **Работа с базами данных** кнопка **Схема данных**.

Практическое занятие № 3

Тема: Проектирование инфологической модели данных

Процесс проектирования БД представляет собой последовательность переходов от неформального словесного описания информационной структуры предметной области к формализованному описанию объектов предметной области в терминах некоторой модели. Можно выделить следующие *этапы проектирования.*

1.Системный анализ и словесное описание информационных объектов предметной области.

2.Проектирование инфологической модели предметной области – частично формализованное описание объектов предметной области в терминах некой инфологической, например, ER-модели.

3.Даталогическое (или логическое) проектирование БД, то есть описание БД в терминах принятой даталогической модели.

4.Физическое проектирование БД, то есть выбор способа размещения БД на внешних носителях.

Системный анализ должен заканчиваться подробным описанием информации об объектах, формулировкой задач с кратким описанием алгоритмов их решения, описанием входных и выходных документов.

Инфологическая модель должна выражать информацию в виде, не зависящем от используемой системы управления базами данных (далее – СУБД). Обычно эта модель отражает описание объектов, их свойства и взаимосвязи в виде схем. В настоящий момент наиболее широкое распространение получила модель Чена (Chen), которая называется «сущность-связь» или ER-модель (Entity Relationship). В ее основе лежат следующие базовые понятия.

Сущность – это класс однотипных объектов. Сущность имеет уникальное имя. Объект имеет свой набор *атрибутов* – свойств объекта.

25

Атрибут, однозначно идентифицирующий конкретный экземпляр сущности, называется ключевым.

Между сущностями могут быть установлены связи. По множественности связи делятся на три типа: *один-к-одному* (один экземпляр одной сущности связан только с одним экземпляром другой сущности), *один- ко-многим* (один экземпляр одной сущности связан с несколькими экземплярами другой сущности), *многие-ко-многим* (один экземпляр одной сущности связан с несколькими экземплярами другой сущности и наоборот). Связь любого типа может быть обязательной, если в данной связи должен участвовать каждый экземпляр, и необязательной. Связь может быть обязательной с одной стороны и необязательной с другой.

Для реализации проекта в конкретной СУБД следует разработать даталогическую модель. На настоящий момент для этой цели используется *реляционная модель*

данных. Основной структурой данных в модели является отношение, именно поэтому модель получила название реляционной (от англ. «relation» - «отношение»).

Существует *алгоритм* преобразования ER-модели в реляционную модель данных:

1.Каждой сущности ставится в соответствие отношение реляционной модели данных.

2.Каждый атрибут сущности становится атрибутом соответствующего отношения, задается тип данных и обязательность или необязательность данного атрибута.

3.Первичный ключ сущности становится ключевым полем соответствующего отношения.

4.В каждое отношение, соответствующее подчиненной сущности, добавляется атрибут основной сущности, и этот атрибут становится *внешним ключом.*

5.Для определения необязательного типа связи у атрибута, соответствующего внешнему ключу, устанавливается необязательность данного атрибута. При обязательном типе связи устанавливается его обязательность.

6.Если в ER-модели присутствуют связи «многие-ко-многим», то для перехода к реляционной модели данных (где такие связи не поддерживаются) вводится дополнительное связующее отношение. Оно связано с каждым исходным связью «один-ко-многим», а его атрибутами служат первичные ключи связываемых отношений.

В результате выполнения даталогического проектирования должна быть разработана схема БД, то есть совокупность отношений, которые моделируют объекты БД и связи между ними.

Разработать БД **СУД** согласно следующему описанию предметной области.

Описание предметной области. В каждом районе города имеется свой *районный суд.* В каждом суде имеются *судьи*, которые занимаются

26

рассмотрением гражданских и уголовных *дел и помощники судей*, которые занимаются судебным делопроизводством. Каждый судья может иметь несколько помощников, но каждый помощник может работать только с одним судьей.

При разработке БД требуется отразить следующую информацию:

для дел - номер, название, дату открытия, дату закрытия, количество томов, кто рассматривает дело;

у судей - код, фамилия, имя, отчество, место работы (районный суд), коллегия;

для районных судов – код, район, адрес; для помощников судей – табельный номер, фамилия, контактный

телефон, чьи поручения исполняет (код судьи).

1. Разработать ER-модель (модель «сущность-связь»).

Согласно описанию предметной области можно выделить следующие сущности: ***районные суды, судьи, дела, помощники судей***.

При разработке модели «сущность – связь» каждая сущность изображается в виде прямоугольника, в верхней части которого отражается имя сущности, а в нижней – атрибуты данной сущности, ключевой атрибут подчеркнут.

В каждом районном суде могут работать несколько судей, но каждый судья может работать только в одном районном суде. Таким образом, между сущностями ***районные суды*** и ***судьи*** устанавливается связь один ко многим.

Каждый судья может вести несколько дел, но каждое дело ведет только один судья. Таким образом, между сущностями ***судьи*** и ***дела*** устанавливается связь один ко многим.

Каждый судья может иметь несколько помощников, но каждый помощник может работать только с одним судьей. Таким образом, между сущностями ***судьи*** и ***помощники судей*** устанавливается связь один ко многим.

Тогда ER-модель будет выглядеть следующим образом (рис. 1):

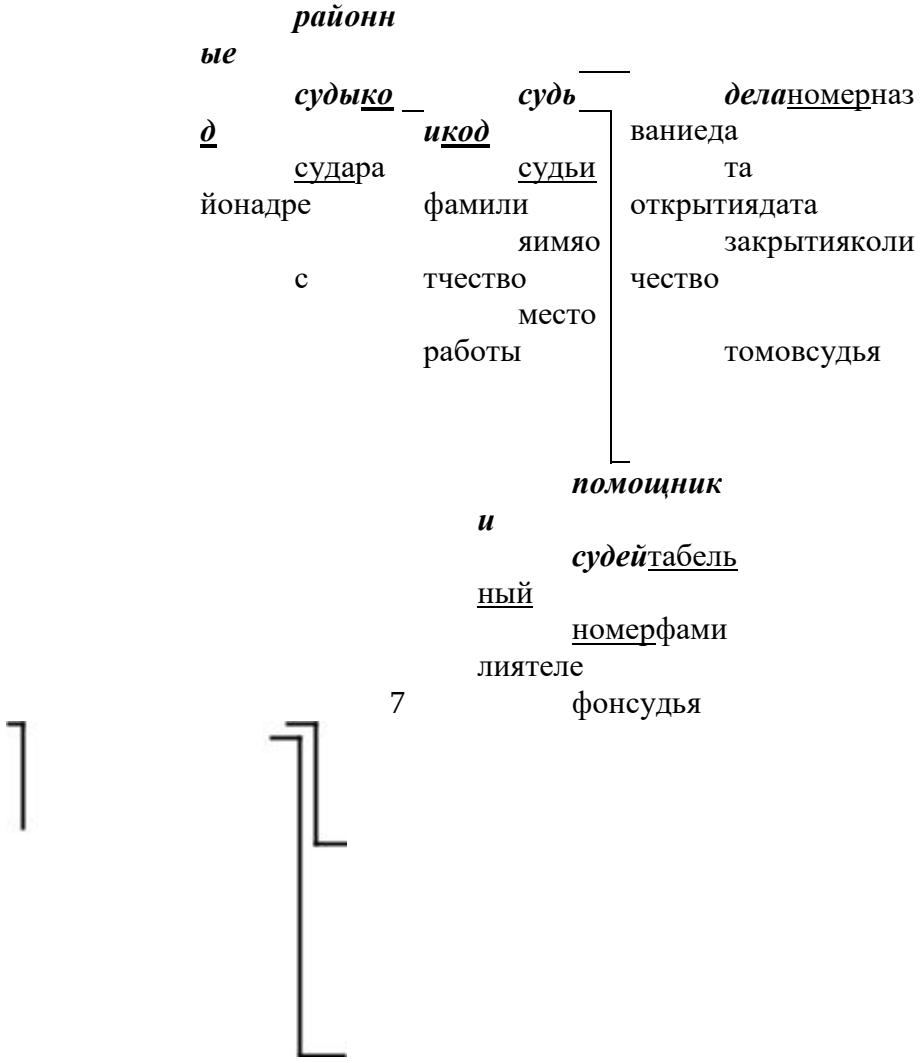


Рис. 1 ER-модедь БД «Суд»

2.Согласно алгоритму преобразования ER модели в реляционную, разработать даталогическую модель БД. Согласно алгоритму преобразования ERмодели в реляционную модель данных каждой сущности будет соответствовать одноименное отношение с соответствующими атрибутами. Какие поля будут полями внешнего ключа? Зачем они нужны? Атрибуты между которыми устанавливается связь должны иметь одинаковый тип и свойства. На рис. 2 показана реляционная модель «Суд», с указанием типов данных.

3.Реализовать разработанную информационную модель в системе управления базами данных (СУБД) Access.

Формирование базы данных (далее - БД) в Access состоит из ряда последовательных этапов. Первый этап этого процесса – *создание таблиц*. Таблицы в Access являются теми первичными, исходными файлами, на основе которых в дальнейшем строится все здание БД. Каждой сущности модели соответствует своя таблица. Имя таблицы совпадает с именем сущности.

Данные в таблице организованы в столбцы (называемые *полями*) и строки (*записи*). Каждому атрибуту сущности соответствует поле в таблице.

райо ***судьи***код

нные судьи

суды (числ.)ф
код суда амилия
(числ
.)район (текст.)и
(текс мя
т.)адрес
(текс (текст.)о
т.) тчество
(текст.)м
есто
работы
(числ.)

1 1
∞
∞

деланомер (числ.)название (текст.)дата открытия (дата/время)дата закрытия (дата/время)количество томов (числ.)судья (числ.)

помощники судей*табельный номер (числ.)фамилия (текст.)телефон (текст.)судья (числ.)*

Рис. 2 Реляционная модель «Суд».

Наиболее детальным и основательным методом формирования таблиц является режим конструктора. В режиме конструктора задаются имена полей и типы данных. В зависимости от характера данных необходимо задать свойства полей.

28

Каждая таблица должна содержать одно или несколько полей, однозначно идентифицирующих каждую запись в таблице. Такое поле называется *ключевым полем* таблицы. Если поле содержит уникальные значения, такие как коды или инвентарные номера, то это поле можно определить как ключевое. Ключевой атрибут сущности становится ключевым полем таблицы.

3.1. Создать новую базу данных **Суд**.

Запустить программу Access 2002 и создать новую базу данных. Для этого воспользоваться пиктограммой на рабочем столе, либо выполнить **Пуск**

– Программы – Microsoft Access. В диалоговом окне Создание файла

(справа) выбрать пункт **Новая база данных**. Присвоить имя **Суд** (в папке своей группы). Завершить создание БД. В результате будет создан файл новой БД. На экране будет отображено **Окно базы данных**.

3.2.Создать таблицу ***районные суды*** в режиме конструктора.

Для этого в **Окне базы данных** выбрать вкладку **Таблицы**. Нажать кнопку **Создать**. В диалоговом окне **Новая таблица** выбрать **Конструктор**, нажать кнопку **ОК.**

При конструировании таблицы необходимо задать имена полей и тип данных. **Имя поля** – это заголовки столбцов таблицы. В разделе **Тип данных** можно задать, какие данные и в каком формате будут введены в таблицу (числовой, текстовый, денежный и т.п.). **Описание** поля является необязательным параметром при конструировании таблицы.

Впервой строке в разделе **Имя поля** набрать ***код суда***. Мышкой переключиться в раздел **Тип данных**. Справа от указателя мыши появится стрелка раскрывающегося списка. Развернуть список и выбрать из него тип *Числовой*. Переключиться на следующую строку.

Вразделе **Имя поля** набрать ***район***, в разделе **Тип данных** выбрать *Текстовый*. В нижней половине окна конструктора расположен раздел **Свойства поля**. Его вид зависит от выбранного типа данных (текстовый, числовой, денежный и т.д.). Заполнить свойства для

поля ***район***. В разделе **Размер** поля указать 25 (символов), в разделе **Обязательное** поле указать *Да,*

вразделе Пустые строки – *нет*.

Вследующей строке раздела **Имя поля** ввести ***адрес***, в разделе тип данных выбрать *Текстовый*. В разделе **Размер** поля указать 50 (символов), в разделе **Обязательное** поле указать *Нет*, в разделе **Пустые строки** – *да.*

Задание ключевого поля. Каждая таблица должна содержать одно или несколько полей, однозначно идентифицирующих каждую запись в таблице. Такое поле называется ключевым. Ключ служит для установления связей между таблицами и для предотвращения ввода повторяющихся данных. В данной таблице в качестве ключевого следует использовать поле ***код суда***. Для того чтобы обозначить поле как ключевое, необходимо в режиме

конструктора выделить поле и нажать кнопку  на панели инструментов.

29

Сохранение таблицы. Все объекты в Access, в том числе таблицы, сохраняются стандартным для Windows способом. Сохранить таблицу под именем ***Районные суды***. Закрыть таблицу.

3.3.Аналогично в режиме конструктора создать таблицы ***Судьи, Дела*** и ***Помощники судей***. Тип данных выбирать в соответствии с разработанной реляционной моделью (см. рис. 2). ***Внимание!!!*** **Атрибуты между**

которыми устанавливается связь должны иметь одинаковый тип и свойства.

3.4. Создать связи между полями таблиц.

Когда между двумя таблицами устанавливается *связь*, это означает, что

величины из одной таблицы ставятся в соответствие величинам другой таблицы. Обычно связывают ключевое поле родительской таблицы с

соответствующим ему полем в *дочерней таблице* (внешним ключом). Установить связи между таблицами согласно разработанной

реляционной модели (рис. 2). Перед созданием связей закрыть все таблицы! В меню **Сервис** выбрать **Схема данных**. В диалоговом окне **Добавление таблицы** отметить все четыре таблицы и нажать кнопку **Добавить**. Закрыть окно **Добавление таблицы.**

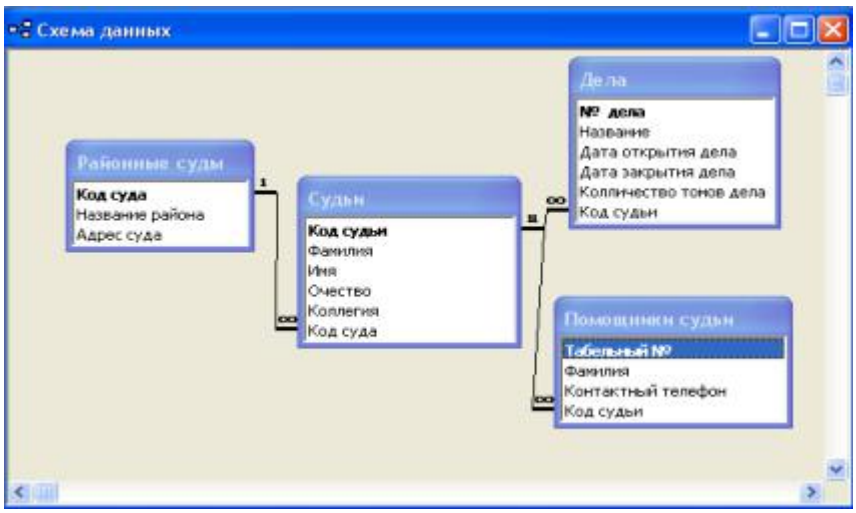
В окне **Схема данных** установить связь между соответствующими полями таблиц ***Судьи*** и ***Дела***. Для этого щелкнуть мышью по полю ***код судьи*** таблицы ***Судьи*** и, удерживая ее, перетащить указатель мыши на поле ***Судья*** таблицы ***Дела***. На экране появится диалоговое окно **Связи**. В левой части этого окна указано связываемое поле родительской таблицы, а в правой – дочерней. Поставить флажки на пунктах **Обеспечение целостности данных**

и Каскадное обновление, нажать Создать. В окне Схемы данных появится линия связи между таблицами *Судьи* и *Дела*. Следует обратить внимание, что

на одном конце линии связи стоит знак **1**, а на другом ∞. Этот тип связи называется *«один-ко-многим».* Связь с отношением *«один-ко-многим»*

является наиболее часто используемым типом связи между таблицами. В такой связи *каждой записи* в таблице *Судьи* могут соответствовать несколько записей в таблице *Дела*, а запись в таблице *Дела* не может иметь *более одной* соответствующей ей записи в таблице *Судьи.*

30



Аналогично установить остальные связи. В результате схема данных должна иметь следующий вид:

Рис. 3. Схема данных базы данных «Суд».

3.4.Заполнить таблицы произвольными данными в режиме Таблица. Сначала заполнить таблицу ***Районные суды*** (3 записи), затем – ***Судьи*** (5 записей), а далее – ***Дела*** (7 записей) и ***Помощники судьи*** (8 записей). Для этого открыть нужную таблицу в режиме Таблица и ввести данные.

Внимание!!! **Дочерняя таблица в поле внешнего ключа может содержать только те значения, которые содержатся в ключевом поле родительской таблицы. Например, в поле *место работы* таблицы *Судьи* могут содержаться только те значения, которые ранее были внесены в поле *код суда***

таблицы районные суды.

Практическое занятие 8. Создание запросов на выборку

Запросы – основной способ получения сведений из базы данных. Запросы являются основным средством просмотра, изменения и анализа информации, которая содержится в одной или в нескольких таблицах БД. С помощью запроса можно отобрать определенную информацию и рассортировать ее по значениям полей.

Существуют запросы на *выборку* и запросы на *изменение.*

Запросы на выборку позволяют извлечь информацию, рассчитать показатели и создать перекрестные ссылки, но не изменяют данные в таблицах. Запросы на выборку можно разделить на **простые запросы** (вывод какой-либо таблицы полностью), **запросы с использованием групповых операций** (например, суммирование значений поля, поиск минимального значения поля, подсчет количества записей) и **запросы с условиями** (с критериями отбора).

Создание запросов на выборку <u>Создание простых запросов.</u>

1. Вывести фамилии всех судей с указанием района, в котором

работает судья.

Для этого перейти на вкладку **Запросы**. Выбрать опцию **Создание запроса в режиме конструктора**. В окне **Добавление таблицы** добавить таблицы **Судья** и **Районный суд**. Закрыть окно Добавление таблицы. В окне конструктора запросов включить в бланк запроса поле **Фамилия** из таблицы **Судья** и поле **Район** из таблицы **Районный суд**. Запустить запрос, щелкнув

по кнопке  операционного меню. Сохранить запрос под именем

Запрос1.

2.Вывести фамилии всех помощников судьи с указанием фамилии судьи, с которым работает помощник.

3.Вывести фамилии и контактные телефоны всех помощников

судьи с указанием района, в котором работает помощник.

Запросы с использованием групповых операций предназначены для первичной обработки полученной информации – ее суммирования, осреднения и т.п. Для того, чтобы опция групповые операции стала доступна,

необходимо щелкнуть по кнопке  на панели инструментов. Тогда в бланке запроса появится строка **Групповые операции**.

4. Сосчитать количество дел, рассматриваемых всеми судьями.

Для этого в бланк запроса включить поле **№ дела** из таблицы **Дела**. В строке **Групповые операции** развернуть список функций по стрелке прокрутки и выбрать функцию **Count**, которая количество записей в поле.

5.Рассчитать среднее количество томов в делах, используя функцию Avg, вычисляющую среднее арифметическое значений поля.

6.Показать последнюю запись таблицы Дела (в запрос включить все поля таблицы), используя функцию Last.

7.Определить минимальное количество томов в деле.

8.Определить максимальное количество томов в деле.

9.Определить суммарное количество томов во всех делах. 10.Определить дату о открытия дела, открытого последним. 11.Определить дату закрытия дела, закрытого раньше других.

12.Показать, сколько дел ведет каждый судья. Для этого в бланк запроса ввести поле Фамилия из таблицы Судья и поле № дела из таблицы Дела. Добавить опцию Групповые операции. В поле № дела выбрать функцию Count.

13.Показать, сколько помощников у каждого судьи.

Запросы с вычисляемыми полями.

14. Вычислить возраст и стаж работы каждого судьи.

Для этого добавить в запрос таблицу **Судьи**. В первое поле бланка запроса включить поле **Фамилия**. Во втором поле бланка запроса вычислить возраст судьи. Для этого щелкнуть правой клавишей мыши и открыть окно

Построитель выражений (или щелкнуть по кнопке  операционного меню. Ввести формулу для вычисляемого поля:

Возраст: Year(Now())-Year([Судьи]![Дата рождения]) ВНИМАНИЕ!!! Встроенные функции вводить в формулу, открыв в

окне обзора папку Встроенные функции. Аргумент [Судьи]![Дата рождения] ввести двойным щелчком по имени соответствующего поля (Дата рождения), открыв в окне обзора соответствующую таблицу (Судьи) из папки Таблицы.

Аналогично в третьем поле запроса вычислить стаж судей.

15.Для всех дел показать год, месяц и день открытия и закрытия

суказанием номера дела и его названия. Для этого с помощью Построителя выражений ввести нужные формулы, используя функции Year, Month и

Day.

Например,

Месяц открытия дела: Month([Дела]![Дата открытия дела])

16.Вычислить средний возраст судей. Для этого в создаваемый запрос добавить Запрос14. Использовать Групповые операции.

17.Вычислить средний стаж работы судей.

Запросы с условиями отбора.

Условия отбора – это ограничение, которое пользователь накладывает на запрос для определения записей, удовлетворяющих некоторым требованиям. Чтобы установить условие отбора для поля в бланке запроса, необходимо ввести в строке **Условие отбора** для данного поля необходимо ввести выражение. При написании выражения необходимо учитывать несколько общих правил. При работе с текстовыми данными их заключают в кавычки

(=«Иванов»). При работе с датами используют символ # (#12.03.1998#). Числа набирают без каких-либо дополнительных символов. Пустые позиции обозначают ключевым словом Null.

18.Показать фамилии судей, работающих в Советском районе.

Для этого в бланк запроса включить поле **Фамилия** таблицы **Судьи** и поле **Район** таблицы **Районный суд**. Для поля Район ввести Условие отбора =

«Советский»

19.Показать полную информацию о делах, которые ведет судья Иванов (фамилию судьи можно взять другую).

20.Показать информацию о помощниках судьи, фамилии которых по алфавиту идут после фамилии Петров. (>«Петров»).

21.Показать полную информацию о делах, открытых после 20.03.2005, с указанием судьи, ведущего дело и района.

33

22.Показать информацию о делах, открытых между 18.05.2006 и 26.09.2006 (Between #18.05.2005# And #26.09.2006#).

23.Показать информацию о делах, в которых количество томов меньше 2.

24.Показать информацию о делах, в которых количество томов от 2 до 4.

25.Показать фамилии судей, возраст которых превышает 50 лет.

Для этого при создании запроса использовать **Запрос14.**

26.Показать фамилии судей, стаж работы которых от 3 до 5 лет.

27.Показать сведения о помощниках судей, не имеющих контактных телефонов.

***28.Показать сведения о делах, открытых в 2006 году (использовать* Запрос15).**

Для отбора записей, начинающихся или заканчивающихся определенными наборами символов, используются подстановочные знаки: * - заменяет любое количество любых символов, ? – заменяет один любой символ.

29.Показать сведения о судьях, фамилии которых начинаются с буквы А (=«А*»).

30.Показать сведения о помощниках судьи, имеющих семизначные телефоны.

Для одного поля или для нескольких полей можно ввести *дополнительные условия отбора*. Если выражения вводятся в несколько ячеек **Условие отбора**, то они автоматически комбинируются с помощью операторов **And (И)** или **Or (ИЛИ)**. Если выражения находятся в одной строке, то используется оператор **And**. Если же выражения находятся в разных строках бланка запроса, то используется оператор Or.

31.Показать сведения о делах, открытых в мае 2006 г.

(использовать **Запрос15**).

32.Показать сведения о делах, закрытых в сентябре или марте.

33.Показать сведения о судьях, работающих в Советском или Вахитовском районах.

34.Показать сведения о помощниках, имееющих тедефоны (Not Nool), фамилии которых заканчиваются на букву «в».

35.Показать информацию о делах, которые ведет судья Иванов, открытых после 20.07.2006.

36.Показать номер и название дела с максимальным количеством томов, а также фамилию судьи, ведущего дело. Для этого включить в бланк запроса поле Фамилия таблицы Судьи и поля № дела, Название, Количество томов таблицы Дела, для поля Количество томов использовать

34

условие **=Запрос8** (использовать **Построитель выражений**, для ссылки на запрос8 выбрать его в папке **Запросы**).

37.Показать полную информацию о деле, закрытом раньше других

(В условии использовать ссылку на **Запрос11**).

38.Показать фамилию судьи и название района, который вел дело, открытое последним.

39.Показать сведения о судьях, стаж работы которых больше среднего стажа.

40.Показать фамилии судей стаж работы которых меньше среднего стажа, а возраст больше среднего возраста судей.

Тема 5. Основы компьютерного делопроизводства в юридической деятельности в среде текстового процессора Word

Практическое занятие 9. Создание шаблона документа и документа на основе шаблона в текстовом процессоре Word.

Задание №1. Создать бланк для писем, факсов с продольным расположением реквизитов

1.Создать новый шаблон, используя команду Создать меню Файл (в диалоговом окне Создать нужно отметить кнопку Шаблон). Образец создаваемого бланка дан в Приложении А

2.Задать формат бланка, используя команду Параметры страницы меню Файл. Размер бумаги – 210х297 мм. Ориентация – книжная, поля: верхнее – 2 см, нижнее – 2 см, левое – 2,5 см, правое – 1 см.

3.Заполнить шаблон, т.е.

а) вставить в шаблон реквизит «эмблема» (команда Рисунок меню Вставка);

б) вставить в шаблон реквизит «наименование предприятия»; в) вставить вписываемые реквизиты «дата» и «индекс (№) документа»,

которые стоят на одной строке. Для этого:

· там, где должна быть горизонтальная черта для даты, поставить знак табуляции, нажав на клавишу [Tab] (на экране знак табуляции отобразится стрелкой вправо, которую можно увидеть, если включен режим просмотра

спецсимволов, т.е. кнопка ¶ на панели инструментов «Стандартная» нажата);

·после знака табуляции поставить пробел;

·поставить символ номера;

·после символа номера поставить один пробел, а затем знак табуляции для второй горизонтальной черты;

·обозначить горизонтальные линии и задать их длину. Для этого выделить мышью первый знак табуляции, задать символу табуляции формат

подчеркнутого символа. Аналогично прочертить линию под вторым знаком

35

табуляции. Длина линий задается путем установки маркера табуляции на линейке. В левой части линейки находится кнопка, с помощью которой можно переключать способ выравнивания относительно знака табуляции.

Можно задать четыре способа выравнивания: – левое выравнивание; –

правое выравнивание; – центральное выравнивание; – выравнивание по десятичной точке. При этом если текст не содержит числа с десятичной точкой, то он помещается просто слева.

· для первого знака табуляции будем использовать левое выравнивание и правое – для второго. Чтобы установить маркер табуляции на линейке, необходимо задать абзацу выравнивание по левому краю, указать курсором мыши позицию на нижней стороне линейки, где будет устанавливаться маркер табуляции и щелкнуть. Если длина линии окажется неудовлетворительной, нужно переместить маркер на новое место с помощью мыши. Чтобы удалить маркер с линейки, нужно мышью стащить его с линейки вниз;

г) вставить реквизит «ссылка на индекс и дату входящего документа». Реквизит «ссылка на индекс и дату входящего документа» – это

реквизит, который проставляется при создании письма-ответа, он может отсутствовать, например, во внутренних документах, поэтому для этого реквизита в шаблоне бланка предприятия необходимо указать только место расположения. Тем самым в шаблон бланка будет вставлена подсказка. Символам подсказки надо присвоить формат скрытого текста, тогда она не будет отображаться при печати. Если при составлении документа на основе этого шаблона нужен реквизит «ссылка на индекс и дату входящего документа», то скрытость надо снять (в документе, а не в шаблоне).

Данный абзац должен быть выровнен по левому краю.

Чтобы данный реквизит лучше читался, нужно дать для его абзаца интервал сверху 3-6 пунктов, используя команду Абзац меню Формат.

д) вставить в шаблон реквизит «почтовый адрес, номер телефона, номер факса» в виде нижнего колонтитула (команда Колонтитулы меню Вид).

4.Выполнить команду Сохранить меню Файл для сохранения созданного шаблона на диске. При правильном создании шаблона при его сохранении автоматически предлагается папка Шаблоны для его хранения, а

впараметре Тип файла автоматически устанавливается тип Шаблоны документа (расширение – .dot). Закрыть файл.

Практическое занятие № 4-5

Тема: Проектирование структуры базы данных

Цель: Закрепить умения и навыки создания баз данных в прикладной программе MS Access, выбора базы данных, сформировать первичные умения и навыки ее сохранения и копирования

Теоретическая часть:

Использование баз данных является одним из приоритетных направлений развития прикладного программного обеспечения.

Как использовать управляющий элемент Open File

Управляющий элемент Open File открывает диалоговое окно, в котором можно просматривать и выбирать файлы. Следует заметить, что управляющий элемент Open File не открывает файл. Файл можно лишь выбрать, а открываться он будет программой, которая встроена в приложение. По существу, у диалогового окна Open File нет своего интерфейса, поэтому оно появляется не на форме, а под ней. Для того чтобы пользователь мог просматривать файлы, надо управлять диалоговым окном открытия файла с помощью его свойств и методов.

Управляющий элемент Save File

Управляющий элемент диалогового окна Save File очень похож на Open File, но он предназначен для того, чтобы просмотреть папки и указать файл, который надо сохранить. Опять же, следует заметить, что на самом деле Save File не сохраняет файл. Управляющий элемент лишь возвращает имя указанного файла.

Свойство OverwritePrompt является характерным лишь для диалогового окна Save File. Если оно (это свойство) имеет значение TRUE, ТО будет запрашиваться подтверждение при записи файла с уже существующим именем. Рекомендуется установить значением этого свойства TRUE. Если пользователь попытается заменить существующий файл, то будет запрошено подтверждение.

Visual Basic .NET включает в себя объект, который называется System.IO. Если правильно использовать различные свойства, объектные свойства и методы объекта System, то над файловой системой можно делать различные операции. Объекты System.I0. Directory и System.io.File обеспечивают расширенное управление файловой системой.

Как скопировать файл

Перед тем как выполнять над файлом какую-либо операцию (удаление, копирование), надо удостовериться, что он существует. Например, если пользователь в текстовом поле печатает полный путь к файлу, а не указывает его местонахождение через кнопку Source, то он может ошибиться. Если программа обратится к несуществующему файлу, то возникнет сбой,

который вам совсем нежелателен. Поскольку над файлами, которые укажет пользователь, надо выполнять различные стандартные операции, то необходима основная функция, с помощью которой можно определить, существует файл или нет. Для этого функция использует метод Exists () объекта System.IO.File.

Метод Exists () проверяет строку, которая содержит имя файла и путь к нему. Если файл существует, то возвращается TRUE , иначе — FALSE. Обратите внимание, если файл не найден, то значение FALSE не надо возвращать явно. Когда создаются переменные типа Boolean, то им сразу присваивается значение FALSE.

Копирование файлов – это стандартная операция. Можно, например, создать программу, которая будет сохранять важные файлы, копируя их в другое место. В общем, это достаточно безопасный процесс, если не пытаться копировать несуществующий файл. Файлы копируются с помощью метода Copy (), объекта System.IO.File.

У метода Copy () есть два аргумента. Первый – это имя файла, который надо перекопировать. Во втором указывают место, назначение и новое имя файла.

Практическая часть:

Создайте в приложении MS ACCESS базу данных, состоящую из двух таблиц: «Физические лица» и «Сотрудники».

Таблица «Физические лица» содержит следующие поля:

«Код» – используется в качестве первичного ключа, тип Integer (Счетчик), индексированное поле;

«Фамилия», «Имя», «Отчество», «Телефон», «Индекс», «Страна», «Город», «Адрес» – текстовые поля;

«Дата рождения» – типа Дата/Время;

«Пол» – поле типа Текстовый.

Таблица «Сотрудники» содержит следующие поля:

«Код» – используется в качестве первичного ключа, тип Integer, индексированное поле;

«Должность» – текстовое поле;

«Разряд», «Зарплата», «Рейтинг» – числовые поля;

Сохраните полученный результат в файле firma.mdb или firma.accdb.

Разработаем форму для просмотра и редактирования информации, содержащейся в таблице «Физические лица». Научимся использовать управляющие элементы Open File и Save File для просмотра файлов.

Последовательность действий для создания простых форм.

1. Запустите платформу Microsoft Visual Studio.

Диалоговое окно Open File

2. Для создания нового приложения выполните команду Файл – Создать – Проект (File Þ New Project). Выберите Приложение Windows Forms. Назовите проект Manipulating Files. Измените название формы, которая выбрана по умолчанию, на fclsManipulatingFiles, текст в форме — на Manipulating Files. Добавьте в форму новое текстовое поле и настройте его свойства, как показано в данной таблице.

войство	*Значение*
Name	txtSource
Location	95,8
Size	184,20
Text	*(оставить пустым)*

3. Добавьте в проект диалоговое окно открытия файла. Для этого сделайте двойной щелчок по соответствующему элементу OpenFileDialog на панели инструментов ToolBox. По существу, у диалогового окна Open File нет своего интерфейса, поэтому оно появляется не на форме, а под ней. Для того чтобы пользователь мог просматривать файлы, надо управлять диалоговым окном открытия файла с помощью его свойств и методов.

Теперь в форму надо добавить кнопку, с помощью которой можно выбрать файл. При выборе файла его имя будет отображено в текстовом поле. Настройте свойства кнопки как указано в таблице:

Свойство	*Значение*
Name	btnOpenFile
Location	8,8
Size	80,23
Text	Source:

Теперь сделайте двойной щелочек на кнопке и добавьте в событие Click такой текст:

```
OpenFileDialog1.InitialDirectory = "C:\"
OpenFileDialog1.Title = "Выбери файл"
```

Наконец, надо отобразить диалоговое окно Open File и когда в нем будет выбран файл, выполнить соответствующее действие. Метод ShowDialog диалогового окна open File работает так же, как и метод форм с тем же названием. Он возвращает результат, в котором указывается, что было выбрано пользователем в диалоговом окне.

```
If OpenFileDialog1.ShowDialog() <> Windows.Forms.DialogResult.Cancel Then
    txtSource.Text = OpenFileDialog1.FileName
Else
    txtSource.Text = ""
End If
```

Эта программа помещает указанные названия файлов в текстовое поле txtSource.

Добавьте кнопку **Cancel**.

Запустите проект нажатием F5 и щелкните на кнопке. Найдите созданную вами базу данных, выберите ее.

Диалоговое окно Save File

4. Создайте новое текстовое поле на форме с такими свойствами:

Свойство	*Значение*
Name	txtDestination
Location	95,40
Size	184,20
Text	(оставить пустым)

Теперь надо создать кнопку, при нажатии которой пользователь сможет выбрать файл для сохранения. Добавьте в форму новую кнопку и установите для нее такие свойства:

Свойство	*Значение*
Name	btnSaveFile
Location	8,40
Size	80,23
Text	Destination:

Добавьте диалоговое окно **Save File**.

Затем сделайте двойной щелчок по созданной кнопке btnSaveFile и добавьте в ее событие Click такой текст:

```
SaveFileDialog1.Title = "Specify Destination Filename"
SaveFileDialog1.OverwritePrompt = True
```

Наконец, надо написать последнюю часть программы, которая помещает указанное имя файла в текстовое поле txtDestination:

```
If SaveFileDialog1.ShowDialog() <> Windows.Forms.DialogResult.Cancel Then
    txtDestination.Text = SaveFileDialog1.FileName
End If
```

Запустите проект, проверьте, как работает диалоговое окно. Сохраните созданную вами базу данных под именем firma_copy.mdb.

5. Скопировать созданную базу данных в другую папку

Для начала добавьте функцию для проверки существования файла в форму класса:

```
Public Class Form1
    Private Function SourceFileExists() As Boolean
        If Not (System.IO.File.Exists(txtSource.Text)) Then
            MsgBox("Такой файл не существует", MsgBoxStyle.Exclamation)
        Else
            SourceFileExists = True
        End If
    End Function
```

6. Теперь добавим в форму новую кнопку. Если щелкнуть по этой кнопке, то файл, указанный в текстовом поле Source будет скопирован под именем, которое указывается в текстовом поле Destination. Установите свойства этой кнопки, как показано в таблице:

Свойство	*Значение*
Name	btnCopyFile
Location	96, 80
Size	75, 23
Text	Copy

Добавьте такой текст:

```
If Not (SourceFileExists()) Then Exit Sub
System.IO.File.Copy(txtSource.Text, txtDestination.Text)
MsgBox("The file has been successfully copied")
```

7. Запустите проект нажатием клавиши <F5>. Для того чтобы протестировать его, сделайте следующее:

1. Щелкните на кнопке Source и выделите файл, содержащий созданную вами базу данных.

2. Для того чтобы появилось диалоговое окно Save File, щелкните на кнопке Destination. Укажите в текстовом поле File Name имя firma_copy.mdb и нажмите кнопку Save. Если будет задан вопрос, заменять ли файл, выберите No и измените имя файла. Не следует использовать имя существующего файла.

3. Для того чтобы скопировать файл, щелкните на кнопке Copy.

После того как появится окно сообщения, в котором подтверждается копирование файла, его нужно найти и открыть с помощью СУБД. Сохраните свою работу.

Контрольные вопросы

1. Для чего предназначен управляющий элемент Open File?
2. Открывает ли файл управляющий элемент Open File?
3. Где появляется управляющий элемент Open File, есть ли у него свой интерфейс?
4. Опишите свойство OverwritePromt диалогового окна Save File
5. Назначение метода Copy (), его аргументы
6. Что необходимо сделать перед тем, как выполнять с файлом операцию копирования?
7. Для чего предназначен метод Exists (), каково его значение?
8. Когда создаются переменные типа Boolean какое значение присваивается им по умолчанию?

Практическое занятие № 6-7

Тема: Проектирование базы данных с использованием CASE-средств

Цель занятия: Получить практический опыт проектирования базы данных с использованием CASE-средств.

Перечень оборудования и программного обеспечения

1. Персональный компьютер
2. Microsoft Office (Word, Visio, Access)
3. DBDesigner online

Краткие теоретические сведения

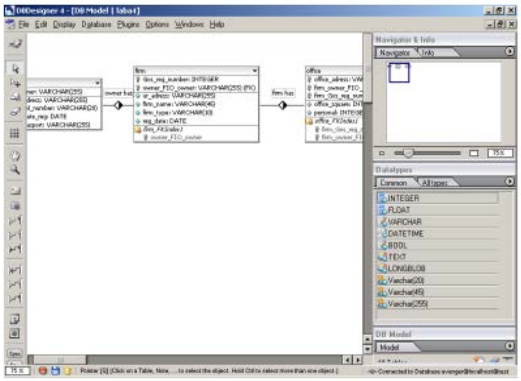
CASE-средства представляют собой программные средства, поддерживающие процессы создания и/или сопровождения информационных систем, такие как: анализ и

формулировка требований, проектирование баз данных и приложений, генерация кода, тестирование, обеспечение качества, управление конфигурацией и проектом. CASE-систему можно определить как набор CASE-средств, имеющих определенное функциональное предназначение и выполненных в рамках единого продукта.

DBDesigner – это свободно распространяемая CASE-система, предназначенная для проектирования, моделирования, создания и поддержки информационных систем. Программа может использоваться для Windows 2000/XP, LinuxKDE/Gnome и MySQL. DBDesigner позволяет:

- создавать модель проектируемой системы;
- преобразовывать модели системы в SQL-код, который можно использовать для создания базы данных с помощью DBDesigner или другого средства;
- проводить реинжиниринг – построение исходной модели программной системы путем исследования ее программных кодов. Эта функция очень удобна в случае, если необходимо разобраться уже существующей базе данных. Для проведения реинжиниринга следует выбрать в меню Database – ReversEngineering;
- создавать базу данных и автоматически вносить в нее изменения, используя соединение с сервером и синхронизацию;
- создавать SQL-запросы для внесения изменений и проведения операций над данными.

Пользовательский интерфейс программы:



Основные этапы проектирования базы данных:

1. Описание предметной области. Определение цели создания базы данных.

2. Определение сущностей предметной области (таблиц), которые должна содержать база данных.

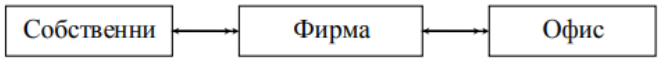
3. Определение атрибутов сущностей (необходимых в таблицах полей).

4. Построение инфологической модели. Определение связей между сущностями (таблицами).

Предметная область:

База данных содержит информацию о фирмах, зарегистрированных в России.

Инфологическая модель:



Собственник (**<u>ФИО собственника</u>**, Адрес проживания собственника, Номер телефона собственника);

Фирма (**<u>Государственный регистрационный № фирмы</u>**, Юридический адрес фирмы, Название фирмы, Тип фирмы);

Офис (**<u>Адрес офиса</u>**, Площадь офиса, Количество персонала офиса);

Датологическая модель:

Здесь составим саму реляционную модель проектируемой БД: раскроем все связи между сущностями как связи между ключами. Для этого добавим в одну из связываемых сущностей дополнительный атрибут – первичный ключ из другой сущности. Причем

добавление будем осуществлять в ту из этих сущностей, где не нарушится понятие ее первичного ключа.

Собственник	Фирма	Офис
ФИО собственника	**Гос. рег. № фирмы**	**Адрес офиса**
Адрес проживания собственника	Юридический адрес фирмы	Площадь офиса
Номер телефона собственника	Название фирмы	Количество персонала офиса
	Тип фирмы	*Гос. рег. № фирмы*
	ФИО собственника	

Собственник (**ФИО собственника**, Адрес проживания собственника, Номер телефона собственника);

Фирма (**Государственный регистрационный № фирмы**, Юридический адрес фирмы, Название фирмы, Тип фирмы, ФИО собственника);

Офис (**Адрес офиса**, Площадь офиса, Количество персонала офиса, Государственный регистрационный № фирмы);

Предметной областью разрабатываемой базы системы является информация об унитарных предприятиях, зарегистрированных в России.

Информация о предприятиях включает в себя информацию о собственнике, о самой фирме, об офисе. Необходимо обеспечить возможность внесения, изменения или удаления данных в базе и проведение различных поисковых операций: поиск по названию фирмы, поиск по ФИО предпринимателя.

1. Моделирование

Модель – это визуальное представление структуры данных. Модель может включать в себя следующие объекты: таблицы и отношения, которые используются обязательно, и дополнительные (например, изображения, записи) – для обеспечения лучшего "понимания" структуры модели.

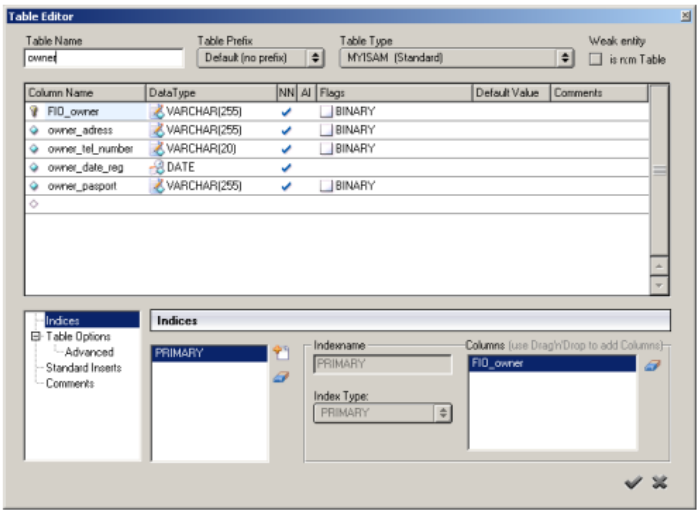
Для создания модели необходимо переключится в DesignMode, выбрав меню **Display - DesignMode.**

Пользовательский интерфейс делает создание модели базы данных очень легким. DBDesigner 4 поддерживает Multible Document Interface(MDI), который позволяет открывать неограниченное число моделей одновременно. При работе, вы можете переключаться между моделями, копируя команды и объекты, чтобы обмениваться ими между моделями.

1*) Создание таблиц*

В левой части холста находится панель инструментов. Необходимо нажать на этой панели кнопку и указать место на холсте, где будет располагаться новая сущность. Появится прямоугольное изображение пустой сущности. Чтобы задать атрибуты сущности, необходимо два раз щелкнуть на изображении сущности. В появившемся окне можно задать название сущности, а также атрибуты этой сущности.

В данной лабораторной работе область представляет собой информацию о фирмах, их владельцах, и о офисах, где располагаются эти фирмы. Для отображения личной информации о собственниках была создана сущность owner. Структура сущности owner показана на рисунке:

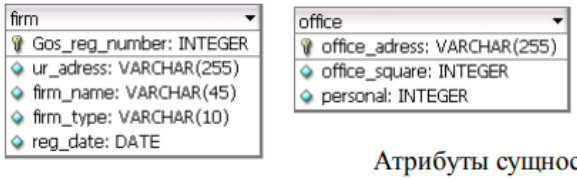


Для этой сущности введены следующие атрибуты:

- ФИО собственника (FIO_owner) (ключевой атрибут),
- Адрес проживания собственника (owner_adress),
- Телефонный номер собственника (owner_tel_number),
- Дата регистрации (owner_date_reg),
- Паспортные данные(owner_pasport)

Флаг в поле NN означает, что содержимое данного поля не может быть нулевым (NotNull). Флаг в поле AI означает, что значение данного поля в каждой следующей строке увеличивается на 1 (AutoIncrement). Иконка напротив имени атрибута означает, что этот атрибут является ключевым.

Были созданы еще две сущности: фирма (firm) и офис (office)



Атрибуты сущности **firm**:

Государственный регистрационный номер фирмы (Gos_reg_number) (ключевой атрибут),

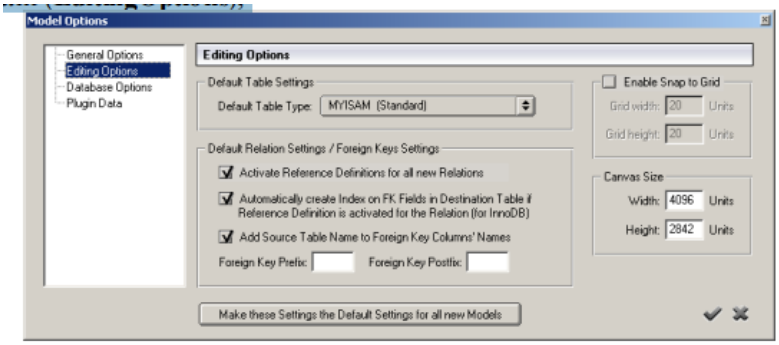
- Юридический адрес фирмы (ur_adress),
- Название фирмы (firm_name),
- Дата регистрации фирмы (reg_date)

Атрибуты сущности office:

- Адрес офиса (office_adress) (ключевой атрибут),
- Площадь офиса (office_square),
- Количество персонала (personal).

1) Формирование отношений

Связь между сущностями определяет связь между будущими таблицами. Для этого необходимо поставить флаг напротив после всех полей в разделе параметры отношений и внешних ключей (раздел DefaultRelationSettings / ForeignKeysSettings) во вкладке редактирования модели (EditingOptions),



Между сущностями имеются следующие связи:

Собственник –> владеет ->Фирмой

Собственник может владеть несколькими фирмами, каждая фирма может иметь только одного собственника. Собственник должен иметь хотя бы одну фирму, фирма обязательно должна принадлежать кому-то. Имеем связь 1:m, класс принадлежности O:O.

Фирма -> имеет ->Офис

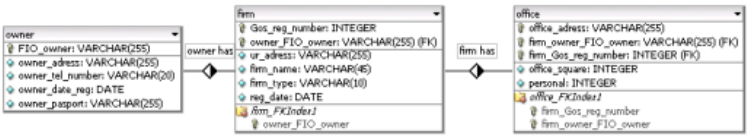
Фирма может иметь несколько офисов, один офис может иметь только одна фирма. Фирма может не иметь ни одного офиса, но офис обязательно должен принадлежать какой-то фирме. Имеем связь 1:m, класс принадлежности H:O.

В программе связи задаются следующим образом.

- Связь 1:1 задается с помощью кнопки .
- Связь 1:n задается с помощью кнопки .
- Связь n:m задается с помощью кнопки .

Задать связь между сущностями можно, нажав на соответствующую кнопку и указав связываемые таблицы. После нажатия кнопки связи, надо нажать на первую таблицу, участвующую в связи, затем на другую. Внешние ключи будут автоматически добавлены в сущности соответственно связи.

Результат связывания сущностей показан на рисунке:



Двойным щелчком по изображению связь можно редактировать свойства связи, такие как название связи и тип связи.

Особенностью программы DBDesigner является то, что в процессе создания ER-диаграмм, отношения будет сформированы автоматически. Это очень удобно при проектировании, т.к. позволяет разработчику не запоминать правила формирования отношений для различных связей между таблицами, а получать все автоматически на основе анализа предметной области.

Связи между таблицами можно корректировать, используя "Редактор связей" (RelationEditor), вызываемый двойным щелчком мыши. В "Редакторе связей" можно задать имя связи, изменить ее тип и задать ограничения на данные таблицы при удалении и добавлении в нее данных.

2. Кодирование

DBDesigner позволяет преобразовывать полученную модель в код на языке SQL, который может быть использован для создания базы данных с помощью других средств, например, с помощью MySQL.

Для получения кода необходимо выбрать в меню File – Export – SQLCreateScript.

Откроется диалоговое окно, представленное на рисунке:

В основных настройках (GeneralSettings) можно назначить экспортировать в SQL код только выделенные таблицы или экспортировать все таблицы модели, также можно задать упорядочить таблицы по внешним ключам.

- Exportselectedtablesonly – кодировать только выбранные таблицы
- OrderTablesbyForeignKeys – позволяет изменить порядок кодирования

В настройках SQL кода (SQLCreatesSettings) можно настроить параметры связанные с первичными ключами и внешними ключами, а также задать настройки относительно индексов.

CopyScripttoClipboard. Позволяет скопировать SQL код в буфер обмена;

SaveScripttofile. Позволяет сохранить SQL код в файл. Файл сохраняется в формате *.sql. Открыть его можно и в текстовом редакторе "Блокнот" Выбрав необходимые параметры, необходимо нажать SaveScripttofile.

Файл с SQL кодом будет сохранен на диске.

```
CREATETABLEfirm (
Gos_reg_number INTEGER UNSIGNED NOT NULL,
owner_FIO_owner VARCHAR(255) NOT NULL,
ur_adress VARCHAR(255) NOT NULL DEFAULT not detected,
firm_name VARCHAR(45) NOT NULL,
firm_type VARCHAR(10) NOT NULL,
reg_date DATE NOT NULL,
PRIMARY KEY(Gos_reg_number, owner_FIO_owner),
INDEX firm_FKIndex1(owner_FIO_owner)
);
CREATE TABLE office (
office_adress VARCHAR(255) NOT NULL,
firm_owner_FIO_owner VARCHAR(255) NOT NULL,
firm_Gos_reg_number INTEGER UNSIGNED NOT NULL,
office_square INTEGER UNSIGNED NOT NULL,
personal INTEGER UNSIGNED NOT NULL,
PRIMARY KEY(office_adress, firm_owner_FIO_owner,
firm_Gos_reg_number),
INDEX office_FKIndex1(firm_Gos_reg_number, firm_owner_FIO_owner)
);
CREATE TABLE owner (
FIO_owner VARCHAR(255) NOT NULL,
owner_adress VARCHAR(255) NOT NULL,
owner_tel_number VARCHAR(20) NOT NULL,
owner_date_reg DATE NOT NULL,
owner_pasport VARCHAR(255) NOT NULL,
```

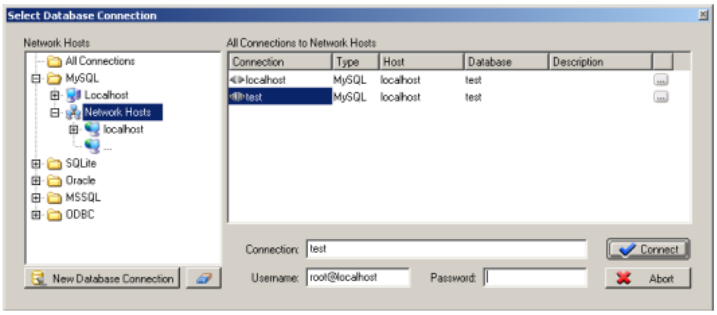
PRIMARYKEY(FIO_owner)
);

3. Работа с базой данных

DBDesigner позволяет также создавать базу данных на сервере и выполнять с ней различные операции. Это обеспечивается за счет подключения DBDesigner MySQL серверу, созданию базы данных и установлению синхронизации между базой на сервере и визуальной моделью. Синхронизация – это сравнение визуальной модели и базы данных, находящейся на сервере. В случае внесения изменений в таблицу, изменения связей между таблицами или удаления таблиц в модели, DBDesigner внесет и соответствующие изменения в базу на сервере.

1) Установление соединения с базой данных на сервере

Для занесения базы данных, соответствующей полученной модели, на серве MySQL, необходимо установить соединение с сервером



ВыполнитеDatabase –> Connect to Database.

ВокнеNetwork Hostsвыберите MySQL

В открывшемся списке баз данных, выберите либо существующую базу, либо создать новую, щелкнув два раза по значку "…" и задав имя новой базы.

Введите название соединения (Connection), имя пользователя (Username) и пароль (Password), если они нужны.

В центральном окне находится список серверов баз данных, с которыми велась работа и для которых указаны IP-адрес, тип, размещение и название. Так как в данной работе предполагается, что сервер MySQL находиться на локальном компьютере, то все необходимые параметры будут установлены автоматически. Однако при использовании сети, необходимо знать IP-адрес сервера и иметь доступ на работу.

Нажмите на кнопку Connect, после чего соединение с базой будет установлено.

2) Синхронизация

Для синхронизации модели и базы на сервере необходимо:

- Выбрать в меню Database - DatabaseSynchronisation и установить соединение с нужной базой.

- В диалоговом окне DatabaseSynchronisation задать необходимые параметры:

ApplychangestoDatabase – вносить изменения модели в базу

Don'tdeleteexisiting Tables – при использовании этой опции таблицы, удаленные из модели, не будут удалены из базы

Execute Standard Inserts when Creating New Tables – создавать стандартныйзапрос на внесение данных в таблицу

Задания

1 Изучить теоретические сведения.

2 В соответствии с вариантом задания выполнить корректировку структуры базы данных, применяя принципы нормализации базы данных.

3 Создать схему базы данных с использованием CASE-средств.

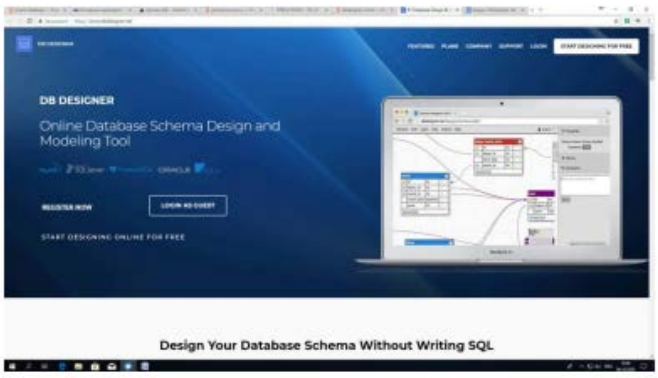
4 Преобразовать схему в SQL код

Порядок выполнения работы

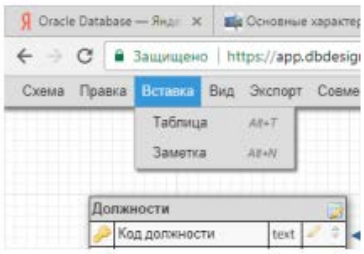
Для создания схемы воспользуемся программой DB DESIGNER ONLINE. DBDesigner – это свободно распространяемая CASE-система, предназначенная для проектирования, моделирования, создания и поддержки информационных систем. Программа может использоваться для Windows 2000/XP, Linux KDE/Gnome и MySQL. DBDesigner позволяет:

- создавать модель проектируемой системы;
- преобразовывать модели системы в SQL-код, который можно использовать для создания базы данных с помощью DBDesigner или другого средства и пр.

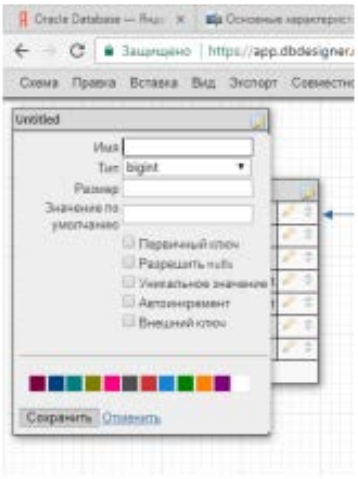
Перед началом работы необходимо зарегистрироваться, указав адрес почтового ящика.



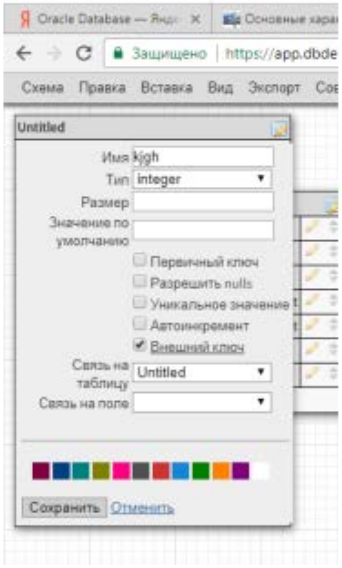
В меню выбираем Вставка/ Таблица.



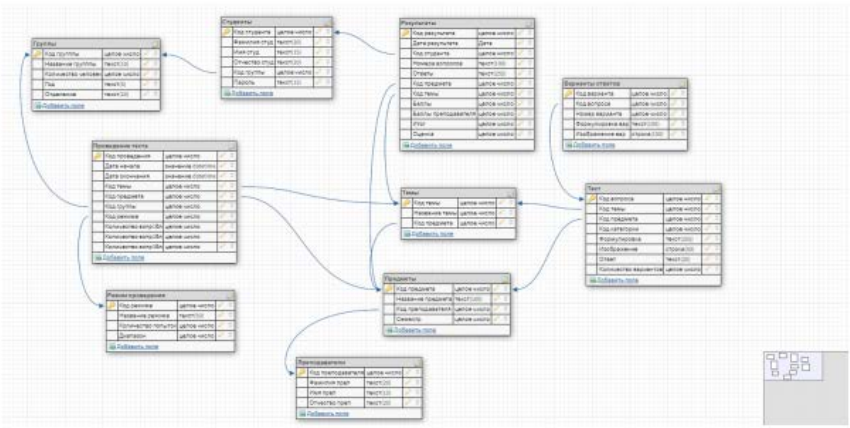
В появившейся области для ввода меняем название таблицы, затем нажимаем Добавить поле, вносим его характеристики.



Если поле является внешним ключом, в открывшемся меню указываем таблицу для связи, поле появится автоматически. После сохранения нового поля, программа сама свяжет таблицы.

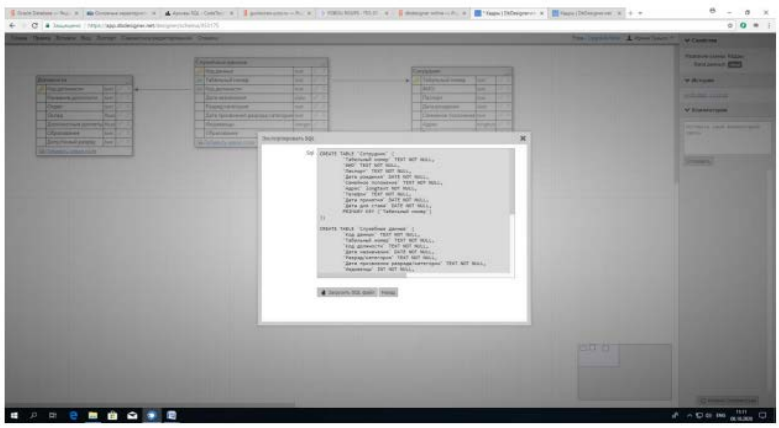


Кроме имени, типа и размера поля желательно задать другие характеристики, например, определить, может ли это поле оставаться пустым или могут ли данные в поле повторяться. Таким образом создать все таблицы базы данных.



Кроме имени, типа и размера поля желательно задать другие характеристики, например, определить, может ли это поле оставаться пустым или могут ли данные в поле повторяться. Таким образом создать все таблицы базы данных.

Для получения SQL кода выбираем в меню раздел Экспорт, и DBDesigner генерирует код создания таблиц.



Этот код можно скопировать и использовать для создания таблиц базы данных.

```
CREATE TABLE `Группы`
(`Код группы` INT NOT NULL,
`Название группы` CHAR(10) NOT NULL,
`Количество человек` INT,
`Год` CHAR(5),
`Отделение` CHAR(20) NOT NULL,
PRIMARY KEY (`Код группы`)
);
CREATE TABLE `Предметы`
(`Код предмета` INT NOT NULL,
`Название предмета` CHAR(100) NOT NULL,
`Код преподавателя` INT NOT NULL,
`Семестр` INT NOT NULL,
PRIMARY KEY (`Код предмета`)
);
CREATE TABLE `Преподаватели` (
`Код преподавателя` INT NOT NULL,
`Фамилия преп` CHAR(20) NOT NULL,
`Имя преп` CHAR(15) NOT NULL,
`Отчество преп` CHAR(20) NOT NULL,
PRIMARY KEY (`Код преподавателя`)
);
CREATE TABLE `Проведение теста` (
`Код проведения` INT NOT NULL,
`Дата начала` DATETIME NOT NULL,
`Дата окончания` DATETIME NOT NULL,
`Код темы` INT NOT NULL,
`Код предмета` INT NOT NULL,
`Код группы` INT NOT NULL,
`Код режима` INT NOT NULL,
`Количество вопр1бл` INT NOT NULL,
`Количество вопр2бл` INT NOT NULL,
`Количество вопр3бл` INT NOT NULL,
PRIMARY KEY (`Код проведения`)
);
CREATE TABLE `Режим проведения` (
`Код режима` INT NOT NULL,
`Название режима` CHAR(50) NOT NULL,
`Количество попыток` INT NOT NULL,
`Диапазон` INT NOT NULL,
PRIMARY KEY (`Код режима`)
);
CREATE TABLE `Темы` (
`Код темы` INT NOT NULL,
`Название темы` INT NOT NULL,
`Код предмета` INT NOT NULL,
PRIMARY KEY (`Код темы`)
);
CREATE TABLE `Результаты` (
`Код результата` INT NOT NULL,
`Дата результата` DATE NOT NULL,
`Код студента` INT NOT NULL,
```

```
`Номера вопросов` CHAR(100) NOT NULL,
`Ответы` CHAR(255) NOT NULL,
`Код предмета` INT NOT NULL,
`Код темы` INT NOT NULL,
`Баллы` INT NOT NULL,
`Баллы преподавателя` INT NOT NULL,
`Итог` INT NOT NULL,
`Оценка` INT NOT NULL,
PRIMARY KEY (`Код результата`)
);
CREATE TABLE `Тест` (
`Код вопроса` INT NOT NULL,
`Код темы` INT NOT NULL,
`Код предмета` INT NOT NULL,
`Код категории` INT NOT NULL,
`Формулировка` CHAR(255) NOT NULL,
`Изображение` VARCHAR(50),
`Ответ` CHAR(20),
`Количество вариантов` INT NOT NULL,
PRIMARY KEY (`Код вопроса`)
);
CREATE TABLE `Студенты` (
`Код студента` INT NOT NULL,
`Фамилия студ` CHAR(20) NOT NULL,
`Имя студ` CHAR(15) NOT NULL,
`Отчество студ` CHAR(20) NOT NULL,
`Код группы` INT NOT NULL,
`Пароль` CHAR(15) NOT NULL,
PRIMARY KEY (`Код студента`)
);
CREATE TABLE `Варианты ответов` (
`Код варианта` INT NOT NULL,
`Код вопроса` INT NOT NULL,
`Номер варианта` INT NOT NULL,
`Формулировка вар` CHAR(100) NOT NULL,
`Изображение вар` VARCHAR(100) NOT NULL,
PRIMARY KEY (`Код варианта`)
);
ALTER TABLE `Предметы` ADD CONSTRAINT `Предметы_fk0` FOREIGN KEY (`Код преподавателя`) REFERENCES `Преподаватели`(`Код преподавателя`);
ALTER TABLE `Проведение теста` ADD CONSTRAINT `Проведение теста_fk0` FOREIGN KEY (`Код темы`) REFERENCES `Темы`(`Код темы`);
ALTER TABLE `Проведение теста` ADD CONSTRAINT `Проведение теста_fk1` FOREIGN KEY (`Код предмета`) REFERENCES `Предметы`(`Код предмета`);
ALTER TABLE `Проведение теста` ADD CONSTRAINT `Проведение теста_fk2` FOREIGN KEY (`Код группы`) REFERENCES `Группы`(`Код группы`);
ALTER TABLE `Проведение теста` ADD CONSTRAINT `Проведение теста_fk3` FOREIGN KEY (`Код режима`) REFERENCES `Режим проведения`(`Код режима`);
ALTER TABLE `Темы` ADD CONSTRAINT `Темы_fk0` FOREIGN KEY (`Код предмета`) REFERENCES `Предметы`(`Код предмета`);
ALTER TABLE `Результаты` ADD CONSTRAINT `Результаты_fk0` FOREIGN KEY (`Код студента`) REFERENCES `Студенты`(`Код студента`);
```

ALTER TABLE `Результаты` ADD CONSTRAINT `Результаты_fk1` FOREIGN KEY (`Код предмета`) REFERENCES `Предметы`(`Код предмета`);

ALTER TABLE `Результаты` ADD CONSTRAINT `Результаты_fk2` FOREIGN KEY (`Код темы`) REFERENCES `Темы`(`Код темы`);

ALTER TABLE `Тест` ADD CONSTRAINT `Тест_fk0` FOREIGN KEY (`Код темы`) REFERENCES `Темы`(`Код темы`);

ALTER TABLE `Тест` ADD CONSTRAINT `Тест_fk1` FOREIGN KEY (`Код предмета`) REFERENCES `Предметы`(`Код предмета`);

ALTER TABLE `Студенты` ADD CONSTRAINT `Студенты_fk0` FOREIGN KEY (`Код группы`) REFERENCES `Группы`(`Код группы`);

ALTER TABLE `Варианты ответов` ADD CONSTRAINT `Варианты ответов_fk0` FOREIGN KEY (`Код вопроса`) REFERENCES `Тест`(`Код вопроса`);

Практическое занятие № 8

Тема: Создание базы данных средствами СУБД. Работа с таблицами: добавление, редактирование, удаление, навигация по записям

Цель: Освоить технологию создания базы данных в среде Microsoft Access. Применение основных приемов работы с базами данных: ввода данных, форматирование шрифта.

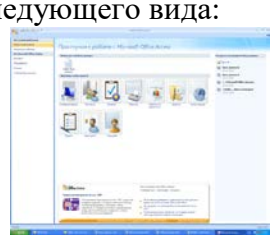
Оборудование: компьютерный кабинет, персональный компьютер, программы Microsoft Access, инструкционная карта.

Ход работы:

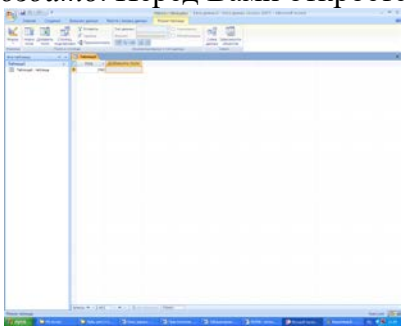
Задание 1. Создание пустой базы данных с помощью шаблонов таблиц.

Технология выполнения задания:

1. Запустите программу СУБД Microsoft Access. Для этого выполните: *Пуск - Все программы - Microsoft office - Microsoft office Access.*
2. Перед Вами откроется окно следующего вида:

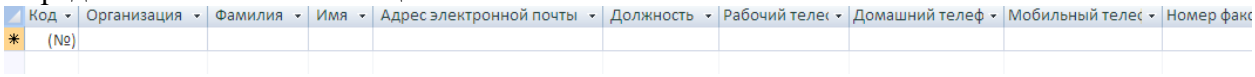


1. Выберите команду *Новая база данных*. Затем введите *имя файла –База работников* и нажмите кнопку *Создать*. Перед Вами откроется окно следующего вида:



Выберите команду *Создание - Шаблоны таблиц - Контакты.*

Перед Вами появится таблица с заголовками:



Заполните ее следующими данными (см. таблицу).

Код	Органи-зация	Фамилия	Имя	Адрес электронной почты	Долж-ность	Рабочий телефон	Домаш-ний телефон	Мобильный телефон	Номер факса	Адрес	Город	Область, край	Индекс	Страна или регион
1	Растр	Иванов	Сергей	Ivanov@mail.ru	инженер	516987	265414	89264586232	264589	Ул.Героев Десантников, 23	Новорос-сийск	Красно-дарский	3117	Россия
2	Иволга	Сидоров	Дмитрий	Sidr@rambler.ru	электрик	264578	514589	89095642378	264578	ул. Кунникова, 32	Новорос-сийск	Красно-дарский	3117	Россия
3	Голден	Петров	Иван	Pertr@land.ru	менеджер	256989	214589	87054268975	564278	ул. Ленина, 12	Новорос-сийск	Красно-дарский	3117	Россия
4	Лайма	Козлова	Элина	Kozl@mail.ru	бухгалтер	264578	214563	89184567896	264578	ул, Советов, 89	Новорос-сийск	Красно-дарский	3117	Россия
5	Комтеко	Лобова	Мария	Lobova@land.ru	директор	568974	245689	89184569875	264532	ул, Рыжова, 96	Новорос-сийск	Красно-дарский	3117	Россия

У Вас должна получится таблица как на рисунке (см. рис.). Сохраните таблицу ( ) под именем *Работник.*

Код	Организация	Фамилия	Имя	Адрес электронной почты	Должность	Рабочий телефон	Домашний телефон	Мобильный телефон	Номер факса
1	Растр	Иванов	Сергей	Ivanov@mail.ru	инженер	516987	265414	89264586232	264589
2	Иволга	Сидоров	Дмитри	Sidr@rambler.ru	электрик	264578	514589	89095642378	264578
3	Голден	Петров	Иван	Pertr@land.ru	менеджер	256989	214589	87054268975	564278
4	Лайма	Козлова	Элина	Kozl@mail.ru	бухгалтер	264578	214563	89184567896	264578
5	Комтеко	Лобова	Мария	Lobova@land.ru	директор	568974	245689	89184569875	264532
(№)									

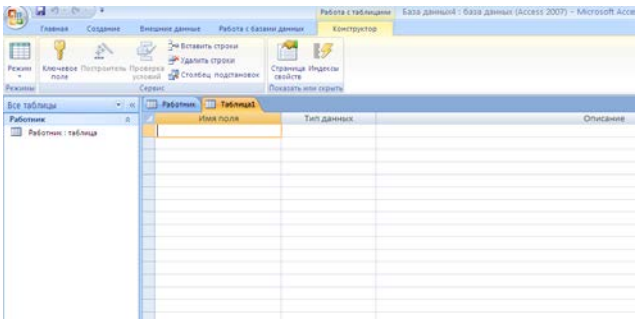
В данной таблице отсортируйте столбец «*Организация*» по алфавиту (*Главная* -).

Задание 2. Создание пустой базы данных с помощью конструктора таблиц.

Технология выполнения задания:

1. Создадим таблицу под именем «*Студент*» с помощью конструктора таблиц. Для этого выполните команду: *Создание – конструктор таблиц*.

Перед Вами откроется окно:



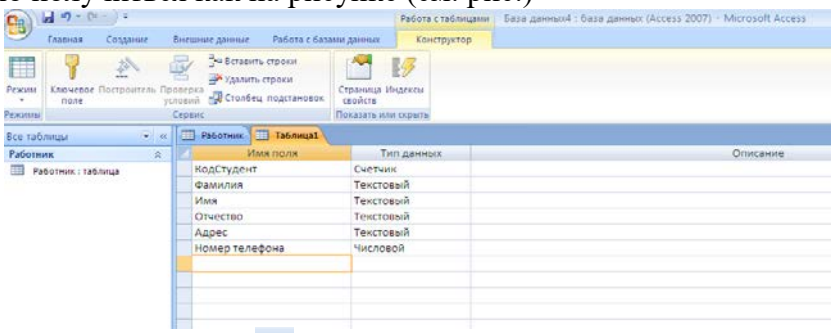
Заполните *Имя поля* следующими данными (заголовками столбцов):

КодСтудент, Фамилия, Имя, Отчество, Адрес, Номер телефона, Специализация. И соответственно *Тип данных*:

КодСтудент – СЧЕТЧИК, *Фамилия, Имя, Отчество, Должность, Адрес, Специализация* – ТЕКСТОВЫЙ,

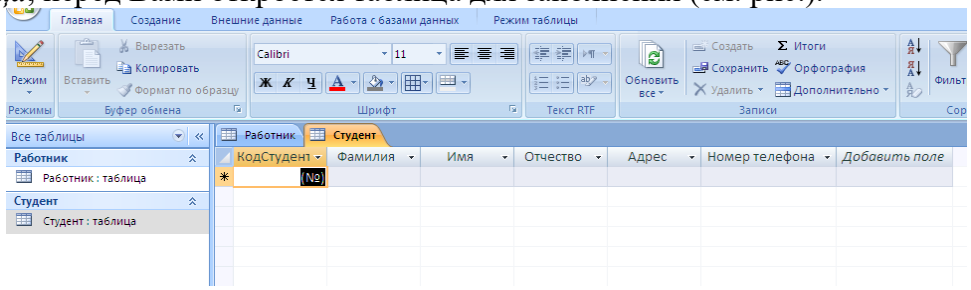
Номер телефона – ЧИСЛОВОЙ.

У Вас должно получиться как на рисунке (см. рис.)



Далее Нажмите сохранить ( ) и назовите таблицу «*Студент*». Он автоматически запросит создать ключевое поле, нажмите кнопку ДА (КодСтудент будет Ключевое поле КодСтудент Счетчик).

Затем двойным щелчком левой кнопкой мыши щелкните слева на таблицу *Студент: таблица*, перед Вами откроется таблица для заполнения (см. рис.).



Заполните эту таблицу следующими данными (см. таблицу) и сохраните.

КодСтудент	Фамилия	Имя	Отчество	Адрес	Номер телефона	Специализация
1	Иванов	Сергей	Александрович	г. Новороссийск	457896	технолог
2	Петров	Сергей	Петрович	г. Москва	7458962	технолог
3	Гаврелеева	Ольга	Ивановна	г. Москва	3698521	бухгалтер
4	Соколова	Инна	Олеговна	г. Новороссийск	852967	бухгалтер
5	Мухина	Олеся	Петровна	г. Москва	8625471	технолог
6	Апареева	Анна	Романовна	г. Люберцы	748596	технолог
7	Глинкина	Дина	Евгеньевна	г. Люберцы	919597	технолог
8	Сорина	Ольга	Сергеевна	г. Москва	9191954	бухгалтер

Задания для самостоятельной работы

Задание 1. Создайте таблицу в *Microsoft office Access* на основе шаблона «*События*». (В той же базе данных «*База работников*» создайте таблицу №3 под именем «*Проведение выставок*», выбрав команду *Создание - Шаблоны таблиц - События*). И заполните таблицу 5-6 записями (*название выставок и дат придумайте сами*). Сохраните.

Задание 2. Создайте таблицу в *Microsoft office Access* с помощью конструктора таблиц. (В той же базе данных «*База работников*» создайте таблицу №4 под именем «*Студенты и задания*»).

Заполните *Имя поля* следующими данными (заголовками столбцов):

КодСтудент, Фамилия, Описание задания, Начальная дата, Конечная дата, Замечания.

И соответственно *Тип данных*:

КодСтудент – СЧЕТЧИК,

Фамилия, Описание задания, Замечания – ТЕКСТОВЫЙ,

Начальная дата, Конечная дата – ДАТА/ВРЕМЯ.

И заполните эту таблицу следующими данными (см. таблицу)

КодСтудент	Фамилия	Описание задания	Начальная дата	Конечная дата	Замечания
1	Иванов	Электронная почта	21.03.09	15.05.09	
2	Петров	Телеконференция	10.02.09	20.05.09	
3	Гаврелеева	Браузер	20.01.09	15.04.09	
4	Соколова	Служба FTP	15.01.09	25.04.09	
5	Мухина	Поисковые системы Интернет	30.01.09	10.05.09	
6	Апареева	Интернет 2	23.02.09	30.05.09	
7	Глинкина	IP-телефония	20.02.09	12.05.09	
8	Сорина	Подключение к Интернету	25.03.09	30.05.09	

Сохраните набранные данные и при автоматическом запросе системы о создании ключевого поля, нажмите кнопку ДА.

Практическое занятие № 9

Тема: Создание взаимосвязей

***Цель работы*:** изучение приемов установки связей между таблицами базы данных.

Создайте в **режиме Конструктор** 3 таблицы: ***Сотрудники, Клиенты и Заказы***. Если все сведения поместить в одной таблице, то она станет очень неудобной для работы. В ней начнутся повторы данных. Всякий раз, когда сотрудник Иванов будет работать с какой-либо фирмой, придется прописывать данные о сотруднике и клиенте заново, в результате чего можно допустить множество ошибок. Чтобы уменьшить число ошибок, можно исходную таблицу разбить на несколько таблиц и установить связи между ними. Это будет более рационально.

Таблица Сотрудники

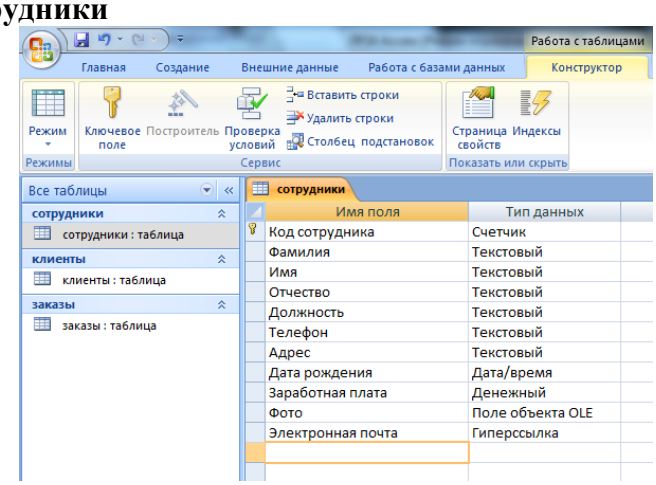


Таблица Клиенты

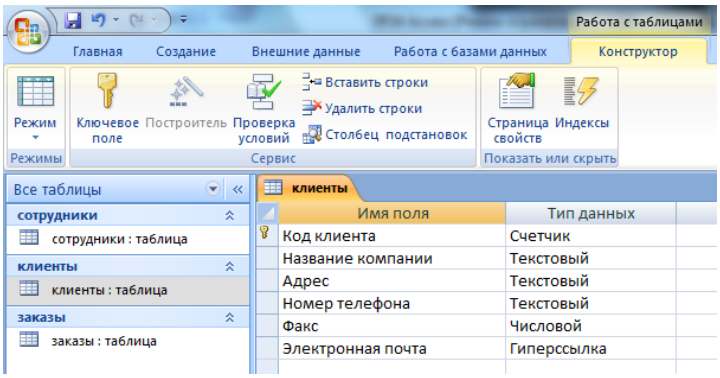
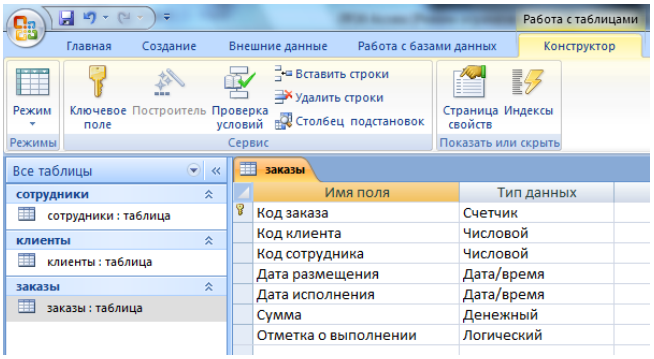


Таблица Заказы



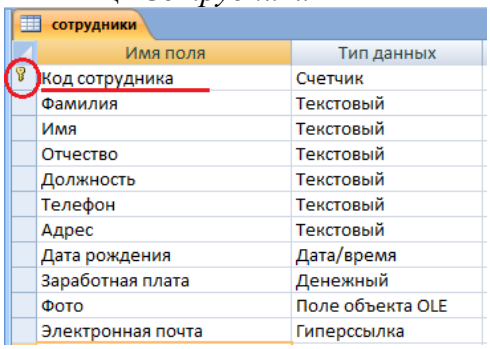
Установите ключевые поля.

Отдельные таблицы, содержащие информацию по определенной теме, необходимо связать в единую структуру базы данных. Для связывания таблиц следует задать ключевые поля.

Ключ состоит из одного или нескольких полей, значения которых однозначно определяют каждую запись в таблице. Наиболее подходящим в качестве ключевого поля является *Счетчик*, так как значения в данном поле являются уникальными (т. е. исключают повторы).

При создании таблиц в режиме конструктора ключевое поле устанавливается автоматически. Откройте созданные Вами таблицы в режиме Конструктор и проверьте установленные ключевые поля:

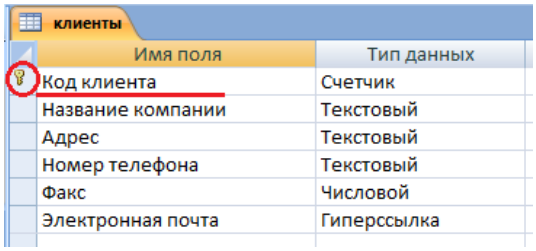
1) в таблице *Сотрудники* ключевое поле *Код сотрудника*



сотрудники

Имя поля	Тип данных
Код сотрудника	Счетчик
Фамилия	Текстовый
Имя	Текстовый
Отчество	Текстовый
Должность	Текстовый
Телефон	Текстовый
Адрес	Текстовый
Дата рождения	Дата/время
Заработная плата	Денежный
Фото	Поле объекта OLE
Электронная почта	Гиперссылка

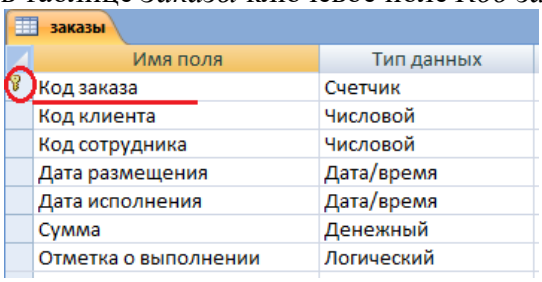
2) в таблице *Клиенты* ключевое поле *Код клиента*



клиенты

Имя поля	Тип данных
Код клиента	Счетчик
Название компании	Текстовый
Адрес	Текстовый
Номер телефона	Текстовый
Факс	Числовой
Электронная почта	Гиперссылка

3) в таблице *Заказы* ключевое поле *Код заказа*



заказы

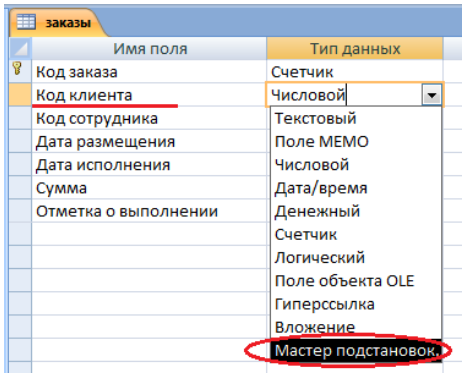
Имя поля	Тип данных
Код заказа	Счетчик
Код клиента	Числовой
Код сотрудника	Числовой
Дата размещения	Дата/время
Дата исполнения	Дата/время
Сумма	Денежный
Отметка о выполнении	Логический

Если значение *Ключевых полей* не задалось автоматически, то задайте их вручную. Для этого откройте **таблицу Сотрудники** в **режиме Конструктора**. Нажмите правой кнопкой мыши на поле **Код сотрудника** и в появившемся контекстном меню выберите команду **Ключевое поле**. Если в таблице необходимо установить несколько ключевых полей, то выделить их можно, удерживая клавишу Ctrl. Для **таблицы Клиенты** установите ключевое поле **Код клиента**, а для **таблицы Заказы - Код заказа**.

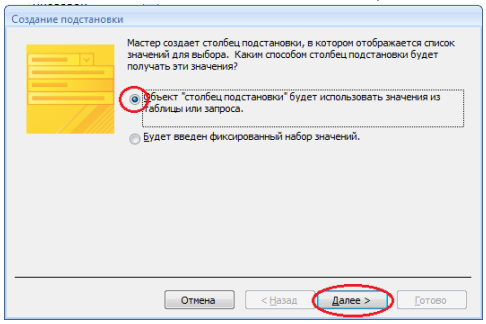
5. Создайте раскрывающиеся списки с помощью **Мастера подстановок.**

Таблица **Заказы** содержит поля **Код сотрудника** и **Код клиента**. При их заполнении могут возникнуть некоторые трудности, так как не всегда удается запомнить все предприятия, с которыми работает фирма, и всех сотрудников с номером кода. Для удобства можно создать раскрывающиеся списки с помощью **Мастера подстановок**.

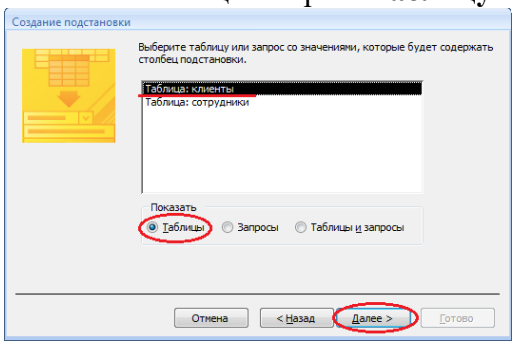
Откройте **таблицу Заказы** в режиме **Конструктора.** Для поля **Код клиента** выберите **тип данных Мастер подстановок.**



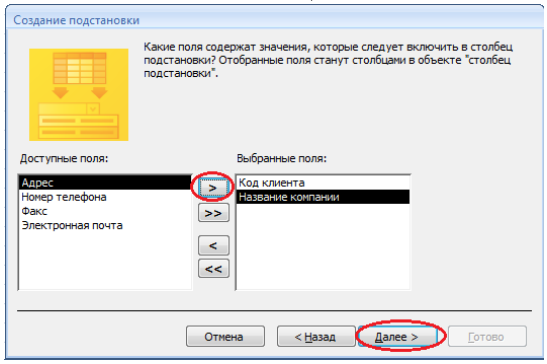
В появившемся окне выберите команду **Объект "столбец подстановки" будет использовать значения из таблицы или запроса** и щелкните на кнопке **Далее.**



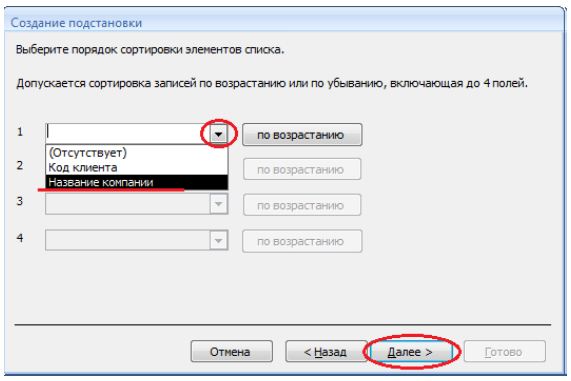
В списке таблиц выберите **таблицу Клиенты** и щелкните на кнопке **Далее**.



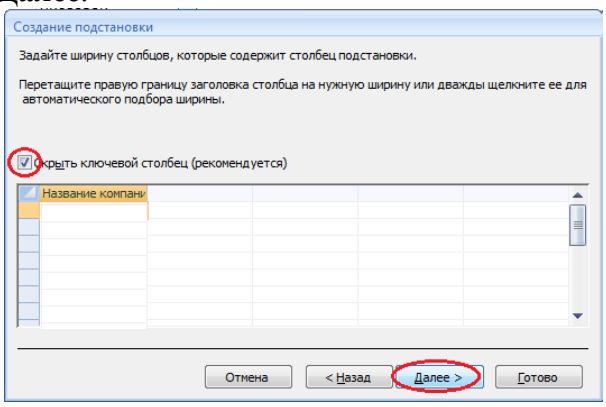
В списке **Доступные поля** выберите поле **Код клиента** и щелкните на кнопке со стрелкой >>, чтобы ввести поле в список **Выбранные поля**. Таким же образом добавьте поле **Название компании** и щелкните на кнопке **Далее**.



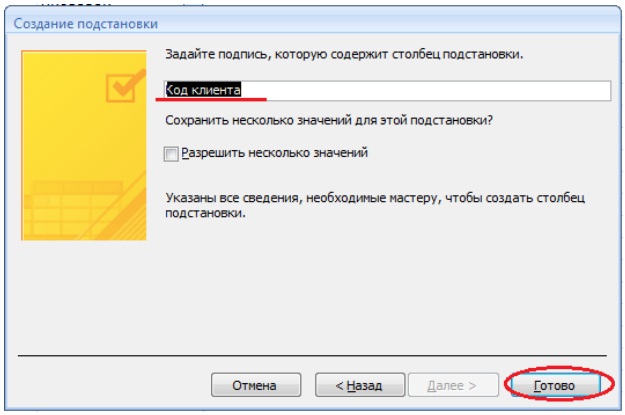
Выберите порядок сортировки списка по полю **Название компании** и нажмите кнопку **Далее**.



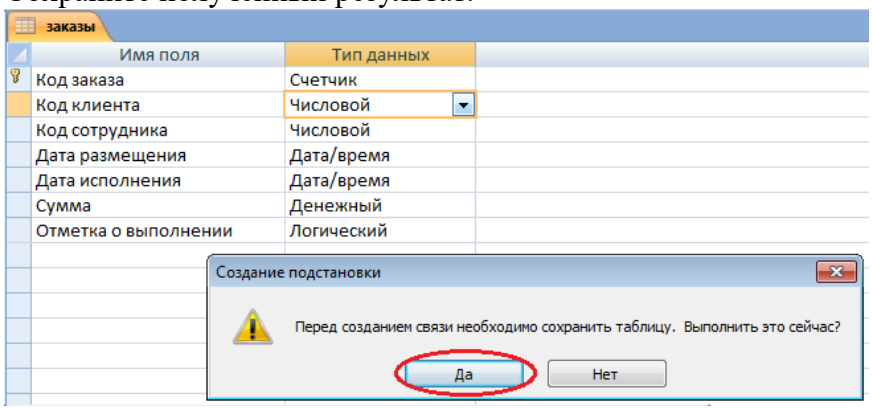
В следующем диалоговом окне задайте необходимую ширину столбцов раскрывающегося списка, установите флажок **Скрыть ключевой столбец** и нажмите кнопку **Далее**.



На последнем шаге **Мастера подстановок** замените при необходимости надпись для поля подстановок и щелкните на кнопке **Готово**.

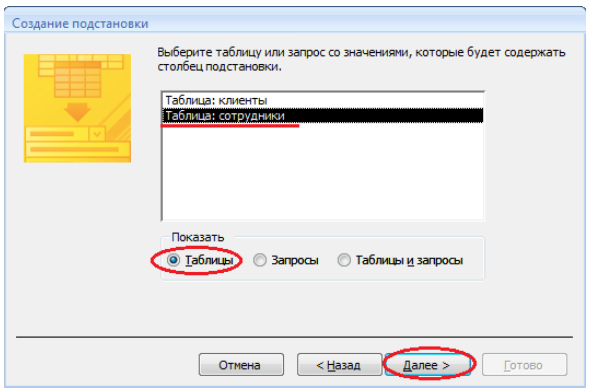


Сохраните полученный результат.

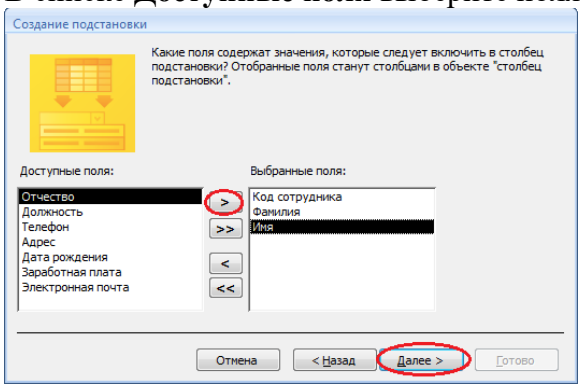


6. Аналогичным образом создайте раскрывающийся список для поля **Код сотрудника.**

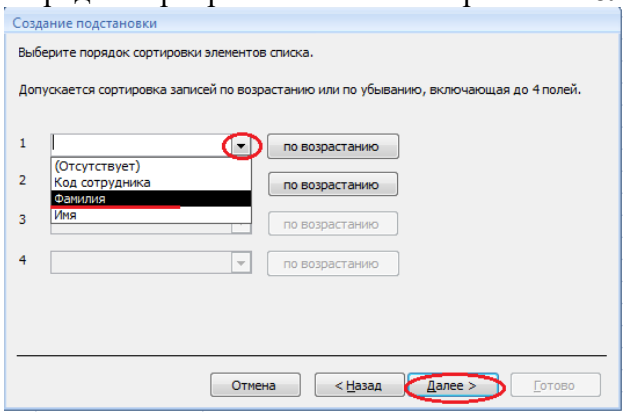
Теперь в списке таблиц выберите **таблицу Сотрудники**



В списке **Доступные поля** выберите поля **Код сотрудника, Фамилия, Имя**.



Порядок сортировки списка выберите по **полю Фамилия**.



Все остальные действия проводятся аналогично пункту 6.

7. Создайте связей между таблицами.

Существует несколько типов отношений между таблицами:

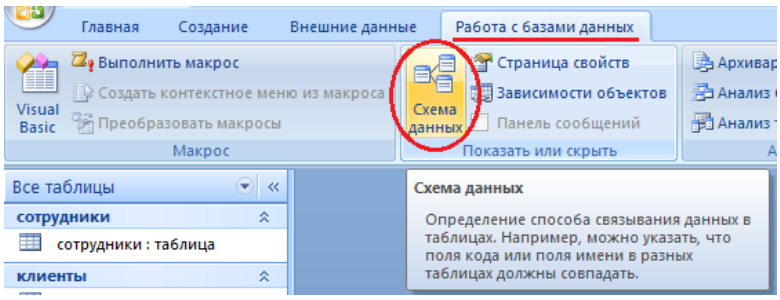
- *при отношении «один-к-одному»* каждой записи ключевого поля в первой таблице соответствует только одна запись в связанном поле другой таблицы, и наоборот. Отношения такого типа используются не очень часто. Иногда их можно использовать для разделения таблиц, содержащих много полей, для отделения части таблицы по соображениям безопасности;

- *при отношении «один-к-многим»* каждой записи в первой таблице соответствует несколько записей во второй, но запись во второй таблице не может иметь более одной связанной записи в первой таблице;

- *при отношении «многие-к-многим»* одной записи в первой таблице могут соответствовать несколько записей во второй таблице, а одной записи во второй таблице могут соответствовать несколько записей в первой.

Закройте все открытые таблицы, так как создавать или изменять связи между открытыми таблицами нельзя.

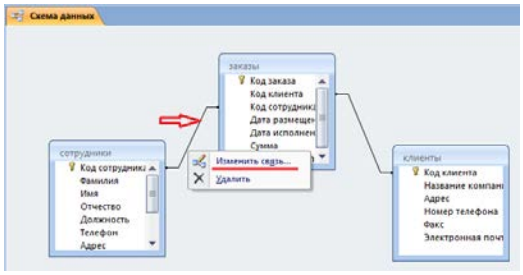
Выполните команду вкладки Лента **Работа с базами данных** кнопка **Схема данных**



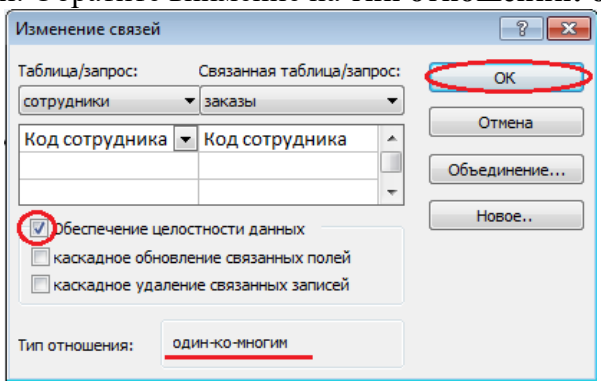
Если ранее никаких связей между таблицами базы не было, то при открытии окна **Схема данных** одновременно открывается окно **Добавление таблицы**, в котором выбираются нужные таблицы. Для добавления в схему данных новой таблицы необходимо щелкнуть правой кнопкой мыши на схеме данных и в контекстном меню выбрать пункт **Добавить таблицу**.

Если связи между таблицами уже были заданы, то откроется окно **Схема данных**, на котором будут отображены таблицы и связи между ними.

Отредактируйте связь между таблицами Сотрудники и Заказы, для этого щелкните правой кнопкой мыши (ПКМ) на линию связи и в открышемся контекстном меню выберите команду **Изменить связь**.



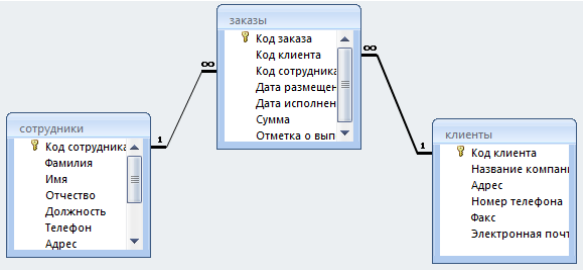
Откроется диалоговое окно **Изменение связей**, в котором включите флажок **Обеспечение целостности данных**. Это позволит предотвратить случаи удаления записей из одной таблицы, при которых связанные с ними данные других таблиц останутся без связи. Обратите внимание на **тип отношений: один-ко-многим.**



Флажки **Каскадное обновление связанных полей** и **Каскадное удаление связанных записей** обеспечивают одновременное обновление или удаление данных во всех подчиненных таблицах при их изменении в главной таблице.Параметры связи можно изменить, нажав на кнопку Объединение. После установления всех необходимых параметров нажмите **кнопку ОК**.

Аналогично измените связь между таблицами Клиенты и Заказы.

В результате должна получиться схема данных, представленная на рисунке.



На схеме данных связи отображаются в виде соединительных линий со специальными значками около таблиц. Связь «один-к-многим» помечается «1» вблизи главной таблицы (имеющей первичный ключ) и «∞» вблизи подчиненной таблицы (имеющей внешний ключ). Связь «один-к-одному» помечается двумя «1» (оба поля таблиц имеют первичные ключи). Неопределенная связь не имеет никаких знаков. Если установлено объединение, то его направление отмечается стрелкой на конце соединительной линии (ни одно из объединенных полей не является ключевым и не имеет уникального индекса).

7. В **таблицу Сотрудники** внесите данные **о 7 работниках**.

сотрудники

Код сотр	Фамилия	Имя	Отчество	Должность	Телефон	Адрес	Дата рожде	Заработная	Фото	Электронная почта
1	Иванов	Сергей	Юрьевич	Директор	89182121567	ул.Батарейная,8	05.07.1987	57 000,00 ₽		Ivanov@mail.ru
2	Орлова	Юлия	Константиновна	Зам директора	89894938474	ул.Красная,10	07.09.1985	50 000,00 ₽		orlova@mail.ru
3	Романов	Вадим	Романович	Менеджер	89883467464	ул.Матвиенко,46	13.05.1989	50 000,00 ₽		romanov@mail,ru
4	Суворов	Максим	Александрович	Менеджер	89184857634	ул.Минская,11	19.01.1983	45 000,00 ₽		syvorov@mailo.ru
5	Марченко	Андрей	Евгеньевич	Бухгалтер	89887765265	ул.Мамаева,1	22.04.1989	42 500,00 ₽		Marchenko@mail.ru
6	Вдовенко	Николай	Андреевич	Упаковщик	89894289642	ул.Рабочая,19	27.08.1990	40 000,00 ₽		Vdovenko@mail.ru
7	Афонин	Олег	Павлович	Грузчик	89641414100	ул.Свободы,27	21.05.1984	38 000,00 ₽		Afonia@mail.ru

8. В **таблицу Клиенты** внесите данные **о 7 предприятиях**, с которыми работает данная фирма.

клиенты

Код клиент	Название компании	Адрес	Номер теле	Факс	Электронная почта
1	NaVi	ул. Советов,47	89188754416	861276543	navi@mail.ru
2	Virtys	ул. Мира,31	89897464633	861209864	virtys@gmail.com
3	Godsent	ул. Рубина,11	89886741641	861273733	godsent12@bk.com
4	NiP	ул. Куникова,89	89647814711	861247474	n1pi@mail.ru
5	Qbfire	ул. Ленина,13	89881614614	861274644	qbf1r3@gmail.com
6	Gambit	ул. Планеристов,99	89894164141	861284174	gambity12@mail.ru
7	Vega	ул. Весенняя,76	89181571751	861267464	v3gaa2133@mail.ru

9. В **таблице Заказы** оформите **5 заявок**, поступивших на фирму.

заказы

Код заказа	Код клиент	Код сотрудника	Дата размещения	Дата исполнения	Сумма	Отметка о выполнении
1	Godsent	Романов	03.03.2018	05.03.2018	30 000,00 ₽	☑
2	Gambit	Суворов	04.03.2018	09.03.2018	20 000,00 ₽	☐
3	NaVi	Суворов	01.03.2018	04.03.2018	54 000,00 ₽	☑
4	Vega	Вдовенко	02.03.2018	07.03.2018	15 000,00 ₽	☑
5	NiP	Вдовенко	13.03.2018		27 000,00 ₽	☑

10. Покажите работу преподавателю.

11. Ответьте на контрольные вопросы.

Контрольные вопросы:

1 С помощью чего можно создавать таблицы?

2 Что такое ключевое поле?

3 Как установить несколько ключевых полей?

4 Как установить связи между таблицами?

5 Какие существуют отношения между таблицами?

6 Что означают на схеме данных «1» и «∞»?

7 Зачем нужен Мастер подстановок?

8 Для чего нужен механизм запросов?

Практическое занятие № 10

Тема: Сортировка, поиск и фильтрация данных

Цель работы: получить практические навыки и умения при организации отбора и сортировки данных в таблицах MS Excel.

Практическая часть

Задание 1

В таблице «Доход сотрудников» выполнить сортировку и фильтрацию данных.

Порядок выполнения работы:

1 Запустить программу Microsoft Excel.

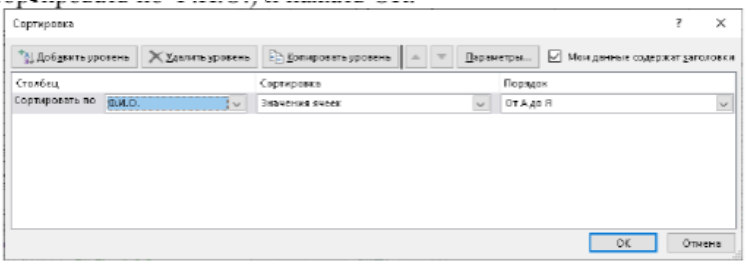
2 Создать таблицу. Заполнить исходными данными. Выполнить необходимое форматирование.

РАСЧЕТ ДОХОДА СОТРУДНИКОВ ОРГАНИЗАЦИИ

Таблица расчетов заработной платы

№	Ф.И.О.	Оклад	Подоходный налог	Отчисления в благотворительный фонд	Всего удержано	К выдаче
1	Петров И.С.	1 250,00 ₽	110,50 ₽	37,50 ₽	148,00 ₽	1 102,00 ₽
2	Антонова Н.г.	1 500,00 ₽	143,00 ₽	45,00 ₽	188,00 ₽	1 312,00 ₽
3	Виноградова Н.Н.	1 750,00 ₽	175,50 ₽	52,50 ₽	228,00 ₽	1 522,00 ₽
4	Гусева И.Д.	1 850,00 ₽	188,50 ₽	55,50 ₽	244,00 ₽	1 606,00 ₽
5	Денисова Н.В.	2 000,00 ₽	208,00 ₽	60,00 ₽	268,00 ₽	1 732,00 ₽
6	Зайцев К.К.	2 250,00 ₽	240,50 ₽	67,50 ₽	308,00 ₽	1 942,00 ₽
7	Иванова К.Е.	2 700,00 ₽	299,00 ₽	81,00 ₽	380,00 ₽	2 320,00 ₽
8	Кравченко Г.И.	3 450,00 ₽	396,50 ₽	103,50 ₽	500,00 ₽	2 950,00 ₽
	Итого	16 750,00 ₽	1 761,50 ₽	502,50 ₽	2 264,00 ₽	14 486,00 ₽

3 Произвести сортировку по фамилиям сотрудников в алфавитном порядке по возрастанию (выделите блок ячеек B6:G13 без итогов, выберите в меню Данные команду Сортировка, сортировать по Ф.И.О.) и нажать ОК.



После сортировки данные будут выглядеть так:

РАСЧЕТ ДОХОДА СОТРУДНИКОВ ОРГАНИЗАЦИИ

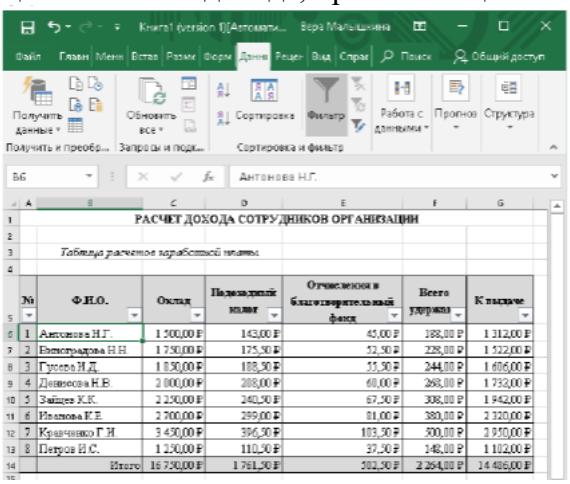
Таблица расчетов заработной платы

№	Ф.И.О.	Оклад	Подоходный налог	Отчисления в благотворительный фонд	Всего удержано	К выдаче
1	Антонова Н.г.	1 500,00 ₽	143,00 ₽	45,00 ₽	188,00 ₽	1 312,00 ₽
2	Виноградова Н.Н.	1 750,00 ₽	175,50 ₽	52,50 ₽	228,00 ₽	1 522,00 ₽
3	Гусева И.Д.	1 850,00 ₽	188,50 ₽	55,50 ₽	244,00 ₽	1 606,00 ₽
4	Денисова Н.В.	2 000,00 ₽	208,00 ₽	60,00 ₽	268,00 ₽	1 732,00 ₽
5	Зайцев К.К.	2 250,00 ₽	240,50 ₽	67,50 ₽	308,00 ₽	1 942,00 ₽
6	Иванова К.Е.	2 700,00 ₽	299,00 ₽	81,00 ₽	380,00 ₽	2 320,00 ₽
7	Кравченко Г.И.	3 450,00 ₽	396,50 ₽	103,50 ₽	500,00 ₽	2 950,00 ₽
8	Петров И.С.	1 250,00 ₽	110,50 ₽	37,50 ₽	148,00 ₽	1 102,00 ₽
	Итого	16 750,00 ₽	1 761,50 ₽	502,50 ₽	2 264,00 ₽	14 486,00 ₽

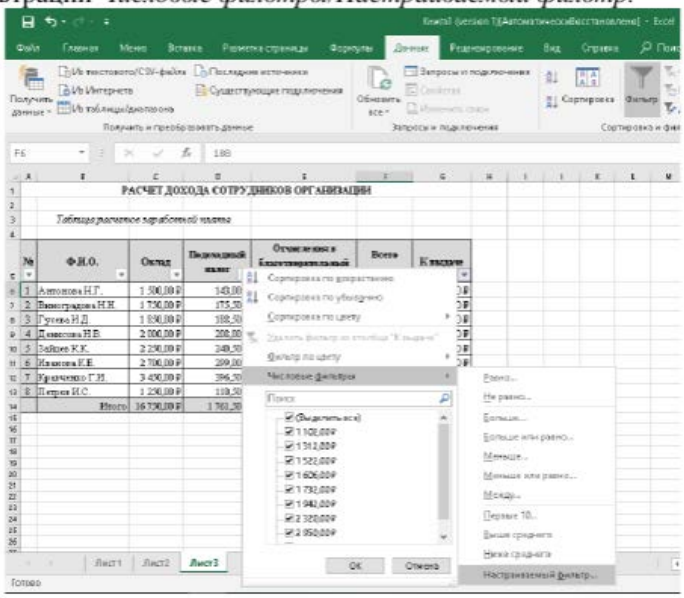
4 Построить гистограмму по итогам расчета (данные столбца «К выдаче»). В качестве подписей оси «Х» указать фамилии сотрудников.

5 Скопировать таблицу на Лист2.

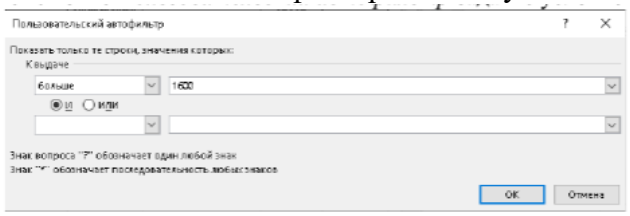
6 Произведите фильтрацию значений дохода, превышающих 1600 р.



Щелкните по стрелке в заголовке поля, на которое будет наложено условие (в столбце «К выдаче»), и вы увидите список всех неповторяющихся значений этого поля. Выберите команду для фильтрации Числовые фильтры/Настраиваемый фильтр.



В открывшемся окне Пользовательский автофильтр задайте условие - больше 1600



Конечный вид таблицы после фильтрации:

РАСЧЕТ ДОХОДА СОТРУДНИКОВ ОРГАНИЗАЦИИ

Таблица расчетов заработной платы

№	Ф.И.О.	Оклад	Подоходный налог	Отчисления в благотворительный фонд	Всего удержано	К выдаче
3	Гусева И.Д.	1 850,00 ₽	188,50 ₽	55,50 ₽	244,00 ₽	1 606,00 ₽
4	Денисова Н.В.	2 000,00 ₽	200,00 ₽	60,00 ₽	260,00 ₽	1 732,00 ₽
5	Зайцев К.К.	2 250,00 ₽	240,50 ₽	67,50 ₽	308,00 ₽	1 942,00 ₽
6	Иванова К.Е.	2 700,00 ₽	299,00 ₽	81,00 ₽	380,00 ₽	2 320,00 ₽
7	Кравченко Г.И.	3 450,00 ₽	396,50 ₽	103,50 ₽	500,00 ₽	2 950,00 ₽
	Итого	16 750,00 ₽	1 761,50 ₽	302,50 ₽	2 364,00 ₽	14 486,00 ₽

6 Сохранить документ с именем Сортировка и фильтрация.

Задание 2

В таблице «Средняя годовая температура воздуха» выполнить форматирование и ввод данных.

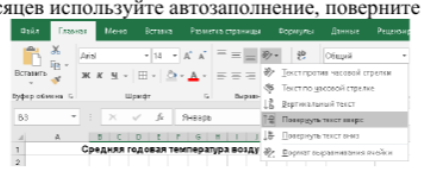
Порядок выполнения работы:

1 Запустить программу Microsoft Excel.

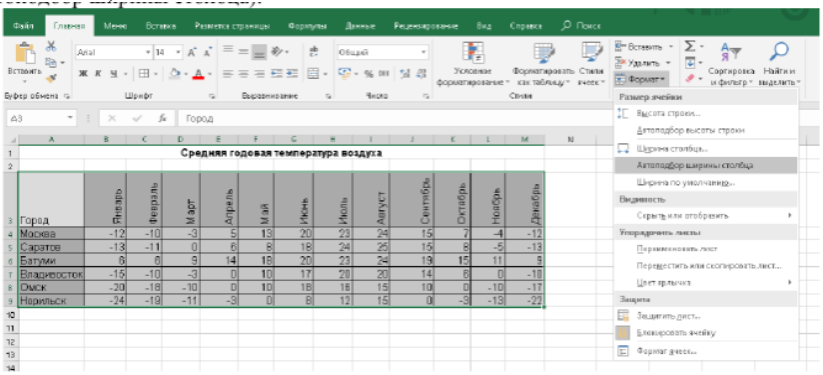
2 Создать таблицу. Заполнить исходными данными. Выполнить необходимое форматирование.

Средняя годовая температура воздуха												
Город	Январь	Февраль	Март	Апрель	Май	Июнь	Июль	Август	Сентябрь	Октябрь	Ноябрь	Декабрь
Москва	-12	-10	-3	5	13	20	23	24	15	7	-4	-12
Саратов	-13	-11	0	6	8	18	24	25	15	8	-5	-13
Батуми	6	6	9	14	18	20	23	24	19	15	11	9
Владивосток	-15	-10	-3	0	10	17	20	20	14	6	0	-10
Омск	-20	-18	-10	0	10	18	16	15	10	0	-10	-17
Норильск	-24	-19	-11	-3	0	8	12	15	0	-3	-13	-22

3. При наборе месяцев используйте автозаполнение, поверните данные на 90 град.



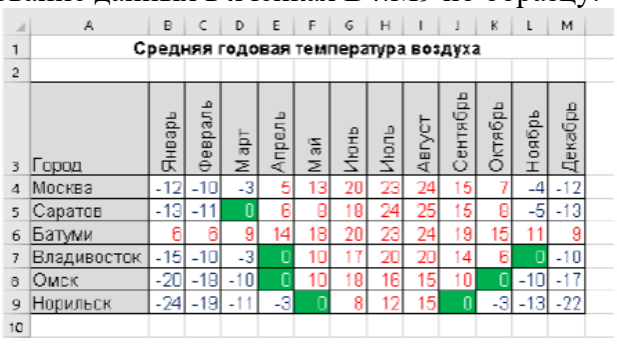
4 Используйте автоподбор ширины ячеек, предварительно выделив таблицу (Формат/ Автоподбор ширины столбца).



После автободбора ширины столбца таблица будет выглядеть так:

Средняя годовая температура воздуха												
Город	Январь	Февраль	Март	Апрель	Май	Июнь	Июль	Август	Сентябрь	Октябрь	Ноябрь	Декабрь
Москва	-12	-10	-3	5	13	20	23	24	15	7	-4	-12
Саратов	-13	-11	0	6	8	18	24	25	15	8	-5	-13
Батуми	6	6	9	14	18	20	23	24	19	15	11	9
Владивосток	-15	-10	-3	0	10	17	20	20	14	6	0	-10
Омск	-20	-18	-10	0	10	18	16	15	10	0	-10	-17
Норильск	-24	-19	-11	-3	0	8	12	15	0	-3	-13	-22

5 Выполнить форматирование данных в ячейках B4:M9 по образцу:



Средняя годовая температура воздуха												
Город	Январь	Февраль	Март	Апрель	Май	Июнь	Июль	Август	Сентябрь	Октябрь	Ноябрь	Декабрь
Москва	-12	-10	-3	5	13	20	23	24	15	7	-4	-12
Саратов	-13	-11	0	6	8	18	24	25	15	8	-5	-13
Батуми	6	6	9	14	18	20	23	24	19	15	11	9
Владивосток	-15	-10	-3	0	10	17	20	20	14	6	0	-10
Омск	-20	-18	-10	0	10	18	16	15	10	0	-10	-17
Норильск	-24	-19	-11	-3	0	8	12	15	0	-3	-13	-22

6 Сохранить документ с именем Средняя годовая температура воздуха.

Самостоятельная работа:

Задание 1

Создание таблицы, сортировка и фильтрация данных.

Порядок выполнения работы:

1 Запустите программу Microsoft Excel.

2 Создайте в книге 11 листов.

3 На Листе1 создайте таблицу «Планеты». Заполните исходными данными. Выполнить необходимое форматирование.

	A	B	C	D	E	F	G
1	ПЛАНЕТЫ						
2							
3	Планета	Масса ($кг*10^{22}$)	Диаметр (км)	Плотность ($г/см^3$)	Температура поверхности, максимальная/минимальная (^0C)	Скорость вращения по орбите (км/с)	Ускорение свободного падения ($м/с^3$)
4	Меркурий	33	4870	5,43	430	47,9	3,7
5	Венера	487	12100	5,25	480	35	8,9
6	Земля	597,6	12756	5,518	58	29,8	0,8
7	Луна	7,35	3476	3,343	-150	1,03	1,62
8	Марс	64	6670	3,95	-150	24,1	3,7
9	Юпитер	190000	143760	1,31	-160	13,1	25,8
10	Сатурн	56800	120240	0,71	-150	9,6	11,3
11	Уран	8700	51300	1,27	-220	6,8	9
12	Нептун	10000	49500	1,77	-213	5,4	11,6
13	Плутон	1,3	2324	2	-230	4,7	0,61
14							

4 Скопируйте созданную таблицу на остальные листы.

5 На Листе2 отсортируйте по алфавиту столбец «Планета».

6 На Листе3 отсортируйте по возрастанию столбец «Плотность».

7 На Листе4 отсортируйте по убыванию столбец «Скорость вращения по орбите».

8 На Листе5 используя фильтры получите список планет, у которых масса более 150

9 На Листе6 используя фильтры получите список планет, названия которых начинаются на буквы «М».

10 На Листе7 используя фильтры получите список планет, плотность которых более 3 и менее 5 г/см3.

11 На Листе8 используя фильтры получите список планет, у которых температура поверхности отрицательная.

12 На Листе9 используя фильтры получите список планет, у которых скорость вращения по орбите больше 10 км/с, но меньше 25 км/с.

13 На Листе10 используя фильтры получите список планет, у которых диаметр меньше 120000 км.

14 На Листе11 используя фильтры получите список планет, у которых ускорение свободного падения больше 9 м/с2.

15 Сохраните документ с именем Планеты.

Практическое занятие № 11

Тема: Способы объединения таблиц.

Цель: Освоить технологию создания БД, состоящую из нескольких таблиц, связать их, составить простые и сложные запросы, отсортировать записи и сформировать отчёт

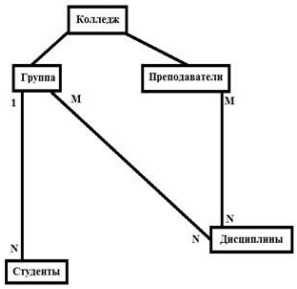
Технология выполнения задания:

Создадим файл базы данных с именем «Колледж_Фамилия» (Например: «Колледж_Иванова») в своей папке. Для этого выполните следующие действия:

1. Запустите СУБД MS Access.
2. В стартовом окне выберите Новая база данных (двойной щелчок мышью).
3. На вкладке Файл выберите Сохранить базу данных как -укажите, на каком диске, в какой папке требуется создать файл базы данных, введите имя файла и нажмите кнопку Сохранить.

1. **Создание таблиц базы данных**

Необходимо создать таблицы для базы данных «Колледж»:



Объекты: Группа, Студенты, Преподаватели, Дисциплина.

Создадим в режиме конструктора таблицу Группа следующей структуры:

	оле	Свойства	
	Г	Тип данных	Текстовый
		Размер поля	3
		Подпись	Группа
		Обязательное поле	Да
		Индексированное поле	Да (совпадения не допускаются)
	ОЛ	Тип данных	Числовой
		Размер поля	Байт
		Формат поля	Основной
		Число десятичных знаков	4
		Подпись	Количество студентов
		Обязательное поле	Нет
		Индексированное поле	Нет

1. Выберите Режим «Конструктор» рис.1 и щелкните по нему мышкой.
2. В появившемся окне «Сохранение» введите имя таблицы Группа рис. 2 и нажмите ОК.
3. В окне Конструктора введите имена полей, укажите тип данных, отредактируйте свойства.
4. Установите поле [НГ] в качестве ключевого поля. Для этого необходимо воспользоваться кнопкой Ключевое поле панели инструментов рис. 3
5. Сохраните таблицу.

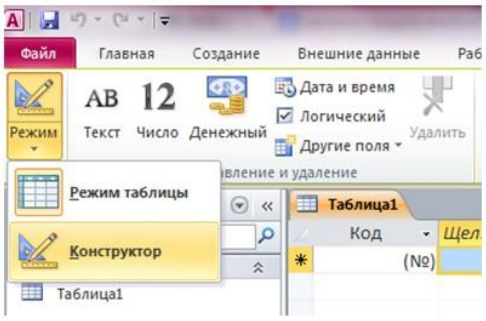
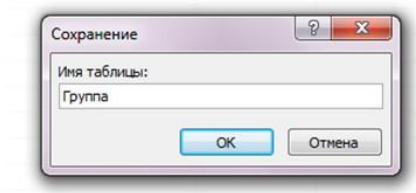


Рисунок 1



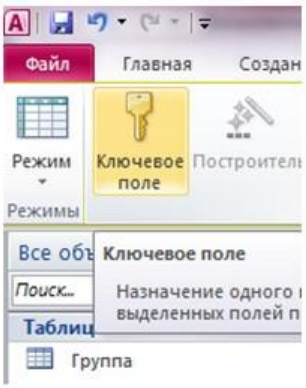


Рисунок 3

Создайте в режиме конструктора таблицу **Студенты** следующей структуры:

	оле	Свойства	
	Г	Тип данных	Текстовый
		Размер поля	3
		Подпись	Группа
		Обязательное поле	Да
		Индексированное поле	Да (совпадения допускаются)
	С	Тип данных	Текстовый
		Размер поля	2
		Подпись	Номер студента в группе
		Обязательное поле	Да
		Индексированное поле	Да (совпадения допускаются)
	ИО	Тип данных	Текстовый
		Размер поля	25
		Подпись	ФИО
		Обязательное поле	Да
	ОДР	Тип данных	Числовой
		Размер поля	Целое
		Формат поля	Основной
		Число десятичных знаков	4
		Подпись	Год рождения
		Обязательное поле	Нет

	ДР	Тип данных	Текстовый
		Размер поля	35
		Подпись	Адрес
		Обязательное поле	Нет

Замечание: В поле [НГ] нужно создать список значений из таблицы Группа, используя тип данных Мастер подстановок.

Установите поля [НГ] и [НС] в качестве ключевого поля.

Создайте в режиме конструктора таблицу **Преподаватели** следующей структуры:

	оле	Свойства	
	АБН	Тип данных	Текстовый
		Размер поля	4
		Подпись	Табельный номер
		Обязательное поле	Да
		Индексированное поле	Да (совпадения не допускаются)
	ИО	Тип данных	Текстовый
		Размер поля	25
		Подпись	ФИО
		Обязательное поле	Нет

Установите поле [ТАБН] в качестве ключевого поля.

II. Установка связей между таблицами

Установим связи между таблицами Группа и Студенты с обеспечением целостности данных в соответствии с логической моделью данных.

Для установки связей необходимо:

1. Закройте (если не закрыты) таблицы, между которыми устанавливаются связи.
2. На вкладке «Работа с базами данных» нажмите кнопку Схема данных рис. 4

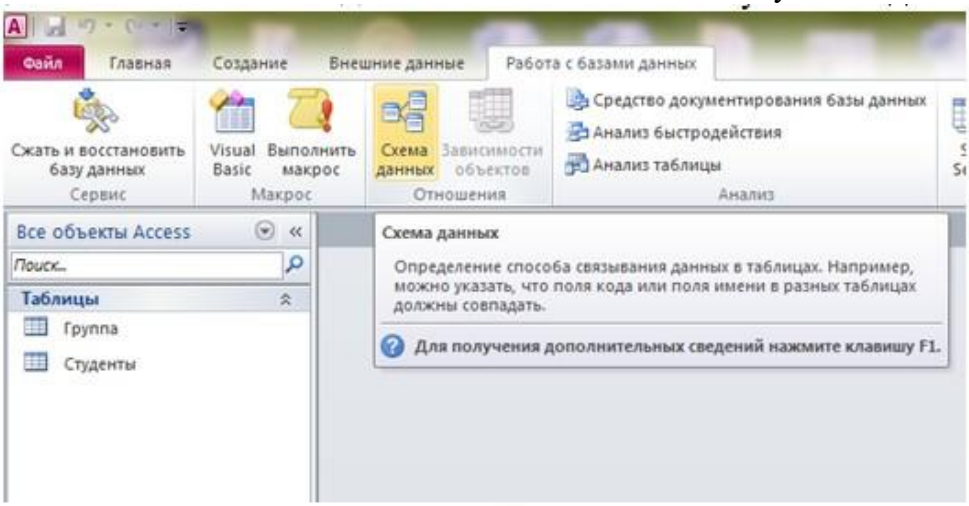


Рисунок 4

2. Появится окно Добавление таблицы с отображением имен таблиц. Выделяйте поочерёдно названия таблиц и нажимайте кнопку Добавить рис. 5

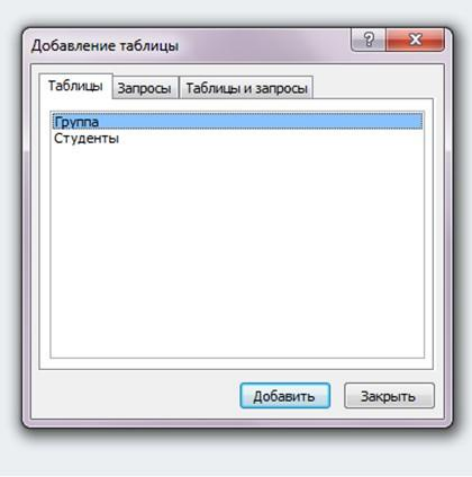


Рисунок 5

1. В окне Схема данных перетащите ключевое поле [НГ] из главной таблицы Группа на поле [НГ] подчиненной таблицы Студенты.
2. В окне Изменение связей установите флажок «Обеспечение целостности данных». Затем установите флажок «Каскадное обновление связанных полей» (изменение ключа в записи главной таблицы приведёт к автоматическому изменению значений внешнего ключа в подчинённых записях) и «Каскадное удаление связанных полей» (удаление записи из главной таблицы приведёт к автоматическому удалению всех связанных записей) Рис. 6.
3. Нажмите кнопку Создать.

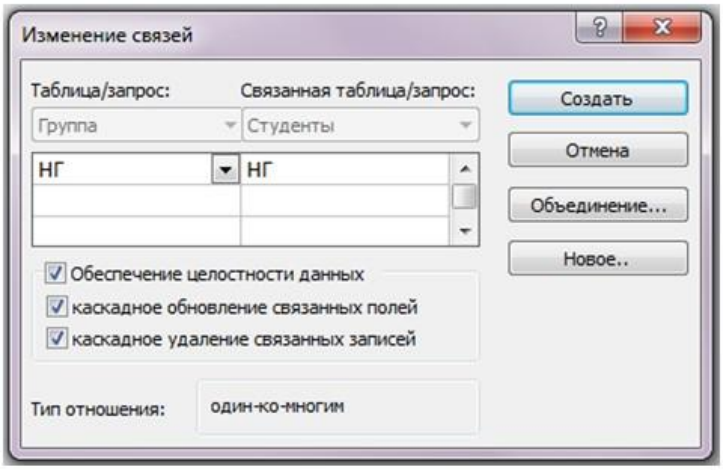


Рисунок 6

Примечание. Для удаления из окна схемы данных лишней таблицы, не связанной с другими таблицами, нужно выделить таблицу (один щелчок) и нажать DELETE. Если таблица связанна с другими, то вначале необходимо удалить связь. Для этого нужно выделить связь (один щелчок) и нажать DELETE.

После установки связей между таблицами окно Схема данных будет иметь вид, как на рис. 7.

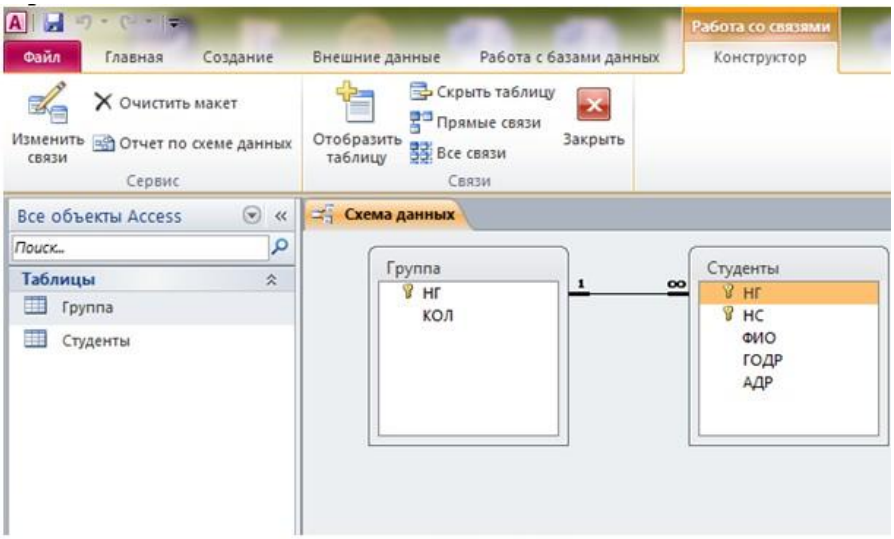


Рисунок 7

Практическое занятие № 12

Тема: Создание базы данных с помощью команд SQL. Редактирование, вставка и удаление данных средствами языка SQL

Цель: получить практический опыт создания таблиц базы данных с помощью команд SQL.

Краткие теоретические сведения

Новые элементы базы данных создаются с помощью предложения SQL CREATE. Чтобы создать таблицу, используйте команду CREATE TABLE, за которой введите поля и типы данных, предназначенные для добавления в таблицу. В качестве разделителей используйте запятые, а весь список заключите в круглые скобки.

Это одна из команд языка определения данных DDL. Команды DDL являются подмножеством команд SQL и используются для создания, изменения и удаления структур базы данных.

Синтаксис:

Создание таблицы	**CREATE TABLE** ***имя_таблицы***	
Определение столбцов	**(ПОЛЕ тип данных,**	**[DEFAULT значение NOT NULL]** *Значение по Нет* *Умолчанию отсутствующих*
Первичный ключ	**Primarykey (поле, …),**	
Определение ограничения	**Constraint имя ограничения**	
Определение внешнего ключа	**Foreign key (поле, …) references имя таблицы(поле, …));**	**On delete Cascade** **Set null** **No action** **Set default** **On update Cascade** **Set null** **No action** **Set default**

Используем следующие правила:

- имя таблицы указывается после ключевого слова CREATE TABLE (если имя состоит из нескольких слов, то его следует заключить в одинарные кавычки);
- далее в круглых скобках следуют имена столбцов таблицы (полей), после которых указывается тип данных, которому будет принадлежать поле;

• не обязательно: затем указывается может ли поле содержать пустые значения (NULL— может быть пустым или NOT NULL — обязательно для заполнения);

• одно из полей назначается первичным ключом (Primary key);

• поля отделяются запятыми.

SQL - Типы данных

Тип данных определяет, какое значение может содержать столбец: целочисленные данные, символьные данные, денежные данные, данные даты и времени, двоичные строки и т. д.

Типы данных SQL

Каждый столбец в таблице базы данных должен иметь имя и тип данных. Разработчик SQL должен решить, какой тип данных будет храниться внутри каждого столбца при создании таблицы. Тип данных является ориентиром для SQL, чтобы понять, какой тип данных ожидается внутри каждого столбца, а также определяет, как SQL будет взаимодействовать с сохраненными данными.

Типы данных SQL разделяются на три группы:

— строковые;

— с плавающей точкой (дробные числа);

— целые числа, дата и время.

Типы данных SQL строковые

Типы данных SQL	Описание
CHAR(size)	Строки фиксированной длиной (могут содержать буквы, цифры и специальные символы). Фиксированный размер указан в скобках. Можно записать до 255 символов
VARCHAR(size)	Может хранить не более 255 символов.
TINYTEXT	Может хранить не более 255 символов.
TEXT	Может хранить не более 65 535 символов.
BLOB	Может хранить не более 65 535 символов.
MEDIUMTEXT	Может хранить не более 16 777 215 символов.
MEDIUMBLOB	Может хранить не более 16 777 215 символов.
LONGTEXT	Может хранить не более 4 294 967 295 символов.
LONGBLOB	Может хранить не более 4 294 967 295 символов.
ENUM(x,y,z,etc.)	Позволяет вводить список допустимых значений. Можно ввести до 65535 значений в SQL Тип данных ENUM список. Если при вставке значения не будет присутствовать в списке ENUM, то мы получим пустое значение. Ввести возможные значения можно в таком формате: ENUM ('X', 'Y', 'Z')
SET	SQL Тип данных SET напоминает ENUM за исключением того, что SET может содержать до 64 значений.

Типы данных SQL с плавающей точкой (дробные числа) и целые числа

Типы данных SQL	Описание
TINYINT(size)	Может хранить числа от -128 до 127
SMALLINT(size)	Диапазон от -32 768 до 32 767
MEDIUMINT(size)	Диапазон от -8 388 608 до 8 388 607
INT(size)	Диапазон от -2 147 483 648 до 2 147 483 647
BIGINT(size)	Диапазон от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807
FLOAT(size,d)	Число с плавающей точкой небольшой точности.
DOUBLE(size,d)	Число с плавающей точкой двойной точности.
DECIMAL(size,d)	Дробное число, хранящееся в виде строки.

Типы данных SQL — Дата и время

Типы данных SQL	Описание
DATE()	Дата в формате ГГГГ-ММ-ДД
DATETIME()	Дата и время в формате ГГГГ-ММ-ДД ЧЧ:ММ:СС
TIMESTAMP()	Дата и время в формате timestamp. Однако при получении значения поля оно отображается не в формате timestamp, а в виде ГГГГ-ММ-ДД ЧЧ:ММ:СС
TIME()	Время в формате ЧЧ:ММ:СС
YEAR()	Год в двух значной или в четырехзначном формате.

Так же значение поля, в том случае если это не запрещено, может быть не указано, для этой цели используется ключевое слово NULL.

Пример:

```
CREATE TABLE GROUPS
(id NUMBER CONSTRAINTgroups_pk PRIMARY KEY,
num_gr CHAR(8) not null,
form CHAR(2) not null,
CONSTRAINT groups_uk UNIQUE (num_gr, form),
CHECK (form in('Д','З')));
```

Команда создает таблицу GROUPS. Столбцы
id (уникальный идентификатор),
num_gr (номер группы, например, 3091),
form (форма обучения, дневная (Д) или заочная (З)).

Пример:

```
CREATE TABLE STUDENTS
(id NUMBER CONSTRAINTstud_pk PRIMARY KEY,
surname CHAR(100)not null,
name CHAR(100)not null,
patron CHAR(100)not null,
gr_id number not null,
CONSTRAINT stud_gr_id_fk FOREIGN KEY (gr_id)
REFERENCESgroups( id) ON DELETE CASCADE);
```

Команда создает таблицу STUDENTS. Столбцы
id (уникальный идентификатор),
surname – фамилия,
name – имя,
patron – отчество студента,
gr_id – внешний ключ, ссылающийся на первичный ключ таблицы
GROUPS.

Каждый студент из таблицы STUDENTS учится в какой-то группе из таблицы GROUPS. При удалении группы, удаляется и информация о всех студентах, в ней обучающихся (это только для примера, необходимости в этом нет, скорее наоборот, группу с обучающимися в ней студентами удалять не следует).

Отметим, что таблицы создаются пустыми, а данные в них вносятся с помощью команды Insert.

Обновление таблиц: удаление и добавление полей

Обновление таблиц выполняется при помощи ключевых слов sql ALTER TABLE. Обновляя таблицу можно:

- удалять поля DROP COLUMN;
- добавлять поля ADD;

Пример:

В таблицу teachers добавить поле phone для номеров телефонов ALTER TABLE teachers ADD phone CHAR (20);

Пример:

Необходимо удалить поле phone из таблицы teachers

```
ALTER TABLE teachers DROP COLUMN phone
```

Команда SELECT ... INTO
Команда SELECT ... INTO позволяет создать новую таблицу на основе данных из других таблиц. Эта команда используется для архивирования данных, резервного копирования таблиц.

Синтаксис:

```
SELECT Поля INTO НоваяТаблица [INБазаДанных] FROM Таблицы;
```

- Поля - имена одного или нескольких полей, которые будут скопированы в новую таблицу.
- НоваяТаблица - Имя создаваемой таблицы БазаДанных - путь и имя внешней базы данных, в которой содержатся таблицы. Если таблицы находятся в текущей базе данных, то этот аргумент необязателен.
- Таблицы - имена таблиц, из которых выбираются записи.

Если имя новой таблицы совпадает с именем уже существующей, то будет сгенерирована ошибка.

Пример:

SELECT ID, Name, Email, Order INTO CopyOfOrders FROM Orders;

Задания

1 Изучить теоретические сведения.

2 В соответствии с вариантом задания создать таблицы базы данных с помощью команд SQL.

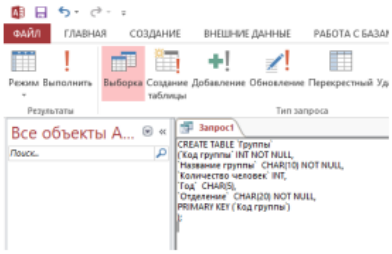
3 В соответствии с вариантом задания выполнить ввод данных в таблицы

Порядок выполнения работы

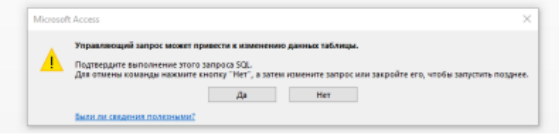
1 Создадим новую таблицу базы данных. Чтобы не писать команды, воспользуемся кодом, сгенерированным в DBDesigner

```
CREATE TABLE `Группы`
(`Код группы` SMALLINT NOT NULL,
`Название группы` CHAR(10) NOT NULL,
`Количество человек` INT,
`Год` CHAR(5),
`Отделение` CHAR(20) NOT NULL,
PRIMARY KEY (`Код группы`)
);
```

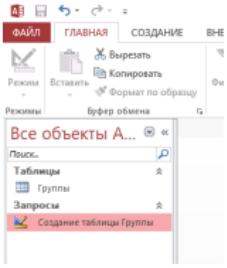
В меню Создание выбираем Конструктор запросов, закрываем окно добавления таблиц, переходим в режим SQL и вставляем команды.



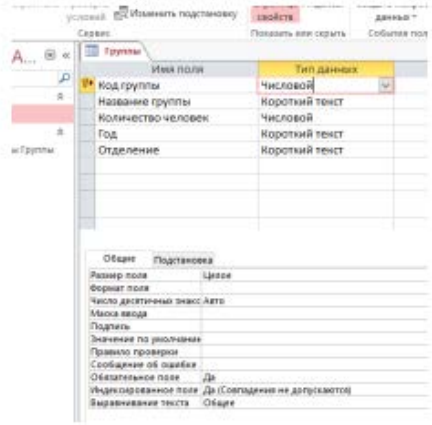
Сохраняем запрос и запускаем на выполнение, подтверждаем выполнение запроса



В результате будет создана таблица



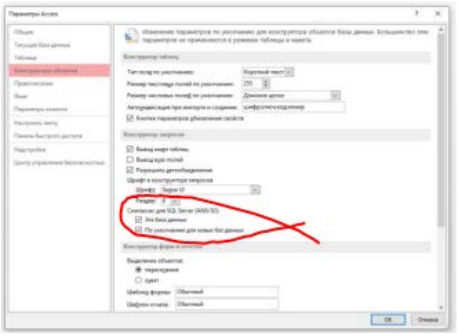
Если просмотрим полученную таблицу в режиме конструктора, то типы и характеристики полей соответствуют требованиям:



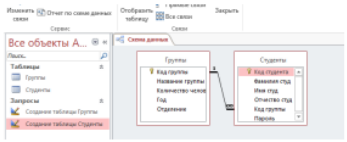
2 Создадим таблицу с внешними ключами, для этого добавим к списку полей команду FOREIGN KEY (`Код группы`) REFERENCES
`Группы` (`Код группы`)

```
CREATE TABLE `Студенты1` (
`Код студента` SMALLINT NOT NULL,
`Фамилия студ` CHAR(20) NOT NULL,
`Имя студ` CHAR(15) NOT NULL,
`Отчество студ` CHAR(20) NOT NULL,
`Код группы` SMALLINT NOT NULL,
`Пароль` CHAR(15) NOT NULL,
PRIMARY KEY (`Код студента`),
FOREIGN KEY (`Код группы`) REFERENCES `Группы` (`Код группы`)
ON DELETE CASCADE
ON UPDATE CASCADE
);
```

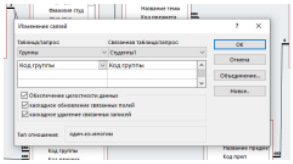
Но прежде, чем создавать такой запрос, необходимо внести изменения в параметры Access. Для этого в меню выбрать Файл – Параметры - Конструкторы объектов. В разделе Синтаксис для SQL Server выбрать Эта база данных и По умолчанию для новых баз данных



После создания таблицы Студенты в разделе Схема данных появятся связанные таблицы



Причем, если открыть окно связи (Правая кнопка мыши, изменить), то каскадное обновление и удаление записей активно):



3 Создадим таблицу базы данных, используя имеющуюся, причем, содержащую данные. Предположим, что часть данных из нее необходимо вынести в отдельную вспомогательную таблицу, тогда в основной таблицу эти данные должны быть заменены ссылкой на вспомогательную таблицу.

Определим действия, необходимые для получения результата:

а) скопировать поля из исходной таблицы во вспомогательную с помощью команды SELECT;

б) во вспомогательной таблице добавить ключевое поле код данных с помощью ALTER TABLE … ADD COLUMN, причем определить это поле как счетчик, тогда записи пронумеруются автоматически;

в) в случае необходимости, можно изменить тип первичного ключа на целый с помощью команды ALTER TABLE … ALTER COLUMN;

г) создать основную таблицу из исходной, но вместо полей, вынесенных во вспомогательную, перенести из вспомогательной ключевое поле, а чтобы в основную таблицу попали нужные значения, к команде добавить условия (SELECT …INTO …FROM …WHERE);

д) после того, как обе таблицы созданы, останется определить связи, т.е. добавить внешние ключи с помощью команды ALTER TABLE … ADD

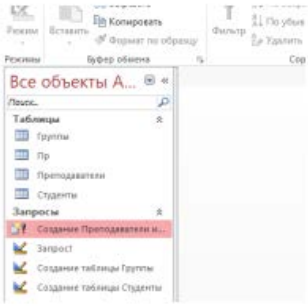
CONSTRAINT.

Предположим, имеется таблица Пр, хранящая данные о предметах и преподавателях.

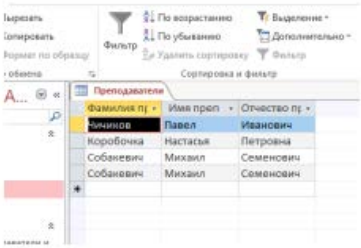
```
CREATE TABLE `Пр`
(`Код предмета` SMALLINT NOT NULL,
`Название предмета` CHAR(100) NOT NULL,
`Фамилия преп` CHAR(20) NOT NULL,
`Имя преп` CHAR(15) NOT NULL,
`Отчество преп` CHAR(20) NOT NULL,
`Семестр` SMALLINT NOT NULL,
PRIMARY KEY (`Код предмета`)
);
```

Необходимо выделить информацию о преподавателях в отдельную таблицу. Для этого можно использовать команду SELECT ... INTO. Она скопирует указанные поля вместе с данными

```
SELECT [Фамилия преп], [Имя преп], [Отчество преп] INTO
Преподаватели FROM Пр
```



Откроем появившуюся таблицу:



Можно заметить, что одна строка повторяется дважды. Чтобы исключить повторение строк, добавим в команду слово DISTINCT:

SELECT DISTINCT [Фамилия преп], [Имя преп], [Отчество преп] INTO
Преподаватели FROM Пр

Получим результат

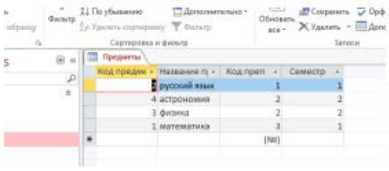
В режиме конструктора в таблице характеристики полей совпадают с Пр, но отсутствует первичный ключ и поле код преподавателя, необходимо внести изменения в таблицу Преподаватели. Добавим поле счетчик и назначим его первичным ключом:

ALTER TABLE Преподаватели ADD COLUMN [Код преп] COUNTER
PRIMARY KEY;

4 Осталось создать таблицу Предметы. В ней ФИО преподавателя из Пр заменим кодом преподавателя из таблицы Преподаватели. Таким образом для создания таблицы Предметы используем две таблицы Пр и Преподаватели, кроме того в поле Код преп необходимо поместить соответствующий код преподавателя. Поэтому в команду создания таблицы добавим условие (иначе вместо 4 в таблице появится 12 строк):

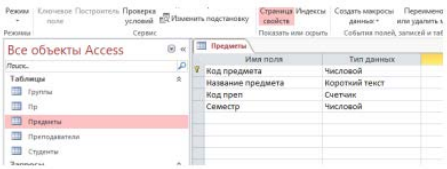
SELECT [Код предмета], [Название предмета], [Код преп], Семестр
INTO Предметы
FROM Пр, Преподаватели
WHERE Пр.[Фамилия преп] = Преподаватели.[Фамилия преп] AND Пр.[Имя
преп] = Преподаватели.[Имя преп];

Результат:

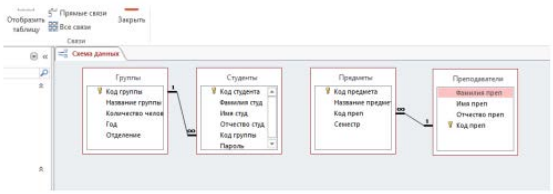


Аналогично предыдущей таблице, здесь отсутствуют ключи, как первичный, так и внешний. Внесем изменения в таблицу с помощью SQL запроса:

ALTER TABLE `Предметы` ADD CONSTRAINT PK_Predm PRIMARY KEY
(`Код предмета’);



ALTER TABLE `Предметы` ADD CONSTRAINT FK_Predm_Prep FOREIGN KEY (`Код преп`) REFERENCES `Преподаватели`(`Код преп`) ON DELETE CASCADE ON UPDATE CASCADE;



Если необходимо изменить тип поля в таблице, используем команду: ALTER TABLE `Предметы` ALTER COLUMN [Код преп] SMALLINT NOT NULL;

Практическое занятие № 13

Тема: Создание и использование запросов. Группировка и агрегирование данных

Цель: Получить практический опыт создания и использование запросов.

Краткие теоретические сведения

Команда SELECT

Команда SELECT - наиболее часто употребляемая команда из всех восьми. Она используется для выборки данных из базы данных. Её синтаксис:

SELECT [Предикат] Поля FROM Таблицы [IN БазаДанных] [WHERE ...] [GROUP BY ...] [HAVING ...] [ORDER BY ...];

Необязательные аргументы заключены в [].

› Предикат - одно из четырёх слов ALL, DISTINCT, DISTINCTROW, TOP. Если предикат не указан, то устанавливается ALL. Предикат ALL позволяет отобрать все записи. При использовании предиката DISTINCT, записи, которые содержат повторяющиеся значения в выбранных в запросе полях, исключаются. Предикат DISTINCTROW исключает из выборки записи, если повторяется вся запись, а не одно из полей. Предикат TOP позволяет отобрать определённое количество записей.

- Поля - имена одного или нескольких полей, выборка которых производится. Для выборки всех полей вместо имен полей можно поставить звёздочку [*].
- Таблицы - имена одной или нескольких таблиц, из которых производится выборка.
- БазаДанных - путь и имя внешней базы данных, в которой содержатся таблицы. Если таблицы находятся в текущей базе данных, то этот аргумент необязателен.

Также возможно использование предложения WHERE вместе с операторами BETWEEN, IN и LIKE.

Оператор BETWEEN позволяет отобрать записи, значение определённого поля которых находится в заданном диапазоне. Например:

SELECT * FROM Orders WHERE ID BETWEEN 10 AND 20; Здесь выбираются все записи, значение поля ID которых находится между 10 и 20.

Оператор IN позволяет отобрать записи, значение поля которых соответствует одному из значений, указанных в скобках.

SELECT * FROM Orders WHERE ID IN (10, 12, 30, 45); Здесь отбираются все записи, значение поля ID которых соответствует одному из значений 10, 12, 30, 45.

Используя предложение WHERE совместно с оператором LIKE, возможен отбор записей, значение одного из полей которых совпадает с маской. Оператор LIKE применим только к текстовым полям. В маске можно использовать следующие символы:

Символ Описание

Подчёркивание [_] Замещает один любой символ.

Процент [%] Замещает последовательность любого числа символов.

Например:

SELECT * FROM Orders WHERE Name LIKE 'Ba_я%' Здесь выбираются все записи, поле Name которых соответствует маске Ba_я%. Обратите внимание, что значения текстового типа в SQL-запросах указываются в кавычках.

Предложение GROUP BY позволяет объединять поля в запросе. Предложение ORDER BY позволяет упорядочивать выбираемые записи. При использовании совместно с предложением ключевого слова ASC можно определить возрастающий порядок, а используя DESC, определяется убывающий порядок.

SELECT * FROM Orders ORDER BY NameASC;

Также можно упорядочивать записи по нескольким полям. Сначала записи упорядочиваются по первому полю, если в нём есть записи, имеющие одинаковые значения, то они упорядочиваются по следующему указанному в предложении ORDER BY полю и т.д. Имена полей пишутся через запятую

[,].

SELECT * FROM Orders ORDER BY NameASC, EmailASC;

Вставка записей

Синтаксис:

INSERT INTO <имя_таблицы>[(<имя_столбца>,<имя_столбца>,...)]

VALUES (<значение>,<значение>,..)

Список столбцов в данной команде не является обязательным параметром. В этом случае должны быть указаны значения для всех полей таблицы в том порядке, как эти столбцы были перечислены в команде

CREATE TABLE

Команда INSERT INTO предназначена для добавления одной или нескольких записей в конец таблицы. Возможны 2 варианта использования этой команды. Первый вариант добавляет одну запись в таблицу, а второй вариант добавляет записи из одной таблицы в другую.

Синтаксис первого варианта:

INSERT INTO Таблица Назначения [(Поля)] VALUES (Значения);

Синтаксис второго варианта:

INSERT INTO Таблица Назначения [(Поля)] [IN База Данных] SELECT

[Таблица.] Поля FROM Таблица;

Таблица Назначения - таблица, в которую добавляются записи.

- Поля - названия полей.
- Таблица - имя таблицы, источника данных.
- База Данных - путь и имя внешней базы данных, в которой содержатся таблицы. Если таблицы находятся в текущей базе данных, то этот аргумент необязателен.
- Значения - значения полей добавляемой записи.

Все поля записи и соответствующие им значения должны быть определены, иначе им будут присвоены значения Null.

Если таблица, в которую добавляются записи, имеет ключевое поле, то в него должны добавляться уникальные, непустые значения. Иначе запись не будет добавлена.

Пример:

INSERT INTO Orders (ID, Name, Email, Order) VALUES (12, 'Вася

Пупкин', ' vasya@pupkin.ru', 'PentiumII 450 MHz');

Добавляется новая запись, в которой полям ID, Name, Email, Order соответствуют значения 12, 'Вася Пупкин', ' vasya@pupkin.ru', 'Pentium II 450

MHz'.

INSERT INTO Orders2001 (ID, Name, Email, Order) SELECT ID, Name,

Email, Order FROM Orders2000;

Этот запрос добавляет все записи из таблицы Orders2000 в таблицу Orders2001.

Модификация записей

Синтаксис:

UPDATE <имя_таблицы> SET <имя_столбца>=<значение>,... [WHERE <условие>]

Если задано ключевое слово WHERE и условие, то команда UPDATE применяется только к тем записям, для которых оно выполняется. Если условие не задано, UPDATE применяется ко всем записям.

Команду UPDATE удобно использовать, если изменяется сразу большое число записей или если изменяемые записи находятся в разных таблицах. Новые значения указываются через запятую для каждого поля.

Использование предложения WHERE аналогично его использованию в команде SELECT.

Пример:

UPDATE Buyers SET Order='Ничего' WHERE ID=7;

Устанавливаем значение поля покупки 'Ничего' у покупателя, номер которого равен 7.

UPDATE Заказы SET СуммаЗаказа = СуммаЗаказа * 1.2,

СтоимостьДоставки = СтоимостьДоставки * 1.1 WHEREСтрана='США';

Этот запрос немного сложнее. Он повышает сумму заказа на 20% и стоимость доставки на 10% для покупателей из США.

Удаление записей

Синтаксис:

DELETE FROM<имя_таблицы>[WHERE<условие>]

Удаляются все записи, удовлетворяющие указанному условию. Если ключевое слово WHERE и условие отсутствуют, из таблицы удаляются все записи.

Аргумент команды DELETE можно не указывать, поскольку он фактически дублируется в предложении FROM.

Пример:

DELETE FROM Buyers WHERE ID=8;

Этот запрос удаляет из таблицы Buyers запись, в которой ID равно 8. Для удаления не всей записи, а только ее поля, следует воспользоваться запросом на изменение записи (команда UPDATE) и поменять значения нужных полей на Null.

Перекрестный запрос

Перекрестный запрос это соединение нескольких таблиц в одной. Одна из таблиц используется для заголовков строк, вторая – для заголовков столбцов, а данные для создаваемой таблицы могут храниться в третьей. Значит, необходимо команда соединения таблиц JOIN. Эта команда соединяет две таблицы, а если необходимо соединить три, воспользуемся вложенными запросами. Кроме того, ключевым словом SQL-оператора перекрестного запроса, задающим его тип, является слово TRANSFORM (преобразовать). Это подразумевает, что значения одного из столбцов (полей) выборки, будут преобразованы в названия столбцов итоговой выборки.

Результаты перекрестного запроса группируются по двум наборам данных, один из которых расположен в левом столбце (столбцах) таблицы, а второй — в верхней строке. В остальном пространстве таблицы отображаются результаты статистических расчетов (Sum, Count и т.д.), выполненных над данными трансформированного поля. И наконец, PIVOT – это оператор, который поворачивает результирующий набор данных, т.е. происходит транспонирование таблицы, при этом используются агрегатные функции, и данные соответственно группируются. Другими словами, значения, которые расположены по вертикали, мы выстраиваем по горизонтали. Но оператор PIVOT можно использовать, только начиная с 2005 sql сервера.

У данного оператора очень специфический, непривычный и некоторые даже скажут сложный синтаксис, как для написания, так и для простого понимания.

Синтаксис оператора PIVOT

SELECT столбец для группировки, [значения по горизонтали],…

FROM таблица или подзапрос
PIVOT(агрегатная функция
FOR столбец, содержащий значения, которые станут именами столбцов
IN ([значения по горизонтали],…)
)AS псевдоним таблицы (обязательно) в случае необходимости ORDER BY;

Инструкция TRANSFORM используется для создания перекрестного запроса. Данные, представленные с помощью перекрестного запроса, изображаются в более компактном виде, чем с помощью запроса-выборки.

Синтаксис:
TRANSFORM Функция
SELECT …;
PIVOT поле;

Функция - групповая функция SQL, обрабатывающая данные ячейки таблицы

Поле - поле или выражение, значения из которого становятся заголовками столбцов.

Запрос в режиме таблицы имеет столько столбцов, сколько различных значений принимает поле. Например, если поле выдает названия месяцев, то получится до 12 столбцов, заголовки которых упорядочены по возрастанию (Август, Апрель…Январь). После аргумента поле можно поместить предложение IN(список_значений). Фиксированные значения в списке_значений разделяются запятыми. При наличии предложения IN каждое значение поля сравнивается со значениями в списке_значений. При совпадении в соответствующем столбце выводится результат вычисления функции. Фиксированные заголовки, которым не соответствуют реальные данные, можно использовать для создания дополнительных столбцов.

Использование предложения PIVOT эквивалентно определению свойства “Заголовки столбцов” в бланке свойств конструктора запросов.

Пример

Результат выполнения может иметь, например, такой вид:

Столбцы “Группа” и “Всего” сформированы инструкцией SELECT, включающей в себя предложения WHERE, FROM, GROUP BY и ORDER BY.

Заголовки остальных столбцов определены предложением PIVOT, а значения в ячейках этих столбцов формирует функция Count из предложения
TRANSFORM.

Задания

1 Изучить теоретические сведения.

2 В соответствии с вариантом задания организовать выборки
данных с помощью запросов:

- на добавление данных с помощью запроса и из другой таблицы;
- на обновление таблиц;
- перекрестный;
- на удаление.

Порядок выполнения работы

1 Запрос на добавление данных Ввод данных в таблицу производится с помощью команды INSERT INTO непосредственным вводом данных (в Access для каждой строки таблицы отдельный запрос):

INSERT INTO Преподаватели ([Фамилия преп],[Имя преп],[Отчество преп],
[Код преп])
VALUES (‘Хлестаков’, ‘Иван’, ‘Александрович’,4),
(‘Уховертов’, ‘Степан’, ‘Ильич’,5);

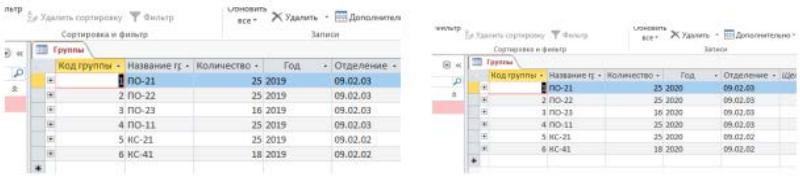
Или из другой таблицы (все записи из таблицы Преподаватели1 будут перенесены в таблицу Преподаватели):

INSERT INTO Преподаватели ([Фамилия преп],[Имя преп],[Отчество преп], [Код преп])

SELECT * FROM Преподаватели1;

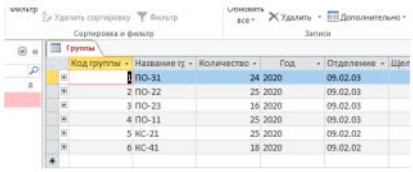
2 Запрос на обновление выполняется с использованием команды UPDATE. Пример замены данных в одном поле:

UPDATE Группы SET Год=2020;

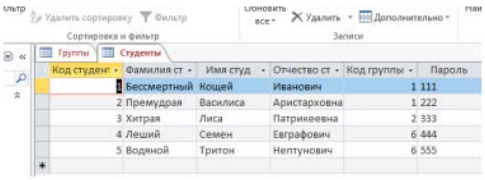


Пример обновления нескольких полей с условием:

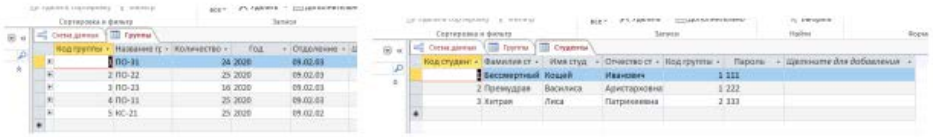
UPDATE Группы SET [Количество человек]=24, [Название группы]=”ПО-31” WHERE [Название группы]=”ПО-21”



3 Запрос на удаление выполняется с использованием команды DELETE FROM. Пример удаления данных о группе КС-41 в 2019 году. В 2020 эта группа уже окончил обучение, поэтому ее надо удалить: DELETE FROM Группы WHERE [Название группы]=”КС-41”; Но таблица Группы связана с таблицей Студенты, в которой имеются записи студентов группы КС-41:



Если при организации связи таблиц не установлено каскадное обновление полей и каскадное удаление записей, то будет выдано сообщение об ошибке, иначе будут удалены записи в двух таблицах



4 Запрос с параметром выполняется с использованием команд SELECT. В примере выборка фамилий студентов из заданной с помощью параметра группы. Так как название группы в таблице Группы, а ФИО студентов в таблице Студенты, то используем условие соответствия кода группы в двух таблицах вместе с условием проверки параметра:

```
SELECT Группы.[Название группы], [Фамилия студ], [Имя студ], [Отчество
студ] FROM Группы, Студенты
WHERE Группы.[Название группы] = [Задайте группу] AND Группы.[Код
группы]=Студенты.[Код группы];
```

Или

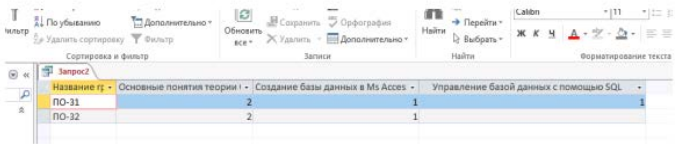
```
SELECT Группы.[Название группы], [Фамилия студ], [Имя студ], [Отчество студ]
FROM Группы
JOIN Студенты
ON Группы.[Код группы]=Студенты.[Код группы]
WHERE Группы.[Название группы] = [Задайте группу];
```

В результате:

5 Перекрестный запрос, вычисляющий количество тестов в базе данных тестовой оболочки, начнем со слова TRANSFORM, далее следует агрегатная функция Count, вычисляющая количество тестов, запланированных по каждой теме для каждой группы, далее объединим три таблицы в операторе SELECT с помощью команды INNER JOIN с вложенным запросом. И наконец, в PIVOT укажем поле, которое станет заголовком столбцов таблицы перекрестного запроса.

```
TRANSFORM Count([Проведение теста].[Код проведения]) AS Всего
SELECT Группы.[Название группы]
FROM Темы INNER JOIN (Группы INNER JOIN [Проведение теста] ON
Группы.[Код группы] = [Проведение теста].[Код группы]) ON Темы.[Код
темы] = [Проведение теста].[Код темы]
GROUP BY Группы.[Название группы]
PIVOT Темы.[Название темы];
```

И в результате:



Практическое занятие № 14

Тема: Коррелированные вложенные запросы

Цель: Научиться создавать запросы с вложенными подзапросами в режиме SQL.

Теоретическая часть:

Вложенные подзапросы

Вложенный подзапрос - это подзапрос, заключенный в круглые скобки и вложенный WHERE (HAVING) фразу предложения SELECT или других предложений, использующих WHERE фразу.

Вложенный подзапрос может содержать в своей WHERE (HAVING) фразе другой вложенный подзапрос и т.д.

Вложенный подзапрос создается для того, чтобы при отборе строк таблицы, сформированной основным запросом, можно было использовать данные из других таблиц.

Ограничения на подзапросы

На подзапросы накладываются следующие ограничения:

- Подзапросы нельзя использовать в списках предложений order by, group by и compute by.
- Подзапрос не может содержать предложения for browse или union.

Список выбора внутреннего подзапроса, которому предшествует операция сравнения, может содержать только одно выражение или название столбца, и подзапрос должен возвращать единственный результат. При этом тип данных столбца, указанного в конструкции where внешнего оператора, должен быть совместим с типом данных в столбце, указанным в списке выбора подзапроса (правила здесь такие же как и при соединении).

В подзапросах не допускаются текстовые (text) и графические (image) данные.

Подзапросы не могут обрабатывать свои результаты внутренним образом, т.е. подзапрос не может содержать конструкций order by, compute, или ключевого слова into.

Коррелирующиеся (повторяющиеся) подзапросы не допускаются в конструкции select обновляемого курсора, определенного с помощью declare cursor (определить курсор).

Количество вложенных уровней для подзапросов не должно превышать 16.

Максимальное число подзапросов на каждой стороне объединения не больше 16.

Типы подзапросов

Существуют два основных типа подзапросов:

Подзапросы, которым предшествует немодифицированная операция сравнения и которые возвращают единственное значение, называются подзапросами- выражениям (скалярными подзапросами).

Подзапросы, которые возвращают список значений и которым предшествует ключевое слово in (принадлежит) или операция сравнения, модифицированная кванторами any (некоторый) или all (все), а также подзапросы, проверяющие существование с помощью квантора exists (существует), называются квантифицированными предикатными подзапросами.

Подзапросы любого из этих типов могут быть либо коррелированными (повторяющимися), либо некоррелированными.

Некоррелированный подзапрос может вычисляться как независимый запрос. Иначе говоря, результаты подзапроса подставляются в основной оператор (или внешний запрос). Это не значит, что SQL-сервер именно так выполняет операторы с подзапросами. Некорреляционные подзапросы могут быть заменены соединением и будут выполняться как соединения SQL-сервером.

Коррелированные подзапросы не могут выполняться как независимые запросы, поскольку они могут обращаться к данным, находящихся в столбцах таблицы, указанной в списке from внешнего запроса. Запросы с коррелированными вложенными подзапросами обрабатываются системой в обратном порядке. Сначала выбирается первая строка рабочей таблицы, сформированной основным запросом, и из нее выбираются значения тех столбцов, которые используются во вложенном подзапросе (вложенных подзапросах). Если эти значения удовлетворяют условиям вложенного подзапроса, то выбранная строка включается в результат. Затем выбирается вторая строка и т.д., пока в результат не будут включены все строки, удовлетворяющие вложенному подзапросу (последовательности вложенных подзапросов).

ЗАДАНИЕ:

Создание инструкции SQL

Для создания инструкции SQL выберите вкладку Создание Þ Конструктор запросов Þ закройте окно добавления таблиц Þ на вкладке Конструктор/ Группа Результаты/SQL режим.

Использование одной и той же таблицы во внешнем и вложенном подзапросе

Задание1: Выдать номера издательств, без повторений, которые издают книги той же предметной области, что и издательство 1. Сохранить как П1.

```
SELECT DISTINCT [Код издательства]
FROM Книги
WHERE [Предметная область] IN
(SELECT [Предметная область]
FROM Книги
WHERE [Код издательства]=1);
```

Задание 2: Изменить запрос, созданный в задании 1, так, чтобы помимо номеров издательств выводилось и название издательства. Сохранить как П2.

```
SELECT DISTINCT Издательства.[Код издательства], Издательства.Название
FROM Издательства, Книги
WHERE (Издательства.[Код издательства]=[Книги].[Код издательства]) AND
(Книги.[Предметная область]) In
(SELECT [Предметная область]
FROM Книги
WHERE [Код издательства]=1);
```

Вложенный подзапрос с оператором сравнения, отличным от IN

Задание 3. Выдать названия издательств, находящихся в том же городе, что и издатель с номером 1. Сохранить как П3.

```
SELECT Название
FROM Издательства
WHERE Город =
(SELECT Город
FROM Издательства
WHERE [Код издательства] =1);
```

В подобных запросах можно использовать и другие операторы сравнения (<>, <=, <, >= или >), однако, если вложенный подзапрос возвращает более одного значения и не используется оператор IN, будет возникать ошибка.

Коррелированные вложенные подзапросы.

Задание 4. Выдать название и город издательства для книг по информатике. Сохранить как П4.

```
SELECT Издательства.Название, Издательства.Город
FROM Издательства
WHERE "Информатика" In
(SELECT [Предметная область]
FROM Книги
WHERE Книги![Код издательства] = Издательства![Код издательства]);
```

Такой подзапрос отличается от рассмотренного в заданиях 1,2 тем, что вложенный подзапрос не может быть обработан прежде, чем будет обрабатываться внешний подзапрос. Это связано с тем, что вложенный подзапрос зависит от значения Издательства![Код издательства], а оно изменяется по мере того, как система проверяет различные строки таблицы Издательства. Следовательно, с концептуальной точки зрения обработка осуществляется следующим образом:

Система проверяет первую строку таблицы Издательства. Предположим, что это строка издательства с номером 1. Тогда значение Издательства![Код издательства]=1 и система обрабатывает внутренний запрос

```
(SELECT [Предметная область]
FROM Книги
WHERE Книги![Код издательства] =1;
```

В результате получим множество названий предметных областей книг издательства 1.

Система может завершить обработку для издательства с номером 1.

Выборка значений Название и Город для Код издательства=1 будет проведена тогда и только тогда, когда в списке предметных областей будет значение «Информатика».

Далее система будет повторять аналогичную обработку для следующего издательства и т.д. до тех пор, пока не будут рассмотрены все строки таблицы Издательства.

Подобные подзапросы называются коррелированными, так как их результат зависит от значений, определенных во внешнем подзапросе. Обработка коррелированного подзапроса, должна повторяться для каждого значения извлекаемого из внешнего подзапроса, а не выполняться раз и навсегда.

Рассмотрим пример использования одной и той же таблицы во внешнем подзапросе и коррелированном вложенном подзапросе

Задание 5. Выдать названия видов печатной продукции, которые издаются только одним издательством. Сохранить как П5.

```
SELECT DISTINCT X.[Вид печатной продукции], X.[Код издательства]
FROM Книги AS X
WHERE X.[Вид печатной продукции] Not In
(SELECT Y.[Вид печатной продукции]
FROM Книги Y
WHERE Y.[Код издательства]<> X.[Код издательства]);
```

Действие этого запроса производится следующим образом:

Поочередно для каждой строки таблицы Книги, обозначенной как X, выделяется вид печатной продукции

Если и только если это значение не входит в некоторую строку, строки Y той же таблицы, а значение столбца код издательства в строке Y не равно его значению в строке X, то значение будет выведено.

Квантор EXISTS (существует) - понятие, заимствованное из формальной логики. В языке SQL предикат с квантором существования представляется выражением EXISTS (SELECT * FROM ...).

Такое выражение считается истинным только тогда, когда результат вычисления "SELECT * FROM ..." является непустым множеством, т.е. когда существует какая-либо запись в таблице, указанной во фразе FROM подзапроса, которая удовлетворяет условию WHERE подзапроса. (Практически этот подзапрос всегда будет коррелированным множеством.)

Задание 6. Выдать названия издательств, издающих литературу по экономике. Сохранить как П6.

```
SELECT Название
FROM Издательства
WHERE EXISTS
(SELECT *
FROM Книги
WHERE [Предметная область]=”Экономика”
AND Книги![Код издательства] = Издательства![Код издательства]);
```

Система последовательно выбирает строки таблицы Издательства, выделяет из них значения столбцов Название и Код издательства, а затем проверяет, является ли истинным условие существования, т.е. существует ли в таблице Книги хотя бы одна строка со значением Предметная область=”Экономика”и значением Код издательства, равным значению Код издательства из таблицы Издательства. Если условие выполняется, то полученное значение столбца Название включается в результат.

Задание 7. Изменить запрос задания 6, так, чтобы выводилось еще и название книги. Сохранить как П7.

Этот первый пример показывает иной способ формулировки запроса для задачи, решаемой и другими путями (с помощью оператора IN или соединения).

EXISTS представляет собой одну из наиболее важных возможностей SQL. Фактически любой запрос, который выражается через IN, может быть альтернативным образом сформулирован также с помощью EXISTS. Однако обратное высказывание несправедливо.

Задание 8. Выдать название и город издательств, не издающих литературу по информатике. Сохранить как П8.

```
SELECT Название, Город
FROM Издательства
WHERE NOT EXISTS
(SELECT *
FROM Книги
WHERE Книги![Код издательства] = Издательства![Код издательства] AND
[Предметная область]=”Экономика”);
```

Функции в подзапросе

Задание 9. Выдать список поставщиков литературы по информатике, которые поставляют эту литературу за минимальную цену. Сохранить как П9.

```
SELECT [Название книги],[Стоимость книги], Название
FROM Книги, Издательства
WHERE Книги![Код издательства] = Издательства![Код_издательства]
AND [Стоимость книги]=
```

(SELECT Min([Стоимость книги])
FROM Книги
GROUP BY [Предметная область]
HAVING Книги.[Предметная область]="Информатика");

Естественно, что это коррелированный подзапрос: здесь сначала определяется минимальная цена книги по информатике, и только затем выясняется его издатель.

Практическое занятие № 15

Тема: Создание в запросах вычисляемых полей. Использование условий

Цель: закрепить навыки по редактированию таблиц; познакомиться с основными видами запросов; научиться создавать запросы на выборку различными способами.

Задание 1. Откройте учебную базу данных, изготовленную на прошлом занятии.

Порядок работы:

Вызовите программу Access. Для этого дважды щелкните по пиктограмме Microsoft Access. Откроется окно системы управления базами данных, в котором появится меню.

Включите мышкой переключатель Открыть базу данных, выделите из списка баз данных, расположенного ниже переключателя, имя вашей базы и щелкните по кнопке ОК.

На экране возникнет диалоговое окно с основными элементами базы данных. В базе данных должны быть три таблицы: Список, Личные данные. Группы. В случае их отсутствия откройте базу данных учителя Пуск → Сетевое окружение → Практическая работа №5 и сохраните её на рабочий стол под своим именем. Для этого нажмите на кнопку Office → Сохранить как… → введите свою фамилию и сохраните на Рабочий стол.

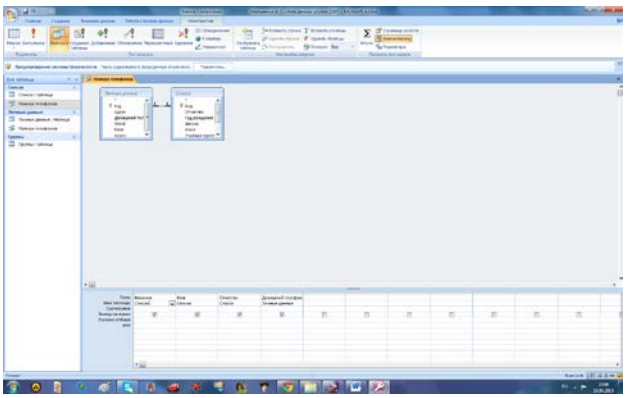
Задание 2. Создайте запрос на выборку с именем Номера телефонов.

Порядок работы:

Выберите закладку Создание, если находитесь в другом диалоговом окне.

Нажмите на кнопку Конструктор запросов .

Добавьте нужные таблицы (Личные данные и Список), выбирая их и щелкая по кнопке Добавить. Так как таблицы связаны, то между ними появится линия, обозначенная цифрами 1 ("один-к-одному").



Закончите выбор, щелкнув по кнопке Закрыть. Появляется возможность выбора полей из разных таблиц.

Выберите поля Фамилия, Имя и Отчество из таблицы Список и Домашний телефон - из таблицы Личные данные. Для этого достаточно сделать двойной щелчок мышкой по имени поля. Второй вариант - перетащить мышкой название поля в клетки запроса.

Сохраните запрос, щелкнув по кнопке Сохранить. Введите имя запроса Номера телефонов и щелкните по кнопке ОК.

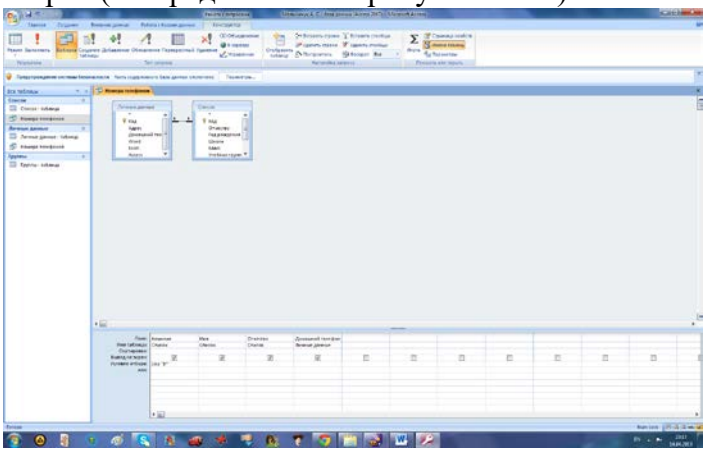
Щелкните по кнопке для представления запроса. Это самый простой вид запроса на выборку. В результате вы получаете новую таблицу с другим набором полей. Перейдите в режим Конструктор.

Замечание.

Условие отбора можно включать аналогично включению фильтра. Например, телефонная книга для всех учащихся, фамилии которых начинаются на букву "В", может быть получена с помощью включения условия Like "В*".

Внимание! Букву В вводите на русской раскладке!

Получите этот запрос (он представлен на рисунке ниже).



Щелкните по кнопке для представления запроса.

Измените имя запроса, выбрав в меню пункт Файл - Сохранить как/Экспорт.

В появившемся диалоговом окне наберите новое имя запроса: Выборка по В и нажмите ОК. Теперь в меню базы данных в окне Запросы будет показано два запроса.

Самостоятельное задание.

Составьте запрос на адреса только девочек, имя которых "Анна". Сохраните запрос с именем Анна.

Составьте запрос на телефоны учащихся, отчество которых начинается на букву "А". Сохраните запрос с именем Выборка по А.

Задание 3. Составьте запрос с использованием логических операций в условии отбора.

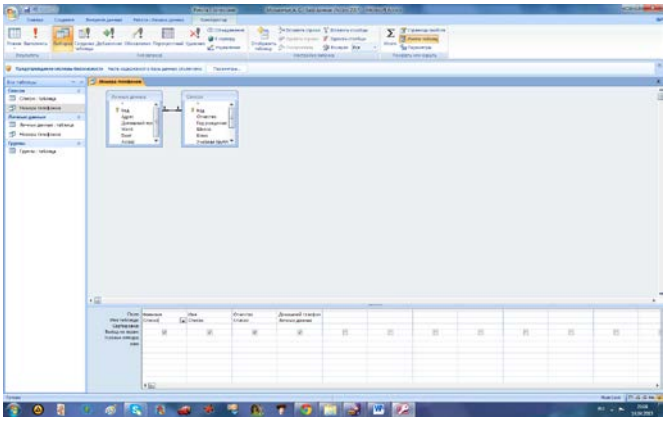
Предположим, что вам нужно составить ведомость для выплаты стипендии всем учащимся, которые учатся без троек. Для этого нужно выбрать записи, в которых оценки по предметам 4 ИЛИ 5.

Порядок работы:

Выберите закладку Создание, если находитесь в другом диалоговом окне.

Нажмите на кнопку Конструктор запросов .

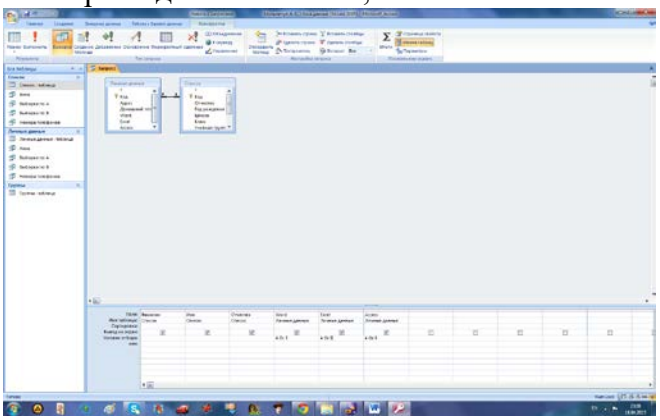
Добавьте нужные таблицы (Личные данные и Список), выбирая их и щелкая по кнопке Добавить. Так как таблицы связаны, то между ними появится линия, обозначенная цифрами 1 ("один-к-одному").



Закончите выбор, щелкнув по кнопке Закрыть. Появляется возможность выбора полей из разных таблиц.

Выберите поля Фамилия, Имя и Отчество из таблицы Список и поля Word, Excel, Access - из таблицы Личные данные. Для этого достаточно сделать двойной щелчок мышкой по имени поля. Второй вариант - перетащить мышкой название поля в клетки запроса.

В строке Условие отбора под полями Word, Excel и Access поставье 4 Or 5.



Щелкните по кнопке для представления запроса.

Сохраните запрос с именем Успеваемость 1, щелкнув по кнопке Сохранить. Теперь в меню базы данных в окне Запросы будет показано три запроса.

Самостоятельное задание.

Составьте запрос на учеников группы 101, у которых оценка по курсу "Освоение базы Access" 4 или 5; сохраните запрос с именем Успеваемость 2.

Составьте запрос на учеников групп 102 и 103, которые имеют оценку по курсу " Освоение программы Word" и "Освоение программы Excel" 4 или 5; сохраните запрос с именем Успеваемость 3.

Задание 4. Составьте запрос на выборку всех записей, кроме тех, в которых указана фамилия Баранова с использованием Построителя выражений.

Порядок работы:

1. Выделите запрос Номера телефонов.
2. Щелкните по кнопке Конструктор.
3. Удалите поле Домашний телефон.
4. Добавьте поле Адрес.
5. Сохраните запрос с именем Адрес, выполнив команду Сохранить как ....
6. Поставьте курсор в ячейку Условие отбора в столбце Фамилия.
7. Удалите надпись в этой ячейке.
8. Щелкните по кнопке - Построитель. Появится окно, в котором можно строить сложные запросы.

Порядок работы:

Щелкните по кнопке Not, это слово появится в верхнем поле. Фамилию Баранова в кавычках наберите вручную.

Щелкните по кнопке ОК. В поле Условие отбора появится данное выражение.

Щелкните по кнопке для представления запроса.

Закройте запрос, сохранив его с именем не_Баранова, выполнив команду Файл - Сохранить как ....

Задание 5. Создайте вычисляемые поля.

Порядок работы:

Выберите закладку Создание, если находитесь в другом диалоговом окне.

Нажмите на кнопку Конструктор запросов .

Добавьте нужные таблицы (Личные данные и Список), выбирая их и щелкая по кнопке Добавить.

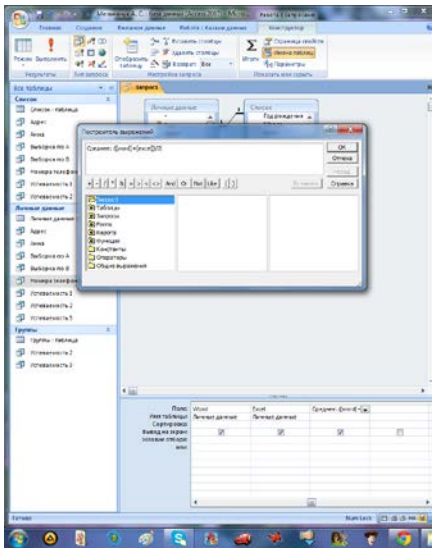
Закончите выбор, щелкнув по кнопке Закрыть.

Выберите поля Фамилия и Имя из таблицы Список и поля Word и Excel - из таблицы Личные данные.

Поставьте курсор на клетку правее Excel (на линии Поле).

Щелкните по кнопке - Построитель.

В появившемся окне напечатайте вручную выражение, представленное на рисунке ниже, и щелкните по кнопке ОК.



Это выражение подставится в новое поле. Нажмите клавишу [Enter]

Сохраните запрос с именем Среднее

Щелкните по кнопке для представления запроса. Новое поле будет иметь имя Среднее

Закройте запрос.

Предъявите преподавателю:

Запросы:

Номера телефонов.

Выборка по В,

Анна,

Выборка по А,

Успеваемость 1,

Успеваемость 2,

Успеваемость 3,

Не Баранова,

Среднее.

Задание 6. Завершите работу с программой Access.

Порядок работы:

Выполните команду кнопка Office - Выход.

Если вы производили редактирование в базе данных, появится вопрос о сохранении изменений. Ответьте на него положительно.

Практическое занятие № 16

Тема: Управление доступом к объектам базы данных

Задание

Создать защищенную паролем базу данных «Организация», содержащую данные о начислениях выплат сотрудникам.

Технология выполнения в среде MS Access 2007

Задание 1. Создать базу данных MS Access с данными о сотрудниках организации, импортировав данные из книги MS Excel.

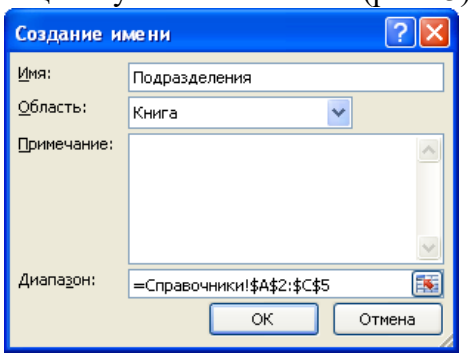
Создание таблиц и запросов, импорт данных

Создать на диске рабочую папку, например, D:\Petrov.

Подготовить книгу «Организация_защита_снята» для экспорта данных в MS Access:

Открыть книгу «Организация_защита_снята» в приложении MS Excel, проверить, что пароль на открытие файла не задан, снята защита с листа «Справочники» (можно изменять данные на листе).

На листе «Справочники» выделить диапазон ячеек A2:C5, содержащий данные о подразделениях. На вкладке Формулы щелкнуть команду Присвоить имя . В окне Создание имени ввести имя Подразделения и щелкнуть на кнопке ОК (рис.13).



В строке адреса рядом со строкой формул при выделении диапазона ячеек A2:C5 должно отображаться «Подразделения».

На листе «Справочники» выделить диапазон ячеек A8:D21, содержащий данные о сотрудниках и присвоить ему имя Сотрудники.

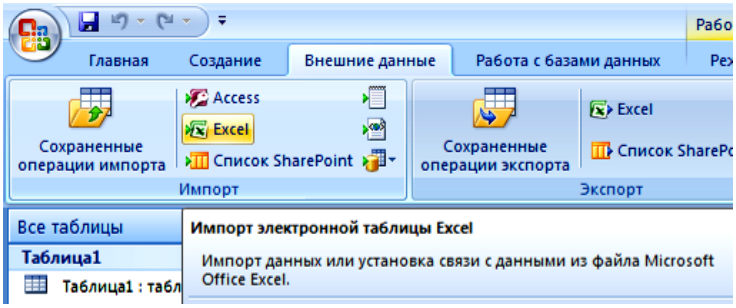
Примечание: В случае неправильного задания имен, следует воспользоваться кнопкой Диспетчер имен для их удаления.

Сохранить изменения в книге «Организация_защита_снята» и закрыть книгу.

Открыть приложение MS Access 2007, в стартовом окне приложения щелкнуть на значке Новая база данных . В нижнем правом углу стартового окна задать имя файла базы данныхОрганизация.accdb, щелкнуть на значке папки и выбрать свою рабочую папку, затем нажать ОК. Для создания базы данных в рабочей папке нажать кнопку Создать. Новая база будет создана и открыта в режиме создания/просмотра таблиц.

Выполнить импорт данных о подразделениях из файла MS Excel «Организация_защита_снята»:

щелкнуть на вкладке Внешние данные на ленте MS Access и щелкнуть кнопку Excel в группе Импорт (рис.14).



Будет запущен мастер Импорт электронной таблицы. В первом окне мастера импорта щелкнуть на кнопке Обзор, выбрать файл «Организация_защита_снята» из рабочей папки и щелкнуть Открыть, затем установить переключатель в положение Импортировать данные источника в новую таблицу в текущей базе данных и нажать ОК.

Во втором окне мастера установить переключатель в позицию именованные диапазоны и выбрать диапазон Подразделения (рис.15), затем нажать Далее.

В следующем окне мастера должен быть установлен флажок Первая строка содержит заголовки столбцов (оставить без изменения), нажать Далее.

В следующем окне мастера проиндексировать создаваемую таблицу по первому полю «Код», для этого установить следующие значения:

имя поля: Код

индекс: Да (Совпадения не допускаются)

тип данных: Длинное целое

Нажать Далее для перехода к следующему окну, в котором будет предложено задать ключевое поле новой таблицы: установить переключатель в позицию определить ключ и выбрать в качестве ключевого поле «Код». Затем нажать Далее.

В последнем окне мастера оставить без изменения название новой таблицы «Подразделения» и нажать Готово. Не сохраняя шаги импорта нажать Закрыть.

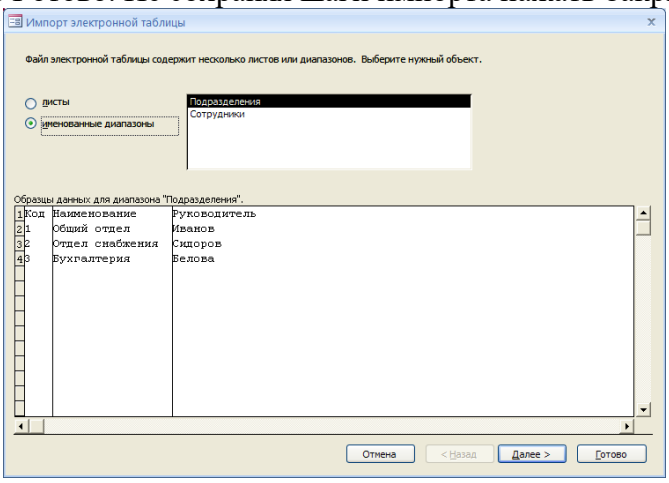


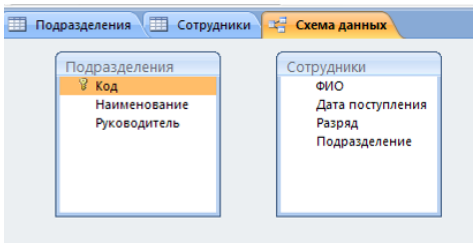
Таблица Подразделение появится в списке таблиц. Дважды щелкнуть на ее имени, чтобы просмотреть данные и убедиться в правильности импорта.

Аналогичным образом импортировать данные из диапазона Сотрудники книги «Организация_защита_снята», в процессе импорта индекс и ключ не создавать.

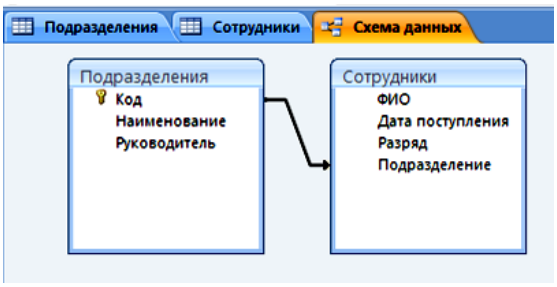
Связать таблицы по значениям кода подразделения:

На вкладке Режим таблицы выбрать команду Схема данных в группеСвязи. Добавить на схему обе таблицы (Подразделения и Сотрудники) с помощью кнопки Добавить окна Добавление таблицы, затем закрыть это окно.

На вкладке Схема данных выделить мышью поле Код таблицы Подразделения (рис.16), а затем, не отпуская кнопки мыши, перетащить его на поле Подразделение таблицы Сотрудники и отпустить кнопку мыши.



Появится окно Изменение связей с именами полей Код и Подразделение. Щелкнуть на кнопке Объединение и выбрать параметры объединения: Объединение ВСЕХ записей из «Подразделения» и только тех записей из «Сотрудники», в которых связанные поля совпадают, затем нажать ОК и Создать. Будет создана связь между таблицами, показанная на рис.17.



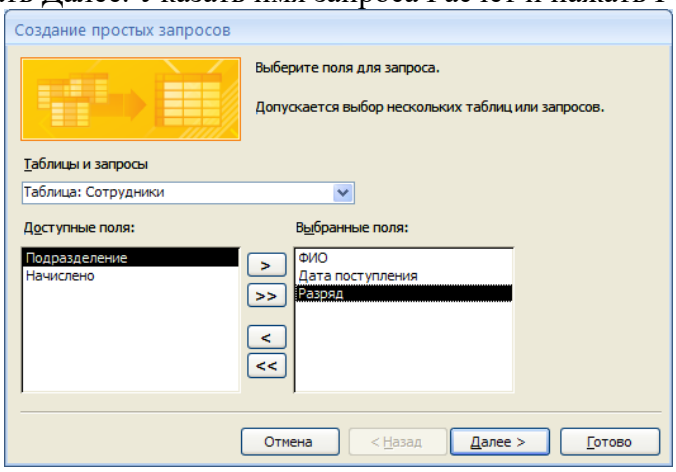
Рассчитать оклады сотрудников в зависимости от их разряда:

В режиме таблицы перейти на вкладку таблицы Сотрудники. На вкладке Режим таблицы выбрать команду Новое поле и выбрать в правой части окна шаблон полейДенежный, дважды щелкнув на нем мышью.

Переименовать добавленный столбец в Начислено, щелкнув на его заголовке правой кнопкой мыши и выбрав команду Переименовать столбец.

Создать запрос для расчета вычислений. На вкладке Создание выбрать команду Мастер запросов, в окне Новый запрос выбрать Простой запрос и нажать ОК.

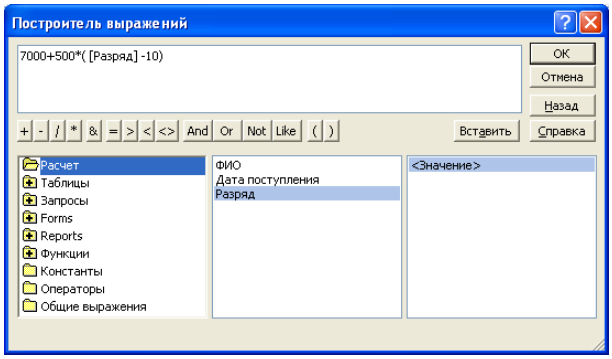
В следующем окне Создание простых запросов выбрать из выпадающего списка Таблица: Сотрудники, а затем с помощью кнопки > перенести из области Доступные поля в область Выбранные поля поля ФИО (Фамилия), Дата поступления и Разряд (рис.18), затем нажать Далее. В следующем окне мастера создания запросов установить переключатель в позицию подробный, нажать Далее. Указать имя запроса Расчет и нажать Готово.



Просмотреть данные запроса Расчет. Перейти к редактированию запроса, выбрав режим Конструктора на вкладке Главная.

В режиме конструктора запроса установить курсор мыши в строке Поле на первое пустое поле (сразу после поля [Разряд]), а затем выполнить команду Построитель на вкладке Конструктор.

В окне построителя выражений ввести формулу для расчета оклада (рис.19), при этом поле [Разряд] выбирается двойным щелчком мыши из списка полей запроса Расчет. Когда формула будет готова, нажать ОК.



После выхода из построителя выражений щелкнуть мышью на любом другом поле запроса расчет. Перед сформированной формулой появится надпись Выражение1: – заменить слово Выражение1 на слово Оклад (двоеточие не удалять).

Посмотреть результаты выполнения запроса Расчет с помощью команды Выполнить на вкладке Конструктор (рис.20).

ФИО	Дата постуг	Разряд	Оклад
Алексеева	02.07.1999	14	9000
Белова	09.01.2003	15	9500
Давыдов	12.05.1996	12	8000
Киркоров	30.08.1982	17	10500
Кукушкин	25.10.1994	16	10000
Иванов	28.02.2004	13	8500
Петров	17.01.2002	10	7000
Пискунова	11.04.2001	10	7000
Селезнев	14.10.2002	15	9500
Семенов	10.10.2001	11	7500
Сидоров	23.09.2000	17	10500
Смирнова	06.09.1994	10	7000
Смиронова	03.12.2004	12	8000

Рассчитать стаж сотрудников:

Вернуться в режим конструктора запроса Расчет. В новом поле построить выражение:

(Date() – [Дата поступления])/365

используя встроенную функцию Date (Функции/Встроенные функции) и список полей запроса Расчет в Построителе выражений.

Полученному полю дать название Стаж (вместо Выражение1).

Посмотреть результаты выполнения запроса Расчет, в случае, если значения не отображаются (поле Стаж заполнено знаками решетки), расширить поле, переместив правую границу его заголовка мышью.

Рассчитать премию, начисляемую сотрудникам в зависимости от стажа (если стаж превышает 7 лет, то премия составит пол-оклада):

Вернуться в режим конструктора запроса Расчет. В новом поле построить выражение:

IIf ([Стаж] >7 ; 0,5* [Оклад] ; 0)

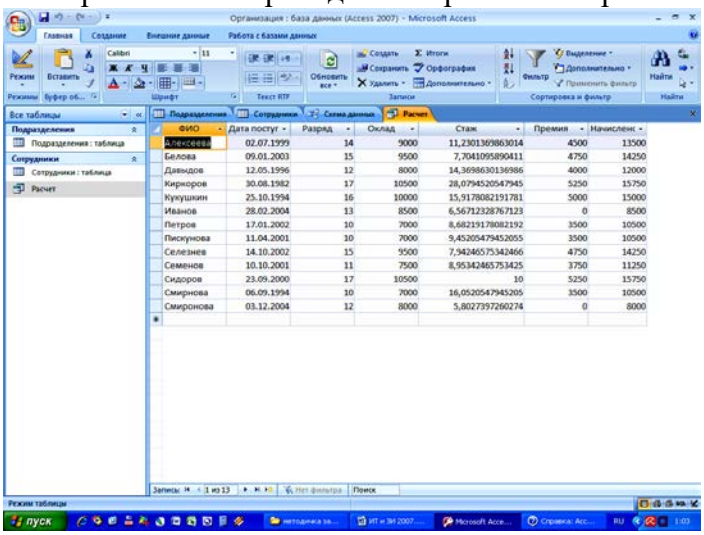
используя встроенную функцию Iif и список полей запроса Расчет в Построителе выражений.

Примечание: Если в Построителе выражений список полей не содержит вновь рассчитанные поля, следует обновить запрос, сохранив его (например, щелкнуть правой кнопкой мыши на заголовке запроса и выбрать команду Сохранить).

Полученному полю дать название Премия (вместо Выражение1).

Посмотреть результаты выполнения запроса Расчет.

В новом поле Начислено рассчитать значение общих начислений сотрудникам как сумму оклада и премии, используя список полей запроса Расчет в Построителе выражений. Результаты выполнения запроса Расчет приведены на рис.21. Сохранить запрос Расчет.



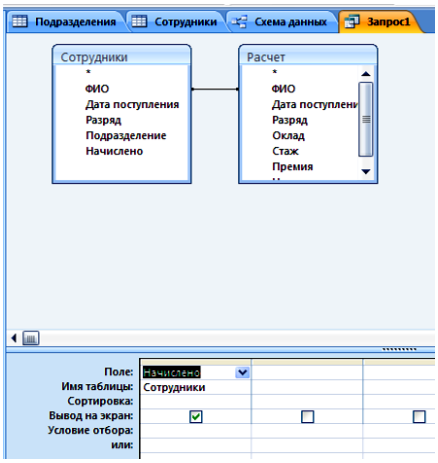
Создать запрос на обновление значения поля Начислено в таблице Сотрудники:

Выбрать команду Конструктор запросов на вкладкеСоздание. Вновь созданный запрос Запрос1 будет открыт в режиме конструктора. В окне Добавление таблицы добавить таблицу Сотрудники и запрос Расчет, затем закрыть это окно.

Примечание: если окно Добавление таблицы не открыто, можно открыть его командой Отобразить таблицу на вкладкеКонструктор.

Дважды щелкнуть по полю Начислено таблицы Сотрудники – поле Начислено добавится в качестве поля запроса Запрос1.

Связать таблицу Сотрудники и запрос Расчет, находящиеся в области данных запроса Запрос1 по полю ФИО (Фамилия) (рис.22).



Преобразовать Запрос1 в запрос на обновление, щелкнув на кнопке Тип запроса: Обновление в группе Тип запроса вкладки Конструктор. В списке полей запроса появится новая строка Обновление.

Установить курсор мыши в строку Обновление поля Начислено запроса Запрос1. Вызвать Построитель выражений. В окне построителя выражений выбрать поле Начислено запроса Расчет. Выражение примет вид: [Расчет]![Начислено], нажать ОК.

Сохранить запрос, при сохранении задать имя запроса Обновление.

Выполнить запрос Обновление с помощью команды Выполнить – будет выдано предупреждение об обновлении 13 записей, в окне предупреждения нажать Да.

Просмотреть данные таблицы Сотрудники и удостовериться, что поле Начислено заполнено.

Создать итоговый запрос, содержащий статистические данные о подразделениях организации:

Вызвать мастер создания простых запросов, в качестве данных запроса выбрать все поля таблицы Подразделения и поле Начислено таблицы Сотрудники.

Выбрать итоговый тип запроса, нажать кнопку Итоги. В окне Итоги установить флажок для функции Sum поля Начислено, установить флажок Подсчет числа записей в Сотрудники, нажать ОК.

Задать имя нового запроса Статистика.

Посмотреть результат выполнения запроса Статистика.

Переименовать поля Sum – Начислено иCount – Сотрудники в Суммарные начисления и Число сотрудников соответственно (рис.23). Сохранить запрос.

Закрыть приложение MS Access.

Сатистика

Код	Наименование	Руководитель	Суммарные начисления	Число сотрудников
1	Общий отдел	Иванов	58 500,00р.	5
2	Отдел снабжения	Сидоров	52 500,00р.	4
3	Бухгалтерия	Белова	48 750,00р.	4

Задание 2. Установить доверие к базе данных, зашифровать базу данных с паролем.

Защита баз данных MS Access 2007

Открыть приложение MS Access 2007. Открыть ранее созданную базу данных Организация, щелкнув на имени базы в правой части стартового окна MS Access. Убедиться, что выдается предупреждение системы безопасности об отключении активного содержимого базы данных

Проверить, что возможен просмотр всех объектов базы данных Организация, кроме запроса на изменение данных Обновление.

Щелкнуть на кнопке Параметры предупреждения системы безопасности, разрешить заблокированное содержимое.

Удостовериться, что теперь возможен запуск запроса Обновление.

Установить доверие к рабочей папке, содержащей файл базы данных:

щелкнуть на кнопке Microsoft Office , а затем – на кнопке Параметры Access.

В группе настроек Центр управления безопасностью щелкнуть на кнопке Параметры центра управления безопасностью.

Выбрать группу Надежные расположения, затем щелкнуть на кнопке Добавить новое расположение. В окне Надежное расположение Microsoft Office выбрать свою рабочую папку с помощью кнопки Обзор. Убедиться, что рабочая папка занесена в список надежных расположений. Сохранить настройки Access и закрыть окно параметров.

Закрыть приложение MS Access.

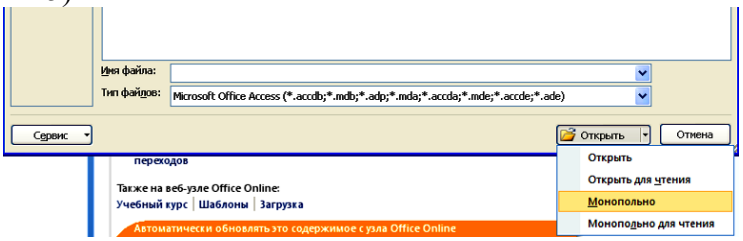
Проверить, что доверие к рабочей папке установлено: вновь открыть приложение MS Access, открыть базу данных Организация. Удостовериться, что предупреждение системы безопасности не выдается, запуск запроса Обновление возможен.

Зашифровать базу данных с паролем:

Закрыть приложение MS Access.

Открыть приложение MS Access. Щелкнуть на кнопке Microsoft Office , выбрать команду Открыть.

В окне Открытие файла базы данных выбрать рабочую папку и базу Организация, затем щелкнуть на значке выпадающего списка кнопки Открыть в правом нижнем углу окна и выбрать Монопольно (рис.25).



После открытия базы выполнить команду Зашифровать паролем на вкладке Работа с базами данных. В окне Задание пароля базы данных дважды ввести пароль high, нажать ОК.

Закрыть и вновь открыть базу данных, удостоверившись, что для открытия базы требуется ввод пароля.

Создать подписанный пакет с базой данных:

Создать новый цифровой сертификат с помощью средства Цифровой сертификат для проектов VBA (Digital Certificate for VBA Projects), находящегося в группе Средства Microsoft Office (Microsoft Office Tools) в меню Microsoft Office, либо использовать сертификат, созданный ранее при выполнении лабораторной работы №2.

Щелкнуть на кнопке Microsoft Office и выполнить команду Опубликовать/Упаковать и подписать. В окне команды выбрать свой сертификат и нажать ОК. Оставить без изменения имя пакета Организация.accdc и расположение в рабочей папке, нажать кнопку Создать.

Просмотреть содержимое рабочей папки с помощью Проводника Windows. Найти файл подписанного пакета, помеченный значком подписи .

В окне Проводника щелкнуть на значке файле подписанного пакета правой кнопкой мыши и выбрать Свойства, перейти на вкладку Цифровые подписи и просмотреть данные о сертификате, нажав кнопку Сведения.

Открыть файл подписанного пакета и удостоверится, что пакет не открывается в MS Access непосредственно, а требует извлечения содержимого в другую базу данных (открывается окно Извлечь базу данных в). Не извлекать базу из пакета, нажав кнопку Отмена.

Практическое занятие № 17

Тема: Установка СУБД. Настройка компонентов СУБД

Цель: приобрести практические навыки работы в среде разработки баз данных MS SQL Server 2017 и усвоить способы управления базами данных.

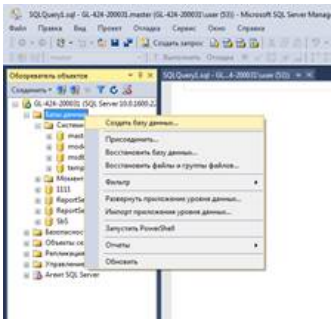
Создание базы данных

SQL Server 2017 предлагает два метода создания баз данных: средствами SQL Server Management Studio или применением команды CREATE DATABASE в среде Query Editor, которую можно сохранить в файле и запускать как сценарий.

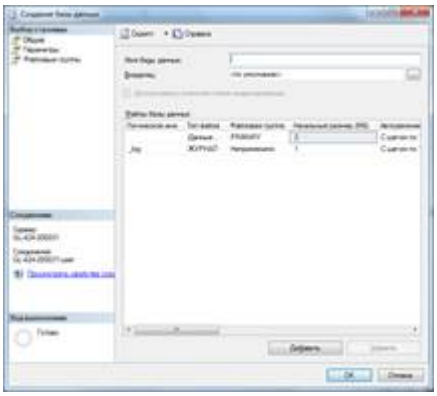
Применение SQL Server Management Studio

Для создания базы данных при помощи SQL Server Management Studio выполните следующую последовательность шагов.

1. Запустите на выполнение SQL Server Management Studio. В левой панели раскройте группу SQL Server, в которой находится имя сервера, на котором вы хотите создать базу данных, а затем раскройте узел самого этого сервера. Затем нажмите правой кнопкой мыши на папку Database и выберите New Database.



2. Откроется диалоговое окно свойств базы данных (Database Properties) с открытой вкладкой General (Общие).

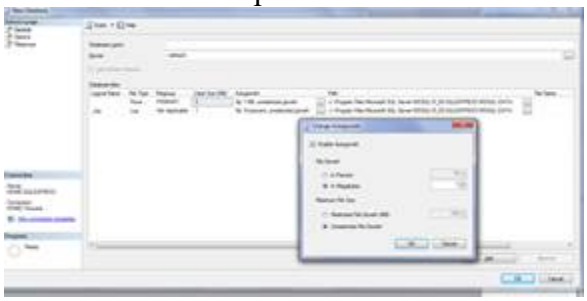


Введите с клавиатуры имя базы данных (логическое имя) в поле Database Name. Затем нужно ввести информацию о размере файла БД и параметрах его роста. Management Studio автоматически создает первичный файл данных, с именем вашей базы данных в качестве префикса и primary в качестве имени группы файлов.

Вы можете изменить имя, местоположение и размер первичного файла, но вы не сможете изменить группу файлов для первичного файла данных. Для каждого файла данных, кроме первичного файла, вы можете задать имя пользовательской группы файлов, и, в соответствии с вашим желанием, эта группа файлов будет создана. По умолчанию, каждый файл располагается на диске в папке, в которой инсталлирован SQL Server.

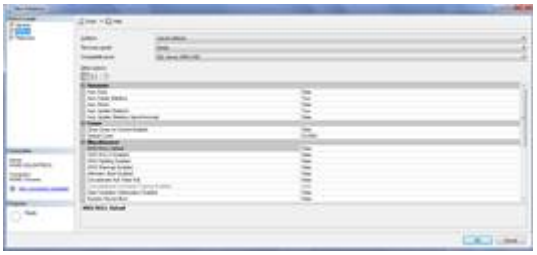
Вы можете изменить эту настройку, задав другой путь с клавиатуры или при помощи экранной кнопки для его поиска («. . .»).

Нажав на селективную кнопку («. . .») вы можете задать приращение файла, выраженное в мегабайтах или в процентах от свободного места, оставшегося в файле. Вы также можете задать максимальный размер файла, указав предел роста, выраженный в мегабайтах, а можете не ограничивать рост файла. Эти настройки можно задавать при создании каждого из файлов, а можете оставить настройки, применяемые по умолчанию, и задать их позднее при помощи окна Database Properties.



Файлы журнала конфигурируются точно так же, как и файлы данных, за исключением того, что вы не сможете задать для них группу файлов, потому что они не принадлежат ни одной из групп файлов. Задайте с клавиатуры местоположение (физическое имя) и начальный размер одного или нескольких файлов журнала. Кроме того, задайте настройки автоматического роста файлов журнала, так же как это было описано для файлов данных.

3. Теперь можно перейти к вкладке Options, на которой доступно большое количество настроек.

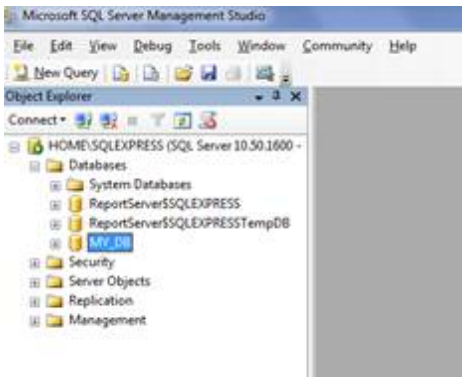


Наибольшего внимания заслуживает параметр с обозначением способа упорядочивания (Collation).

Следующим важным параметром является уровень совместимости (Compatibiliti level). От него зависит то, будут ли поддерживаться некоторые синтаксические конструкции и ключевые слова, предусмотренные в версии SQL Server 2017. С его помощью можно вернуться к использованию ключевых слов, принятых в предыдущих версиях SQL Server.

Выбор остальных параметров настройки зависит от требований к конкретной инсталляции.

4. После того как вы настроите все файлы так, как вам это нужно, нажмите на OK, и SQL Server создаст базу данных. Вернитесь в Enterprise Manager и нажмите на папку Database сервера, в который вы только что добавили новую базу данных. Вы увидите в правой панели Management Studio, что SQL Server добавил значок-иконку для этой базы данных.



Применение операторов T-SQL в Query Editor

Создавать или изменять свои базы данных можно при помощи операторов T-SQL, не пользуясь графическим пользовательским интерфейсом (GUI). Можно создавать свои собственные сценарии создания баз данных.

Сценарий, вводимый в Query Editor, представляет собой набор SQL команд. Создание базы данных осуществляется с помощью команды:

CREATE DATABASE имя_базы_данных
[ON [PRIMARY] (NAME = ’логическое_имя_файла’,
FILENAME = ’физическое_имя_файла’
[, SIZE = размер]
[, MAXSIXE = {максимальный_размер | UNLIMITED}]
[, FILEGROWTH = шаг_приращения_размера [Mb | Kb | %])
[{FILEGROUP имя_файловой_группы}]
[, ...n]
[LOG ON (NAME = ’логическое_имя_файла’,
FILENAME = ’физическое_имя_файла’
[, SIZE = размер]
[, MAXSIXE = {максимальный_размер | UNLIMITED}]
[, FILEGROWTH = шаг_приращения_размера [Mb | Kb | %])
[, ...n]

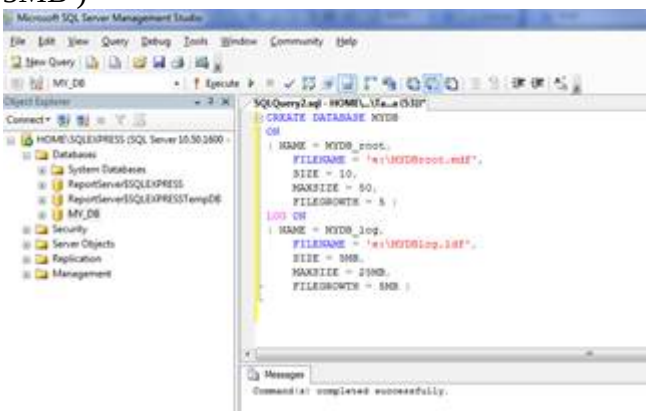
Описание параметров оператора CREATE DATABASE:

- ON – ключевое слово, указывает службе SQL Server, что в команде должны быть заданы расположение файлов данных, их имена, объем и величина объема приращения;
- NAME – логическое имя файла, по которому происходит обращение к этому файлу со стороны SQL Server. По умолчанию совпадает с физическим именем файла, определенном в параметре FILENAME;;
- FILENAME – физическое имя файла с указанием полного пути с обязательным указанием расширения файла;
- SIZE – исходный объем в мегабайтах. Параметр не обязателен, его можно опустить. Минимально возможное значение 512 Кб. Размер основного файла по умолчанию равен размеру БД model. По умолчанию размер дополнительных файлов данных и журнала равен 1 Мб;
- FILEGROWTH – приращение объема файла после его заполнения. Приращение можно указать в мегабайтах или процентах от текущего объема. Если этот параметр не указан, то устанавливается значение UNLIMITED, позволяющее увеличивать файлам размер без ограничений. По умолчанию приращение – 10%, если не указаны единицы, то цифра воспринимается в мегабайтах;
- LOG ON – ключевое слово, указывает службе SQL Server, что в команде должны быть заданы расположение файлов журнала, их имена, объем и величина объема приращения;
- COLLETE – необязательный параметр, указывает схему сортировки, применяемой в БД;

- PRIMARY – определяет файл как первичный или как член первичной файловой группы, если опущено, то основным файлом становится первый файл в операторе и для хранения используется первичная файловая группа;
- MAXSIZE – указывает максимальный размер, до которого может увеличиваться файл;
- FILEGROUP – определяет имя группы файлов, в которую помещается файл.

В приведенном ниже тексте дан пример создания базы данных с именем MyDB, содержащей: первичный файл данных (MyDB_root) и один файл журнала транзакций (MYDB_log). Наберите данный текст в панели запросов и запустите этот сценарий нажатием клавиш F5 или Ctrl-E

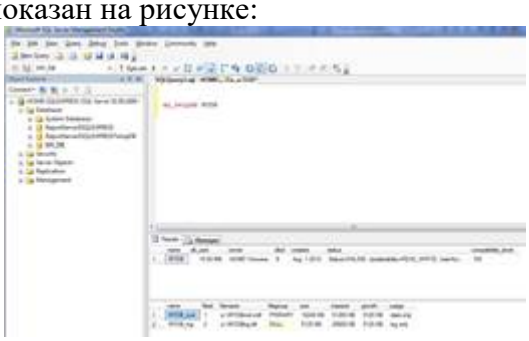
```
CREATE DATABASE MYDB
ON
( NAME = MYDB_root,
   FILENAME = 'e:\MYDBroot.mdf',
   SIZE = 10,
   MAXSIZE = 50,
   FILEGROWTH = 5 )
LOG ON
( NAME = MYDB_log,
   FILENAME = 'e:\MYDBlog.ldf',
   SIZE = 5MB,
   MAXSIZE = 25MB,
   FILEGROWTH = 5MB )
```



Выполните следующий код, чтобы убедиться, что БД была создана с нужными параметрами:

```
sp_helpdb MYDB
```

Результат выполнения показан на рисунке:



Можно создать и SQL-скрипт для создания нашей базы данных на, например, другом экземпляре SQL сервера. Для этого в окне Object Explorer в контекстном меню базы данных выбираем Script Database as -> CREATE To -> New Query Editor Window:

И в новом окне SQL редактора появится скрипт для создания нашей базы данных. Это скрипт можно сохранить и запустить потом на другом SQL сервере для создания базы данных с такой же структурой.

Удаление баз данных

Базы данных можно удалять как при помощи Management Studio, так и командами T-SQL.

Применение Management Studio

1. Находясь в Management Studio, раскройте группу SQL Server, а затем раскройте имя сервера, на котором установлена база данных.

2. Раскройте папку Databases, чтобы стали видны имеющиеся базы данных.

3. Нажмите правой кнопкой мыши на имя удаляемой базы данных, а затем выберите Delete в контекстном меню.

Применение команд SQL

Помните, что удаление базы данных является неотменяемым действием. Для удаления базы данных применяется T-SQL команда DROP DATABASE. Ниже приведены примеры команд, которые удалят базу данных MyDB и все ее файлы.

```
Use master
(для запуска команды DROP DATABASE вы должны применять базу данных master)
GO
DROP DATABASE  MyDB
(Единственным параметром этой команды является имя удаляемой базы)
GO
```

После удаления базы данных следует обновить резервную копию базы данных master, чтобы в ней содержалась свежая информация о пользовательских базах данных и не содержалась информация о только что удаленной базе данных, также обратите внимание, что база данных не может быть удалена, когда к ней имеют доступ пользователи. Перед удалением базы данных нужно отсоединить от нее всех пользователей.

Отсоединение базы данных от сервера

Отключение базы данных означает, что она становится недоступной и может быть перенесена на другой носитель. Для того чтобы отключить базу данных, необходимо вначале отключить всех пользователей. Процедура отключения БД заключается в следующем:

1. Откройте окно Enterprise Manager и найдите узел вашей базы данных.

2. Щелкните на значке базы данных правой кнопкой мыши и выберите в контекстном меню команду Tasks (Задачи), а затем, в развернувшемся подменю, щелкните на Detach Database (Отсоединить базу данных). Щелкните на Ok.

Так как ваша база данных более недоступна ни для каких обновлений или модификаций, то ее можно копировать любым способом, применимым для копирования обычных файлов.

Подключение базы данных

Когда база отключена, ее можно скопировать на несколько разных носителей, совершив, таким образом, своеобразное "клонирование". Перед подключением базы данных, убедитесь, что копия базы находится там же, где и всегда, если это не так, то при попытке подключения возникнет ошибка.

Подключение базы данных выполняется следующим образом:

1. Убедитесь, что окно Management Studio открыто, и что выделен узел вашей базы данных.

2. Щелкните на нем правой кнопкой мыши и выберите в контекстном меню команду Attach (Присоединить).

3. Вы войдете в диалоговое окно, где нужно найти местоположение вашей присоединяемой БД. Кроме того, для этой операции пользователь должен принадлежать к группе администраторов. После того, как все настройки сделаны, нажмите Ok.

Изменение владельца базы данных производится с помощью специальной хранимой процедуры. Владельцем можно сделать любую учетную запись, которая в настоящий момент не является пользователем базы, следующим образом:

sp_changedbowner [[@loginname=] ‘имя_пользователя’

Переименование базы данных:

sp_renamedb [@dbname=] ‘старое_имя’, [@newname=] ‘новое_имя’

Для переименования базы данных ее необходимо перевести в однопользовательский режим работы.

Для управления уже существующими файлами журнала и файлами данных, добавления дополнительных файлов данных или журнала, удаления файлов, а также для работы с файловыми группами используется команда:

ALTER DATABASE база_данных
{ ADD FILE <указание_на_файл> [TO FILEGROUP наименование]
| ADD LOG FILE <указание_на_файл>
| REMOVE FILE логическое_имя_файла
| ADD FILEGROUP имя_группы
| REMOVE FILEGROUP имя_группы
| MODIFY FILE <указание_на_файл>
| MODIFY FILEGROUP имя_группы свойство_группы }
где <указание_на_файл> =
(NAME = ’логическое_имя_файла’,
FILENAME = ’физическое_имя_файла’
[, SIZE = размер]
[, MAXSIXE = {максимальный_размер | UNLIMITED}]
[, FILEGROWTH = шаг_приращения_размера [Mb | Kb | %])

Данная команда позволяет добавлять файл в существующую файловую группу, удалять файлы (при этом удаляется и физический файл), добавлять и удалять файловые группы, изменять физические параметры уже существующих файлов, а также изменять свойства файловых групп: READONLY, READWRITE, DEFAULT (при определении этого свойства, в эту группу будет заносится файлы, у которых в параметрах не определена принадлежность к группе; установленной по умолчанию изначально считается первичная файловая группа).

Сжатие базы данных

Сжатие базы данных – это процесс уменьшения размеров файлов базы данных за счет удаления неиспользуемых частей файла. Существует три способа сжатия базы данных:

- автоматическое сжатие при установке соответствующего параметра в настройках базы данных;
- удаление свободного пространства из файлов базы данных с помощью утилит администрирования MS SQL Server;
- уменьшение размера указанных файлов (или файловых групп), а также очистка содержимого файлов для их последующего удаления.

Автоматическое сжатие данных выполняется постоянно с определенными интервалами, если установлен параметр базы данных autoshrink. При операциях автоматического сжатия нельзя определить, какую часть базы данных необходимо сжать. MS

SQL Server пытается освободить значительную часть базы данных самостоятельно. Эти операции выполняются в период наименьшей активности пользователей.

Сжатие всей базы данных вручную осуществляется с использованием следующей команды:

DBCC SHRINKDATABASE
(database_name | database_id | 0
 [, target_percent]
 [, { NOTRUNCATE | TRUNCATEONLY }]
)
[WITH NO_INFOMSGS

Описание параметров:

- database_name | database_id | 0 - Имя или идентификатор базы данных, которая должна быть сжата. Если указано значение 0, используется текущая база данных;

- target_percent - процент свободного пространства, которое должно остаться в базе данных после сжатия;

- NOTRUNCATE - сжимает данные в файлах с помощью перемещения распределенных страниц из конца файла на место нераспределенных страниц в начале файла. Аргумент target_percent является необязательным. Свободное место в конце файла операционной системе не возвращается, и физический размер файла не изменяется. Следовательно, если указан аргумент NOTRUNCATE, сжатие файлов данных незначительно. Аргумент NOTRUNCATE применим только к файлам данных. Файл журнала не затрагивается;

- TRUNCATEONLY - освобождает все свободное пространство в конце файла операционной системе, но не перемещает страницы внутри файла. Файл данных сокращается только до последнего выделенного экстента. Аргумент target_percent не обрабатывается, если указан аргумент TRUNCATEONLY. Аргумент TRUNCATEONLY оказывает влияние на файл журнала. Для усечения только файла данных используйте инструкцию DBCC SHRINKFILE;

- WITH NO_INFOMSGS - добавляет все информационные сообщения со степенями серьезности от 0 до 10.

Права на сжатие базы данных выданы только членам роли sysadmin и владельцам базы данных. После сжатия базы данных выводится отчет, в котором указывается:

- количество страниц, до которых сжимается файл;
- расчетное число страниц, в которые могут быть помещены все данные файла;
- количество страниц, содержащих данные;
- количество страниц, на которые файл может быть еще сжат.

Нельзя сжать базу данных до размера меньше первоначального.

Сжатие базы данных можно осуществить также и путем сжатия каждого ее файла с помощью следующей команды:

DBCC SHRINKFILE (‘имя_файла’, [‘конечный_размер’] [, EMPTYFILE | NOTRUNCATE | TRUNCATEONLY])

Описание параметров:

- имя_файла – логическое имя файла, который необходимо сжать;
- конечный_размер – желательный размер (целое число в мегабайтах), который должен иметь файл после выполнения сжатия. Если этот параметр не указан или меньше минимально допустимого размера, то файл сжимается до минимально возможного размера;
- EMPTYFILE – выполняется перенос данных из файла в другие файлы файловой группы;
- NOTRUNCATE – освободившееся место не возвращается операционной системе, т.е. размер файла не уменьшается на самом деле. При этом данные располагаются более компактно и смещаются к началу файла;
- TRUNCATEONLY – происходит обрезание файла, начиная с последней используемой страницы. Никакого перемещения данных не происходит.

Резервное копирование данных

MS SQL Server предлагает следующие типы резервного копирования информации:

- полная копия базы данных, которая является отправной точкой при восстановлении базы данных после сбоя, однако в зависимости от объема данных этот процесс может занимать много времени, поэтому не рекомендуется выполнять его слишком часто. Полная копия содержит все данные, содержащиеся в базе данных на момент окончания резервирования;
- копия журнала транзакций, необходима для фиксирования всех изменений данных, произошедших в системе с момента последнего резервного копирования. Сама копия журнала содержит сведения о транзакциях и лишь только вместе с копией базы данных позволяет вернуться к состоянию, предшествующему сбою;
- дифференциальная копия данных содержит изменения данных, произошедшие с момента последнего создания полной копии базы данных. При этом сохраняются только страницы подвергшиеся изменениям. Таким образом, для восстановления базы данных достаточно самой последней дифференциальной копии.

Для выполнения резервного копирования необходимо выбрать носитель, т.е. определить устройство, которое будет использоваться для создания копий. Для добавления устройства используется хранимая процедура

sp_addumpdevice ‘тип_устройства’, ‘логическое_имя’, ‘физическое_имя’

Описание параметров:

- тип_устройства – тип устройства резервного копирования. Допустимые значения: TAPE (магнитная лента), DISK (магнитный диск), Virtual_Device (виртуальное хранилище);
- логическое_имя, физическое_имя – логическое и физическое имя устройства резервного копирования соответственно.

Для создания резервной копии базы данных, журнала транзакций, файлов и файловых групп необходимо воспользоваться командой:

```
BACKUP {LOG | DATABASE } имя_БД
[ FILE = ‘логическое_имя_файла’, ...]
[ FILEGROUP = ‘имя_группы’ ]
TO логическое_имя_устройства
[ WITH
[ DESCRIPTION = ‘комментарий’ ]
[ DIFFERENTIAL ]
[ EXPIREDATE = ‘дата’ ]
[ INIT | NOINIT ] ... ]
```

Описание параметров:

- DIFFERENTIAL – создается дифференциальная копия базы данных;
- EXPIREDATE – определяется дата, после которой резервная копия считается устаревшей и может быть перезаписана;
- INIT | NOINIT – система осуществляет или нет инициализацию устройства.

Пример. Создадим устройство для резервирования:

```
sp_addumpdevice 'disk', 'backupdisk', 'e:\MYDB.bak'
```

Воспользуемся функцией BACKUP для выполнения резервного копирования:

```
BACKUP DATABASE MYDB TO backupdisk
```

Практическое занятие № 18

Тема: Созданием форм и отчетов

Цель: уметь создавать формы и отчёты по таблицам баз данных.

Краткие теоретические сведения:

Формы и отчёты создаются на соответствующих вкладках в базе данных. Для их создания необходимо перейти на соответствующую вкладку, щёлкнуть по кнопке Создать и выбрать вариант создания: автоматически, с помощью Мастера, с помощью Конструктора

Задание: Подготовить базу данных для школьной библиотеки, содержащую составную форму. В качестве исходных таблиц используйте таблицуЧитателис данными об учащихся и таблицуАбонементс данными о книжном фонде

Откройте MicrosoftAccessи создайте новую базу данныхБиблиотека.

Создайте с помощью конструктора таблицу Читатели со следующими полями: Фамилия, Группа, Год рождения, Адрес, Телефон.

Заполнить таблицу не менее 5 записей.

Вторая таблица будет содержать сведения о книжном фонде. Создайте новую таблицу Абонемент, в режиме конструктора, определив для нее следующие поля:

№ — библиотечный номер, однозначно определяющий издание в фонде библиотеки. Тип поля- числовой (ключевое поле).

Автор(текстовый).

Название книги(поле МЕМО - чтобы можно было разместить длинные названия книг).

Год издания(числовой).

Ввести не менее 5 записей.

Абонимент : таблица

№	Автор	Название книги	Год издания
456234		Система управления базами данны Microsoft Access 2000	1999
782345		Microsoft Excel для Windows 2000	2001
780652	Ботт Эд	MS Office 98	1997
854278	вейскас ,	Эффективная работа в MS Access 2000	2000
			0

Рисунок 1

Для того чтобы можно было вести учет выдачи книг, создайте третью таблицу Учет, в которой будет размещен результат заполнения связанной формы. Определите следующие поля таблицы:

№ (ключевое);

ФИО читателя (текстовое);

Дата выдачи(Дата\время, маска ввода 00.00.0000)

Закройте таблицу, сохранив ее под именем Учет. Откройте таблицуУчети добавьте библиотечные номера из таблицыАбонементпутем копирования поля целиком и вставки в необходимую таблицу (рис. 2). Ввести не менее 5 записей.

На схеме данных установите связь между таблицами Абонементи Учет, связав поля№и№ (рис. 3).Затем установите связь между таблицамиУчетиЧитатели, связав поляФИО читателяиФИО.

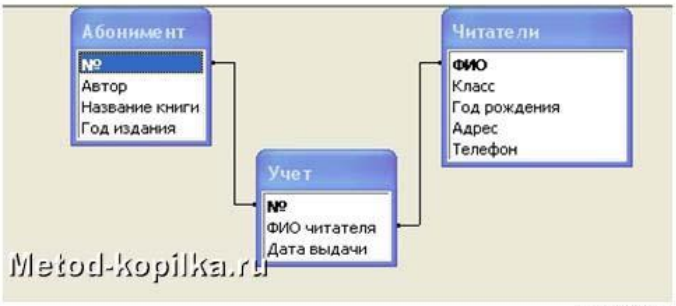


Рисунок 3

Перейти на вкладку Формы. Создать автоформу в столбец по таблице Читатели: щёлкнуть по кнопке Создать, выбрать Автоформа в столбец и выбрать таблицу Читатели,ОК.

С помощью Мастера создать форму для таблицы Абонемент: щёлкнуть по кнопке Создать, выбрать Мастер форм, выбрать таблицу Абонемент, далее, следуя указаниям Мастера, создать форму, выбрав произвольные параметры.

Создание составной формы. Выберите следующие поля основной формы Таблица: Абонемент

№,

Автор,

Название книги.

Для подчиненной формы определите поля Таблица: Учет

ФИО читателя,

Дата выдачи.

Введите заголовок формы Учет книжного фонда.В готовую форму остается вносить фамилию читателя и дату выдачи книги. Откройте ФормуУчет книжного фондав режиме конструктора и добавьте надпись в полеЗаголовокформы (рис. 4).

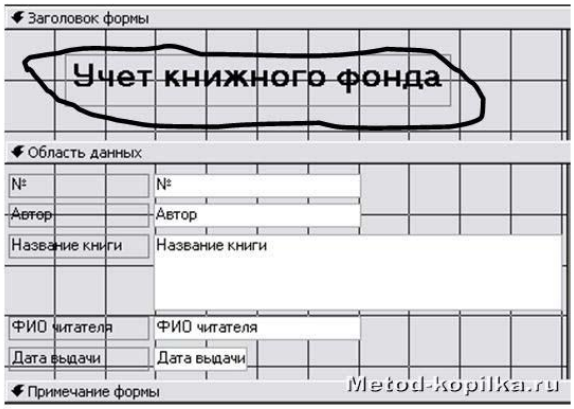


Рисунок 4

Управление данными

Дальнейшая работа заключается в том, чтобы вносить ФИО читателя и дату при выдаче каждой книги и удалять эти сведения, когда читатель сдает книгу. Откройте таблицу Учети проверьте, как отображены в ней данные, внесенные в форму. Вернитесь к форме Учет книжного фонда. Внесите изменения (Кто-то сдал книгу, кто-то взял). Имейте в виду, что один человек может взять одновременно несколько книг. Закройте форму. Проверьте данные таблицыУчет (рис. 5).

9. Создать отчёт по таблицам, аналогично созданию форм.

Практическое занятие № 19

Тема: Создание меню. Генерация, запуск

Большинство коммерческих программ имеет в верхней части приложения свою панель элементов, на которой есть разные кнопки. Давайте напишем программу, которая создает панель меню без кнопок.

```
Public Function funCreateMenu(strMenu As String) As Boolean
    Dim myBar As CommandBar
    'Создаем панель меню
    myBar = appAccess.CommandBars.Add(strMenu, msoBarTop, True)
    funCreateMenuControls strMenu '<10> Создаем кнопки меню
    myBar.Visible = True 'Отображаем меню
    funCreateMenu = True 'Возвращаем результат
End Function
```

СОЗДАНИЕ КНОПОК МЕНЮ

В этой части лекций рассказывается о программе, которая создает две кнопки Помощник и Справка. При нажатии их будет отображаться файл справки и помощник по Вашей программе.

```
Public Function funCreateMenuControls(strMenu As String) As Boolean
```

```
    Dim but As CommandBarButton

    'Добавляем первую кнопку
    but = appAccess.CommandBars(strMenu).Controls.Add(msoControlButton)
    With but
       .BeginGroup = True 'Начинаем размещение с начала группы
       .FaceId = 1 'Устанавливаем код кнопки
       .Style = msoButtonCaption 'Выбираем стандартный тип
       .Caption = "Справка" 'Называем кнопку
       .OnAction = "funCreateNewHelp" 'Определяем программу справки
    End With
    'Добавляем вторую кнопку
    but = appAccess.CommandBars(strMenu).Controls.Add(msoControlButton)
    With but
       .Caption = "Помощник" 'Называем кнопку
       .Style = msoButtonCaption 'Выбираем стандартный тип
       .FaceId = 2 'Устанавливаем код кнопки
       .OnAction = "funCreateAssistant" 'Определяем программу помощника
    End With
    funCreateMenuControls = True 'Возвращаем результат
End Function
```

Система меню приложения.

Перед тем, как приступить непосредственно к созданию приложения, можно создать все требуемые объекты (базу данных, входящие в нее таблицы, экранные формы, отчеты, запросы). Затем отдельные объекты могут быть объединены посредством меню.

Планирование приложения.

Первым этапом создания любого приложения является определение требований, предъявляемых к законченному приложению. Прежде всего, необходимо определить состав информации, которая будет содержаться в создаваемой базе данных.

После определения состава данных, которые будут храниться в базе данных, создаются все входящие в нее таблицы. На этом этапе определяются отношения между таблицами, структура каждой таблицы и совпадающие поля для связывания таблиц.

Состав информации невозможно определить без учета конкретных задач, для решения которых предназначается создаваемое приложение. Поэтому одновременно с составом информации необходимо определить те средства, которые получит в свое распоряжение пользователь при работе с приложением.

Приложение должно содержать эффективную справочную систему, которая не требует специального руководства. В среде Windows предпочтительнее всего создать справочную систему в принятом в Windows стандарте, так как в этом случае пользователь сможет получить любую информацию в знакомой ему среде.

После того как спроектированы таблицы, входящие в состав базы данных, и определены основные требования, предъявляемые к приложению, можно приступить к созданию структуры меню приложения или, как часто его называют, дерева меню. Дерево меню можно организовать на основе функций, выполняемых приложением (например Продажа, Поступление товара) или же на основе таблиц, используемых в приложении (Покупатель, Товар). Оба варианта широко используются на практике, выбор одного из них или использование альтернативного принципа построения дерева меню определяется конкретными требованиями. Прежде чем начинать описывать структуру меню с помощью конструктора меню, следует нарисовать его эскиз на бумаге, посоветоваться с пользователями приложения по поводу его структуры. Пока меню нарисовано только на бумаге, можно легко его изменить, добавить новые пункты и удалить лишнее. По тщательно разработанному

эскизу меню достаточно легко определить набор экранных форм и отчетов, которые потом следует создать, и подключить к конкретному пункту меню.

Создание строки меню.

Примером строки меню является основное меню Visual FoxPro. Можно создать собственную строку меню, которая будет замещать основное меню Visual FoxPro или добавляться к нему. Для создания строки меню необходимо выполнить следующие действия:

1. Открыть окно конструктора меню.

2.Описать пункты меню.

3. Отобразить строку меню на экране.

4. Определить действия, выполняемые при выборе пункта меню.

Окно конструктора меню.

Для создания нового меню в конструкторе меню можно воспользоваться следующими способами:

–Выполнить команду меню File | New. В открывшемся окне диалога «New» установить опцию Menu и нажать кнопку New File.

–В окне проекта перейти на вкладку «Other» и выбрать группу «Menus». Затем нажать кнопку New окна проекта.

–Находясь в группе «Menus» окна проекта, нажать кнопку New на стандартной панели инструментов Visual FoxPro. В открывшемся окне диалога «New» установить опцию Menu и нажать кнопку

New File.

Независимо от используемого способа на экране откроется окно диалога «New Menu», предлагающее два варианта создаваемого меню (рис. 1):

–Menu – создание меню в виде строки меню;

–Shortcut – создание всплывающего меню, в котором основные пункты меню расположены вертикально.

При выборе любого из вариантов появляется окно конструктора меню, а в основном меню Visual FoxPro добавляется новый пункт Menu (рис. 2). Создание каждого из видов меню совершенно идентично, хотя каждый вид описывается разными командами Visual FoxPro. Различия эти можно заметить, только просмотрев тексты файлов .MPR, в которых хранятся тексты команд определения меню. Эти файлы следует редактировать только в крайнем случае, при необходимости дополнительной настройки создаваемого меню.

Определение текстов пунктов строки меню.

Для определения текстов пунктов строки меню необходимо нажать кнопку Insert и напечатать текст в поле Prompt. Для определения типа пункта меню следует выбрать требуемый элемент из списка Result.

Возможен выбор следующих типов пунктов меню:

Тип

Описание

Command

При выборе пункта меню будет выполняться связанная с ним команда

Pad Name

При выборе пункта меню никаких действий выполняться не будет. Как правило, используется в качестве дополнительного пояснения к меню.

Submenu

При выборе пункта меню раскрывается связанное с данным пунктом ниспадающее меню.

Procedure

При выборе пункта меню вызывается процедура, определенная для данного пункта меню.

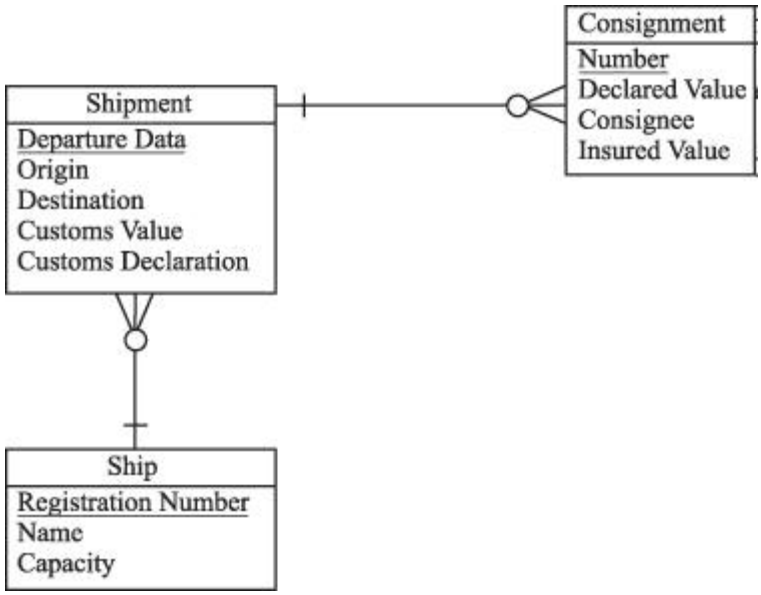
Ввод команды, выполняемой при выборе пункта меню, осуществляется в поле, которое расположено правее описания типа пункта меню. При выборе типов Procedure и Submenu в

окне конструктора правее описания типа пункта меню появляется кнопка Create, при нажатии на которую можно перейти в окно создания процедуры или начать создание ниспадающего меню для выбранного пункта меню.

Сохранение, генерация и запуск меню.

Для сохранения меню необходимо выполнить команду меню File | Save as. В открывшемся окне диалога «Save as» в поле Папка: требуется указать каталог, в который сохраняется файл, а в поле Save Menu ввести имя сохраняемого меню. Для сохранения служит кнопка Сохранить.

Кнопка Preview окна конструктора меню позволяет просмотреть внешний вид создаваемого меню, но не позволяет его активизировать. Для использования меню в приложениях его необходимо предварительно сгенерировать. Для этого служит команда Menu | Generate. При этом открывается окно диалога «Generate Menu» (рис. 3). В поле ввода окна диалога необходимо ввести наименование файла, который будет создан в результате генерации. Используя кнопку … можно выбрать путь и имя файла в стандартном окне Windows.



Задание.

1.Нарисуйте на бумажном носителе структуру меню Вашего проекта. Обоснуйте эту структуру и опишите подход, применяемый Вами при построении меню.

2.Создайте меню для Вашего проекта в конструкторе меню. Отобразите в отчете тексты процедур и команды, используемые в меню.

3.Просмотрите вид меню на экране, и отметьте поведение каждого пункта меню.

4.Проверьте правильность указания главного меню приложения в главной процедуре. Отметьте команды, отвечающие за настройку главного меню приложения.

Практическое занятие № 20

Тема: Профилирование запросов клиентских приложений

Профилирование запросов в Mysql применяется для оценки производительности вашего приложения. При разработке средних и больших приложений приходится иметь дело с сотнями запросов распределенными по вашему коду, которые исполняются каждую секунду. Без техники профилирования запросов бывает очень сложно найти из-за чего страдает быстродействие вашего приложения.

Что такое лог медленных запросов в MySQL?

Лог медленных запросов в MySQL — лог, отмечающий медленные и потенциально проблемные запросы. MySQL по умолчанию поддерживает такой функционал, но он отключен. При помощи настройки определенных переменных сервера мы можем указать какие именно запросы нас интересуют. Чаще всего нам нужны запросы которые требуют

определенное количество времени на своё выполнение или запросы, которые некорректно обрабатывают индексы.

Настройка переменных профилирования

Основные переменные для настройки лога запросов:

slow_query_log G

slow_query_log_file G

long_query_time G / S

log_queries_not_using_indexes G

min_examined_row_limit G / S

Замечание: G — глобальные переменные, S — системные переменные

slow_query_log — булево значение включающее лог

slow_query_log_file — абсолютный путь к файлу лога. Владельцем каталога должен быть пользователь mysqld, а также директория должна иметь корректные разрешения для чтения и записи. Чаще всего демон mysql работает от имени пользователя mysql.

Для проверки выполните следующую команды:

```
ps -ef | grep bin/mysqld | cut -d' ' -f1
```

Copy

Вывод команды даст вам имя текущего пользователя и пользователя mysqld. Пример настройки директории /var/log/mysql:

```
cd /var/log
sudo mkdir mysql
sudo chmod 755 mysql
sudo chown mysql:mysql mysql
```

Copy

long_query_time — время в секундах для проверки продолжительности запроса. Например, при значении 5, каждый запрос продолжительностью более 5 секунд будет записан в лог.

log_queries_not_using_indexes — булево значение, включает сохранение запросов не использующих индексы. Такие запросы очень важны при анализе.

min_examined_row_limit — указывает минимальное значение количества рядов данных для анализа. Значение 1000 проигнорирует запросы возвращающие меньше 1000 рядов значений.

Установить эти переменные можно в конфигурационном файле MySQL, динамически через MySQL GUI или командную строку MySQL. Если переменные указаны в конфигурационном файле, то сервер установит их при очередном запуске. Обычно этот файл располагается по адресу /etc, /usr, /etc/my.cnf или /etc/mysql/my.cnf. Вот команды поиска файла конфигурации (иногда следует расширить поиск на другие корневые каталоги):

```
find /etc -name my.cnf
find /usr -name my.cnf
```

Copy

Когда найдете файл, добавьте требуемые переменные в разделе [mysqld]:

```
[mysqld]
; …
slow-query-log = 1
slow-query-log-file = /var/log/mysql/localhost-slow.log
long_query_time = 1
log-queries-not-using-indexes ; тут не надо значение
```

Copy

Изменения вступят только при очередном запуске MySQL, если вам необходимо динамическое изменение параметров, используйте другие методы установки переменных:

```
mysql> SET GLOBAL slow_query_log = 'ON';
mysql> SET GLOBAL slow_query_log_file = '/var/log/mysql/localhost-slow.log';
mysql> SET GLOBAL log_queries_not_using_indexes = 'ON';
mysql> SET SESSION long_query_time = 1;
mysql> SET SESSION min_examined_row_limit = 100;
```
Copy

Проверить значения переменных можно следующим образом:

```
mysql> SHOW GLOBAL VARIABLES LIKE 'slow_query_log';
mysql> SHOW SESSION VARIABLES LIKE 'long_query_time';
```
Copy

Основной недостаток динамической установки — значения будут потеряны при запуске системы. Рекомендуется указывать важные параметры в конфиге MySQL.

Заметка: Синтаксис динамической установки параметров через команду SET и с использованием конфигурационного файла немного различается, например slow_query_log/slow-query-log. В официальной документации СУБД вы найдете полное описание синтаксиса. Формат Option-File используется для файла конфигурации, System Variable Name — имена переменных при динамической установке значений.

Генерация данных для профилирования запроса

Мы рассмотрели основные пункты настройки профилирования, теперь создадим интересующие нас запросы. Этот пример использовался на запущенном MySQL сервере без каких либо предварительных настроек лога. Примеры запросов можно запускать как через MySQL GUI, так и командными средствами СУБД. При мониторинге лога медленных запросов, часто открывают два окна с соединением: одно для запуска запросов, другое — для просмотра лога.

Откройте консоль MySQL и авторизуйтесь от имени SUPER ADMIN. Для начала создайте тестовую БД и таблицу, внесите некоторые данные в неё и включите профилирование. Этот пример следует запускать в окружении разработки:

```
$> mysql -u -p
```
Copy

```
mysql> CREATE DATABASE profile_sampling;
mysql> USE profile_sampling;
mysql> CREATE TABLE users ( id TINYINT PRIMARY KEY AUTO_INCREMENT, name VARCHAR(255) );
mysql> INSERT INTO users (name) VALUES ('Walter'),('Skyler'),('Jesse'),('Hank'),('Walter Jr.'),('Marie'),('Saul'),('Gustavo'),('Hector'),('Mike');

mysql> SET GLOBAL slow_query_log = 1;
mysql> SET GLOBAL slow_query_log_file = '/var/log/mysql/localhost-slow.log';
mysql> SET GLOBAL log_queries_not_using_indexes = 1;
mysql> SET long_query_time = 10;
mysql> SET min_examined_row_limit = 0;
```
Copy

Теперь у нас есть БД с тестовыми данными. Мы запустили профилирование, но время срабатывания и количество строк мы специально установили малым. Чтобы просмотреть лог воспользуйтесь командой:

```
cd /var/log/mysql
ls -l
```
Copy

По идеи, файла лога ещё не должно существовать, так как мы не сделали запросов к БД. Если же он существует, то это означает что профилирование было настроено ранее, а это может исказить результаты примера. Выполните в консоли :

mysql> USE profile_sampling;

mysql> SELECT * FROM users WHERE id = 1;

Copy

Наш запрос использует Primary Key индекс из таблицы. Запрос отработал очень быстро с использованием индекса, поэтому в логе он не должен отразиться. Обратите внимание, файл лога не должен был создаться.

Теперь выполните следующее:

mysql> SELECT * FROM users WHERE name = 'Jesse';

Copy

Здесь мы не использовали индексов. Теперь мы должны увидеть этот запрос в логе:

sudo cat /var/log/mysql/localhost-slow.log

Time: 140322 13:54:58

User@Host: root[root] @ localhost []

Query_time: 0.000303 Lock_time: 0.000090 Rows_sent: 1 Rows_examined: 10

use profile_sampling;

SET timestamp=1395521698;

SELECT * FROM users WHERE name = 'Jesse';

Copy

Рассмотрим ещё один пример. Поднимите планку количества строк в ответе и выполните следующий запрос:

mysql> SET min_examined_row_limit = 100;

mysql> SELECT * FROM users WHERE name = 'Walter';

Copy

Запрос не отразится в логе, так как мы не превысили 100 строк в ответе на запрос.

Заметка: Если данные не отображаются в логе, то в первую очередь нужно рассмотреть следующие факторы. Первое — права на каталог, где хранится файл лога. Группа и пользователь должны соответствовать пользователю mysqld, права должны быть — chmod 755. Второе — возможно профилирование было настроено ранее. Удалите все существующие значения переменных профилирования из файла конфигурации и перезапустите сервер или установите переменные динамически. Если вы использовали динамический метод, то выйдете и зайдите снова в консоль MySQL.

Анализ данных профилирования запросов

Рассмотрим вышеприведенный пример:

Time: 140322 13:54:58

User@Host: root[root] @ localhost []

Query_time: 0.000303 Lock_time: 0.000090 Rows_sent: 1 Rows_examined: 10

use profile_sampling;

SET timestamp=1395521698;

SELECT * FROM users WHERE name = 'Jesse';

Copy

Здесь мы видим:

Время, когда был запущен запрос

Пользователь, который выполнил запрос

Время работы запросы

Продолжительность блокировки

Количество выбранных строк

Количество проанализированных строк

Эти данные очень полезны, так как при их помощи мы сможем найти и устранить причину замедления системы. Так же разработчик или администратор MySQL всегда сможет

увидеть проблемные запросы и хочется отметить, что тут их найти гораздо быстрее, чем изучая программный код приложения. При длительном профилировании можно отследить условия работы при низком быстродействии.

Использование mysqldumpslow

Лог постоянно записывает данные, как правило, пишет он намного больше, чем из него читают. При большом размере лога, читать его становится проблематично. В состав MySQL входит инструмент mysqldumpslow, который помогает сохранить целостность лога. Сама программа совмещена с MySQL (на Linux системах). Для её использования выполните необходимую команду и передайте ей путь до лог файла:

sudo mysqldumpslow -t 5 -s at /var/log/mysql/localhost-slow.log

Copy

Существует ряд параметров помогающих настроить вывод команды. В примере ниже мы увидим пять последних запросов отсортированных по средней продолжительности. В результате читать лог становится намного удобнее. (вывод изменен, чтобы показать реальные значения в логе):

Count: 2 Time=68.34s (136s) Lock=0.00s (0s) Rows=39892974.5 (79785949), root[root]@localhost

SELECT PL.pl_title, P.page_title

FROM page P

INNER JOIN pagelinks PL

ON PL.pl_namespace = P.page_namespace

WHERE P.page_namespace = N

…

Copy

Что мы видим:

Count — количество вхождений запроса в лог

Time — среднее и общее время запроса

Lock — время блокировки таблицы

Rows — Количество выбранных строк

Команда исключает числовые и строковые данные запроса, то есть запросы с одинаковым условием WHERE будут считаться одинаковыми. Благодаря такому инструменту не приходится постоянно просматривать лог. За счет большого количества параметров команды, можно отсортировать вывод как вам удобно. Так же существуют разработки сторонних разработчиков с похожим функционалом, например pt-query-digest.

Разбивка запросов

Следует уделить внимание ещё одному инструменту, который позволяет разбивать сложные запросы. Чаще всего приходится брать запрос из лога, а потом запускать его прямо в консоли MySQL. Сначала надо включить профилирование, а затем выполнить запрос:

mysql> SET SESSION profiling = 1;

mysql> USE profile_sampling;

mysql> SELECT * FROM users WHERE name = 'Jesse';

mysql> SHOW PROFILES;

Copy

После включения профилирования, SHOW PROFILES покажет таблицу связывающую Query_ID и SQL выражение. Найдите соответствующий Query_ID и выполните следующий запрос (замените # на ваш Query_ID):

mysql> SELECT * FROM INFORMATION_SCHEMA.PROFILING WHERE QUERY_ID=#;

Copy

Пример вывода:

SEQ STATE DURATION

1 starting 0.000046

2 checking permissions 0.000005
3 opening tables 0.000036
Copy

STATE — шаг в процессе выполнения запроса, DURATION — продолжительность шага в секундах. Этот инструмент не так уж часто используется, но иногда он может быть чрезвычайно полезен в определении причины снижении скорости запроса.

Детальное описание столбцов:

http://dev.mysql.com/doc/refman/5.5/en/profiling-table.html

Детальное описание шагов:

http://dev.mysql.com/doc/refman/5.5/en/general-thread-states.html

Заметка: этот инструмент не следует использовать в рабочем режиме сервера, за исключением анализа специфических запросов.

Практическое занятие № 21

Тема: Разработка хранимых процедур и триггеров

Цель: Получить практический опыт создания хранимых процедур

Краткие теоретические сведения

Функциональность скриптов T-SQL, которые выполняются на SQL Server, может быть очень сильно расширена при помощи набора хранимых процедур SP_OA (от OleAutomation - SP_OACREATE, SP_OAMETHOD, SP_OASETPROPERTY, SP_OAGETPROPERTY и остальных). При использовании этих хранимых процедур вы можете во время выполнения скрипта создавать программные объекты (COM-серверы, имеющиеся на вашем компьютере), вызывать их свойства и методы, а затем удалять.

Что чаще всего делается при помощи хранимых процедур SP_OA:

- запись результатов выполнения запросов в файл (например, очень удобно выгружать в файл данные в XML), выполнение других дисковых операций;
- запуск внешних программ для загрузки/выгрузки данных (например, пакетов DTS);
- изменение рабочей среды Windows: подключение сетевых дисков и принтеров, создание переменных окружения, изменение параметров в реестре;
- проведение резервного копирования нестандартными способами;
- отслеживание событий операционной системы и приложений на своем и чужих компьютерах;
- работа со службой каталогов (создание/изменение учетных записей пользователей, групп, компьютеров в домене и на локальном компьютере);
- отправка сообщений электронной почты по SMTP (при помощи объектов CDO.Configuration и CDO.Message), что стандартными способами SQL Server делать не умеет;
- использование различных возможностей SQL-DMO (подробнее в соответствующем модуле).

Очень часто также разработчики пишут свои COM-серверы и вызывают их из кода Transact-SQL, таким образом расширяя возможности SQL Server.

Некоторые из этих возможностей доступны при помощи bat - файлов и просто вызова внешних исполняемых файлов при помощи расширенной хранимой процедуры XP_CMDSHELL, однако при помощи SP_OA работать во многих ситуациях удобнее:

- больше функциональность - из скрипта всегда можно вызвать внешний исполняемый файл;
- можно использовать различные свойства и методы программных объектов - нет необходимости ограничиваться только передачей параметров исполняемому файлу;
- нет альтернативы при необходимости получения дополнительной информации по работе операционной системы - пакетными файлами вернуть эту информацию на SQL Server очень сложно (например, информация о свободном месте на диске, наличии учетной записи/членстве ее в группе, информация о работающих программах и службах и т.п.)

Поскольку синтаксис SP_OACREATE нельзя признать самым удобным для написания сложных конструкций и их отладки, то рекомендуется вначале проверять работоспособность программных объектов, доступность их свойств и методов при помощи более специализированных средств, таких, как PrimalScript, а затем уже переносить в код Transact-SQL.

Рассмотрим работу с SP_OA на простом примере. Предположим, что нам необходимо записать результаты выполнения запроса к базе данных на SQL Server в файл output.txt в корневом каталоге диска C:.Запрос будет самый обычный - SELECT companyname FROM northwind.dbo.customers WHERE CustomerID = 'ALKFI'

Если нужно скачать множество значений, то придется использовать цикл с курсорами. Поскольку это сильно усложнит код, а для нашего примера это непринципиально, то мы будем записывать только одно значение.

Пример обычного скрипта Windows, который записывает информацию в текстовый файл на диске:

```
'Создаем объект файловой системы - FSO
Set FSO = Create Object("Scripting.FileSystemObject")
'Создаем объект TextStream (у нас он называется oFile) для записи
'текстовых данных в файл. 8 - значит открыть на добавление данных,
True - 'создать, если еще нет, -1 - писать в Unicode
Set oFile = FSO.OpenTextFile("C:\output.txt", 8, True, -1)
'вызываем метод Write для записи информации в файл
oFile.Write("Наши данные")
'Удаляем созданные нами программные объекты
Set oFile = Nothing
Set FSO = Nothing
```

А теперь переводим то же самое на язык TSQL:

1) вначале - небольшая подготовка: скачиваем результаты запроса в переменную:

```
DECLARE @Varnvarchar(40) Select @Var = CompanyName FROM Customers WHERE CustomerId = 'ALFKI'
```

2) затем создаем объект файловой системы и получаем на него ссылку (handleid) в формате Int - то, что пойдет в переменную FSO. Создание объекта производится при помощи хранимой процедуры SP_OACreate.

```
-- Объявляем переменные (сразу, чтобы не забыть)
DECLARE @FSO int
DECLARE @hrint
DECLARE @srcvarchar(255)
DECLARE @descvarchar(255)
DECLARE @oFileint
-- Создаем сам объект FSO
EXEC @hr = sp_OACreate 'Scripting.FileSystemObject', @FSO OUT
-- и - в соответствии с рекомендациями Microsoft ловим ошибки при помощи --- стандартной конструкции:
IF @hr<> 0
BEGIN
EXEC sp_OAGetErrorInfo @FSO, @src OUT, @desc OUT
SELECT hr=convert(varbinary(4),@hr), Source=@src, Description=@desc
RETURN
END
```

3) теперь при помощи метода OpenTextFile объекта FSO получаем ссылку на еще один объект (типа TxtStream), который будет называться oFile.

Необходимый инструмент - хранимая процедура SP_OAMethod:

```
EXEC @hr = sp_OAMethod @FSO, 'OpenTextFile', @oFileOUT,
```

```
'C:\output.txt', 8 , True, -1
IF @hr<> 0
BEGIN
EXEC sp_OAGetErrorInfo @FSO, @src OUT, @desc OUT
SELECT hr=convert(varbinary(4),@hr), Source=@src, Description=@desc
RETURN
END
```

Обратить внимание на синтаксис процедуры SP_OAMethod: первый параметр - всегда указатель на созданный объект в виде локальной целочисленной переменной; второй параметр - всегда название вызываемого метода; третий параметр - только исходящий, то, что вернет данный метод.

Указывается всегда, если есть хотя бы один входящий параметр. Если метод объекта ничего возвращать не собирается, то обязательно нужно указать NULL.

Далее через запятую, в обычном формате указываем остальные входящие и исходящие параметры.

4) далее еще раз используем процедуру SP_OAMethod, но уже для вызова метода Write объекта oFile. Null для исходящего параметра обязателен!

```
EXEC @hr = sp_OAMethod @oFile, 'Write', NULL, @Var
IF @hr<> 0
BEGIN
EXEC sp_OAGetErrorInfo @FSO, @src OUT, @desc OUT
SELECT hr=convert(varbinary(4),@hr), Source=@src, Description=@desc
RETURN
END
```

Осталось в соответствии с правилами хорошего тона убрать за собой мусор из оперативной памяти:

```
EXEC @hr = sp_OADestroy @FSO
IF @hr<> 0
BEGIN
EXEC sp_OAGetErrorInfo @FSO, @src OUT, @desc OUT
SELECT hr=convert(varbinary(4),@hr), Source=@src, Description=@desc
RETURN
END
EXEC @hr = sp_OADestroy @oFile
IF @hr<> 0
BEGIN
EXEC sp_OAGetErrorInfo @oFile, @src OUT, @desc OUT
SELECT hr=convert(varbinary(4),@hr), Source=@src, Description=@desc
RETURN
END
```

Все, результат запроса добавлен в текстовый файл. Некоторые дополнительные моменты по SP_OA:

- получить значение свойства/изменить значение свойства созданного объекта можно при помощи хранимых процедур SP_OAGetProperty и SP_OASetProperty соответственно. Работа с ними выглядит так же, каки работа с SP_OAMethod, и сложностей не представляет;

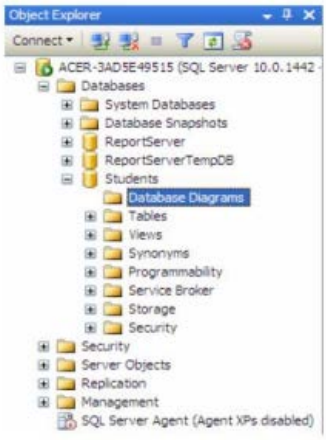
- хранимая процедура SP_OAGetErrorInfo позволяет получить информацию об ошибках, которые могут возникнуть при выполнении операций с внешними программными объектами. Рекомендованный

Microsoft способ применения был приведен выше;

- для всех процедур SP_OA передавать параметры можно только по позиции, но не по их имени;

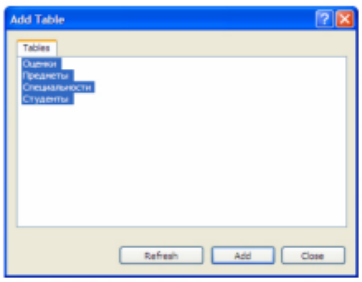
- SP_OADestroy можно, в принципе, не вызывать - созданные при помощи SP_OACreate программные объекты автоматически удаляются из памяти при завершении работы скрипта;

-при первом запуске SP_OACreate в оперативной памяти выделяется специальная программная область - так называемая shared OLE Automationstoredprocedureexecutionenvironment, которая будет жить до перезагрузки сервера. Если по каким-то причинам эту память нужно освободить, то можно выполнить хранимую процедуру SP_OAStop. Однако нужно быть осторожным - если это время выполняется любая другая процедура SP_OA, то она вернет ошибку. Память будет снова выделена автоматически при следующем запуске SP_OACreate.



Создадим диаграмму, обеспечивающую целостность данных БД "Students". Для создания новой диаграммы в БД "Students" щелкните ПКМ по папке "DatabaseDiagrams" и в появившемся меню выберем пункт "NewDatabaseDiagram". Сначала появится окно с вопросом о добавлении нового объекта "Диаграмма".

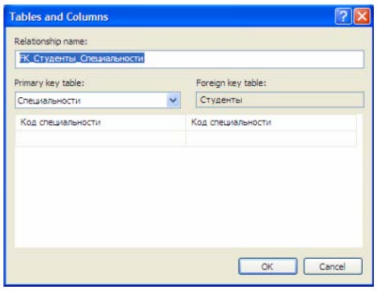
В этом окне нужно нажать кнопку "Yes". Затем появится окно "AddTable" предназначенное для добавления таблиц в новую диаграмму.



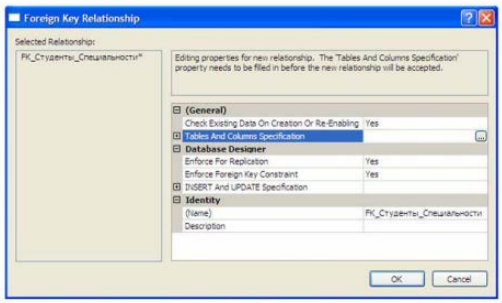
В окне добавления таблиц выделите все таблицы нашей БД и нажмите кнопку "Add". Закройте окно "AddTable" нажатием на кнопку "Close".

Появится окно диаграммы, где будут отображены отобранные таблицы. Теперь необходимо определить связи между таблицами.

Перетащите поле "Код специальности" из таблицы "Специальности" на такое же поле в таблице "Студенты". Появится окно создания связи между таблицами "TablesandColumns".



В окне создания связи нажмите кнопку "Ok". Появится окно настройки свойств связи "ForeignKeyRelationship".

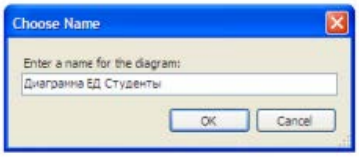


Оставьте свойства связи без изменений и в окне свойств связи нажмите кнопку "Ok".В диаграмме между таблицами "Студенты" и "Специальности" появится связь в виде ломанной линии.

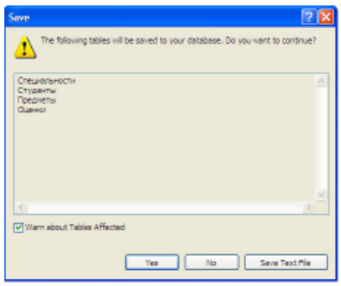
Аналогичным образом создайте связь таблицы "Студенты" с таблицей "Оценки", перетащив поле "Код студента" из таблицы "Студенты" наодноименное поле в таблице "Оценки". Затем, свяжите таблицы "Предметы" и "Оценки", перетащив поле "Код предмета" из таблицы "Предметы" на поля "Код предмета 1", "Код предмета 2" и "Код предмета 3" таблицы "Оценки". После выполнения вышеперечисленных действий диаграмма примет следующий вид.

Закройте окно с диаграммой, щелкнув мышью по кнопке закрытия, расположенной в верхнем правом углу окна с диаграммой. Появится окно с вопросом о сохранении новой диаграммы, где необходимо нажать кнопку "Yes".

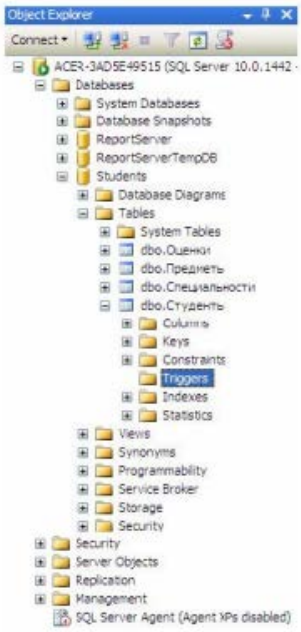
Появится окно определения имени новой диаграммы "ChooseName". В окне определения имени, задайте имя диаграммы как "Диаграмма БД Студенты" и нажмите кнопку "Ok".



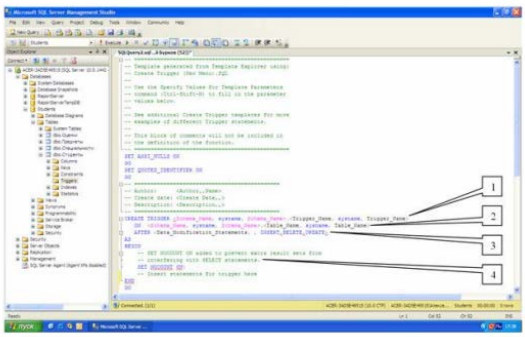
Появится окно "Save" с запросом сохранения таблиц, входящих в диаграмму. В данном окне необходимо нажать кнопку "Yes".



Перейдем к созданию триггеров. Создадим триггеры для таблицы "Студенты". Триггеры создаются отдельно для каждой таблицы и располагаются в обозревателе объектов в папке "Triggers". В нашем случае, папка "Triggers" входит в состав таблицы "Студенты".

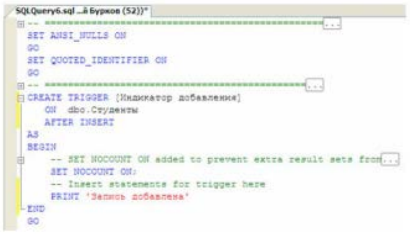


Для начала создадим триггер, выводящий сообщение "Запись добавлена" при добавлении записи в таблицу "Студенты". Создадим новый триггер, щелкнув ПКМ по папке "Triggers" в таблице "Студенты" и в появившемся меню выбрав пункт "NewTrigger". Появится следующее окно с новым триггером:



Рассмотрим структуру триггеров: Область определения имени функции (Trigger_Name); Область, показывающая для какой таблицы создается триггер (Table_Name); Область, показывающая когда выполнять триггер (INSERT - при создании записи в таблице, DELETE - при удалении и UPDATE - при изменении) и как его выполнять (AFTER - после выполнения операции, INSTEAD OF - вместо выполнения операции); Тело триггера, содержит команды

языка программирования запросов T-SQL. В окне нового триггера наберите код как показано на рисунке.



Практическое занятие № 22

Тема: Управление правами доступа к базам данных

Цель: Получить навыки работы по назначению прав и установки уровня доступа для пользователей MS SQL Server 2008

Аутентификация — это процесс проверки права пользователя на доступ к тому или иному ресурсу. Чаще всего аутентификация осуществляется с помощью ввода имени и пароля.

SQL Server 2008 поддерживает два режима аутентификации: с помощью Windows и с помощью SQL Server. Первый режим позволяет реализовать решение, основанное на однократной регистрации пользователя и едином пароле при доступе к различным приложениям (*Single SignOn solution, SSO*). Подобное решение упрощает работу пользователей, избавляя их от необходимости запоминания множества паролей и тем самым снижая риск их небезопасного хранения (вспомним стикеры с паролями, наклеенные на мониторы). Кроме того, данный режим позволяет использовать средства безопасности, предоставляемые операционной системой, такие как применение групповых и доменных политик безопасности, правил формирования и смены паролей, блокировка учетных записей, применение защищенных протоколов аутентификации с помощью шифрования паролей (Kerberos или NTLM).

Аутентификация с помощью SQL Server предназначена главным образом для клиентских приложений, функционирующих на платформах, отличных от Windows. Этот способ считается менее безопасным, но в SQL Server2008 он поддерживает шифрование всех сообщений, которыми обмениваются клиент и сервер, в том числе с помощью сертификатов, сгенерированных сервером. Шифрование также повышает надежность этого способа аутентификации.

Задание на лабораторную работу

1. В утилите SQL Server Management Studio выполнить примеры, которые даны по ходу работы.
2. Создать базу данных 3-4 сущности согласно предметной области из таблицы «Индивидуальное задание». Номер варианта соответствует порядковому номеру студента в журнале.
3. *По индивидуальному варианту базы данных определить 2-3 должностных лица,* которые могут работать с таблицами БД. Для каждого должностного лица определить набор привилегий, которыми он может пользоваться.
4. В утилите SQL Server Management Studio создать под каждое должностное лицо соответствующую роль, наделить эту роль определенными привилегиями. Далее создать по одному пользователю на каждую должность и присвоить им соответствующие роли.
5. Сохранить последовательно операторы с указанием заданий в файле с названием ФамилияСтудента_Лаб_5_№варианта.

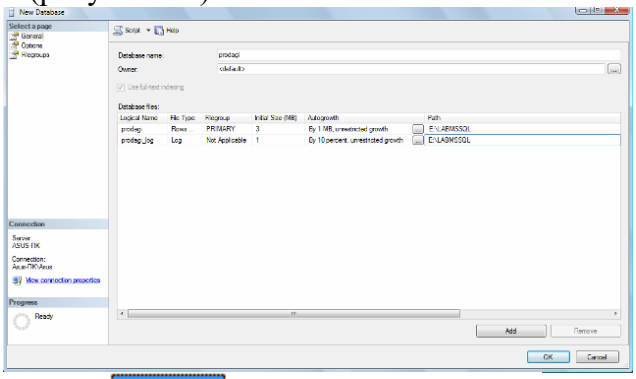
Ход работы

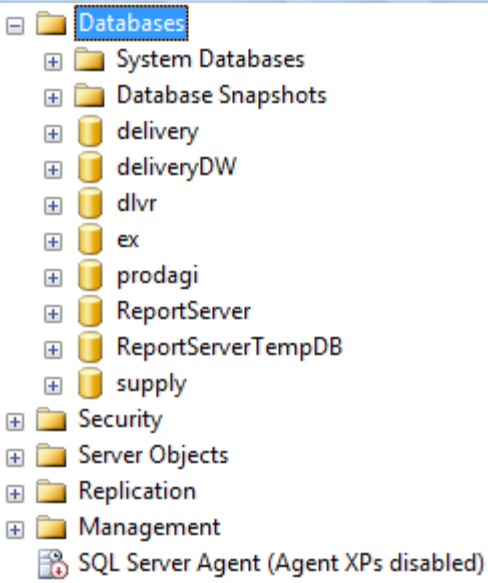
i. **Создание базы данных**

1. Создать на диске компьютера (D:, E: и т.п.) каталог с произвольным именем (например, E:\LABMSSQL)
2. Запустить Microsoft SQL Server Management Studio, для чего:

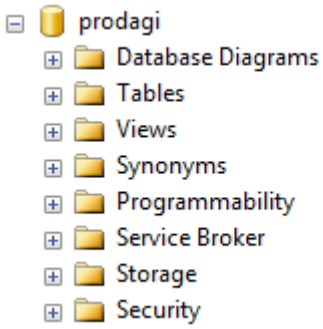
- в панели задач выбрать пункт Microsoft SQL Server 2008;
- выбрать подпункт SQL Server Management Studio;
- в окне подключения (рисунок 5.1) нажать кнопку Connect;

После появления на экране среды Microsoft SQL Server Management Studio в окне Object Explorer выбрать пункт Databases, нажать правую кнопку мыши и в появившемся меню выбрать пункт New Database…. В результате на экране появится окно, позволяющее ввести основные параметры новой базы данных. Необходимо ввести имя новой базы данных – prodagi и определить место размещения файлов - E:\LABMSSQL (рисунок 5.2). После ввода данных нажать кнопку OK. Новая база данных появится в списке баз данных в окне Object Explorer (рисунок 5.3).

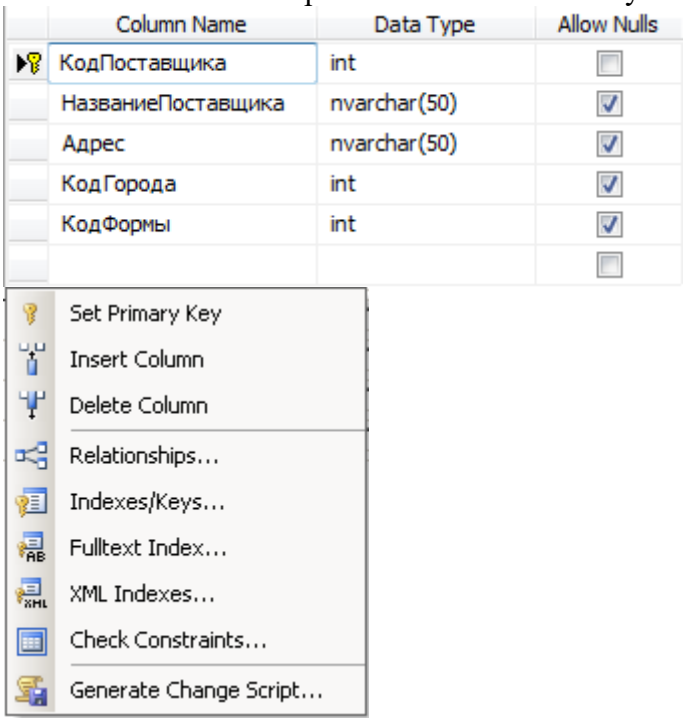




Выбрать созданную базу данных и раскрыть список ее объектов (рисунок 5.4).



В списке объектов базы данных щелкнуть правой кнопкой мыши по пункту Tables и в появившемся меню выбрать пункт New Table…. Ввести поля новой таблицы (рисунок 5.5), определив при этом типы данных и ключевое поле (для этого нужно щелкнуть по полю правой кнопкой мыши и выбрать в меню соответствующий пункт (рисунок 5.6)).



6. **Закрыть вкладку со структурой новой таблицы. Сохранить новую таблицу с именем «Поставщики» (без кавычек).**

Добавление пользователя в БД

1. Откройте проект базы данных.
2. Если **Представление схемы** не отображается, в меню **Вид** выберите пункт**Представление схемы базы данных**.
3. В **представлении схемы** щелкните правой кнопкой мыши папку "Безопасность", выберите команду **Добавить**, а затем щелкните**Пользователь**.

Появляется диалоговое окно **Добавление нового элемента**. В поле **^ Имя** введите имя пользователя, которого требуется создать.

4. Нажмите кнопку **Добавить**.

Пользователь создан и добавлен в проект базы данных. В **обозревателе решений** отображается файл, содержащий определение этого пользователя. Объект базы данных для нового пользователя отображается в **представлении схемы**.

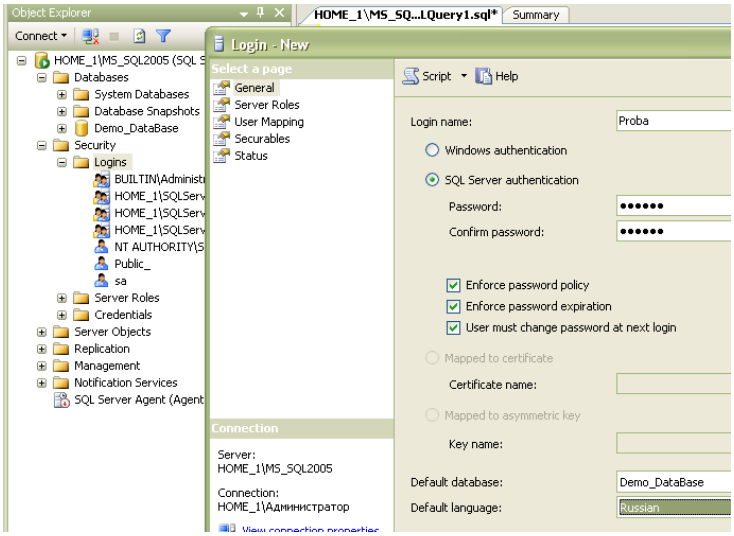


Рисунок 5.7 Раздел Security для работы с пользователями и создание нового пользователя (при SQL Server аутентификации нужно снять галочки с **Enforce password policy**)

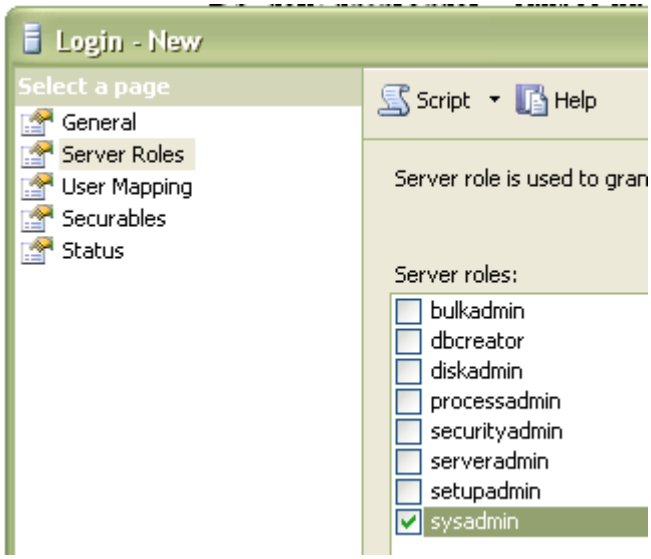


Рисунок 5.8 Настройка серверной роли для нового пользователя (весь список серверных ролей с их привилегиями в конце работы)

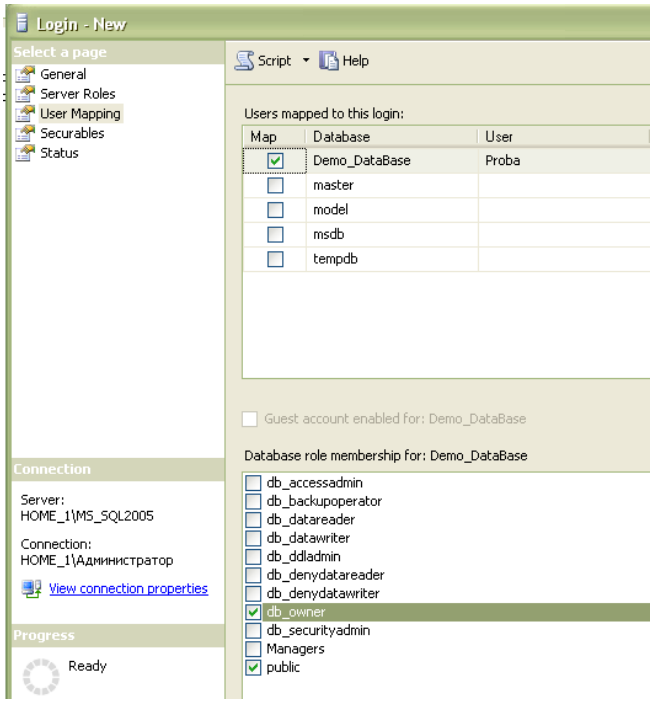


Рисунок 5.8 Настройка серверной роли для нового пользователя (весь список серверных ролей с их привилегиями в конце работы)

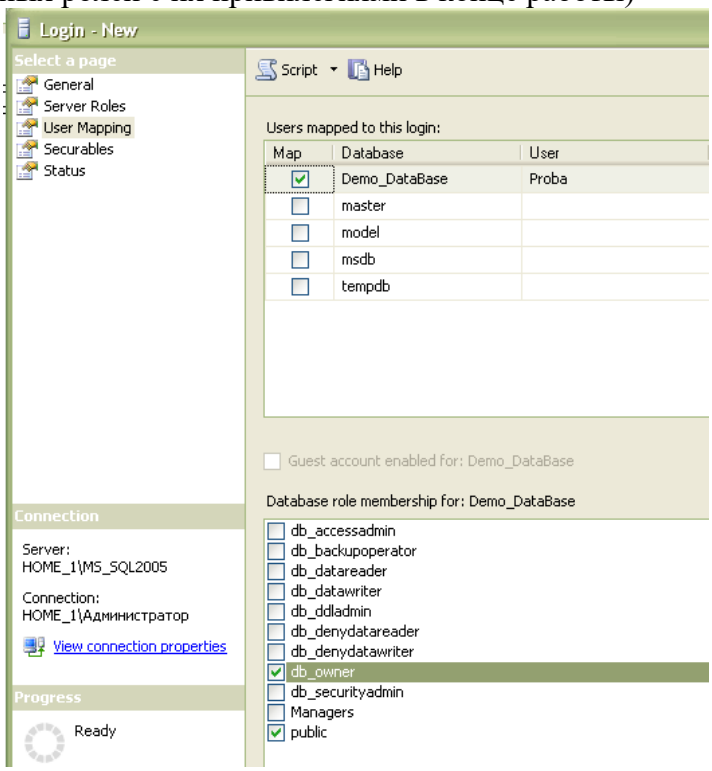


Рисунок 5.9 Настройка роли базы данных для нового пользователя (весь список ролей баз данных с их привилегиями ниже)

Перечень ролей БД:

Public – минимальные права доступа к БД (на просмотр)

Db_owner – может выполнять любые действия с БД

Db_accessadmin – добавляет и удаляет пользователей БД

Db_sequrityadmin – управляет ролями в Бди разрешениями на запуск команд и работу с объектами БД

Db_ddladmin – добавляет, изменяет и удаляет объекты БД

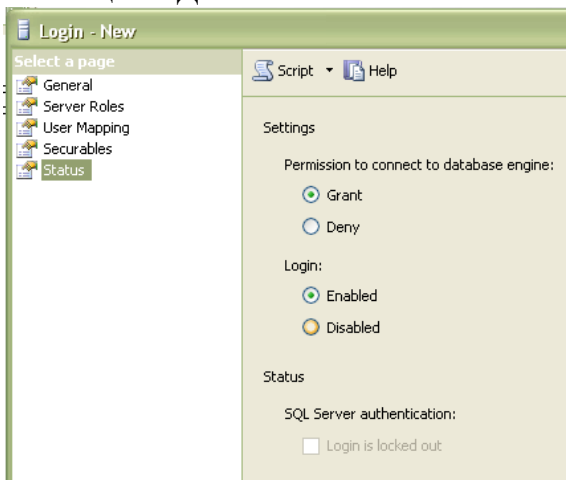
Db_backupoperator – осуществляет резервное копирования БД

Db_datareader – может просматривать все данные в каждой таблице в БД

Db_datawriter - может добавлять, удалять и изменять данные в каждой таблице в БД

Db_denydatareader – запрет на просмотр всех данных в каждой таблице в БД

Db_denydatawriter - запрет на добавление, удаление и изменение всех данных в каждой таблице в БД



Практическое занятие № 23.

Тема: Аудит данных с помощью средств СУБД и триггеров

Цель: Изучить способы создания триггеров на языке Transact-SQL, приобрести практические навыки разработки и использования триггеров в приложениях баз данных.

Задания

1. Изучите приведенный в лабораторной работе теоретический материал.
2. Изучите приведенные в лабораторной работе примеры создания триггеров.
3. Выполните индивидуальное задание (см. после теоретических сведений):

- создание DML-триггеров;
- создание DDL-триггера;
- получить метаинформацию по создаваемым триггерам.

4. Покажите работу преподавателю.
5. Ответьте на контрольные вопросы.

Краткие теоретические сведения

Триггер – это специальный тип хранимых процедур, запускаемых сервером автоматически в ответ на изменение таблиц или представлений.

Каждый триггер привязывается к конкретной таблице или представлению.

Триггеры применяются для обеспечения целостности данных и реализации сложной бизнес-логики, а также создания записи аудита о модификации данных

Все производимые триггером модификации данных рассматриваются как одна транзакция. В случае обнаружения ошибки или нарушении целостности данных происходит откат этой транзакции. Тем самым внесение изменений будет запрещено. Будут также отменены все изменения, уже сделанные триггером.

Типы триггеров

Триггеры подразделяются на триггеры языка определения данных (DataDefinitionLanguage – DDL), на триггеры языка манипулирования данными (DataManipulationLanguage – DML) и триггеры входа. Последний тип триггера запускается при регистрации в экземпляре SQL Server.

Триггеры DML используют логические (концептуальные) таблицы DELETED и INSETED. По своей структуре они подобны таблице, на которой определен триггер, то есть таблице, к которой применяется действие пользователя. В таблицах DELETED и INSETED содержатся старые или новые значения строк, которые могут быть изменены действиями пользователя.

По типу изменения данных в таблице существует четыре типа триггеров:

- Insert trigger – триггеры запускаются при попытке вставки данных с помощью команды insert;
- UPDATE TRIGGER – триггеры запускаются при попытке изменения данных с помощью команды UPDATE;
- DELETE TRIGGER – триггеры этого типа запускаются при попытке удаления данных с помощью команды DELETE;
- Триггеры, создаваемые с учетом одновременного возникновения и совпадения событий.

По типу поведения триггеры классифицируются на триггеры FOR (AFTER) и INSTEAD OF.

Триггер FOR (AFTER) выполняется после выполнения команды, изменяющей данные в таблице. Если же команда по каким-либо причинам не может быть успешно завершена, то триггер также не выполняется. Изменения данных в результате выполнения запроса пользователя и выполнение триггера осуществляется в теле одной транзакции. Если триггер не выполняется, то также будет выполнен откат и пользовательских изменений.

По умолчанию в SQL Server 2008 R2 все триггеры являются триггерами типа FOR (AFTER).

Триггеры типа FOR (AFTER) невозможно определить для представлений. Для каждой таблицы можно определить более одного триггера AFTER для каждой операции (INSERT, UPDATE, DELETE).

Триггер INSTEAD OF выполняется в обход действий, вызывавших их срабатывание, заменяя эти действия. Например, обновление таблицы, в которой есть триггер INSTEAD OF, вызовет срабатывание этого триггера. В результате вместо оператора обновления выполняется код триггера. Это позволяет размешать в триггере сложные операторы обработки, которые дополняют действия оператора, модифицирующего таблицу.

Триггеры INSTEAD OF могут быть определены для таблиц и представлений. Можно определить только один триггер INSTEAD OF для каждой операции (INSERT, UPDATE, DELETE).

Триггеры INSTEAD OF не разрешены для обновляемых представлений, использующих параметр WITH CHECK OPTION. SQL Server вызывает ошибку, если триггер INSTEAD OF добавляется к обновляемому представлению с параметром WITH CHECK OPTION. Пользователь должен удалить этот параметр при помощи инструкции ALTER VIEW перед определением триггера INSTEAD OF.

Нельзя устанавливать триггеры на временные или системные таблицы, хотя на такие таблицы разрешено ссылаться в операторах T-SQL самого триггера.

Нельзя триггеры INSTEAD OF DELETE и INSTEAD OF UPDATE определять в таблицах, где заданы ограничения каскадной ссылочной целостности ON DELETE или ON UPDATE соответственно.

К таблице разрешено привязывать триггеры обоих классов: INSTEAD OF и FOR (AFTER). Если в таблице определены ограничения и триггеры обоих классов, то первым из них срабатывает триггер INSTEAD OF, затем обрабатываются ограничения и последним срабатывает FOR (AFTER)-триггер. При нарушении ограничения выполняется откат действий INSTEAD OF-триггера. Если нарушаются ограничения или происходят какие-либо другие события, не позволяющие модифицировать таблицу, FOR (AFTER)-триггер не исполняется.

Как и в случае хранимых процедур, глубина вложенности триггеров достигает 32 уровней, также возможно рекурсивное срабатывание триггеров

С помощью системной хранимой процедуры sp_settriggerorder возможно указать, какой триггер будет выполняться первым, а какой последним. Однако назначить триггер, который будет выполняться вторым, третьим и т. д. (но не последним) нельзя.

Синтаксис оператора создания триггера

CREATETRIGGER<имя триггера>

ON[<имясхемы.>]{ <имя таблицы | <имя представления> }

[WITH ENCRYPTION | EXECUTE AS {<CALLER | SELF | <USER>>}]
{ FOR | AFTER | INSTEAD OF }
{ [DELETE] [,] [INSERT] [,] [UPDATE] }
[WITH APPEND]
[NOT FOR REPLICATION]
AS
{sql_statement| EXTERNAL NAME <method specifier> }

Имя схемы – имя схемы, которой принадлежит триггер DML. Триггеры DML ограничены областью схемы таблицы или представления, для которых они созданы. Аргумент <имя схемы> не может быть указан для триггеров DDL или входа.

Имя триггера идентификатор триггера должен соответствовать правилам для идентификаторов, за исключением того, что trigger_name не может начинаться с символов # или ##.

Талица|Представление – таблица или представление, в которых выполняется триггер DML.

WITH ENCRYPTION – шифрует и делает недоступным (включая администратора) текст инструкции CREATETRIGGER. Использование аргумента WITH ENCRYPTION не позволяет публиковать триггер как часть репликации SQL Server. Параметр WITH ENCRYPTION не может быть указан для триггеров CLR.

EXECUTE AS – указывает контекст безопасности, в котором выполняется триггер. Позволяет управлять учетной записью пользователя, используемой экземпляром SQL Server для проверки разрешений на любые объекты базы данных, ссылаемые триггером.

FOR | AFTER – тип AFTER указывает, что триггер DML срабатывает только после успешного выполнения всех операций в инструкции SQL, запускаемой триггером. Все каскадные действия и проверки ограничений, на которые имеется ссылка, должны быть успешно завершены, прежде чем триггер сработает.

INSTEAD OF – указывает, что триггер DML срабатывает вместо инструкции SQL, используемой триггером, переопределяя таким образом действия выполняемой инструкции триггера. Аргумент INSTEAD OF не может быть указан для триггеров DDL или триггеров входа.

{ [DELETE] [,] [INSERT] [,] [UPDATE] } – определяет инструкции изменения данных, по которым срабатывает триггер DML, если он применяется к таблице или представлению. Необходимо указать как минимум одну инструкцию. В определении триггера разрешены любые их сочетания в любом порядке.

Для триггеров INSTEAD OF параметр DELETE не разрешен в таблицах, имеющих ссылочную связь с указанием каскадного действия ON DELETE. Точно так же параметр UPDATE не разрешен в таблицах, имеющих ссылочную связь с указанием каскадного действия ON UPDATE.

WITH APPEND – указывает, что требуется добавить триггер существующего типа. Аргумент WITH APPEND не может быть использован для триггеров INSTEAD OF или при явном указании триггера AFTER. Аргумент WITH APPEND может использоваться только при указании параметра FOR без INSTEAD OF или AFTER из соображений поддержки обратной совместимости. Аргумент WITH APPEND не может быть указан, если указан параметр EXTERNAL NAME (в случае триггера CLR).

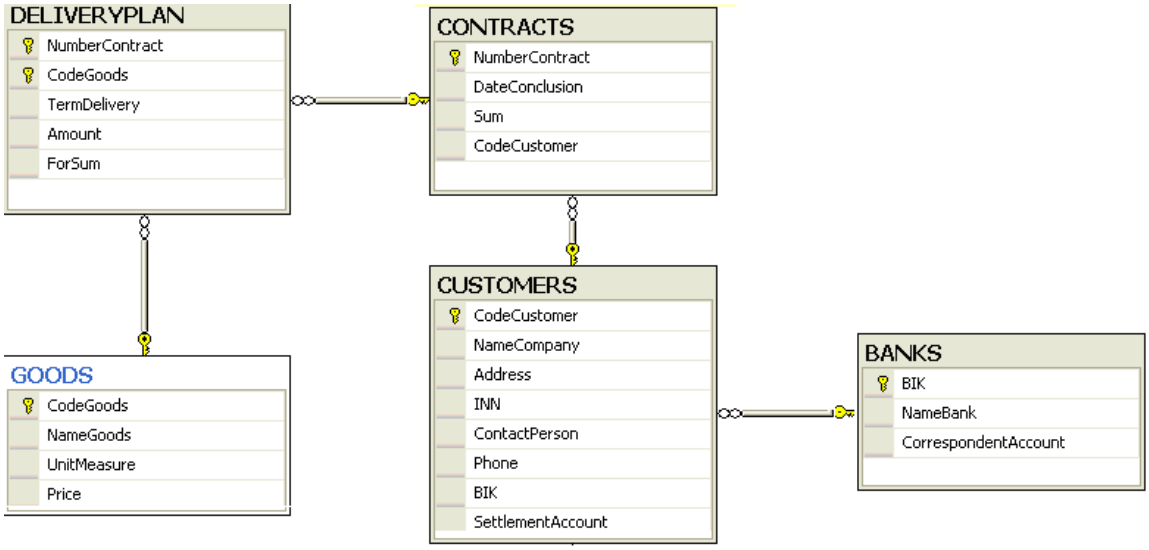
Предложение будет удалено в следующей версии Microsoft SQL. Не следует использовать его при создании новых приложений

NOT FOR REPLICATION– при создании триггера с этим параметром запрещается его запуск при модификации таблиц механизмами репликации. Этот параметр часто используется при создании системных триггеров поддержки подсистемы репликации.

sql_statement –набор команд, которые будут выполнены при запуске триггера (тело триггера).

Примеры триггеров

Примеры триггеров приведены для базы данных, диаграмма которой представлена на рис. 1.



1. Запретить вставлять новые строки в таблицу BANKS, выводя при этом сообщение «Вставка строк запрещена»:
   CREATE TRIGGER BANKS_zapINS
   ON dbo.BANKS
   FOR INSERT AS
   PRINT 'Вставка строк запрещена'
   ROLLBACK TRAN
2. Не заключать ДОГОВОРА с ПОКУПАТЕЛЯМИ, если на фирме прекращена поставка товаров на склад.
   CREATE TRIGGER noDeliveryPlan
   ON dbo.DeliveryPlan
   FOR INSERT, UPDATE
   AS
   IF EXISTS
   (SELECT *
   FROM Inserted I
   JOIN dbo.GOODS G ON I.CodeGoods = G.CodeGoods
   WHERE G.DiscontinuedDate IS NOT NULL)
   BEGIN
   RAISERROR ('ПРЕКРАШЕНА ПОСТАВКА ТОВАРА’,16,1)
   ROLLBACKTRAN
   END
3. Запретить поставки товары ПОКУПАТЕЛЮ, если потребуется отпустить более половины запаса товара.
   CREATE TRIGGER controlAmount
   ON dbo.GOODS
   FOR UPDATE
   AS
   IF EXISTS
   (SELECT *
   FROM INSERTED I JOIN DELETED DON I.CodeGoods = D.CodeGoods
   WHERE (D.Amount-I.Amount)>D.Amount/2 AND
   (D.Amount- I.Amount)>0)
   BEGIN

RAISERROR (Запрещено отпускать товар в количестве, превышающим 50%% товарного запаса',16,1)
ROLLBACKTRAN
END

4. Создать для таблицы DELIVERYPLAN триггер типа DELETE, который будет выводить информацию о попытках удаления и количестве удаляемых строк:
   CREATE TRIGGER del_DeliveryPlan
   ON DeliveryPlan
   FOR DELETE
   AS
   PRINT 'Попыткаудаления '+STR(@@ROWCOUNT)+' строкизтаблицыDeliveryPlan'
   PRINT 'Пользователь '+ CURRENT_USER
   IF CURRENT_USER!= 'dbo'
   BEGIN
   PRINT 'Удаление запрщено'
   ROLLBACK TRANSACTION
   END
   ELSEPRINT 'Удаление разрешено'

Созданный триггер будет выводить информацию о количестве строк, которое пытается удалить пользователь, и имя пользователя, выполнившего команду DELETE. Если пользователь не 'dbo', то удаление запрещается и выдается соответствующее предупреждение.

5. Изменение (модификация) триггера.

Изменить триггер BANKS_zapINS так, чтобы было разрешена вставка строки в таблицу BANKS_zapINS, но при этом дополнительно записывал в специальную таблицу все данные вставляемой строки с указанием имени пользователя, пытавшегося вставить строку, а также времени вставки данных.

Для этого нам необходимо создать в базе данных дополнительную таблицу, в которой будет храниться список вставляемых строк. Выполним это с помощью следующей команды:

SELECT TOP 0 *, suser = SUSER_SID(), _date = GETDATE()
INTO BANKS_buffer FROM BANKS

Теперь изменим триггер:

ALTER TRIGGER BANKS_zapINS ON BANKS FOR INSERT AS
INSERT INTO BANKS_buffer SELECT *, SUSER_SID(), GETDATE()
FROMinsertedPRINT 'Операция вставки строки зафиксирована'

6. Удалить триггер
   DROP TRIGGER BANKS_zapINS

Получение информации о триггере

1. Получить код Transact-SQL, выполняемого при вызове триггера:
   sp_helptext [@objname =] 'name'
   name – имя триггера, о котором необходимо получить информацию.
2. Получить список триггеров, определенных для конкретной таблицы базы данных, используется следующая хранимая процедура:
   sp_helptrigger [@tabname =] 'table' [,[@triggertype =] 'type']
   table – имя таблицы, для которой нужно получить список созданных триггеров.

type – определяет тип триггеров, о которых будет выведена информация. Если этот аргумент опущен, то будет возвращен список всех триггеров.

Список столбцов возвращаемого результата и их назначение:

TRIGGER_NAME (sysname) – имя триггера, присвоенное ему при создании или после переименования;

TRIGGER_OWNER (sysname) – имя владельца триггера;

ISUPDATE (int) – значение 1 означает, что триггер будет вызываться при выполнении команды UPDATE;

ISDELETE (int) – значение 1 означает, что триггер будет вызываться при выполнении команды DELETE;

ISINSERT (int) – значение 1 означает, что триггер будет вызываться при выполнении команды INSERT.

3. Просмотр списка объектов, от которых зависит триггер:

   sp_depends [@objname] 'object'

   object – содержит имя триггера, о котором необходимо получить информацию.

Возвращаемый результат разделен на две таблицы: первая для объектов, от которых зависит триггер, вторая – для объектов, зависящих от триггера.

Список столбцов первой таблицы следующий:

NAME (nvarchar(40)) – имя объекта, от которого зависит триггер;

TYPE (nvarchar(16)) – тип объекта, от которого зависит триггер;

UPDATED (nvarchar(9)) – определяет, является ли объект изменяемым;

SELECTED (nvarchar(8)) – определяет, включается ли объект в результат выборки SELECT;

COLUMN (sysname) – имя столбца или другого параметра, от которого конкретно зависит триггер.

Список столбцов второй таблицы:

NAME (nvarchar(40)) — имя объекта, который зависит от триггера;

TYPE (nvarchar(16)) — тип объекта, который зависит от триггера.

Создание триггера в среде MS SQL Server Management Studio

Для создания триггера необходимо вызвать соответствующий пункт контекстного меню объекта **Триггеры базы данных** в папке **Программирование**. В правой части окна среды появиться шаблон триггера.

Индивидуальная работа

1. Создайте триггеры для таблиц проектируемой БД данных, используя окно редактора запросов, и проверьте их работу.
   a. триггер на добавление записи в одну из таблиц БД с выводом сообщения об этом событии;
   b. триггер, запускаемый при занесении новой строки в одну из таблиц БД. Триггер должен увеличивать счетчик числа добавленных строк;
   c. триггер, запускаемый при удалении записи в родительской таблице и запрещающий ее удаление, если есть связанные с ней записи в дочерней таблице;
   d. триггер, запрещающий ввод записи в дочернюю таблицу, если значение поля внешнего ключа не совпадает ни с одним значением первичного ключа родительской таблицы. Обеспечить вывод сообщения об этом событии;
   e. создайте триггер, который при вводе записи в таблицу, имеющую вычисляемое поле, вычисляет это поле (если такой таблицы нет, то согласуйте задание с преподавателем).
2. Получить список триггеров, определенных для конкретной (заданной) таблицы БД. Дать комментарии по возвращаемому результату.
3. Создание триггера DDL и его тестирование.

a. Создайте в проектируемой БД таблицу Test, содержащую один столбец с именем ID. Тип данных столбца – целые числа; неопределенные значения в столбце не допустимы.
b. Введите в таблицу 2-3 записи, используя оператор INSERT.
c. Создайте триггер DDL:

```
CREATE TRIGGER ddl_drop_table_test
ON DATABASE
FOR DROP_TABLE
AS
PRINT 'Вы пытаетесь удалить таблицу в базе данных.'
```

PRINT 'Если Вы действительно хотите удалить таблицу, то отключите триггер DDL.'
PRINT 'После того, как таблица будет удалена, необходимо вновь включить триггер'
ROLLBACK TRANSACTION

d. Создайте и запустите на исполнение запрос на удаление таблицы Test.
e. Самостоятельно изучите синтаксис операторов отключения и активирования триггера.
f. Выполните последовательно команды:

- отключить триггер ddl_drop_table_test;
- удалить таблицу Test;
- проверить, что удаление таблицы завершилось успешно, создав запрос на выборку из таблицы Test;
- активировать триггер ddl_drop_table_test;
- проверить работу триггера при попытке удаления какой-либо таблицы проектируемой Вами базы данных.

Контрольные вопросы

1. Что такое ограничения целостности?
2. Перечислите типы ограничений целостности.
3. Какие ограничения целостности можно поддержать с помощью триггеров?
4. При каких изменениях в базе данных активизируются триггеры DDL?
5. Можно ли действия, выполняемые триггером, закодировать в хранимой процедуре?
6. В чем заключаются отличия триггеров и хранимых процедур?
7. Дайте комментарии по синтаксису оператора CREATE TRIGGER.
8. Каково назначение таблиц INSERTED и DELETED?
9. Какая системная процедура позволяют получить код триггера?
10. Как узнать от каких объектов базы данных зависит триггер?
11. Как получить список триггеров конкретной таблицы?

Практическое занятие № 24

Тема: Резервное копирование и восстановление баз данных

Цель: ознакомиться с основными конструкциями SQL, технологиями среды MS SQL Server Management, объектами SMO (среды MS Visual Studio) для резервного копирования и восстановления БД.

Задание №1. необходимо создать резервные копии базы данных «MMM» с использованием полного резервного копирования, разностного резервного копирования и резервного копирования журнала транзакций.

Ход работы:

1. Запустите SQL Server Management Studio (SSMS), подключитесь к своему экземпляру SQL Server, используя технологию 1.
2. Создайте папку с именем c:\Student\ВашаПапка\test.
3. Откройте окно нового запроса. Измените контекст на базу данных master, используя технологию 6. Наберите и исполните следующую команду, чтобы создать полную резервную копию базы данных:

   BACKUP DATABASE MMM TO DISK = 'C:\.....TEST\AW.BAK’

Ознакомьтесь с результатами запроса – какая информация обработана, сколько страниц, сколько файлов.

4. Внесите изменение в таблицу «Модель» базы данных MMM. Добавьте одну запись (придумайте сами)/
5. Откройте окно нового запроса наберите и исполните следующую команду, чтобы создать резервную копию журнала транзакций и сохранить только что внесенное изменение:

   BACKUP LOG MMM TO DISK = ‘C:\.....TEST\AW1.TRN'

Ознакомьтесь с результатами запроса – какая информация обработана, сколько страниц, сколько файлов.

6. Внесите еще одно изменение в таблицу«Модель».

7. Откройте окно нового запроса наберите и исполните следующую команду, чтобы создать разностную резервную копию базы данных:

BACKUP DATABASE MMM TO DISK = ‘C:\.....\TEST\AWDIFF1.BAK' WITH DIFFERENTIAL

Ознакомьтесь с результатами запроса – какая информация обработана, сколько страниц, сколько файлов.

8. Внесите еще одно изменение в таблицу«Модель».
9. Откройте окно нового запроса наберите и исполните следующую команду, чтобы создать полную резервную копию базы данных в указанном месте на диске:

BACKUP LOG MMM TO DISK = ‘C:\....TEST\AW2.TRN'

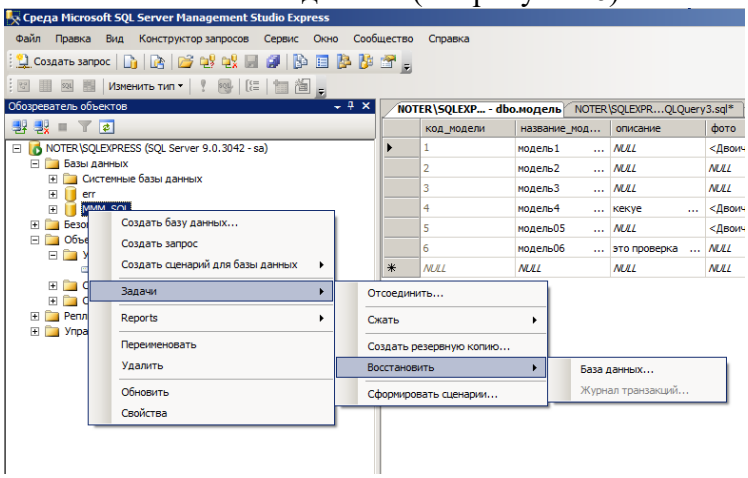
Ознакомьтесь с результатами запроса – какая информация обработана, сколько страниц, сколько файлов.

Задание №2. необходимо провести восстановление базы данных «MMM» из сделанных в задании №1 резервных копий.

Ход работы:

1. Если необходимо, запустите SSMS, подключитесь к своему экземпляру SQL Server, используя технологию 1.
2. Выполните восстановление БД из первой полной резервной копии (C:\...TEST\AW.BAK) средствами оболочки SSMS. Для этого выполните:

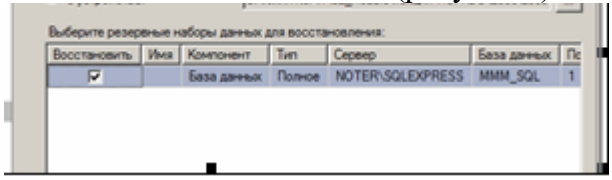
- В обозревателе объектов вызовите контекстное меню на вашей БД и выберите задачу восстановления базы данных (см. рисунок 6).



- В открывшемся окне необходимо задать следующие параметры восстановления

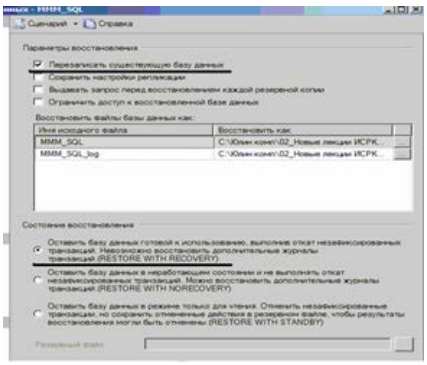
На закладке «Общие» необходимо выбрать:

1. Базу данных для восстановления (вашу MMM)
2. Выбрать источник набора данных для восстановления с устройства ◊ файл C:\...TEST\AW.BAK
3. После определения файла-источника данных необходимо флажком выбрать базу данных для восстановления (рисунок 7).



На закладке «Параметры»

1. необходимо включить опцию «Перезаписать БД» и «оставить БД готовой к использованию», (рисунок 8).



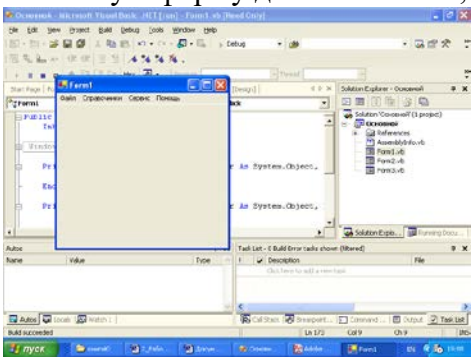
3. Нажмите ОК
4. После восстановления БД, откройте таблицу «Модель» и убедитесь, что она не содержит всех добавлений, вносимых вами в процессе выполнения упражнения, так как восстановление происходило из первой резервной копии (без изменений).

Задание №3. необходимо организовывать со стороны клиентского приложения, созданного в Visual Studio удаленное администрирование БД (резервное копирование).

Ход работы:

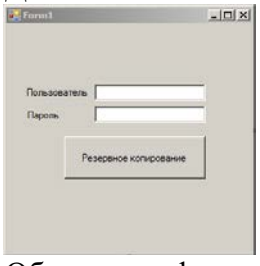
В Visual Studio

1. Создайте новый проект Windows Application и сохраните его в своей папке под именем Лабы_MMM_2 семестр.
2. В главную форму добавьте меню, изображенное на рисунке 9:



Файл (Открыть, Закрыть, Выход)
Справочники (Модель, Магазин, Дерево моделей)
Заказы (Работа с заказами)
Отчеты (Прайс-лист, Бланк заказов)
Администрирование БД (Резервное копирование, Безопасность)
Сервис (Калькулятор)
Помощь (Справка, О программе)

3. Добавьте новую форму в проект
4. Добавьте на только что созданную форму компоненты в соответствии с рисунком 10.



5. Обеспечьте функциональную работу формы (напишите обработчик кнопки «Резервное копирование» с использованием объектов SMO. Описание объектов SMO, их свойств и методов см. в лекционном материале.)
6. Добавьте возможность открытия данной формы при выборе в главной форме пункта меню Администрироване БД ◊ Резервное копирование
7. Запустите проект, проверьте работу формы.

8. Закройте проект
9. Убедитесь в появлении файла резервной копии на диске (файл, который указан в тексте программы).
10. Откройте SSMS. Добавьте в таблицу «Модель» новую строку данных (самостоятельно).
11. Средствами оболочки SSMS, выполните восстановление БД из резервной копии, созданной вашей программой
12. Убедитесь, что после восстановления добавленных строк в таблице «Модель» нет.

Задание №4. Ответьте на вопросы теста и представьте результаты преподавателю.

1. Вы выполняете разностное резервное копирование базы длимых AdveniurtWorks каждые четыре часа, начиная с 04:00. полная резервная копия создается в полночь. Какие данные будут содержаться в разностной резервной копии сделанной в полдень?
    1. A Страницы данных, измененные после полуночи.
    2. B. Экстенты, измененные после полуночи.
    3. C. Страницы данных, измененные после 08:00
    4. D. Экстенты. измененные после 08:00.
2. Вы выполняете полное резервное копирование 6азы данных Adventure Wortks,, которое завершается в полночь. Разностное резервное копирование выполняется по расписанию каждые четыре часа, начиная с 04:00. Резервное копирование журнала транзакций происходит по расписанию каждые пять минут. Какую информацию будет содержать резервная копия журнала транзакций, созданная в 09:15?

1. A. Все транзакции, начатые после 09:10.
2. B. Транзакции, завершенные после 09:10.
3. C. Страницы, измененные после 09:10.
4. D. Экстенты, измененные после 09:10.

ПЕРЕЧЕНЬ РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1.1. Печатные и (или) электронные учебные издания (включая учебники и учебные пособия)

1. Батаев А. В. Операционные системы и среды: учебник для студ. учреждений сред.проф. образования / А. В. Батаев, Н. Ю. Налютин, С. В. Синицын – М.: Издательский центр «Академия», 2019 – 272 с.

2. Партыка, Т. Л. Операционные системы, среды и оболочки: учебное пособие / Т.Л.Партыка, И.И. Попов. — 5-е изд., перераб. и доп. — Москва:ФОРУМ: ИНФРА-М, 2017. — 560 с. : ил. — (Профессиональное образование). [Электронный ресурс; Режим доступа http://znanium.com]

3. Гостев, И. М. Операционные системы: учебник и практикум для среднего профессионального образования / И. М. Гостев. — 2-е изд., испр. и доп. — Москва: Издательство Юрайт, 2019. — 164 с. — (Профессиональное образование). [Электронный ресурс; Режим доступа https://www.biblio-online.ru]

4. Нестеров, С. А. Базы данных: учебник и практикум для среднего профессионального образования / С. А. Нестеров. — Москва: Издательство Юрайт, 2019. — 230с. — (Профессиональное образование). [Электронный ресурс; Режим доступа https://www.biblio-online.ru]

5. Стружкин, Н. П. Базы данных: проектирование: учебник для среднего профессионального образования / Н. П. Стружкин, В. В. Годин. — Москва: Издательство Юрайт, 2019. — 477 с. — (Профессиональное образование). [Электронный ресурс; Режим доступа https://www.biblio-online.ru]

6. Стружкин, Н. П. Базы данных: проектирование. Практикум: учебное пособие для среднего профессионального образования / Н. П. Стружкин, В. В. Годин. — Москва: Издательство Юрайт, 2019. — 291 с. — (Профессиональное образование). [Электронный ресурс; Режим доступа https://www.biblio-online.ru]

7. Голицына, О. Л. Основы проектирования баз данных: учеб. пособие / О.Л. Голицына, Т.Л. Партыка, И.И. Попов. — 2-е изд., перераб. и доп. — Москва:ФОРУМ: ИНФРА-М, 2019. — 416 с.: ил. — (Среднее профессиональное образование). [Электронный ресурс; Режим доступа http://znanium.com]

8. Гордеев, С. И. Организация баз данных в 2 ч. Часть 1: учебник для среднего профессионального образования / С. И. Гордеев, В. Н. Волошина. — 2-е изд., испр. и доп. — Москва: Издательство Юрайт, 2019. — 310 с. — (Профессиональное образование). [Электронный ресурс; Режим доступа https://www.biblio-online.ru]

9. Гордеев, С. И. Организация баз данных в 2 ч. Часть 2: учебник для среднего профессионального образования / С. И. Гордеев, В. Н. Волошина. — 2-е изд., испр. и доп. — Москва: Издательство Юрайт, 2019. — 513 с. — (Профессиональное образование). [Электронный ресурс; Режим доступа https://www.biblio-online.ru]

10. Сети и телекоммуникации: учебник и практикум для среднего профессионального образования / К. Е. Самуйлов [и др.]; под редакцией К. Е. Самуйлова, И. А. Шалимова, Д. С. Кулябова. — Москва: Издательство Юрайт, 2019. — 363 с. — (Профессиональное образование). [Электронный ресурс; Режим доступа https://www.biblio-online.ru]

11. Дибров, М. В. Компьютерные сети и телекоммуникации. Маршрутизация в ip- сетях в 2 ч. Часть 1: учебник и практикум для среднего профессионального образования / М. В. Дибров. — Москва: Издательство Юрайт, 2019. — 333 с. — (Профессиональное образование). [Электронный ресурс; Режим доступа https://www.biblio-online.ru]

12. Дибров, М. В. Компьютерные сети и телекоммуникации. Маршрутизация в ip- сетях в 2 ч. Часть 2: учебник и практикум для среднего профессионального образования / М. В. Дибров. — Москва: Издательство Юрайт, 2019. — 351 с. —

(Профессиональное образование). [Электронный ресурс; Режим доступа https://www.biblio-online.ru.

1.2. Методические издания по всем входящим в реализуемые основные образовательные программы учебным предметам, курсам, дисциплинам (модулям) в соответствии с учебным планом:

1. Тен М.Б. МДК.01.01 Операционные системы Методические указания по выполнению практических занятий для обучающихся всех форм обучения образовательных учреждений среднего профессионального обучения специальности 10.02.05 «Обеспечение информационной безопасности автоматизированных систем» (10.00.00 ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ) – г. Нижневартовск: ННТ (филиал) ФГБОУ ВО «ЮГУ», 2020 [Электронный ресурс; Режим доступа: Полнотекстовая коллекция учебно-методических изданий ЮГУ]

1.3. Периодические издания:

Теоретический и научно-методический журнал «Среднее профессиональное образование» + ПриложениеЭлектронные источники:

1. Информационно-справочная система по документам в области техническойзащиты информации www.fstec.ru
2. Информационный портал по безопасности www.SecurityLab.ru.
3. Образовательные порталы по различным направлениям образования и тематике http://depobr.gov35.ru
4. Российский биометрический портал www.biometrics.ru
5. Сайт журнала Информационная безопасность http://www.itsec.ru
6. Сайт Научной электронной библиотеки www.elibrary.ru
7. Справочно-правовая система «Гарант» www.garant.ru
8. Справочно-правовая система «Консультант Плюс» www.consultant.ru
9. Федеральная служба по техническому и экспортному контролю (ФСТЭК России)www.fstec.ru
10. Федеральный портал «Информационно-коммуникационные технологии вобразовании» http://www.ict.edu.ru
11. Федеральный портал «Российское образование www.edu.ru