

Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Саратовский государственный технический университет  
имени Гагарина Ю.А.»

Филиал федерального государственного бюджетного образовательного  
учреждения высшего образования  
«Саратовский государственный технический университет  
имени Гагарина Ю.А.» в г. Петровске

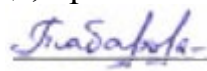
УТВЕРЖДАЮ  
Директор филиала СГТУ  
имени Гагарина Ю.А. в г. Петровске  
Е.А.Бесшапошникова  
«25» 2024 г.



## **МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ**

по дисциплине  
ОП.08 «Основы проектирования баз данных»  
направление подготовки  
09.02.07 «Информационные системы и программирование»

Методические указания рассмотрены  
на заседании предметной (цикловой) комиссии  
общепрофессиональных дисциплин,  
профессиональных модулей специальностей  
технического профиля  
«14» июня 2024 года, протокол №12

Председатель ПЦК  /Ю.А.Табарова/

Петровск 2024

### Пояснительная записка

Методические указания по выполнению практических работ подготовлены на основе рабочей программы учебной дисциплины ОП.08 «Основы проектирования баз данных», разработанной на основе ФГОС СПО по специальности 09.02.07 «Информационные системы и программирование» и соответствующих общих (ОК) и профессиональных (ПК) компетенций:

ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам.

ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности.

ОК 04. Эффективно взаимодействовать и работать в коллективе и команде

ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста.

ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках.

ПК 11.1. Осуществлять сбор, обработку и анализ информации для проектирования баз данных.

ПК 11.2. Проектировать базу данных на основе анализа предметной области.

ПК 11.3. Разрабатывать объекты базы данных в соответствии с результатами анализа предметной области.

ПК 11.4. Реализовывать базу данных в конкретной системе управления базами данных.

ПК 11.5. Администрировать базы данных.

ПК 11.6. Защищать информацию в базе данных с использованием технологии защиты информации

При выполнении практических работ студент должен **знать**:

- основы теории баз данных;
- модели данных;
- особенности реляционной модели и проектирование баз данных;
- изобразительные средства, используемые в ER- моделировании;
- основы реляционной алгебры;
- принципы проектирования баз данных;
- обеспечение непротиворечивости и целостности данных;
- средства проектирования структур баз данных;
- язык запросов SQL

При выполнении практических работ студент должен **уметь**:

- проектировать реляционную базу данных;
- использовать язык запросов для программного извлечения сведений из баз данных.

Содержание практических занятий определено рабочей программой и тематическим планированием, соответствует теоретическому материалу изучаемых разделов учебной дисциплины.

Объем практических занятий по дисциплине определяется учебным планом по данной специальности.

Продолжительность практического занятия – 2 академических часа. Перед проведением практического занятия преподавателем организуется инструктаж, а по его окончании – обсуждение итогов.

Комплект методических указаний по выполнению практических работ по дисциплине ОП.08 «Основы проектирования баз данных» содержит 11 практических занятий.

**Перечень практических работ  
по дисциплине ОП.08 «Основы проектирования баз данных»**

**ПРАКТИЧЕСКАЯ РАБОТА № 1**

Тема: Создание проекта БД. Создание БД. Редактирование и модификация таблиц. Создание ключевых полей. Задание индексов. Установление и удаление связей между таблицами.

**ПРАКТИЧЕСКАЯ РАБОТА № 2**

Тема: Редактирование, добавление и удаление записей в таблице. Применение логических условий к записям. Открытие, редактирование и пополнение табличного файла

**ПРАКТИЧЕСКАЯ РАБОТА № 3**

Тема: Создание формы. Управление внешним видом формы

**ПРАКТИЧЕСКАЯ РАБОТА № 4**

Тема: Создание меню различных видов. Модификация и управление меню.

**ПРАКТИЧЕСКАЯ РАБОТА № 5**

Тема: Создание интерфейса входной формы

**ПРАКТИЧЕСКАЯ РАБОТА № 6**

Тема: Создание файла проекта базы данных. Использование исполняемого файла проекта БД, приемы создания и управления.

**ПРАКТИЧЕСКАЯ РАБОТА № 7**

Тема: Выборка данных из БД.

**ПРАКТИЧЕСКАЯ РАБОТА № 8**

Тема: Модификация содержимого БД. Добавление, удаление и обновление данных.

**ПРАКТИЧЕСКАЯ РАБОТА № 9**

Тема: Символы подстановки и регулярные выражения (LIKE)

**ПРАКТИЧЕСКАЯ РАБОТА № 10**

Тема: Вычисляемые поля

**ПРАКТИЧЕСКАЯ РАБОТА № 11**

Тема: Проведение сортировки и фильтрации данных. Поиск данных по одному и нескольким полям. Поиск данных в таблице. Группировка данных (GROUP BY)

## **ИНСТРУКЦИИ ДЛЯ ОБУЧАЮЩИХСЯ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ**

Прежде чем приступить к выполнению заданий, внимательно прочитайте данные рекомендации. Практические работы включают в себя задания следующих видов.

### **1. Работа за компьютером**

В ходе выполнения практических работ студент должен:

- выполнять требования по охране труда
- соблюдать инструкцию по правилам и мерам безопасности в кабинете информационных технологий
- строго выполнять весь объем работы, указанный в задании
- соблюдать требования эксплуатации компьютерной техники (правила включения и выключения)
- предоставить отчет о проделанной работе по окончании выполненной работы, который должен содержать:

1. Название работы.
2. Цель работы.
3. Задание и его решение.
4. Вывод о проделанной работе.

Текст отчета по практической работе должен быть набран на компьютере шрифтом Times New Roman размером 14 пт. (при оформлении текста используется текстовый редактор MS Word). Шрифт, используемый в иллюстративном материале (таблицы и рисунки), рекомендуется уменьшить до 12 пт. Межстрочный интервал в основном тексте - полуторный. В иллюстративном материале межстрочный интервал рекомендуется сделать одинарным. Поля страницы должны быть: левое поле - 30 мм; правое поле – 15 мм; верхнее и нижнее поле - 20 мм.

Каждый абзац должен начинаться с красной строки. Отступ абзаца – 1,25 см от левой границы текста.

Студент должен выполнить практическую работу самостоятельно (или в группе, если это предусмотрено заданием). Практическая работа выполняется согласно заданию и методическим рекомендациям. После выполнения практической работы обучающийся самостоятельно себя контролирует путем ответов на вопросы. Результат работы представляется преподавателю в виде файла (файлов) в личном каталоге, защищается обучающимися.

По ходу выполнения работы при возникновении вопросов обучающийся может получить консультацию у преподавателя или самостоятельно воспользоваться лекционным материалом, рекомендуемой литературой.

### **2. Ответ на поставленные вопросы (с аргументацией)**

Прочитайте вопрос и вникните в него.

Для удобства подчеркните ту, фразу, которая, по вашему мнению, является главной. Это поможет вам быстрее сориентироваться при ответе на вопрос.

Если вы считаете, что можете ответить на вопрос без помощи лекции и дополнительной литературы – приступайте. Если же вопрос заставляет вас

сомневаться, откройте лекционную тетрадь (учебник или дополнительную литературу), прочитайте необходимый пункт, вникните в содержание и после этого приступайте за работу.

**ГЛАВНОЕ!** Не переписывайте отрывки лекции в рабочую тетрадь! Четко отвечайте на ПОСТАВЛЕННЫЙ вопрос!

Не забудьте привести аргументацию (обоснование) вашей позиции, если вопрос предполагает личностное отношение к проблеме.

### **3. Заполнение таблиц и схем**

Прочитайте название таблицы или схемы.

Исходя из названия, вы поймете цель предстоящей работы.

Воспользуйтесь материалами лекций или другими источниками, чтобы заполнить таблицу (схему).

Используйте цветные графические материалы для выделения строк, столбцов или элементов схем.

Особое внимание обращайтесь на четкость при отборе материала: делайте записи кратко и четко!

**4. Поиск информации в сети** — использование web-браузеров, баз данных, пользование информационно-поисковыми и информационно-справочными системами, автоматизированными библиотечными системами, электронными журналами. Поиск и обработка информации включает подготовку фрагмента практического занятия.

## ПРАКТИЧЕСКАЯ РАБОТА № 1

**Тема: Создание проекта БД. Создание БД. Редактирование и модификация таблиц. Создание ключевых полей. Задание индексов. Установление и удаление связей между таблицами**

**Цель работы:** Получение навыков работы по созданию структуры таблиц, модификации структуры таблиц, заполнению таблиц. Создание ключевых полей, индексированных полей, установка связей между таблицами. Удаление информации из связанных таблиц и восстановление этой информации.

**Оборудование:** ПК, интернет, программное обеспечение – MS Access, инструкции по выполнению работы.

**Справочный материал:**

*Разработка структуры БД*

Выполнение начинается с разработки структуры БД. На этом этапе должны быть детально проанализированы условия задания и, на их основе, определено количество таблиц, необходимых для описания всех характеристик анализируемой предметной области. Кроме того, необходимо определить какие поля в таблицах будут использованы в качестве ключевых, а также определить каким образом будет осуществляться связь между таблицами. Если невозможно установить связи посредством использования ключевых полей, определить таблицы, которые будут использоваться только для связи между другими таблицами.

*Создание таблиц.*

Для каждого поля конкретной таблицы необходимо определить его тип и размер и тщательно проверить, удовлетворяет ли диапазон значений выбранного типа тем значениям, которые может реально принимать данное поле. При необходимости, для некоторых полей можно установить *Условие на значение* и задать сообщение, выдаваемое на экран в случае несоответствия введенного значения заданному условию или присвоить значения, принимаемые по умолчанию. Можно также определить формат вводимой информации для конкретных полей. Заполнить соответствующей информацией каждый из разделов создаваемой структуры таблицы: *Имя поля, Тип данных и Описание*. Раздел описаний необязателен для заполнения, но информация, введенная в данный раздел отображается в строке состояния при вводе данных для конкретного поля, облегчая процесс ввода.

*Создание индексов и ключевых полей.*

Информацию в таблицах можно упорядочить, создав индекс для конкретного поля или нескольких полей. Желательно, чтобы для таблиц были созданы ключевые поля. Для установления связей между таблицами наличие таких полей обязательно. Ключевое поле может быть простым или составным, т.е. состоять из нескольких полей для однозначной идентификации каждой записи в таблице.

*Сохранение таблиц*

По окончании создания структуры таблицы в режиме *Конструктора* ее необходимо сохранить. Для сохранения выполнить: *Файл/Сохранить*.

### *Заполнение таблиц.*

Открыть таблицу в режиме *Таблицы*. Заполнить необходимой информацией, подготовив для заполнения не менее десяти записей для основной таблицы. Сохранение не требуется, т.к. сохранение производится сразу при переходе к следующей записи. Заккрыть заполненную таблицу. Аналогично поступить с остальными таблицами.

### *Установка связей между таблицами.*

Выполнить команду *Работа с базами данных /Схема данных*.

1. Появится окно *Схема данных*. Если связи устанавливаются впервые, оно будет содержать диалоговое окно *Добавление таблицы*. Если окно *Добавление таблицы* отсутствует, его можно открыть, пиктограмму *Добавить таблицу*, либо одноименный пункт в контекстном меню (правая клавиша мыши по рабочей области).
2. Выбрать таблицу, которая будет использоваться для установки связей, затем выполнить щелчок на кнопке *Добавить*, для добавления таблицы в окно *Схема данных*.
3. Повторить действия, описанные в п.2 для каждой таблицы, участвующей в установке связи.
4. Для создания связей между таблицами переместить поле (или поля), которое необходимо связать на соответствующее поле другой таблицы. В большинстве связей ключевое поле первой таблицы связывается с аналогичным полем второй таблицы. После перемещения поля появится диалоговое окно *Связи*.
5. В диалоговом окне представлены названия таблиц, между которыми устанавливаются связи и имена полей для связи. Полям, на основе которых создаются связи между таблицами, не обязательно иметь одинаковые имена, однако они должны быть одного типа. Исключение составляют поля счетчиков, которые можно связывать с числовыми полями.
6. Для автоматической поддержки целостности БД установить флажок *Обеспечение целостности данных*. Кроме этого флажка в окне представлены и другие:
  - Каскадное обновление связанных полей. При включении данного режима изменения, сделанные в связанном поле первой таблицы, автоматически вносятся в поля связанной таблицы, содержащей те же данные.
  - Каскадное удаление связанных полей. При включении данного режима удаление записей в первой таблице приводит к удалению соответствующих записей связанной таблицы.
7. Выполнить щелчок на кнопке *Создать*. Затем закрыть окно *Связи*. При запросе о сохранении связи выполнить щелчок на кнопке *Да*.

### *Завершение работы с БД.*

Для завершения работы с БД необходимо закрыть окно БД, затем закрыть окно приложения.

### **Содержание работы:**

**Задание 1.** Создать структуры таблиц, ключевые и индексные поля. Заполнить таблицы данными, установить связи, удалить данные, восстановить их.



**Постановка задачи:** Создать базу данных ОТДЕЛ КАДРОВ, поместив в нее три таблицы:

СОТРУДНИК, СОСТАВ СЕМЬИ и ШТАТНОЕ РАСПИСАНИЕ, содержащие информацию о сотрудниках предприятия.

**Описание прикладной области Отдел кадров предприятия.**

Анализ предметной области показывает, что для автоматизации работы отдела кадров целесообразно создать базу данных ОТДЕЛ КАДРОВ, состоящую из трех таблиц: СОТРУДНИК, СОСТАВ СЕМЬИ, ШТАТНОЕ РАСПИСАНИЕ. Таблицы будут связаны между собой следующим образом: Таблица СОТРУДНИК с таблицей СОСТАВ СЕМЬИ связываются по полю Идент код, а с таблицей ШТАТНОЕ РАСПИСАНИЕ - по полю Должн.

Наименование таблицы	Структура таблицы
Сотрудник	Идентификационный код, Фамилия, Имя, Отчество, Пол, Дата рождения, Место рождения, Образование, Должность, Стаж работы, Семейное положение, Дата зачисления на работу, Телефон, Домашний адрес
Состав семьи	Идентификационный код, Взаимоотношения, Фамилия, Имя, Отчество, Год рождения
Штатное расписание	№ по порядку, Название подразделения, Должность, Количество штатных единиц, Должностной оклад, Фонд з/платы за месяц, Фонд з/платы за год.

**Состав и характеристика полей таблицы “Штатное расписание”.**

Название поля	Имя поля	Характеристики поля	
		Тип данных	Возможности
№ по порядку	НПП	Числовой	Длинное целое, обязательное
Название подразделения	Назв подраз	Текстовый	30 символов, обязательное
Должность	Должность	Текстовый	15 символов, обязательное
Количество штатных единиц	Кол ед	Числовой	Длинное целое, обязательное
Должностной оклад	Оклад	Числовой	Длинное целое, обязательное
Фонд з/платы за месяц	ФЗПМ	Числовой	Длинное целое, обязательное
Фонд з/платы за год	ФЗПГ	Числовой	Длинное целое, обязательное

**Состав и характеристика полей таблицы “Сотрудник”.**

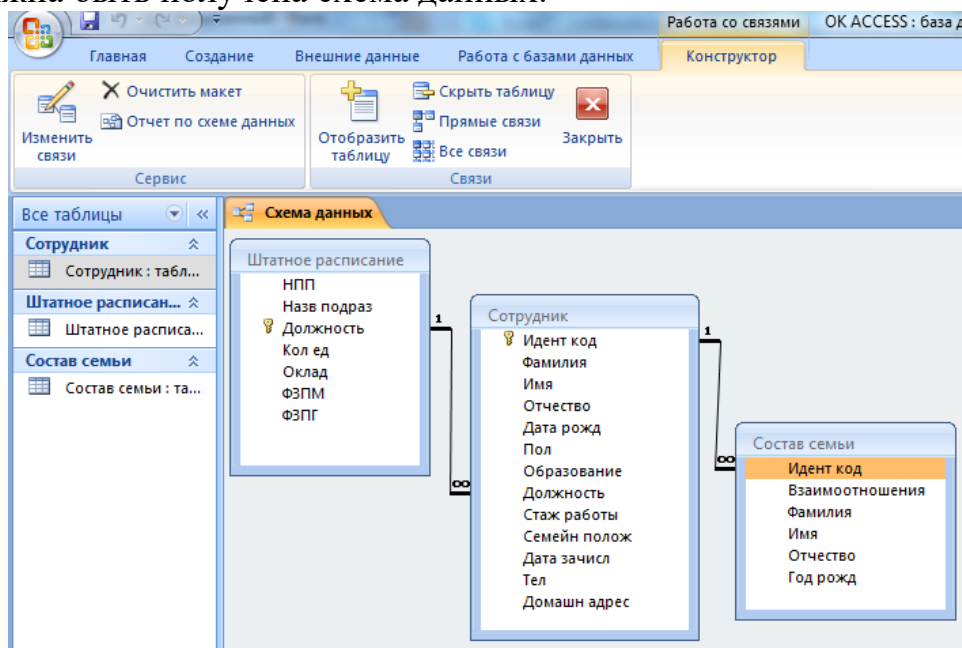
Название поля	Имя поля	Характеристики поля	
		Тип данных	Возможности

Идентификационный код	<u>Идент код</u>	Текстовый	10 символов, обязательное
Фамилия	Фамилия	Текстовый	20 символов, обязательное
Имя	Имя	Текстовый	15 символов, обязательное
Отчество	Отчество	Текстовый	15 символов, обязательное
Пол	Пол	Текстовый	50 символов, не обязательное
Дата рождения	Дата рожд	Дата/время	Маска ввода 00.00.0000, не обязательное
Место рождения	Место рожд	Текстовый	15 символов, не обязательное
Образование	Образование	Текстовый	15 символов, обязательное
Должность	Должность	Мастер подстановок	15 символов, индексированное, допускается совпадение, обязательное
Стаж работы	Стаж работы	Числовой	Длинное целое, обязательное
Семейное положение	Семейн полож	Текстовый	11 символов, не обязательное
Дата зачисления на работу	Дата зачисл	Дата/время	Маска ввода 00.00.0000, не обязательное
Телефон	Тел	Текстовый	8 символов, не обязательное
Домашний адрес	Домашн адрес	Поле МЕМО	не обязательное

**Состав и характеристика полей таблицы “Состав семьи”.**

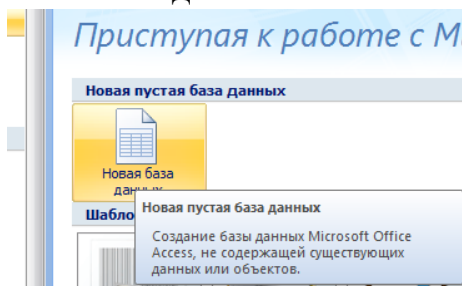
Название поля	Имя поля	Характеристики поля	
		Тип данных	Возможности
Идентификационный код	Идент код	Мастер подстановок	10 символов, обязательное
Взаимоотношения	Взаимоотношения	Текстовый	10 символов, не обязательное
Фамилия	Фамилия	Текстовый	20 символов, обязательное
Имя	Имя	Текстовый	15 символов, обязательное
Отчество	Отчество	Текстовый	15 символов, обязательное
Год рождения	Год рожд	Дата/время	Маска ввода 00.00.0000, обязательное

Должна быть получена схема данных:

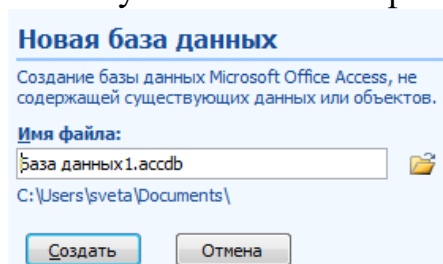



**Порядок выполнения:**

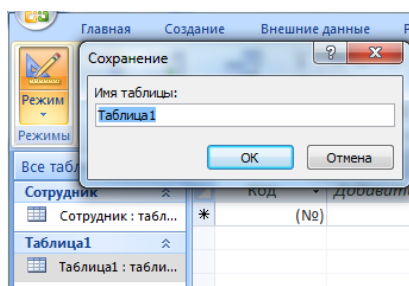
1. Запустить Microsoft Access
2. щелкаем на пиктограмме Новая база данных



3. В правой части окна появится информация об имени файла и указана директория для его хранения. По умолчанию имя файла - **База данных1.accdb**.

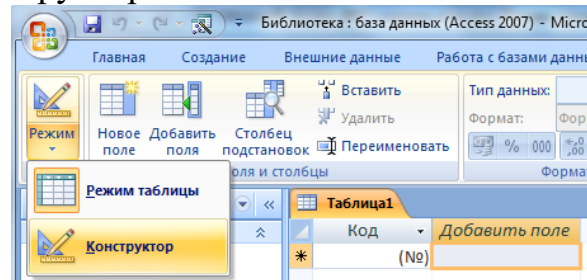


4. Далее щелкнуть справа по пиктограмме  и ввести имя файла *Библиотека*
5. В результате получаем:

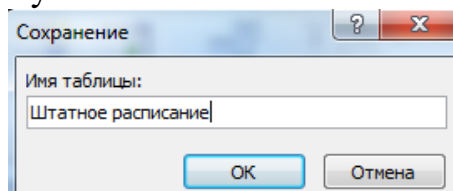


6. Нажимаем кнопку *Создать*

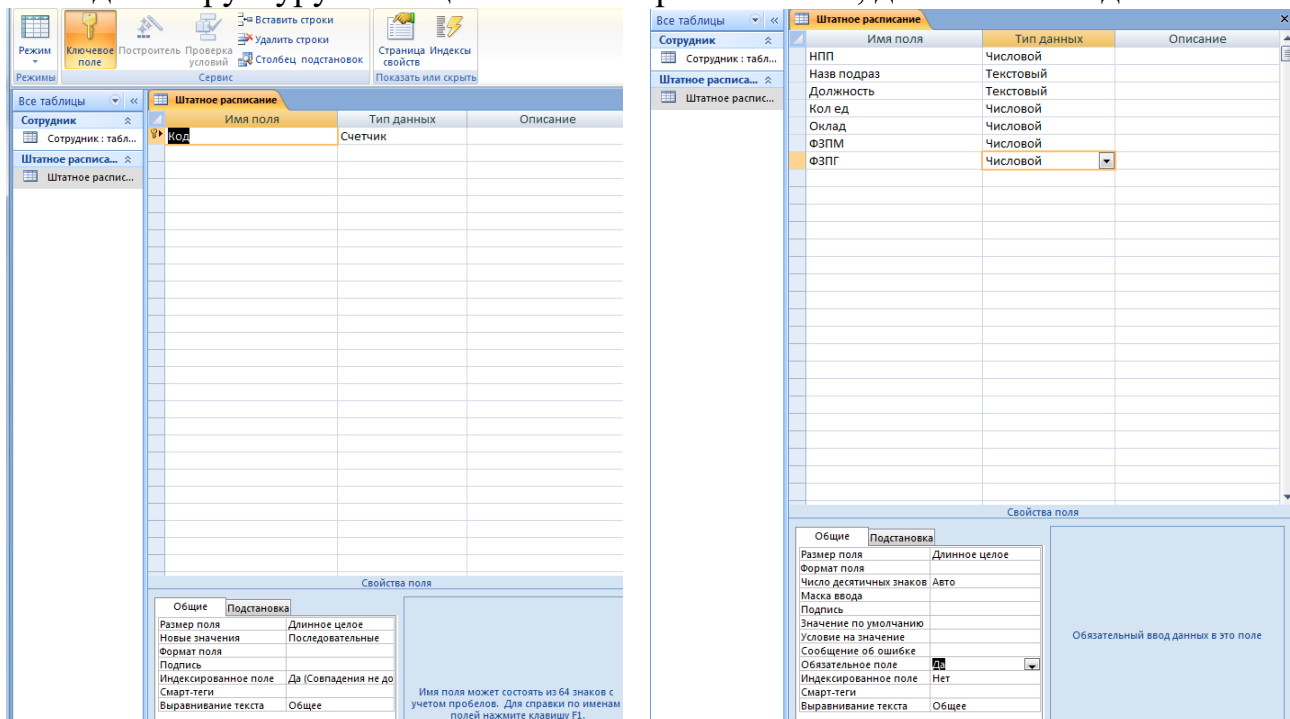
7. Далее необходимо перейти в режим Конструктор и создать структуру первой таблицы базы данных. Для этого необходимо щелкнуть на пиктограмме Режим и выбрать режим Конструктор.



8. Откроется окно Сохранение, в котором надо указать имя Штатное расписание и нажать кнопку ОК.

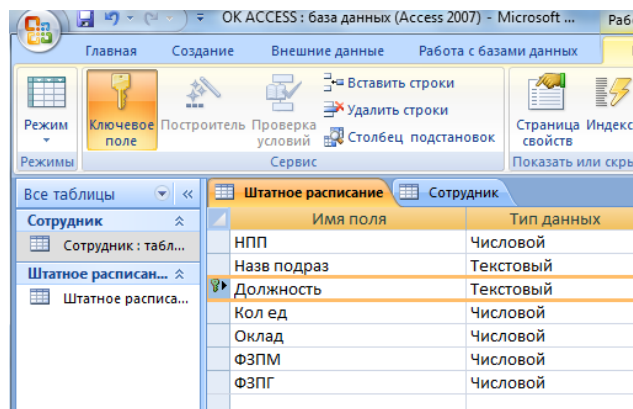
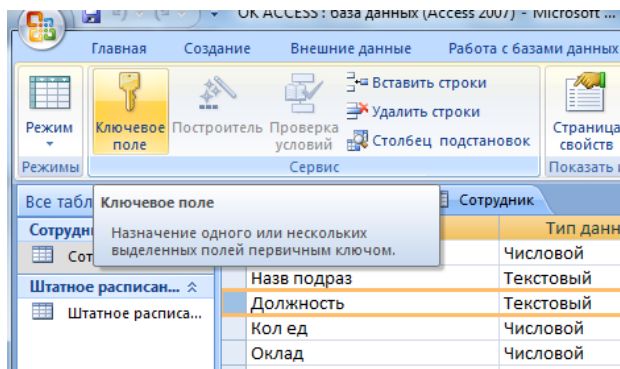


9. Создать структуру таблицы «Штатное расписание», данные не вводить



После создания структуры таблицы необходимо задать ключевое поле.

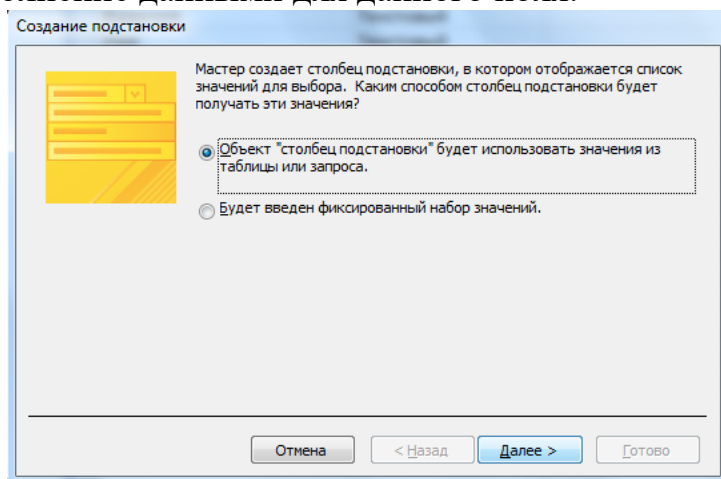
Как ключевое поле выбираем поле **Должность**, т.к. оно не содержит записей, что повторяются, а также будет использовано для связи с таблицей «Сотрудник».



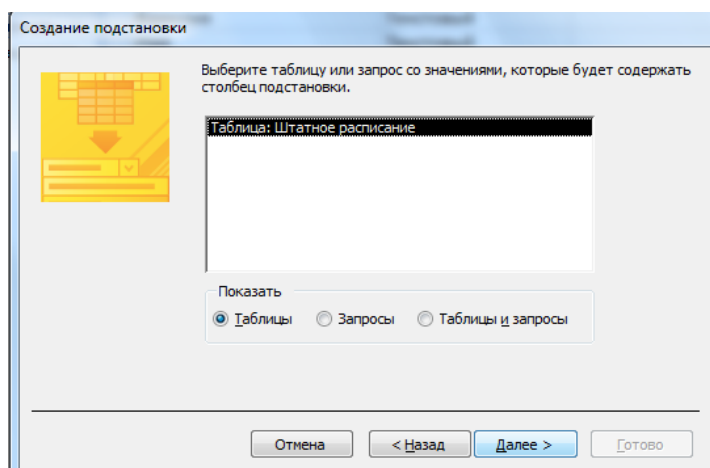
10. Создать структуру таблицы «Сотрудник», данные не вводить:

- меню Создание
- Таблица
- Конструктор
- имя Сотрудник и т.д.

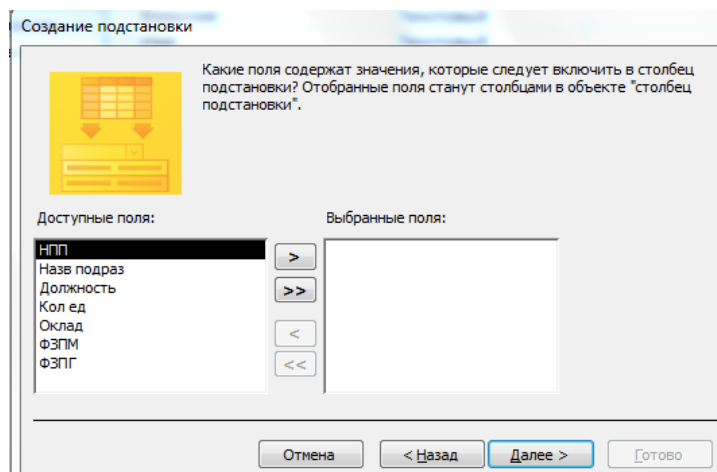
Для поля **Должность** выбираем тип **Мастер подстановок**. Это позволит облегчить заполнение данными для данного поля.



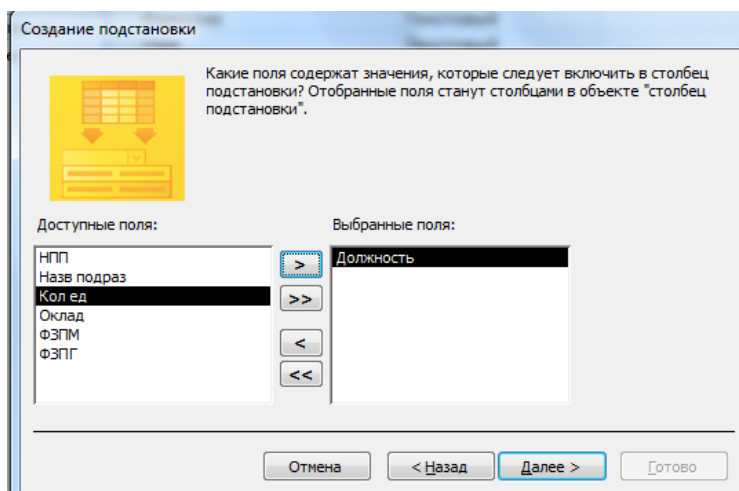
Далее



Далее

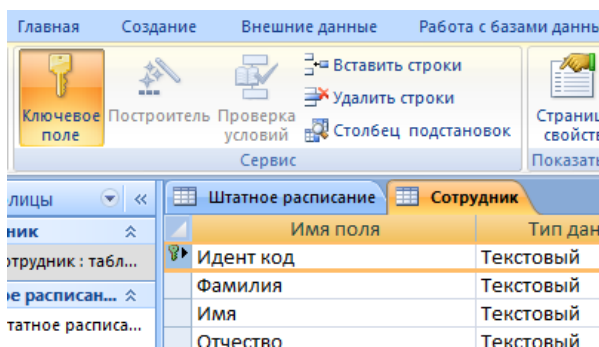


Далее



Готово

Как ключевое поле выбираем поле **Идент код**, т.к. оно не содержит записей, что повторяются, а также будет использовано для связи с таблицей «Состав семьи».



11. Создать структуру таблицы «Состав семьи», данные не вводить:

При создании поля **Идент код** как столбец подстановки используется поле **Идент код** из таблицы «Сотрудник».

Создание подстановки

Выберите таблицу или запрос со значениями, которые будут содержать столбец подстановки.

Таблица: Сотрудник  
Таблица: Штатное расписание

Показать  
☒ Таблицы  
☐ Запросы  
☐ Таблицы и запросы

Отмена < Назад Далее > Готово

Создание подстановки

Какие поля содержат значения, которые следует включить в столбец подстановки? Отобранные поля станут столбцами в объекте "столбец подстановки".

Доступные поля: Выбранные поля:

Фамилия  
Имя  
Отчество  
Дата рожд  
Пол  
Образование  
Должность  
Стаж работы

Идент код

Отмена < Назад Далее > Готово

Поле **Иден код** выбрать как индексное поле. Для этого в разделе **Свойства поля** выбрать строку **Индексированное поле** и выбрать из выпадающего списка **Да (допускаются совпадения)**.

Общие	Подстановка
Размер поля	10
Формат поля	
Маска ввода	
Подпись	
Значение по умолчанию	
Условие на значение	
Сообщение об ошибке	
Обязательное поле	Нет
Пустые строки	Да
Индексированное поле	Нет
Сжатие Юникод	Нет
Режим ИМЕ	Да (Допускаются совпадения)
Режим предложений ИМЕ	Да (Совпадения не допускаются)
Смарт-теги	

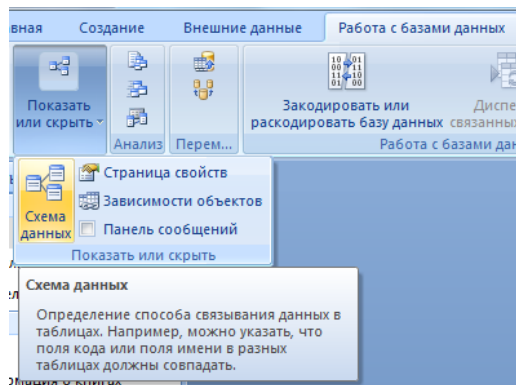
Таблицы будут связаны между собой таким образом: таблица **Сотрудник** с таблицей **Состав семьи** связываются по полю **Идент код**, с таблицей **Штатное расписание** – по полю **Должность**.

12.Закрывать все созданные структуры таблиц

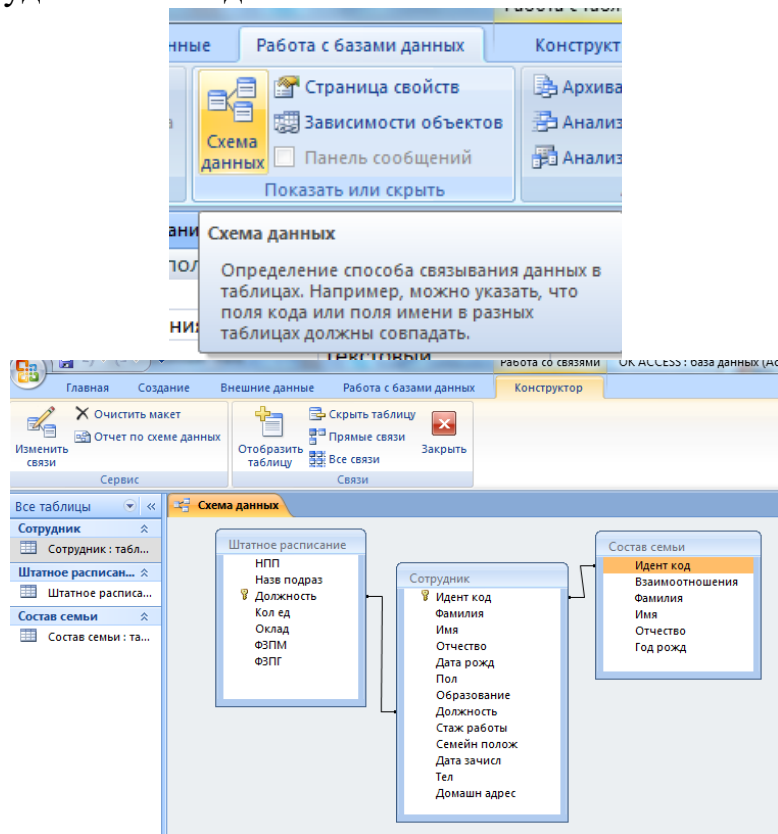
**Задание 2.** Создать связи между таблицами.

1.Создание связей между таблицами:

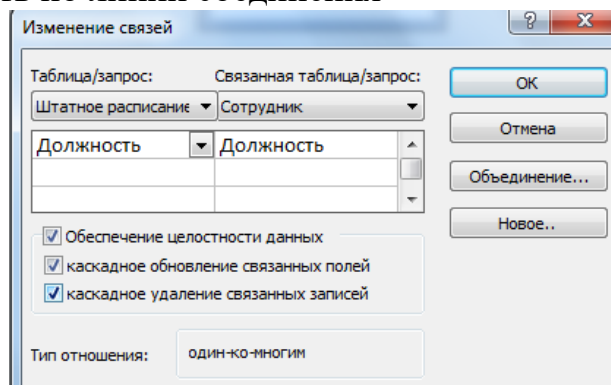
- меню Работа с базами данных
- Показать или скрыть



- Схема данных, появится окно Добавление таблицы
- Выделить все таблицы, Добавить каждую
- кнопка Заккрыть
- Схема данных будет иметь вид:



Дважды щелкнуть по линии соединения





Изменение связей

Таблица/запрос: Связанная таблица/запрос:

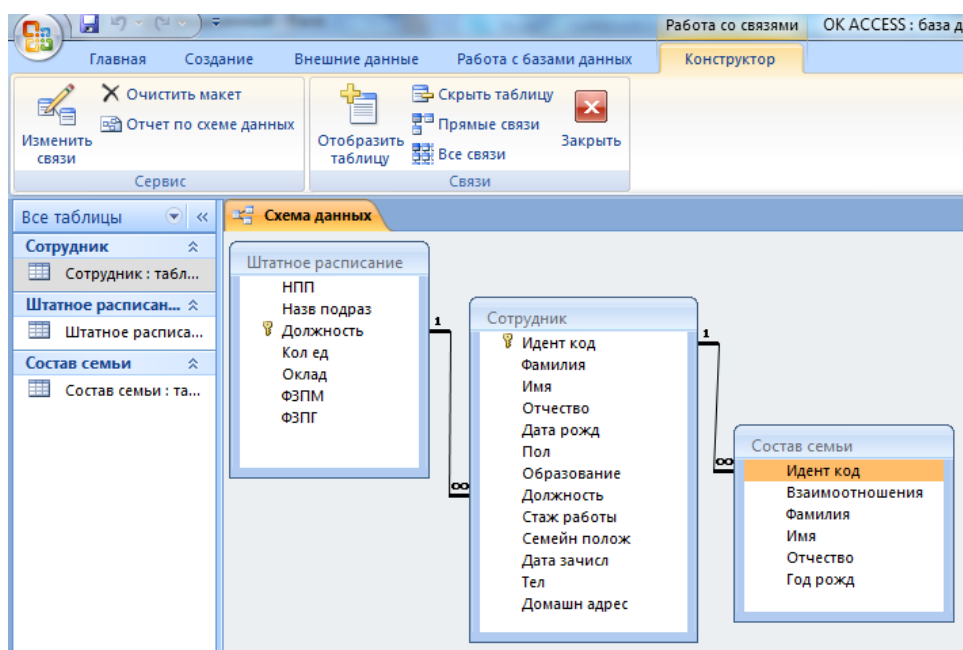
Сотрудник Состав семьи

Идент код Идент код

☒ Обеспечение целостности данных  
☒ каскадное обновление связанных полей  
☒ каскадное удаление связанных записей

Тип отношения: один-ко-многим

OK  
Отмена  
Объединение...  
Новое..



**Задание 3.** Внести данные во все таблицы

Таблица-объект ШТАТНОЕ РАСПИСАНИЕ

Нпп	Назв подр	Должн	Кол ед	Оклад	ФЗПМ	ФЗПГ
1	Дирекция	бухгалтер	2	230	460	5520
2	Дирекция	гл.бухгалтер	1	430	430	5160
3	Дирекция	директор	1	530	530	6360
4	Уч.кафедра	диспетчер	1	100	100	1200
5	Уч.кафедра	доцент	1	500	500	6000
6	Уч.кафедра	зав.кафедрой	1	430	430	5160
7	Дирекция	зам.директора	1	500	500	6000
8	Уч.кафедра	методист	2	200	400	4800
9	Дирекция	начальник ОК	1	150	150	1800
10	Уч.кафедра	преподаватель	4	350	1800	21600
11	Уч.кафедра	статистик	1	100	100	1200
12	Уч.кафедра	специалист	2	150	300	3600

**Таблица-объект СОТРУДНИК**

Идент код	Фамилия	Имя	Отчество	Пол	Дата рожд	Место рожд	Образов	Должн	Сем полож	Дата зач	Телефон
1314152347	Старченко	Светлана	Борисовна	ж	22.04.43	г.Казань	высшее	Бухгалтер	замужем	24.09.90	65-12-13
1545678990	Архипов	Сергей	Иванович	м	23.03.49	г.Харьков	высшее	Гл. бухгалтер	женат	10.12.88	нет
1624790203	Круговой	Геннадий	Иванович	м	22.04.45	г.Омск	высшее	Директор	вдовец	10.12.88	68-14-13
2748576413	Царева	Анна	Николаевна	ж	30.07.50	г.Харьков	ср.техн.ич.	Диспетчер	замужем	01.01.96	47-23-15
2934789231	Каменев	Татьяна	Дмитриевна	ж	24.06.59	г.Курск	высшее	Доцент	замужем	30.12.90	65-67-72
2955443781	Безродный	Владимир	Михайлович	м	05.09.53	г.Харьков	высшее	Зав. кафедрой	женат	01.09.92	32-32-14
1014654788	Садчиков	Аркадий	Викторович	м	10.01.57	г.Тамбов	высшее	Зам. директора	холост	10.12.88	10-12-10
2055894321	Бронзов	Станислав	Иванович	м	12.11.60	г.Москва	ср.техн.ич.	Методист	женат	31.08.94	23-10-70
1178943214	Мапошенко	Юрий	Николаевич	м	21.11.64	г.Омск	высшее	Начальник ОК	женат	31.08.94	43-35-13
2200987654	Коваль	Александр	Николаевич	ж	31.03.65	г.Киев	высшее	Преподаватель	замужем	01.10.92	47-67-33
2233668943	Строков	Олег	Викторович	м	05.08.65	г.Орел	ср.техн.ич.	Статистик	женат	10.09.92	69-05-03
2314743296	Бородулин	Андрей	Васильевич	м	31.12.69	г.Киев	высшее	Специалист	холост	31.08.95	27-14-12

**МЕМО-поле Таблицы СОТРУДНИК**

Адрес
ул.Гв.Широнинцев 21,кв.30
пер.Хрустальный 8
ул.Светлая 14,кв.55
ул.Артема 24, кв.1
ул.Героев труда 28-Б,кв.76
пр.Правды 44, кв.55
пер.Короленко 2, кв.1
ул. Революции 6, кв.2
ул.Пушкинская 54,кв2
ул.Иванова 5, кв.2
пр. Косиора 162, кв161
пр.Гагарина 117, кв.20

**Таблица-объект СОСТАВ СЕМЬИ**

<b>Идент код</b>	<b>Отношение</b>	<b>Фамилия</b>	<b>Имя</b>	<b>Отчество</b>	<b>Дата рожд</b>
1314152347	отец	Старченко	Николай	Иванович	12/01/1917
1314152347	мать	Старченко	Людмила	Яковлевна	25/12/1920
1545678990	сын	Архипов	Дмитрий	Сергеевич	01/09/1988
2748576413	муж	Царев	Петр	Алексеевич	14/11/1948
2934789231	муж	Каменев	Александр	Иванович	15/02/1952
2955443781	дочь	Безродная	Алла	Владимировна	24/06/1991
1014654788	мать	Садчикова	Мария	Ивановна	29/04/1930
2055894321	дочь	Бронзова	Инна	Станиславовна	15/12/1998
1178943214	сын	Мапошенко	Игорь	Юрьевич	22/06/1992
1178943214	сын	Мапошенко	Владимир	Юрьевич	23/08/1995
2233668943	дочь	Строкова	Юлия	Олеговна	28/07/1985
2233668943	дочь	Строкова	Наталия	Олеговна	14/03/1990

**Задание 4.** Создать структуры таблиц, ключевые и индексные поля. Заполнить таблицы данными, установить связи, удалить данные, восстановить их.

**Варианты заданий:**

- 1.Склад компьютерной техники
- 2.Учебное заведения
- 3.Дизайнерское агентство

## ПРАКТИЧЕСКАЯ РАБОТА № 2

**Тема: Редактирование, добавление и удаление записей в таблице.**

**Применение логических условий к записям. Открытие, редактирование и пополнение табличного файла**

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

**Оборудование:** ПК, интернет, программное обеспечение – MS Access, инструкции по выполнению работы.

### Справочный материал:

Запрос – это средство, с помощью которого извлекается из базы данных информация, отвечающая определенным критериям. Результаты запроса представляют не все записи из таблицы, а только те, которые удовлетворяют запросу.

Запросы состоят из ряда условий, каждое условие состоит из трех элементов:

1. поле, которое используется для сравнения;
  2. оператор, описывающий тип сравнения;
  3. величина, с которой должно сравниваться значение поля
- Запросы могут быть простые, сложные перекрестные.

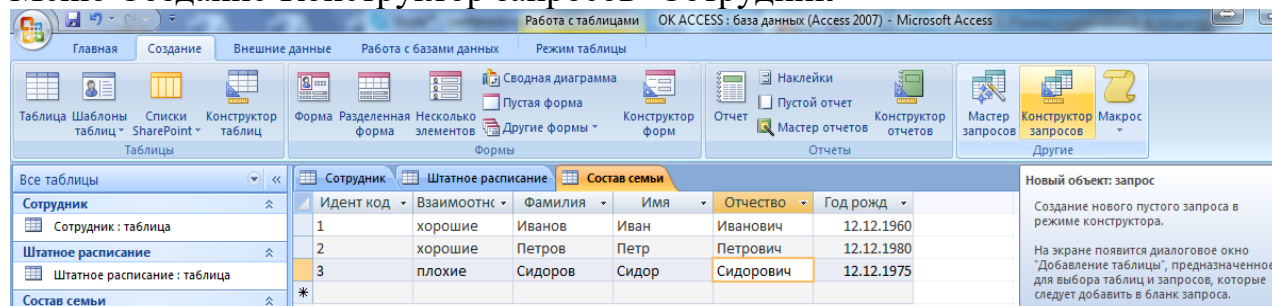
### Содержание работы:

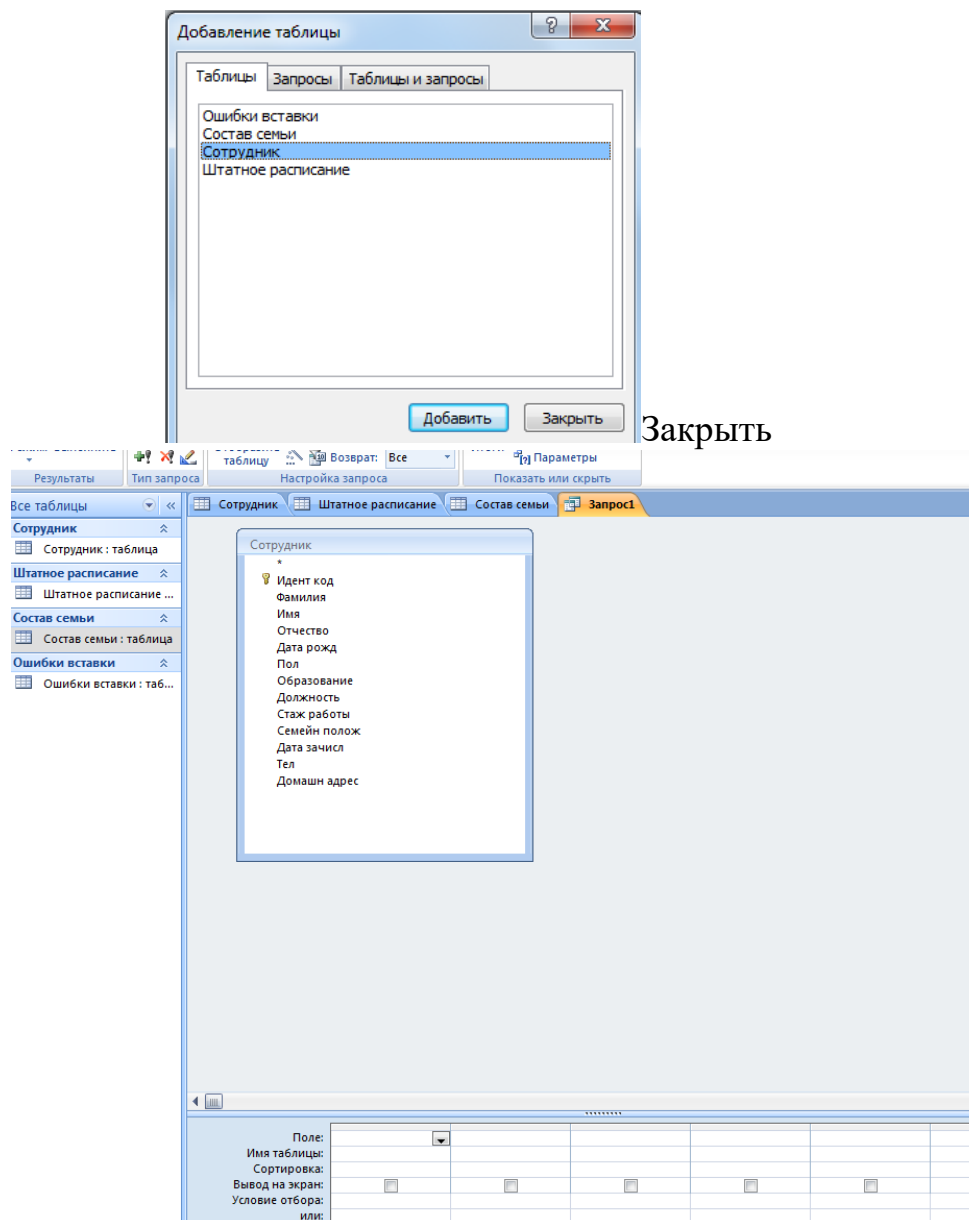
**Задание 1.** Создать запрос на выборку данных из одной таблицы

1.Создадим запрос, что содержит поля: Идент код, Фамилия, Имя, Отчество, Дата нар, который отображает список только тех сотрудников, фамилии которых начинаются с буквы "К". Список отсортируем по дате рождения по возрастанию.

Для этого необходимо выполнить такую последовательность действий.

Меню Создание-Конструктор запросов -Сотрудник





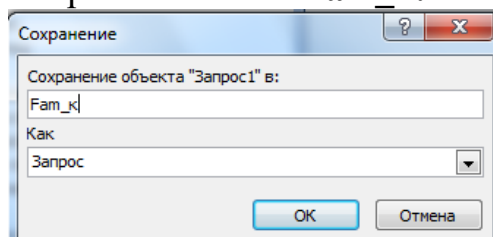
2.Выбираем объект **Запросы**, щелкаем пункт меню **Создать**. Открывается окно **Новый запрос**, в котором выбираем режим создания запроса **Конструктор**. Открывается окно **Запросы: запрос на выборку** и активизируется окно **Добавление таблицы**, в котором следует выбрать из списка таблицу **Сотрудник** (щелкнув мышью на имя таблицы), после чего нажать на кнопку **Добавить** и закрыть окно **Добавление таблицы**.

3.Дальше необходимо выбрать нужные поля и задать способы сортировки и условия отбора из таблицы. Для этого:

- выделить поля **Идент код**, **Фамилия**, **Имя**, **Отчество**, **Дата рожд** с помощью мыши в комбинации с клавишами **SHIFT** или **CTRL** и отбуксировать на бланк построения запроса **QBE** в строку **Поле**. Поля можно перемещать в бланк **QBE** и в одиночку.



Закрывать окно конструктора запроса, сохранить в памяти с именем. В окне базы данных появится файл запроса с именем **Fam\_k**.



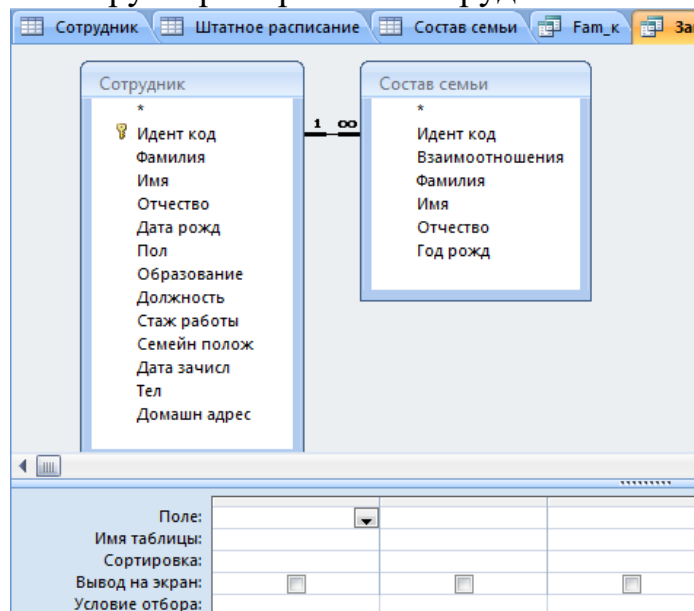
### Файл - Сохранить как

Выполнить запрос на выборку. Для этого выделить запрос **Fam\_k** и щелкнуть по кнопке **Открыть**. На экран выводится таблица, в которой отображаются все записи с фамилиями, которые начинаются на букву К, записи отсортированные по дате рождения по возрастанию.

Идент код	Фамилия	Имя	Отчество	Дата рожд
4	К	К	К	01.01.2000
*				

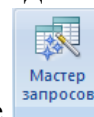
**Задание 2.** Создать запрос на выборку данных с из двух или трех таблиц.

1. Меню Создание-Конструктор запросов - Сотрудник и Состав семьи, Закрывать



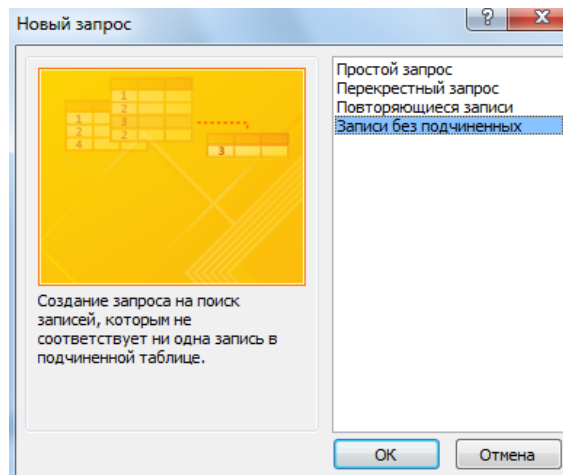
2. Создадим запрос, в результате выполнения которого будут полученные сведения о сотрудниках, которые не имеют родственников.

3. Для отчета включим поля, которые содержат идентификационный код, фамилию, имя, отчество рабочего, а также его дату рождения.

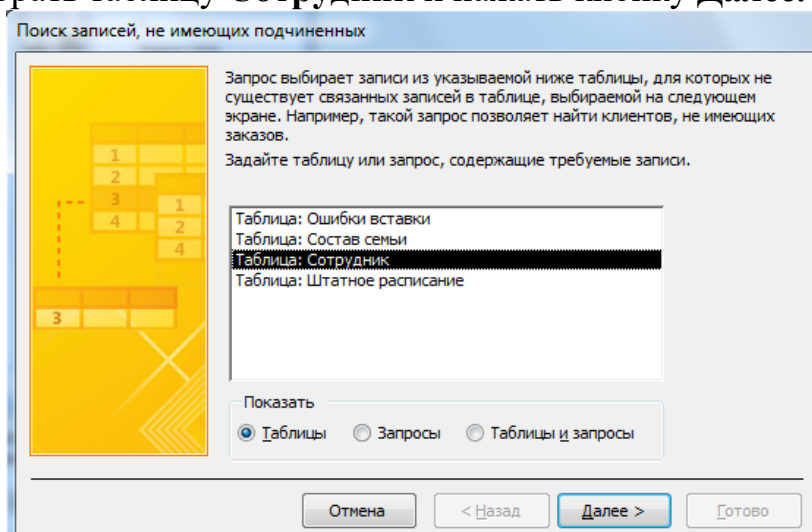


4. При выбранной вкладке **Запрос** щелкнуть по кнопке

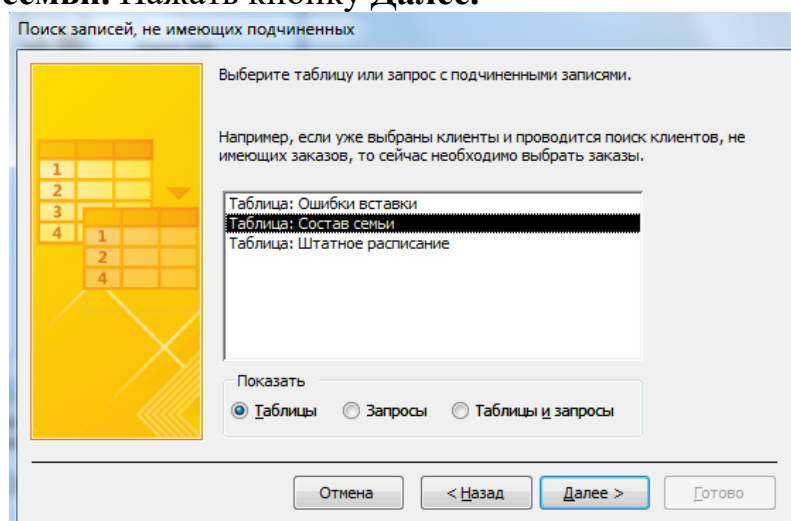
5. Открывается окно **Новый запрос**, в котором выбрать режим создания запроса **Записи без подчиненных**.



6. В первом окне с названием "Поиск записей, не имеющих подчиненных" мастер выведет на экран список для выбора основной таблицы, в котором выбрать таблицу **Сотрудник** и нажать кнопку **Далее**.



7. В следующем окне выбрать таблицу, что содержит подчиненные записи. Это таблица **Состав семьи**. Нажать кнопку **Далее**.



8. В следующем окне мастера проверить, что таблицы **Сотрудник** и **Состав семьи** связаны по полю **Идент код** (поля, по которым связанные таблицы, выделены). Если это не так, в каждом списке полей обеих таблиц выделить



поле **Идент код** и щелкнуть на кнопку «<=>», что расположена между списками. Нажать кнопку **Далее**.

Поиск записей, не имеющих подчиненных

Какие данные содержатся в обеих таблицах?  
Например, и таблица "Клиенты", и таблица "Заказы" содержат поле "Клиент". Соответствующие поля могут иметь и различные имена.  
Выберите подходящее поле в каждой таблице и нажмите кнопку <=>.

Поля в 'Сотрудник':

- Идент код
- Фамилия
- Имя
- Отчество
- Дата рожд
- Пол
- Образование
- Должность

Поля в 'Состав семьи':

- Идент код
- Взаимоотношения
- Фамилия
- Имя
- Отчество
- Год рожд

Соответствие: Идент код <=> Идент код

Отмена < Назад Далее > Готово

9. На экране появится новое окно, в котором отображенные поля, которые могут быть включены к отчету. В этом окне в левом поле в списке выделить по очереди поля **Идент код**, **Фамилия**, **Имя**, **Отчество**, **Дата рожд**, которые должны отображаться в отчете, и перенести их в левое поле с помощью кнопки «>>».

Поиск записей, не имеющих подчиненных

Выберите поля для отображения в результате выполнения запроса:

Доступные поля:

- Пол
- Образование
- Должность
- Стаж работы
- Семейн полож
- Дата зачисл
- Тел
- Домашн адрес

Выбранные поля:

- Идент код
- Фамилия
- Имя
- Отчество
- Дата рожд

Отмена < Назад Далее > Готово

Далее

Поиск записей, не имеющих подчиненных

Задайте имя запроса:

"Сотрудник" без подчиненных в "Состав семьи"

Указаны все сведения, необходимые для создания запроса с помощью мастера.

Дальнейшие действия после создания запроса:

- ☒ Просмотреть результаты запроса.
- ☐ Изменить структуру запроса.

Отмена < Назад Далее > Готово

Дальше нажать кнопку **Готово**.

10.Проверяем исправленный отчет и храним его под именем **Без родственников**.

Идент код	Фамилия	Имя	Отчество	Дата рожд
1	К	К	К	01.01.2000
*				

### Задание 3. Создать параметрический запрос.

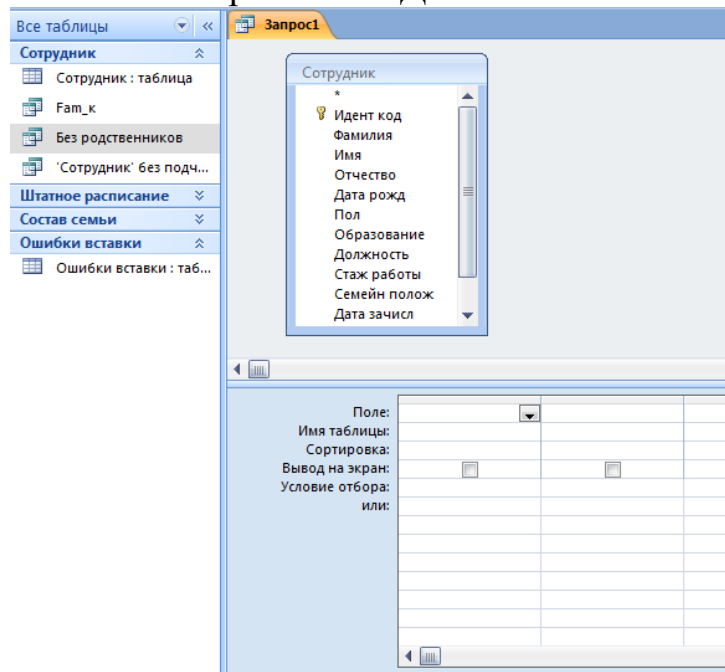
При выполнении параметрического запроса выводится диалоговое окно с приглашением ввести параметр для условия отбора записей. Параметров может быть несколько.

1.Создадим запрос, в результате выполнения которого будут выводиться поля **Фамилия, Имя, Отчество, Идент код** и **Стаж работы** сотрудника, фамилия которого будет указана в запросе как параметр отбора.

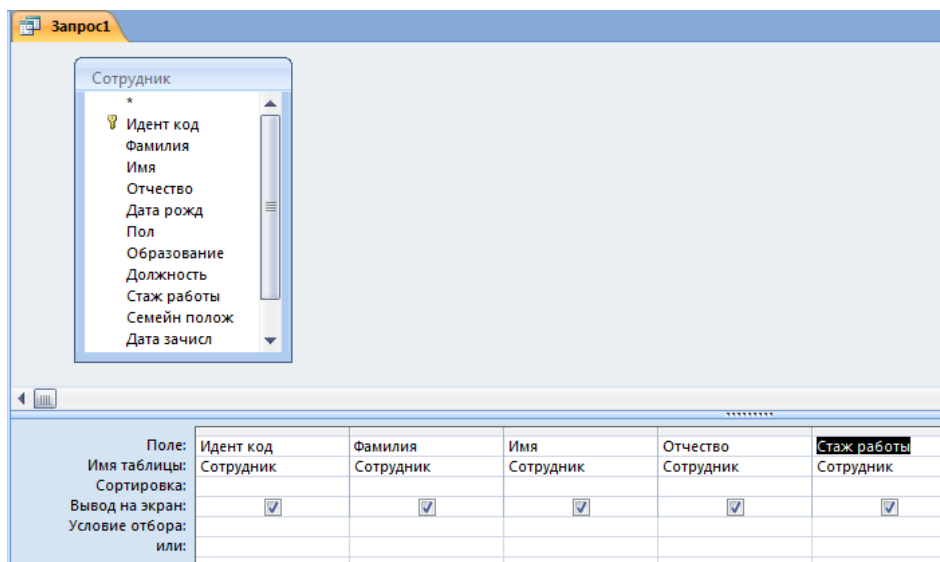
2.При выбранном режиме работы **Запрос** щелкнуть по кнопке **Создать. Меню Создание-Конструктор запросов -Сотрудник, Заккрыть**

3.Открывается окно **Новый запрос**, в котором выбрать режим создания запроса **Конструктор**.

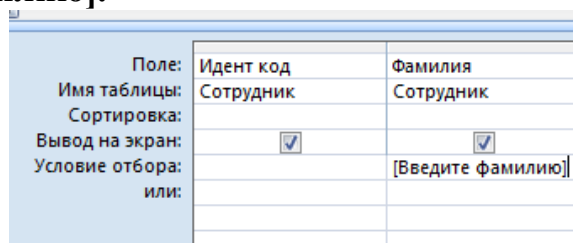
4.Открывается окно **Запрос 1: запрос на выборку** и активизируется окно **Добавление таблицы**, в котором выбрать таблицу **Сотрудник**, щелкнуть по кнопке **Добавить**, после чего закрыть окно **Добавление таблицы**.



5.С помощью мыши переместить поля **Фамилия, Имя, Отчество** и **Идент код, Стаж работы** из выбранной таблицы в бланк построения запроса.

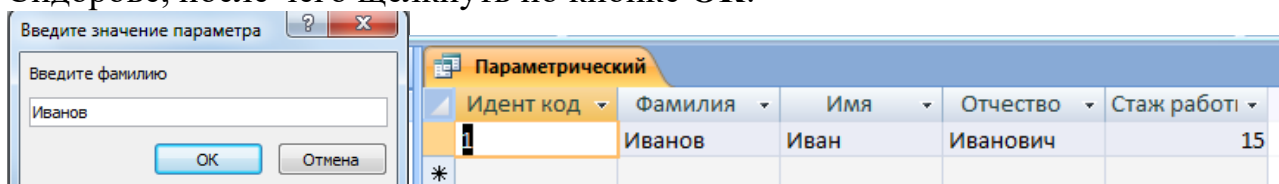


6. В столбце **Фамилию** в строке **Условие отбора** ввести в квадратных дужках сообщения, которое будет выводиться на экран при выполнении запроса, а именно: **[Введите фамилию]**.



7. Сохранить как с именем **Параметрический**

8. Выполните запрос, щелкнув по пункту меню **Открыть**. На экране появится окно **Введите значение параметра** для ввода фамилии сотрудника, информацию о котором необходимо получить, например об Иванове или Сидорове, после чего щелкнуть по кнопке **ОК**.



На экране появится таблица с данными о выбранном сотруднике.

**Задание 4.** Создать запросы на обновление и удаление данных.

**Задание 5.** Создать запросы к базе данных по выбранным вариантам заданий из Практической работы № 1.

## ПРАКТИЧЕСКАЯ РАБОТА № 3

**Тема: Создание формы. Управление внешним видом формы**

**Цель работы:** научиться создавать формы ввода-вывода; научиться создавать кнопочные формы.

**Оборудование:** ПК, интернет, программное обеспечение – MS Access, инструкции по выполнению работы.

**Справочный материал:**

Форма – это средство, упрощающее ввод, редактирование и отображение информации, хранящейся в таблицах базы данных. Она представляет собой окно с набором элементов управления.

Форма сама по себе не хранит информацию, она просто обеспечивает удобный способ доступа к информации, хранящейся в одной или нескольких таблицах. Формы по сравнению с обработкой данных в режиме таблицы обладают следующими преимуществами:

- Форма позволяет в каждый момент сфокусировать внимание на отдельной записи;
- Элементы управления на форме можно расположить логичным образом, облегчающим чтение и работу с данными;
- Отдельные элементы управления обладают возможностями облегчить ввод и изменение отдельных данных;
- Некоторые объекты баз данных, такие как рисунки, анимации, звуки и видеоклипы, могут отображаться только в режиме формы, но не в режиме таблицы.

Создание кнопочной формы.

Кнопочное меню представляет собой форму, на которой расположены элементы управления – кнопки с поясняющими надписями. Щелчок на кнопке открывает соответствующую таблицу, запрос, форму или отчет. Меню – удобный инструмент работы с базами данных, и он практически всегда присутствует в базах созданных для предприятий или фирм.

Кнопочное меню создают с помощью Диспетчера кнопочных форм.

**Содержание работы:**

**Задание 1.** Создать форму к базе данных


- 1.Откройте базу данных «Библиотека». Создайте форму с помощью Мастера форм на базе таблицы Сотрудник.
- 2.Откройте таблицу Сотрудник. Выберите закладку Формы, щелкните мышкой по кнопке Другие формы. В появившемся диалоговом окне выберите Мастер форм.
- 3.В поле Таблицы/Запросы выберите таблицу Сотрудник, в поле Доступные поля выберите поля Фамилия, Имя и перенесите их стрелкой в поле Выбранные поля. Также перенесите поля Дата рождения Номер телефона, Домашний адрес, щелкните по кнопке Далее.
- 4.Выберите внешний вид формы – Табличный, щелкните по кнопке Далее.
- 5.Выберите требуемый стиль (н-р, Обычная), щелкните по кнопке Далее.
- 6.Задайте имя формы Личные данные и щелкните по кнопке Готово. В результате получите форму, в которой можно менять данные и вводить новые значения.
- 7.Закройте форму.
8. Аналогично создайте формы к двум другим таблицам БД, разных видов.

## **Задание 2.** Создать форму с помощью инструмента Пустая форма


1. Создайте форму Личные данные с помощью инструмента Пустая форма
2. На вкладке Создание в группе Формы щелкните Пустая форма. Access открывает пустую форму в режиме макета и отображает область Список полей.
3. В области Список полей щелкните знак плюс (+) рядом с таблицей или таблицами, содержащими поля, которые нужно включить в форму.
4. Чтобы добавить поле к форме, дважды щелкните его или перетащите его на форму. Чтобы добавить сразу несколько полей, щелкните их последовательно, удерживая нажатой клавишу CTRL. Затем перетащите выбранные поля на форму.
5. Закройте окно списка полей.
6. Перейдите в режим Конструктора

Примечание 1. Размер окошка для названия поля и для его значений меняются мышкой. Для этого выделите черный квадратик рамки (рамка станет цветной), установите курсор на границу рамки и с помощью двунаправленной стрелки измените размеры рамки.

Примечание 2. С помощью кнопок панели инструментов Шрифт меняйте соответственно цвет фона, текста, линии/границы и т.д.

7. Расположите элементы удобно по полю.
  8. Задайте размер текста поля Фамилия равным 24 пт, шрифт - синего цвета.
  9. Сохраните форму с именем Данные студентов.
  10. Посмотрите все способы представления форм: в режиме Конструктора, режиме Макета и режиме Форм.
  11. Закройте форму, сохранив ее.
  12. Добавьте в таблицу Личные данные логическое поле Институт (т.е., собирается ли в дальнейшем учащийся поступать в институт). Значение этого поля ДА или НЕТ.
  13. Откройте таблицу Личные данные в режиме Конструктор. Добавьте поле с именем Институт и типом Логический. Закройте таблицу.
  14. Перейдите на закладку Формы и откройте форму Личные данные в режиме Конструктор
  15. Щелкните по кнопке Список полей на панели инструментов, выделите название Институт и перетащите его мышкой в область данных, появится значок  и надпись Институт.
  16. Расположите новые элементы по правилам оформления формы (с помощью мыши).
  17. Закройте Список полей
- Примечание 3. Если флажок установлен, поле в таблице имеет значение ДА, если снят, то НЕТ.
18. Перейдите в режим Раздельная форма и посмотрите записи. Установите флажки у восьми разных сотрудников.
  19. Закройте форму, ответив утвердительно на вопрос о сохранении.

## **Задание 3.** Создать кнопочную форму с помощью Конструктора

1. Щелкните по кнопке Создать. Выберите Конструктор. Появится пустая форма. Задайте мышкой ширину формы, равную 10см, а высоту – 7см.
  2. Выберите на панели инструментов Элементы управления> кнопку Аа – Надпись. Курсор мышки примет вид крестика с «приклеенной» буквой А. Щелкните мышкой по месту начала надписи и введите: БАЗА ДАННЫХ «БИБЛИОТЕКА».
  - 3.Нажмите клавишу Enter. Выберите размер букв 18, а выравнивание - по центру. Цвет фона – голубой. Растяните мышкой надпись на ширину окна.
  4. Выберите на панели элементов значок  - Кнопка. Щелкните мышкой по тому месту области данных, где должна быть кнопка. Появится диалоговое окно Создание кнопок.
  - 5.Выберите категорию Работа с формой, а действие Открыть форму, и щелкните по кнопке Далее.
  - 6 Выберите форму Штатное расписание, открываемую этой кнопкой щелкните по кнопке Далее. В следующем окне также щелкните по кнопке Далее.
  - 7.В следующем окне поставьте переключатель в положение Текст, наберите в поле слова Штатное расписание и щелкните по кнопке Далее.
- Примечание 4. Размер и расположение кнопок можно менять мышкой в режиме Конструктор.
- 8.Аналогично создайте кнопки для форм Сотрудники и Состав семьи.
  - 9.Перейдите в режим формы. Теперь при щелчке мышью по соответствующим кнопкам будут открываться соответствующие формы для работы.
  - 10.Закройте форму.

**Задание 4.** Создать кнопочную форму с помощью Диспетчера кнопочных форм.

- 1.Откройте вкладку Работа с базами данных, команда - Диспетчер кнопочных форм. Вы получите диалоговое окно Диспетчер кнопочных форм.
- 2.Щелкните в этом окне по кнопке Изменить. В следующем окне щелкните по кнопке Создать и в появившемся окне измените содержимое полей для формы Сотрудники: Команду (открыть форму для изменения) и Форму выбирайте из списка, а не набирайте вручную. Щелкните по кнопке ОК.
3. Аналогично создайте еще три элемента кнопочной формы: Штатное расписание, Состав семьи.
4. Добавьте кнопку закрытия базы данных. Для этого щелкните по кнопке Создать, наберите в поле Текст слово Выход, а в поле Команда выберите Выйти из приложения. Закройте диалоговые окна.
- 5.Откройте окно Кнопочная форма в режиме Конструктора или Макета, измените цвет надписи и название вашей базы данных, сохраните форму.
- 6.Украсьте вашу форму рисунком. Для этого щелкните по значку Эмблема и выберите в открывшемся окне папку с рисунками, выберите понравившийся и вставьте в свою кнопочную форму.

7.Перейдите в режим формы, проверьте работу всех кнопок кнопочной формы.

8.Завершите работу с базой данных, нажав на кнопку Выход.

**Задание 5.** Создайте формы и кнопочную форму по выбранному варианту заданий из предыдущих работ.

#### **ПРАКТИЧЕСКАЯ РАБОТА № 4**

**Тема:** Создание меню различных видов. Модификация и управление меню.

**Цель работы:** овладение навыками создания и модификации структуры меню

**Оборудование:** ПК, интернет, программное обеспечение – MS Access, инструкции по выполнению работы.


### **Справочный материал:**

Для хорошего тона создают общий интерфейс базы данных, которая сдаётся в эксплуатацию. Пользователь может выбрать направление на общем интерфейсе, а затем переходить к группе задач, которые его интересуют. В учебных целях, создадим ещё одну форму, которая будет начальной для работы с базой данных. Конечно, рассмотренные варианты, создания интерфейсов с использованием макросов, гиперссылок, формы навигации, могут лечь в основу общего интерфейса, но мы постараемся создать новый вид интерфейса, основанный на элементах управления, которые ещё не рассматривались.

### **Содержание работы:**

**Задание 1.** Создать меню для разработанной базы данных Библиотека.

#### **1. Подготовка формы**

Создайте основу формы, что бы в дальнейшем её дополнять новыми элементами, для этого достаточно спроектировать заголовок формы. Откройте форму в режиме Конструктора, в поле заголовка вставьте рисунок, отражающий логотип организации, воспользовавшись пиктограммой , отмасштабируйте рисунок. Обратите внимание, что одновременно с рисунком на поле заголовка появилось окно для надписи, заполните его (не забывайте про свойства этого элемента), как показано на рисунке 1. Сохраните форму, например под именем «Общая форма».

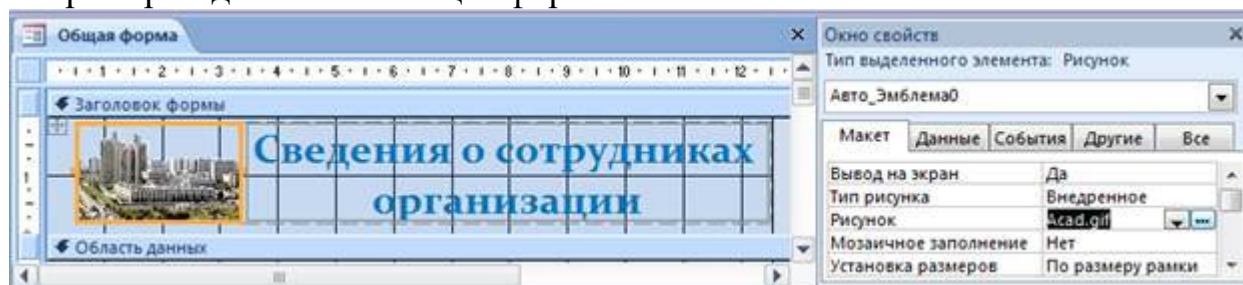


Рис. 1. Создание заголовка для общей формы

#### **2. Использование группы переключателей для выбора варианта просмотра объектов базы данных**

Откройте созданную форму в режиме Конструктор. На поле «Область данных» перенесите элемент управления «Группа переключателей» (Рис. 2), предварительно раскрыв на панели список с элементами управления. Переключатели обычно применяются в тех случаях, когда предлагается выбрать один вариант из нескольких предложенных (как правило, количество переключателей создают небольшое, в противном случае, удобнее использовать элемент управления – Список).



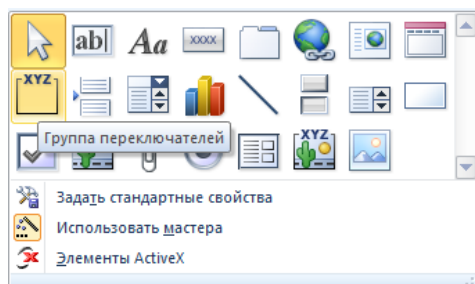


Рис. 2. Раскрытый список элементов управления

В появившемся окне (Рис. 3) каждая строка соответствует названию переключателя. Чтобы добавить переключатель, заполните текстом строку со звёздочкой (Рис. 3) и нажмите на кнопку **Далее >**. Мастер создания переключателей будет последовательно предлагать диалоговые окна, в которых следует выбирать варианты решения (Рис. 4, 5).

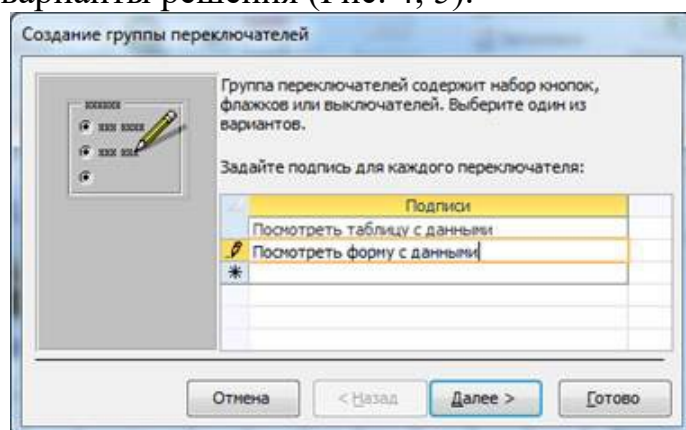


Рис. 3. Ввод текста около переключателя

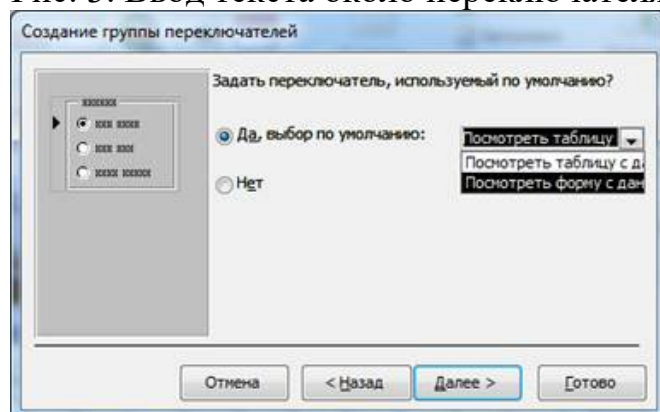


Рис. 4. Задание начального состояния переключателя

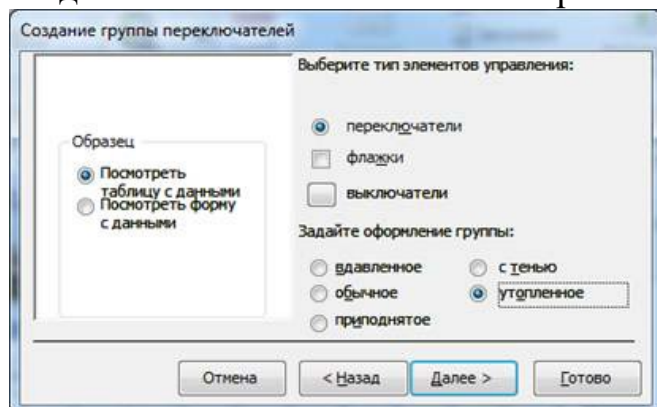



Рис. 5. Выбор варианта представления блока с переключателями

Создайте два элементарных макроса, которые будут использоваться для открытия таблицы «Личные сведения» при щелчке по первому переключателю и для открытия формы, например, «Форма с макросами». Напомним, что для создания независимого макроса, необходимо на вкладке «Создание» щёлкнуть по значку  - Макросы. Открыть список макрокоманд, выбрать **ОткрытьФорму**, и заполнить бланк макроса (Рис. 6). В рассматриваемом примере показан макрос для открытия формы, этот макрос понадобится для подключения к переключателю номер 2. Сохраните макрос, например с именем «Подключить форму». Для переключателя с номером 1 потребуется макрос для открытия таблицы, создайте новый по аналогии с примером и сохраните под именем, например, «ЛичныеСвед».

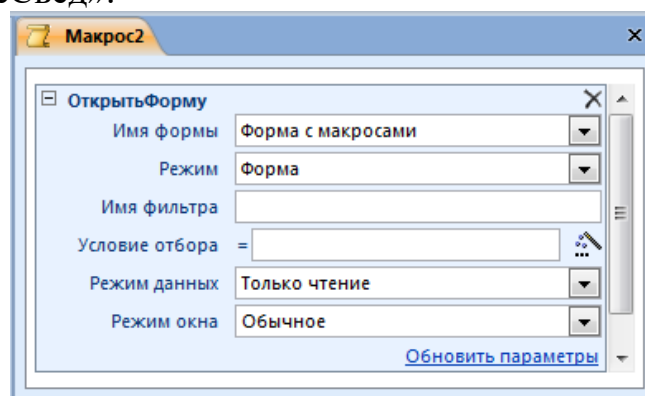



Рис. 6. Бланк макроса для открытия формы «Форма с гиперссылками»

Проведите операцию назначения макроса событию, которое вызывает пользователь при взаимодействии с переключателем. Выделите на форме в режиме Конструктор первый переключатель, в окне свойств активизируйте ярлык «События», для строки «Кнопка вниз» выберите из списка  макрос «ЛичныеСвед» (Рис. 7). Почему выбрали событие «Кнопка вниз»? Когда создавали поле с переключателями, тогда поставили отметку, что первый будет активным, это значит, что он получил фокус, если этому событию назначить макрос, то при открытии формы, автоматически будет открываться таблица «Личные сведения», что бы этого не случилось, выбрано событие, которое имитирует нажатие мышкой на кнопку в элементе переключателя.

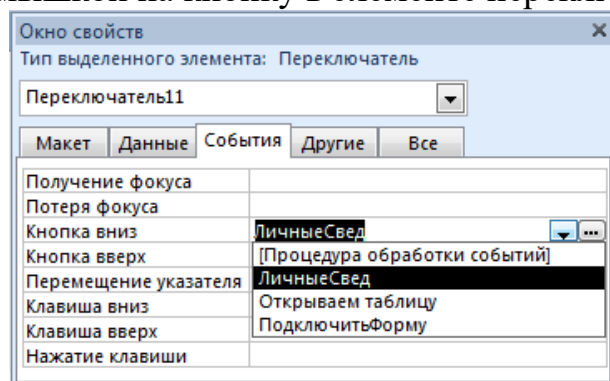


Рис. 7. Назначение макроса событию «Кнопка вниз»

Второй переключатель становится активным только после того, как по кнопке будет проведён щелчок мыши, это значит, либо в нём устанавливается фокус, либо кнопка идёт вниз (Рис. 8). Поэтому макрос «ПодключитьФорму» можно привязать к одному из указанных событий. Кстати, событие «Кнопка

вверх» происходит в тот момент, когда производится нажатие на любую другую кнопку, следовательно, можно написать макрос для этого события, который будет управлять закрытием формы, таблицы или запроса.

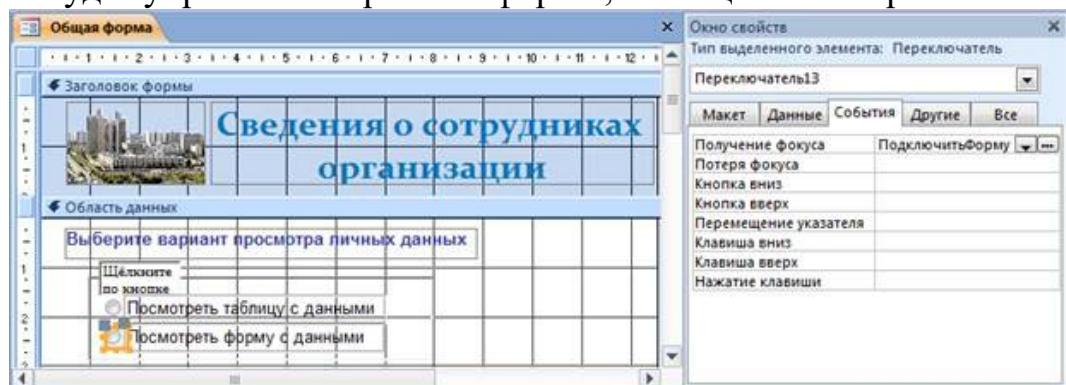








Рис. 8. Расположение группы переключателей на общей форме

### 3. Использование списков на форме

Среди элементов управления существует два элемента:  - Поле со списком и  - Список. Которые имеют аналогичные свойства, разница заключается в том, что поле первый элемент имеет кнопку для раскрытия списка, а второй имеет только линейку прокрутки. Но, при внедрении на форму списка с помощью Мастера, можно добиться различных эффектов. Рассмотрим пример, когда с помощью элемента управления  «Список» на форме можно отобразить таблицы с данными.

3.1. Откройте форму «Общая форма». Раскройте окно с элементами управления проверьте состояние команды  **Использовать мастера**, этот элемент должен быть активным. Поместите на форму элемент управления  - Список. В окне мастера «Создание списков» выберите отметку  **Объект "список" получит значения из другой таблицы или другого запроса.** (Рис. 9).

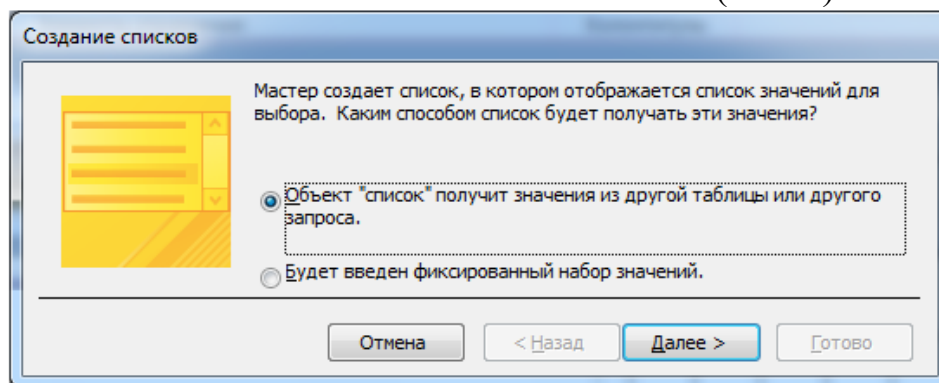



Рис. 9. Окно мастера для создания списка на форме

3.2. Напомним, что при работе с Мастером следует в последующих окнах выбирать вариант, который устраивает разработчика и нажимать на кнопку  **Далее >**. На последующих рисунках отображены шаги работы с Мастером при создании списка на форме. Так, прежде всего, следует выбрать источник данных, для примера это будет таблица «Личные сведения» (Рис. 10). На следующем этапе (Рис. 11) в будущий список переносят в любой

последовательности наименования полей из источника данных. Для удобного восприятия данных, устанавливается порядок сортировки (Рис. 12).

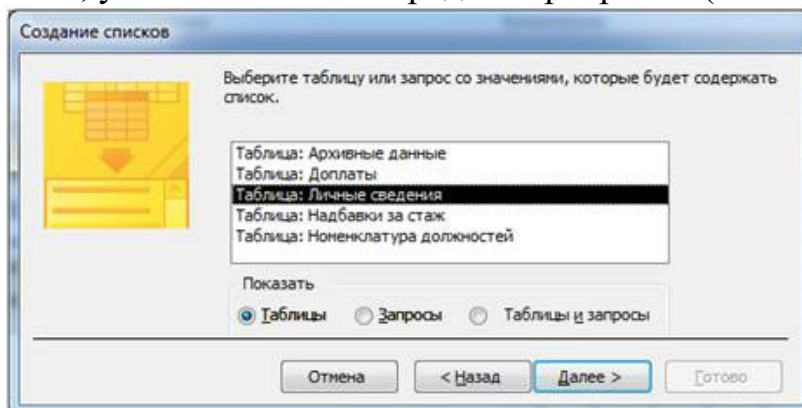


Рис. 10. Выбор наименования таблицы, поля из которой понадобятся

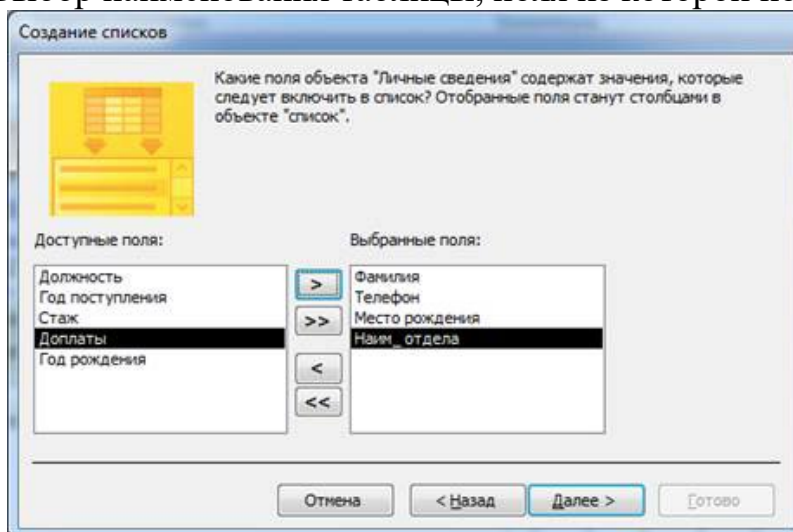


Рис. 11. Отбор полей для таблицы

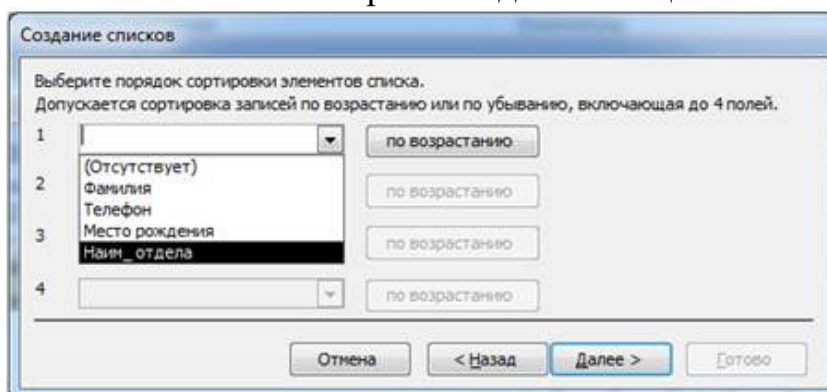


Рис. 12. Установка порядка сортировки в таблице

На этапе просмотра готовой таблицы (Рис. 13) уберите отметку в элементе ☐ Скрыть ключевой столбец (рекомендуется), так как в таблице «Личные сведения» ключевым элементом было назначено поле «Фамилия».



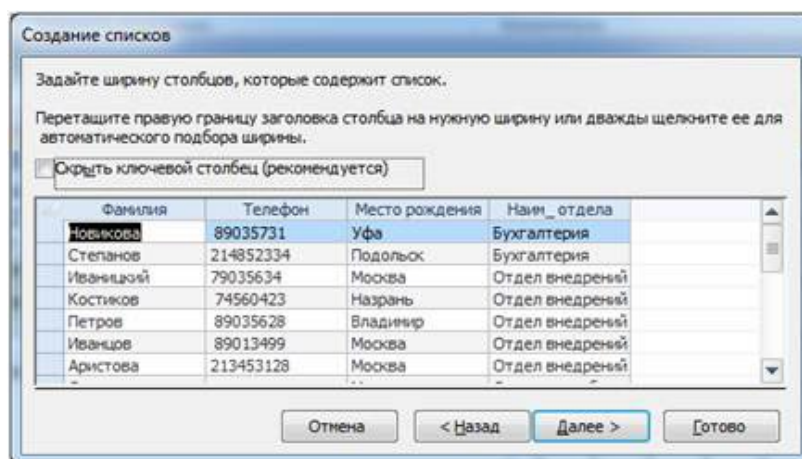


Рис. 13. Отображение таблицы в Мастере

Обратите внимание, что на следующем шаге будет предложено выбрать поле, которое в дальнейшем можно будет использовать для формирования запроса к данной таблице, например, выделите поле «Место рождения» (Рис. 14).

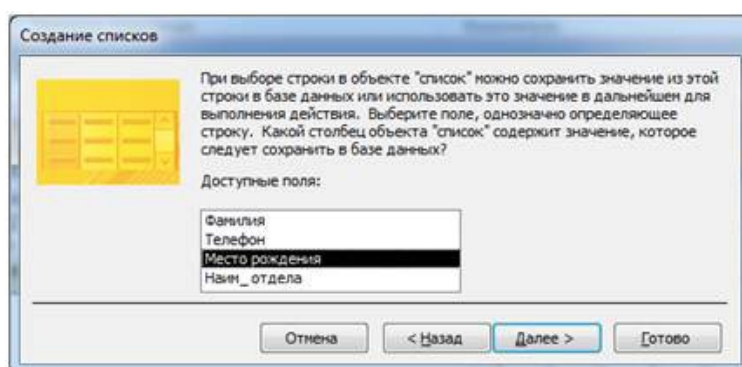


Рис. 14. Выбор столбца в объекте для дальнейшего использования

На заключительном этапе работы с Мастером (Рис. 15) требуется задать заголовок для списка, который будет отображаться в виде таблицы на форме.

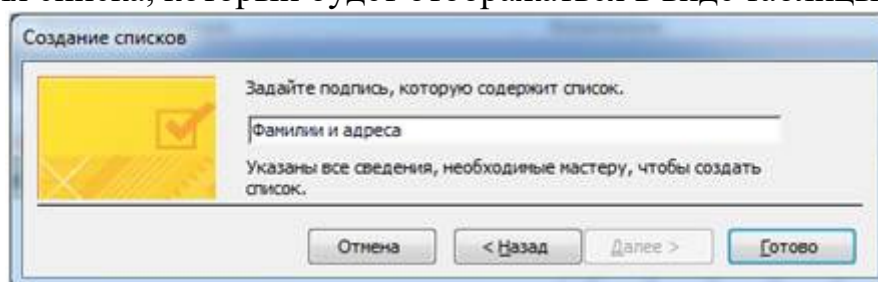



Рис. 15. Задание подписи, который будет содержать созданный список

Нажав на кнопку **Готово**, вы увидите в режиме Конструктор форму (Рис. 16), на которой будут только заготовка для списка и его заголовок. Можете воспользоваться окном свойств этого элемента, что бы установить те параметры, которые вас устраивают. Чтобы увидеть на форме результаты творчества, перейдите в режим формы , на которой отобразится список, созданный с помощью Мастера (Рис. 17).

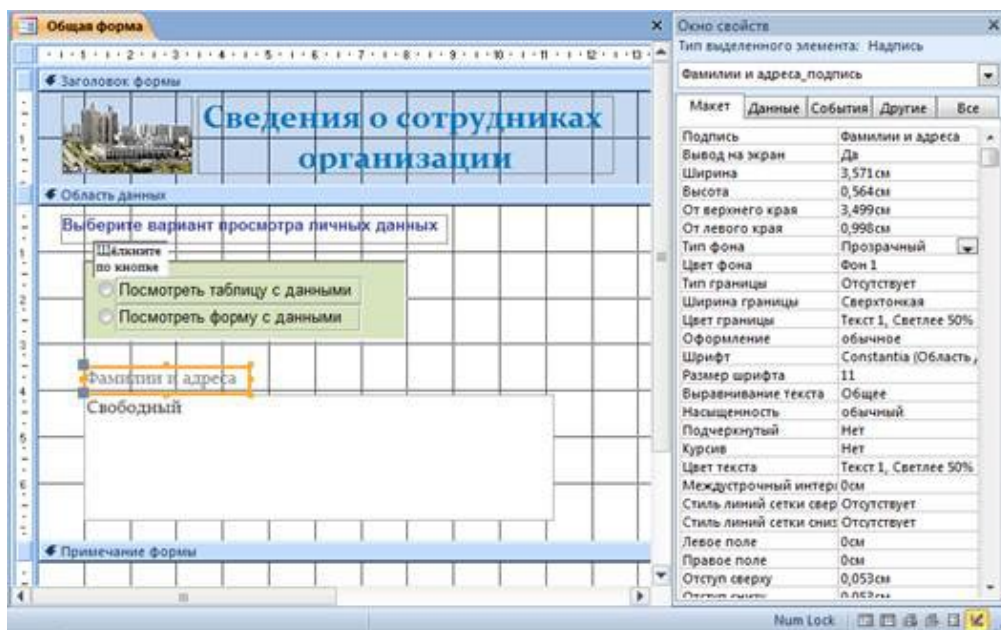


Рис. 16. Отображение созданного списка на форме в режиме Конструктор

Фамилии и адреса

Новикова	89035731	Уфа	Бухгалтерия
Степанов	214852334	Подольск	Бухгалтерия
Иваницкий	79035634	Москва	Отдел внедрений
Костиков	74560423	Назрань	Отдел внедрений
Петров	89035628	Владимир	Отдел внедрений
Иванцов	89013499	Москва	Отдел внедрений



Рис. 17. Список на форме, который был создан с помощью Мастера

Как видите, без использования, каких либо средств, на форме можно расположить список, сформированный в виде таблицы. Следует отметить, что данный элемент управления позволяет подключать программные модули, написанные на Visual Basic, чтобы вести обработку данных.




#### 4. Создание программных кодов для обработки событий

Событием называется действие, которое вызывает пользователь или генерирует система. Событие производится над объектом, поэтому необходимо совершить действие, например, щёлкнуть мышкой, нажать на клавишу, передвинуть указатель мыши. Система совершает действия при загрузке формы, закрытии окон и т.п. Общение пользователя с интерфейсом любого приложения состоит из цепочки событий, которые он совершает. После того, как событие совершено, необходимо подключить программу, в которой заложен алгоритм изменения свойств определённого элемента управления, либо осуществление логических операций, либо проведение преобразования данных (в том числе и выполнения вычислений). Программные коды формируются с помощью макрокоманд, когда создаётся макрос. Более интересно создавать программные модули с помощью языка программирования Visual Basic (VB). В данном разделе автор поставил перед собой задачу – показать некоторые приёмы разработки программ обработки событий с помощью самостоятельного использования VBE – Visual Basic Editor. Безусловно, для серьёзных программ требуются знания в области алгоритмизации и программирования, а так же опыт работы

с VB (автор не теряет надежду, что большинство, изучающих практику работы с приложениями MS Office, начнут серьезно заниматься программированием).

Попытаемся использовать, накопленный опыт по разработке интерфейсов базы данных, при создании программных модулей. В качестве учебной задачи, остановимся на совершенствовании начальной формы под именем «Общая форма». Для этого разместим на форме элементы управления:  - Поле со списком и  - Кнопка. Затем напишем небольшую программу для этих элементов управления, а затем покажем, как можно обойтись без кнопки. В поле со списком должны войти наименования форм, которые созданы для подразделений организации. А при выборе из списка необходимого наименования формы, должна открываться форма. Управлять этим процессом будем с помощью кнопки.

#### 5. Размещение элементов управления на форме

Создадим на форме Поле со списком с помощью Мастера. Прежде убедитесь в том, что команда  **Использовать мастера** находится в активном состоянии (её можно увидеть в окне с элементами управления). На этот раз в элементах управления выберите , после открытия окна выберите режим  **Будет введен фиксированный набор значений.** (Рис. 18).

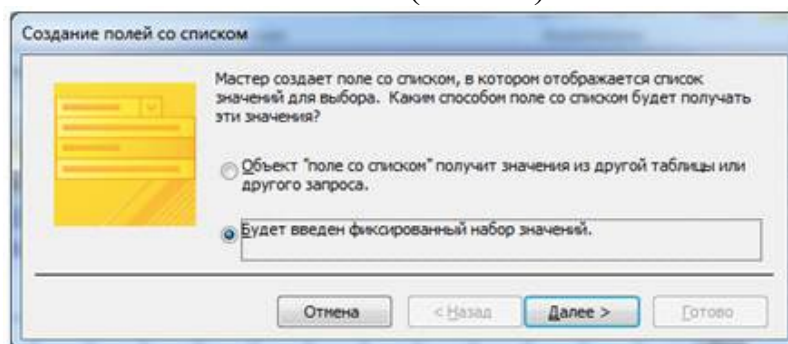


Рис. 18. Выбор варианта формирования списка

На следующем шаге заполните строки для столбца (выберите один столбец), в список введите наименования готовых форм, которые обозначены в окне переходов базы данных. Для начала введите три строки (Рис. 19), в дальнейшем будет показано, как такой список можно дополнить новыми записями. Введите заголовок для списка (Рис. 20).

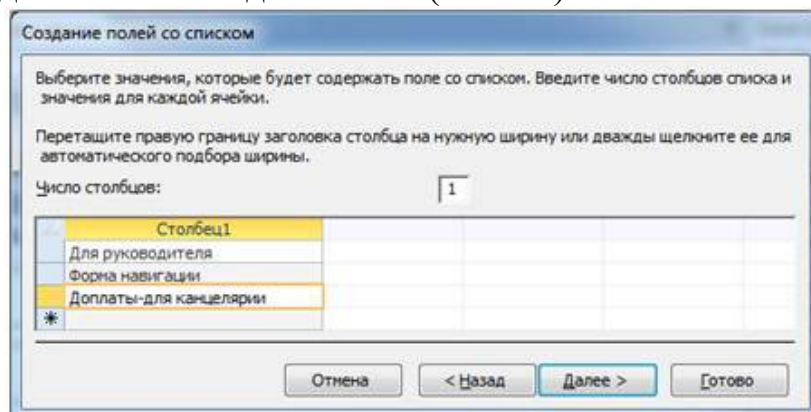


Рис. 19. Заполнение строк таблицы наименованиями форм

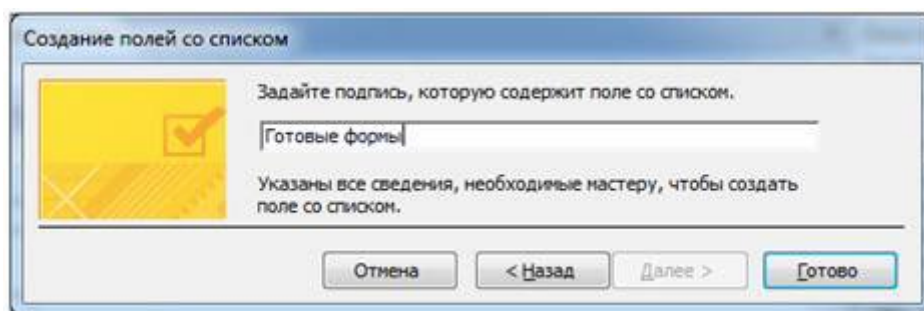





Рис. 20. Ввод заголовка для списка

В режиме Конструктор на форме «Общая форма» появится только заголовок для списка и окно списка. Опять воспользуйтесь окном свойств и отредактируйте текст, размер, фон и т.п. Увидеть результат создания списка можно после того, как форма будет запущена. Если возникает необходимость дополнить список новыми записями, то это можно сделать непосредственно в

режиме  - Форма. Раскройте список, появится значок  - Изменить элементы списка (Рис. 21 слева), щёлкните по значку, после чего в него можно добавлять новые наименования, удалять и редактировать записи (Рис. 21 – справа). В примере добавлена новая строка «Список для отдела кадров». Если вы находитесь в режиме Конструктор, то щёлкните правой кнопкой мыши по элементу список, а затем в раскрывшемся меню по строке  Изменить элементы списка...

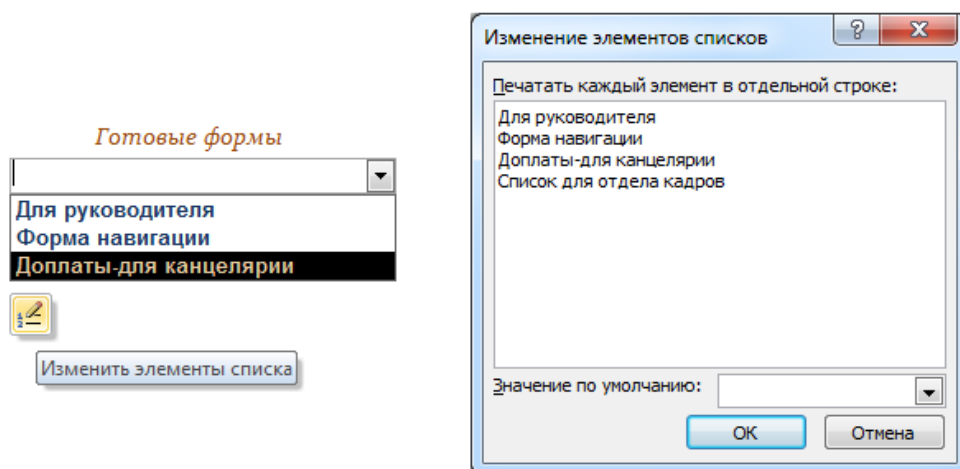


Рис. 21. Вызов диалогового окна для изменения элементов списка



Теперь, на форме создайте элемент  - Кнопка, в этом случае  Использовать мастера следует отключить, т.к. в режиме Конструктор можно использовать свойства этого элемента и обойтись без Мастера создания кнопок, это будет гораздо быстрее. На рисунке 22 показан фрагмент формы с окном свойств кнопки. При работе со свойствами: *Подпись*, *Расположение подписи к рисунку*, *Рисунок* (открыто окно для выбора варианта рисунка, который можно разместить на кнопке) и другие, которые позволяют добиться определённого результата.

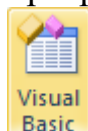




Рис. 22. Размещение рисунка и подписи на кнопке с помощью окна свойств

## 6. Разработка программных кодов

Цель создания программных кодов – организовать открытие готовых форм при выборе определённого названия из списка на форме. Чтобы создать программу, необходимо открыть редактор Visual Basic for Applications (VBA).



Эту операцию можно выполнить с помощью кнопки на панели, предварительно открыв вкладку «Работа с базами данных», после чего будет открыт редактор, внешний вид которого показан на рисунке 23. Редактор имеет собственный интерфейс, на панели которого располагается строка меню и кнопки быстрого вызова. Основное поле редактора делится на несколько областей, в рассматриваемом примере их три. Область проектов слева (Project), в которой отображается форма базы данных «Общая форма». Вторая область – Свойства (Properties). Третья область – Коды (Code), в которой создаётся программа (эта область содержит созданную программу для обработки событий при работе с формой и элементами управления на ней).

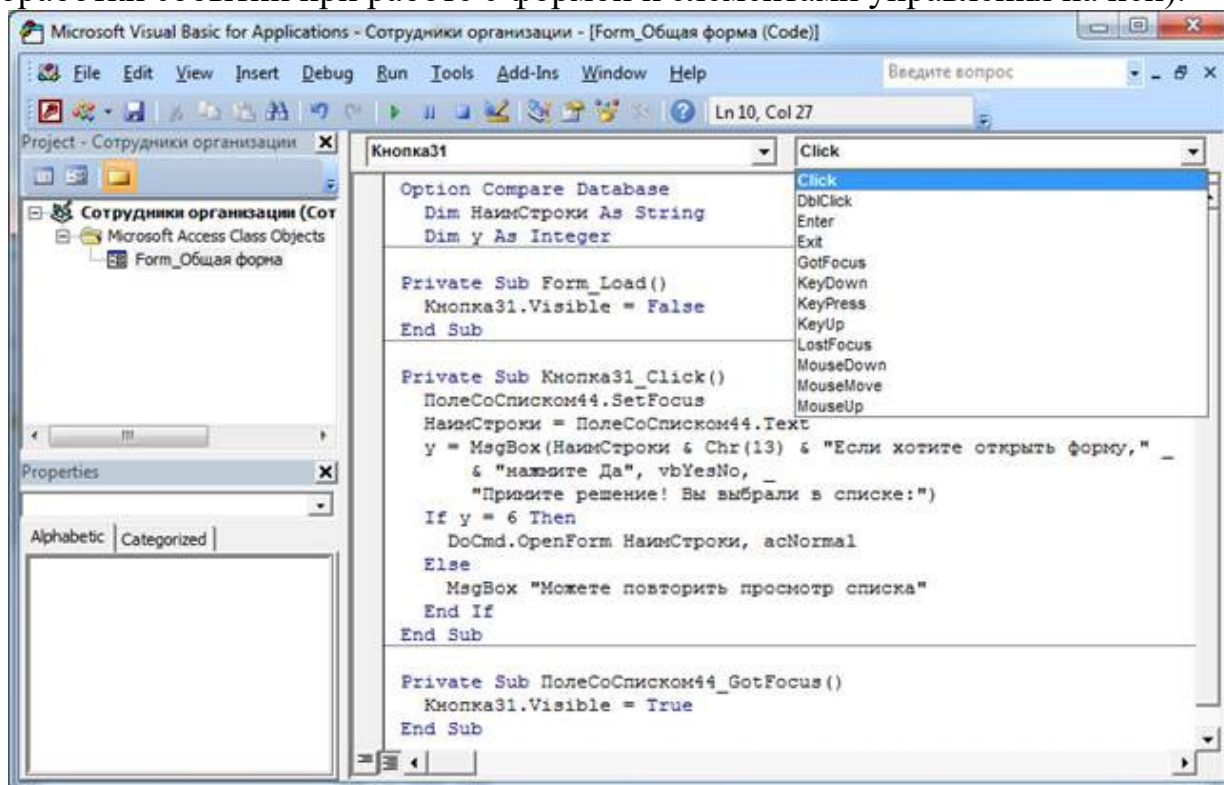


Рис. 23. Программные коды описания переменных и загрузки формы

Напомним, что Visual Basic относится к событийно-процедурным языкам программирования. Их особенностью является то, что логика программы основывается на выделении процедур, которые обрабатывают события, вызываемые либо системой, либо пользователем. При составлении программы сначала вводят общее (General) объявление переменных (Declaration) для формы, а затем создают отдельные процедуры. В рассматриваемом примере понадобится использовать две переменные, первая переменная - «НаимСтроки» предназначена для временного сохранения текста из строки, которую выбрали в раскрывающемся списке элемента на форме «Поле со списком». Вторая переменная понадобится для анализа нажатия на кнопку в диалоговом окне (о нём будет сказано ниже), эту переменную в программе обозначим буквой у, присвоим ей тип данных – целый (Integer), как показано на рисунке 24. Переменные вступят в действие после события – открыть форму «Общая форма».

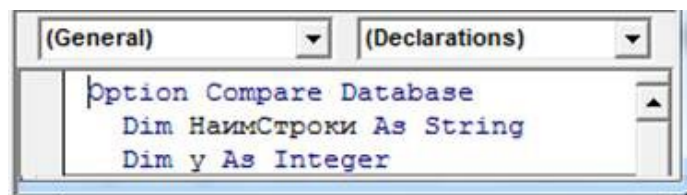



Рис. 24. Программа с общим описанием переменных для программы

Одновременно с загрузкой формы, мы решили скрыть на форме кнопку  Открыть форму, это делается для того, чтобы не возникла ошибка при первом нажатии на кнопку пользователя, когда в строке списка ничего нет. Процедура, которая отвечает за процесс, скрывания кнопки на форме (Рис. 25), состоит всего из одного оператора: `Кнопка31.Visible = False` кнопке, которой система присвоила номер 31, устанавливаем в свойстве `Visible` (Видимый) параметр – `False` (скрыть).

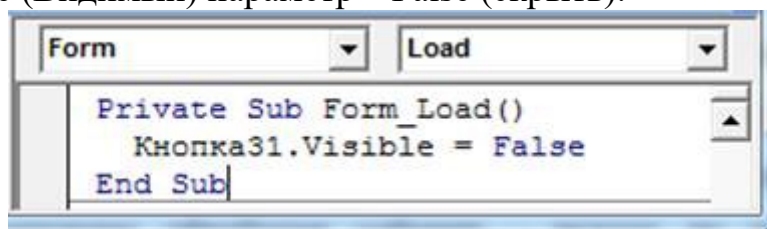



Рис. 25. Процедура открытия формы

В программе есть ещё одна процедура (Рис. 26), которая необходима для отображения кнопки на форме в тот момент, когда поле со списком получает фокус, т.е. это поле активно.

```
Private Sub ПолеСоСписком44_GotFocus()  
    Кнопка31.Visible = True  
End Sub
```

Рис. 26. Процедура отображения кнопки на форме, в случае наведения мышки на поле списка (получение фокуса)

Посмотрите программу для обработки события – нажать на кнопку  Открыть форму, которая находится на форме целиком, для того, что бы легче было ориентироваться при её создании и правки (Рис. 27).

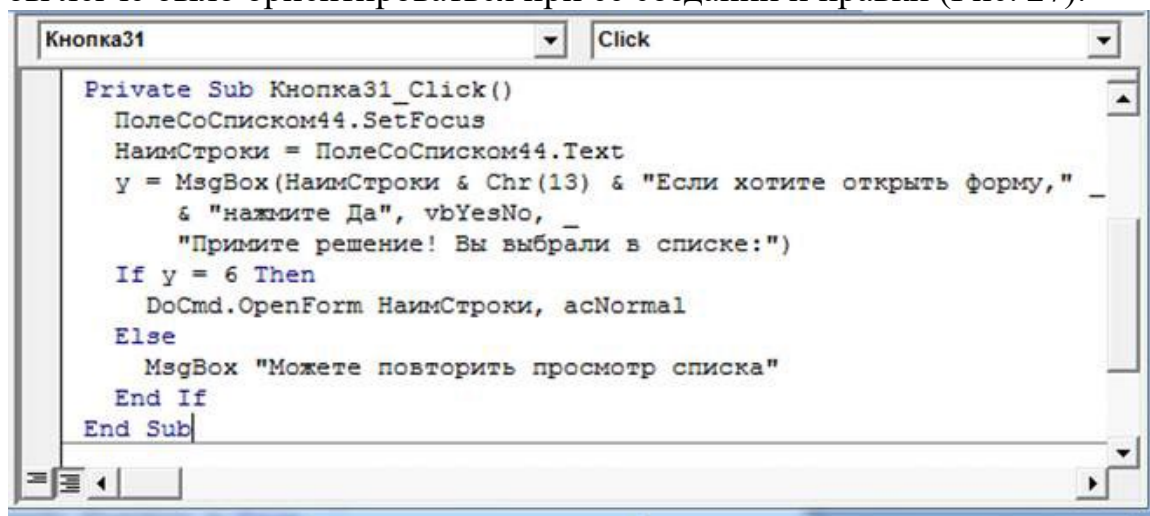


Рис. 27. Процедура обработки события – нажать на кнопку

Процедура Private Sub Кнопка31\_Click() отвечает за перехват выделенного наименования формы в строке списка, подготовки диалогового окна для пользователя о принятии решения по выполнению команды загрузки формы на экран компьютера. Поэтому, первым оператором в процедуре стоит ПолеСоСписком44.SetFocus – установить фокус в строке поля со списком. Обратите внимание, что в программе наименования объектов и переменных пишут слитно (символ пробел недопустим). Вторая строка программы обозначает, что переменной НаимСтроки присваивается значение, которое находится в активной строке ПолеСоСписком44, после точки идёт указание свойства этого объекта – Text (то, что находится в виде записи в строке списка, преобразуется в текстовую переменную). Переменной y присваивается значение из функции MsgBox() - вывод на экран сообщения. На рисунке 27 эта функция занимает три строки (можно всё записать в одной строке), это сделано, что бы было удобнее читать содержимое функции, которое находится в круглых скобках. Признаком переноса строки в программном коде является сочетание двух символов – Пробел и знак подчёркивания. С помощью функции MsgBox() формируется диалоговое окно, в котором на основном поле появляется надпись: текст, который находится в переменной НаимСтроки, переход на новую строку (функция Chr(13)), продолжение текста – «Если хотите открыть форму, нажмите Да»; далее идёт стандартное описание кнопок VbYesNo, которые появятся на диалоговом окне («Да» и «Нет»); сообщение для информационной строки «Примите решение! Вы выбрали в списке:». При нажатии на кнопку «Да», функция сгенерирует целое число 6, которое будет присвоено переменной y. Далее, используется конструкция условного оператора If (запись условия) Then. После ключевого слова Then вставляют операторы, которые будут задействованы, если условие выполняется. Ключевое слово Else необходимо, чтобы после него вставить операторы, которые будут задействованы, если условие не будет выполнено.

Заканчивается условный оператор командой - End If . В данном примере, если условие будет выполнено (нажата кнопка «Да»), то выполнится открытие объекта базы данных форма с наименованием, которое содержится в переменной НаимСтроки. В противном случае, на экране пользователя появится окно с сообщением «Можете повторить просмотр списка»

В результате разработки главной формы («Общая форма»), пользователи получают возможность использовать все виды таблиц, запросов и форм, к которым можно обратиться с главного интерфейса базы данных, а так же с интерфейсов, которые предназначены для отдельных подразделений организации. На рисунке 28 представлен интерфейс пользователя - «Общая форма» с сообщением в диалоговом окне о выборе другой формы.

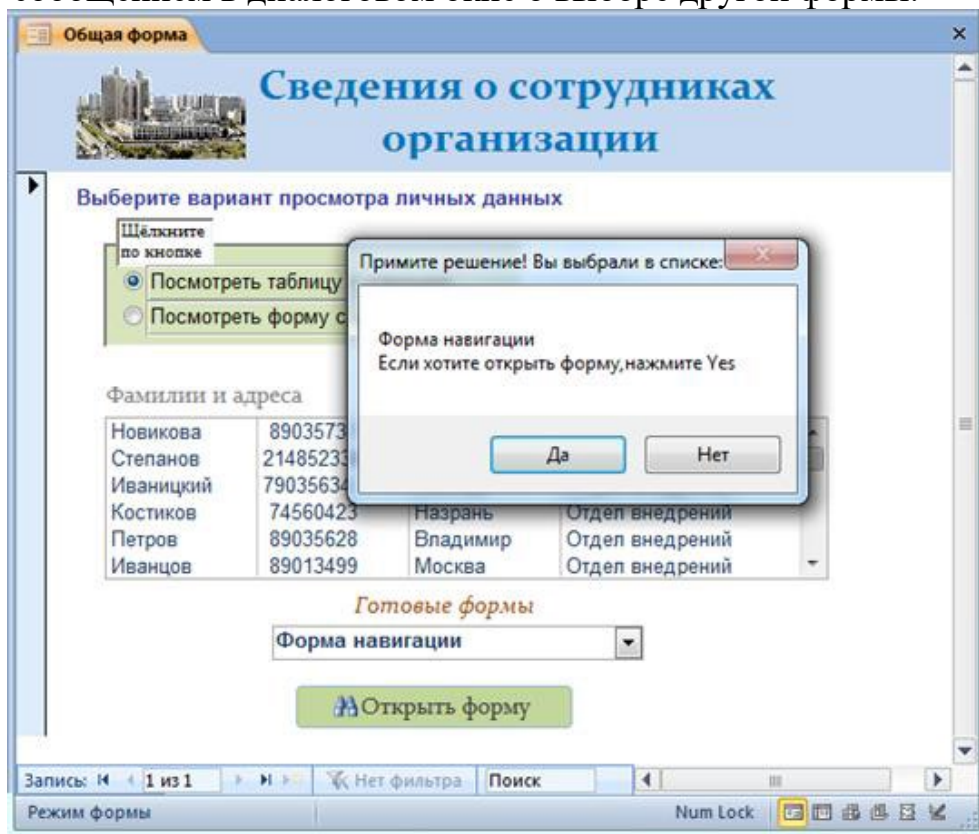


Рис. 28. Внешний вид общей формы в базе данных

**Задание 2.** Создать меню для разработанной по выбранным вариантам базы данных из предыдущих работ.



## ПРАКТИЧЕСКАЯ РАБОТА № 5

### Тема: Создание интерфейса входной формы

**Цель работы:** овладение навыками создания интерфейса с пользователем.

**Оборудование:** ПК, интернет, программное обеспечение – MS Access, инструкции по выполнению работы.

#### Содержание работы:

**Задание 1.** Создать для каждой таблицы базы данных Библиотека формы, содержащие кнопки, поля ввода, поля со списком. Добавить изображение, кнопки закрытия формы и выхода из приложения.

Например, для таблицы Сотрудники

**Задание 2.** Создать для каждой таблицы, созданных баз данных по выбранным вариантам заданий из предыдущих работ, формы, содержащие кнопки, поля ввода, поля со списком. Добавить изображение, кнопки закрытия формы и выхода из приложения.

## ПРАКТИЧЕСКАЯ РАБОТА № 6

**Тема:** Создание файла проекта базы данных. Использование исполняемого файла проекта БД, приемы создания и управления.

**Цель работы:** научить применять операторы языка SQL для создания БД

**Оборудование:** ПК, интернет, программное обеспечение – MS Access, инструкции по выполнению работы.

**Содержание работы:**

**Задание 1.** Создать базу данных с именем prim, используя язык SQL

```
CREATE DATABASE prim
```

Создаем таблицу «Комнаты» с именем room, состоящую из двух полей:

- номер комнаты roomnum, символьное поле типа CHAR длиной 14 символов;

- номер телефона tel, тип поля INTEGER длиной 4 знака.

Первичным ключом таблицы является поле roomnum.

```
CREATE TABLE room ;
```

```
(roomnum C(14) PRIMARY KEY, ;
```

```
Tel
```

```
I(4) )
```

Создаем таблицу «Сотрудники» с именем person, состоящую из шести полей:

- табельный номер tubnum, символьное поле типа CHAR длиной 4 символа;

- ФИО fio, символьное поле типа CHAR длиной 15 символов с ограничением на обязательное заполнение данными;

- должность jt, символьное поле типа CHAR длиной 15 символов;

- дата рождения birthd, поле типа DATE (дата: год — месяц — день);

- город city, символьное поле типа CHAR длиной 15 символов;

- номер комнаты roomnum, символьное поле типа CHAR длиной 14 символов.

Первичным ключом таблицы является поле tubnum. Вторичный ключ roomnum с именем perroom связывает данную таблицу с таблицей room с заданным первичным ключом по аналогичному полю roomnum:

```
FOREIGN KEY roomnum TAG perroom REFERENCES room
```

```
CREATE TABLE person;
```

```
( tubnum C(4) PRIMARY KEY,;
```

```
fio C(25) NOT NULL,;
```

```
jt C(15),;
```

```
birthd D,;
```

```
city C(15),;
```

```
roomnum C(4),;
```

```
FOREIGN KEY roomnum TAG perroom REFERENCES room)
```

Создаем таблицу «Дети» с именем baby, состоящую из трех полей:

- табельный номер tubnum, символьное поле типа CHAR длиной 4 символа;

- имя ребенка bname, символьное поле типа CHAR длиной 10 символов;

– возраст age, числовое поле типа NUMERIC с длиной действительной части 4 символа;

Первичным ключом таблицы является конкатенация полей tubnum и bname, ключ имеет имя baby. Вторичный ключ tubnum с именем btub связывает данную таблицу с таблицей person с заданным первичным ключом по аналогичному полю tubnum.

```
CREATE TABLE baby;  
( tubnum C(4),;  
  bname C(10),;  
  age N(4,0),;  
  PRIMARY KEY tubnum+bname TAG baby,;  
  FOREIGN KEY tubnum TAG btub REFERENCES person)
```

Создаем представление с именем mbaby, содержащее поле fio из таблицы person и поле bname из таблицы baby. Представление образует временную таблицу, содержащую список имен детей с ФИО сотрудника, являющегося родителем данного ребенка:

```
CREATE VIEW mbaby;  
AS SELECT person.fio, baby.bname;  
FROM person, baby;  
WHERE person.tubnum= baby.tubnum
```

**Задание 2.** Создать базу данных по индивидуальным заданиям, используя язык SQL.

Варианты заданий для самостоятельного выполнения

*Вариант 1*

Создать базу данных, состоящую из следующих трех таблиц с перечисленными полями.

Таблица 1 Книги: Регистрационный номер; Название книги; Название издательства; Год издания; Количество страниц.

Первичным ключом таблицы является поле «Регистрационный номер». Внешним ключом является поле «Название издательства»

Таблица 2 Авторы: Регистрационный номер; ФИО; Год рождения; Город проживания.

Первичным ключом таблицы является конкатенация полей «Регистрационный номер» и «ФИО». Внешним ключом таблицы является поле «Регистрационный номер».

Таблица 1 связана с Таблицей 2 через ключи, образованные полями «Регистрационный номер».

Таблица 3 Издательства: Название издательства; Владелец издательства; Адрес издательства; Город издательства.

Первичным ключом таблицы является поле «Название издательства».

Таблица 3 связана с Таблицей 1 через ключи, образованные полями «Название издательства». В БД должно быть представление, построенное на основе запроса, содержащего «Название книги», «Название издательства», «Владелец издательства».

### *Вариант 2*

Создать БД, состоящую из следующих трех таблиц с перечисленными полями.

Таблица 1 Автомобили: Регистрационный номер, Название марки и модели; Год выпуска; Цвет кузова.

Первичным ключом таблицы является поле «Регистрационный номер». Внешним ключом является поле «Название марки и модели».

Таблица 2 Владельцы: Регистрационный номер; ФИО; Год рождения; Город проживания; Адрес.

Первичным ключом таблицы является конкатенация полей «Регистрационный номер» и «ФИО». Внешним ключом таблицы является поле «Регистрационный номер»

Таблица 1 связана с Таблицей 2 через ключи, образованные полями «Регистрационный номер».

Таблица 3 Марки и модели: Название марки и модели; Мощность двигателя; Тип коробки передач; Тип кузова.

Первичным ключом таблицы является поле «Название марки и модели».

Таблица 3 связана с Таблицей 1 через ключи, образованные полями «Название марки и модели». В БД должно быть представление, построенное на основе запроса, содержащего «Регистрационный номер», «ФИО», «Город проживания», «Адрес».

### *Вариант 3*

Создать базу данных, состоящую из следующих трех таблиц с перечисленными полями.

Таблица 1 Песни: Название песни; Язык песни; Музыкальный стиль; Время звучания.

Первичным ключом таблицы является поле «Название песни».

Таблица 2 Авторы: Псевдоним; ФИО (должно быть обязательно заполнено); Дата рождения; Город проживания. Адрес.

Первичным ключом таблицы является поле «Псевдоним».

Таблица 3 Исполнители: Псевдоним; ФИО (должно быть обязательно заполнено); Дата рождения; Город проживания; Адрес; Характеристика голоса.

Первичным ключом таблицы является поле «Псевдоним».

Таблица 4 Связи: Название песни; Псевдоним автора; Псевдоним исполнителя;

Первичным ключом таблицы является конкатенация полей «Название песни», «Псевдоним автора», «Псевдоним исполнителя». Вторичными ключами являются поля «Название песни», «Псевдоним автора», «Псевдоним исполнителя».

Таблица 4 связана с Таблицами 1, 2 и 3 через ключи, образованные полями «Название песни», «Псевдоним автора», «Псевдоним исполнителя» с соответствующими полями Таблиц 1, 2 и 3 В БД должно быть представление, построенное на основе запроса, содержащего «Название песни», «ФИО» (исполнителя).

### *Вариант 4*



Создать базу данных, состоящую из следующих трех таблиц с перечисленными полями.

Таблица 1 Предметы: Название предмета; Количество тем; Количество часов; Количество контрольных.

Первичным ключом таблицы является поле «Название предмета».

Таблица 2 Учителя: Номер паспорта; ФИО (должно быть обязательно заполнено); Дата рождения; Город проживания; Адрес.

Первичным ключом таблицы является поле «Номер паспорта».

Таблица 3 Ученики: Регистрационный код; ФИО (должно быть обязательно заполнено); Дата рождения; Город проживания; Адрес.

Первичным ключом таблицы является поле «Регистрационный код».

Таблица 4 Изучаемые предметы: Название предмета; Регистрационный код.

Первичным ключом таблицы является конкатенация обеих полей таблицы. Вторичными ключами являются поля «Название предмета», «Регистрационный код».

Таблица 4 связана с Таблицами 1 и 2 через ключи, образованные полями «Название предмета» и «Регистрационный код» с соответствующими полями Таблиц 1 и 3.

Таблица 5 Преподаватели предметов: Название предмета; Номер паспорта. Первичным ключом таблицы является конкатенация обеих полей таблицы.

Вторичными ключами являются поля «Название предмета», «Номер паспорта».

Таблица 5 связана с Таблицами 1 и 3 через ключи, образованные полями «Название предмета» и «Номер паспорта» с соответствующими полями Таблиц 1 и 2 В БД должно быть представление, построенное на основе запроса, содержащего «Название предмета», «ФИО» (учителя).

## ПРАКТИЧЕСКАЯ РАБОТА № 7

**Тема: Модификация содержимого БД. Добавление, удаление и обновление данных.**

**Цель работы:** научить добавлять данные к созданным таблицам с помощью оператора языка SQL – INSERT INTO; обновлять и удалять данные и таблицы.

**Оборудование:** ПК, интернет, программное обеспечение – MS Access, инструкции по выполнению работы.

**Справочный материал:**

Пример 1. Добавление целых строк

Оператор **INSERT** используется для вставки (добавления) строк в таблицу базы данных. Добавление можно осуществить несколькими способами:

- добавить одну полную строку
- добавить часть строки
- добавить результаты запроса.

Итак, чтобы добавить новую строку в таблицу, нам необходимо указать название таблицы, перечислить названия колонок и указать значение для каждой колонки с помощью конструкции **INSERT INTO название\_таблицы (поле1, поле2 ... ) VALUES (значение1, значение2 ...)**. Рассмотрим на примере.  
**INSERT INTO Sellers (ID, Address, City, Seller\_name, Country) VALUES ('6', '1st Street', 'Los Angeles', 'Harry Monroe', 'USA')**

ID	Address	City	Seller_name	Country
1	500 Park Street	Montreal	Michelle Green	Canada
2	1000 5th Avenue	San Francisco	Kim Howard	USA
3	42 Galaxy Road	New York	John Smith	USA
4	123 Main Street	Toronto	Denise L. Stephens	Canada
5	4th Avenue	Ottawa	Semuel Piter	Canada
6	1st Street	Los Angeles	Harry Monroe	USA

Также можно изменять порядок указания названий колонок, однако одновременно нужно менять и порядок значений в параметре **VALUES**.

Пример 2. Добавление части строк

В предыдущем примере при использовании оператора **INSERT** мы явно отмечали имена столбцов таблицы. Используя данный синтаксис, мы можем пропустить некоторые столбцы. Это значит, что вы вводите значение для одних столбцов но не предлагаете их для других. Например:

**INSERT INTO Sellers (ID, City, Seller\_name) VALUES ('6', 'Los Angeles', 'Harry Monroe')**

ID	Address	City	Seller_name	Country
1	500 Park Street	Montreal	Michelle Green	Canada
2	1000 5th Avenue	San Francisco	Kim Howard	USA
3	42 Galaxy Road	New York	John Smith	USA
4	123 Main Street	Toronto	Denise L. Stephens	Canada
5	4th Avenue	Ottawa	Semuel Piter	Canada
6		Los Angeles	Harry Monroe	

В данном примере мы не указали значение для двух столбцов **Address** и **Country**. Вы можете исключать некоторые столбцы из

оператора **INSERT INTO**, если это позволяет производить определение таблицы. В этом случае должно соблюдаться одно из условий: этот столбец определен как допускающий значение **NULL**(отсутствие какого-либо значения) или в определение таблицы указанное значение по умолчанию. Это означает, что, если не указано никакое значение, будет использовано значение по умолчанию. Если вы пропускаете столбец таблицы, которая не допускает появления в своих строках значений **NULL** и не имеет значения, определенного для использования по умолчанию, СУБД выдаст сообщение об ошибке, и эта строка не будет добавлена.

### Пример 3. Добавление отобранных данных

В предыдущей примерах мы вставляли данные в таблицы, прописывая их вручную в запросе. Однако оператор **INSERT INTO** позволяет автоматизировать этот процесс, если мы хотим вставлять данные из другой таблицы. Для этого в SQL существует такая конструкция как **INSERT INTO ... SELECT ...**. Данная конструкция позволяет одновременно выбирать данные из одной таблицы, и вставить их в другую. Предположим мы имеем еще одну таблицу **Sellers\_EU** с перечнем продавцов нашего товара в Европе и нам нужно их добавить в общую таблицу **Sellers**. Структура этих таблиц одинакова (то же количество колонок и те же их названия), однако другие данные. Для этого мы можем прописать следующий запрос:

```
INSERT INTO Sellers (ID, Address, City, Seller_name, Country) SELECT ID, Address, City, Seller_name, Country FROM Sellers_EU
```

Нужно обратить внимание, чтобы значение внутренних ключей не повторялись (поле **ID**), в противном случае произойдет ошибка. Оператор **SELECT** также может включать предложения **WHERE** для фильтрации данных. Также следует отметить, что СУБД не обращает внимания на названия колонок, которые содержатся в операторе **SELECT**, для нее важно только порядок их расположения. Поэтому данные в первом указанном столбце, что были выбраны из-за **SELECT**, будут в любом случае заполнены в первый столбец таблицы **Sellers**, указанной после оператора **INSERT INTO**, независимо от названия поля.

### Пример 4. Копирование данных из одной таблицы в другую

Часто при работе с базами данных возникает необходимость в создании копий любых таблиц, с целью резервирования или модификации. Чтобы сделать полную копию таблицы в SQL предусмотрен отдельный оператор **SELECT INTO**. Например, нам нужно создать копию таблицы **Sellers**, нужно будет прописать запрос следующим образом:

```
SELECT * INTO Sellers_new FROM Sellers
```

Усі об'єкти Access	ID	Address	City	Seller_name	Country
Таблиці	1	500 Park Street	Montreal	Michelle Green	Canada
Customers	2	1000 5th Avenue	San Francisco	Kim Howard	USA
Sellers	3	42 Galaxy Road	New York	John Smith	USA
Sellers_new	4	123 Main Street	Toronto	Denise L. Stephens	Canada
Sumproduct	5	4th Avenue	Ottawa	Semuel Piter	Canada
	6	1st Street	Los Angeles	Harry Monroe	USA

В отличие от предыдущей конструкции **INSERT INTO ... SELECT ...**, когда данные добавляются в существующую таблицу, конструкция **SELECT ... INTO ... FROM ...** копирует данные в новую таблицу. Также можно сказать, что первая конструкция импортирует данные, а вторая - экспортирует. При использовании конструкции **SELECT ... INTO ... FROM ...** следует учитывать следующее:

- можно использовать любые предложения в операторе **SELECT**, такие как **GROUP BY** и **HAVING**
- для добавления данных из нескольких таблиц можно использовать объединение
- данные возможно добавить только одну таблицу, независимо от того, из скольких таблиц они были взяты.

#### Пример 5. Обновление таблиц

Для того, чтобы изменить таблицу в SQL используется оператор **ALTER TABLE**. При использовании данного оператора необходимо ввести следующую информацию:

- имя таблицы, которую мы хотим изменить
- перечень изменений, которые мы хотим сделать.

Для примера давайте добавим новую колонку в таблицу **Sellers**, в которой будем указывать телефон реализатора:

**ALTER TABLE Sellers ADD Phone CHAR (20)**

ID	Address	City	Seller_name	Country	Phone
1	500 Park Street	Montreal	Michelle Green	Canada	
2	1000 5th Avenue	San Francisco	Kim Howard	USA	
3	42 Galaxy Road	New York	John Smith	USA	
4	123 Main Street	Toronto	Denise L. Stephens	Canada	
5	4th Avenue	Ottawa	Semuel Piter	Canada	
6	1st Street	Los Angeles	Harry Monroe	USA	

#### Пример 6. Удаление данных

Кроме добавления столбцов, мы можем их удалять. Давайте теперь удалим поле **Phone**. Для этого пропишем следующий запрос:

**ALTER TABLE Sellers DROP COLUMN Phone**

#### **Содержание работы:**

**Задание 1.** В созданные таблицы по выбранным вариантам из Практической работы №6, добавить по 5 строк.

**Задание 2** В каждой созданной таблице по выбранным вариантам из Практической работы №6 обновить данные.

**Задание 3.** Из каждой созданной таблицы по выбранным вариантам из Практической работы №6 удалить любые данные.

## ПРАКТИЧЕСКАЯ РАБОТА № 8

### Тема: Выборка данных из БД.

**Цель работы:** научить создавать сложные запросы и подзапросы, комбинированные запросы на языке SQL

**Оборудование:** ПК, интернет, программное обеспечение – MS Access, инструкции по выполнению работы.

#### Справочный материал:

##### Пример 1. Фильтрация с помощью подзапросов

Таблицы баз данных, которые используются в СУБД Access, являются реляционными таблицами, т.е. все таблицы можно связать между собой по общим полям. Допустим у нас хранятся данные в двух разных таблицах и нам нужно выбрать данные в одной из них, в зависимости от того, какие данные в другой. Для этого создадим еще одну таблицу в нашей базе данных. Это будет, например, таблица **Sellers** с информацией о поставщиках:

ID	Address	City	Seller_name	Country
1	500 Park Street	Montreal	Michelle Green	Canada
2	1000 5th Avenue	San Francisco	Kim Howard	USA
3	42 Galaxy Road	New York	John Smith	USA
4	123 Main Street	Toronto	Denise L. Stephens	Canada

Теперь мы имеем две таблицы – **Sumproduct** и **Sellers**, которые имеют одинаковое поле **City**. Предположим, нам нужно посчитать, сколько товаров было продано только в Канаде. Сделать это нам помогут подзапросы. Итак, сначала напишем запрос для выборки городов, которые находятся в Канаде:

**SELECT City FROM Sellers WHERE Country = 'Canada'**

City
Montreal
Toronto

Теперь передадим эти данные в следующий запрос, который будет выбирать данные из таблицы **Sumproduct**:

**SELECT SUM(Quantity) AS Qty\_Canada FROM Sumproduct WHERE City IN ('Montreal','Toronto')**

Qty_Canada
10222

Также мы можем объединить эти два запроса в один. Таким образом, один запрос, который выводит данные, будет главным, а второй запрос, который передает входные данные, будет вспомогательным (подзапросом). Для вложения подзапроса используем конструкцию **WHERE ... IN (...)**:

**SELECT SUM(Quantity) AS Qty\_Canada FROM Sumproduct WHERE City IN (SELECT City FROM Sellers WHERE Country = 'Canada')**

Qty_Canada
10222

Видим, что мы получили аналогичные данные, как и с помощью двух отдельных запросов. Таким же образом, мы можем увеличивать глубину вложенности запросов, вкладывая подзапросы сколько угодно раз.

### Пример 2. Использование подзапросов в качестве расчетных полей

Мы также можем использовать подзапросы в качестве расчетных полей. Отразим, например, количество реализованной продукции по каждому продавцу с помощью следующего запроса:

**SELECT Seller\_name, (SELECT SUM(Quantity) FROM Sumproduct WHERE Sellers.City = Sumproduct.City) AS Qty FROM Sellers**

Seller_name	Qty
Michelle Green	5129
Kim Howard	5347
John Smith	3897
Denise L. Stephens	5093

Первый оператор **SELECT** отражает два столбца - **Seller\_name** и **Qty**. Поле **Qty** является расчетным, оно формируется в результате выполнения подзапроса, который взят в круглые скобки. Этот подзапрос выполняется по одному разу для каждой записи в поле **Seller\_name** и в общем будет выполнен четыре раза, поскольку выбрано имена четырех продавцов.

Также в подзапросе, предложение **WHERE** выполняет функцию объединения, поскольку с помощью **WHERE** мы соединили две таблицы по полю **City**, используя полные названия столбцов (*Таблица.Поле*).

#### **Содержание работы:**

**Задание 1.** Создать по 5 запросов к созданным таблицам из Практической работы № 6



## ПРАКТИЧЕСКАЯ РАБОТА № 9

### Тема: Символы подстановки и регулярные выражения (LIKE)

**Цель работы:** научить применять оператор Like для поиска и отбора данных по шаблону.

**Оборудование:** ПК, интернет, программное обеспечение – MS Access, инструкции по выполнению работы.

#### Справочный материал:

Часто, для фильтрации данных, нужно осуществлять выборку не по точному совпадению условия, а по приближенному значению. То есть когда, например, мы ищем товар, название которого соответствует определенному шаблону или содержит определенные символы или слова. Для таких целей в SQL существует оператор **LIKE**, который ищет приближенные значения. Для конструирования такого шаблона используются **метасимволы** (специальные символы для поиска части значения), а именно: "знак процента" (%) или звездочка (\*), "символ подчеркивания" (\_) или "знак вопроса" (?), "квадратные скобки" ([ ]).

#### Пример 1. Метасимвол знак процента (%) или звездочка (\*)

Из нашей таблицы, например, отберем записи, относящиеся только к товарам, содержащих в своем названии слово **Skis** (лыжи). Для этого составим соответствующий шаблон:

**SELECT \* FROM Sumproduct WHERE Product LIKE '\*Skis\*'**

ID	Month	Product	City	Quantity	Amount
10	April	Skis Long	Montreal	854	\$209 230,00
11	April	Skis Long	Toronto	25	\$6 125,00
12	April	Skis Long	San Francisco	663	\$162 435,00
13	April	Skis Long	New York	21	\$5 145,00
14	April	Skis Short	Montreal	21	\$4 389,00
15	April	Skis Short	Toronto	4	\$836,00
16	April	Skis Short	San Francisco	522	\$109 098,00
17	April	Skis Short	New York	136	\$28 424,00
30	February	Skis Long	Montreal	522	\$127 890,00
31	February	Skis Long	Toronto	125	\$30 625,00
32	February	Skis Long	San Francisco	663	\$162 435,00
33	February	Skis Long	New York	21	\$5 145,00

Как видим, СУБД отобрала только те записи, где в колонке **Product** были товары, содержащие слово **Skis**. Также отметим, что в данном примере используется метасимвол "звездочка" (\*), поскольку СУБД Access не поддерживает "знак процента" (%) для оператора **LIKE**.

#### Пример 2. Метасимвол знак подчеркивания (\_) или знак (?)

Знак подчеркивания или вопросительный знак применяется для того, чтобы заменить один символ в слове. Давайте в слове **Bikes** заменим все гласные буквы на "вопросительный знак" (?) и посмотрим на результат:

**SELECT \* FROM Sumproduct WHERE Product LIKE 'B?k?s'**

ID	Month	Product	City	Quantity	Amount
2	April	Bikes	Montreal	12	\$4 500,00
3	April	Bikes	Montreal	56	\$21 000,00
4	April	Bikes	San Francisco	854	\$320 250,00
5	April	Bikes	New York	25	\$9 375,00
22	February	Bikes	Montreal	663	\$248 625,00
23	February	Bikes	San Francisco	21	\$7 875,00
24	February	Bikes	San Francisco	54	\$20 250,00
25	February	Bikes	New York	658	\$246 750,00
42	January	Bikes	Montreal	75	\$28 125,00

Мы использовали метасимвол "вопросительный знак" (?), поскольку СУБД Access не поддерживает "знак подчеркивания" (\_) для оператора LIKE.

Пример 3. Метасимвол квадратные скобки ([ ])

Метасимвол "квадратные скобки" ([ ]) используется для одновременного указания набора символов, по которым нужно выполнить поиск.

**SELECT \* FROM Sumproduct WHERE City LIKE '[TN]\*'**

ID	Month	Product	City	Quantity	Amount
5	April	Bikes	New York	25	\$9 375,00
7	April	Skates	Toronto	854	\$84 546,00
9	April	Skates	New York	663	\$65 637,00
11	April	Skis Long	Toronto	25	\$6 125,00
13	April	Skis Long	New York	21	\$5 145,00
15	April	Skis Short	Toronto	4	\$836,00
17	April	Skis Short	New York	136	\$28 424,00
19	April	Snow Board	Toronto	522	\$160 776,00
21	April	Snow Board	New York	663	\$204 204,00

В примере выше, мы отобрали записи, где в поле **City** названия городов начинаются с буквы **T** или **N**. Также, в данном случае, мы можем использовать еще один метасимвол, который выполняет обратное действие. Добавим в наше регулярное выражение восклицательный знак (!), что будет означать "не равно" (для СУБД Access) или знак степени (^) (для других СУБД).

**SELECT \* FROM Sumproduct WHERE City LIKE '[!TN]\*'**

ID	Month	Product	City	Quantity	Amount
2	April	Bikes	Montreal	12	\$4 500,00
3	April	Bikes	Montreal	56	\$21 000,00
4	April	Bikes	San Francisco	854	\$320 250,00
6	April	Skates	Montreal	56	\$5 544,00
8	April	Skates	San Francisco	25	\$2 475,00
10	April	Skis Long	Montreal	854	\$209 230,00
12	April	Skis Long	San Francisco	663	\$162 435,00
14	April	Skis Short	Montreal	21	\$4 389,00
16	April	Skis Short	San Francisco	522	\$109 098,00

То есть, последний созданный нами запрос будет читаться как: выбрать все колонки из таблицы **Sumproduct** и только те записи, где в поле **City** названия городов не начинаются на буквы **T** или **N**. Дополнительно



отметим, что набор букв в метасимволе "квадратные скобки" отвечает только за одну позицию в тексте.

Мы можем получить аналогичный результат, если воспользоваться уже известным нам оператором **NOT**, однако с восклицательным знаком (!) запись будет короче.

## Содержание работы:

**Задание 1.** Применить метасимволы ("знак процента" (%)) или звездочка (\*), "символ подчеркивания" ( \_ ) или "знак вопроса" (?), "квадратные скобки" ( [ ]) к созданным таблицам по выбранным вариантам заданий из Практической работы № 6.

## ПРАКТИЧЕСКАЯ РАБОТА № 10

### Тема: Вычисляемые поля

**Цель работы:** научить применять операторы языка SQL для создания вычисляемых полей.

**Оборудование:** ПК, интернет, программное обеспечение – MS Access, инструкции по выполнению работы.

#### Справочный материал:

##### Пример 1. Выполнение математических операций

Одним из способов использования расчетных полей является выполнение математических операций над выбранными данными. Давайте на примере рассмотрим как это происходит, используя снова нашу таблицу **Sumproduct**. Предположим, нам нужно вычислить среднюю цену приобретения каждого товара. Для этого нужно переделить колонку **Amount** (сумма) на **Quantity** (количество):

**SELECT DISTINCT Product, Amount/Quantity FROM Sumproduct**

Product	Expr1001
Bikes	375
Skates	99
Skis Long	245
Skis Short	209
Snow Board	308

Как видим, СУБД отобразила все наименования товаров и отобразила их среднюю стоимость в отдельном столбце, который был создан во время выполнения запроса. Также можно заметить, что мы использовали дополнительный оператор **DISTINCT**, который нам нужен для отображения уникальных названий товаров (без него мы бы получили дублирование записей).

##### Пример 2. Использование псевдонимов

В предыдущем примере мы рассчитывали среднюю стоимость покупки каждого товара и отображали значение в расчетном столбце. Однако в дальнейшем, нам неудобно обращаться к этому полю, так как его название является неинформативным для нас (СУБД дала название полю - **Expr1001**). Однако мы можем назвать поле самостоятельно, заранее указав его название в запросе, то есть дать псевдоним. Давайте перепишем предыдущий пример и укажем псевдонима для расчетного поля:

**SELECT** **DISTINCT Product,**  
**Amount/Quantity AS AvgPrice FROM Sumproduct**

Product	AvgPrice
Bikes	375
Skates	99
Skis Long	245
Skis Short	209
Snow Board	308

Видим, наше расчетное поле получило собственное название **AvgPrice**. Для этого мы использовали оператор **AS**, после которого указали необходимое

нам название. Стоит отметить, что в SQL поддерживаются только основные математические операции: сложение (+), вычитание (-), умножение (\*), деление (/). Также для изменения очередности выполнения операции можно использовать круглые скобки.

Часто псевдонимы используют не только чтобы называть расчетные поля, но и для переименования действующих. Это может быть необходимым, если действующее поле имеет длинное название или название не достаточно информативным.

### Пример 3. Соединение полей (конкатенация)

Кроме математических операций мы можем объединять текст и выводить его в отдельном поле. Давайте рассмотрим, каким образом можно осуществить склеивание (конкатенацию) текста. Имеем такой пример:

**SELECT Month + ' ' + Product AS NewField, Quantity FROM Sumproduct**

NewField	Quantity
April Bikes	12
April Bikes	56
April Bikes	854
April Bikes	25
April Skates	56
April Skates	854
April Skates	25
April Skates	663
April Skis Long	854
April Skis Long	25
April Skis Long	663
April Skis Long	21

В этом примере мы соединили значение в двух столбцах и вывели результат в новое поле **NewField**.

### **Содержание работы:**

**Задание 1.** В созданных таблицах из Практической работы № 6 применить расчетные поля (если в таблицах нет числовых значений, то добавьте в них такие поля)

## ПРАКТИЧЕСКАЯ РАБОТА № 11

### Тема: Проведение сортировки и фильтрации данных. Поиск данных по одному и нескольким полям. Поиск данных в таблице. Группировка данных (GROUP BY)

**Цель работы:** изучить операторы языка SQL для сортировки, фильтрации и группировки данных.

**Оборудование:** ПК, интернет, программное обеспечение – MS Access, инструкции по выполнению работы.

**Справочный материал:**

Пример 1. Сортировка выбранных данных.

Давайте всю нашу таблицу посортируем по сумме реализации продукции, а именно по столбцу **Amount**.

**SELECT \* FROM Sumproduct ORDER BY Amount**

ID	Month	Product	City	Quantity	Amount
47	January	Skates	New York	4	\$396,00
15	April	Skis Short	Toronto	4	\$836,00
35	February	Skis Short	Toronto	4	\$836,00
74	March	Skis Short	Montreal	4	\$836,00
52	January	Skis Long	San Francisco	4	\$980,00
41	February	Snow Board	New York	4	\$1 232,00
18	April	Snow Board	Montreal	4	\$1 232,00
79	March	Snow Board	Toronto	4	\$1 232,00
62	March	Bikes	Montreal	4	\$1 500,00
28	February	Skates	San Francisco	21	\$2 079,00
26	February	Skates	Montreal	21	\$2 079,00
46	January	Skates	Montreal	21	\$2 079,00
8	April	Skates	San Francisco	25	\$2 475,00

Видим, что запрос посортировал записи по возрастанию в поле **Amount**. Обязательно нужно соблюдать последовательность расположения операторов, т.е. оператор **ORDER BY** должен идти в самом конце запроса. В противном случае будет получено сообщение об ошибке.

Пример 2. Сортировка по нескольким полям.

Теперь посортируем наш пример дополнительно за еще одним полем. Пусть это будет поле **City**, которое отображает место реализации продукции.

**SELECT \* FROM Sumproduct ORDER BY Amount, City**

ID	Month	Product	City	Quantity	Amount
47	January	Skates	New York	4	\$396,00
74	March	Skis Short	Montreal	4	\$836,00
35	February	Skis Short	Toronto	4	\$836,00
15	April	Skis Short	Toronto	4	\$836,00
52	January	Skis Long	San Francisco	4	\$980,00
18	April	Snow Board	Montreal	4	\$1 232,00
41	February	Snow Board	New York	4	\$1 232,00
79	March	Snow Board	Toronto	4	\$1 232,00
62	March	Bikes	Montreal	4	\$1 500,00
26	February	Skates	Montreal	21	\$2 079,00
46	January	Skates	Montreal	21	\$2 079,00
28	February	Skates	San Francisco	21	\$2 079,00

Очередность сортировки будет зависеть от порядка расположения полей в запросе. То есть, в нашем случае сначала данные будут рассортированы по колонке **Amount**, а затем по **City**.

Пример 3. Направление сортировки.

Несмотря на то, что по умолчанию оператор **ORDER BY** сортирует по возрастанию, мы можем также прописать сортировки значений по убыванию. Для этого в конце каждого поля проставляем оператор **DESC** (что является сокращением от слова **DESCENDING**).

**SELECT \* FROM Sumproduct ORDER BY Amount DESC, City**

ID	Month	Product	City	Quantity	Amount
4	April	Bikes	San Francisco	854	\$320 250,00
22	February	Bikes	Montreal	663	\$248 625,00
25	February	Bikes	New York	658	\$246 750,00
70	March	Skis Long	Montreal	854	\$209 230,00
10	April	Skis Long	Montreal	854	\$209 230,00
21	April	Snow Board	New York	663	\$204 204,00
59	January	Snow Board	Toronto	663	\$204 204,00
63	March	Bikes	Montreal	522	\$195 750,00
12	April	Skis Long	San Francisco	663	\$162 435,00
32	February	Skis Long	San Francisco	663	\$162 435,00
71	March	Skis Long	Toronto	663	\$162 435,00
58	January	Snow Board	Montreal	522	\$160 776,00
80	March	Snow Board	San Francisco	522	\$160 776,00

В данном примере, значение в поле **Amount** были посортированы по убыванию, а в поле **City** - по возрастанию. Оператор **DESC** применяется только для одного столбца, поэтому при необходимости его нужно прописывать после каждого поля, которое принимает участие в сортировке.

Пример 4. Простое фильтрование оператором WHERE.

Давайте из нашей таблицы, например, отберем записи, относящиеся только к определенному товару. Для этого мы укажем дополнительный параметр отбора, который будет фильтровать значение по колонке **Product**.

**SELECT \* FROM Sumproduct WHERE Product = 'Bikes'**

ID	Month	Product	City	Quantity	Amount
2	April	Bikes	Montreal	12	\$4 500,00
3	April	Bikes	Montreal	56	\$21 000,00
4	April	Bikes	San Francisco	854	\$320 250,00
5	April	Bikes	New York	25	\$9 375,00
22	February	Bikes	Montreal	663	\$248 625,00
23	February	Bikes	San Francisco	21	\$7 875,00
24	February	Bikes	San Francisco	54	\$20 250,00
25	February	Bikes	New York	658	\$246 750,00
42	January	Bikes	Montreal	75	\$28 125,00
43	January	Bikes	Toronto	12	\$4 500,00
44	January	Bikes	San Francisco	136	\$51 000,00
45	January	Bikes	New York	21	\$7 875,00
62	March	Bikes	Montreal	4	\$1 500,00
63	March	Bikes	Montreal	522	\$195 750,00
64	March	Bikes	San Francisco	125	\$46 875,00
65	March	Bikes	New York	212	\$79 500,00

Как видим, условие отбора взято в одинарные кавычки, что является обязательным при фильтровании текстовых значений. При фильтровании числовых значений кавычки не нужны.

**Пример запроса для отбора числовых значений:**

**SELECT \* FROM Sumproduct WHERE Amount > 40000 ORDER**

**BY Amount**

ID	Month	Product	City	Quantity	Amount
39	February	Snow Board	Toronto	136	\$41 888,00
61	January	Snow Board	New York	136	\$41 888,00
64	March	Bikes	San Francisco	125	\$46 875,00
44	January	Bikes	San Francisco	136	\$51 000,00
48	January	Skates	San Francisco	522	\$51 678,00
9	April	Skates	New York	663	\$65 637,00
27	February	Skates	Toronto	663	\$65 637,00
65	March	Bikes	New York	212	\$79 500,00
7	April	Skates	Toronto	854	\$84 546,00
67	March	Skates	Toronto	854	\$84 546,00
36	February	Skis Short	San Francisco	522	\$109 098,00
16	April	Skis Short	San Francisco	522	\$109 098,00

В этом примере мы отобрали записи, в которых выручка от реализации составила более **40 тыс. \$** и, дополнительно, все записи посортировали по возрастанию по полю **Amount**.

**Пример 5.** Фильтрация по диапазону значений (BETWEEN).

Для отбора данных, которые лежат в определенном диапазоне, используется оператор **BETWEEN**. В следующем запросе будут отобраны все значения, лежащие в пределах от **1000 \$** в **2000 \$** включительно, в поле **Amount**.

**SELECT \* FROM Sumproduct WHERE Amount BETWEEN 1000 AN**

**D 2000**

ID	Month	Product	City	Quantity	Amount
18	April	Snow Board	Montreal	4	\$1 232,00
41	February	Snow Board	New York	4	\$1 232,00
62	March	Bikes	Montreal	4	\$1 500,00
79	March	Snow Board	Toronto	4	\$1 232,00

Очередность сортировки будет зависеть от порядка расположения полей в запросе. То есть, в нашем случае сначала данные будут посортированы по колонке **Amount**, а затем по **City**.

**Пример 6.** Выборка пустых записей (IS NULL).

В **SQL** существует специальный оператор для выборки пустых записей (называется **NULL**). Пустой записью считается любая ячейка в таблице, в которую не введены какие-либо символы. Если в ячейку введен **0** или **пробел**, то считается, что поле заполнено.

**SELECT \* FROM Sumproduct WHERE Amount IS NULL**

ID	Month	Product	City	Quantity	Amount
5	April	Bikes	New York	25	
19	April	Snow Board	Toronto	522	

В примере выше, мы нарочно удалили два значения в поле **Amount**, чтобы продемонстрировать работу оператора **NULL**.

### Пример 7. Расширенное фильтрации (AND, OR).

Язык **SQL** не ограничивается фильтрацией по одному условию, для собственных целей вы можете использовать достаточно сложные конструкции для выборки данных одновременно по многим критериям. Для этого в **SQL** есть дополнительные операторы, которые расширяют возможности оператора **WHERE**. Таковыми операторами являются: **AND**, **OR**, **IN**, **NOT**. Приведем несколько примеров работы данных операторов.

**SELECT \* FROM Sumproduct WHERE Amount > 40000 AND City = 'Toronto'**

ID	Month	Product	City	Quantity	Amount
7	April	Skates	Toronto	854	\$84 546,00
19	April	Snow Board	Toronto	522	\$160 776,00
27	February	Skates	Toronto	663	\$65 637,00
39	February	Snow Board	Toronto	136	\$41 888,00
59	January	Snow Board	Toronto	663	\$204 204,00
67	March	Skates	Toronto	854	\$84 546,00
71	March	Skis Long	Toronto	663	\$162 435,00
75	March	Skis Short	Toronto	522	\$109 098,00

**SELECT \* FROM Sumproduct WHERE Month= 'April' OR Month= 'March'**

ID	Month	Product	City	Quantity	Amount
15	April	Skis Short	Toronto	4	\$836,00
74	March	Skis Short	Montreal	4	\$836,00
18	April	Snow Board	Montreal	4	\$1 232,00
79	March	Snow Board	Toronto	4	\$1 232,00
62	March	Bikes	Montreal	4	\$1 500,00
8	April	Skates	San Francisco	25	\$2 475,00
77	March	Skis Short	San Francisco	21	\$4 389,00
14	April	Skis Short	Montreal	21	\$4 389,00
2	April	Bikes	Montreal	12	\$4 500,00
13	April	Skis Long	New York	21	\$5 145,00
72	March	Skis Long	San Francisco	21	\$5 145,00

Давайте объединим операторы **AND** и **OR**. Для этого сделаем выборку велосипедов (**Bikes**) и коньков (**Skates**), которые были проданы в марте (**March**).

**SELECT \* FROM Sumproduct WHERE Product = 'Bikes' OR Product = 'Skates' AND Month= 'March'**



ID	Month	Product	City	Quantity	Amount
2	April	Bikes	Montreal	12	\$4 500,00
3	April	Bikes	Montreal	56	\$21 000,00
4	April	Bikes	San Francisco	854	\$320 250,00
5	April	Bikes	New York	25	\$9 375,00
22	February	Bikes	Montreal	663	\$248 625,00
23	February	Bikes	San Francisco	21	\$7 875,00
24	February	Bikes	San Francisco	54	\$20 250,00
25	February	Bikes	New York	658	\$246 750,00
42	January	Bikes	Montreal	75	\$28 125,00
43	January	Bikes	Toronto	12	\$4 500,00
44	January	Bikes	San Francisco	136	\$51 000,00
45	January	Bikes	New York	21	\$7 875,00
62	March	Bikes	Montreal	4	\$1 500,00
63	March	Bikes	Montreal	522	\$195 750,00
64	March	Bikes	San Francisco	125	\$46 875,00
65	March	Bikes	New York	212	\$79 500,00
66	March	Skates	Montreal	56	\$5 544,00
67	March	Skates	Toronto	854	\$84 546,00
68	March	Skates	San Francisco	212	\$20 988,00
69	March	Skates	New York	56	\$5 544,00

Видим, что в нашу выборку попало за много значений (кроме марта (**March**), также январь (**January**), февраль (**February**) и апрель (**April**)). Оператор **AND** имеет более высокий приоритет, чем оператор **OR**, поэтому сначала были отобраны записи с коньками, которые проданные в марте, а потом все записи, касающиеся велосипедов.

Итак, чтобы получить правильную выборку, нам нужно изменить приоритеты выполнения команд. Для этого используем **скобки**, как в математике. Тогда, сначала будут обработаны операторы в скобках, а затем - все остальные.

**SELECT \* FROM Sumproduct WHERE (Product = 'Bikes' OR Product = 'Skates') AND Month= 'March'**

ID	Month	Product	City	Quantity	Amount
62	March	Bikes	Montreal	4	\$1 500,00
63	March	Bikes	Montreal	522	\$195 750,00
64	March	Bikes	San Francisco	125	\$46 875,00
65	March	Bikes	New York	212	\$79 500,00
66	March	Skates	Montreal	56	\$5 544,00
67	March	Skates	Toronto	854	\$84 546,00
68	March	Skates	San Francisco	212	\$20 988,00
69	March	Skates	New York	56	\$5 544,00

Пример 8. Расширенная фильтрация (оператор IN).

**SELECT \* FROM Sumproduct WHERE ID IN (4, 12, 58, 67)**

ID	Month	Product	City	Quantity	Amount
4	April	Bikes	San Francisco	854	\$320 250,00
12	April	Skis Long	San Francisco	663	\$162 435,00
58	January	Snow Board	Montreal	522	\$160 776,00
67	March	Skates	Toronto	854	\$84 546,00

Оператор **IN** выполняет ту же функцию, что и **OR**, однако имеет ряд преимуществ:

- При работе с длинными списками, предложение с **IN** легче читать;



- Используется меньшее количество операторов, что ускоряет обработку запроса;
- Самое важное преимущество **IN** в том, что в его конструкции можно использовать дополнительную конструкцию **SELECT**, что открывает большие возможности для создания сложных подзапросов.

Пример 9. Расширенная фильтрация (оператор NOT).

**SELECT \* FROM Sumproduct WHERE NOT City IN ('Toronto', 'Montreal')**

ID	Month	Product	City	Quantity	Amount
47	January	Skates	New York	4	\$396,00
52	January	Skis Long	San Francisco	4	\$980,00
41	February	Snow Board	New York	4	\$1 232,00
28	February	Skates	San Francisco	21	\$2 079,00
8	April	Skates	San Francisco	25	\$2 475,00
77	March	Skis Short	San Francisco	21	\$4 389,00
57	January	Skis Short	New York	21	\$4 389,00
13	April	Skis Long	New York	21	\$5 145,00
72	March	Skis Long	San Francisco	21	\$5 145,00
33	February	Skis Long	New York	21	\$5 145,00
69	March	Skates	New York	56	\$5 544,00
40	February	Snow Board	San Francisco	21	\$6 468,00

Ключевое слово **NOT** позволяет убрать ненужные значения из выборки. Также его особенностью является то, что оно проставляется перед названием столбца, участвующего в фильтровании, а не после.

Группировка данных позволяет разделить все данные на логические наборы, благодаря чему становится возможным выполнение статистических вычислений отдельно в каждой группе.

Пример 10. Создание групп (GROUP BY)

Группы создаются с помощью предложения **GROUP BY** оператора **SELECT**. Рассмотрим на примере.

**SELECT Product, SUM(Quantity) AS Product\_num FROM Sumproduct GROUP BY Product**

Product	Product_num
Bikes	3450
Skates	4376
Skis Long	5251
Skis Short	2879
Snow Board	3510

Данным запросом мы извлекли информацию о количестве реализованной продукции в каждом месяце. Оператор **SELECT** приказывает вывести два столбца **Product** - название продукта и **Product\_num** - расчетное поле, которое мы создали для отображения количества реализованной продукции (формула поля **SUM (Quantity)**). Предложение **GROUP BY** указывает СУБД сгруппировать данные по столбцу **Product**. Стоит также отметить, что **GROUP BY** должен идти после предложения **WHERE** и перед **ORDER BY**.

Пример 11. Фильтрующие группы (HAVING)

Так же, как мы фильтровали строки в таблице, мы можем осуществлять фильтрацию по сгруппированным данным. Для этого в **SQL** существует

оператор **HAVING**. Возьмем предыдущий пример и добавим фильтрацию по группам.

```
SELECT Product,  
SUM(Quantity) AS Product_num FROM Sumproduct GROUP  
BY Product HAVING SUM(Quantity)>4000
```

Product	Product_num
Skates	4376
Skis Long	5251

Видим, что после того, как была посчитана количество реализованного товара в разрезе каждого продукта, СУБД "отсекла" те продукты, которых было реализовано меньше 4000 шт.

Как видим, оператор **HAVING** очень похож на оператора **WHERE**, однако между собой они имеют существенное отличие: **WHERE** фильтрует данные до того, как они будут сгруппированы, а **HAVING** - осуществляет фильтрацию после группировки. Таким образом, строки, которые были изъяты предложением **WHERE** не будут включены в группу. Итак, операторы **WHERE** и **HAVING** могут использоваться в одном предложении.

Рассмотрим пример:

```
SELECT Product,  
SUM(Quantity) AS Product_num FROM Sumproduct WHERE Product<>'Skis  
Long' GROUP BY Product HAVING SUM(Quantity)>4000
```

Product	Product_num
Skates	4376

Мы к предыдущему примеру добавили оператор **WHERE**, где указали товар **Skis Long**, что в свою очередь повлияло на группирование оператором **HAVING**. Как результат видим, что товар **Skis Long** не попал в перечень групп с количеством реализованной продукции больше 4000 шт.

### Пример 12. Группировка и сортировка

Как и при обычной выборке данных, мы можем сортировать группы после группировки оператором **HAVING**. Для этого мы можем использовать уже знакомый нам оператор **ORDER BY**. В данной ситуации его применения аналогичное предыдущим примерам. К примеру:

```
SELECT Product,  
SUM(Quantity) AS Product_num FROM Sumproduct GROUP  
BY Product HAVING SUM(Quantity)>3000 ORDER BY SUM(Quantity)
```

или просто укажем номер поля по порядку, по которому хотим сортировать:

```
SELECT Product,  
SUM(Quantity) AS Product_num FROM Sumproduct GROUP  
BY Product HAVING SUM(Quantity)>3000 ORDER BY 2
```

Product	Product_num
Bikes	3450
Snow Board	3510
Skates	4376
Skis Long	5251

Видим, что для сортировки сводных результатов нам нужно просто прописать предложения с **ORDER BY** после оператора **HAVING**. Однако есть один нюанс. СУБД **Access** не поддерживает сортировку групп по псевдонимами колонок, то есть в нашем примере, чтобы сортировать значения, мы не сможем в конце запроса прописать **ORDER BY Product\_num** .

### **Содержание работы:**

**Задание 1.** Примените сортировку к каждой таблице, созданным в Практической работе № 6.

**Задание 2.** Примените все виды фильтрации к таблицам, созданным в Практической работе № 6.

**Задание 3.** В созданных в Практической работе № 6 таблицах применить группировку по полям.

## **Информационное обеспечение обучения**

### **Печатные издания:**

#### **Основные учебные издания:**

1. Кумскова, И.А. Базы данных: учебник / Кумскова И.А. — Москва: КноРус, 2021. — 400 с. — ISBN 978-5-406-08303-1. — URL: <https://book.ru/book/940108>
2. Чулюков, В.А. Проектирование баз данных. Практический курс: учебное пособие / Чулюков В.А., Астахова И.Ф., Башарина С.О., Сидорова О.А. — Москва: Русайнс, 2020. — 163 с. — ISBN 978-5-4365-5748-9. — URL: <https://book.ru/book/938011>

#### **Дополнительные учебные издания:**

3. Грошев, А. С. Основы работы с базами данных: учебное пособие / А. С. Грошев. — 3-е изд. — Москва: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. — 255 с. — ISBN 978-5-4497-0914-1. — Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. — URL: <https://www.iprbookshop.ru/102038.html>
4. Молдованова, О. В. Информационные системы и базы данных: учебное пособие для СПО / О. В. Молдованова. — Саратов: Профобразование, 2021. — 177 с. — ISBN 978-5-4488-1177-7. — Текст: электронный // Электронный ресурс цифровой образовательной среды СПО PROФобразование: [сайт]. — URL: <https://profspo.ru/books/106617>

### **3.2.2. Интернет-ресурсы:**

#### **Электронно-библиотечная система:**

5. ЭБС «IPRbooks», ООО «Ай Пи Ар Медиа»
6. ЭБС «Znanium»
7. ЭБС «PROФобразование»
8. ЭБС «Book.ru»