Министерство общего и профессионального образования Свердловской области
государственное автономное профессиональное образовательное учреждение Свердловской области «Ирбитский мотоциклетный техникум»
(ГАПОУ СО «ИМТ»)

ПРОГРАММА ПОДГОТОВКИ СПЕЦИАЛИСТОВ СРЕДНЕГО ЗВЕНА ПО СПЕЦИАЛЬНОСТИ

ПО СПЕЦИАЛЬНОСТИ
09.02.04 ИНФОРМАЦИОННЫЕ СИСТЕМЫ (по отраслям)

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ СТУДЕНТОВ АВТОНОМНОГО УЧРЕЖДЕНИЯ ГАПОУ СО «ИМТ»

по выполнению практических работ
по дисциплине ОП.18 АДМИНИСТРИРОВАНИЕ БАЗ ДАННЫХ

Составитель: Щербаков Н.П. преподаватель ГАПОУ СО «ИМТ»,

Методические рекомендации для студентов Автономного учреждения ГАПОУ СО «ИМТ» по выполнению практических работ по дисциплине ОП.07 Основы проектирования баз данных разработаны в соответствии с рабочей программой, утвержденной директором ГАПОУ СО «ИМТ».

СОДЕРЖАНИЕ

1. Пояснительная записка

На современном этапе развития общества информация превратилась в один из наиболее важных ресурсов, а информационные системы стали необходимым инструментом практически во всех сферах деятельности. Информационные системы служат для сбора, обработки, хранения и выдачи информации. А основными задачами таких информационных систем являются:
Разработка баз данных, выполняющих все указанные функции;
Разработка пользовательского программного обеспечения для работы с базами данных.

Выполнение этих требований приводит к задаче построения единой системы баз данных, общих для информационного поля. В рамках этих задач мы изучаем разработку и администрирование баз данных как отдельный модуль. А данные методические указания преследуют цель на практических примерах привить основы владения СУБД.

Основные цели лабораторных работ:

1. Изучить методологию проектирования баз данных
2. Изучить технологию организации процессов обработки данных в базах данных
3. Изучить интерфейс СУБД реляционного типа
4. Изучить язык структурированных запросов SQL
5. Научиться управлению, в т.ч. удаленному, базами данных

2. Лабораторные работы

ПР1-2 Логические модели данных

Цель: Изучение типы логических моделей данных

Задание

1. На основании лекционного материала подготовить схемы баз данных разных логических моделей – реляционной, иерархической, сетевой. Темы баз данных:
   корабли второй мировой
   автосервис
   компьютерная фирма
   аэропорт
   регистратура поликлиники
   журнал колледжа
   склад готовой продукции фирмы вторсырья
   анкеты студентов
   магазин автозапчастей
2. В реляционной модели данных должны быть связи всех типов – один к одному, один ко многим и многие ко многим.

ПР3-4 Способы организации памяти для хранения данных

Цель: Изучение сбособов организации памяти хранения данных и типов данных в таблицах

Задание

1. Выбрать предметную область (например: склад, больница, аптека, аэропорт и т.д.) и составить для нее:
а) описание предметной области (от имени конечного пользователя);
б) ER-диаграмму.
Ограничение: от 5ти до 10ти сущностей для описания области. Каждый объект должен иметь хотя бы один атрибут. Описание и диаграмма включается в отчет по лабораторной работе. Отчет выполняется в печатном виде на листах формата А4 согласно общепринятым правилам оформления лабораторных работ.
2. Используя MS Access перенести полученную модель в БД, используя таблицы и схему данных.

ПР5-8 Разработка физической модели данных (4 часа)

Цель: научиться использовать операторы языка SQL для работы с данными БД.

Задание

1. Написать запрос для выбора автомобилей определенного цвета. Цвет задается в виде параметра в условии WHERE (например, 'белый').
2. Определить количество автомобилей, у которых номер фондового извещения начинается на "10" и не заканчивается на "39"
3. По каждой штатной группе а/м определить, сколько а/м каждой марки было выпущено в заданном году. Вывести названия групп и названия марок на экран.
4. Определить, какие а/м данного класса переданы в подразделения после указанной даты. Указать также номер автомобиля и дату документа передачи каждого а/м.
5. Произвести выборку автомобилей из двух полей «номер авто», «класс авто» (подставлять название из отношения MENU). Если поле «класс» в таблице MENU не существует, то выводить строку «Класс средства неизвестен» с помощью функции iif.
6. Определить, сколько а/м каждой марки имеют год выпуска меньший, чем округленный до целого средний год выпуска а/м заданной пользователем марки.
7. Определить какое количество а/м каждой марки в каком году было произведено (перекрестный запрос: марки а/м на год производства)

ПР9-10 Способы создания запросов (2 часа)

Цель: Развитие у студентов навыков программирования приложений, использующих БД, знакомство с частями SDL и DML языка SQL.

Задание

1. Создать базу данных по любой предметной области (желательно по курсовой работе), которая должна минимум содержать таблицу, состоящую минимум из 6 полей, где обязательно должно присутствовать поле типа date. Для создания таблиц БД использовать скриптовый файл или макрокоманду, содержащую набор SQL-команд из части языка SDL;
2. Реализовать процедуры Добавления, Удаления, Поиска и Изменения, с помощью SQL;
3. Организовать оконный интерфейс для функций, созданных на предыдущем этапе (добавления, удаления, поиска и изменения);
4. Поиск должен осуществляться с использованием индексов, т.е. поля, по которым осуществляется поиск, должны быть проиндексированы. Для создания индексов использовать CREATE INDEX.

Выполнение обязательных пунктов = 70%

Бонус (+ 15%): Для получения дополнительных баллов реализовать кодовые поля в основной таблице и справочник(и) для расшифровки этих полей (подобно базе allauto.mdb).

Бонус (+ 15%): Организовать механизм авторизации – вход в БД по паролю для нескольких пользователей (статья справки «Пароли (MDB)»).

ПР11-14 Автоматизация расчетов с помощью запросов (4 часа)

Цель: моделирования механизмов расчетов в запросах, протоколирование и отката команд.

Задание

1. Необходимо реализовать главную форму, запускаемую автоматически при открытии БД (для allauto.mdb). Эта форма должна позволять редактировать данные о а/м в таблице AUTO: добавление, удаление, изменение автомобиля (или автомобилей). При запуске приложения в нормальном режиме не выводить окно базы данных.
2. Реализовать протоколирование – журнал изменений. Должны быть реализованы функции отката изменений БД (таких как добавление, удаление, изменение записи). Для этого организовать специальную форму, позволяющую осуществлять:

• "Откат назад" – откат назад на одно изменение в базе (не активен, если не было изменений или выполнены все откаты назад);

• "Откат вперед" – откат вперед на одно изменение (может быть не активен). Не забудьте проиндексировать таблицу MENU.

Выполнение обязательных пунктов = 55%

Бонус (+ 15%): предусмотреть, что при аварийном завершении программы, существует возможность восстановления всей цепочки отката, и лишь только при "нормальном" завершении работы с программой цепочка отката обнуляется (для этого используйте макросы).

Бонус (+ 30%): Реализовать процедура поиска для главной формы. "Поиск" - содержит следующие подпункты (функции):

1. "Параметры" - выдается форма, содержащие перечень не менее 3х кодовых полей (любые, из таблицы AUTO). При

выборе любого из этих пунктов выдается перечень возможных значений, который может принять данное кодовое поле (выпадающий список значений из menu). У значений могут быть свои подзначения, поэтому необходимо организовать рекурсивную (или циклическую) процедуру обхода дерева значений (словаря данных) – т.е. для каждого поля выводить не только его значения, но и подзначения. У ВСЕХ ДОЛЖНО БЫТЬ КОДОВОЕ ПОЛЕ "МАРКА ТРАНСПОРТНОГО СРЕДСТВА".
При выборе значений, заносить их в строку поиска, типа: <код. поле>=<значение> AND
2. "Строка поиска" - выводит на экран содержимое строки поиска.
3. "Очистить строку поиска" - обнуляет строку поиска.
4. "Найти" – поиск записей в таблице auto в соответствии со строкой поиска, если строка поиска пуста, то выводятся все записи.
Все поля, которые используется для поиска, должны быть расшифрованы.

ПР15-16 Технология разработки экранных форм и вывода отчетов

Цель: научиться создавать удобные экранные формы для трёх основных видов работ в информационной системе: просмотра, ввода и редактирования, научиться пользоваться этими формами, изучить различные права доступа к данным.

Задание

Пользователям информационных систем предоставляются разные права доступа к данным. Одним разрешается только просмотр, другим – просмотр и редактирование и т. д. Для удобства работы с данными и обеспечения прав собственности разрабатывают три вида форм: для *просмотра, ввода* и *редактирования.*

В формах для просмотра запрещено изменять данные, поэтому в них не нужно отображать списки допустимых значений. Так как при просмотре желательно видеть сразу много записей, то формы этого вида открывают в виде таблиц. Форму для редактирования удобно открывать сразу на записи, в которую нужно внести изменение, и отображать в ней списки допустимых значений .

Форма для ввода в момент начала ввода новой записи содержит пустые поля. Если набор вариантов вводимых данных небольшой, то следует пользоваться списками допустимых значений.

Создание формы для просмотра. В окне базы данных выберите объект «Формы» и пункт меню «Создать». Откроется окно «Новая форма». Выберите в верхней части окна пункт меню «Конструктор» и в нижней части окна укажите таблицу «преподаватели». Нажмите «ОК». Должны появиться 3 окна: форма, список полей и панель элементов. Список полей и панель элементов можно открыть через пункт меню Access «Вид» в верхней части экрана.

Перетащите мышкой из списка полей в форму поля «ФИО», «Дата_рожд.», «Зарплата», «Биография» и «Фотография». Эти поля таблицы отобразятся в форме также в виде полей. Для поля «Фотография» будет создан элемент «Присоединённая рамка объекта».

При перетаскивании в форму поля «Должность» Access создаёт элемент «список», в котором будут показаны все допустимые значения должности. Так как при просмотре список не нужен, то для «Должности» нужно создать элемент «поле» вручную. Для этого щёлкните мышкой по элементу «поле» на панели элементов. Переместите указатель мышки в форму и раздвиньте поле до подходящих, на Ваш взгляд, размеров. Введите вместо «поле0» название поля. Теперь нужно привязать поле формы к полю «Должность» таблицы «преподаватели». Щёлкните правой кнопкой мышки по создаваемому полю в форме. В появившемся меню выберите пункт «Свойства». Появится окно свойств поля. Выберите вкладку «Данные» и в свойстве «Данные» выберите «Должность».

Аналогично создайте поля для «Степени» и «Звания».

Для ввода названия формы откройте раздел формы «Заголовок», выполнив пункты меню Access «Вид» □□ «Заголовок/примечание формы». Название формы должно выглядеть примерно так:

П Р Е П О Д А В А Т Е Л И

(форма для просмотра)

В форме, предназначенной только для просмотра, необходимо запретить пользователю вносить какие-либо изменения в данные. Щёлкните правой кнопкой мышки по квадратику в левом верхнем углу формы и в раскрывшемся меню выберите пункт «Свойства». Должно открыться окно «Форма». Выберите вкладку «Данные». Установите для свойства «Тип набора записей» значение «Статический набор».

Отредактируйте по своему вкусу названия полей и формы.

Перейдите в режим формы и проверьте её работу. Форму можно просматривать в трёх режимах:

- простая форма,
- ленточная форма,
- таблица.

Чаще всего при просмотре используется режим «Таблица». Просмотрите форму во всех трёх режимах. Режимы просмотра устанавливаются Конструктором . В режиме конструктора откройте окно свойств формы и выберите вкладку «Макет». Подберите нужные значения свойств «Режим по умолчанию» и «Допустимый режим». Режим «Ленточная форма» не может использоваться при включении в форму подчинённых форм.

Создание формы для редактирования. Воспользуйтесь помощью мастера. В окне базы данных выберите объект «Формы» и затем – «Создание формы с помощью мастера». В появившемся окне «Создание формы»

выберите таблицу «преподаватели» и все поля за исключением поля «код_преп». Следуйте далее указаниям мастера. Форму назовите «преп_ред». Если при просмотре Вам не понравится вид формы, то перейдите в режим конструктора и внесите изменения в свойства либо всей формы, либо отдельных элементов.

Для проверки работы формы внесите изменения в данные о преподавателях.

Создание формы для ввода новых записей. Форма для ввода отличается от формы для редактирования только значением свойства «Ввод данных».

Скопируйте форму «преп_ред». Для этого в окне базы данных щёлкните правой клавишей мышки по названию формы «преп_ред» и выберите «копировать», затем на любом свободном месте окна базы данных снова щёлкните правой клавишей мышки и выберите «вставить». Назовите форму «преп_ввод». Откройте форму «преп_ввод» в режиме конструктора и измените значение свойства «Вод данных» на «да». Значение «да» переводит форму в режим, в котором не видны уже имеющиеся в таблице записи, все поля формы пустые. После ввода данных очередной записи и перехода к следующей введённая запись автоматически заносится в базу и форма готова к вводу новой записи.

Удобно заполнять поля, когда они расположены друг под другом. Перетащите мышкой поля и их названия так, чтобы каждое поле со своим названием (кроме фотографии) располагалось в отдельной строке. Фотографию можно расположить справа от остальных полей.

Перейдите в режим формы и введите данные о двух-трёх преподавателях.

ПР17-18 Установка атрибутов и ключей

Цель: научится создавать форму, с помощью которой множество объектов (формы, отчёты, запросы) объединяются в единую, управляемую пользователем систему с соответствующими атрибутами и ключами.

Задание

Вся работа пользователя с информационной системой организуется через главную, как правило, кнопочную форму. Создадим форму (рис. 1), из которой будут вызываться созданные ранее формы.

Создание формы, не связанной с таблицей базы данных. Создайте форму в режиме конструктора, не указывая источник данных.

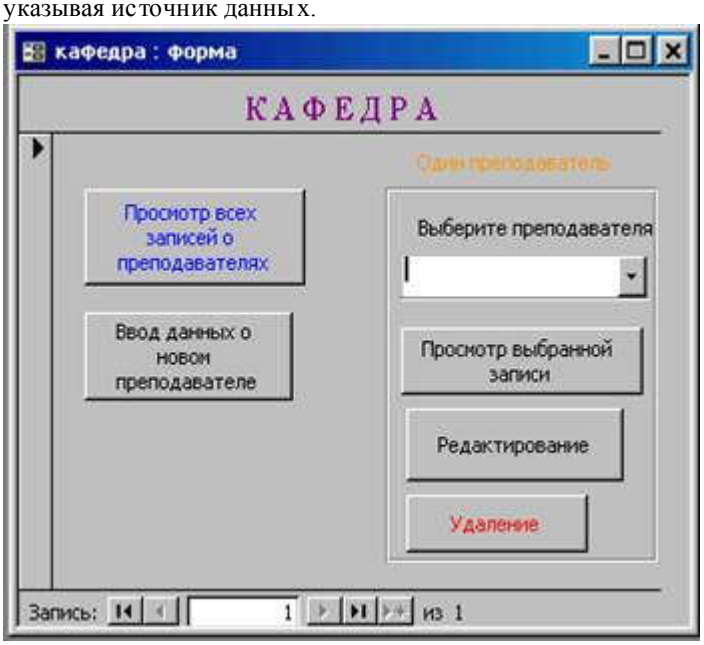


Рис. 1. Пример формы

Создание поля со списком для выбора записи. Создайте поле со списком преподавателей. Из этого списка пользователь сможет выбрать преподавателя, сведения о котором нужно просмотреть. Выберите на панели элементов поле со списком и поместите его в форму. Подтвердите, что поле со списком будет использовать значения из таблицы или запроса и щёлкните по кнопке «далее». Выберите таблицу «преподаватели» и снова щёлкните по кнопке «далее».

Для отображения в поле со списком нужно кроме поля «ФИО» выбрать ключевое поле «код_преп» и показывать в форме при просмотре списка только «ФИО». Все значения ключевого поля по определению оригинальны. Люди с одинаковыми фамилиями и инициалами иногда встречаются. Если указать для отображения в списке только поле «ФИО», то Access (и другие СУБД) всегда будет выбирать из таблицы первую по порядку из двух записей, имеющих одинаковые значения поля «ФИО». Если в строке списка есть и «код_преп» и «ФИО», то из таблицы будет выбрана запись, содержащая значение поля «код_преп».

Для быстрого выбора из длинного списка можно в поле ввести первую букву нужной фамилии.

Создание кнопки для просмотра данных обо всех преподавателях. Если нужны данные о многих преподавателях, то целесообразно открыть форму для просмотра в режиме таблицы. Создадим кнопку и связанную с ней процедуру, открывающую форму в режиме таблицы.

Выберите на панели элементов кнопку и поместите её в форму. В открывшемся окне «Создание кнопок» выберите категорию «Работа с формой» и действия «Открыть форму». Далее выберите форму «преп_просм», далее – переключатель «Открыть форму и показать все записи». далее – переключатель «Текст» и наберите в поле надпись на кнопке «Просмотр всех преподавателей». Задайте имя кнопки «откр_таб_преп». Созданная кнопка будет открывать «преп_просм» в режиме формы.

Для того чтобы форма «преп_просм» открывалась в режиме таблицы, внесите изменения в процедуру, которую Access автоматически создал вместе с кнопкой. Для этого в режиме конструктора откройте окно свойств кнопки «откр_таб_преп» и в нём вкладку «События». Щёлкните мышкой по свойству «Нажатие мышки», а затем щёлкните по квадратной кнопке с многоточием, расположенной справа. Откроется окно Visual Basic. В процедуре «Откр_таб_преп_Click()» в строчку

DoCmd.OpenForm stDocName, , , stLinkCriteria

вставьте после первой запятой слово «acFormDS» (это параметр, задающий открытие формы в режиме таблицы). Строка примет вид:

DoCmd.OpenForm stDocName, acFormDS, , stLinkCriteria

Эта строка содержит команду на открытие формы. Значения «слов» в строке следующие:

- **DoCmd** – выполнить команду**;**
- **OpenForm –** команда «открыть форму»;
- **StDocName –** переменная, содержащая имя формы.

Через запятую перечислены параметры команды.

Закройте окно Visual Basic и проверьте работу кнопки.

Создание кнопок для просмотра и редактирования данных об определённом преподавателе. Создадим кнопку, с помощью которой будет открываться форма для просмотра данных об одном преподавателе. ФИО преподавателя выбирается из поля со списком.

Выберите на панели элементов кнопку и поместите её в форму. В открывшемся окне «Создание кнопок» выберите категорию «Работа с формой» и действия «Открыть форму». Далее выберите форму «преп_просм», далее – переключатель «Открыть форму для отобранных записей».

В следующем окне нужно указать, что в открываемой форме «преп_просм» будет показана запись из таблицы «преподаватели», содержащая значение поля «код_преп» (поля из таблицы базы данных), равное выбранному из поля со списком в форме «кафедра».

Далее выберите переключатель «Текст» и наберите в поле надпись на кнопке «Просмотр одного преподавателя». Задайте имя кнопки «откр_один_преп».

Кнопка для редактирования данных о преподавателе создаётся аналогично. Отличие состоит только в названиях кнопки и открываемой формы. Назовите кнопку «Редактирование и удаление», так как в дальнейшем в форме для редактирования будет кнопка для удаления записи .

Кнопка для ввода записи создаётся аналогично двум предыдущим, только не нужно связывать поля и исправлять процедуру. Выберите переключатель «Открыть форму и показать все записи»**,** надпись на кнопке «Ввод преподавателя». Задайте имя кнопки «ввод_преп».

Кнопка для удаления записи. Чтобы уменьшить возможности ошибиться при удалении записи, необходимо:

- в момент нажатия кнопки удаления видеть удаляемую запись,
- после нажатия кнопки для удаления записи пользователь должен подтвердить приказ об удалении.

Создайте кнопку для удаления записи в форме для редактирования. Выберите категорию «Обработка записей» и действие «Удалить запись». На кнопке напишите «Удаление». Попробуйте удалить одну запись.

Внимание! При вводе новых записей номер удалённой записи не используется. В последовательности номеров образуется неустранимый разрыв.

На рис. 1 в форме «кафедра» изображена необязательная кнопка «Удаление». Для её работы используется следующая процедура:

```
Private Sub КнопкаУд_Click()
  Dim N_record As String
  Dim dbs As Database
  a = MsgBox("Удалять?", vbYesNo)
  If a = vbNo Then
    Exit Sub
  Else
    N_record = Me![ПолеСоСписком0]
    Set dbs = CurrentDb
    dbs.Execute "DELETE FROM преподаватели WHERE код_преп=" & N_record
  End If
End Sub
```

Процедурой *КнопкаУд_Click()* удаляется из таблицы преподаватели запись, код (поле «код_преп») которой равен значению кода преподавателя, выбранному из поля со списком в форме «кафедра» (Me![ПолеСоСписком0]). Следует отметить, что перед удалением записи пользователь должен видеть всё её содержимое. Поэтому лучше кнопку для удаления поместить только в форме для просмотра или в форме для редактирования.

Самостоятельно создайте кнопки для выхода из форм для просмотра, редактирования и ввода.

ПР19-20 Установка и нормализация отношений в базе данных (различные нормальные формы)

Цель: научиться устанавливать связи между таблицами и строить подчинённые формы.

Задание

Цель работы: научиться устанавливать связи между таблицами и строить подчинённые формы.

Между реальными объектами, данные о которых хранятся в базе, могут существовать логические связи. Например, многие группы студентов входят в состав одного факультета, и ни одна группа не может относиться сразу к двум факультетам. Говорят, что между факультетом и группой имеется связь «один ко многим».

Отображение связи «один ко многим». В режиме конструктора дополните структуру таблицы «Группы» полем «код_ф», затем перейдите в режим просмотра таблицы и распределите группы по факультетам, то есть введите в поле «код_ф» коды факультетов, хранящиеся в поле «код_фак» таблицы «факультет».

Подмена кода объекта именем. Access позволяет во время просмотра данных в таблице «Группы» подменять код факультета его названием, взятым из таблицы «факультет». Для этого откройте таблицу «Группы» в режиме конструктора, выберите в свойствах поля «код_ф» вкладку «Подстановка» и установите следующие значения свойств:

- для свойства «Тип элемента управления» - Поле со списком,
- для свойства «Тип источника строк» - Таблица или запрос,
- для свойства «Источник строк» - факультет,
- для свойства «Присоединённый столбец» - 1,
- для свойства «Число столбцов» - 2,
- для свойства «Ширина столбцов» – 0,
- для свойства «Ограничиться списком» – да.

Теперь перейдите в режим таблицы. В поле «код_ф» должны появиться названия факультетов.

Внесение связей между таблицами в схему данных. В Access можно создать схему данных, в которой явно указываются связи между полями разных таблиц. Схема данных используется для поддержки целостности базы данных и, кроме того, упрощает построение форм и запросов.

Закройте таблицу «Группы». Выполните пункты меню Сервис Схема данных. В любом месте открывшегося окна щёлкните правой кнопкой мышки. В появившемся меню выберите пункт «Добавить таблицу» и добавьте таблицы «Группы» и «факультет». Левой кнопкой мышки соедините поле «код_фак» и поле «код_ф» связываемых таблиц. Должно появиться окно «Изменение связей».

Установите флажок «Обеспечение целостности данных». Теперь, если не устанавливать два других флажка, Access при попытке удалить из таблицы «факультет» любую запись будет проверять, нет ли в таблице «Группы» кода удаляемой записи. Если такой код есть, то появится сообщение об ошибке. Другими словами, пока в базе данных указано, что на факультете есть хотя бы одна группа, удалить этот факультет нельзя.

Флажок «Каскадное удаление связанных записей» служит для автоматического удаления из таблицы «Группы» всех записей, связанных с записью, удаляемой из таблицы «факультет».

Флажок «Каскадное обновление связанных полей» служит для автоматического обновления в таблицы со стороны «многие» значения связанного поля, обновляемого в таблице со стороны «1». В нашем случае связанное поле – счётчик и оно не может обновляться.

После закрытия окна «Изменение связей» на схема данных появится связь «один ко многим» между таблицами «факультет» и «Группы» (рис. 2).

Создание подчинённой формы. Создайте новую форму. Назовите её «факультеты». Отобразите в ней все поля, кроме «код_фак».

Рис. 2. Схема данных

Теперь в форме «факультеты» нужно создать подчинённую форму, в которой будут показываться данные о группах, входящих в состав показываемого в главной форме факультета. Раздвиньте границы формы «факультеты так, чтобы в ней поместилась таблица с данными о группах. На панели элементов найдите элемент «Подчинённая форма/отчёт» и установите его в форме «факультеты».

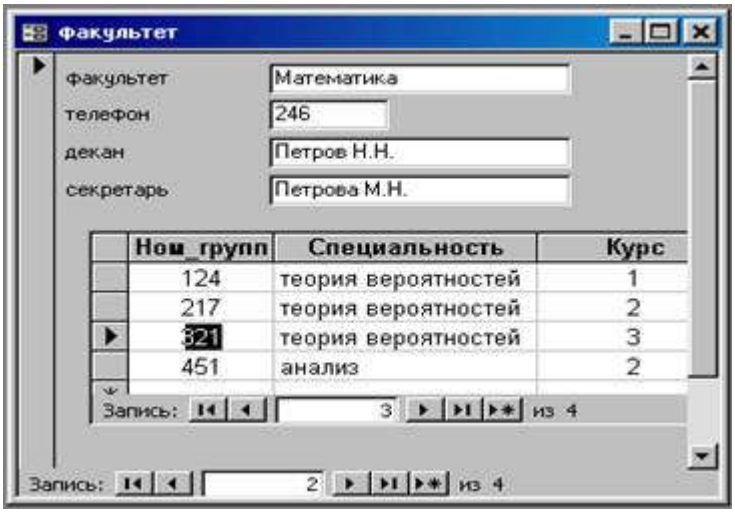


Рис.3. Форма *факультет* с подчинённой формой

В окне мастера подчинённых форм установите переключатель «Имеющиеся таблицы и запросы» и щёлкните по кнопке «Далее». В следующем окне мастера подчинённых форм выберите таблицу «группы» и все поля, кроме «Код_гр». Выберите переключатель «Выбор из списка» и оставьте предложенное название подчинённой формы. На рис. 3 показана форма «факультет» с подчиненной формой,. в которой выведены группы, входящие в состав просматриваемого в основной форме факультета. Перейдите в режим формы и просмотрите несколько за писей.

ПР21-22 Построение схем баз данных

Цель: научиться создавать вспомогательные таблицы для сведения одной связи «многие ко многим» к двум связям «один ко многим» и обеспечения целостности данных.

Задание

В данной работе нужно создать таблицу «дисциплины», вспомогательную таблицу «преп_дис» и схему связей между между ними и ввести данные в созданные таблицы.

Пример объектов, имеющих связи «многие ко многим» - преподаватели и предметы (дисциплины). Один преподаватель может вести несколько дисциплин, и одну дисциплину могут вести в разных группах разные преподаватели. Наиболее рациональный способ хранения данных о связях «многие ко многим» - это создание вспомогательной таблицы, в каждой записи которой хранятся код преподавателя и код предмета. Между основными таблицами и вспомогательной устанавливаются связи «один ко многим». Вспомогательной таблицы позволяет избежать избыточности данных в базе, уменьшить количество ошибок при вводе и упростить запросы к базе данных.

Создание таблиц. Создайте таблицу «дисциплины», имеющую следующую структуру:

№ п.п.	Имя поля	Тип данных	Размер поля
1	код_дис	Счётчик	Длинное целое
2	предмет	Текстовый	40
3	лекции	Числовой	Целое
4	практика	Числовой	Целое

Сделайте поле «код_дис» ключевым. В полях «лекции» и «практика» хранится коичество учебных часов, отводимых на лекции и практику соответственно. Введите в таблицу «дисциплины» несколько записей.

Создайте таблицу «преп_дис», имеющую следующую структуру:

№ п.п	Имя поля	Тип данных	Размер поля
1	код_преп_дис	счётчик	Длинное целое
2	код_преп	Числовой	Длинное целое
3	код_дис	Числовой	Длинное целое

Введите в таблицу «преп_дис» несколько записей, связывающих преподавателей и дисциплины. На рис. 4 показан пример, в котором преподаватель Андреев В.К. ведёт Pascal, информатику и Fortran, а Семёнова Е.М. ведёт информатику и Pascal.

Внесение связей в схему данных аналогично внесению связей между таблицами «Группы» и «факультет» в предыдущей лабораторной работе. Только с таблицей «преп_дис» связаны две таблицы (рис. 5).

ПР23-24 Построение схем различного уровня сложности.

Цель: научиться создавать:

- подчинённые формы для просмотра данных из основных таблиц с использованием вспомогательных,

- экранные формы, с помощью которых при вводе пользователем данных, имеющих связи «многие ко многим», автоматически заполняется вспомогательная таблица (см. лаб. работу №6) со связями «один ко многим».

Задание

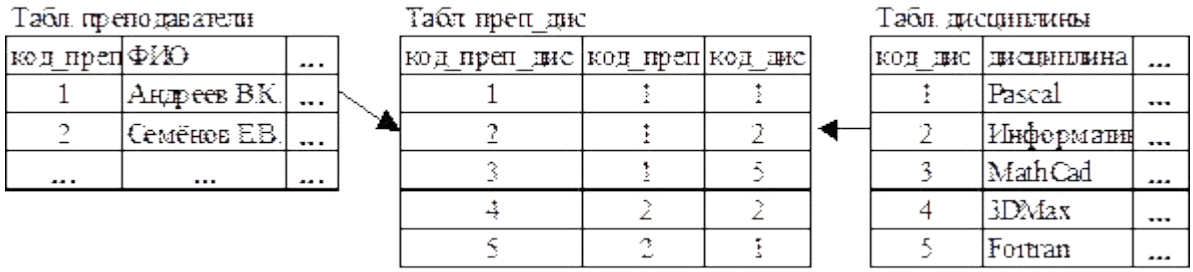


Рис. 4. Связывание данных с помощью вспомогательной таблицы

Дополнение формы «преп_просм» подчинённой формой с данными о предметах, которые ведёт преподаватель. Откройте форму «преп_просм» в режиме конструктора. На панели элементов найдите элемент «Подчинённая форма/отчёт» и установите его в открытой форме.

Далее описываются два способа создания подчинённой формы со связями «многие ко многим»:

- с помощью мастера подчинённых форм,
- с использованием запроса к базе данных.

. С помощью мастера создать подчинённую форму проще, но при отсутствии на компьютере мастера необходимо использовать запрос к базе данных.

Создание подчинённой формы с помощью мастера. В окне мастера подчинённых форм установите переключатель «Имеющиеся таблицы и запросы» и щёлкните по кнопке «Далее».

В следующем окне мастера подчинённых форм выберите

- таблицу «дисциплины» и в ней все поля,
- таблицу «преп_дис» и в ней поля «код_преп» и «код_дис».

Перейдите в следующее окно и выберите переключатель «Выбор из списка» и оставьте предложенное название подчинённой формы.

Перейдите в режим формы и просмотрите несколько записей.

Рис. 5. Схема данных с использованием вспомогательной таблицы

Создание подчинённой формы с использованием запроса к базе данных. Сначала нужно создать запрос на выборку из базы данных. (Подробно методы создания запросов рассматриваются во второй части данного пособия.) Откройте окно базы данных. В левой части окна выберите пункт меню «Запросы». В верхней части окна выберите вкладку «Создать». Создайте запрос в режиме конструктора.

В окне «Добавление таблицы» выберите таблицы «преподаватели», «дисциплины» и «преп_дис». Закройте окно «Добавление таблицы». окне «запрос на выборку» Перетащите мышкой поля из списков полей таблиц расположенных в верхней части окна «запрос на выборку», в нижнюю часть так , как это показано на рис. 6.

Выполните запрос с помощью пунктов меню Запрос □□ Запуск. Результат запроса – новая (временная) таблица, состоящая из записей, содержащих все допустимые комбинации значений полей из трёх исходных таблиц. Закройте окно запроса. Назовите запрос «преп_дисц».

Откройте форму «преп_просм» в режиме конструктора. Установите элемент «Подчинённая форма/отчёт» в форму «преп_просм». Отмените использование мастера подчинённых форм (или закройте окно с сообщением о том, что мастер не установлен).

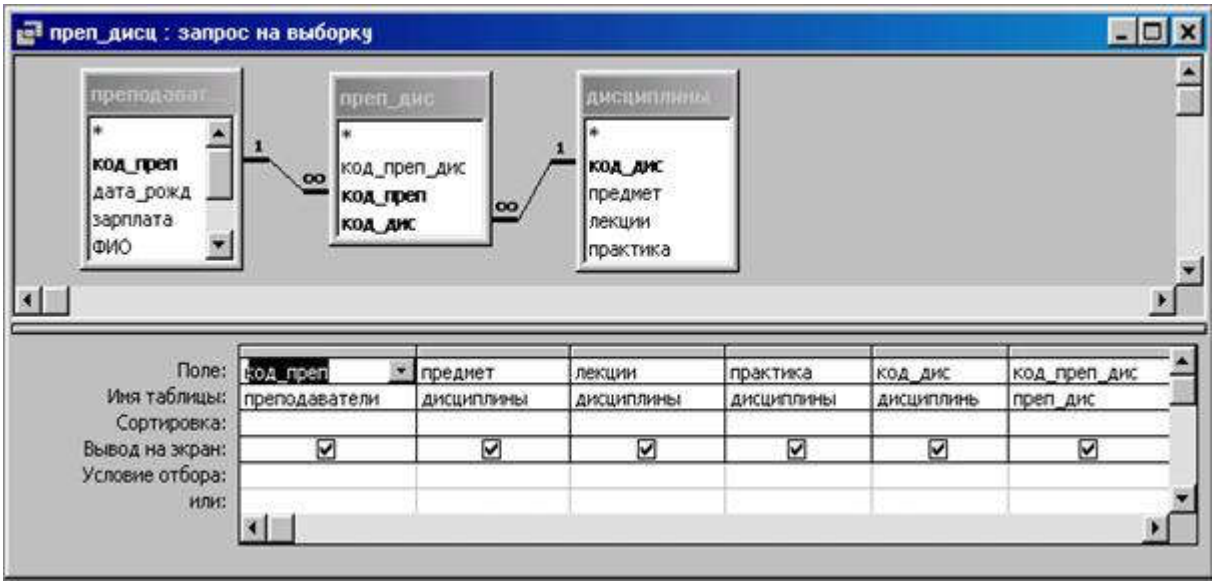


Рис. 6. Запрос, сформированный средствами Access

Подведите указатель мышки к квадратику в верхнем левом углу подчинённой (внедрённой) формы и щёлкните правой кнопкой. В появившемся окне выберите пункт «Свойства». Появится окно «Форма/отчёт». Выберите в нём вкладку «Данные». Установите для свойства объект-источник значение «Запрос.преп_дисц». Перейдите в режим просмотра формы. Примерный вид формы показан на рис. 7.

Дополнение формы «преп_ред» подчинённой формой с данными о предметах, которые ведёт преподаватель. Любым из двух способов, описанных выше в данной лаб. работе, вставьте в форму «преп_ред» подчинённую форму.

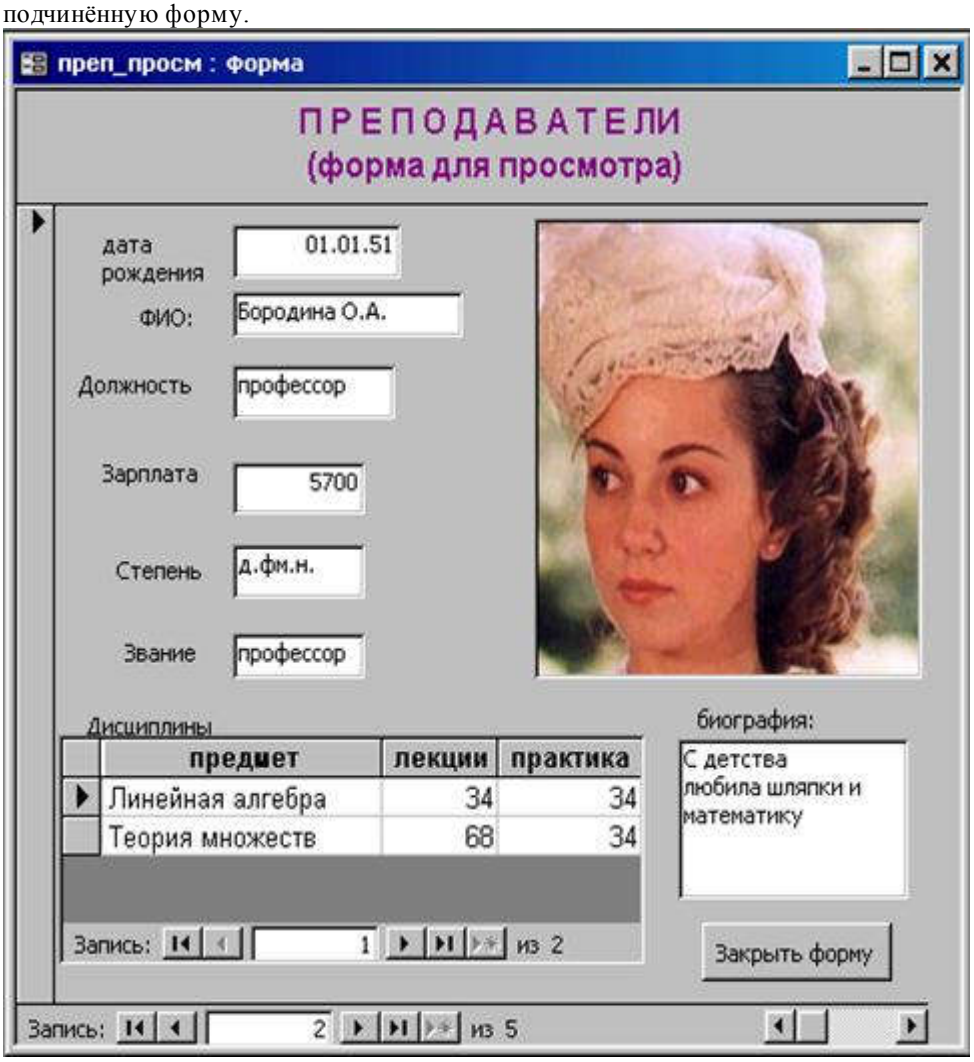


Рис. 7. Форма для просмотра данных о преподавателях

Данные о дисциплинах используются для формирования учебной нагрузки многих преподавателях, поэтому редактировать их нужно в специальной форме, а в подчинённой форме их редактирование нужно запретить. Для этого установите свойство подчинённоё формы «доступ» в состояние «нет».

Самостоятельно создайте формы для ввода, просмотра и редактирования данных о дисциплинах, аналогичные формам для работы с данными о преподавателях. Эти формы понадобятся для построения информационной системы «Кафедра». Назовите формы «дис_ввод», дис_просм» и «дис_ред».

ПР25-26 Манипулирование данными (хранение, добавление, редактирование данных)

Цель: создать форму для автоматического заполнения вспомогательной таблицы, хранящей, добавляющей и редактирующей только коды записей двух таблиц со связей «многие ко многим».

Задание

Предполагается, что при эксплуатации создаваемой формы (рис. 8) пользователь, распределяющий нагрузку преподавателей, будет указывать преподавателя и ведомую им дисциплину, а СУБД Access, исходя из этих данных, будет создавать в таблице «преп_дис» запись с кодами выбранных преподавателя и дисциплины.

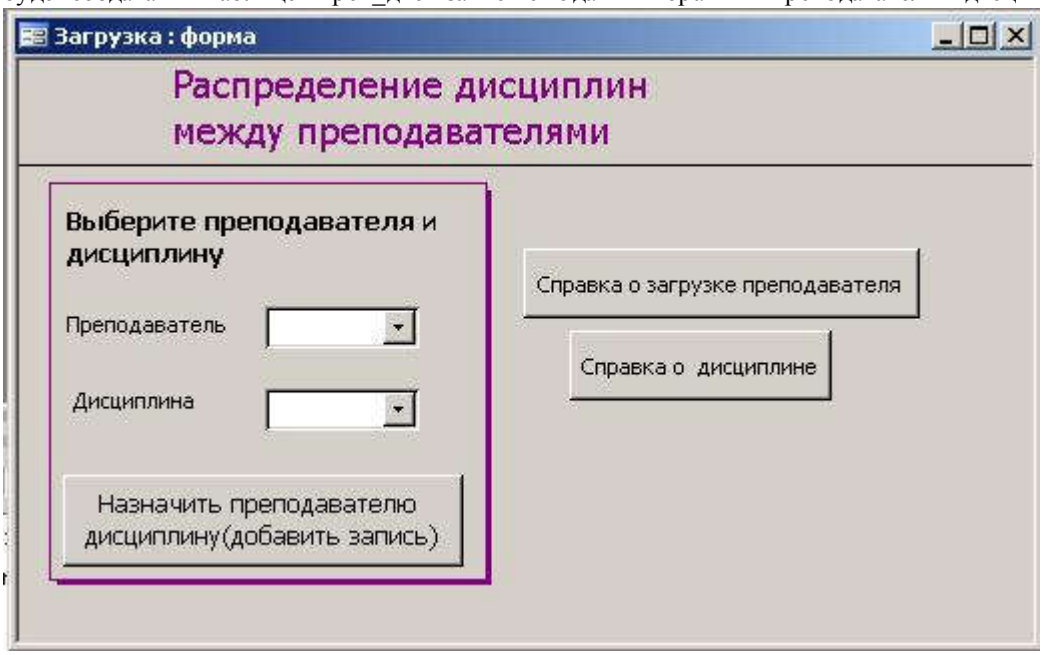


Рис.8. Форма для ввода учебной загрузки преподавателей

Кнопки *«Сприавка о . . .»* нужны для того, чтобы открыть формы для просмотра преподавателей и дисциплин и определить, достаточно ли загружен преподаватель и сколько преподавателей уже преподают данную дисциплину.

Создайте форму в режиме конструктора. Выберите таблицу «преп_дис» в качестве источника записей. Укажите, что форма предназначена для ввода.

Создайте поле со списком для выбора преподавателя. Для этого выберите на панели элементов поле со списком и поместите его в форму. Подтвердите, что поле со списком будет использовать значения из таблицы или запроса и щёлкните по кнопке «далее». Выберите таблицу «преподаватели» и снова щёлкните по кнопке «далее».

Включите в поле со списком поля «код_преп» и «ФИО». Подтвердите, что нужно скрыть ключевой столбец. Далее нужно указать, что выбранное из таблицы «преподаватели» значение поля «код_преп» нужно сохранить в поле «код_преп» создаваемой записи в таблице «преп_дис».

Создайте поле со списком для выбора дисциплины. Все действия такие же, как и при создании предыдущего поля со списком. Только нужно выбрать таблицу «дисциплины» и поле «код_дис».

Кнопки создаются так же, как и в лаб. работе № 3.

ПР27-28 Сортировка, поиск и фильтрация и объединение данных

Цель: объединить в единую систему с помощью управляющей формы все объекты созданные в лабораторных работах №№ 1, 2, 3, 6, 7, 8. Управляющая форма позволит пользоваться информационной системой, не зная ни структуры базы данных, ни названий форм.

Задание

В предыдущих лабораторных работах были созданы таблицы и формы для ввода, просмотра, редактирования и удаления данных о преподавателях и дисциплинах кафедры, а также данных об учебной загрузке преподавателей.

Главная форма. В лабораторной работе № 3 была разработана кнопочная форма (рис. 1). Необходимо дополнить её средствами для работы с дисциплинами и распределения загрузки преподавателей (рис.9).

Поддержка целостности при вводе и редактировании данных. Для предотвращения ввода по ошибке одних и тех же данных несколько раз в базе данных используются ключи. Ключом называются один или несколько столбцов (атрибутов) таких, что в таблице не найдётся ни одной пары строк с одинаковым набором данных в выбранных в качестве ключа столбцах.

В теории баз данных принято все ключи таблицы называть потенциальными. Один ключ выделяют по каким-либо соображениям и называют *первичным*, а остальные – *альтернативными*. В Access первичный ключ называют просто ключом. Для того, чтобы в Access установить контроль за уникальностью значений в каком-либо поле, нужно это поле назначить индексированным и указать, что совпадения в нём не допускаются.

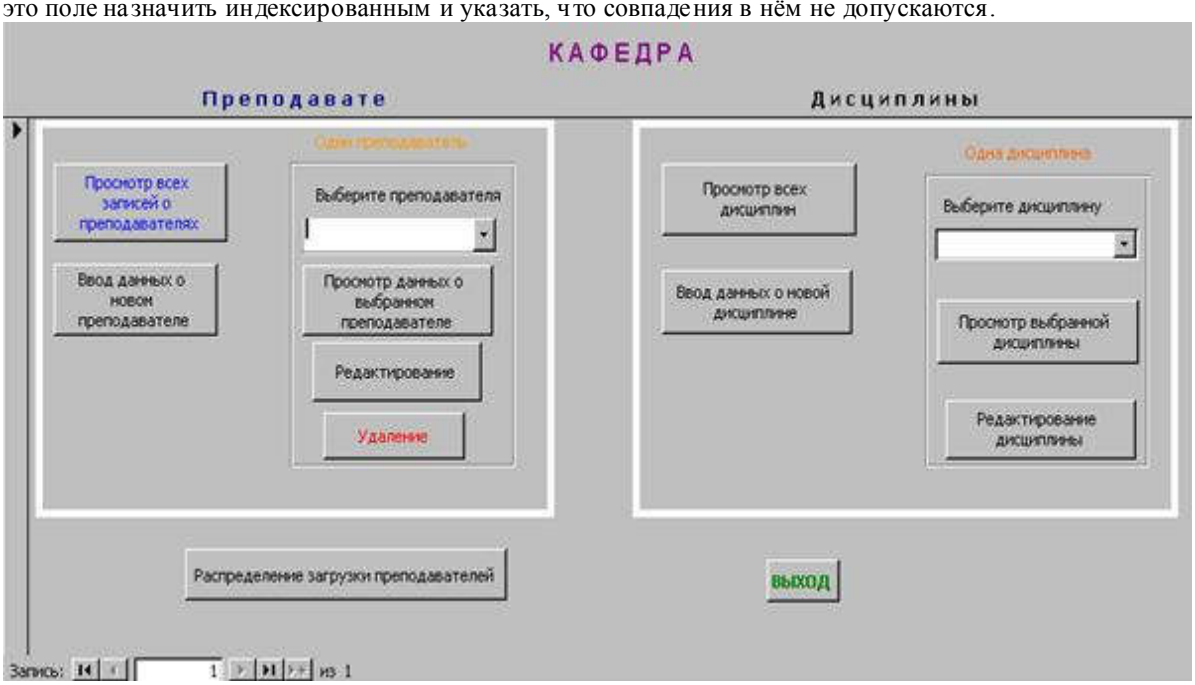


Рис. 9. Главная форма информационной системы «Кафедра»

В таблице «дисциплины» два потенциальных ключа: «код_дис» и «предмет». Поле «код_дис» назначено (первичным) ключом. Поэтому необходимо сделать столбец «предмет» индексированным с запретом поторяющихся значений.

Откройте таблицу «дисциплины» в режиме конструктора. Установите курсор на поле «предмет» и выберите для свойство «индексированное поле» значение «Да (Совпадения не допускаются)».

В таблице «преп_дис» нельзя допустить повторения пары значений «код_преп» и «код_дис». Нужно создать *составной ключ* из этих полей. Откройте таблицу «преп_дис» в режиме просмотра и убедитесь в том что ни одному преподавателю не назначена дважды одна дисциплина. Перейдите в режим конструктора и выделите поля «код_преп» и «код_дис». Щёлкните по значку с изображением ключа на панели инструментов.

Теперь при попытке повторно ввести данные в таблицы «дисциплины» и «преп_дис» Access будет сообщать об ошибке.

Комплексная отладка информационной системы. После создания информационной системы необходимо убедиться в правильности её работы. В процессе отладки необходимо имитировать условия работы реального пользователя системы.

Проверьте правильность работы всех элементов главной формы.

Введите через форму для ввода данные о трёх-четырёх преподавателях.

Введите через форму для ввода данные о пяти-шести дисциплинах.

Через форму «загрузка» распределите учебную загрузку между преподавателями. Убедитесь в том, что попытка дважды назначить одному преподавателю один и тот же предмет вызывает сообщение об ошибке.

Проверьте работу форм для просмотра и редактирования.

Удалите с помощью кнопок удаления в формах редактирования несколько записей о преподавателях и дисциплинах.

Исправьте обнаруженные ошибки, добавьте недостающие формы и элементы.

ПР29-30 Технология разработки и построения запросов вСУБД

Цель: научиться создавать с помощью конструктора одно- и многотабличные запросы на выборку записей с заданным набором полей и удовлетворяющие заданным условиям.

Задание

Во всех современных СУБД запросы пишутся на языке SQL (Structured Query Language – язык структурированных запросов). В ACCESS также есть возможность писать запросы на SQL, но разработчики ACCESS ориентируют пользователя на максимальное использование средств автоматизации для создания и ведения баз данных. К этим средствам относится конструктор запросов, с помощью которого можно быстро создать многие запросы.

Запрос на просмотр всех данных одной таблицы. (Этот запрос неявно генерируется СУБД Access при открытии таблицы в режиме «таблица».) Для создания запроса откройте окно базы данных. Выберите объект «Запросы». В верхней части окна выберите вкладку «Создать». Создайте запрос в режиме конструктора.

В окне «Добавление таблицы» выберите таблицу «Заказы». Закройте окно «Добавление таблицы». В окне «запрос на выборку» Перетащите символ «*» (звёздочка) из списка полей таблицы «Заказы» в крайнее левое поле в нижней части окна. После этого запрос готов. Выполните его, выбрав в меню пункты Запрос Запуск.

Для лучшего понимания дальнейших упражнений внимательно просмотрите содержимое таблицы «Заказы».

Во всех современных СУБД запросы пишутся на языке SQL. Просмотрите созданный запрос в режиме SQL (пункты меню Вид Режим SQL). Запрос имеет вид

```
SELECT Заказы.*
FROM Заказы;
```

Запрос читается так: выбрать (SELECT) все поля (Заказы.*) из таблицы (FROM) Заказы.

Запрос на просмотр всех записей с заданным набором полей. Измените предыдущий запрос. Замените символ «*» именем поля «Клиент». Добавьте в запрос поля «ДатаИсполнения», «СтоимостьДоставки» и «НазваниеПолучателя». Просмотрите созданный запрос в режиме таблицы и в режиме SQL.

Выборка из таблицы записей, удовлетворяющих заданным условиям. Нужно выбрать из таблицы «Заказы» все заказы, у которых стоимостью доставки не меньше 35 р. и меньше 40 р. Для этого внесите изменения в предыдущий запрос. В условиях отбора поля «СтоимостьДоставки» запишите «>=35 AND <40». *Здесь и далее кавычки «» в условие не входят.*

Выполните запрос. Должно быть отобрано 22 заказа (записи).

Просмотрите созданный запрос в режиме SQL. Обратите внимание на условие после ключевого слова WHERE.

Добавьте в запрос ещё одно условие. В условия отбора поля «НазваниеПолучателя» запишите «Like ‘R*’», означающее «выбрать названия получателей, начинающиеся на R».

Многотабличные запросы. Связанные между собой данные, хранящиеся в нескольких таблицах можно выбирать одним запросом.

В таблицах «Заказы» и «Заказано» хранятся данные о заказах. Таблицы связаны с помощью поля «КодЗаказа». В одном заказе заказывается несколько товаров. В таблице «Заказы» хранятся общие сведения о заказе, а в таблице «Заказано» – сведения о заказанных товарах из этого заказа. Тип связи между таблицами «Заказы» и «Заказано» – один ко многим. Такая же связь установлена между таблицами «Товары» и «Заказано» через поле «КодТовара».

Создайте запрос на выборку всех марок товаров, заказанных клиентом ANTON. Для этого нужно в режиме конструктора запросов

- выбрать все три таблицы «Заказы», «Заказано» и «Товары»,
- из таблицы «Заказы» выбрать поле «КодКлиента» , установить для него условие отбора «ANTON» и запретить его вывод на экран,
- из таблицы «Товары» выбрать поле «Марка» и сортировку по возрастанию,
- чтобы одна марка товара не была выбрана несколько раз, установить свойство запроса «уникальные значения» в положение «да».

В режиме SQL запрос, сгенерированный Access, выглядит следующим образом:

```
SELECT DISTINCT Товары.Марка
FROM Товары INNER JOIN (Заказы INNER JOIN Заказано ON Заказы.КодЗаказа =
Заказано.КодЗаказа) ON Товары.КодТовара = Заказано.КодТовара
WHERE (((Заказы.КодКлиента)="ANTON"))
ORDER BY Товары.Марка;
```

ПР31-32 Автоматизация расчетов с помощью запросов в СУБД

Цель: научиться создавать и изменять средствами SQL таблицы и индексы, задавать ограничения целостности.

Задание

Под созданием и изменением здесь подразумевается только создание и изменение структуры и параметров таблиц, а не хранящихся в них данных. Часть языка SQL, служащая для решения этих задач, называется языком описания данных (Date Definition Language – DDL).

Создание таблицы. Для создания таблицы в SQL служит команда CREATE TABLE.Синтаксис простейшего варианта команды CREATE TABLE:

CREATE TABLE <**Имя таблицы**>
(<**имя поля**> <**тип данных**>[(<**размер**>)],
<**имя поля**> <**тип данных**>[(<**размер**>] ...);

В стандартном языке SQL применяются следующие типы данных:

- INTEGER – до 10 цифр и знак;
- SMALL – до 5 цифр и знак;
- DECIMAL(p,q) – 0<p<16 всего позиций, q – цифр после запятой;
- FLOAT – вещественное, определяется СУБД (REAL в ACCESS);
- DOUBLE PRECISION – вещественное, определяется СУБД (FLOAT в ACCESS!!!), точность и диапазон больше, чем у FLOAT;
- CHAR(n) – строка из n (n<256) символов.

Практически во всех СУБД, поддерживающих SQL, применяются дополнительно следующие типы данных:

- VARCHAR(n) – строка из n символов (n_{max} >4096 определяется СУБД);
- DATE – формат определяется специальной командой (по умолчанию mm/dd/yy);
- TIME – формат определяется специальной командой (по умолчанию hh.mm.ss);
- DATETIME – комбинация даты и времени;
- MONEY – денежный.

Подробнее о типах данных, поддерживаемых СУБД ACCESS, смотрите в справке ACCESS в ответе на вопрос «Типы данных SQL».

Пример команды на создание таблицы «Страна» (название, площадь, численность населения в млн чел.):

```
CREATE TABLE Страна
(название CHAR(60),
площадь REAL,
население REAL);
```

Самостоятельно с помощью команды CREATE TABLE создайте таблицу «Изделие» со следующими атрибутами:

- название
- цена
- вес
- дата изготовления
- фирма-производитель

Подберите соответствующие типы данных.

Создание запросов на SQL в Access начинается вызовом конструктора запросов. Для этого в окне базы данных нужно выбрать объект «Запросы», пункт меню «Создать» и в окне «Новый запрос» пункт «Конструктор». Далее, не выбирая таблицу, закройте окно «Добавление таблицы» и перейдите в режим SQL. Переход в режим SQL : меню Access Вид Режим SQL.

Набрав в окне SQL запрос на создание таблицы «Изделие», выполните его.

Введите в созданную таблицу данные о трёх произвольных товарах. В режиме конструктора таблиц проверьте, как интерпретировал типы данных ACCESS.

Внесение изменений в структуру таблицы делается с помощью команды ALTER TABLE, имеющей следующий синтаксис:

```
ALTER TABLE <имя таблицы> {ADD {COLUMN <имя поля> <тип поля>[(<размер>)]
[NOT NULL]    [CONSTRAINT <имя индекса>] |
  ALTER COLUMN <имя поля> <тип поля>[(<размер>)]|
  CONSTRAINT <описание составного индекса>} |
  DROP {COLUMN <имя поля> | CONSTRAINT <имя индекса>} };
```

Команда, с помощью которой к таблице «Страна» добавляется поле «столица» выглядит так:

```
ALTER TABLE Страна
ADD COLUMN столица VARCHAR(40) NOT NULL UNIQUE;
```

На поле «столица» наложены 2 ограничения: не допускается пустое поле (NOT NULL) и название столицы должно быть уникальным (UNIQUE).

Для добавления к таблице «Страна» поля, являющегося первичным ключом, служит команда

```
ALTER TABLE Страна
ADD COLUMN Id_strana INTEGER NOT NULL PRIMARY KEY;
```

В поле Id_strana должен храниться номер записи. Приведённая выше команда *не создаёт автоматического счётчика*.

Самостоятельно с помощью команд ALTER TABLE добавьте к таблице «Изделие» следующие поля:

- Id_izdelie – номер записи и первичный ключ;
- сорт – сорт (1-й, 2-й, …), не допускается пустых полей.

Замечание. Перед выполнеием команды ALTER TABLE необходимо из таблицы «Изделие» удалить все записи, так как в имеющихся записях новые поля не могут иметь значение NULL, т.е. быть пустыми.

Создание таблицы с ограничениями столбцов и ограничениями таблицы. В качестве примера создадим таблицу «отдых» связанную с таблицей «страна». Тип связи «многие к одному». Таблица «отдых» будет иметь следующие поля:

- Id_otd – номер записи (первичный ключ);
- Id_st - внешний ключ, связывающий с таблицей «Страна»;
- курорт – название курорта;
- гостиница;
- продолжит – продолжительность отдыха в днях.

Совокупность значений полей «Id_st», «курорт» и «гостиница» должна быть уникальной, то есть, потенциальным ключом. Таким образом, в таблице «курорт» будет 2 ключа. Описанная таблица создаётся следующей командой:

```
CREATE TABLE отдых
( Id_otd INTEGER NOT NULL PRIMARY KEY,
  Id_st INTEGER REFERENCES Страна(Id_strana),
  курорт CHAR(80),
  гостиница CHAR(60),
  стоимость REAL,
  продолжительность SMALLINT,
  UNIQUE (Id_st,курорт,гостиница));
```

Самостоятельно с помощью команд CREATE TABLE создайте таблицу «поставка», связанную с таблицей «Изделие».

Таблица «Поставка» должна иметь следующие поля:

- номер записи (первичный ключ),
- внешний ключ, связывающий с таблицей «Изделие»;
- адрес клиента,
- дату поставки,
- количество товара,
- стоимость доставки.

Совокупность значений внешнего ключа, адреса клиента и даты поставки должна быть уникальной, то есть, потенциальным ключом.

После создания таблицы «Поставка» введите в неё несколько записей. Для того, чтобы убедиться в правильной работе потенциального ключа, попытайтесь ввести две записи с одинаковыми значениями внешнего ключа, адреса клиента и даты поставки.

Добавьте таблицы «Изделие» и «Поставка» к схеме данных (Меню ACCESS □□ Схема данных). ACCESS должен автоматически обнаружить связь «один ко многим», заданную при создании таблиц командой CREATE TABLE.

Создание индекса. Если таблица велика (обычно большой считается таблица, содержащая сотни тысяч записей), то для ускорения поиска в ней данных строятся индексы. Синтаксис команды для создания индекса следующий:

```
CREATE [UNIQUE] INDEX <имя индекса>
ON <имя таблицы> <(список полей)>
```

Для построения индекса по полю «название» в таблице «Страна» служит следующая команда:

```
CREATE UNIQUE INDEX названиеIn
ON страна(название)
```

Самостоятельно постройте уникальный индекс для поля «Название» таблицы «Изделие». Проверьте его действие, введя две записи с одним наименованием изделия.

ПР33-34 Разработка клиентской части базы данных в инструментальной оболочке

Цель: научиться формировать на языке SQL простейшие запросы к базе данных, использовать в запросах выражения, включающие в себя арифметические операции, функции для работы со строками и датами, агрегатные функции.

Задание

Запрос на выборку всей таблицы. В лабораторной работе № 10 такой запрос уже был сформирован средствами Access. Можно упростить вид запроса, если вместо запроса

SELECT Заказы.* FROM Заказы;

написать запрос

SELECT * FROM Заказы

Язык SQL позволяет опускать имя таблицы перед именем поля в тех случаях, когда в запросе используется одна таблица, или имя поля не повторяется в нескольких таблицах в многотабличном запросе.

Создание запросов на SQL в Access начинается вызовом конструктора запросов. Для этого в окне базы данных нужно выбрать объект «Запросы», пункт меню «Создать» и в окне «Новый запрос» пункт «Конструктор». Далее выберите таблицу «Заказы» и перейдите в режим SQL. Переход в режим SQL: меню Access Вид Режим SQL.

Зокончите формирование запроса и выполните его.

Вывод избранных полей, замена имён полей псевдонимами, сортировка записей. Поля таблицы выводятся на экран дисплея в том порядке, в котором они перечислены в запросе. Имена полей при выводе результатов запроса часто неудобны для чтения. Их можно заменить в запросе псевдонимами, как показано в примере:

SELECT КодЗаказа AS Заказ, НазваниеПолучателя AS Получатель, АдресПолучателя AS Адрес,
ДатаИсполнения AS Дата
FROM Заказы
ORDER BY НазваниеПолучателя ASC;

В примере *КодЗаказа, НазваниеПолучателя, АдресПолучателя* и *ДатаИсполнения* – имена полей в таблице «Заказы». При выводе результатов запроса на экран дисплея имена полей будут заменены соответствующими псевдонимами, указанными после слова *AS*.

Предложение

ORDER BY НазваниеПолучателя ASC

служит для сортировки отобранных записей по возрастанию (т.е. в алфавитном порядке) значения поля *НазваниеПолучателя*. Если нужно сортировать по убыванию, то вместо ASC нужно использовать DESC (сокращение от descending).

Сформируйте и выполните этот запрос.

Вывод записей без дублирования. Сформируйте и выполните следующий запрос

SELECT НазваниеПолучателя AS Получатель
FROM Заказы
ORDER BY НазваниеПолучателя DESC.

Названия получателей многократно повторяются, так как выбраны все записи таблицы. Чтобы не было дублирования записей, добавьте в запрос после слова SELECT слово DISTINCT. Иногда в СУБД режим DISTINCT установлен по умолчанию. Для вывода *всех* записей в этом случае после слова SELECT вставляется слово ALL.

Использование в запросе выражений. В списке вывода можно указывать не только имена полей и их псевдонимы, но и выражения, включающие в себя арифметические действия и функции.

Умножение. Сформируйте запрос на вывод из таблицы «Заказано» кода товара, цены, количества и общей стоимости заказанного товара. Запрос выглядит так:

SELECT КодТовара,Цена,Количество,Цена*Количество AS Стоимость
FROM Заказано;

Самостоятельно дополните запрос стоимостью со скидкой.

Использование функций. Функция **STR()** предназначена для преобразования в текстовый тип. Для вывода на экран дисплея стоимости товара в тысячах рублей с указанием единицы измерения служит следующий запрос:

SELECT КодТовара,*str*(Цена*Количество/1000)+' тыс. руб' AS Стоимость FROM Заказано;

Для того чтобы в колонке «Стоимость» печатались число и текст, нужно преобразовать число в текстовый тип и объединить с текстом 'тыс. руб.'. Для преобразования служат функция *str(<выражение числового типа>)* и операция слияния «+» (конкатенация).

Сформируйте запрос, в котором из таблицы «Заказы» выбираются 5 полей и результат выводится в две колонки. В первую колонку выводится поле «КодЗаказа», а в колонке с псевдонимом «Адрес клиента» объединены следующие поля: ИндексПолучателя, СтранаПолучателя, ГородПолучателя, НазваниеПолучателя.

Не забудьте поставить между объединяемыми полями адреса запятую с пробелом. Результат запроса (показаны две первые строки) должен иметь вид:

Код заказа	Адрес клиента
10248	90110, Финляндия, Оулу, Wartian Herkku

Код заказа	Адрес клиента
10249	44087, Германия, Мюнстер, Toms Spezialitaten

Функция выделения части даты DATEPART(). Познакомьтесь с описанием этой функции в справке Access (Содержание, раздел «Справочник по языку Visual Basic», пункт «Functions», буква D).

Определите с помощью запроса к таблице «Заказы», за какие годы были поставки товаров.

Агрегатные функции. (В Access они называются статистическими). Подсчитаем общее количество записей в таблице «Заказы» и количество записей содержащих данные в поле «ОбластьПолучателя», то есть, количество записей с непустым полем «ОбластьПолучателя». Для этого выполним следующий запрос:

```
SELECT count(*),count(ОбластьПолучателя)
FROM Заказы;
```

В запросе используется агрегатная функция *COUNT()*. Используя агрегатные функции MAX(), MIN() и AVG(), составьте запрос для подсчёта максимальной минимальной и средней цены товара в таблице «Товары».

Используя агрегатную функцию SUM(), составьте запрос для подсчёта общей стоимости доставки всех заказанных товаров в таблице «Заказы».

Сохраните все созданные Вами запросы и покажите их преподавателю.

ПР35-36 Построение концептуальной модели базы данных

Цель: научиться составлять условия, которым должны удовлетворять выбираемые из таблицы записи, научиться использовать специальные предикаты SQL.

Задание

Синтаксис оператора выборки по условию следующий:

```
SELECT <список полей и выражений>
FROM <имя таблицы>
WHERE <условие> .
```

Условие – это выражение, принимающее значение «истина» или «ложь». Такое выражение называется *предикатом*. В предикате могут использоваться:

- поля,
- константы,
- арифметические действия +, - , *, / , возведение в степень **,
- операторы отношения =, <, >, <= ,>=, <>,
- логические операции NOT, OR, AND.

В SQL имеются специальные предикаты BETWEEN, IN, LIKE, IS NULL, ANY или SOME, ALL, EXISTS.

Использование в условии логических операций. Пусть из таблицы «Заказано» нужно выбратьтовары с ценой не меньше ста рублей и не больше двухсот, вывести все поля таблицы, кроме поля «КодЗаказа» и сформировать расчётное поле, в котором показать стоимость товара с учётом скидки. Запрос имеет вид:

```
SELECT КодТовара, Цена, Количество,
Скидка, Цена* Количество*(1- Скидка) AS [Стоимость со скидкой]
FROM Заказано
WHERE Цена>100 And Цена<200;
```

Обратите внимание на то, что скидка хранится в базе данных не в процентах, а в сотых долях от стоимости товара. Скидка 7% в базе хранится как 0.07. Выполните этот запрос.

Измените условие в запросе так, чтобы выбирались товары со скидкой больше 7% и либо имеющие цену больше двухсот рублей либо количество не меньше тридцати.

Отсортируйте выбранные записи в порядке убывания цен.

Использование агрегатных функций для отобранных записей. Сформируйте и выполните следующие запросы к таблице «Заказано»:

подсчитать суммарную стоимость всех заказов, с ценой меньше100 руб.;

подсчитать количество записей, в которых код заказа принимает значения 10252 или 10255 или больше 11000;

найти минимальную, максимальную и среднюю цены для товаров с количеством, равным 10.

Использование в условии выборки функций для работы с датами. При работе с базами данных часто приходится производить операции с датами. Познакомьтесь с описаниями функций для работы с датами в справке Access (Содержание, раздел «Справочник по языку Visual Basic», пункт «Functions», буквы C, D).

Функция CDATE(). Найдём в таблице «Заказы» все заказы, исполненные между 01.07.1996 и 01.01.1997. Для этого нужно преобразовать даты с помощью функции CDATE() из текстового типа в тип «date». Запрос имеет вид:

```
SELECT *
FROM Заказы
WHERE ДатаИсполнения>=CDATE('01.07.1996') AND ДатаИсполнения<CDATE('01.01.1997');
```

Сформируйте и выполните этот запрос.

Даты можно вычитать друг из друга. Разность получается в днях.

Самостоятельно сформируйте запрос на выборку из таблицы «Заказы» всех заказов, исполненных после 15.04.1998 и выполненных более чем за 5 дней.

Функция DatePart() служит для выделения из даты её части, например, года. Познакомьтесь с описанием этой функции в справке Access (Содержание, раздел «Справочник по языку Visual Basic» пункт «Functions», буква D).

С помощью функции DatePart() найдите все заказы, размещённые

a) в первом квартале 1997 г;
b) по понедельникам в январе за все годы.

Использование в условии выборки списка значений. Для этих целей служит специальный предикат IN, имеющий следующий синтаксис:

<выражение> IN (<список значений>)

Найдём в таблице «Заказано» все товары со скидками, равными 5%, 10%, 25% и 30%. Запрос имеет вид:

```
SELECT Цена, Скидка, КодЗаказа
FROM Заказано
WHERE Скидка In (0.05,0.1,0.25,0.3)
ORDER BY Скидка,Цена;
```

В запросе записи упорядочены по скидкам и цене в возрастающем порядке.

Самостоятельно найдите в таблице «Заказы» все заказы, размещённые:

a) в мае , августе и декабре в1996 и 1998 годах;
b) пятого августа в 1996, 1997 и 1998 годах.

Использование в условии выборки диапазона значений Диапазон значений задаётся с помощью предиката BETWEEN, имеющего следующий синтаксис:

<выражение> BETWEEN <значение 1> AND <значение 2>

Если значения числовые, то границы диапазона включаются в в выборку.

Запрос на выборку из таблицы «Заказано» товаров с ценами между 500 и 1000 руб. можно составить так:

```
SELECT *
FROM Заказано
WHERE Цена BETWEEN 500 AND 1000
ORDER BY Скидка,Цена;
```

Внесите в этот запрос дополнительное условие: *или со скидкой между 5% и 10%.*

Можно выбирать символьные величины, например, названия стран, которые начинаются с букв из заданной последовательности. Выберем из таблицы «Заказы» все заказы из стран, названия которых начинаются на А, Б, В, …, К:

```
SELECT *
FROM Заказы
WHERE СтранаПолучателя Between 'А' And 'Л'
ORDER BY СтранаПолучателя
```

Обратите внимание на то, что диапазон поиска на одну букву больше, чем в условии задачи. Access ищет все названия стран, начинающиеся на А,Б, …,К и имеющие любую длину, а на букву ‘Л’ ищет названия страны длиной в одну букву, то есть страну ‘Л’. Можно в качестве конца диапазона указать ‘Кя’, тогда также будут найдены все страны на букву ‘К’.

Самостоятельно выберите из таблицы «Заказы:

a) только поле СтранаПолучателя, при условии, что названия стран начинаются на А, Б, В или Р, С, Т и выборке не должно быть повторений названий стран.
b) алфавитном порядке города (поле ГородПолучателя) от Лилля до Парижа.

Формирование с помощью предиката LIKE условных выражений со строковыми полями. В ACCESS предикат LIKE называют оператором.

Оператора LIKE сравнивает строковое поле со строковым выражением. Пусть, из таблицы «Заказы» нужно выбрать все заказы при условии, что название города, в котором находится получатель, начинается на «Л». Запрос выглядит так:

```
SELECT *
FROM Заказы
WHERE ГородПолучателя Like 'Л*';
```

Символ «*» означает «любая последовательность из нуля или более символов». Кроме «*» используются и другие символы групповой замены (willcards).

Познакомьтесь с описанием оператора LIKE в справке Access (Содержание, раздел «Справочник по языку Visual Basic», пункт «Operators», LIKE Operator). В табл. 2 приводятся примеры использования в операторе LIKE символов групповой замены.

Таблица 2

Тип совпадения	Образец	Совпадение (True)	Несовпадение (False)

Несколько знаков	a*a	aa, aBa, aBBBa	aBC
	ab	abc, AABB, Xab	aZb, bac
	ab*	abcdefg, abc	cab, aab
Специальный знак	a[*]a	a*a	aaa
Одиночный знак	a?a	aaa, a3a, aBa	aBBBa
Одиночная цифра	a#a	a0a, a1a, a2a	aaa, a10a
Диапазон знаков	[a-z]	f, p, j	2, &
Вне диапазона	[!a-z]	9, &, %	b, a
Не цифра	[!0-9]	A, a, &, ~	0, 1, 9
Комбинированное выражение	a[!b-m]#	An9, az0, a99	abc, aj0

Сформируйте, используя оператор LIKE, и выполните следующие запросы к таблице «Заказы»:

a) выбрать все заказы с названием города получателя, начинающимся на А, Л или П;

b) изменить предыдущий запрос, выбирая только поле ГородПолучателя и не допуская повторений;

c) выбрать все заказы с названием города получателя, начинающимся на Л, а со второй буквой – «и» или «о»;

d) выбрать названия городов, состоящие из пяти букв и начинающиеся на букву П;

e) выбрать заказы, в которых адрес получателя сдержит запятую;

f) выбрать заказы, в которых адрес получателя начинается с цифры;

g) выбрать заказы, в которых адрес получателя начинается не с цифры;

h) выбрать заказы, в которых адрес получателя начинается не с букв с С (лат.) по L и не с цифры;

i) выбрать заказы, в которых адрес получателя начинается с цифры и имеет длину, не более 20 символов.

Для определения длины строки используется функция LEN(«строковое выражение»).

Сохраните запросы и покажите их преподавателю.

ПР37-38 Создание логической модели данных с помощью утилиты автоматизированного проектирования базы данных

Цель: научиться применять агрегатные функции к группам записей, имеющим общие свойства.

Задание

Предложение GROUP BY. Синтаксис:

GROUP BY <список полей>

Предложение GROUP BY применяется для разбиения таблицы на группы строк и применения к каждой группе агрегатных функций. Рассмотрим пример. Пусть необходимо в таблице «Заказано» подсчитать для каждого наименования (кода) товара количество заказов и максимального количества товара в одном заказе. Для уменьшения количества выводимых результатов ограничимся кодами товара от 1 до 5. Задача решается с помощью следующего запроса :

```
SELECT Заказано.КодТовара, Count(*) AS [К-во заказов], Max([Количество]) AS [Макс_к-во_товара]
FROM Заказано
WHERE КодТовара<6
GROUP BY КодТовара ;
```

Результаты запроса:

КодТовара	К-во заказов	Макс_к-во_товара
1	38	80
2	45	100
3	14	60
4	23	50
5	10	70

Самостоятельно составьте и выполните следующие запросы к таблице «Заказы»:

a) подсчитать для каждой страны количество заказов, минимальную, среднюю и максимальную стоимость доставки;

b) подсчитать для каждого города суммарную стоимость доставки всех заказов, отсортировать выбранные записи по суммарной стоимости доставки;

c) подсчитать для каждой страны суммарную стоимость доставки всех заказов;

d) подсчитать для каждого города каждой страны суммарную стоимость доставки всех заказов;

Предложение HAVING служит для задания условий, содержащих агрегатные функции.

Пример. Выбрать в таблице «Заказы» города для которых сделано более десяти заказов. Вывести название города и количество заказов. Запрос выглядит так:

```
SELECT ГородПолучателя, Count(КодЗаказа) AS [К-во заказов]
FROM Заказы
GROUP BY ГородПолучателя
HAVING Count(КодЗаказа)>10;
```

Самостоятельно составьте и выполните следующие запросы:

a) выбрать из таблицы «Заказано» коды товаров, у которых максимальная скидка больше 20%;
b) выбрать из таблицы «Заказано» для каждого товара код товара, среднюю, минимальную и максимальную цены при условии, что средняя цена меньше 2000

Сохраните запросы и покажите их преподавателю

ПР39-44 Создание физической модели данных с помощью утилиты автоматизированного проектирования базы данных (6 часов)

Цель: освоить методы выборки из базы данных информации, размещённой в нескольких соединяемых таблицах

Задание

Для дальнейших упражнений понадобятся следующие таблицы из базы данных «Борей»: «Заказы», «Заказано», «Сотрудники», «Клиенты» и «Товары». Импортируйте в Вашу базу данных недостающие таблицы.

Создайте с помощью оператора CREATE TABLE таблицы «писатель» и книга».и заполните их так , как показано на рис 10. В колонке «Автор» таблицы «книга» указаны коды писателей из таблицы «писатель». Несколько клеток в последних строках обеих таблиц специально оставлены пустыми. В строке 7 таблицы «книги» указан код 12, отсутствующий в таблице «писатель».

Табл. книга

КодКн	Наим	Автор
1	В и М	1
2	А.К.	1
3	Воскр	1
4	Стихи	2
5	Ев. О.	3
6	Р и Л	3
7	Обломов	12
8	ПётрI	
9	На дне	

Табл. писатель

КодП	ФИО
1	Толстой Л.Н.
2	Есенин
3	Пушкин
4	Тургенев
	Горький

Рис. 10. Заполнение таблиц *книги* и *писатели*

Таблицы, состоящие из двух столбцов, в одном из которых хранится наименование объекта, а в другом – номер или код, называется справочником. Код из справочника используется в других таблицах вместо имени объекта. Замена наименования кодом уменьшает вероятность ошибки при вводе данных и позволяет при изменении наименования, например фамилии, внести изменение только в одно поле справочника.

Многотабличный запрос с описанием связей между таблицами в предложении WHERE. Запрос на выборку наименований книг и их авторов выглядит так:

```
SELECT a.Наим,b.ФИО
FROM книга a,писатель b
WHERE a.Автор=b.КодП
```

Введите и выполните этот запрос. Обратите внимание на то, что данные из строк с пустыми полями в результаты запроса не входят.

В многотабличном запросе можно использовать любые условия для отбора данных. Если не использовать никаких условий, то будет выведено декартово произведение из всех строк, таблиц, используемых в запросе, то есть 4*9=36 строк. Удалите из последнего запроса предложение WHERE и выполните получившийся запрос.

Внутренние соединения. В SQL в многотабличных запросах с условием отбора данных можно применять конструкцию, называемую внутренним соединением (INNER JOIN). Сформированный выше запрос на выборку наименований книг и их авторов можно переписать так

```
SELECT a.Наим,b.ФИО
FROM книга a INNER JOIN писатель b ON a.Автор=b.КодП
```

В результате данного запроса не выбраны названия книг и авторы из записей, не удовлетворяющих условию: книга.Автор=писатель.КодП.

Левые и правые внешние соединения. Чтобы кроме данных из записей, удовлетворяющих условию запроса, включить в выборку названия книг из записей, не удовлетворяющих условию запроса, применяется «левое внешнее соединение» (LEFT JOIN):

```
SELECT a.Наим,b.ФИО
FROM книга a LEFT JOIN писатель b ON a.Автор=b.КодП.
```

Для книг «Обломов», «Пётр I» и «На дне» поле ФИО в результатах этого запроса останется пустым

Чтобы кроме данных из записей, удовлетворяющих условию запроса, включить в выборку ФИО писателей из записей, не удовлетворяющих условию запроса, применяется «правое внешнее соединение» (RIGHT JOIN):

```
SELECT a.Наим,b.ФИО
FROM книга a RIGHT JOIN писатель b ON a.Автор=b.КодП
```

Введите и выполните последние 3 запрос. Сравните результаты выборок между собой и с первым запросом данной лаб. работы.

Самостоятельно составьте и выполните следующие запросы:

- выбрать все имеющиеся в базе названия произведений Пушкина и Толстого Л.Н. и вместе с ФИО авторов;
- используя таблицы «заказы» и «клиенты», выбрать названия клиентов, их представителей (поле «ОбращатьсяК») и даты размещения их заказов при условии, что клиенты из Лондона;
- используя таблицы «заказы» «заказано», «товары» и «клиенты», выбрать названия клиентов, их представителей (поле «ОбращатьсяК») и коды и марки выбранных ими товаров при условии, что названия клиентов начинаются на “F” и заказы оформлял сотрудник Кротов.

Объединение таблицы с самой собой применяется в тех случаях, когда в одной строке результатов выполнения запроса требуется выводить данные из разных строк исходной таблицы.

Пример. Допустим, в таблице «Сотрудники» имеются поля «ФИО», «должность» и «отдел». Требуется найти всех программистов и их начальников. Известно, что программисты работают почти во всех отделах. Запрос выглядит так:

```
SELECT a.ФИО AS программист,b.ФИО AS начальник_отдела
FROM Сотрудники a,Сотрудники b
WHERE a.должность=’программист’ AND b.должность=’нач_отдела’
    AND a.отдел=b.отдел
```

Результаты будут выглядеть примерно так:

Программист	Начальник_отдела
Андреев А.Б.	Петров К.Ю.
Борисова Г.П.	Петров К.Ю.
Смирнов П.С.	Новикова А.Г.

Самостоятельно составьте и выполните следующий запрос: из таблицы «Клиенты» выберите всех продавцов и представителей из тех городов, в которых есть и продавцы и представители.

ПР45-46 Разработка серверной части базы данных в инструментальной оболочке

Цель: написание программного интерфейса для хранения изменений данных в БД и организации доступа к ним

Задание

Реализовать возможность просматривать содержимое базы auto на заданную дату для одного поля. Для этого доработать программу протоколирования (л.р.№4), добавив пункты:

1. "Переход на заданную дату" – выводятся только те данные в базе, которые были актуальны на заданную дату(выводится запрос на ввод даты); 2. "Переход на текущую дату" – аналогично п.1, только на текущую дату время.

В программе должны быть объявлены 2 переменных: текущая дата и заданная дата. При запуске программы спрашивается текущая дата (по умолчанию действительная текущая дата). На экране должна отображаться текущая дата и заданная дата (по умолчанию равна текущей дате).

Процедура поиска ищет данные удовлетворяющее заданному условию и актуальные на дату, указанную в переменной, содержащей заданную дату. При откате на заданную дату доступен лишь просмотр базы и поиск. Выполнение обязательного задания = 55%

Бонус (+ 45%): Реализовать возможность просматривать содержимое базы AUTO на заданную дату для всех полей базы.

ПР47-48 Модель сервера баз данных

Цель: познакомиться с интерфейсом взаимодействия с PostgreSQL, а также научиться применять некоторые нетривиальные возможности СУБД в качестве сервера БД

Задание

Выберите предметную область (можно из л/р №1) и опишите структуру БД, используя SQL-запросы. К обязательным требованиям относится использование:

1. объектно-реляционных связей;
2. ограничений в таблицах;
3. массивов;
4. последовательностей;
5. а также backup и restore БД для переноса с домашнего ПК.

Прием работы производится только, если она удовлетворяет всем требованиям.

Прием работы

Прием происходит при наличии оформленного отчета и работающей БД.

Вопросы

1. Что такое PostgreSQL, какой язык использует в качестве языка БД, к какому классу ПО (открытое или закрытое) относится? Какая архитектура? Какие клиентские приложения входят в пакет?
2. Как организуется объектно-реляционные связи в СУБД postgreSQL и какие особенности организации могут приводить к визуальному нарушению ограничений установленных в таблицах?
3. Что такое ограничения полей, ограничения таблиц? Как они используются и для чего?
4. Как использовать поля-массивы: как обращаться к элементам массивов, как создавать массивы?
5. Автоматизация стандартных процедур.
6. Что такое последовательности? Как могут быть использованы?
7. Что такое триггеры? На каких языках могут быть реализованы?

ПР49-50 Компоненты SQL server 2008

Цель: познакомиться с возможностями языка SQL server 2008

Задание

Данная лабораторная работа не является обязательной для выполнения, однако необходима для получения оценки выше «4» на экзамене.

1. Создайте отношение A. Для этого отношения определите два поля:

поле – номер (тип число) и поле – строка_значений (тип одномерная матрица). Таблица будет иметь следующую структур:

номер_матрицы	строка_значений
1	‘{2,4}’
1	‘{-3,1}’
2	‘{0}’
2	‘{5}’
3	‘{-2,-4}’

Что соответствует матрицам $A1 = \begin{pmatrix} 2 & 4 \\ -3 & 1 \end{pmatrix}$, $A2 = \begin{pmatrix} 0 \\ 5 \end{pmatrix}$ и $A3 = \begin{pmatrix} -2 & -4 \end{pmatrix}$.

2. Создайте отношение B, содержащее четыре поля: номер_операции (целое), номер_первой матрицы (целое), номер_второй матрицы (целое), название_функции (текст).
3. Создайте третье отношение – C, которое имеет поля: номер_операции (целое) и поле результат (numeric).
4. Создайте функции сложения, вычитания, транспонирования, умножения матриц, которые работают с матрицами, определенными в отношении A, т.е. на вход получают номер матрицы (или матриц – для бинарных операций).
5. Создайте функцию вычисления определителя матрицы по ее номеру.
6. Создайте триггер, который:

- при добавлении новой строки в отношение B (с новым номером_операции) производит расчет для этой операции, вызвав соответствующую функцию для переданных матриц, и записывает полученный результат в отношение C в виде нового кортежа;
- при обновлении строки в отношении B производит пересчет, согласно новых переданных параметров, если пересчет произведен без ошибок, то обновляет соответствующий кортеж в отношении C;

- при выборе строки из отношения В производит пересчет, согласно существующих параметров.

Прием работы

Прием происходит при наличии оформленного отчета и работающей БД.

Вопросы

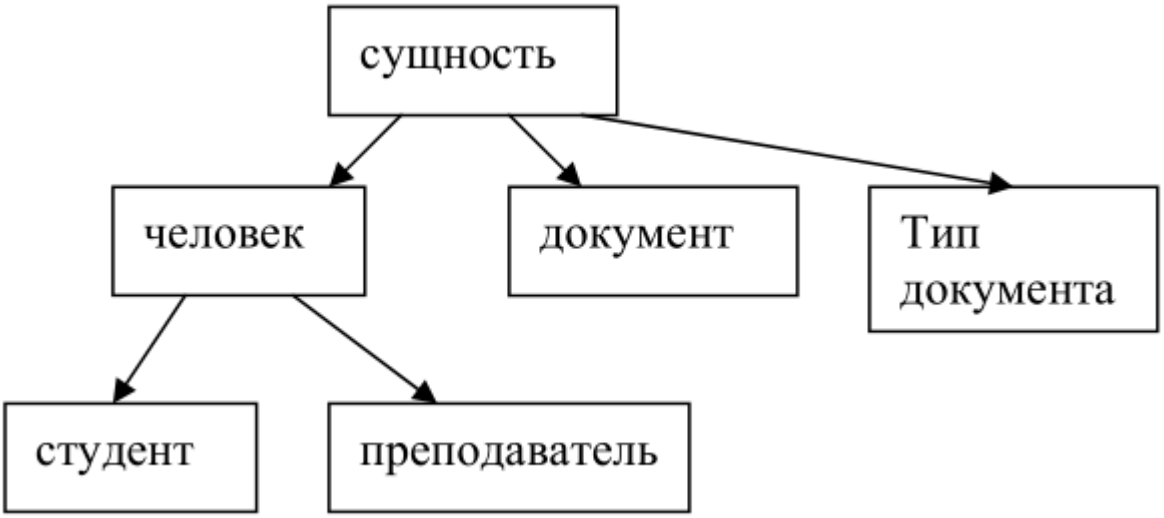
1. Чем отличается использование атрибута %ROWTYPE от типа RECORD?
2. Что такое PL/pgSQL и из каких блоков состоит процедура на этом языке?
3. Что такое триггеры и триггерные функции?
4. Как можно вставить данные в переменную типа RECORD?
5. Какие циклы существуют в языке PL/pgSQL.

ПР51-52 Модели клиент-сервер

Цель: моделирование объектного подхода «Клиент-сервер» на реляционной БД.

Задание

1. Создать несколько отношений, связанных в виде иерархии, как это показано на рисунке:



2. Самостоятельно определить атрибуты этих отношений.
3. Иерархию реализовывать с использованием наследования (л/р 6).
4. Создать представление, которое выбирает все атрибуты объекта и его наследников в один кортеж. В случаи, если для какого-то из атрибутов имеется несколько значений необходимо формировать поле в следующем виде:

{‘<1ое значение атрибута>’, ‘<2ое значение атрибута>’, …}, где <i-ое значение атрибута> - значение атрибута в i-ом кортеже для объекта. Для этого написать собственную агрегатную функцию(ии), работающую с типами integer, text, timestamp.

5. Определить универсальные функции для удаления, добавления, обновления любого объекта, которым передается имя отношения, фильтр (если нужно), массив имен полей (если нужно), массив новых значений полей (если нужно).
6. На основании функций из п.5 определить для каждого объекта БД (кроме «сущность») функции: добавить, изменить, удалить. В функциях должен быть реализован контроль за уникальностью объекта.
7. Запретить добавление данных в отношения с использованием SQL запросов (т.е. не через интерфейсные функции из пункта 5).Для этого определить необходимые триггеры.
8. Написать функции (PL/PGSQL) вывода существующих документов человека (например, паспорт, з/к – если студент, № пропуска – если преподаватель). Функция использует представление созданное в пункте 4. Не использовать внешние ключи (реляционные связи) для связывания отношений находящихся в одной ветке иерархии, но использовать их (если необходимо) для связи объектов на одном уровне иерархии. Значение потенциального ключа в базовых таблицах не должно повторяться даже при выполнении запроса без параметра ONLY.
9. Структура БД, ограничения, правила наследования, процедуры, представления, а также данные должны быть представлены в виде SQL-скрипта.

ПР53-56 Системные базы данных (4 часа)

Цель: научиться работе в системных базах данных. Получить навыки работы с системными запросами и отчетами

Задание

Предикат IS NULL. Для выяснения смысла значения NULL рассмотрим пример. Пусть в городе N ведётся база данных, в которой хранятся данные обо всех жителях, включая детей. Очевидно, что в графу «профессия» записи о ребёнке поместить нечего, так как у ребёнка ещё нет профессии. Графа профессия может оказаться пустой и в том случае, когда в момент занесения данных профессия жителя не была известна. Предполагается, что графа будет заполнена позже. Для неизвестного значения в SQL применяется специальное обозначение NULL. Значение NULL имеют по умолчанию все поля, в которые ничего не заносилось.

NULL применяется в полях всех типов и само не имеет типа. Значение NULL можно использовать только в специальном предикате IS NULL, имеющем следующий синтаксис:

<выражение> IS [NOT]NULL

Предикат IS NULL принимает значение «истина» только, если выражение равно NULL.

Для работы с NULL-значениями полей создайте в базе данных таблицу NullPusto, состоящую из двух текстовых полей длиной по 30 символов. Назовите поля «ФИО» и «адр». Введите в таблицу данные из табл. 3.

Создайте и выполните следующие запросы к таблице NullPusto:

a) выбрать все записи с NULL;
b) выбрать все записи с “”;
c) выбрать все записи, в которых есть адреса;
d) выбрать все записи, в которых нет адресов;
e) подсчитать количество записей, содержащих NULL;
f) подсчитать количество записей, содержащих NULL и “”.

Сохраните запросы и покажите их преподавателю.

Таблица 3

Поле		Значение в поле «адр»
ФИО	адр	
А	К	“К”
Б		“” (две двойные кавычки)
В		NULL
Г	М	“М”
Д		NULL
Е		“” (две двойные кавычки)
Ж		NULL

Подзапросы. С помощью SQL можно вкладывать один запросы внутрь другого. Внутренний запрос называют подзапросом. Обычно, внутренний запрос генерирует значение, которое проверяется в предикате внешнего запроса, определяющего верно оно или нет. Например, в следующем запросе выбираются из таблицы «Товары» те товары, цена которых меньше средней цены всех товаров таблицы:

```
SELECT *
FROM Товары
WHERE Цена<(SELECT AVG(Цена) FROM Товары);
```

Самостоятельно с помощью подзапроса выберите из таблицы «Заказано» заказы на товары с маркой «Pavlova». Марки товаров хранятся в таблице «Товары».

Предикат EXISTS имеет синтаксис

EXISTS подзапрос

и принимает значение ИСТИНА (TRUE), если подзапрос содержит хотя бы одну строку.

В следующем запросе выбираются фамилии всех сотрудников, оформлявших заказы для клиента ANTON, при условии, что хотя бы один заказ для клиента ANTON был размещён в мае любого года.

```
SELECT DISTINCT b.Фамилия
FROM Заказы a, Сотрудники b
WHERE EXISTS (SELECT * FROM Заказы WHERE КодКлиента='ANTON' AND DatePart('m',ДатаРазмещения)=5)
   AND a.КодСотрудника = b.КодСотрудника AND a.КодКлиента='ANTON';
```

Самостоятельно, используя таблицы «Сотрудники», «Клиенты» и «Заказы», создайте и выполните запрос на выборку всех клиентов из Рио-Де-Жанейро, если был сделан хотя бы один заказ из Рио-Де-Жанейро, оформленый сотрудником Кротовым.

Предикаты количественного сравнения ANY, SOME и ALL имеют синтаксис

оператор сравнения {ANY | SOME | ALL} подзапрос.

ANY и SOME – синонимы.

Пример использования предиката ANY:

```
SELECT КодЗаказа
FROM Заказы
WHERE СтоимостьДоставки
< ANY(SELECT СтоимостьДоставки FROM Заказы WHERE ГородПолучателя  ='Ванкувер');
```

Для исследования особенностей предиката ANY проделайте следующее упражнение:

a) выберите из таблицы «Товары» цены товаров от поставщика с кодом 2; запишите эти цены;
b) используя ANY, выберите все товары, цены которых больше цен поставщика 2; сравните выбранные цены с записанными;
c) повторите предыдущий пункт, используя вместо ANY предикат ALL; сравните результаты.

Сохраните все выполненные запросы и покажите их преподавателю

ПР57-58 Оптимизация запросов, управляемых правилами

Цель: научиться объединять в одной выводимой таблице строки, полученные разными запросами и создавать новую таблицу базы данных из существующих таблиц.

Задание

Предложение UNION применяется для объединения результатов нескольких запросов в одной выводимой таблице. Количество столбцов во всех запросах должно быть одинаковым и типы соответствующих столбцов должны быть сравнимыми. В следующем примере выводятся адреса и города клиентов и заказов. Параметр ALL разрешает выводить дубликаты строк.

```
SELECT ALL Адрес,Город,'Заказы ' AS Источник
FROM Клиенты
UNION
SELECT ALL АдресПолучателя AS Адрес,ГородПолучателя AS Город,'Клиенты ' AS Источник
FROM Заказы;
```

Выполните этот запрос.

Самостоятельно выберите из таблиц «Клиенты» и «Сотрудники» следующие данные:

- фамилию и имя;
- должность;
- город.

В дополнительном столбце укажите , из какой таблицы выбрана запись.

Создание таблицы из существующих таблиц с помощью SELECT … INTO. Во многих СУБД конструкция SELECT … INTO <имя таблицы> используется для создания новой таблицы и вывода в неё результатов запроса. Например, таблица «Страны» с названиями всех стран, в которые направляются заказы, создаётся в результате выполнения следующего запроса:

```
SELECT DISTINCT СтранаПолучателя
INTO Страны
FROM Заказы;
```

Самостоятельно с помощью SELECT … INTO создайте таблицу «Клиенты2», содержащую данные из таблицы «Клиенты» обо всех клиентах, живущих в Лондоне.

Сохраните выполненные запросы и покажите их преподавателю

ПР59-60 Обьектно-ориентированные модели данных

Цель: освоить способы редактирования, вставки и удаления записей в рамках объектно-ориентированной модели

Задание

Вставка в таблицу одной или нескольких строк с помощью оператора INSERT. Синтаксис оператора INSERT:

```
INSERT INTO <имя таблицы>
[(<имя столбца>)]
{VALUES (<значение> .,..)}
|<выражение запроса>
|{DEFAULT VALUES};
```

Пример. Добавим в созданную в лаб. работе №15 таблицу «книга» книгу М. Горького «Детство». Так как в таблице «писатель» Горькому не присвоен код, то в добавляемой строке будут заполняться только столбцы «КодКн» и «Наим». Описанная строка добавляется с помощью оператора

```
INSERT INTO  книга
(КодКн,Наим)
VALUES (10,'Детство');
```

Столбец «КодКн» не является счётчиком, поэтому он указан в списках столбцов и добавляемых значений. Счётчик в операторе INSERT указывать не надо.

Самостоятельно добавьте в таблицу «писатель» Толстого А.Н. и в таблицу «книга» - роман «Сёстры».

Изменение (редактирование) данных в таблице с помощью оператора UPDATE. Синтаксис оператора UPDATE:

UPDATE <имя таблицы>
SET {<имя столбца>={<выражение для вычисления значения>
|NULL |DEFAULT}}
[WHERE <предикат>]

Пример. Укажем в таблице «писатель» код Горького:

UPDATE писатель
SET КодП=10
WHERE ФИО='Горький';

Самостоятельно с помощью оператора UPDATE занесите в таблицу «книга» все недостающие значения полей.

Удаление строк таблицы с помощью оператора DELETE. Синтаксис оператора DELETE:

DELETE FROM <имя таблицы>
[WHERE <предикат>]

Пример. Удалим из таблицы «книга» книгу «На дне»

DELETE FROM **книга** WHERE **КодКн=7**

Самостоятельно с помощью оператора DELETE удалите из таблицы «писатель» Тургенева.

Сохраните выполненные запросы и покажите их преподавателю

ПР61-62 Cache и WWW-технологии

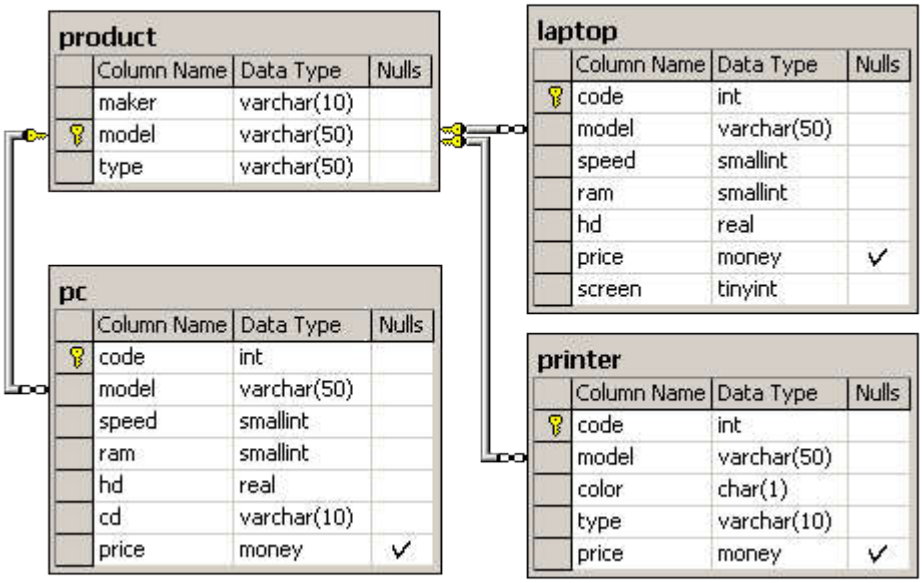
Цель: освоить технологии удаленно администрирования – cashe & www

Задание

1. Создать базу данных «Компьютерная фирма». В таблицах PC, Laptop, Printer сделать по 5 записей. В таблицу Product – соответственно 15.
2. Создать SQL запросы следующих типов:
   - Запрос с упорядочиванием поля (по возрастанию, убыванию);
   - запрос с переименованием полей;
   - объединить две таблицы;
   - объединить таблицы по заданному признаку;
3. Получить у преподавателя индивидуальное задание для защиты.

Краткая информация о базе данных "Компьютерная фирма":

Схема БД состоит из четырех таблиц:
Product(maker, model, type)
PC(code, model, speed, ram, hd, cd, price)
Laptop(code, model, speed, ram, hd, price, screen)
Printer(code, model, color, type, price)
Таблица Product представляет производителя (maker), номер модели (model) и тип ('PC' - ПК, 'Laptop' - ПК-блокнот или 'Printer' - принтер). Предполагается, что номера моделей в таблице Product уникальны для всех производителей и типов продуктов. В таблице PC для каждого ПК, однозначно определяемого уникальным кодом – code, указаны модель – model (внешний ключ к таблице Product), скорость - speed (процессора в мегагерцах), объем памяти - ram (в мегабайтах), размер диска - hd (в гигабайтах), скорость считывающего устройства - cd (например, '4x') и цена - price. Таблица Laptop аналогична таблице PC за исключением того, что вместо скорости CD содержит размер экрана -screen (в дюймах). В таблице Printer для каждой модели принтера указывается, является ли он цветным - color ('y', если цветной), тип принтера - type (лазерный – 'Laser', струйный – 'Jet' или матричный – 'Matrix') и цена - price.



ПР63-64 Структурированный язык запросов SQL

Цель: Обощение пройденного по структурированному языку запросов SQL

Задание

Создайте в базе данных «Компьютерная фирма» следующие запросы:

1. Вывести следующие поля принтеров: производитель, модель, цвет, цена. Упорядочив их по цене с переименованием столбцов на русский язык.
2. Вывести все поля компьютеров, в названии моделей которых не встречается заданная буква.
3. Вывести количество ноутбуков, у которых оперативная память более 1 ГБ и частота процессора выше 1,8 Ггц.
4. Вывести самый дорогой цветной принтер.
5. Вывести все поля PC тех записей, где цена ниже средней.
6. Вывести весь прайс-лист компьютерной фирмы в одном списке, показывая только три поля: производитель, модель, цена (Union).
7. Если в таблицах PC и laptop есть модели с одинаковой частотой процессора, то вывести ноутбуки с объемом жесткого диска 500 Гб и выше. (Exixts)
8. Вывести матричные принтеры, если все принтеры дороже 1000 рублей. (exists)
9. Сгруппировать ноутбуки по частоте процессора. В каждой категории самую высокую цену (group by)
10. Сгруппировать ноутбуки по диагонали монитора, в каждой категории показать наименьший объем жесткого диска. При этом показывать только те группы, которых в каждом разряде больше одного (group by having).

ПР65-66 Операторы обработки SQL запроссов

Цель: Получить первичные навыки с операторами обработки запросов SQl

Задание

1. Дополнить существующую базу данных «Компьютерный магазин» (см. Практическую работу №2). В таблицах PC. Laptop и printer добавить по 5 значений. Соответственно переписать эти 15 значений в таблицу product.
2. В этой таблице освоить следующие виды запросов:
   - счет полей – count;
   - вывод максимального, минимального, среднего значения, суммы;
   - подзапросы;
   - объединение запросов;

Примеры запросов:

1. Посчитать количество товаров, цена которых выше заданной.
2. Вывести количество товаров заданного производителя.
3.Показать самый дешевый товар в таблице.
4. Вывести все поля товаров, чья цена ниже средней.
5. Показать все товары в таблице, кроме самого дорогого.
6. Объединить таблицы pc, laptop, printer в одном запросе при помощи Union, выводя только модели и их цену.

ПР67-68 Построение запросов к базе данных на языке SQL (различных типов)

Цель: дополнить свой опыт в обращении с языком запросов SQL запросами новых типов

Задание

Выполните следующие виды запросов с имеющейся базой данных «Компьютерная фирма».

1. Вывести все компьютеры, имеющие в своей комплектации cd-привод. (is null)
2. Вывести все поля принтеров, если в таблице PC и таблице принтеров совпадает цена хотя бы одного товара. (Exists)
3. Вывести все значения ноутбуков, имеющих HDD более 1 ТБ, если существуют компьютеры с объемом HDD более 1 ТБ. (exists)
4. Вывести те значения ноутбуков, цена которых выше хотя бы одного PC. (any, some).
5. Вывести те компьютеры, которые дороже самого дорогого ноутбука (all).

ПР69-70 Создание хранимых процедур в базах данных (различных типов)

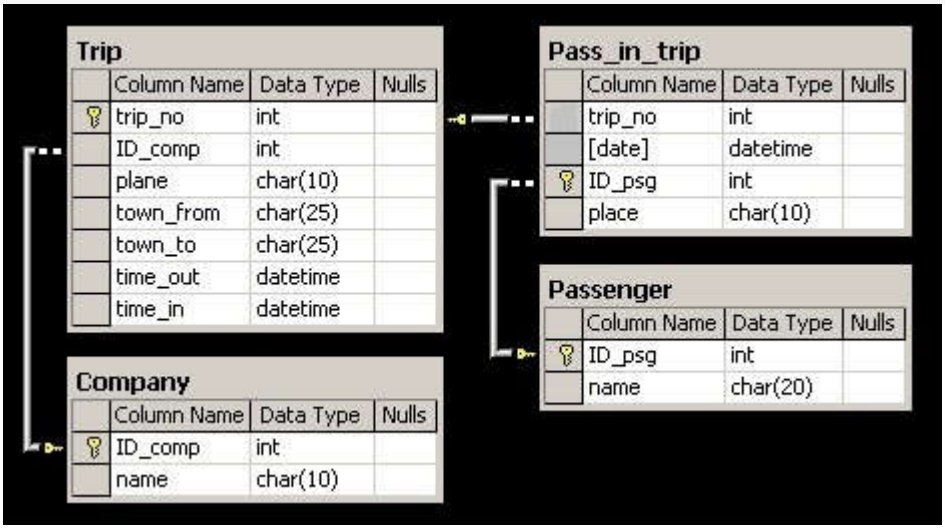
Цель: При помощи команд запросов создать следующую базу данных, состоящую из четырех таблиц.

Задание

БД АЭРОФЛОТ

Схема БД состоит из четырех отношений:
Company (ID_comp, name)
Trip(trip_no, ID_comp, plane, town_from, town_to, time_out, time_in)
Passenger(ID_psg, name)
Pass_in_trip(trip_no, date, ID_psg, place)
Таблица Company содержит идентификатор и название компании, осуществляющей перевозку пассажиров. Таблица Trip содержит информацию о рейсах: номер рейса, идентификатор компании, тип самолета, город отправления, город прибытия, время отправления и время прибытия. Таблица Passenger содержит идентификатор и имя пассажира. Таблица Pass_in_trip содержит информацию о полетах: номер рейса, дата вылета (день), идентификатор пассажира и место, на котором он сидел во время полета. При этом следует иметь в виду, что
- рейсы выполняются ежедневно, а длительность полета любого рейса менее суток; town_from <> town_to;
- время и дата учитывается относительно одного часового пояса;
- время отправления и прибытия указывается с точностью до минуты;
- среди пассажиров могут быть однофамильцы (одинаковые значения поля name, например, Bruce Willis);
- номер места в салоне – это число с буквой; число определяет номер ряда, буква (a – d) – место в ряду слева направо в

алфавитном порядке;
- связи и ограничения показаны на схеме данных.



Задание:

При помощи запросов:

1. Создать запрос, который создаст пустую таблицу (обязательно с ключевыми полями).
2. Создать 4 запроса, каждый из которых заполнит одну запись таблицы с необходимым количество значений.

Создаем базу данных Аэрофлот

а) Заполняем таблицу company тремя полями.
б) Заполняем таблицу Trip. У каждой компании должно быть по два полета. Следовательно в таблице должно быть 6 значений.
в) Заполняем таблицу Pass_In_Trip. У каждого полета должно быть по 2 пассажира. Следовательно в таблице 12 значений.
г) Заполняем 12 значений ID пассажиров.

По заданию преподавателя удалить или отредактировать один из запросов.

ПР71-72 Создание триггеров в базах данных (различных типов).

Цель: Получить навыки в работе с триггерами и операторами работы со статистическими функциями – group by, having.

Задание

Задание: Создать несколько запросов к таблице «Компьютерная фирма»

1. Создать запрос, который сгруппирует компьютеры по количеству оперативной памяти и выведет самый дешевый в своей группе.
2. Сгруппировать ноутбуки по объему жесткого диска и вывести самый быстрый из них.
3. Сгруппировать принтеры по параметру «цветной/черно-белый» и вывести количество каждого, среднюю стоимость каждой категории. При этом вывести только те принтеры, чья средняя стоимость ниже 20 000 р.

ПР73-74 Управление транзакциями

Цель: При помощи команд запросов создать следующую базу данных, состоящую из четырех таблиц.

Задание

1. Ознакомиться с интерфейсом приложения SQLiteStudio, разобраться с интерфейсом, принципом создания, редактирования структуры базы данных, запросов в формате SQl.
2. Создать базу данных «Фирма вторсырья» в формате db. Заполнить каждое поле не менее 5 значений.
3. Создать следующие запросы к ней.
   3.1 Сколько денег поступило в Income_o в текущем месяце?
   3.2 Сколько раз производилась выдача денег в outcome_o в текущем месяце?
   3.3 Укажите даты в Income, когда внесенная сумма в эти дни была более 1000 рублей (Sum).

ФИРМА ВТОРСЫРЬЯ

Фирма имеет несколько пунктов приема вторсырья. Каждый пункт получает деньги для их выдачи сдатчикам вторсырья. Code – сквозная нумерация транзакций, point – индивидуальная маркировка транзакций, inc, out – сумма поступления или выдачи денежных средств.

Сведения о получении денег на пунктах приема записываются в таблицу:

Income_o(point, date, inc)

Первичным ключом является (point, date). При этом в столбец date записывается только дата (без времени), т.е. прием денег (inc) на каждом пункте производится не чаще одного раза в день. Сведения о выдаче денег сдатчикам вторсырья записываются в таблицу:

Outcome_o(point, date, out)

В этой таблице также первичный ключ (point, date) гарантирует отчетность каждого пункта о выданных деньгах (out) не чаще одного раза в день.

В случае, когда приход и расход денег может фиксироваться несколько раз в день, используется другая схема с таблицами, имеющими первичный ключ code:

Income(code, point, date, inc)

Outcome(code, point, date, out)

Здесь также значения столбца date не содержат времени.

Income

	Column Name	Data Type	Nulls
🔑	code	int	
	point	tinyint	
	[date]	datetime	
	inc	smallmoney	

Outcome

	Column Name	Data Type	Nulls
🔑	code	int	
	point	tinyint	
	[date]	datetime	
	out	smallmoney	

Income_o

	Column Name	Data Type	Nulls
🔑	point	tinyint	
🔑	[date]	datetime	
	inc	smallmoney	

Outcome_o

	Column Name	Data Type	Nulls
🔑	point	tinyint	
🔑	[date]	datetime	
	out	smallmoney	

ПР75-76 Кеширование памяти, перехват исключительных ситуаций и обработка ошибок.

Цель: Получение практических навыков в операция кеширования памяти, перехвата исключительных ситуаций.

Задание

1. Создать структуру базы данных аэрофлот
2. Заполнить ее минимум на 3 авиакомпании, у каждой из которой должно быть минимум по два полета, в каждом из которых должно быть минимум по 2 пассажира.
3. Задать все необходимые связи Foreign Key. Для удобства необходимо ключи задать сразу после создания таблиц, и лишь после этого заполнять таблицу значениями.

БД АЭРОФЛОТ

Схема БД состоит из четырех отношений:

Company (ID_comp, name)

Trip(trip_no, ID_comp, plane, town_from, town_to, time_out, time_in)

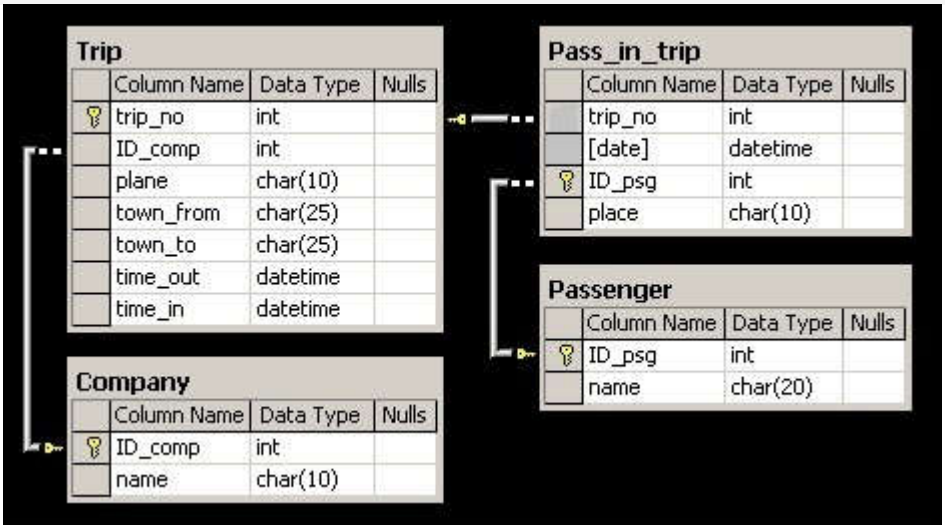
Passenger(ID_psg, name)

Pass_in_trip(trip_no, date, ID_psg, place)

Таблица Company содержит идентификатор и название компании, осуществляющей перевозку пассажиров. Таблица Trip содержит информацию о рейсах: номер рейса, идентификатор компании, тип самолета, город отправления, город прибытия, время отправления и время прибытия. Таблица Passenger содержит идентификатор и имя пассажира. Таблица Pass_in_trip содержит информацию о полетах: номер рейса, дата вылета (день), идентификатор пассажира и место, на котором он сидел во время полета. При этом следует иметь в виду, что

- рейсы выполняются ежедневно, а длительность полета любого рейса менее суток; town_from <> town_to;
- время и дата учитывается относительно одного часового пояса;
- время отправления и прибытия указывается с точностью до минуты;
- среди пассажиров могут быть однофамильцы (одинаковые значения поля name, например, Bruce Willis);
- номер места в салоне – это число с буквой; число определяет номер ряда, буква (a – d) – место в ряду слева направо в

алфавитном порядке;
- связи и ограничения показаны на схеме данных.



ПР77-78 Обеспечение достоверности информации при использовании баз данных.

Цель: При помощи запросов известных типов убедиться в достоверности передаваемых данных на примере базы данных MySQL

Задание

1. Убедиться, что в базе данных «Компьютерная фирма», созданной в MySQL, есть по 5 записей в каждом прайсе и соответственно 15 записей в таблице product.
2. Создать следующие запросы к созданной в mySQL базе данных «Компьютерная фирма»
   a. Вывести только те цены ноутбуков, аналогичных которым нет в PC (except).
   b. Вывести все существующие в продаже объемы винчестеров у PC кроме того, который по объему совпадает с самым большим объем из Laptop (except)
   c. Вывести только те объемы оперативной памяти у ноутбуков, у которых есть равнозначные среди персональных компьютеров (Intersect).
   d. Создать таблицу из самых дорогих товаров в своем прайсе. Она должна содержать модель цену, (union, подзапрос).

ПР79-82 Распределение привилегий пользователей (4 часа)

Цель: Научитьяс управлять учетными записями в БД, создавать и редактировать привилегии на основе шаблонных типов аккаунтов

Задание

Под созданием и изменением здесь подразумевается только создание и изменение структуры и параметров таблиц, а не хранящихся в них данных. Часть языка SQL, служащая для решения этих задач, называется языком описания данных (Date Definition Language – DDL).

Создание таблицы. Для создания таблицы в SQL служит команда CREATE TABLE. Синтаксис простейшего варианта команды CREATE TABLE:

CREATE TABLE <**Имя таблицы**>
(<**имя поля** > <**тип данных**>[(<**размер**>)],
<**имя поля** > <**тип данных**>[(<**размер**>] ...);

В стандартном языке SQL применяются следующие типы данных:

- INTEGER – до 10 цифр и знак;
- SMALL – до 5 цифр и знак;
- DECIMAL(p,q) – 0<p<16 всего позиций, q – цифр после запятой;
- FLOAT – вещественное, определяется СУБД (REAL в ACCESS);
- DOUBLE PRECISION – вещественное, определяется СУБД (FLOAT в ACCESS!!!), точность и диапазон больше, чем у FLOAT;
- CHAR(n) – строка из n (n<256) символов.

Практически во всех СУБД, поддерживающих SQL, применяются дополнительно следующие типы данных:

- VARCHAR(n) – строка из n символов (n_{max} >4096 определяется СУБД);
- DATE – формат определяется специальной командой (по умолчанию mm/dd/yy);

- TIME – формат определяется специальной командой (по умолчанию hh.mm.ss);
- DATETIME – комбинация даты и времени;
- MONEY – денежный.

Подробнее о типах данных, поддерживаемых СУБД ACCESS, смотрите в справке ACCESS в ответе на вопрос «Типы данных SQL».

Пример команды на создание таблицы «Страна» (название, площадь, численность населения в млн чел.):

```
CREATE TABLE Страна
(название CHAR(60),
площадь REAL,
население REAL);
```

Самостоятельно с помощью команды CREATE TABLE создайте таблицу «Изделие» со следующими атрибутами:

- название
- цена
- вес
- дата изготовления
- фирма-производитель

Подберите соответствующие типы данных.

Внесение изменений в структуру таблицы делается с помощью команды ALTER TABLE, имеющей следующий синтаксис:

```
ALTER TABLE <имя таблицы> {ADD {COLUMN <имя поля> <тип поля>[(<размер>)]
[NOT NULL]    [CONSTRAINT <имя индекса>] |
  ALTER COLUMN <имя поля> <тип поля>[(<размер>)]|
  CONSTRAINT <описание составного индекса>} |
  DROP {COLUMN <имя поля>  | CONSTRAINT <имя индекса>} };
```

Команда, с помощью которой к таблице «Страна» добавляется поле «столица» выглядит так:

```
ALTER TABLE Страна
ADD COLUMN столица VARCHAR(40) NOT NULL UNIQUE;
```

На поле «столица» наложены 2 ограничения: не допускается пустое поле (NOT NULL) и название столицы должно быть уникальным (UNIQUE).

Для добавления к таблице «Страна» поля, являющегося первичным ключом, служит команда

```
ALTER TABLE Страна
ADD COLUMN Id_strana INTEGER NOT NULL PRIMARY KEY;
```

В поле Id_strana должен храниться номер записи. Приведённая выше команда *не создаёт автоматического счётчика*.

Самостоятельно с помощью команд ALTER TABLE добавьте к таблице «Изделие» следующие поля:

- Id_izdelie – номер записи и первичный ключ;
- сорт – сорт (1-й, 2-й, …), не допускается пустых полей.

Замечание. Перед выполнеием команды ALTER TABLE необходимо из таблицы «Изделие» удалить все записи, так как в имеющихся записях новые поля не могут иметь значение NULL, т.е. быть пустыми.

Создание таблицы с ограничениями столбцов и ограничениями таблицы. В качестве примера создадим таблицу «отдых» связанную с таблицей «страна». Тип связи «многие к одному». Таблица «отдых» будет иметь следующие поля:

- Id_otd – номер записи (первичный ключ);
- Id_st - внешний ключ, связывающий с таблицей «Страна»;
- курорт – название курорта;
- гостиница;
- продолжит – продолжительность отдыха в днях.

Совокупность значений полей «Id_st», «курорт» и «гостиница» должна быть уникальной, то есть, потенциальным ключом. Таким образом, в таблице «курорт» будет 2 ключа. Описанная таблица создаётся следующей командой:

```
CREATE TABLE отдых
( Id_otd INTEGER NOT NULL PRIMARY KEY,
 Id_st INTEGER REFERENCES Страна(Id_strana),
 курорт CHAR(80),
 гостиница CHAR(60),
 стоимость REAL,
 продолжительность SMALLINT,
 UNIQUE (Id_st,курорт,гостиница));
```

Самостоятельно с помощью команд CREATE TABLE создайте таблицу «поставка», связанную с таблицей «Изделие».

Таблица «Поставка» должна иметь следующие поля:

- номер записи (первичный ключ),
- внешний ключ, связывающий с таблицей «Изделие»;
- адрес клиента,
- дату поставки,
- количество товара,
- стоимость доставки.

Совокупность значений внешнего ключа, адреса клиента и даты поставки должна быть уникальной, то есть, потенциальным ключом.

После создания таблицы «Поставка» введите в неё несколько записей. Для того, чтобы убедиться в правильной работе потенциального ключа, попытайтесь ввести две записи с одинаковыми значениями внешнего ключа, адреса клиента и даты поставки.

Добавьте таблицы «Изделие» и «Поставка» к схеме данных (Меню ACCESS □□ Схема данных). ACCESS должен автоматически обнаружить связь «один ко многим», заданную при создании таблиц командой CREATE TABLE.

Создание индекса. Если таблица велика (обычно большой считается таблица, содержащая сотни тысяч записей), то для ускорения поиска в ней данных строятся индексы. Синтаксис команды для создания индекса следующий:

CREATE [UNIQUE] INDEX <имя индекса>
ON <имя таблицы> <(список полей)>

Для построения индекса по полю «название» в таблице «Страна» служит следующая команда:

CREATE UNIQUE INDEX названиеIn
ON страна (название)

Самостоятельно постройте уникальный индекс для поля «Название» таблицы «Изделие». Проверьте его действие, введя две записи с одним наименованием изделия.

ПР 83-84 Установка антивирусной защиты.

Цель: Получить первичные навыки защиты баз данных при помощи устанавливаемого антивирусного программного обеспечения

Задание

1. Ознакомиться с теоретическими положениями.
2. Загрузить утилиту db.exe интересующей Вас подсистемы.
3. Заполнить системную таблицу с именем SIIM.
4. Сформировать эталон документа.
5. Запустить программу ORD.exe (t.exe) и заполнить (создать) экземпляр документа. Ознакомиться с режимом редактирования и просмотра документа.

IY. СОДЕРЖАНИЕ ОТЧЕТА

1. Краткое описание принципов построения информационных объектов (документов) в среде ORD.

2. Описание содержимого таблиц SIIM, SIORD, SIUEK, SIAGF для разрабатываемого документа (аналогично приведенному примеру).

Список рекомендуемой литературы.

1. Фуфаев Э.В. Базы данных: учебное пособие для студентов учреждений СПО. - М.: Академия 2012
2. Базы данных. Шнырев С.Л., — НИЯУ МИФИ, 2011, — 222 с.
3. Базы данных и Delphi. Осипов Д.Л., — БХВ-Петербург, 2011, — 733 с.
4. Искусство создания базы данных в Microsoft Office Access 2007. Быкова В.В., — СФУ, 2011, — 259 с.
5. Базы данных. Щелоков С. А., — ОГУ, 2014, — 298 с
6. Распределенные базы данных. — Изд-во СКФУ, 2015, — 129 с.
7. InterBase и Delphi. Клиент-серверные базы данных. Осипов Д.Л., — ДМК Пресс, 2015, — 536 с.
8. Базы данных : Работа с формами в СУБД MS Access 2007. Абросимова М.А. сост., — УГУЭС, 2013, — 30 с.
9. Базы данных . Проектирование и создание программного приложения в СУБД MS Access. Абросимова М. А., — Уфимский государственный университет экономики и сервиса, 2014,— 55 с.