МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Кузбасский государственный технический университет имени Т. Ф. Горбачева»
Институт профессионального образования
Кафедра информационных и автоматизированных производственных систем

Составитель О.С. Семенова

ОП. 08 Основы проектирования баз данных

Методические указания к практическим работам

Рекомендовано цикловой методической комиссией
информационных систем и программирования
в качестве электронного издания для использования
в образовательном процессе

Кемерово
2018

Рецензенты:

И.В. Чичерин, к.т.н., доцент кафедры информационных и автоматизированных производственных систем

Семенова Ольга Сергеевна
Основы проектирования баз данных: методические указания к практическим работам [Электронный ресурс]: для студентов специальности СПО 09.02.07 «Информационные системы и программирование» / сост. О.С. Семенова, КузГТУ. – Электрон. издан. – Кемерово, 2018.

Приведенные методические указания к практическим работам по курсу «Основы проектирования баз данных» позволяют углубить знания, полученные в ходе аудиторных занятий; способствуют закреплению теоретических положений; развивают навыки по их практическому применению.

Практическая работа №1
Модели данных

Цель работы: Изучить модели представления данных.

Теоретические положения

В классической теории баз данных, модель данных есть формальная теория представления и обработки данных в системе управления базами данных (СУБД), которая включает, по меньшей мере, три аспекта:

1) аспект структуры: методы описания типов и логических структур данных в базе данных;

2) аспект манипуляции: методы манипулирования данными;

3) аспект целостности: методы описания и поддержки целостности базы данных.

Аспект структуры определяет, что из себя логически представляет база данных, аспект манипуляции определяет способы перехода между состояниями базы данных (то есть способы модификации данных) и способы извлечения данных из базы данных, аспект целостности определяет средства описаний корректных состояний базы данных.

К классическим моделям данных относятся: иерархическая; сетевая; реляционная. Помимо этого, в последние годы стали появляться и активно внедряться на практике следующие модели данных: постреляционная; многомерная; объектно-ориентированная. Разрабатывают также всевозможные системы, которые основаны на других моделях данных, расширяющих известные модели. К ним относят; объектно-реляционные, дедуктивно-объектно-ориентированные, семантические, концептуальные и ориентированные модели.

Иерархическая модель. Первая версия СУБД появилась в 1968г. Она содержала модель, представляющую собой упорядоченные наборы деревьев. Данная модель данных построена по принципу иерархии типов объектов (один тип объекта - главный, другие – подчиненные. Главный и подчиненные объекты связаны по типу "один ко многим". Для каждого подчиненного типа объекта имеется лишь один исходный тип объекта. Основной недостаток данной модели – достаточно длительный поиск необходимой информации.

Сетевая модель. В данной модели любой объект может быть как главным, так и подчиненным. Каждый объект имеет возможность участвовать в любом числе взаимодействий. Другими словами, любая информационная единица может иметь множество предков и множество потомков. В моделях подобного ро-

да связи заложены внутри описаний объектов. Достоинством является гибкость модели, т.е. имеется возможность повышения быстродействия системы. Недостатком является нагрузка на информационные ресурсы.

Реляционная модель данных. Свое название получила от английского термина relation, что означает «отношение». При соблюдении определенных условий отношение можно представить в виде двумерной привычной для человека таблицы.

При табличной организации данных отсутствует иерархия элементов. Строки и столбцы могут быть просмотрены в любом порядке, поэтому высока гибкость выбора любого подмножества элементов в строках и столбцах. Любая таблица в реляционной базе состоит из строк, которые называют **записями**, и столбцов, которые называют **полями**. На пересечении строк и столбцов находятся конкретные значения данных. Для каждого поля определяется множество его значений.

В реляционной модели данных применяются разделы реляционной алгебры, откуда и была заимствована соответствующая терминология.В реляционной алгебре поименованный столбец отношения называется **атрибутом**, а множество всех возможных значений конкретного атрибута – **доменом**. Строки таблицы со значениями разных атрибутов называют **кортежами**. Атрибут, значение которого однозначно идентифицирует кортежи, называется **ключевым** (или просто ключом). Так **ключевое поле** – это такое поле, значения которого в данной таблице не повторяется. В отличие от иерархической и сетевой моделей данных в реляционной отсутствует понятие группового отношения. Для отражения ассоциаций между кортежами разных отношений используется дублирование их ключей. Сложный ключ выбирается в тех случаях, когда ни одно поле таблицы однозначно не определяет запись.

Записи в таблице хранятся упорядоченными по ключу. Ключ может быть простым, состоящим из одного поля, и сложным, состоящим из нескольких полей. Сложный ключ выбирается в тех случаях, когда ни одно поле таблицы однозначно не определяет запись.

Кроме первичного ключа в таблице могут быть вторичные ключи, называемые еще внешними ключами, или индексами. **Индекс** – это поле или совокупность полей, чьи значения имеются в нескольких таблицах и которое является первичным ключом в одной из них. Значения индекса могут повторяться в некоторой таблице. Индекс обеспечивает логическую последовательность записей в таблице, а также прямой доступ к записи.

Основная масса современных БД для компьютеров являются реляционными.

Достоинства реляционной модели данных – это простота, удобство реализации на ЭВМ, наличие теоретического обоснования и возможность формирования гибкой схемы БД, которая допускает настройку при формировании запросов.Подобная модель используется, как правило, в базах данных среднего размера. При увеличении количества таблиц в базе данных снижается скорость работы с ней. Возникают также проблемы при создании систем со сложными структурами данных (например, систем автоматизации проектирования).

Объектно-ориентированные БД включают в свой состав 2 модели данных: реляционную и сетевую, и применяются при создании крупных баз данных со сложными структурами. Объектно-ориентированная и объектно-реляционная модели данных появились в результате распространения объектно-ориентированного подхода в программировании. Объектная модель данных предлагает рассматривать БД как множество объектов, обладающих свойствами инкапсуляции, наследования и т.д.

Постреляционная модель данных представляет собой расширенную реляционную модель, снимающую ограничение неделимости данных, хранящихся в записях таблиц. Постреляционная модель данных допускает многозначные поля — поля, значения которых состоят из подзначений. Набор значений многозначных полей считается самостоятельной таблицей, встроенной в основную таблицу.

Многомерная модель. Многомерные СУБД являются узкоспециализированными СУБД, предназначенными для интерактивной аналитической обработки информации. Основные свойства, присущие к этим СУБД: агрегируемость, историчность и прогнозируемость данных.

Задания к практической работе №1

1. Построить иерархическую, сетевую модель данных. Указать корневой тип, подтипы, узлы.
   а) Многоуровневая файловая система;
   б) Объектная модель Word.
2. Построить реляционную модель данных. Указать атрибуты, кортежи, степень отношения, кардинальное число, домен.
   а) Студенты КузГТУ;
   б) Книги.
3. Построить многомерную модель данных.
   а) Интенсивность движения автомобилей на перекрестках по часам суток;
   б) Добыча полезных ископаемых в РФ;

4. Построить объектно-ориентированную модель данных.
   а) ВУЗ;
   б) Школа.

Практическая работа №2
Проектирование реляционной БД. Нормализация таблиц

Цель работы: Изучить основные принципы проектирования реляционных баз данных методом нормализации.

Теоретические положения

Классическая технология проектирования реляционных баз данных связана с теорией нормализации, основанной на анализе функциональных зависимостей между атрибутами отношений. Понятие функциональной зависимости является фундаментальным в теории нормализации реляционных баз данных. Функциональные зависимости определяют устойчивые отношения между объектами и их свойствами в рассматриваемой предметной области. Именно поэтому процесс поддержки функциональных зависимостей, характерных для данной предметной области, является базовым для процесса проектирования.

Процесс проектирования с использованием декомпозиции представляет собой процесс последовательной нормализации схем отношений, при этом каждая последующая итерация соответствует нормальной форме более высокого уровня и обладает лучшими свойствами по сравнению с предыдущей.

Каждой нормальной форме соответствует некоторый определенный набор ограничений, и отношение находится в некоторой нормальной форме, если удовлетворяет свойственному ей набору ограничений.

В теории реляционных БД обычно выделяется следующая последовательность нормальных форм:

- первая нормальная форма (1NF);
- вторая нормальная форма (2NF);
- третья нормальная форма (3NF);
- нормальная форма Бойса-Кодда (BCNF);
- четвертая нормальная форма (4NF);
- пятая нормальная форма, или форма проекции-соединения (5NF или PJNF).

Основные свойства нормальных форм:

- каждая следующая нормальная форма в некотором смысле улучшает свойства предыдущей;
- при переходе к следующей нормальной форме свойства предыдущих нормальных форм сохраняются.

Объект базы данных находится в *первой нормальной форме* тогда, когда каждый ее атрибут атомарен. Атрибут атомарен тогда, когда его значение теряет смысл при перестановке любой из его частей или при любом разбиении его на части. То есть, одно поле – одно значение.

Объект базы данных находится во *второй нормальной форме* тогда, когда он находится в первой нормальной форме и при этом любой его атрибут, не входящий в состав потенциального ключа, функционально полно зависит от каждого потенциального ключа. Это правило говорит об отделении функционально полных зависимостей на отдельные структуры.

Объект базы данных находится в *третьей нормальной форме* тогда, когда он находится во второй нормальной форме и отсутствуют транзитивные зависимости не ключевых объектов от ключевых. Транзитивная зависимость– это очевидная зависимость между полями. Если полеА равно *x*, то поле Б обязательно будет равно *y*. А если полеБ равно *z*, то тогда поле С будет равно *m*. Такой зависимости между объектами быть не должно.

Отношение находится в *нормальной форме Бойса-Кодла*, если оно находится в третьей нормальной форме, и каждый детерминант отношения является возможным ключом отношений.

В большинстве случаев, достижение третьей нормальной формы, или даже формы Бойса-Кодла, считается достаточным для реальных проектов баз данных. Однако в теории нормализации существуют нормальные формы высших порядков, которые уже связаны не с функциональными зависимостями между атрибутами отношений, а отражают более тонкие вопросы семантики предметной области, и связаны с другими видами зависимостей.

Задания к практической работе №2

1. Разработайте структуру БД методом нормализации.
   а) Библиотечная система;
   б) Почтовое отделение;
   в) Автовокзал.

Практическая работа №3
Проектирование реляционной БД. ER-метод

Цель работы: Изучить основные принципы проектирования реляционной базы данных ER-методом.

Теоретические положения

При создании моделей данных может использоваться метод семантического моделирования. Семантическое моделирование основывается на значении структурных компонентов или характеристик данных, что способствует правильности их интерпретации (понимания, разъяснения). В качестве инструмента семантического моделирования используются различные варианты *диаграмм сущность-связь* (ER — Entity-Relationship) — ERD.

Существуют различные варианты отображения ERD, но все варианты диаграмм сущность-связь исходят из одной идеи — рисунок всегда нагляднее текстового описания. ER-диаграммы используют графическое изображение сущностей предметной области, их свойств (атрибутов), и взаимосвязей между сущностями.

Сущность (таблица, отношение) — это представление набора реальных или абстрактных объектов (людей, вещей, мест, событий, идей, комбинаций и т. д.), которые можно выделить в одну группу, потому что они имеют одинаковые характеристики и могут принимать участие в похожих связях. Каждая сущность должна иметь наименование, выраженное существительным в единственном числе. Каждая сущность в модели изображается в виде прямоугольника с наименованием.

Можно сказать, что Сущности представляют собой множество реальных или абстрактных вещей (людей, объектов, событий, идей и т. д.), которые имеют общие атрибуты или характеристики.

Экземпляр сущности (запись, кортеж) – это конкретный представитель данной сущности.Атрибут сущности (поле, домен) — это именованная характеристика, являющаяся некоторым свойством сущности.

Связь — это некоторая ассоциация между двумя сущностями. Одна сущность может быть связана с другой сущностью или сама с собою. Связи позволяют по одной сущности находить другие сущности, связанные с ней.

Каждая связь может иметь один из следующих типов связи:

- один-к-одному,
- многое-ко-многим,

- один-ко-многим.

Связь типа *один-к-одному* означает, что один экземпляр первой сущности (левой) связан с одним экземпляром второй сущности (правой). Связь один-к-одному чаще всего свидетельствует о том, что на самом деле мы имеем всего одну сущность, неправильно разделенную на две.

Связь типа *многие-ко-многим* означает, что каждый экземпляр первой сущности может быть связан с несколькими экземплярами второй сущности, и каждый экземпляр второй сущности может быть связан с несколькими экземплярами первой сущности. Тип связи многие-ко-многим является временным типом связи, допустимым на ранних этапах разработки модели. В дальнейшем этот тип связи должен быть заменен двумя связями типа один-ко-многим путем создания промежуточной сущности.

Связь типа *один-ко-многим* означает, что один экземпляр первой сущности (левой) связан с несколькими экземплярами второй сущности (правой). Это наиболее часто используемый тип связи. Левая сущность (со стороны «один») называется родительской, правая (со стороны «много») – дочерней.

При разработке ER-моделей необходимо обследовать предметную область (организацию, предприятие) и выявить:

1) Сущности, о которых хранятся данные в организации (предприятии), например, люди, места, идеи, события и т.д., (будут представлены в виде блоков);
2) Связи между этими сущностями (будут представлены в виде линий, соединяющих эти блоки);
3) Свойства этих сущностей (будут представлены в виде имен атрибутов в этих блоках).

Рассмотрим пример разработки информационной системы «Контингент студентов института». Для этого необходимо изучить предметную область (образовательное учреждение) и процессы, происходящие в ней.

В первую очередь требуется ознакомиться с нормативной документацией, изучить существующий документооборот, проанализироватьсвязи. В результате определяетсяцель и задачи системы и формулируется постановка задачи.

Главная задача системы – сбор и обработка информации об основных участниках учебного процесса: студентах и преподавателях, формирование необходимых печатных форм (документов), используемых преподавателями в период зачётной недели и экзаменационной сессии, генерация сводных отчётов по результатам сессии для работников дирекции. При разработке системы следует учитывать, что она основывается на документации, поступающей из приёмной комиссии, дирекции и других подразделений института. Информация об

успеваемости студентов должна накапливаться и храниться в течение всего периода обучения. В системе должен использоваться справочник специальностей и дисциплин (предметов), изучаемых студентами.

Таким образом, проектируемая система должна выполнять следующие действия:

1. Хранить информацию о студентах и их успеваемости.
2. В институтах/факультетах по определённой специальности печатать экзаменационные ведомости и другие документы.

Выделим все существительные в этих предложениях — это предполагаемые сущности и проанализируем их:

- Студент — явная сущность.
- Успеваемость — явная сущность.
- ?Институт/факультет — нужно выяснить один или несколько институтов в университете? Если несколько, то это — предполагаемая новая сущность.
- ? Специальность — нужно выяснить одна или несколько специальностей в институте/факультете? Если несколько, то это — ещё одна сущность.
- Предмет — предполагаемая сущность.

На первоначальном этапе моделирования данных информационной системы явно выделены две основные сущности: Студент и Успеваемость. Критерием успеваемости является наличие отметки о сдачи экзаменов. Сразу возникает очевидная связь между сущностями — «студент сдаёт несколько экзаменов» и «экзамены сдаются каждым студентом». Явная связь Один-ко-многим. Первый вариант диаграммы представлен на рисунке 1.

Рисунок 1 – Первый вариант ER-диаграммы

Мы знаем, что студенты учатся в институтах/факультетах, на определённой специальности и сдают экзамены по дисциплинам (предметам). Анализ предметной области показал, что студенты учатся в нескольких институтах университета по нескольким специальностях и сдают экзамены по определённому перечню предметов.

Исходя из этого, мы добавляем в ER-модель ещё несколько сущностей. В результате она будет выглядеть так, как показано на рисунке 2.

На следующей стадии проектирования модели вносим атрибуты сущностей в диаграмму (предполагаем, что атрибуты выявлены на стадии обследования объекта и при анализе аналогов существующих систем) и получаем окончательный вариант ER— диаграммы (рисунок 3).

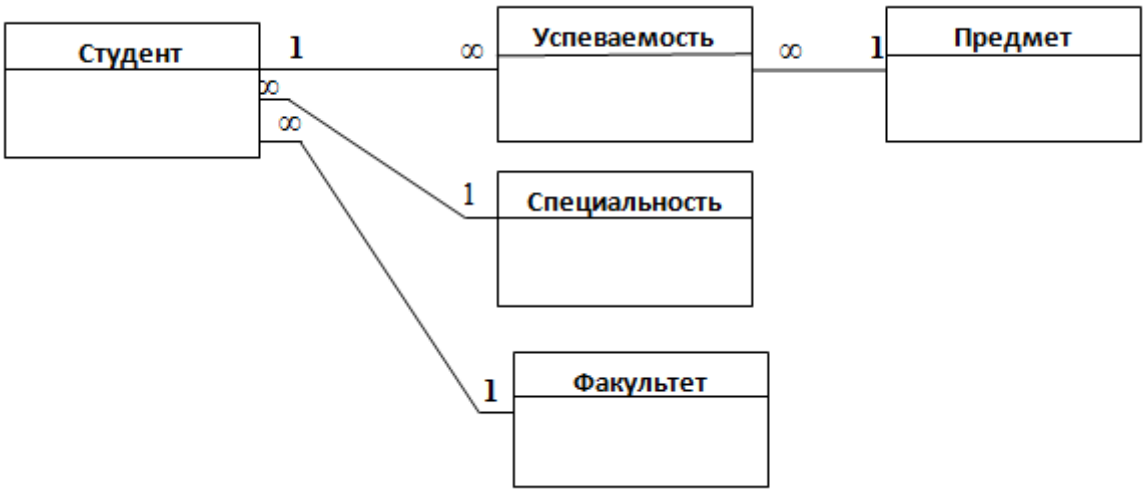

Рисунок 2 – Второй вариант ER-диаграммы

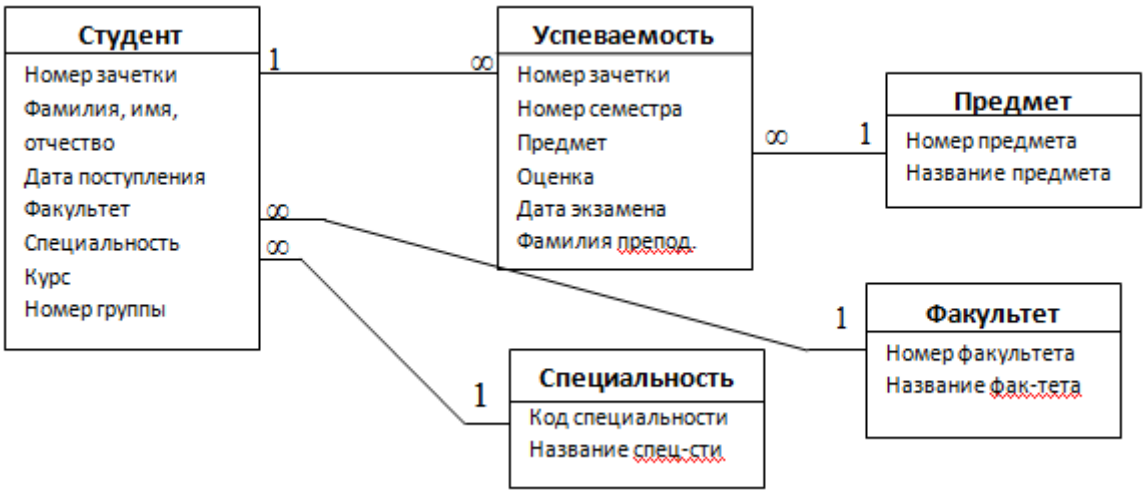

Рисунок 3 – ER-диаграмма БД

Предложенные этапы моделирования являются условными и нацелены на формирование общих представлений о процессе моделирования.

Разработанный выше пример ER-диаграммы является примером концептуальной диаграммы, не учитывающей особенности конкретной СУБД. На основе данной концептуальной диаграммы можно построить физическую диаграмму, которая будут учитывать такие особенности СУБД, как допустимые типы, наименования полей и таблиц, ограничения целостности и т.п.

Для преобразования концептуальной модели в физическую необходимо знать, что:

- каждая сущность в ER-диаграмме представляет собой таблицу базы данных.
- каждый атрибут становится полем соответствующей таблицы.

 В некоторых таблицах необходимо вставить новые атрибуты (поля), которых не было в концептуальной модели — это ключевые атрибуты родительских таблиц, перемещённыхв дочерние таблицы для того, чтобы обеспечить связь между таблицами посредством внешних ключей.

Задания к практической работе №3

1. Постройте ER-диаграмму БД в заданной предметной области.

а) Библиотечная система;
б) Почтовое отделение;
в) Автовокзал.

Практическая работа №4
Создание проекта БД. Создание БД. Редактирование и модификация таблиц

Цель работы: Научиться создавать таблицы различными способами, форматировать данные в таблицах.

Теоретические положения

Техническое задание на проектирование базы данных предоставляет заказчик.

При подготовке технического задания составляют:

- список исходных данных, с которыми работает заказчик;
- список выходных данных, которые необходимы заказчику для управления структурой своего предприятия;
- список выходных данных, которые не являются необходимыми для заказчика, но которые он должен предоставить в другие организации (в вышестоящие структуры, в органы статистического учета, прочие административные и контролирующие организации).

При этом очень важно не ограничиваться взаимодействием с головным подразделением заказчика, а провести обсуждение со всеми службами и подразделениями, которые могут оказаться поставщиками данных в базу или их потребителями.

Выяснив основную часть данных, которые заказчик потребляет или поставляет, можно приступать к созданию структуры базы, то есть структуры ее основных таблиц.

1. Работа начинается с составления генерального списка полей – он может насчитывать десятки и даже сотни позиций.
2. В соответствии с типом данных, размещаемых в каждом поле, определяют наиболее подходящий тип для каждого поля.
3. Далее распределяют поля генерального списка по базовым таблицам. На первом этапе распределение производят по функциональному признаку. Цель – обеспечить, чтобы ввод данных в одну таблицу производился, по возможности, в рамках одного подразделения, а еще лучше – на одном рабочем месте.
4. В каждой из таблиц намечают *ключевое поле.* В качестве такого выбирают поле, данные в котором повторяться не могут. Например, для таблицы данных о студентах таким поле может служить индивидуальный

шифр студента или номер зачетной книжки (см. рисунок 3). Для таблицы, в которой содержаться расписание занятий, такого поля можно и не найти, но его можно создать искусственным комбинированием полей «Время занятия» и «Номер аудитории». Эта комбинация неповторима, так как в одной аудитории в одно и то же время не принято проводить два различных занятия. Если в таблице нет полей, которые можно было бы использовать, как ключевые, всегда можно ввести дополнительное поле типа Счетчик – оно не может содержать повторяющихся данных по определению.

5. С помощью карандаша и бумаги расчерчивают связи между таблицами. Такой чертеж называется *схемой данных.* Связь между таблицами организуется на основе общего поля, причем в одной из таблиц оно обязательно должно быть ключевым.
6. Разработкой схемы данных заканчивается «бумажный» этап работы над техническим предложением. Эту схему можно согласовать с заказчиком, после чего приступать к непосредственному созданию базы данных.

Базы данных могут содержать различные объекты. Основными объектами любой базы данных являются таблицы. Простейшая база данных имеет хотя бы одну таблицу. Соответственно, структура простейшей базы данных тождественно равна структуре ее таблицы.

Структуру двумерной таблицы образуют столбцы и строки. Их аналогами в простейшей базе данных являются *поля и записи.* Если записей в таблице пока нет, значит, ее структура образована только набором полей.

Поля базы данных не просто определяют структуру базы – они еще определяют групповые свойства данных, записываемых в ячейки, принадлежащие каждому из полей. Ниже перечислены основные свойства полей таблиц баз данных на примере СУБД MicrosoftAccess.

- Имя поля – определяет, как следует обращаться к данным этого поля при автоматических операциях с базой (по умолчанию имена полей используются в качестве заголовков столбцов таблиц).
- Тип поля – определяет тип данных, которые могут содержаться в данном поле.
- Размер поля – определяет предельную длину (в символах) данных, которые могут размещаться в данном поле.
- Формат поля – определяет способ форматирования данных в ячейках, принадлежащих полю.

- Маска ввода – определяет форму, в которой вводятся данные в поле (средство автоматизации ввода данных).
- Подпись – определяет заголовок столбца таблицы для данного поля (если подпись не указана, то в качестве заголовка столбца используется свойство Имя поля).
- Значение по умолчанию – то значение, которое вводится в ячейки поля автоматически (средство автоматизации ввода данных).
- Условие на значение – ограничение, используемое для проверки правильности ввода данных (средство автоматизации ввода, которое используется, как правило, для данных, имеющих числовой тип, денежный тип или тип даты).
- Сообщение об ошибке – текстовое сообщение, которое выдается автоматически при попытке ввода в поле ошибочных данных.
- Обязательное поле – свойство, определяющее обязательность заполнения данного поля при наполнении базы.
- Пустые строки – свойство, разрешающее ввод пустых строковых данных (от свойства Обязательное поле отличается тем, что относится не ко всем типам данных, а лишь к некоторым, например к текстовым).
- Индексированное поле – если поле обладает этим свойством, все операции, связанные с поиском или сортировкой записей по значению, хранящемуся в данном поле, существенно ускоряются. Кроме того, для индексированных полей можно сделать так, что значение в записях будут проверяться по этому полю на наличие повторов, что позволяет автоматически исключить дублирование данных.

Поскольку в разных полях могут содержаться данные разного типа, то и свойства у полей могут различаться в зависимости от типа данных. Так, например, список вышеуказанных свойств полей относится в основном к полям текстового типа. Поля других типов могут иметь или не иметь эти свойства, но могут добавлять к ним и свои. Например, для данных, представляющих действительные числа, важным свойством является количество знаков после десятичной запятой. С другой стороны, для полей, используемых для хранения рисунков, звукозаписей, видео клипов и других объектов OLE, большинство вышеуказанных свойств не имеют смысла.

Таблицы баз данных, как правило, допускают работу с гораздо большим количеством разных типов данных. Так, например, базы данных

MicrosoftAccess работают со следующими типами данных.

- Текстовый – тип данных, используемый для хранения обычного неформатированного текста ограниченного размера (до 255 символов).
- Числовой – тип данных для хранения действительных чисел.
- Поле Мемо – специальный тип данных для хранения больших объемов текста (до 65 535 символов). Физически текст не хранится в поле. Он храниться в другом месте базы данных, а в поле храниться указатель на него, но для пользователя такое разделение заметно не всегда.
- Дата/время – тип данных для хранения календарных дат и текущего времени.
- Денежный – тип данных для хранения денежных сумм. Теоретически, для их записи можно было бы пользоваться и полями числового типа, но для денежных сумм есть некоторые особенности (например, связанные с правилами округления), которые делают более удобным использование специального типа данных, а не настройку числового типа.
- Счетчик – специальный тип данных для уникальных (не повторяющихся в поле) натуральных чисел с автоматическим наращиванием. Естественное использование – для порядковой нумерации записей.
- Логический - тип для хранения логических данных (могут принимать только два значения, например Да или Нет).
- Гиперссылка – специальное поле для хранения адресов URLWebобъектов Интернета. При щелчке на ссылке автоматически происходит запуск браузера и воспроизведение объекта в его окне.
- Мастер подстановок – это не специальный тип данных. Это объект, настройкой которого можно автоматизировать ввод данных в поле так, чтобы не вводить их вручную, а выбирать их из раскрывающегося списка.

Создание таблиц в СУБД Access возможно с помощью 2-х режимов:

- *Режим таблицы*

В новой версии Access появилась возможность создавать таблицу, не задумываясь о формате данных, которые вы вводите в соответствующий столбец. Программа автоматически отследит вводимую информацию и предложит соответствующий тип данных и наиболее часто используемый формат представления информации. После ввода текста в ячейку второго столбца с правой стороны появится третий столбец с наименованием «До-

бавить поле». Для удобства работы с таблицей столбцы необходимо переименовывать, для этого существует контекстное меню.

- *Режим конструктора* (рисунок 4)

При создании таблицы в режиме конструктора необходимо самостоятельно прописывать не только названия полей, но и тип данных (Текстовый, Поле MEMO, Числовой, Дата/время, Денежный, Счетчик, Логический, Поле объекта OLE, Гиперссылка, Мастер подстановок).

В окне конструктора таблиц также устанавливаются необходимые свойства полей (Размер поля, Формат поля, Маска ввода, Пустые строки, Сжатие Юникод, Подпись, Значение по умолчанию, Условие на значение, Сообщение об ошибке, Обязательное поле, Индексированное поле).

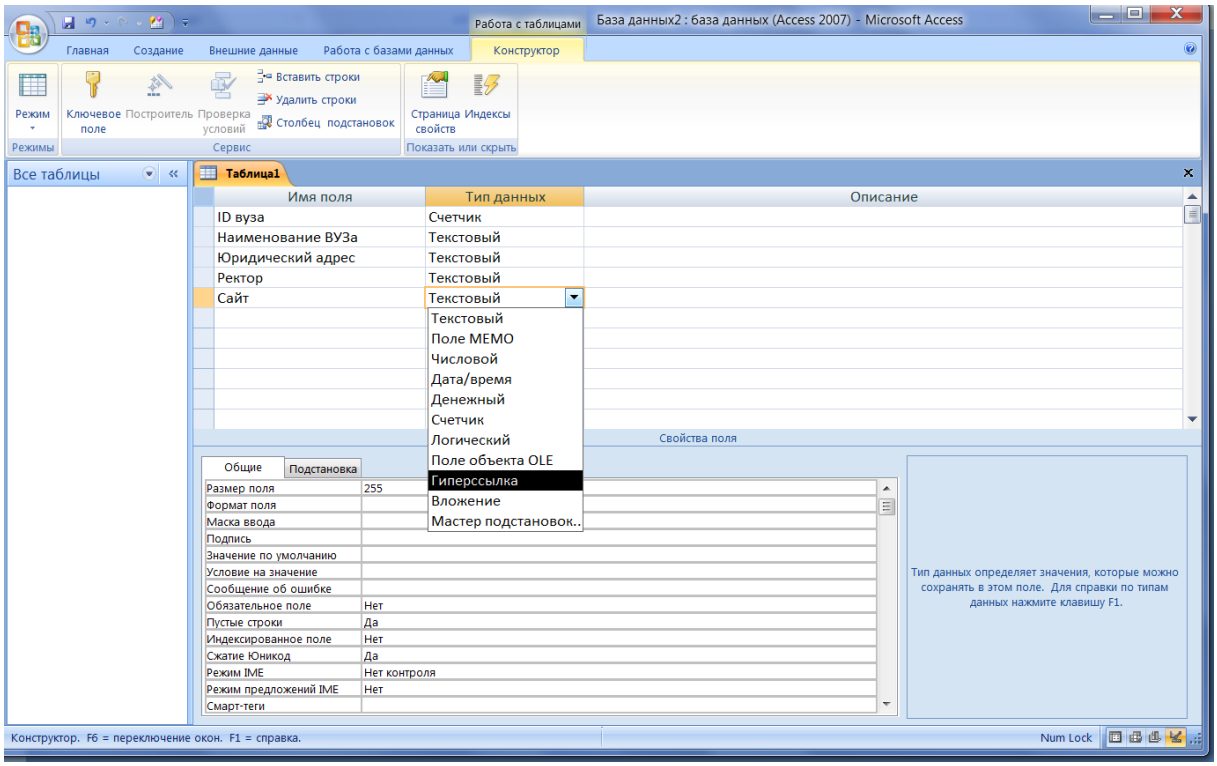

Рисунок 4 – Режим конструктора

Иногда требуется, чтобы данные в поле выводились в определенном формате, чтобы они выделялись или их было проще воспринимать. Для этого можно применять подходящие пользовательские форматы (таблица 1).

Таблица 1 – Форматирование данных в таблице

Знак	**Описание**
#	Используется для отображения цифры. Каждый экземпляр знака определяет позицию одной цифры. Если в определенной позиции нет значения, отображается пробел.

0	Используется для отображения цифры. Каждый экземпляр знака определяет позицию одной цифры. Если в определенной позиции нет значения, отображается нуль (0).
.	Определяют местоположение десятичных разделителей.
!	Используется для принудительного выравнивания всех значений по левому краю.
%	Используется в качестве последнего знака в строке формата. Умножает значение на 100 и выводит после результата символ процента.
"Текст литерала"	Заключите любой текст, который должны видеть пользователи, в двойные кавычки.
[цвет]	Используется для применения цвета ко всем значениям в части строки форматирования. Необходимо заключить имя в квадратные скобки и использовать одно из следующих имен: black, blue, cyan, green, magenta, red, yellow или white (для нерусифицированного Access). Пример: @[Красный] - *текст выводится красным цветом*
@	Используется для отображения любого символа в соответствующей позиции в строке формата. После вывода существующих данных все остальные заполнители отображаются как пробелы. Например, если используется строка формата @@@@@ и текст ABC, текст выравнивается по левому краю с двумя начальными пробелами.
<	Используется для преобразования всего текста в нижний регистр. Этот символ необходимо добавлять в начало строки формата, но перед ним может находиться восклицательный знак (!).
>	Используется для вывода всего текста прописными буквами. Этот символ необходимо добавлять в начало строки формата, но перед ним может находиться восклицательный знак (!). Пример: >@ - *текст отображается прописными буквами*

Задание к практической работе №4

1. В результате инфологического проектирования были определены следующие характеристики объектов, информация о которых должна находиться в базе данных: *Фамилия исполнителя, год рождения исполнителя, год создания группы, адрес исполнителя, E-mail исполнителя, номер сотового телефона исполнителя, E-mail группы, название группы, руководитель группы, название альбома, дата выпуска альбома, название компози-*

ций, номер композиции в альбоме по порядку, жанр композиции, фотография исполнителя, фотография обложки альбома, цена альбома.

Построить логическую модель базы данных. Выбрать оптимальный тип данных для каждого поля. Заполнить таблицы данными.

2. Настроить отображение данных с помощью пользовательского форматирования:

А) Название альбомов выделить красным цветом, а исполнителей – желтым.

Б) Для поля «Год рождения исполнителя» установить условие на значение > 1900. Установить соответствующее значение свойства «Сообщение об ошибке».

В) ФИО исполнителя перевести в верхний регистр.

Г) Если у группы нет электронного адреса (поле «E-mail группы» пустое), то выводить надпись «нет E-mail».

Д) К полю «цена альбома» автоматически добавлять слово «рублей». Цену выводить с точностью до 1 знака после запятой.

Е) Добавить логическое поле «Лицензионный» со значениями «да»/«нет».

Практическая работа №5
Создание ключевых полей. Задание индексов.
Установление и удаление связей между таблицами

Цель работы: Научиться создавать ключевые поля в таблицах, задавать индексы, устанавливать и удалять связи между таблицами.

Ключевое поле – это одно или несколько полей, комбинация значений которых однозначно определяет каждую запись в таблице. Если для таблицы определены ключевые поля, то СУБД предотвращает дублирование или ввод пустых значений в ключевое поле. Ключевые поля используются для быстрого поиска и связи данных из разных таблиц при помощи запросов, форм и отчетов.

Можно выделить три типа ключевых полей: счетчик, простой ключ и составной ключ. Рассмотрим каждый из этих типов.

Для создания ключевого поля типа Счетчик необходимо в режиме Конструктора таблиц:

1. Включить в таблицу поле счетчика.
2. Задать для него автоматическое увеличение на 1.
3. Указать это поле в качестве ключевого путем нажатия на кнопку Ключевое поле (Primary Key) на панели инструментов Конструктор таблиц (Table Design).

Если до сохранения созданной таблицы ключевые поля не были определены, то при сохранении будет выдано сообщение о создании ключевого поля. При нажатии кнопки Да (Yes) будет создано ключевое поле счетчика с именем Код (ID) и типом данных Счетчик (AutoNumber).

Для создания простого ключа достаточно иметь поле, которое содержит уникальные значения (например, коды или номера). Если выбранное поле содержит повторяющиеся или пустые значения, его нельзя определить как ключевое. Для определения записей, содержащих повторяющиеся данные, можно выполнить запрос на поиск повторяющихся записей. Если устранить повторы путем изменения значений невозможно, следует либо добавить в таблицу поле счетчика и сделать его ключевым, либо определить составной ключ.

Составной ключ необходим в случае, если невозможно гарантировать уникальность записи с помощью одного поля. Он представляет собой

комбинацию нескольких полей. Для определения составного ключа необходимо:

1. Открыть таблицу в режиме Конструктора.
2. Выделить поля, которые необходимо определить как ключевые.
3. Нажать кнопку Ключевое поле (Primary Key) на панели инструментов Конструктор таблиц (Table Design).

Чтобы удалить ключ, необходимо:

1. Открыть таблицу в режиме Конструктора.
2. Выбрать имеющееся ключевое поле (ключевые поля).
3. Нажать на кнопку Ключевое поле (Primary Key), при этом кнопка должна принять положение Выкл., а из области выделения должен исчезнуть значок (значки) ключевого поля.

Индексы представляют собой структуру, позволяющую выполнять ускоренный доступ к строкам таблицы на основе значений одного или более ее столбцов. Наличие индекса может существенно повысить скорость выполнения некоторых запросов и сократить время поиска необходимых данных за счет физического или логического их упорядочивания. Индекс – это набор ссылок, упорядоченных по определенному столбцу таблицы, который в данном случае будет называться индексированным столбцом. Хотя индекс и связан с конкретным столбцом (или столбцами) таблицы, все же он является самостоятельным объектом базы данных.

Физически индекс – всего лишь упорядоченный набор значений из индексированного столбца с указателями на места физического размещения исходных строк в структуре базы данных. Когда пользователь выполняет обращающийся к индексированному столбцу запрос, СУБД автоматически анализирует индекс для поиска требуемых значений. Однако, поскольку индексы должны обновляться системой при каждом внесении изменений в их базовую таблицу, они создают дополнительную нагрузку на систему. Индексы обычно создаются с целью удовлетворения определенных критериев поиска после того, как таблица уже находилась некоторое время в работе и увеличилась в размерах. Создание индексов не предусмотрено стандартом SQL, однако большинство диалектов поддерживают как минимум следующий оператор:

CREATE [UNIQUE] INDEX имя_индекса
ON имя_таблицы(имя_столбца[ASC|DESC][,...n])

Указанные в операторе столбцы составляют ключ индекса. Индексы могут создаваться только для базовых таблиц, но не для представлений.

Если в операторе указано ключевое слово UNIQUE, уникальность значений ключа индекса будет автоматически поддерживаться системой. Требование уникальности значений обязательно для первичных ключей, а также возможно и для других столбцов таблицы (например, для альтернативных ключей). Хотя создание индекса допускается в любой момент, при его построении для уже заполненной данными таблицы могут возникнуть проблемы, связанные с дублированием данных в различных строках. Следовательно, уникальные индексы (по крайней мере, для первичного ключа) имеет смысл создавать непосредственно при формировании таблицы. В результате система сразу возьмет на себя контроль за уникальностью значений данных в соответствующих столбцах.

Если созданный индекс впоследствии окажется ненужным, его можно удалить с помощью оператора DROP INDEX имя_индекса.

Последний этап проектирования БД – установка связей между таблицами. На этом этапе *фактически* регистрируются связи между первичными и внешними ключами. Связи между таблицами устанавливаются на ленте «Работа с базами данных» – «Схема данных» (рисунок 5).

Ошибки при связывании полей возникают если:

1) Связываемые поля имеют различный тип данных (Исключение: поле типа счетчик всегда связывается с числовым).
2) Данные в полях противоречат друг другу (Внешний ключ содержит данные, отличные от значений первичного ключа).

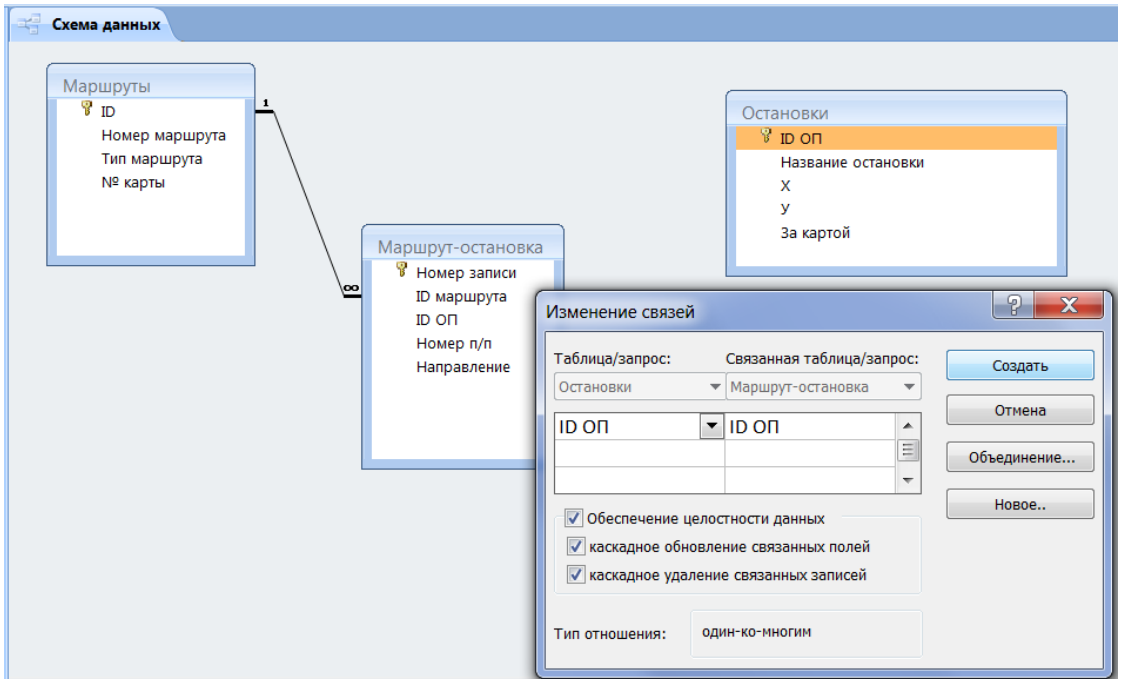

Рисунок 5 – Установка связей между таблицами

Задание к практической работе №5

1. В созданных таблицах определить первичные ключи. Связать таблицы. Проиндексировать базу данных.

Практическая работа №6
Создание формы. Управление внешним видом формы

Цель работы: Научиться создавать формы в СУБД.

Теоретические положения

Форма – это объект базы данных, который можно использовать для ввода, изменения или отображения данных из таблицы или запроса. Формы могут применяться для управления доступом к данным: с их помощью можно определять, какие поля или строки данных будут отображаться. Например, некоторым пользователям достаточно видеть лишь несколько полей большой таблицы. Если предоставить им форму, содержащую только нужные им поля, это облегчит для них использование базы данных. Для автоматизации часто выполняемых действий в форму можно добавить кнопки и другие функциональные элементы.

Формы можно рассматривать как окна, через которые пользователи могут просматривать и изменять базу данных. Рационально построенная форма ускоряет работу с базой данных, поскольку пользователям не требуется искать то, что им нужно. Внешне привлекательная форма делает работу с базой данных более приятной и эффективной, кроме того, она может помочь в предотвращении неверного ввода данных. В MicrosoftOfficeAccess 2007, 2010 предусмотрены новые средства (рисунок 6), помогающие быстро создавать формы, а также новые типы форм и функциональные возможности, благодаря которым база данных становится более практичной.

При помощи инструмента «Форма» можно создать форму одним щелчком мыши. При использовании этого средства все поля базового источника данных размещаются в форме (рисунок 6). Можно сразу же начать использование новой формы либо при необходимости изменить ее в режиме макета или конструктора.

Разделенная форма – это новая возможность в Microsoft Office Access 2007, позволяющая одновременно отображать данные в двух представлениях – в режиме формы и в режиме таблицы.

Эти два представления связаны с одним и тем же источником данных и всегда синхронизированы друг с другом. При выделении поля в одной части формы выделяется то же поле в другой части. Данные можно добавлять, изменять или удалять в каждой части формы (при условии, что

источник записей допускает обновление, а параметры формы не запрещают такие действия).

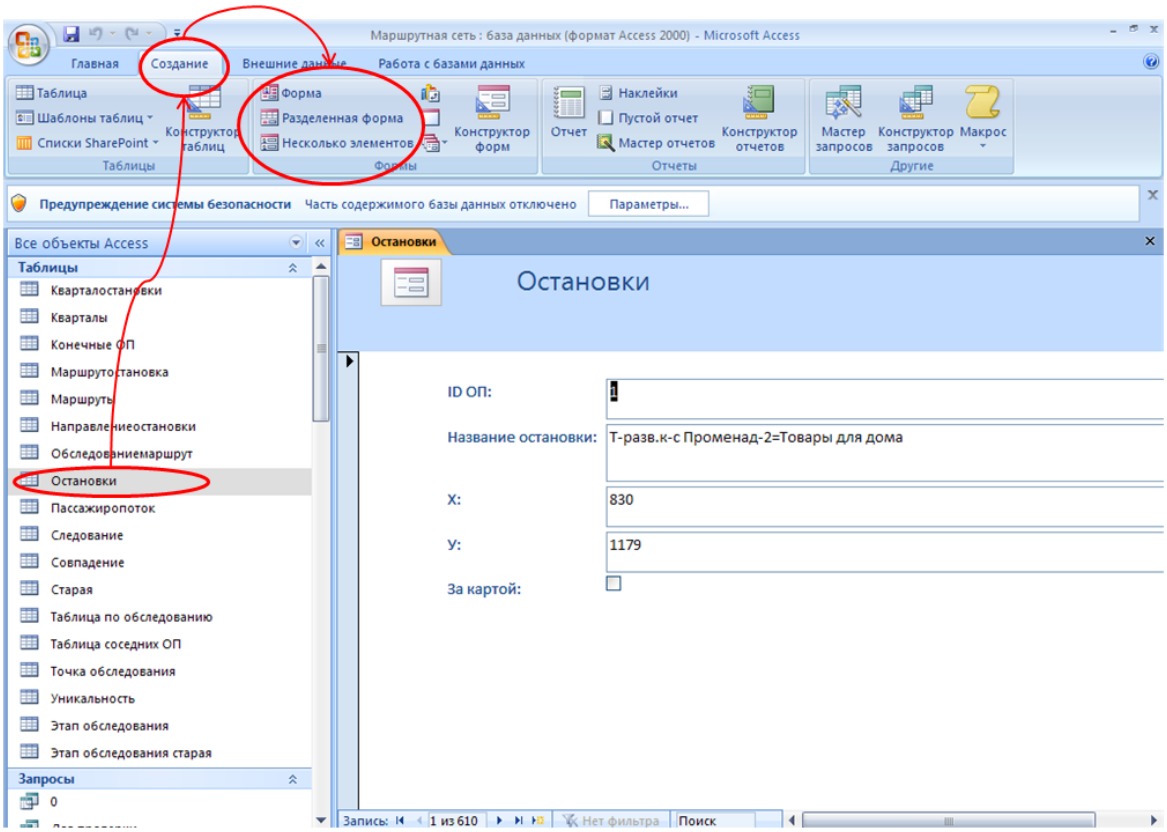

Рисунок 6 – Создание формы для ввода и просмотра информации

Работа с разделенной формой дает преимущества обоих типов формы в одной форме. Например, можно воспользоваться табличной частью формы, чтобы быстро найти запись, а затем просмотреть или изменить запись в другой части формы.

Чтобы создать разделенную форму при помощи инструмента «Разделенная форма», выполните следующие действия.

1. В области переходов щелкните таблицу или запрос с данными, которые должны отображаться в форме, или откройте таблицу или запрос в режиме таблицы.
2. На вкладке *Создание* в группе *Формы* щелкните *Разделить форму*.

Приложение Access создаст форму и отобразит ее в режиме макета. В режиме макета можно внести изменения в структуру формы при одновременном отображении данных. Например, при необходимости можно настроить размер полей в соответствии с данными.

В форме, созданной с помощью средства «Форма», одновременно отображается только одна запись. Если нужна форма, в которой отображается сразу несколько записей, и при этом требуются более широкие воз-

можности настройки, чем у таблицы, можно воспользоваться инструментом «Несколько элементов».

1. В области переходов щелкните таблицу или запрос с данными, которые должны отображаться в форме.
2. На вкладке *Создание* в группе *Формы* щелкните *Несколько элементов*.

Для получения большей свободы выбора полей, отображаемых на форме, вместо упомянутых выше инструментов можно воспользоваться мастером форм. Кроме того, можно указать способ группировки и сортировки данных, а также включить в форму поля из нескольких таблиц или запросов, при условии, что заранее заданы отношения между этими таблицами и запросами.

Если мастер или инструменты для создания форм не подходят, можно воспользоваться инструментом «Пустая форма», чтобы создать форму. Таким образом, можно очень быстро построить форму, особенно, если в ней будет лишь несколько полей.

Элементами управления называются улучшающие интерфейс пользователя объекты, которые используются для отображения данных или выполнения других действий и позволяют просматривать данные и работать с ними, например надписи и рисунки. Наиболее часто используемый элемент управления — поле. Кроме того, используются такие элементы управления, как надписи, флажки и элементы управления подчиненных форм и отчетов.

Элементы управления могут быть присоединенными, свободными и вычисляемыми.

- *Присоединенный элемент управления* – элемент управления, источником данных которого служит поле таблицы или запроса, называется присоединенным элементом управления. Присоединенный элемент управления служит для отображения значений полей базы данных. Значения могут быть текстовыми, числовыми, логическими, датами, рисунками или диаграммами. Например, для текстового поля в форме, в котором отображается фамилия служащего, могут использоваться данные поля «Фамилия» в таблице «Служащие».
- Свободные элементы управления – элементы управления, не имеющие источника данных (например, поля или выражения). Свободные элементы управления используются для вывода на экран сведений, линий,

прямоугольников и рисунков. Примером свободного элемента является надпись, которая отображает заголовок формы.

- Вычисляемые элементы управления – элементы управления, источником данных которых является выражение, а не поле. Для задания значения, которое должно содержаться в таком элементе управления, необходимо задать *выражение*, служащее источником данных элемента. Выражение – это сочетание операторов (таких как = и +), имен других элементов управления, имен полей, функций, возвращающих единственное значение, и констант. Например, в следующем выражении цена изделия рассчитывается с 25% скидкой путем умножения значения поля «Цена за единицу» на константу (0,75):

*= [Цена за единицу] * 0,75*

В выражении могут использоваться данные поля в базовой таблице или запросе формы или данные из другого элемента управления формы.

Отчеты используются для формирования выходного документа, предназначенного для вывода на печать. Отчёт удобно создавать с помощью *Мастера отчётов* (рисунок 7). Кроме *Мастера отчетов* можно использовать:

- Конструктор отчетов
- Инструмент Отчёт (Report)
- Пустой Отчёт

Отчеты целесообразно выполнять с помощью Мастера или других указанных инструментов, а дорабатывать их, то есть вносить необходимые изменения можно в режиме макета или конструктора. В Microsoft Access 2007 предусмотрено два режима внесения изменений и дополнений в отчёт: режим макета и режим конструктора.

Режим макета – это более наглядный режим редактирования и форматирования (изменения) отчетов, чем режим конструктора. В тех случаях, когда в режиме макета невозможно выполнить изменения в отчете, то целесообразно применять режим конструктора.

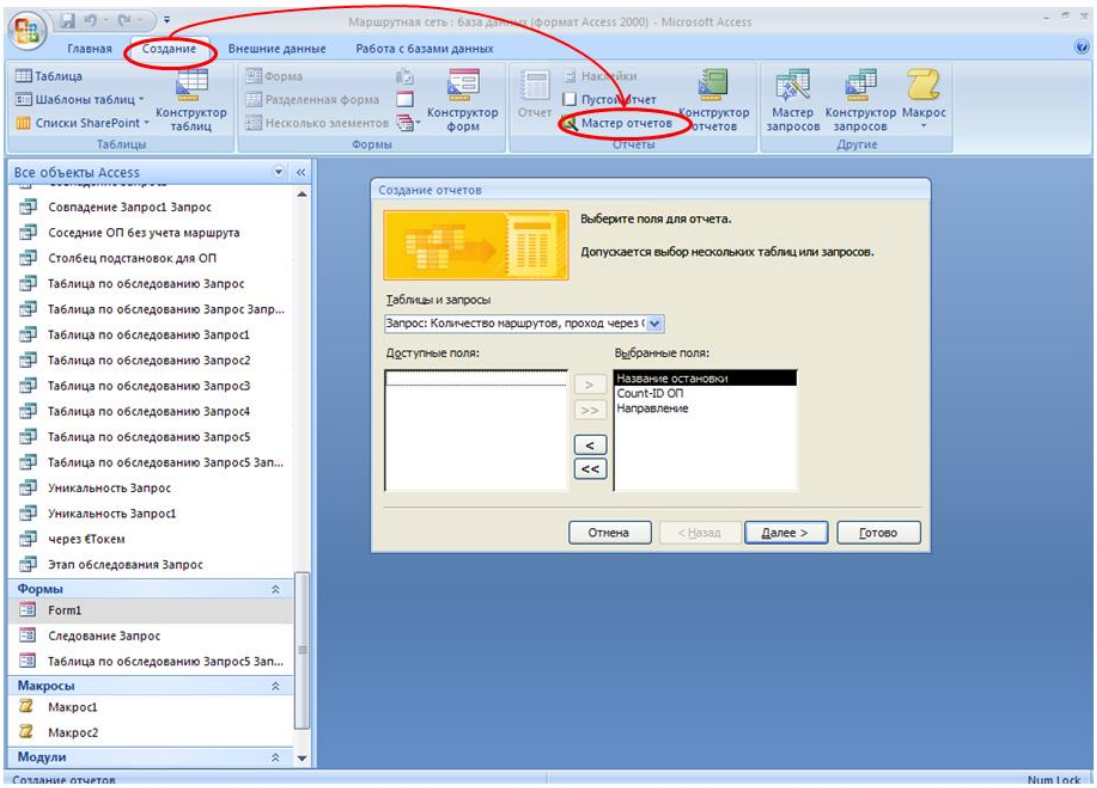

Рисунок 7 – Запуск мастера отчётов

Задание к практической работе №6

а) Создать форму. Разместить на форме кнопки:

- Вывести список исполнителей и выпущенных ими дисков
- Найти суммарный доход по каждому диску
- Найти всех исполнителей, в названии которых присутствует введенное пользователем слово
- Вывести список дисков, выпущенных в заданном году
- Вывести на экран отчет о доходах каждого исполнителя по каждому диску.

Запрограммировать кнопки.

б) Создать отчет о доходах каждого исполнителя по каждому диску.

Практическая работа №7
Создание запросов

Цель работы: Научиться создавать запросы в Access.

Теоретические положения

Запрос – это запрограммированное на специальном языке (SQL) требование к системе на выполнение некоторых действий с записями одной или нескольких таблиц.Запросы создаются пользователем для выборки нужных сведений из одной или нескольких связанных таблиц. С помощью запроса можно также обновить, удалить или добавить данные в таблицы или создать новые таблицы на основе уже существующих.

Запросы создаются с помощью:

1) мастера запросов, который работает в диалоговом режиме;
2) конструктора запросов (рисунок 8).Для этого
   - после запуска конструктора запросов автоматически появляется окно «Добавление таблицы», в котором выбирается одна или несколько таблиц, необходимых для решения поставленных целей;
   - устанавливаются связи между таблицами.
   - добавляются в запрос необходимые поля;
   - устанавливается порядок сортировки, условия и т.д.

Типы запросов:

1. *Запрос на выборку.* Позволяет выбрать записи из одной или нескольких таблиц согласно условию и представить их в табличной форме.

   Для создания запроса необходимо ввести значения следующих параметров (рисунок 8):

- «Поле» – вводится имя поля
- «Имя таблицы» – вводится имя таблицы
- «Сортировка» – указывается тип сортировки
- «Вывод на экран» – указывается, нужно ли значение поля выводить на экран
- «Условие отбора» – вводится условие для отбора данных из поля.

Для выполнения запроса необходимо на ленте выбрать команду «Выполнить». Результат выводится в виде таблицы (рисунок 9).

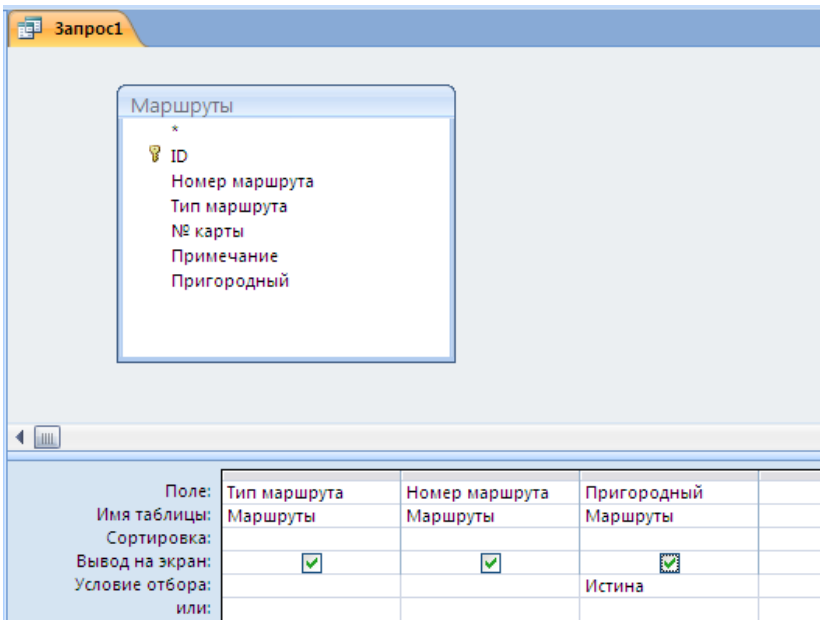

Рисунок 8 – Конструктор запросов. Запрос на выборку

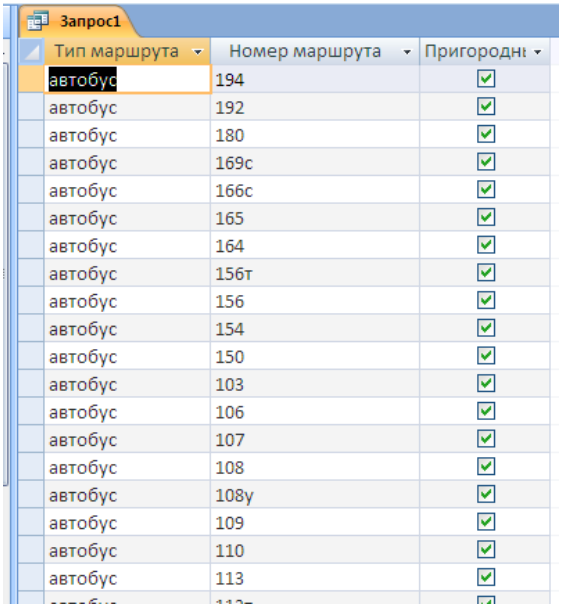

Рисунок 9 – Выборка информации из таблицы

2. *Запрос на добавление.* Добавление записей в таблицу из текущей или внешней базы данных.
3. *Запрос на обновление.* Позволяет изменить записи в одной или нескольких таблицах согласно условию.
4. *Запрос на удаление.* Позволяет удалить записи из таблиц согласно условию.
5. *Запрос на создание таблицы.* Создание таблицы из внешнего файла БД или из таблицы в текущей БД.

6. *Перекрестный запрос.* Позволяет вывести данные из таблиц в компактной форме.
7.

Задание к практической работе №7

1. Создать запросы:

- Выбрать всех исполнителей, родившихся в 1970-1980гг. Данные вывести в алфавитном порядке.
- Вывести на экран названия CD и даты их выпуска для исполнителей, родившихся в n-том году (n-вводится пользователем).
- Вывести на экран исполнителей, выпустивших в 1990 г. Диски, названия которых начинаются на букву «Б».
- Вывести на экран названия композиций, начинающихся с цифры и относящиеся к жанру «поп».
- Вывести количество дисков у каждого исполнителя.
- Вывести на экран только те композиции у групп, которые начинаются на буквы «а», «б», «в».
- Построить диаграмму количества дисков у исполнителей.
- Увеличить цену на 10% для альбомов, названия которых начинаются на букву «А».
- Добавить поле «Тираж» в таблицу «Диски». Для каждого исполнителя подсчитать доход от продаж дисков.
- Найти количество дисков у каждого исполнителя.
- Найти суммарную прибыль всех исполнителей.

Практическая работа №8
Макросы и модули

Цель работы: Отработать навыки создания макросов и модулей.

Теоретические положения

Макросы. Макрос представляет набор макрокоманд, который создается для автоматизации часто выполняемых задач. Макросы нужны пользователям для начального освоения СУБД, чтобы впоследствии перейти к программированию на VBA.

Макросы в Access 2007 создаются в специально предназначенном для этого окне Конструктора макросов (рисунок 10). Для создания макроса необходимо выбрать вкладку*Создать* и в раскрывающейся кнопке *Макрос* выбрать команду *Макрос*. Откроется окно конструктора макросов, которое состоит из панели описаний, расположенной в верхней части окна, и панели аргументов – в нижней. Панель описаний по умолчанию содержит три столбца *Макрокоманда*, *Аргументы* и *Примечание*.

В поле Макрокоманда можно выбрать необходимую команду (ОткрытьТаблицу, ОткрытьЗапрос, ОткрытьОтчет, ЗапускМакроса и т.д.) нажатием кнопкой мыши. Иначе в строку *Макрокоманда* можно просто перетащить любой объект базы данных (таблицу, форму и т.д.).

После выбора макрокоманды на панели аргументов окна конструктора макросов могут появиться строки, предназначенные для задания значений аргументов соответствующей макрокоманды. Набор строк на этой панели зависит от конкретной макрокоманды. Заданные значения аргументов также появляются во втором столбце *Аргументы* панели описаний.

Столбец *Примечание* служит для ввода комментария, который описывает выполняемое действие.

Рассмотрим простой способ создания макрокоманды. Например, разработаем ее для открытия таблицы *Группы* базы данных *Студент*.

Последовательность выполнения:

1. Вы уже открыли окно *Конструктора макросов*. Теперь с помощью мыши перетащите таблицу *Группы* из *Области переходов* в столбец *Макрокоманда*. В результате в нем появится макрокоманда Открыть таблицу, причем Access автоматически заполнит поля панели аргументов (рисунок 10).

2. В поле Режим данных задайте значение*Только чтение*, что позволит сделать записи этой таблицы недоступными для редактирования.

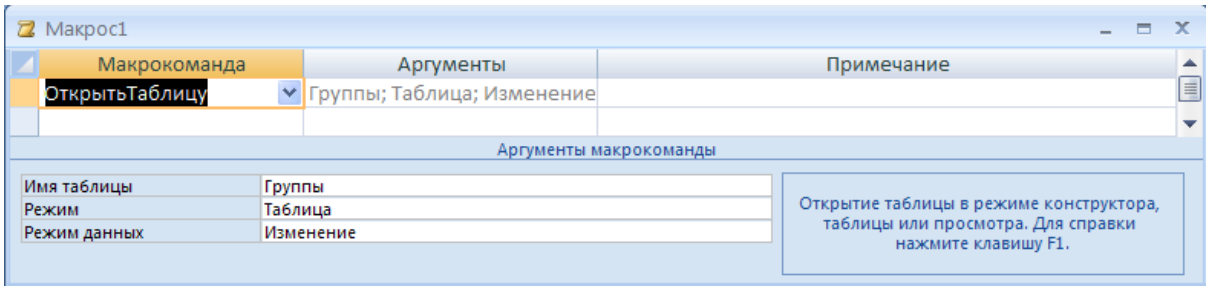

Рисунок 10 – Окно конструктора макросов

Для того чтобы ввести значение аргумента макрокоманды вы можете выбрать аргумент из списка или ввести в его поле выражение. Справа от полей некоторых аргументов расположена кнопка построителя, в окне которого вводятся различные выражения. Перед выражением нужно ставить знак равенства (=), за исключением аргумента *Выражение* макрокоманды *ЗадатьЗначение* и аргумента *Число повторов* макрокоманды *ЗапускМакроса*.

Макрос может содержать несколько макрокоманд, которые выполняются последовательно. Добавим к созданной нами макрокоманде, которая открывает таблицу Группы в режиме Только чтение, добавим макрокоманду, сообщающую пользователю о том, что таблица *Товары* доступна только для просмотра информации.

Последовательность выполнения:

1. Перейдите на следующую строку окна конструктора макросов и в столбце *Макрокоманда* выберите макрокоманду *Сообщение*.
2. В поле *Сообщение* панели аргументов введите текст «*Данныедоступнытолькодляпросмотра*». Аргумент *Сигнал* служит для определения того, будет ли вывод сообщения сопровождаться звуковым сигналом. В аргументе *Тип* выберите вид диалогового окна. Укажите, например, значение *Предупреждающее*!. В поле аргумент *Заголовок*, позволяющего задать текст заголовка окна сообщения, введите текст *«О данных»*.
3. Сохраните макрос под именем*Открыть группы*.

После запуска макроса будет отображена таблица Группы, открытая в режиме просмотра, и сообщение, представленное на рисунке 11.

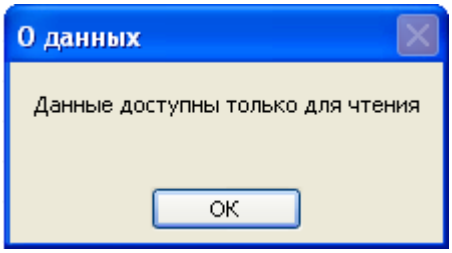

Рисунок 11 – Сообщение

Рассмотрим еще один пример. Допустим, вам необходимо создать макрос, открывающий форму *Список*.

1. Откройте окно конструктора макросов и из окна базы данных перенесите в него форму *Список*.

- Аргумент *Имя формы* содержит список всех форм базы данных.
- Аргумент *Режим* определяет режим, в котором форма должна быть открыта. Он может принимать значения: Форма, Конструктор, Просмотр, Таблица и др.
- Аргумент *Имя фильтра* применяется для отбора и/или сортировки записей в форме. Это может быть запрос или фильтр, записанный в виде запроса.
- Аргумент *Условие отбора* определяет условие для отбора записей, отображаемых в форме. Условие представляет собой выражение.
- Аргумент *Режим данных* определяет способ работы с данными и может принимать одно из значений: Добавление, Изменение или Только чтение.
- Аргумент *Режим окна* определяет тип окна: Обычное, Невидимое, Значок и Окно диалога.

2. В поле *Имяфильтра*, воспользовавшись ранее созданным фильтром *СР-11*, введите текст *СР-11*.

3. В новой строке макроса выберите макрокоманду *Сообщение*, а на панели аргументов введите текст сообщения: *Фильтр 11 группа*. Задайте тип диалогового окна вывода сообщения, выбрав значение *Информационное* аргумента *Тип*.

4. Сохраните макрос под именем *11 группа*.

При разработке приложения с использованием макросов количество макросов может оказаться очень большим. Поэтому важно хорошо организовать доступ к нужным макросам. Для этого в одном объекте Макрос можно объединить несколько макросов.

Например, рекомендуется все макросы, связанные с событиями в форме или отчете, объединить в отдельный объект, соответствующий данной форме или отчету. Для того чтобы можно было это сделать, каждый макрос группы должен иметь свое имя, а имя объекта Макрос будет являться именем группы макросов.

Последовательность выполнения:

1. Откройте *Конструктор макросов*. Нажмите кнопку *Имена макросов* на ленте с кнопками. На панели описаний в окне *Конструктора* появится еще один столбец — *Имя макроса*.

2. В столбец *Имя макроса* введите имя первого макроса «*Открытие отчета*».

3. В следующий столбец введите все макрокоманды макроса и соответствующие аргументы для каждой макрокоманды (рисунок 12).

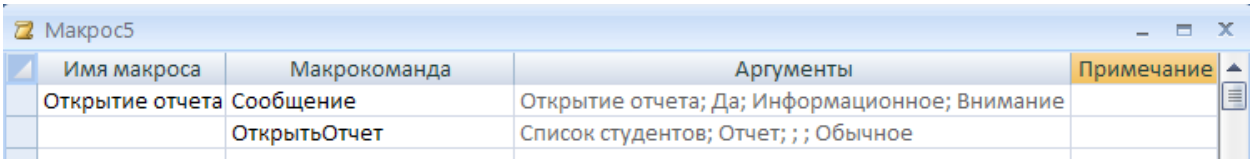

Рисунок 12 – Окно конструктора макросов

4. Пропустите одну строку.

5. Повторите шаги 2 и 3 для нового макроса (рисунок 13).

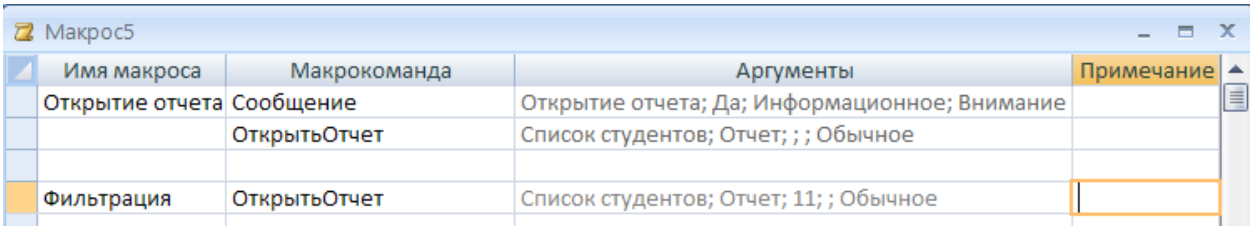

Рисунок 13 – Окно конструктора макросов

При обработке макроса пустые строки игнорируются, поэтому их удобно использовать для разделения макросов в группе.

Для запуска одного из макросов группы выберите на вкладке *Работа с базами данных* кнопку *Выполнить макрос*  . При этом откроется диалоговое окно *Запуск макроса*, в котором необходимо выбрать полное имя нужного макроса: *имяГруппы.имяМакроса*.

Универсальным способом запуска макроса является использование кнопки*Выполнить макрос* вкладки *Работа с базами данных*. В появившемся диалоговом окне нужно указать (ввести или выбрать из списка) полное имя макроса, т.е. имяГруппы.имяМакроса. Этот способ позволяет выполнить макрос из любого активного окна, будь то окно макросов, окно базы данных или, например, Конструктор форм.

Для того чтобы использовать в макросе какой-либо объект базы данных, необходимо сослаться на него. При этом вы должны знать, в какое семейство входит используемый объект. Все формы принадлежат семейству Формы, а отчеты – семейству Отчеты. Полная ссылка на форму или отчет должна состоять из двух частей.

Синтаксис:

ИмяСемейства!ИмяОбъекта

Если имя объекта содержит пробелы или специальные символы, его необходимо заключить в квадратные скобки.

Примеры

Формы![Личные данные]

Отчеты!Справка

Ссылка на свойство отчета или формы состоит из трех частей – к ссылке на объект добавляется ссылка на конкретное его свойство.

Синтаксис:

ИмяСемейства!ИмяОбъекта.ИмяСвойства

Для создания ссылки на элемент управления или его свойство необходимо указать имя этого элемента управления. Ссылка на элемент управления в форме или отчете очень схожа со ссылкой на свойство отчета или формы и состоит из трех частей.

Синтаксис:

ИмяСемейства!ИмяОбъекта!ИмяЭлемента

Пример

Формы!Список!Код

Для ссылки на свойство элемента управления необходимо к ссылке на сам элемент управления добавить имя его свойства. Такая ссылка будет состоять из четырех частей.

Синтаксис:

ИмяСемейства!ИмяОбъекта!ИмяЭлемента.ИмяСвойства

Наиболее часто макросы используются в приложении Access для обработки событий. *Событие* — это любое действие, распознаваемое объектом, и можно определить реакцию объекта на событие. События происходят в результате действий пользователя. Примером событий является вывод на экран формы, отчета, ввод данных в текстовое поле, нажатие кнопки мыши или клавиши. Каждому из этих событий можно назначить макрос, которые будут автоматически выполняться в ответ на произошедшее событие.

Пример. Откроем форму *Список* в базе данных. Допустим, мы хотели бы видеть не только информацию, представленную в этой форме, но и данные об адресе и оценках конкретного студента. Было бы хорошо создать кнопку *Адреса*, при нажатии которой появлялась бы форма *Личные данные*. Для того чтобы получить желаемое, создадим макрос, который будет выполняться, когда произойдет событие *Нажатие кнопки* в форме *Список*.

Последовательность выполнения:

1. Откройте форму *Список* в режиме *Конструктора*.

2. Создайте кнопку в области заголовка формы. На вопрос в появившемся окне ответьте *Отмена*. Откройте окно свойств только что созданной кнопки.

3. Раскройте вкладку *События*. Обратите внимание, сколько разных событий связано только с командной кнопкой. Помимо обычного нажатия, которое мы сейчас и будем использовать, в набор событий кнопки входят получение и потеря фокуса, двойной щелчок кнопкой мыши, простое перемещение указателя мыши над кнопкой и др. Такое многообразие событий дает разработчику большие возможности по созданию удобного интерфейса пользователя.

4. Найдите в списке событий *Нажатие кнопки* и установите курсор в соответствующую ячейку. В поле со списком первым его элементом будет [Процедура обработки событий], а далее идет перечень всех макросов, существующих в приложении. Поскольку нужного нам макроса в списке нет, давайте его создадим. Для этого нажмите кнопку *Построителя*, находящуюся справа от поля. Выберите *Макросы* и нажмите *ОК*. Откроется окно макросов. В макрос нужно добавить единственную макрокоманду Открыть Форму. Значения аргументов этой макрокоманды:

- *Имя формы*: Личные данные
- *Режим*: Форма
- *Условие отбора*: [КодСтудента]=[Формы]![Список]![Код]
- *Режим данных*: Только чтение
- *Режим окна*: Обычное

5. Закройте окно макроса, сохранив изменения. В окне свойств кнопки в поле Нажатие кнопки появится надпись [Внедренный макрос].

6. Раскройте вкладку *Макет* и введите в поле *Подпись* название кнопки: *Адреса*. То же самое имя рекомендуется ввести в поле *Имя* на вкладке *Другие*.

7. Теперь остается перейти в режим Формы и проверить, как макрос отрабатывает событие *Нажатие кнопки*.

Иногда необходимо, чтобы макрокоманды в макросе выполнялись только при определенных условиях. Для того, чтобы задать условие необходимо щелкнуть на кнопке Условия на линейке кнопок в окне конструктора макросов и в появившемся столбце ввести условие выполнения макрокоманды.

Макрокоманда выполняется только в том случае, если условие, заданное для нее в одноименном столбце, Истинно. Иначе она пропускается

и выполняется следующая макрокоманда. Чтобы выполнение и последующих макрокоманд зависело от указанного условия, в столбце Условие для них нужно указать многоточие (…).

Последовательность выполнения:

1. Создайте форму, представленную на рисунке 14. При создании кнопки *Просмотр* на вопрос в появившемся окне ответьте *Отмена*. Эта форма позволяет пользователю просмотреть определенный отчет, устанавливая соответствующий переключатель и нажимая кнопку *Просмотр*. Переключатели находятся внутри одного элемента управления, который называется *Группа0*.

2. Создайте макрос *мПросмотр*, содержащий условия для проверки необходимости выполнения определенных макрокоманд (рисунок 15). Этот макрос будет запускаться с помощью командной кнопки *Просмотр* в форме *фПросмотр*.

3. Откройте форму *фПросмотр* в режиме конструктора и назначьте событию *Нажатие кнопки* макрос *Просмотр*.

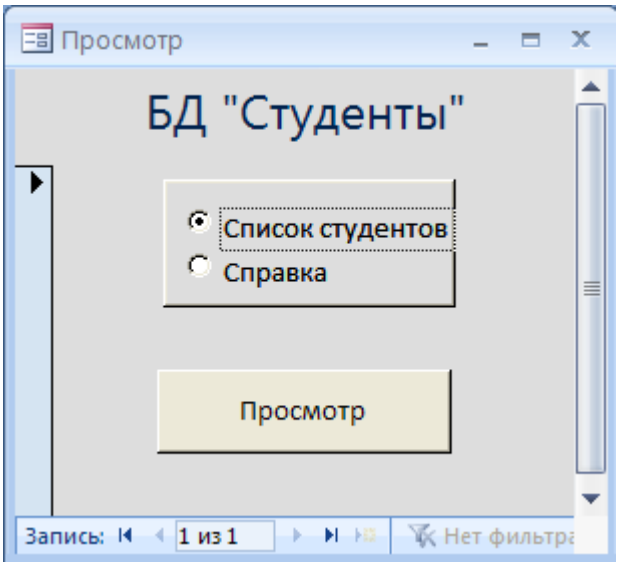

Рисунок 14 –Форма

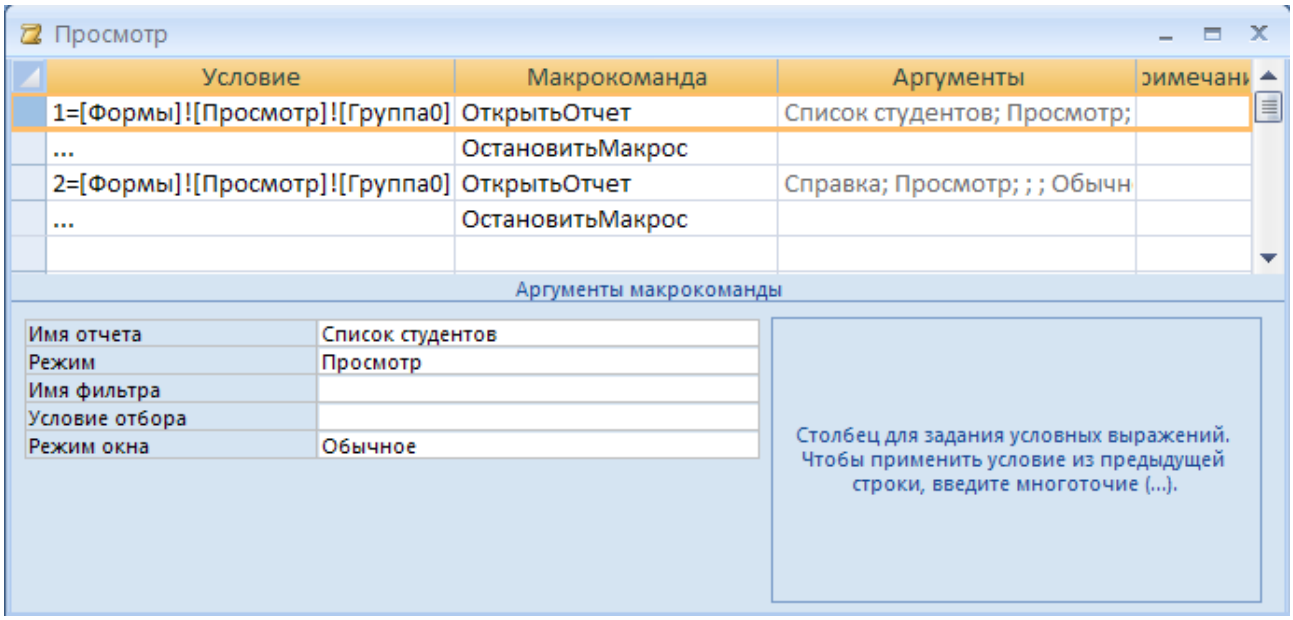

Рисунок 15 – Окно конструктора макросов

Следует заметить, что в Access 2010 появился новый конструктор макросов, упрощающий создание сложных макросов и позволяющий сократить количество ошибок при кодировании. Для этого в конструкторе применяются раскрывающиеся списки, технология *IntelliSense*, повторное использование существующих макросов, перетаскивание, а также копирование и вставка через буфер обмена.

Новый конструктор макросов для Access 2010 больше похож на текстовый редактор. Трех столбцов больше нет. Вместо них макрокоманды и условные операторы отображаются в раскрывающихся списках в привычном для программистов формате. Аргументы отображаются во встроенном диалоговом окне, как показано на рисунке 16.

Добавить новую макрокоманду или условный оператор очень просто. Макрокоманду можно выбрать в раскрывающемся списке, из контекстного меню по щелчку правой кнопкой мыши либо в области *Каталог макрокоманд*, расположенной в правой части конструктора макросов.

Можно дважды щелкнуть макрокоманду либо перетащить ее в конструктор макросов. По умолчанию в *Каталоге макрокоманд* и в поле со списком*Добавить новую макрокоманду* отображаются макрокоманды, выполняемые в базах данных, не являющихся доверенными. Чтобы вывести все макрокоманды, необходимо щелкнуть на ленте по команде*Показать все макрокоманды*.

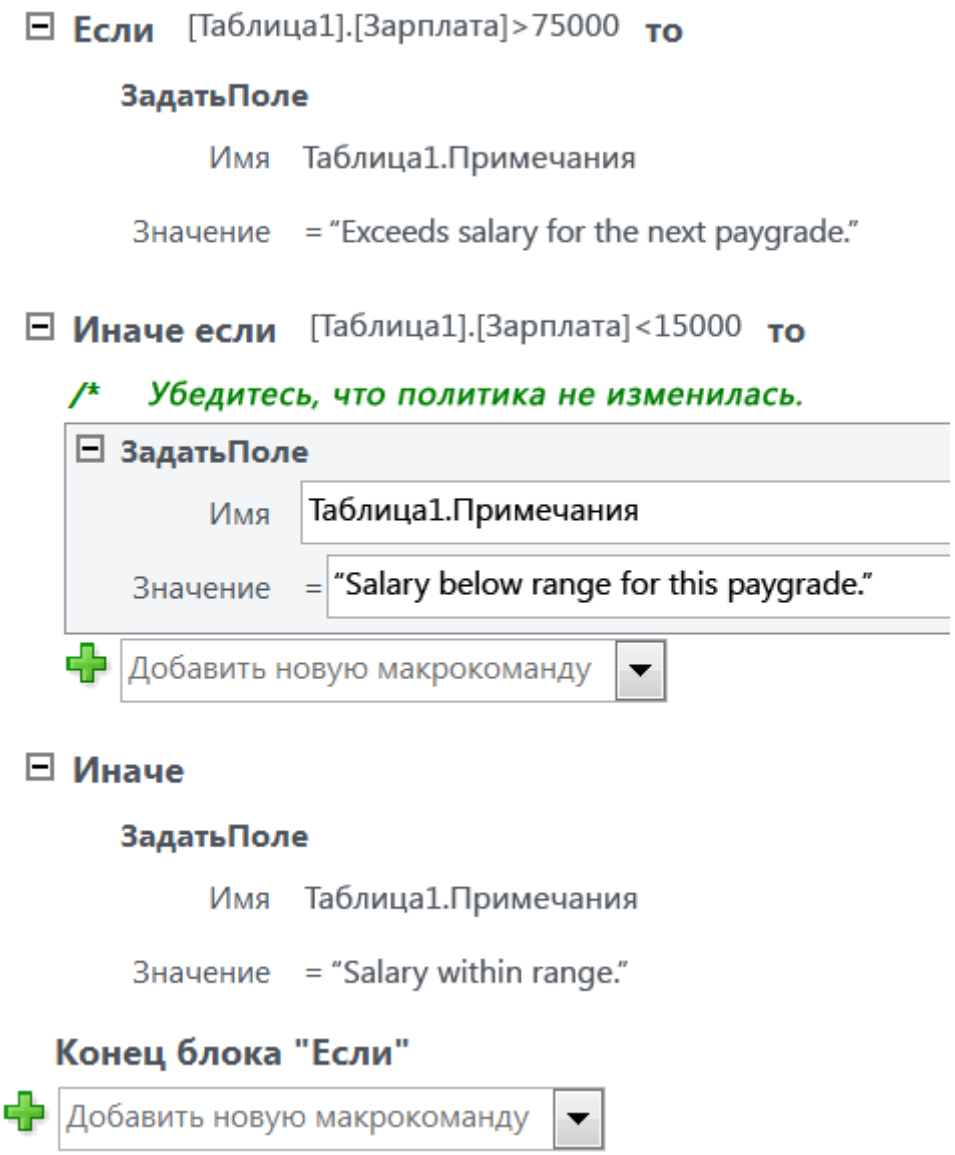

Рисунок 16 – Окно конструктора макросов в Access 2010

Модули. *Модули* являются объектами Access, такими же, как таблицы, запросы, формы и т.д. *Модуль* – это набор подпрограмм и функций, написанных на языке программирования VisualBasicforApplication (VBA) и собранных в одну программную единицу.

Существует два типа модулей: *стандартные модули* и *модули класса*. *Стандартные модули* содержат общие процедуры, которые могут использоватьсяпри обработке событий в различных объектах, для вычисления значений в формах, отчетах и т.д. Эти процедуры не связаны с конкретным объектом и могут использоваться другими приложениями Access.

Вторым типом модуля в Access является *модуль класса*. *Модуль класса* отличается от стандартного модуля тем, что кроме процедур он содержит описание объекта и используется для создания объектов. Процедуры, определенные в этом модуле, являются методами и свойствами объек-

та. Существует два типа модулей класса: *базовые модули класса, модули форм и отчетов*. Модули форм и отчетов связаны с соответствующими формами и отчетами и отличаются от базовых модулей лишь тем, что существуют в памяти только до тех пор, пока открыты связанные с ними форма или отчет.

Чтобы создать стандартный модуль или модуль класса, нужно: 1) выбрать команды *Модули –> Создать;* 2) создать необходимые процедурына языке программирования VisualBasicforApplication, 3) сохранить модуль.

Задание к практической работе №8

1. Создайте макрос для открытия формы или отчета.
2. Создайте макрос, состоящий из нескольких макрокоманд.
3. Создайте группу макросов.
4. Создайте макрос, который будет выполняться при щелчке на кнопке.
5. Создайте макрос, используя *Условие*.
6. Создайте стандартный модуль.

Практическая работа №9
Обращение к объектам БД с помощью встроенного языка программирования VBA

Цель работы: Приобрести навыки обращения к элементам базы данных (таблицам, запросам) с помощью языка программирования Visual-Basic.

Теоретические положения

Чтобы создать приложение, которое показывает, редактирует и обновляет данные из различных баз данных, включая MicrosoftAccess, dBASE. MicrosoftFoxPro или Paradox, можно воспользоваться элементом управления данными.

Его можно применить, чтобы получить доступ к MicrosoftExcel, Lotus 1-2-3 или стандартным текстовым файлам ASCII так, как если бы они были настоящими базами данных. В дополнение к тому, элемент управления данными позволяет получить доступ к удаленным базам данных ODBC и управлять ими (например, базами данных MicrosoftSQLServer или Oracle).

Элемент управления данными осуществляет доступ к данным, используя тот же процессор баз данных MicrosoftJet, что работает в MicrosoftAccess. Эта технология обеспечивает прямой доступ ко многим стандартным форматам баз данных, и позволяетписать приложения обработки данных без программирования.

Элемент управления данными может без программирования решать следующие задачи:

- связаться с локальной или удаленной базой данных;
- открыть указанную таблицу базы данных;
- передавать поля данных в связанные элементы управления, где их можновыводить на экран или менять в них значения;
- добавить новые записи или обновить базу данных
- перехватывать ошибки, возникающие при обращении к данным;
- закрыть базу данных.

Чтобы создать простое приложение базы данных необходимо:

1. Добавить элемент управления данными на форму.
2. Установить его свойства (указать базу данных и таблицу, из которых нужно получать данные).
3. Добавить связанные элементы управления (например, окна с текстом,

окна списков, и другие элементы управления, которые необходимо связать с элементом управления данных.)

4. Указать в свойствах связанных элементов управления источник данных и поля данных, которые предполагается выводить.

Элемент управления данными предлагает высокий уровень функциональности, которым можно воспользоваться, не написав ни строчки программы – просто устанавливая его свойства и управляя ими, и включая в проект связанные с данными элементы управления. Однако для расширения функций элемента управления данными в программе на VisualBasic можно самостоятельно манипулировать элементом управления данными и создаваемым им объектом *Recordset*.

Когда приложение запущено, элемент управления данными работает вместе с базой данных, предоставляя доступ к текущей совокупности записей, или набору записей (Recordset). То есть, элемент управления данными создает объект Recordset. Как и у любого объекта, у этого объекта есть свойства, которые его характеризуют, и методы (действия, которые объект может совершать). Навигация предполагает передвижение или изменение текущей записи в наборе записей. Навигация осуществляется с помощью методов (таблица 2).

Таблица 2 – Методы объекта Recordset

Метод	Описание	Пример
AddNew	Добавление новой пустой записи к таблице базы данных	Data1.Recordset.AddNew
MoveNext	Перемещение указателя записи на следующую запись в объекте Recordset	Data1.Recordset.MoveNext
MoveFirst	Перемещение указателя записи на первую запись в объекте Recordset	Data1.Recordset.MoveFirst
MoveLast	Перемещение указателя записи на последнюю запись в объекте Recordset	Data1.Recordset.MoveLast
Update	Сохранение данных в новой записи. Добавление новой записи к таблице.	Data1.Recordset.Update
Delete	Удаление текущей записи	Data1.Recordset.Delete
Close	Закрытие набора записей	Data1.Recordset.Close
Edit	Редактирование текущей записи	Data1.Recordset.Edit

Обращение к данным, находящимся в полях, осуществляется с помощью свойства Fields (таблица 3).

Таблица 3 – Свойства объекта Recordset

Свойство	Описание	Пример
Fields	Значение поля. К полю можно обращаться по имени или по номеру. Поля нумеруются, начиная с нуля.	Data3.Recordset.Fields("Направление").Value = направление **ИЛИ** ФИО = Data2.Recordset.Fields(2).Value

Eof	Признак окончания таблицы. Если достигнута последняя запись в таблице, то свойство принимает значение True, иначе – False	If Data2.Recordset.EOF = True then
		Form1.print “Вся таблица просмотрена”
		End IF

Синтаксис, применяемый для управления объектами DAODatabase, аналогичен тому, который используется для управления другими объектами VisualBasic:

Таким образом, если требуется написать программу перехода к последней записи в наборе записей, можно обратиться к набору записей, как к объекту, а затем применить к нему метод MoveLast:

Data1.Recordset.MoveLast

Если нужно считать значение определенного поля в текущей записи, нужно написать

MyString = Data1.Recordset.Fields("Title").Value

Пример 1. Вывести все записи из таблицы БД в текстовое окно

Data22.Recordset.MoveFirst

```
Do While Data22.Recordset.EOF = False
    Номер_маршрута = Data22.Recordset.Fields("Номер").Value
    Схема = Data22.Recordset.Fields(“Схема").Value
    Text1.Text=Text1.Text &Номер_маршрута&Схема
    Data22.Recordset.MoveNext
Loop
Data22.Recordset.Close
```

Пример 2. Добавить новую запись к таблице

```
Data3.Recordset.AddNew
Data3.Recordset.Fields("ID ОП").Value = ID_ОП
Data3.Recordset.Fields("ID соседнегоОП").Value = ID_ОП_соседнего Da-
ta3.Recordset.Fields("ID маршрута").Value = ID_маршрута Da-
ta3.Recordset.Fields("Направление").Value = направление Da-
ta3.Recordset.Update
```

Пример 3. Найти автора с минимальной премией

```
Data1.Recordset.MoveFirst
Min=100000000
Do While Data1.Recordset.EOF = False
    Автор = Data1.Recordset.Fields(«ФИОавтора").Value
    Премия = Data1.Recordset.Fields(“премия").Value
    IF Премия<min then min=Премия
        Найден_автор=Автор
    End If
    Data1.Recordset.MoveNext
Loop
Text1.Text=“Найден автор” & Найден_автор & “Премия=” & Премия
```

Задание к практической работе №9

Разместить на форме командные кнопки. Добавить элементы для вывода данных (например, надписи). Написать программу на языке VBA.

1. Вывести на экран список всех исполнителей.
2. Вывести на экран всю информацию об исполнителе, фамилию которого вводит пользователь.

Примечание: Можно использовать функцию Inputbox.

Синтакис: *ФИО= Inputbox (“Введи фамилию для поиска”)*

3. Добавить новую запись к таблице «Исполнители».

Примечание: Перед выполнением следующих заданий необходимо выполнить резервное копирование БД!

4. Удалить исполнителя, ФИО которого введены с клавиатуры.

5. Изменить регистр данных, находящихся в таблице «Исполнители» на верхний.

Примечание: Можно использовать функции UCase (см. справку Access)

6. Найти количество записей в таблице «Диски».

7. Найти количество дисков, выпущенных в июне.

Примечание: Можно использовать функции DatePart (см. справку Access)

8. Для всех дисков, выпущенных в июне, добавить к характеристике диска слова «Выпущен в июне».

9. Заменить в таблице «Композиции» слова «1 песня» на «One song».

Примечание: Можно использовать функции InStr, Mid (см. справку Access).

10. Вывести на экран список всех исполнителей.

11. Вывести на экран всю информацию о исполнителе, фамилию которого вводит пользователь.

Примечание: Можно использовать функцию Inputbox.

Синтакис: ФИО= Inputbox (“Введи фамилию для поиска”)

12. Найти суммарное количество дисков у всех исполнителей.

13. Используя язык SQL, в VBA сгенерировать запрос для добавления записи в таблицу «Диски» с любыми, заранее заданными значениями полей.

14. Используя язык SQL, в VBA сгенерировать параметрический запрос для добавления N записей в таблицу «Композиции». Значения параметров (N, ФИО, адрес, …) вводятся с клавиатуры.

15. Используя язык SQL, написать программу в VBA для добавления полей «пол» и «военнообязанный» (логическое/Logical) к таблице «Исполнители». Для добавления использовать конструкцию SQL “ALTER TABLE …” . Поле «пол» заполнить вручную.

16. Используя язык SQL, написать программу в VBA для изменения значения поля «военнообязанный» на «Да» для всех мужчин и на «Нет» для всех женщин.

16. Используя язык SQL, в VBA сгенерировать запрос для удаления всех дисков, выпущенных в заданном пользователем интервале дат.

Список литературы

1. Федорова, Г. Н. Основы проектирования баз данных [Текст] : учебное пособие для образовательных учреждений, реализующих программы среднего профессионального образования по специальности "Информационные системы (по отраслям)" : [для студентов СПО] / Г. Н. Федорова. – Москва : Академия, 2017. – 224 с. – Доступна электронная версия: http://www.academia-moscow.ru/catalogue/4831/296505/

2. Голицына, О. , Л. Основы проектирования баз данных. – Москва : НИЦ ИНФРА-М, 2016. – 416 с. – Режим доступа: http://znanium.com/go.php?id=552969. – Загл. с экрана. (14.12.2018)

3. Шустова, Л. И. Базы данных. – Москва : НИЦ ИНФРА-М, 2018. – 304 с. – Режим доступа: http://znanium.com/go.php?id=967755. – Загл. с экрана. (14.12.2018)

4. Цветкова, М. С. Информатика [Электронный ресурс] : учебник для использования в учебном процессе образовательных учреждений СПО на базе основного общего образования с получением среднего общего образования / М. С. Цветкова, И. Ю. Хлобыстова. – Москва : Академия, 2017. – 352 с. – Режим доступа: http://academia-moscow.ru/reader/?id=227485#copy. – Загл. с экрана. (14.12.2018)

Содержание