МЕТОДИЧЕСКИЕ УКАЗАНИЯ

для выполнения лабораторных и самостоятельных работ

по дисциплине «Базы данных»

для студентов направления:

081100.62 - Государственное и муниципальное управление»

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизации обработки информации (АОИ)

УТВЕРЖДАЮ

Зав. кафедрой АОИ
д.т.н. профессор

_____Ю.П. Ехлаков

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

для выполнения лабораторных и самостоятельных работ

по дисциплине «Базы данных»

для студентов направления:

081100.62 - Государственное и муниципальное управление»

Разработчик:
Доцент каф. АОИ
к.т.н., доцент

_____П.В. Сенченко

Томск 2012

СОДЕРЖАНИЕ

Введение

Лабораторный практикум направлен на приобретение навыков разработки баз данных, создания пользовательских запросов и элементов пользовательского интерфейса в среде СУБД MS Access, проектирования концептуальной модели предметной области в среде Power Designer, создания физической модели и структуры базы данных на ее основе.

Процесс изучения дисциплины направлен на формирование следующих компетенций:

- владение основными способами и средствами информационного взаимодействия, получения, хранения, переработки, интерпретации информации, наличием навыков работы с информационно-коммуникационными технологиями; способностью к восприятию и методическому обобщению информации, постановке цели и выбору путей ее достижения (ОК-8);
- умение обобщать и систематизировать информацию для создания баз данных, владением средствами программного обеспечения анализа и моделирования систем управления (ПК-17);
- владение технологиями защиты информации (ПК-27);
- способность осуществлять технологическое обеспечение служебной деятельности специалистов (по категориям и группам должностей государственной гражданской службы и муниципальной службы) (ПК-46).

По завершении лабораторного практикума студенты с учетом полученных теоретических знаний должны:

Уметь:

- построить концептуальную информационную модель предметной области в концепции баз данных;
- проектировать реляционную модель данных для выбранной предметной области с использованием нормализации;
- производить моделирование предметной области, уметь строить для нее ER-диаграмму и отображать ER-диаграмму в схему реляционной базы данных;
- реализовать простые информационные технологии с использованием функциональных возможностей современных СУБД;
- разрабатывать программные объекты для работы с базами данных: экранные формы, отчеты, разрабатывать все виды запросов при помощи построителей запросов.

- разрабатывать запросы на языке SQL;
- строить индексы;
- обеспечивать защиту данных средствами СУБД;

Владеть:

- методикой проектирования баз данных на основе нормализации отношений.
- средствами разработки баз данных и простых элементов пользовательского интерфейса в современных СУБД.
- методикой проектирования баз данных на основе разработки ER-модели предметной области.
- как минимум одним средством автоматизированного проектирования ER-диаграмм (Power Designer, Erwin и др.).

Проверка формирования заявленных компетенций, знаний, умений и навыков осуществляется путем защиты лабораторных работ, обоснования выбранных технических решений и способов достижения результата. Особое внимание при оценке компетенции ПК 17 уделяется защите индивидуального задания.

На проведение лабораторных работ отводится 36 аудиторных часов.

На самостоятельную подготовку, в том числе на подготовку к лабораторным занятиям, отводится 54 часа.

Лабораторная работа № 1 «Организация хранения данных в СУБД MS Access»

Тема: Организация хранения данных в СУБД MS Access. Создание таблиц, Построение схемы БД.

Раздел дисциплины: Обоснование концепции баз данных.

Цель работы: разработать структуру базы данных (БД) для выбранной предметной области, содержащую не менее пяти взаимосвязанных таблиц.

Продолжительность: 4 часа.

Организация базы данных в среде MS Access

Microsoft Access – это функционально полная реляционная СУБД. ***База данных*** в MS Access представляет собой совокупность объектов, хранящихся в одном файле с расширением mdb (рис.1).

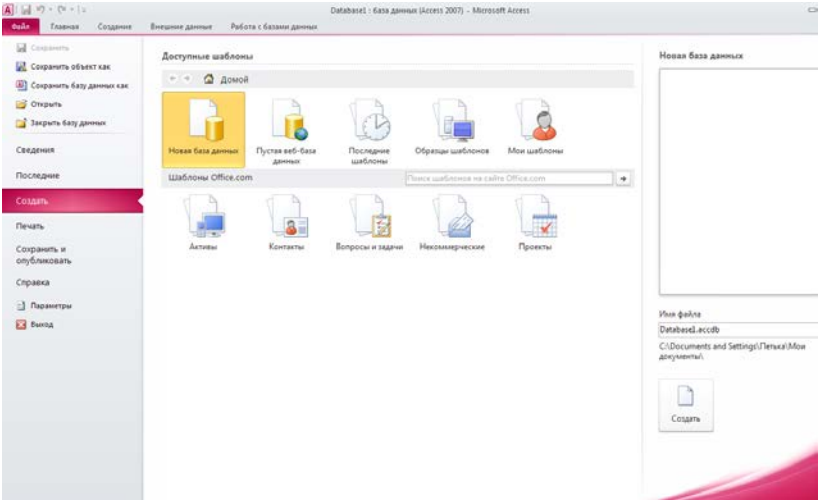

Рис. 1. Окно БД MS Access

Поддерживаются следующие типы объектов: таблицы, формы, запросы, отчеты, макросы, программные модули.

Ниже представлены характеристики БД в СУБД MS Access XP:

- размер файла базы данных Microsoft Access (.mdb) — 2 Гбайт за вычетом места, необходимого системным объектам;
- число объектов в базе данных — 768;
- модули (включая формы и отчеты, свойство Наличие модуля (HasModule) который имеет значение True) – 1 000;
- число знаков в имени объекта — 64;
- число знаков в пароле — 14;
- число знаков в имени пользователя или имени группы — 20;
- число одновременно работающих пользователей — 255;

Основным объектом в БД является ***таблица***, хранящая данные о том или ином предмете реального мира. Остальные типы объектов – это различные способы представления информации из таблиц (формы, отчеты, динамические наборы) или действия над таблицами (запросы, макросы, модули).

Запрос – это объект, позволяющий как извлекать данные из таблиц с использованием различных критериев, задаваемых пользователем, так и производить различные изменения в таблицах БД. С помощью запроса можно выбрать, изменить или сгруппировать какие-либо данные, содержащиеся в одной или нескольких таблицах. Ответ на запрос также выглядит в виде таблицы и называется ***динамическим набором записей***.

Форма – это объект, предназначенный для ввода, изменения и просмотра записей в удобном виде на экране. Форма может содержать данные из одной или нескольких взаимосвязанных таблиц, а также не связанные с таблицами данные. Для создания и изменения формы используется методика визуального программирования.

Отчет – это объект, предназначенный для печати данных в определенном пользователем виде. Отчет позволяет сгруппировать записи, производить расчеты и выводить как промежуточные, так и полные итоговые значения.

Макрос – это набор из одной или нескольких макрокоманд, позволяющих производить различные операции с объектами БД. Например, с помощью макроса при загрузке БД можно автоматически открыть нужные формы или при нажатии кнопки в форме выполнить различные действия (печать формы, открытие другой формы и т.п.) Макрокоманды выбираются из списка стандартных макрокоманд, например.

Модуль - это набор процедур и функций на языке Visual Basic. Модули обычно используют для создания достаточно сложных информационных систем. Каждый модуль может быть привязан к объектам форм и отчетам.

Каждый объект имеет структуру, характерную для его типа. Например, таблицы состоят из полей и записей. Формы и отчеты состоят из элементов управления, заголовка и др. Модули состоят из процедур и функций; макросы из макрокоманд. Многие из структурных элементов объектов также считаются объектами.

Все объекты имеют уникальные имена. Имя объекта может состоять из 64 символов, включая пробелы и другие знаки, кроме символов точка (.), восклицательный знак (!), апостроф (‘), квадратные скобки [ ]. Рекомендуется не включать в имена объектов пробелы и избегать слишком длинных имен, что затрудняет программирование приложений.

Свойство представляет собой характеристику объекта, например, имя, размер, цвет, тип данных поля и т.п. Свойства текущего объекта сведены в таблицу и доступны для изменения в окне свойств, которое открывается при нажатии кнопки  на панели инструментов. Набор свойств различен для каждого типа объектов.

Над любым объектом можно выполнить три стандартных действия (им соответствуют три кнопки в окне БД): открыть текущий объект для работы; создать новый объект текущего типа; изменить текущий объект (конструктор).

Порядок выполнения лабораторной работы

Для запуска MS Access выберите иконку  в меню программ MS Windows. Чтобы начать разработку новой базы данных, следует в меню *Файл* выбрать команду *Создать* после чего выбрать пункт *Новая база данных* и присвоить имя новой БД. Затем возможно создание объектов БД “вручную” либо с помощью Мастера, который автоматически генерирует объект в диалоге с пользователем. Независимо от способа создания объекта режим конструктора позволяет в любой момент изменить его структуру и свойства.

Создание структуры таблиц

В СУБД MS Access отношение БД называют таблицей, кортежи отношения – записями, атрибуты – полями.

Для создания структуры таблицы в окне База данных необходимо выбрать пункт Таблица и нажать кнопку Создать. В результате откроется диалоговое окно Создание таблицы, в котором следует выбрать режим Новая таблица. Создание структуры таблицы необходимо производить в режиме конструктора таблиц.

В результате выполнения указанных действий Access выводит на экран окно пустой таблицы в режиме конструктора (рис. 2).

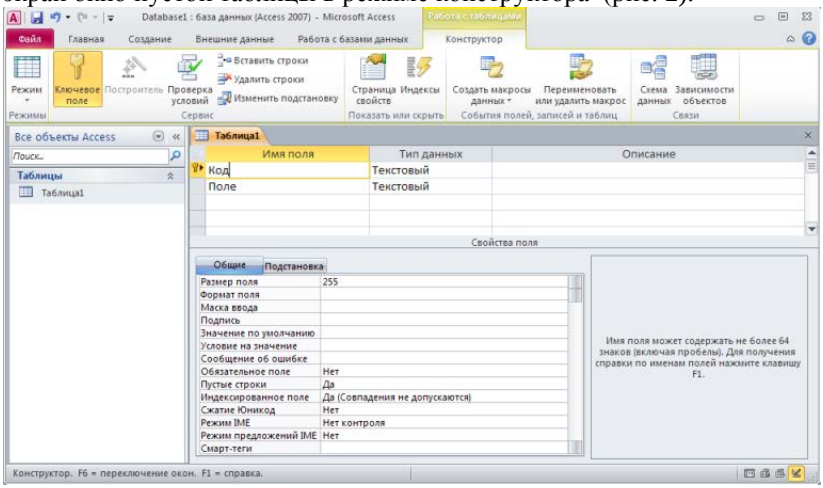

Рис. 2. Новая таблица в режиме конструктора

После того как окно таблицы откроется, активизируется панель инструментов Конструктор таблицы. При определении полей таблицы для каждого поля необходимо ввести имя, тип данных и краткое описание.

Обязательными свойствами каждого поля являются имя, тип и размер. Имя поля задается в столбце *Поле* по тем же правилам, что имена других объектов. Во втором в столбце *Тип данных* открывается список возможных типов данных. Требуемое значение типа данных можно либо выбрать из списка, либо ввести непосредственно с клавиатуры, не прибегая к помощи списка. *Тип данных* определяет, какого вида данные будут храниться в поле – текст, числа, даты и т.д. Важно правильно определить тип поля до того, как начнется ввод данных, в противном случае при изменении типа данные могут быть искажены или утеряны. Ниже приведены типы данных, используемых в СУБД MS Access:

- текстовый (до 255 символов);
- числовой с разной степенью точности;
- дата / время;
- примечания (МЕМО) - до 64000 символов;
- счетчик (для служебных полей, типа КодТовара и т.п.);
- денежный;
- логический (да / нет);
- гиперссылка
- OLE (для хранения данных, сформированных другими прикладными программами - рисунков, схем, звукозаписей, форматированных текстов и т.п.).

Для текстовых и числовых полей пользователь может задать необходимый *размер* поля, при этом следует учитывать специфику хранимых в конкретном поле данных.

Для каждого поля можно задать *дополнительные свойства* – способ отображения (формат), подпись, используемая в запросах, формах и отчетах, значение поля по умолчанию, правила контроля для ввода данных. Определение этих свойств в ряде случаев позволяет ускорить разработку прикладной программы, описать часть ограничений целостности БД, которые будут проверяться автоматически. Для типов данных *текстовый* и *мемо* может быть задан пользовательский формат ввода значений данных, описание которого приведено в разделе справочной системы Access Форматирование текста.

Описание форматов для различных типов данных представлено в таблице 1.

Описание форматов данных Таблица 1

Наименование формата	**Описание**
Для типов данных: **Числовой, Денежный**	
стандартный формат	устанавливается по умолчанию (разделители и знаки валют отсутствуют)
Денежный	символы валют и два знака после запятой
Евро	Используется денежный формат с символом евро (€) вне зависимости от символа денежной единицы
Фиксированный	выводится, по крайней мере, один разряд
с разделителями разрядов	два знака после запятой и разделители тысяч
Процентный	процент
Экспоненциальный	экспоненциальный формат (например 3.46 * 10^3)
Для типа данных **Дата/Время** существует следующий набор форматов поля:	
длинный формат	Среда, 29 января 2003 г.
средний формат	29 – янв – 03
Наименование формата	**Описание**
краткий формат	29.01.03
длинный формат времени	10:30:10 PM
средний формат времени	10:30 PM
краткий формат времени	15:30

Для *логического* типа данных используется следующий набор форматов: Да/Нет, Истина/Ложь, Вкл/Выкл.

Число десятичных знаков – для числового и денежного типов данных задает число знаков, выводимых после запятой. По умолчании устанавливается значение Авто, при котором для форматов *денежный*, *фиксированный*, *с разделителем разрядов* и *процентный* выводятся два десятичных знака после запятой. Для формата *стандартный*, число выводимых знаков определяется текущей точностью числовых значений. Можно задать фиксированное число десятичных знаков от 0 до 15.

Маска ввода – для *текстового, числового, денежного* типов данных, а так же для типов Дата/Время задается маска ввода, которую пользователь увидит при вводе данных в это поле (например, разделители (_._.__) для поля типа Дата).

Для обеспечения уникальности записей в каждой таблице необходимо наличие первичного ключа – **ключевого поля** таблицы. Общепринятые правила при определении первичного ключа:

- в качестве ключа чаще всего выбирают числовой или символьный код, который используется только для внутренних целей БД и не доступен для изменения пользователем;
- тип ключевого поля – «счетчик» или «числовой».

При необходимости первичный ключ в таблице может состоять из нескольких полей – составной первичный ключ.

Для определения первичного ключа необходимо убедиться, что курсор установлен в поле, которое будет определено как ключ (или выделить несколько полей для составного ключа), затем в меню Правка выбрать команду Ключевое поле либо нажать кнопку  на панели инструментов, в результате должен появиться значок ключа слева от имени поля.

Связи между таблицами

Связи между таблицами являются необходимым элементом структуры БД. После определения нескольких таблиц необходимо определить, как данные таблицы связаны между собой, для обеспечения полноценной работы с БД – эти связи Access будет использовать в запросах, формах и отчетах.

Для определения связей в БД необходимо в меню *Сервис* выбрать пункт *Схема данных*, либо нажать кнопку  на панели управления В результате откроется диалоговое окно Схема данных, а затем – диалоговое окно Добавление таблицы, в котором следует выбрать соединяемые таблицы.

При установлении связи необходимо помнить, что для второй (подчиненной) таблицы должен быть определен внешний ключ – поле, предназначенное для связи с главной таблицей тип данных и размер которого совпадают с полем первичного ключа главной таблицы. Например, для сопоставления сведений о товарах и оплате за проданный товар следует определить связь по полю «Код_товара» в двух таблицах: «Список товаров» (Код_товара, Наименование, Единица измерения) и «Оплата» (Код_товара, Дата_продажи, Сумма). В первой таблице общее поле является первичным ключом, а во второй – внешним ключом.

Для установления непосредственной связи между двумя выбранными таблицами следует перенести с помощью мыши ключевое поле одной таблицы в другую. В результате откроется диалоговое окно Связи (рис. 3)

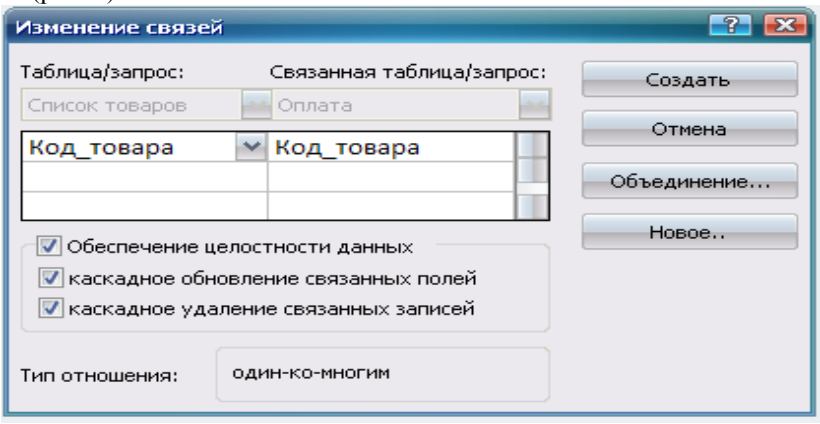

Рис. 3. Окно изменения связей между таблицами

В этом окне следует выбрать опцию Обеспечение целостности данных, после чего выбрать пункты Каскадное обновление связанных полей и Каскадное удаление связанных записей, тем самым, обеспечив требование целостности по ссылкам. Для сохранения связи нажмите кнопку Создать, в результате чего в окне Схема данных будет отображена связь между выбранными таблицами (рис. 4.).

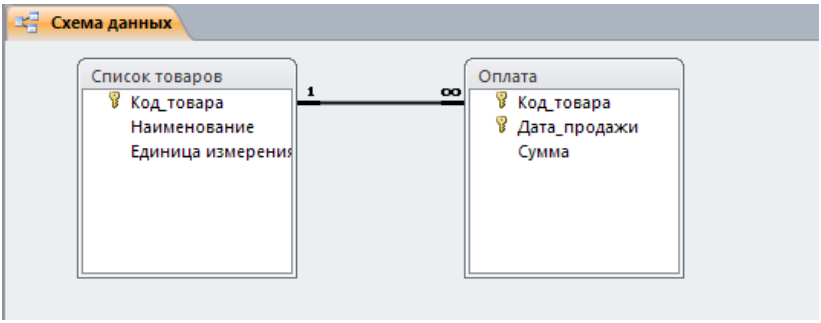

Рис. 4. Определение связей между таблицами

В СУБД MS Access допускаются следующие типы связей:

«один-ко-многим» является наиболее часто используемым типом связи между таблицами. Например, между таблицами «Список Товаров» и «Оплата» существует отношение «один-ко-многим»: товар одного наименования может продаваться различным покупателям, но каждая оплата была произведена за определенный товар.

"многие-ко-многим" реализуется только с помощью третьей таблицы, ключ которой состоит из ключевых полей тех таблиц, которые необходимо связать. Например, между таблицами «Сотрудники» и «Должности» имеется отношение «многие-ко-многим», поскольку один сотрудник может занимать несколько должностей и на одну должность может занимать несколько сотрудников, такая связь может быть реализована с помощью дополнительной таблицы «Занимаемые должности».

«один-к-одному». В этом случае каждая запись в одной таблице может быть связана только с одной записью в другой таблице и наоборот. Этот тип связи используют редко, поскольку такие данные могут быть помещены в одну таблицу. Например, такую связь используют для разделения очень больших по структуре таблиц, для отделения части таблицы по соображениям защиты и т.п.

Следует помнить, что нельзя изменить тип данных для поля, которое связывает таблицу с другой таблицей. Предварительно нужно удалить установленную связь.

После создания схемы базы данных необходимо заполнить созданные таблицы, для чего выберите нужную таблицу и нажмите кнопку Открыть в окне Базы данных.

Лабораторная работа № 2 «Создание запросов в СУБД MS Access»

Тема: Создание запросов в СУБД MS Access. Создание запросов с помощью визуального средства построителя запросов.

Раздел дисциплины: Концепция модели данных

Цель работы: создать запросы на выборку, на выборку с параметрами, на обновление записей, на удаление записей в созданных ранее таблицах.

Продолжительность: 4 часа.

Типы запросов, создаваемых в Microsoft Access

В среде MS ACCESS можно создавать следующие типы запросов:

- запросы на выборку;
- запросы с параметрами;
- перекрестные запросы;
- запросы на изменение (запросы на создание таблицы, удаление, обновление, добавление записей);
- запросы SQL (запросы на объединение, запросы к серверу, управляющие запросы, подчиненные запросы).

Наиболее часто используемым запросом является запрос на выборку, возвращающий данные из одной или нескольких таблиц, а также результаты, которые при желании пользователь может изменить. Запрос на выборку можно использовать для группировки записей, для вычисления сумм, средних значений, пересчета и других действий.

Запрос с параметрами – это запрос, содержащий в себе условие для возвращения записей или значение, которое должно содержаться в поле таблицы. Например, можно создать запрос, в результате которого выводится приглашение на ввод временного интервала. В результате будут возвращены все записи, находящиеся между двумя указанными датами.

Запросы на изменение – это запросы, при запуске которых за одну операцию вносятся изменения в несколько записей. Существует четыре типа запросов на изменение: на удаление, на обновление и добавление записей, а также на создание таблицы.

Запрос на удаление удаляет группу записей из одной или нескольких таблиц. С помощью запроса на удаление можно удалить только всю запись, а не содержимое отдельных полей внутри нее.

Запрос на удаление позволяет удалить записи как из одной таблицы, так и из нескольких таблиц со связями «один-к-одному» или с «один-ко-многим», если при определении связей было установлено каскадное удаление.

Запрос на обновление записей вносит общие изменения в группу записей одной или нескольких таблиц. Например, можно с помощью одного запроса увеличить на 30 процентов стипендию студентов 3-го курса. Запрос на обновление записей позволяет изменять данные только в существующих таблицах.

Порядок выполнения работы

Все запросы, создаваемые в рамках данной лабораторной работы, должны быть реализованы с помощью построителя запросов в режиме конструктора (рис. 5).

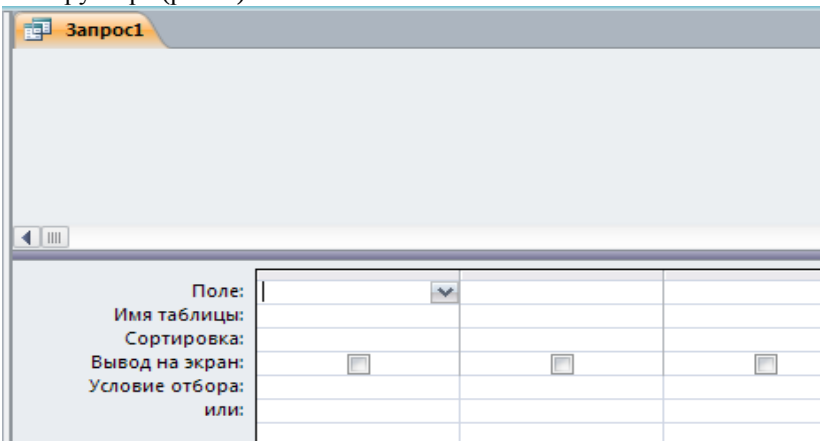

Рис. 5. Бланк построителя запросов

Для создания нового запроса в окне базы данных (рис. 1) перейдите на вкладку Запросы и нажмите кнопку Создание запроса в режиме конструктора. В появившемся окне (рис. 6) выберите таблицу (таблицы) – источник запроса. Если запрос уже открыт, то для перехода в режим конструктора следует нажать кнопку Вид  на панели инструментов.

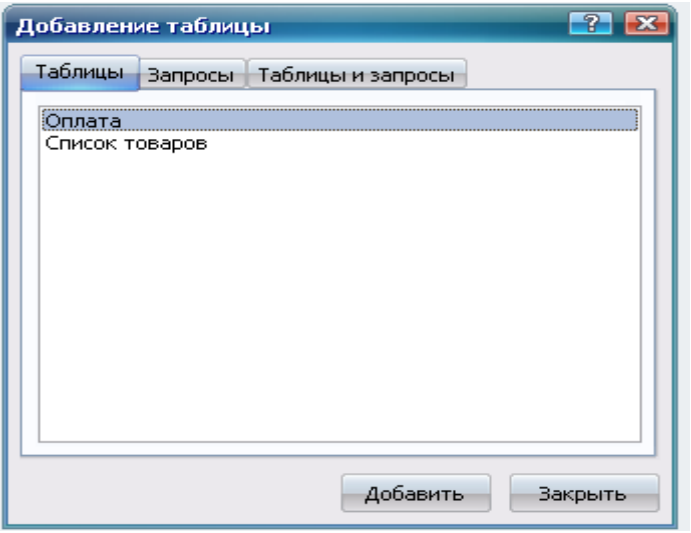

Рис. 6. Добавление таблицы

По умолчанию, любой запрос, создаваемый с помощью конструктора, является запросом на выборку. Для изменения типа запроса в меню Запрос следует выбрать тип создаваемого запроса либо на панели инструментов нажать кнопку Тип запроса .

Для сохранения запроса необходимо нажать кнопку на панели инструментов и ввести имя запроса, под которым он будет сохранен.

Создание запроса на выборку

При создании запроса на выборку, необходимо определить поля, которые будут содержаться в результирующем наборе данных, для этого в строке Имя таблицы (рис.7) выбрать название таблицы, а в строке Поле выбрать названия полей. Для сортировки записей в строке Сортировка можно указать тип сортировки: по возрастанию или по убыванию значений.

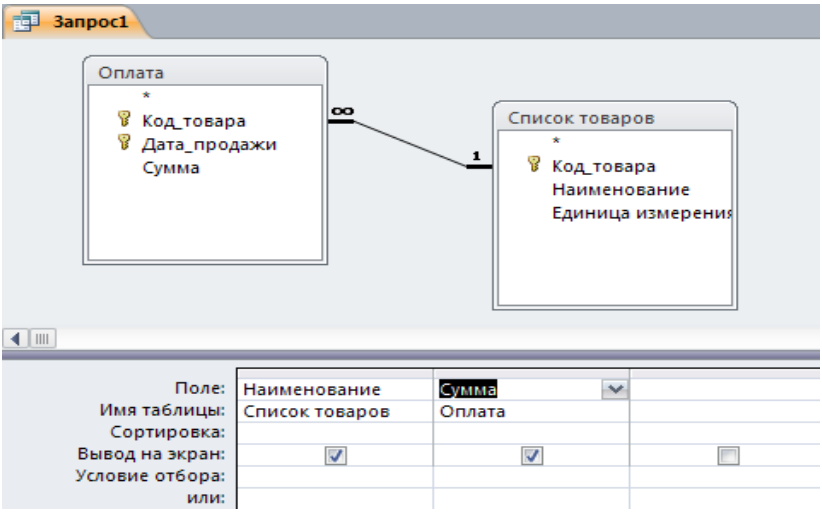

Рис. 7. Запрос на выборку

Связи между таблицами в окне построителя запросов определяются по тому же принципу, что и при определении связей в схеме данных. Для запуска запроса нажмите кнопку Запуск  на панели инструментов.

Создание запроса с параметрами

Различают два типа запросов с параметрами: с приглашением на ввод условий отбора и с явным указанием условия отбора

Запрос с параметрами отображает одно или несколько определенных диалоговых окон, выводящих приглашение пользователю ввести условия отбора.

Для создания запроса с параметрами создайте новый запрос на выборку, после чего, в режиме конструктора для каждого поля, для которого предполагается использовать параметр, введите в ячейку строки Условие отбора текст приглашения, заключенный в квадратные скобки. Это приглашение будет выводиться при запуске запроса. Текст подсказки должен отличаться от имени поля, но может включать его (рис. 8), введенное в окне приглашения значение будет являться значением параметра.

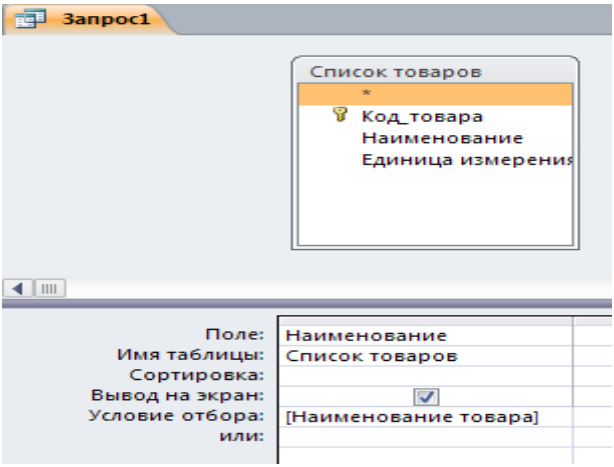

Рис. 8. Запрос на выборку с параметрами

При запуске запроса (рис.8) будет выведена подсказка Наименование товара. Для явного указания условия отбора, в построителе запроса, текстовый параметр необходимо заключить в кавычки: "Сахар", значение числового параметра указывается без дополнительных символов.

Создание запроса на обновление записей

Для создания запроса на обновление создайте запрос, выбрав таблицу или таблицы, включающие записи, которые необходимо обновить, и поля, которые должны быть использованы в условиях отбора. В режиме конструктора запроса нажмите стрелку рядом с кнопкой Тип запроса на панели инструментов и выберите команду Обновление.

Выберите поля, значения которых необходимо обновить, после чего в строку Обновление введите выражение или значение, которое должно быть использовано для изменения полей (рис. 9.)

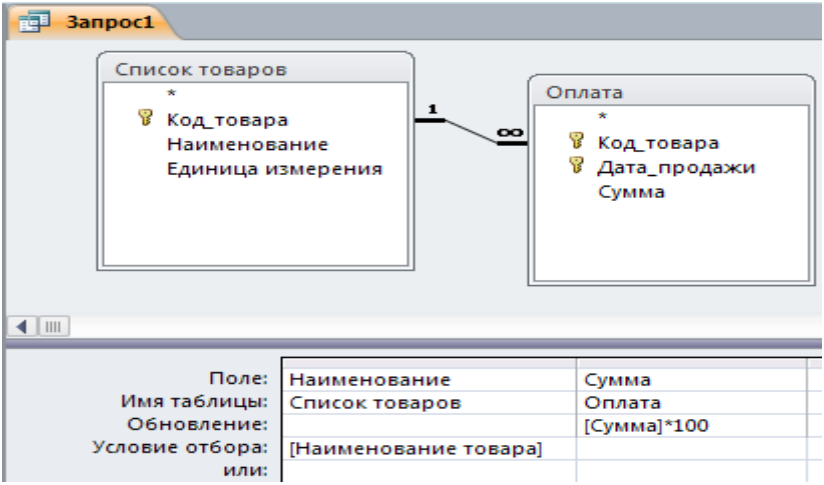

Рис. 9. Запрос на обновление

В строке Условие отбора укажите условие отбора – значения каких записей следует изменить, в противном случае значения выбранных полей будут изменены во всех записях таблицы.

Для просмотра обновляемых записей нажмите кнопку Вид на панели инструментов. Выводимый список будет содержать значения, которые будут изменены. Для возврата в режим конструктора запроса снова нажмите кнопку Вид на панели инструментов. Для запуска запроса нажмите кнопку Запуск на панели инструментов. Чтобы остановить выполнение запроса, нажмите клавиши Ctrl+Break.

Создание запроса на удаление

Для создания запроса на удаление создайте запрос, выбрав таблицу, содержащую записи, которые необходимо удалить.

В режиме конструктора запроса нажмите стрелку рядом с кнопкой Тип запроса на панели инструментов и выберите команду Удаление либо выберите тип запроса в меню Запрос.

Следует помнить, что при удалении записей с помощью запроса на удаление отменить данную операцию невозможно. Следовательно, прежде чем выполнить данный запрос, необходимо просмотреть выбранные для удаления данные. Для этого на панели инструментов нажмите кнопку Вид и просмотрите в режиме таблицы удаляемые записи.

Для избежания удаления всех записей из таблицы в условии отбора следует указать значение условия для выборки удаляемых записей.

В результате выполнения запроса на удаление будут удалены записи из подчиненных таблиц, для которых установлено Каскадное удаление связанных записей.

Лабораторная работа № 3 «Создание форм в СУБД MS Access»

Тема: Создание форм в СУБД MS Access. Создание экранных форм и их использование для ввода данных.

Раздел дисциплины: Реляционная модель.

Цель работы: создать ленточную, табличную и сложную формы в базе данных MS Access, используя в качестве источника записей созданные ранее таблицы и запросы.

Продолжительность: 4 часа.

Формы MS Access

Формы являются типом объектов базы данных, и используются для отображения и ввода данных в таблицы БД. Форму можно также использовать как кнопочную форму, открывающую другие формы или отчеты БД, а также как пользовательское диалоговое окно.

Обычно для формы выбирается источник записей. В базе данных Microsoft Access источником записей может быть таблица, запрос или инструкция SQL. В проекте Microsoft Access источником записей может быть таблица, представление, инструкция SQL или сохраненная процедура. Другие выводящиеся в форме сведения, такие как заголовок, дата и номера страниц и др., сохраняются в макете формы.

Связь между формой и ее источником записей создается при помощи графических объектов, которые называют элементами управления – это объекты графического интерфейса пользователя (такие как поле, флажок, полоса прокрутки или кнопка), позволяющий пользователям управлять приложением. Элементы управления используются для отображения данных или параметров, для выполнения действий, либо для упрощения работы с интерфейсом пользователя. Наиболее часто используемым для вывода и ввода данных типом элементов управления является поле.

Существует несколько типов форм: формы можно открывать в виде таблицы, линейной формы и в режиме простой формы. В этих режимах пользователи могут динамически изменять макет формы для изменения способа представления данных. Существует возможность упорядочивать заголовки строк и столбцов, а также применять фильтры к полям. При каждом изменении макета сводная форма немедлен-

но выполняет вычисления заново в соответствии с новым расположением данных.

Форму можно создать тремя различными способами:

– При помощи *автоформы* на основе таблицы или запроса. С помощью автоформ можно создавать формы, в которых выводятся все поля и записи базовой таблицы или запроса. Если выбранный источник записей имеет связанные таблицы или запросы, то в форме также будут присутствовать все поля и записи этих источников записей.

– При помощи *мастера* на основе одной или нескольких таблиц или запросов. Мастер задает подробные вопросы об источниках записей, полях, макете, требуемых форматах и создает форму на основании полученных ответов.

– Вручную в режиме *конструктора*. Сначала создается базовая форма, которая затем изменяется в соответствии с требованиями в режиме конструктора.

В ходе выполнения лабораторной работы должны быть созданы *ленточная, табличная и сложная (содержащая подчиненную)* формы.

Ленточная форма – это форма, в которой поля, образующие одну запись, расположены в одной строке; их подписи выводятся один раз в верхней части (заголовке) формы.

Табличная форма – это форма, в которой поля записей расположены в формате таблицы, где каждой записи соответствует одна строка, а каждому полю один столбец. Имена полей служат заголовками столбцов.

Подчиненной формой называют форму, вставленную в другую форму. Первичная форма называется *главной* формой, а форма внутри формы называется *подчиненной*. Комбинацию «форма/подчиненная форма» часто называют также иерархической формой или комбинацией «родительской» и «дочерней» форм.

Подчиненные формы особенно удобны для вывода данных из таблиц или запросов, связанных с отношением «один-ко-многим». Например, можно создать форму с подчиненной формой для вывода данных из таблицы «Типы» и из таблицы «Товары». Данные в таблице «Типы» находятся на стороне «один» отношения. Данные в таблице «Товары» находятся на стороне «многие» отношения — каждый тип может иметь несколько товаров. В главной форме отображаются данные на стороне отношения «один». В подчиненной форме отображаются данные на стороне отношения «многие».

Главная форма и подчиненная форма в этом типе форм связаны таким образом, что в подчиненной форме выводятся только те записи, которые связаны с текущей записью в главной форме. Например, когда

главная форма отображает тип «Напитки», подчиненная форма отображает только те товары, которые входят в тип «Напитки».

Порядок выполнения работы

Создание ленточной табличной и макета сложной формы следует осуществлять с помощью мастера форм. Мастер задает подробные вопросы об источниках записей, полях, макете, требуемых форматах и создает форму на основании полученных ответов.

Для создания ленточной формы с помощью мастера форм в окне базы данных перейдите на вкладку Формы и выберите пункт Создание формы с помощью мастера. В предложенном окне (рис. 10) выберите таблицу или созданный ранее запрос на выборку, который будет использоваться в качестве источника записей формы, выберите поля таблицы/запроса, которые будут доступны для редактирования в создаваемой форме. Затем нажмите кнопку Далее.

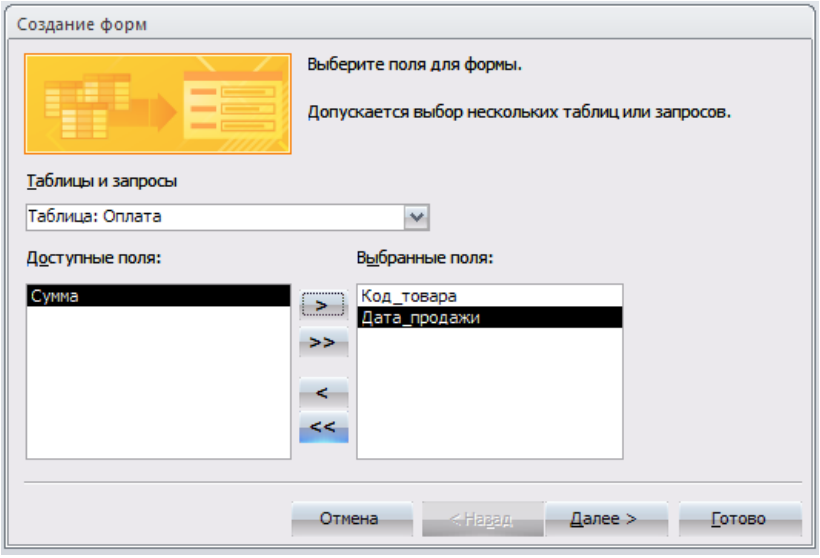

Рис. 10. Первое окно мастера форм

В следующем окне (рис. 11) выберите тип формы Ленточный, нажмите кнопку Далее и следуйте дальнейшим указаниям мастера форм.

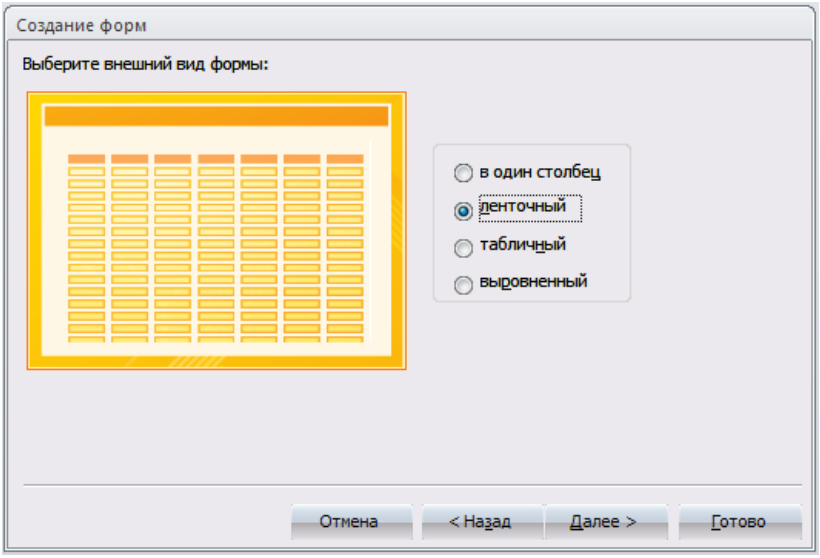

Рис. 11 Второе окно мастера форм

Для создания табличной формы с помощью мастера, следуйте предложенным выше инструкциям, а в окне (рис. 11.) выберите тип формы *табличный*. При этом рекомендуется табличную форму создавать с учетом того, что она может являться подчиненной формой в сложной форме, т.е. источником записей должна быть таблица, содержащая внешний ключ, т.е. связанная с другой таблицей М:1.

Для создания главной формы в окне (рис. 11.) выберите тип формы *в один столбец*, учитывая, что источником записей должна быть таблица, соединенная в схеме данных с подчиненной таблицей с типом связи 1:М.

Созданные формы будут отображены в окне базы данных в разделе Формы, их макет можно изменить в режиме конструктора для чего необходимо выбрать нужную форму и нажать кнопку Конструктор в окне БД.

В созданной ленточной форме создайте кнопки для навигации по записям и кнопку закрытия формы. Для этого откройте форму в режиме конструктора и в панели инструментов выберите объект Кнопка (рис. 12) и поместите её в область примечания формы.

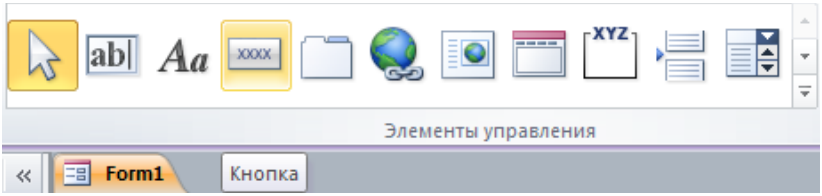

Рис. 12. Панель элементов

После этого откроется окно мастера кнопок (рис. 13) в котором необходимо выбрать категорию *переходы по записям* и в действиях – *первая запись* для перехода к первой записи набора данных, затем нажмите кнопку *Далее* и следуйте дальнейшим инструкциям мастера.

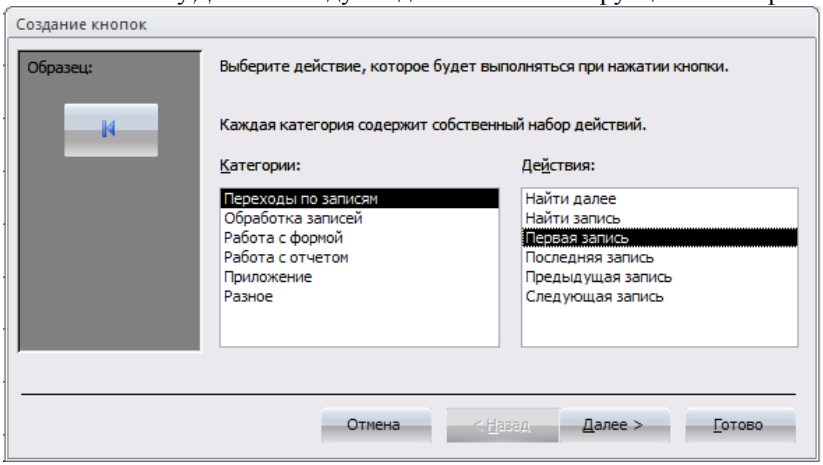

Рис. 13. Мастер кнопок

Аналогично создайте кнопки для перехода на последнюю, предыдущую и следующую запись. При создании кнопки для закрытия формы в мастере кнопок выберите категорию *работа с формой* и действие *закрытие формы*.

Создать подчиненную форму можно либо на основе имеющихся в БД таблиц и запросов либо на основе созданных ранее форм, учитывая при этом, что и главная, и подчиненная формы должны иметь общие поля, необходимые для установления связи между ними.

Для создания подчиненной формы откройте созданную ранее главную форму в режиме конструктора и в панели инструментов выберите объект подчиненная форма и вставьте его в главную форму, поле чего откроется окно мастера подчиненных форм (рис. 14)

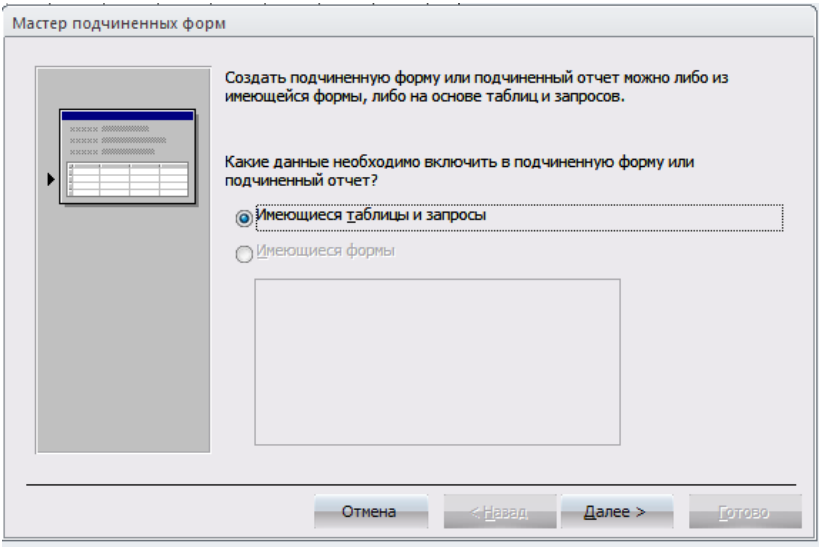

Рис. 14. Мастер подчиненных форм

Выберите из списка предложенных форм созданную ранее форму в табличном виде, при условии, что она удовлетворяет всем требованиям для подчиненных форм, в противном случае создайте подчиненную форму на основе имеющихся таблиц и запросов. Нажмите кнопку *Далее* и следуйте указаниям мастера.

В главной форме создайте кнопки перехода по записям и кнопку закрытия формы, так же как и для ленточной формы, кроме того, создайте кнопку *Добавить новую запись*, по нажатию которой добавлялась бы пустая запись в главную форму, для этого в мастере кнопок выберите категорию *обработка записей* и действие *добавить запись*. Создайте также кнопку, нажатие которой вызовет открытие созданной ранее ленточной формы, для этого в мастере кнопок выберите категорию *работа с формой* и действие *открытие формы*.

Лабораторная работа № 4 «Создание отчетов в СУБД MS Access»

Тема: Создание отчетов в СУБД MS Access. Создание отчетов и их использование для вывода информации.

Раздел дисциплины: Реляционная модель.

Цель работы: создать отчеты в базе данных MS Access, используя в качестве источника записей созданные ранее таблицы и запросы.

Продолжительность: 4 часа.

Отчеты MS Access

Отчет — это гибкое и эффективное средство для организации данных при выводе на печать. С помощью отчета имеется возможность вывести данные на печать в том виде, в котором они требуется конкретному пользователю.

Основные данные для формирования отчета берется из базовой таблицы, запроса или инструкции SQL, являющихся источниками данных для отчета. Другие сведения (заголовки, примечания отчетов, количество страниц и другая сопроводительная информация) вводятся при разработке отчета.

Пользователь имеет возможность разработать отчет самостоятельно или создать отчет с помощью мастера. Мастер по разработке отчетов MS Access выполняет всю рутинную работу и позволяет быстро разработать макет отчета. Работа мастера отчетов аналогична Работе других мастеров в среде MS Access. После создания основной части отчета разработчик может переключиться в режим конструктора и внести изменения в стандартный макет.

Данные в отчете могут быть сгруппированы по различным полям, например, сведения о продаже товара могут быть сгруппированы по месяцам, либо по наименованию товара. При этом в отчете можно указать способ сортировки выводимых на печать данных. Кроме этого, в отчете, по тому же принципу, как и в форме, можно создать подчиненный отчет.

При включении в главный отчет подчиненного отчета, содержащего данные, относящиеся к данным в главном отчете, необходимо установить связь подчиненного отчета с главным. Связь обеспечивает соответствие записей, выводящихся в подчиненном отчете, записям в главном отчете.

При создании подчиненного отчета с помощью мастера или путем переноса с помощью мыши отчета или таблицы в другой отчет, Microsoft Access автоматически выполняет синхронизацию главного и подчиненного отчета в следующих случаях:

Отчеты базируются на таблицах, между которыми установлены связи в окне Схема данных. При создании отчетов на основе запроса или запросов синхронизация отчета с подчиненным отчетом выполняется автоматически, если связи установлены для базовых таблиц запроса или запросов. Если связи базовых таблиц установлены корректно, Microsoft Access выполнит синхронизацию главного и подчиненного отчетов автоматически.

Главный отчет базируется на таблице, содержащей ключевое поле, а подчиненный отчет базируется на таблице, содержащей поле с тем же именем и с тем же или совместимым типом данных. Это же условие должно выполняться для базовых таблиц запросов, если отчеты базируются на запросах.

Перед созданием отчета убедитесь, что на компьютере настроен используемый по умолчанию принтер.

Порядок выполнения работы

В ходе выполнения работы необходимо создать отчет, содержащий подчиненный отчет.

Для создания отчета в окне базы данных выберите вкладку Отчеты и нажмите кнопку Создание отчета с помощью мастера.

В первом окне мастера отчетов выберите имя таблицы или запроса, содержащих данные, по которым строится отчет. MS Access по умолчанию использует эту таблицу или запрос как базовый источник данных для отчета. Для удобства используйте тот же источник записей, что и для главной формы в предыдущей лабораторной работе.

В следующем окне (рис. 15) установите уровни группировки данных (если необходимо). Нажмите кнопку *Далее*.

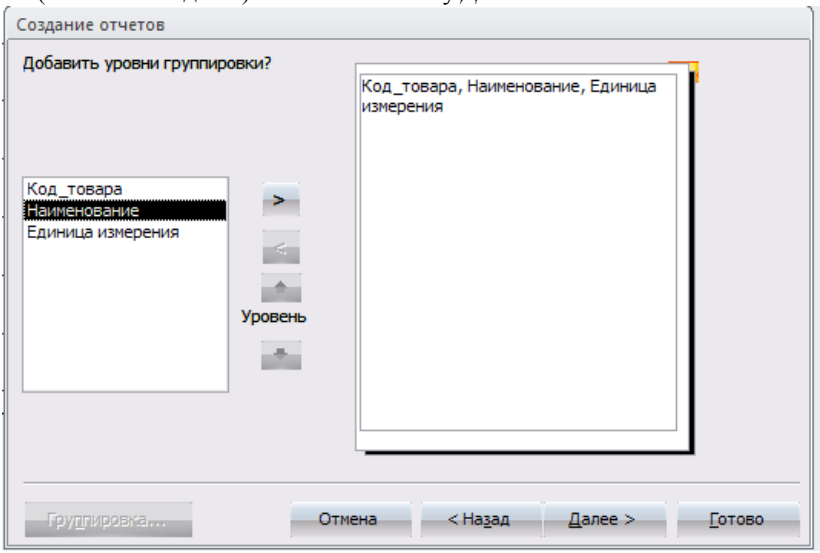

Рис. 15. Мастер отчетов, окно группировки

В следующем окне выберите способ сортировки данных в отчете, нажмите кнопку *Далее* и следуйте дальнейшим инструкциям мастера отчетов.

Для изменения макета отчета откройте его в режиме конструктора (рис. 16).

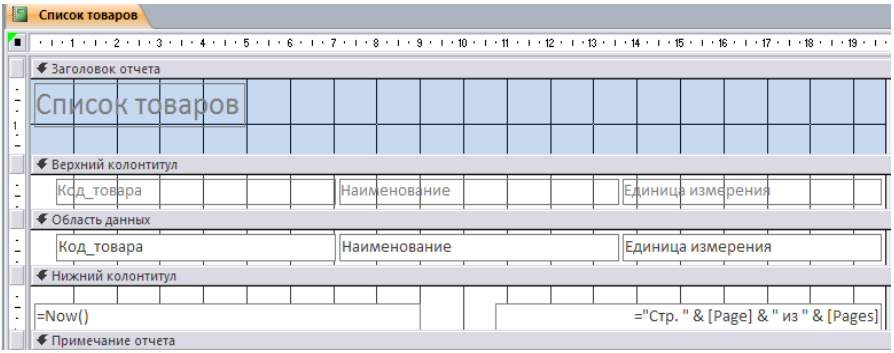

Рис. 16. Отчет, открытый в режиме конструктора

Чтобы изменить внешний вид отчета, нажмите кнопку Автоформат на панели инструментов и выберите новый внешний вид для отчета.

Для изменения внешнего вида одного элемента управления, например надписи, выделите его, после чего на панели инструментов Форматирование MS Sans Serif 8 Ж К Ч выберите другой шрифт, размер шрифта или другие параметры.

Для изменения формата отображения данных в элементе управления, например в поле, убедитесь, что данный элемент выделен и нажмите кнопку Свойства на панели инструментов для вывода окна свойств.

При необходимости добавьте в отчет другие поля из таблицы, являющейся источником данных отчета следующим образом: нажмите кнопку Список полей на панели инструментов для отображения списка всех полей базовой таблицы и с помощью мыши «перенесите» выбранное поле в отчет.

Создайте заголовок и примечание отчета, куда добавьте надпись, содержащую Ваши ФИО и № группы. Чтобы создать надпись, нажмите кнопку Надпись Aa на панели элементов. Затем выберите в отчете место, куда ее следует поместить, введите нужный текст и нажмите клавишу Enter.

Поменяйте порядок сортировки и группировки данных в отчете, для этого нажмите кнопку Сортировка и группировка на панели инструментов для вывода диалогового окна (рис. 17).

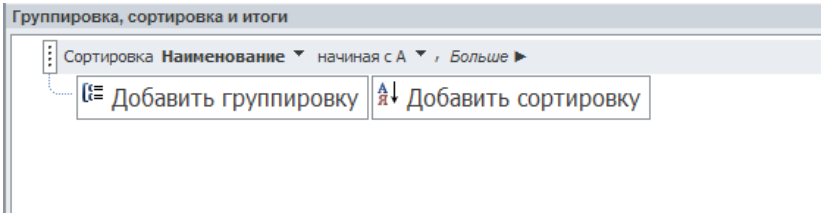

Рис. 17. Окно сортировки и группировки данных

Добавьте новые поля для группировки и сортировки, создайте заголовки и примечание групп.

Для создания подчиненного отчета нажмите кнопку Подчиненная форма/отчет на панели элементов, затем установите указатель в отчете на том месте, куда требуется поместить подчиненный отчет, и нажмите кнопку мыши. Выполняйте инструкции, выводящиеся в диалоговых окнах мастера.

После нажатия кнопки Готово элемент управления «Подчиненная форма/отчет» будет вставлен в главный отчет. Кроме того, будет создан отдельный отчет, выводящийся как подчиненный отчет.

При включении в главный отчет подчиненного отчета, содержащего данные, относящиеся к данным в главном отчете, необходимо установить связь подчиненного отчета с главным. Связь обеспечивает соответствие записей, выводящихся в подчиненном отчете, записям в главном отчете.

При создании подчиненного отчета с помощью мастера или путем переноса с помощью мыши отчета или таблицы в другой отчет, MS Access автоматически выполняет синхронизацию главного и подчиненного отчета в следующих случаях:

Так же как и при создании подчиненных форм следует учитывать, что отчеты базируются на таблицах, между которыми установлены связи в окне Схема данных. Для подчиненного отчета используйте источник записей такой же, как и для подчиненной формы. Помните, что при создании отчетов на основе запроса или запросов синхронизация отчета с подчиненным отчетом выполняется автоматически, если связи установлены для базовых таблиц запроса или запросов. Если связи базовых таблиц установлены корректно, MS Access выполнит синхронизацию главного и подчиненного отчетов автоматически.

Главный отчет базируется на таблице, содержащей ключевое поле, а подчиненный отчет базируется на таблице, содержащей поле с тем же именем и с тем же или совместимым типом данных. Это же условие должно выполняться для базовых таблиц запросов, если отчеты базируются на запросах.

При связывании главного и подчиненного отчетов Microsoft Access использует свойства Основные поля (LinkMasterFields) и Подчиненные поля (LinkChildFields) элемента управления «Подчиненная форма/отчет». Если по каким-либо причинам Microsoft Access не связывает главный и подчиненный отчет автоматически, пользователь имеет возможность задать значения этих свойств самостоятельно.

Для открытия отчета в сложной форме создайте кнопку открытия отчета для просмотра, используя для этого мастер кнопок, по методике, изложенной в лабораторной работе № 3.

Лабораторная работа № 5 «Создание SQL-запросов»

Тема: Создание SQL-запросов.

Раздел дисциплины: Реляционные языки.

Цель работы: создать SQL-запросы на создание таблицы, на выборку с параметрами, на обновление записей, на удаление записей, на добавление данных, на удаление таблицы, на создание индексов.

Продолжительность: 8 часов.

Основы SQL

Запрос SQL — это запрос, создаваемый при помощи инструкций SQL. Язык SQL (Structured Query Language) используется при создании запросов, а также для обновления и управления реляционными БД.

В среде MS Access, когда пользователь создает запрос в режиме конструктора запроса (с помощью построителя запросов), MS Access автоматически создает эквивалентную инструкцию SQL. Фактически, для большинства свойств запроса, доступных в окне свойств в режиме конструктора, имеются эквивалентные предложения или параметры языка SQL, доступные в режиме SQL. При необходимости, пользователь имеет возможность просматривать и редактировать инструкции SQL в режиме SQL. После внесения изменений в запрос в режиме SQL его вид в режиме конструктора может измениться.

Некоторые запросы, которые называют запросами SQL, невозможно создать в бланке запроса. Для запросов к серверу, управляющих запросов и запросов на объединение необходимо создавать инструкции SQL непосредственно в окно запроса в режиме SQL. Для подчиненного запроса пользователь должен ввести инструкцию SQL в строку Поле или Условие отбора в бланке запроса.

Синтаксиса написания SQL-предложений:

– в описании команд слова, написанные прописными латинскими буквами, являются зарезервированными словами SQL;

– фрагменты SQL-предложений, заключенные в фигурные скобки и разделенные символом «|», являются альтернативными. При формировании соответствующей команды для конкретного случая необходимо выбрать одну из них;

– фрагмент описываемого SQL-предложения, заключенный в квадратные скобки **[ ],** имеет необязательный характер и может не использоваться;

– многоточие …, стоящее перед закрывающейся скобкой, говорит о том, что фрагмент, указанный в этих скобках, может быть повторен;

Описание команд SQL

Выборка записей

Инструкция **SELECT**. При выполнении инструкции SELECT СУБД находит указанную таблицу или таблицы, извлекает заданные столбцы, выделяет строки, соответствующие условию отбора, и сортирует или группирует результирующие строки в указанном порядке в виде набора записей.

Синтаксис команды:

SELECT [предикат] { * | таблица.* | [таблица.]поле_1
[AS псевдоним_2] [, [таблица.]поле_2[AS псевдоним_2] [, ...]]}
FROM выражение [, ...]
[WHERE...]
[GROUP BY...]
[HAVING...]
[ORDER BY...]

где предикат — один из следующих предикатов отбора: ALL, DISTINCT, DISTINCTROW, TOP. Данные ключевые слова используются для ограничения числа возвращаемых записей. Если они отсутствуют, по умолчанию используется предикат ALL;

* указывает, что результирующий набор записей будет содержать все поля заданной таблицы или таблиц. Следующая инструкция отбирает все поля из таблицы «Студенты»: SELECT * FROM Студенты;

таблица — имя таблицы, из которой выбираются записи;

поле_1, поле_2 — имена полей, из которых должны быть отобраны данные;

псевдоним_1, псевдоним_2 — ассоциации, которые станут заголовками столбцов вместо исходных названий полей в таблице;

выражение — имена одной или нескольких таблиц, которые содержат необходимые для отбора записи;

предложение GROUP BY в SQL-предложении объединяет записи с одинаковыми значениями в указанном списке полей в одну запись. Если инструкция SELECT содержит статистическую функцию SQL, например Sum или Count, то для каждой записи будет вычислено итоговое значение;

предложение HAVING определяет, какие сгруппированные записи, выданные в результате выполнения запроса, отображаются при использовании инструкции SELECT с предложением GROUP BY. После того как записи результирующего набора будут сгруппированы с помощью предложения GROUP BY, предложение HAVING отберет те из них, которые удовлетворяют условиям отбора, указанным в предложении HAVING;

предложение ORDER BY позволяет отсортировать записи, полученные в результате запроса, в порядке возрастания или убывания

на основе значений указанного поля или полей.

Следует отметить, что инструкции SELECT не изменяют данные в базе данных. Приведем минимальный синтаксис инструкции SELECT: SELECT поля FROM таблица.

Если несколько таблиц, включенных в предложение FROM, содержат одноименные поля, перед именем такого поля следует ввести имя таблицы и оператор « . » (точка). Предположим, что поле «Номер_группы» содержится в таблицах «Студенты» и «Группы». Следующая инструкция SQL отберет поле «Номер_группы» и «ФИО_студента» из таблицы «Студенты» и «ФИО_куратора» из таблицы «Группы» при номере группы, равном 432-1:

SELECT Группы.Номер_группы, Группы.ФИО_куратора, Студенты.ФИО_студента

FROM Группы, Студенты

WHERE Группы.Номер_группы = Студенты.Номер_группы AND

На рис. 18 приведен пример выполнения данного запроса.

Таблицы БД

СТУДЕНТЫ

Номер_зачетной книжки	ФИО_студента	Дата рождения	Место рождения	Номер_группы
1992412-11	Карасев А.А.	27.08.75	г. Чита	412-1
1992432-11	Данилов О. В.	27.08.75	г. Алматы	432-1
1992432-12	Раевский А. И.	20.05.75	г. Бишкек	432-1
1992432-22	Глазов О.А	04.07.75	г. Киров	432-1

ГРУППЫ

Номер_Группы	ФИО_куратора
412-1	Самойлов С.С.
432-1	Авдеев Р.М

Результат выполнения запроса

Номер_группы	ФИО_куратора	ФИО_студента
432-1	Авдеев Р.М	Данилов О. В.
432-1	Авдеев Р.М	Раевский А. И.
432-1	Авдеев Р.М	Глазов О.А

Рис. 18. Пример выполнения запроса на выборку

Помимо обычных знаков сравнения (=,<,>,<=,>=, <>) в языке SQL в условии отбора используются ряд ключевых слов:

Is not null — выбрать только непустые значения;

Is null — выбрать только пустые значения;

Between … And определяет принадлежность значения выражения указанному диапазону.

Синтаксис:

выражение [Not] Between значение_1 And значение_2 ,

где выражение — выражение, определяющее поле, значение которого проверяется на принадлежность к диапазону;

значение_1, значение_2 – выражения, задающие границы диапазона.

Если значение поля, определенного в аргументе выражение, попадает в диапазон, задаваемый аргументами значение_1 и значение_2 (включительно), то оператор Between...And возвращает значение True; в противном случае возвращается значение False. Логический оператор Not позволяет проверить противоположное условие: что выражение находится за пределами диапазона, заданного с помощью аргументов значение_1 и значение_2.

Оператор Between...And часто используют для проверки: попадает ли значение поля в указанный диапазон чисел. В следующем примере выдается список студентов, получающих стипендию от 800 до 900 рублей:

```
SELECT ФИО_студента, Размер_стипендии
FROM Студенты
WHERE Размер_стипендии Between 800 And 900
```

На рис. 19 приведен результат выполнения запроса.

СТУДЕНТЫ

Номер_зачетной книжки	ФИО_студента	Размер_стипендии
1992412-11	Карасев А.А.	900
1992432-11	Данилов О. В.	800
1992432-12	Раевский А. И.	950
1992432-22	Глазов О.А	850

Результирующий набор данных

ФИО_студента	Размер_стипендии
Карасев А.А.	900
Данилов О. В.	800
Глазов О.А	850

Рис. 19. Результат выполнения запроса на выборку с использованием операторов Between...And

Если выражение, значение_1 или значение_2 имеет значение Null, оператор Between...And возвращает значение Null.

Оператор Like используется для сравнения строкового выражения с образцом.

Синтаксис:

выражение Like «образец»,

где выражение — выражение SQL, используемое в предложении WHERE; образец — строка, с которой сравнивается выражение.

Оператор Like используется для нахождения в поле значений, соответствующих указанному образцу. Для аргумента образец можно задавать полное значение (например, Like «Иванов») или использовать подстановочные знаки для поиска диапазона значений (например, Like "Ив*").

Приведем перечень подстановочных символов и пример их использования в языке Jet SQL согласно документации Microsoft.

Параметры оператора Like

Тип совпадения	Образец	Совпадение (True)	Несовпадение (False)
Несколько символов	a*a	aa, aBa, aBBBa	aBC
	ab	abc, AABB, Xab	aZb, bac
Специальный символ	a[*]a	a*a	aaa
Несколько символов	ab*	abcdefg, abc	cab, aab
Одиночный символ	a?a	aaa, a3a, aBa	aBBBa
Одиночная цифра	a#a	a0a, a1a, a2a	aaa, a10a
Диапазон символов	[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

Внутреннее соединение

Операция **INNER JOIN** объединяет записи из двух таблиц, если связующие поля этих таблиц содержат одинаковые значения.

Синтаксис операции:

FROM таблица_1 INNER JOIN таблица_2 ON таблица_1.поле_1 оператор таблица_2.поле_2

где таблица_1, таблица_2 — имена таблиц, записи которых подлежат объединению;

поле_1, поле_2 — имена объединяемых полей. Поля должны иметь одинаковый тип данных и содержать данные одного рода, однако эти поля могут иметь разные имена;

оператор — любой оператор сравнения: "=," "<," ">," "<=," ">=," или "<>".

Операцию INNER JOIN можно использовать в любом предложении FROM. Это самые обычные типы связывания. Они объединяют записи двух таблиц, если связующие поля обеих таблиц содержат одинаковые значения. Предыдущий пример использования команды SELECT можно записать с использованием конструкции INNER JOIN следующим образом:

SELECT Группы.Номер_группы, Группы.ФИО_куратора, Студенты.ФИО_студента

FROM Группы INNER JOIN Студенты

ON Группы.Номер_группы = Студенты.Номер_группы;

Внешнее соединение

Операции **LEFT JOIN, RIGHT JOIN** объединяют записи исходных таблиц при использовании в любом предложении FROM.

Операция LEFT JOIN используется для создания внешнего соединения, при котором все записи из первой (левой) таблицы включаются в результирующий набор, даже если во второй (правой) таблице нет соответствующих им записей.

Операция RIGHT JOIN используется для создания внешнего объединения, при котором все записи из второй (правой) таблицы включаются в результирующий набор, даже если в первой (левой) таблице нет соответствующих им записей.

Синтаксис операции:

FROM таблица_1 [LEFT | RIGHT] JOIN таблица_2

ON таблица_1.поле_1 оператор таблица_2.поле_2

Например, операцию LEFT JOIN можно использовать с таблицами «Студенты» (левая) и «Задолженность_за_обучение» (правая) для отбора всех студентов, в том числе тех, которые не являются задолжниками:

```
SELECT Студенты.ФИО_студента,
Задолженность_за_обучение.Сумма_задолженности
FROM Студенты LEFT JOIN Задолженность_за_обучение
ON Студенты.Номер_зачетной_книжки =
 Задолженность_за_обучение.Номер_зачетной_книжки;
```

Поле «Номер_зачетной_книжки» в этом примере используется для объединения таблиц, однако, оно не включается в результат выполнения запроса, поскольку не включено в инструкцию SELECT. Чтобы включить связующее поле (в данном случае поле «Номер_зачетной_книжки») в результат выполнения запроса, его имя необходимо включить в инструкцию SELECT.

Важно отметить, что операции LEFT JOIN или RIGHT JOIN могут быть вложены в операцию INNER JOIN, но операция INNER JOIN не может быть вложена в операцию LEFT JOIN или RIGHT JOIN.

Перекрестные запросы

В некоторых СУБД (в частности в MS Access) существует такой вид запросов как перекрестный. В перекрестном запросе отображаются результаты статистических функций — суммы, средние значения и др., а также количество записей. При этом подсчет выполняется по данным из одного полей таблицы. Результаты группируются по двум наборам данных, один из которых расположен в левом столбце таблицы, а другой в заголовке таблицы. Например, при необходимости вычислить средний балл студентов за семестр, обучающихся на разных кафедрах, необходимо реализовать перекрестный запрос, в результате выполнения которого будет создана таблица, где заголовками строк будут служить номер семестра, заголовками столбцов — названия кафедр, а в полях таблицы будет рассчитан средний балл.

Для создания перекрестного запроса необходимо использовать следующую инструкцию:

TRANSFORM статистическая_функция
инструкция_SELECT
PIVOT поле [IN (значение_1[, значение_2[, ...]])],

где статистическая_функция — статистическая функция SQL, обрабатывающая указанные данные;

инструкция_SELECT — запрос на выборку;

поле — поле или выражение, которое содержит заголовки столбцов для результирующего набора;

значение_1, значение_2 — фиксированные значения, используемые при создании заголовков столбцов.

Составим SQL-запрос, реализующий описанный выше пример. В качестве исходного набора данных используется таблица Успеваемость (рис. 20).

Успеваемость : таблица

НОМЕР_ЗАЧЕТНОЙ_КНИЖКИ	Семестр	Оценка	Кафедра
102	1	3	АСУ
102	2	3	АСУ
102	3	4	АСУ
102	4	4	АСУ
110	1	5	АОИ
110	2	3	АОИ
110	3	3	АОИ
110	4	4	АОИ
110	5	4	АОИ

Рис. 20. Таблица УСПЕВАЕМОСТЬ

В результате выполнения нижеприведенного перекрестного SQL-запроса формируется следующая таблица (рис. 21):

```
TRANSFORM AVG(Успеваемость.Оценка) AS Сред_балл
SELECT Успеваемость.Семестр
FROM Успеваемость
GROUP BY Успеваемость.Семестр
PIVOT Успеваемость.Кафедра
```

SQL_запрос : перекрестный запрос

Семестр	АОИ	АСУ
1	5	3
2	3	3
3	3	4
4	4	4
5	4	

Рис. 21. Результат выполнения перекрестного запроса

Таким образом, когда данные сгруппированы с помощью перекрестного запроса, можно выбирать значения из заданных столбцов или выражений и определять как заголовки столбцов. Это позволяет просматривать данные в более компактной форме, чем при работе с запросом на выборку.

Подчиненные запросы

Часто возникает ситуация, когда желаемый результат нельзя получить с помощью одного SQL-запроса. Одним из способов решения

такой задачи является использование подчиненных запросов в составе главного SQL-запроса. Подчиненным SQL-запросом называют инструкцию SELECT, включаемую в инструкции SELECT, SELECT...INTO, INSERT...INTO, DELETE или UPDATE или в другой подчиненный запрос. Подчиненный запрос может быть создан одним из трех способов, синтаксис которых представлен ниже:

1) сравнение [ANY | ALL | SOME] (инструкцияSQL)

2) выражение [NOT] IN (инструкцияSQL)

3) [NOT] EXISTS (инструкцияSQL),

где сравнение — выражение и оператор сравнения, который сравнивает выражение с результатами подчиненного запроса;

выражение — выражение, для которого проводится поиск в результирующем наборе записей подчиненного запроса;

инструкцияSQL — инструкция SELECT, заключенная круглые скобки.

Подчиненный запрос можно использовать вместо выражения в списке полей инструкции SELECT или в предложениях WHERE и HAVING. Инструкция SELECT используется в подчиненном запросе для задания набора конкретных значений, вычисляемых в выражениях предложений WHERE или HAVING.

Предикаты ANY или SOME, являющиеся синонимами, используются для отбора записей в главном запросе, которые удовлетворяют сравнению с записями, отобранными в подчиненном запросе. В следующем примере отбираются все студенты, средний балл которых за семестр больше 4.

```
SELECT * FROM Студенты
WHERE Номер_зачетной_книжки = ANY
(SELECT Номер_зачетной_книжки FROM Успеваемость
WHERE оценка > 4)
```

Предикат ALL используется для отбора в главном запросе только тех записей, которые удовлетворяют сравнению со всеми записями, отобранными в подчиненном запросе. Если в предыдущем примере предикат ANY заменить предикатом ALL, результат запроса будет включать только тех студентов, у которых средний балл больше 4. Это условие является значительно более жестким.

Предикат IN используется для отбора в главном запросе только тех записей, которые содержат значения, совпадающие с одним из отобранных подчиненным запросом. Следующий пример возвращает

сведения обо всех студентах, средний балл которых за семестр был больше 4.

```
SELECT * FROM Студенты
WHERE Номер_зачетной_книжки in
(SELECT Номер_зачетной_книжки FROM Успеваемость
WHERE оценка > 4)
```

Предикат NOT IN используется для отбора в главном запросе только тех записей, которые содержат значения, не совпадающие ни с одним из отобранных подчиненным запросом.

Предикат EXISTS (с необязательным зарезервированным словом NOT) используется в логическом выражении для определения того, должен ли подчиненный запрос возвращать какие-либо записи.

В подчиненном запросе можно использовать псевдонимы таблиц для ссылки на таблицы, перечисленные в предложении FROM, расположенном вне подчиненного запроса. В следующем примере отбираются фамилии и имена студентов, чья стипендия равна или больше средней стипендии студентов, обучающихся в той же группе. В данном примере таблица СТУДЕНТЫ получает псевдоним С1:

```
SELECT Фамилия,
Имя, Номер_группы, Стипендия
FROM СТУДЕНТЫ AS C1
WHERE Стипендия >=
(SELECT Avg(Стипендия)
FROM СТУДЕНТЫ
WHERE C1.Номер_группы = СТУДЕНТЫ.Номер_группы) Order by Номер_группы;
```

В последнем примере зарезервированное слово AS не является обязательным.

Некоторые подчиненные запросы можно использовать в перекрестных запросах как предикаты (в предложении WHERE). Подчиненные запросы, используемые для вывода результатов (в списке SELECT), нельзя использовать в перекрестных запросах.

Создание новой таблицы

Инструкция **CREATE TABLE** создает новую таблицу и используется для описания ее полей и индексов. Если для поля добавлено ограничение NOT NULL, то при добавлении новых записей это поле должно содержать допустимые данные. Синтаксис:

CREATE TABLE таблица (поле_1 тип [(размер)]

[NOT NULL] [индекс_1] [, поле_2 тип [(размер)]

[NOT NULL] [индекс_2] [, ...]] [, CONSTRAINT составной Индекс [, ...]]),

где таблица — имя создаваемой таблицы;

поле_1, поле_2 — имена одного или нескольких полей, создаваемых в новой таблице. Таблица должна содержать хотя бы одно поле;

тип — тип данных поля в новой таблице;

размер — размер поля в символах (только для текстовых и двоичных полей);

индекс_1, индекс_2 — предложение CONSTRAINT, предназначенное для создания простого индекса;

составной Индекс — предложение CONSTRAINT, предназначенное для создания составного индекса.

следующем примере создается новая таблица с двумя полями:

CREATE TABLE Студенты (Номер_зачетной_книжки integer PRIMARY KEY, ФИО_студента TEXT (50), Место_рождения TEXT (50));

В результате выполнения этого запроса будет создана таблица со следующей схемой (рис. 22):

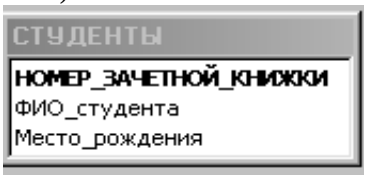

Рис. 22. Схема таблицы СТУДЕНТЫ

Предложение ***CONSTRAINT*** используется в инструкциях ALTER TABLE и CREATE TABLE для создания или удаления индексов. Существуют два типа предложений CONSTRAINT: для создания простого индекса (по одному полю) и для создания составного индекса (по нескольким полям). Синтаксис:

простой индекс:

CONSTRAINT имя {PRIMARY KEY|UNIQUE | NOT NULL]}

составной индекс:

CONSTRAINT имя

{PRIMARY KEY (ключевое_1[, ключевое_2 [, ...]]) |

UNIQUE (уникальное_1[, уникальное_2 [, ...]]) |

NOT NULL (непустое_1[, непустое_2 [, ...]]) |

FOREIGN KEY (ссылка_1[, ссылка_2 [, ...]])

REFERENCES внешняя Таблица [(внешнее Поле_1 [, внешнее Поле_2 [, ...]])]},

где имя — имя индекса, который следует создать;

ключевое_1, ключевое_2 — имена одного или нескольких полей, которые следует назначить ключевыми;

уникальное_1, уникальное_2 — имена одного или нескольких полей, которые следует включить в уникальный индекс;

непустое_1, непустое_2 — имена одного или нескольких полей, в которых запрещаются значения Null;

ссылка_1, ссылка_2 — имена одного или нескольких полей, включенных во внешний ключ, которые содержат ссылки на поля в другой таблице;

внешняя Таблица — имя внешней таблицы, которая содержит поля, указанные с помощью аргумента внешнееПоле;

внешнее Поле_1, внешнее Поле_2 — имена одного или нескольких полей во внешней Таблице, на которые ссылаются поля, указанные с помощью аргумента ссылка_1, ссылка_2. Это предложение можно опустить, если данное поле является ключом внешней Таблицы.

Предложение CONSTRAINT позволяет создать для поля индекс одного из двух описанных ниже типов:

1) уникальный индекс, использующий для создания зарезервированное слово UNIQUE. Это означает, что в таблице не может быть двух записей, имеющих одно и то же значение в этом поле. Уникальный индекс создается для любого поля или любой группы полей. Если в таблице определен составной уникальный индекс, то комбинация значений включенных в него полей должна быть уникальной для каждой записи таблицы, хотя отдельные поля и могут иметь совпадающие значения;

2) ключ таблицы, состоящий из одного или нескольких полей, использующий зарезервированные слова PRIMARY KEY. Все значения ключа таблицы должны быть уникальными и не значениями Null. Кроме того, в таблице может быть только один ключ.

Для создания внешнего ключа можно использовать зарезервированную конструкцию FOREIGN KEY. Если ключ внешней таблицы

состоит из нескольких полей, необходимо использовать предложение CONSTRAINT. При этом следует перечислить все поля, содержащие ссылки на поля во внешней таблице, а также указать имя внешней таблицы и имена полей внешней таблицы, на которые ссылаются поля, перечисленные выше, причем в том же порядке. Однако, если последние поля являются ключом внешней таблицы, то указывать их необязательно, поскольку ядро базы данных считает, что в качестве этих полей следует использовать поля, составляющие ключ внешней таблицы.

В следующем примере создается таблица ЗАДОЛЖЕННОСТЬ_ЗА_ОБУЧЕНИЕ с единственным полем НОМЕР_ЗАЧЕТНОЙ_КНИЖКИ и внешним ключом f1_i, связанным с полем НОМЕР_ЗАЧЕТНОЙ_КНИЖКИ в таблице СТУДЕНТЫ:

CREATE TABLE Задолженность_за_обучение

(Код_задолженности integer PRIMARY KEY, Номер_зачетной_книжки integer, CONSTRAINT f1_i FOREIGN KEY (Номер_зачетной_книжки) REFERENCES Студенты (Номер_зачетной_книжки));

Внешний вид схемы БД, состоящей из таблиц СТУДЕНТЫ и ЗАДОЛЖЕННОСТЬ_ЗА_ОБУЧЕНИЕ, представлен на рис. 23.

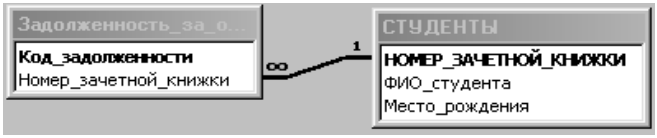

Рис. 23 Схема данных

Изменение структуры таблицы

Инструкция **ALTER TABLE** изменяет структуру таблицы, созданной с помощью инструкции CREATE TABLE.

Синтаксис:

ALTER TABLE таблица {ADD {COLUMN поле тип[(размер)] [NOT NULL]

[CONSTRAINT индекс] | CONSTRAINT составной Индекс} |

DROP {COLUMN поле I CONSTRAINT имя Индекса} }

где таблица — имя изменяемой таблицы;

поле — имя поля, добавляемого в таблицу или удаляемого из нее;

тип — тип данных поля;

размер — размер поля;

индекс — индекс для поля;

составной Индекс — описание составного индекса, добавляемого к таблице;

имя Индекса — имя составного индекса, который следует удалить.

С помощью инструкции ALTER TABLE существующую таблицу можно изменить несколькими способами:

1) добавить новое поле в таблицу с помощью предложения ADD COLUMN. В этом случае необходимо указать имя поля, его тип и размер. Например, следующая инструкция добавляет в таблицу СТУДЕНТЫ текстовое поле ПРИМЕЧАНИЯ длиной 50 символов:

ALTER TABLE Студенты ADD COLUMN Примечания TEXT(50)

Если для поля добавлено ограничение NOT NULL, то при добавлении новых записей это поле должно содержать допустимые данные;

2) добавить составной индекс с помощью зарезервированных слов ADD CONSTRAINT;

3) удалить поле с помощью зарезервированных слов DROP COLUMN. В этом случае необходимо указать только имя поля;

4) удалить составной индекс с помощью зарезервированных слов DROP CONSTRAINT. В этом случае указывается только имя составного индекса, следующее за зарезервированным словом CONSTRAINT.

Создание индекса с помощью инструкции CREATE INDEX

CREATE INDEX создает новый индекс для существующей таблицы. Синтаксис команды:

CREATE [UNIQUE] INDEX индекс
ON таблица (поле [ASC|DESC][, поле [ASC|DESC], ...])
[WITH { PRIMARY | DISALLOW NULL | IGNORE NULL }]

где индекс — имя создаваемого индекса;

таблица — имя существующей таблицы, для которой создается индекс;

поле — имена одного или нескольких полей, включаемых в индекс. Для создания простого индекса, состоящего из одного поля, вводится имя поля в круглых скобках сразу после имени таблицы. Для создания составного индекса, состоящего из нескольких полей, перечисляются имена всех этих полей. Для расположения элементов индекса в убывающем порядке используется зарезервированное слово DESC; в противном случае будет принят порядок по возрастанию.

Чтобы запретить совпадение значений индексированных полей в разных записях, используется зарезервированное слово UNIQUE. Необязательное предложение WITH позволяет задать условия на значения. Например:

– с помощью параметра DISALLOW NULL можно запретить значения Null в индексированных полях новых записей;

– параметр IGNORE NULL позволяет запретить включение в индекс записей, имеющих значения Null в индексированных полях;

– зарезервированное слово PRIMARY позволяет назначить индексированные поля ключом. Такой индекс по умолчанию является уникальным, следовательно, зарезервированное слово UNIQUE можно опустить.

Удаление таблицы/индекса

Инструкция **DROP** удаляет существующую таблицу из базы данных или удаляет существующий индекс из таблицы. Синтаксис:

DROP {TABLE таблица | INDEX индекс ON таблица}

где таблица — имя таблицы, которую следует удалить или из которой следует удалить индекс;

индекс — имя индекса, удаляемого из таблицы.

Прежде чем удалить таблицу или удалить из нее индекс, необходимо ее закрыть. Следует отметить, что таблица удаляется из базы данных безвозвратно.

Удаление записей

Инструкция **DELETE** создает запрос на удаление записей из одной или нескольких таблиц, перечисленных в предложении FROM и удовлетворяющих предложению WHERE.

Синтаксис команды:

DELETE [Таблица.*]

FROM таблица

WHERE условие Отбора

где Таблица — необязательное имя таблицы, из которой удаляются записи;

таблица — имя таблицы, из которой удаляются записи;

условие Отбора — выражение, определяющее удаляемые записи.

С помощью инструкция DELETE можно осуществлять удаление большого количества записей. Данные из таблицы также можно удалить и с помощью инструкции DROP, однако при таком удалении теряется структура таблицы. Если же применить инструкцию DELETE,

удаляются только данные. При этом сохраняются структура таблицы и все остальные ее свойства, такие, как атрибуты полей и индексы.

Запрос на удаление удаляет записи целиком, а не только содержимое указанных полей. Нельзя восстановить записи, удаленные с помощью запроса на удаление. Чтобы узнать, какие записи будут удалены, необходимо посмотреть результаты запроса на выборку, использующего те же самые условие отбора в предложении Where, а затем выполнить запрос на удаление.

Добавление записей

Инструкция **INSERT INTO** добавляет запись или записи в таблицу.

Синтаксис команды:

а) запрос на добавление нескольких записей:

INSERT INTO назначение [(поле_1[, поле_2[, ...]])]

SELECT [источник.]поле_1[, поле_2[, ...]

FROM выражение

б) запрос на добавление одной записи:

INSERT INTO назначение [(поле_1[, поле_2[, ...]])]

VALUES (значение_1[, значение_2[, ...])

где назначение — имя таблицы или запроса, в который добавляются записи;

источник — имя таблицы или запроса, откуда копируются записи;

поле_1, поле_2 — имена полей для добавления данных, если они следуют за аргументом «Назначение»; имена полей, из которых берутся данные, если они следуют за аргументом источник;

выражение — имена таблицы или таблиц, откуда вставляются данные. Это выражение может быть именем отдельной таблицы или результатом операции INNER JOIN, LEFT JOIN или RIGHT JOIN, а также сохраненным запросом;

значение_1, значение_2 — значения, добавляемые в указанные поля новой записи. Каждое значение будет вставлено в поле, занимающее то же положение в списке: значение_1 вставляется в поле_1 в новой записи, значение_2 — в поле_2 и т.д. Каждое значение текстового поля следует заключать в кавычки (' '), для разделения значений используются запятые.

Инструкцию INSERT INTO можно использовать для добавления одной записи в таблицу с помощью запроса на добавление одной записи, описанного выше. В этом случае инструкция должна содержать имя и значение каждого поля записи. Нужно определить все поля записи, в

которые будет помещено значение, и значения для этих полей. Если поля не определены, в недостающие столбцы будет вставлено значение по умолчанию или значение Null. Записи добавляются в конец таблицы.

Инструкцию INSERT INTO можно также использовать для добавления набора записей из другой таблицы или запроса с помощью предложения SELECT ... FROM, как показано выше в запросе на добавление нескольких записей. В этом случае предложение SELECT определяет поля, добавляемые в указанную таблицу НАЗНАЧЕНИЕ. Инструкция INSERT INTO является необязательной, однако, если она присутствует, то должна находиться перед инструкцией SELECT.

Запрос на добавление записей копирует записи из одной или нескольких таблиц в другую таблицу. Таблицы, которые содержат добавляемые записи, не изменяются.

Вместо добавления существующих записей из другой таблицы, можно указать значения полей одной новой записи с помощью предложения VALUES. Если список полей опущен, предложение VALUES должно содержать значение для каждого поля таблицы; в противном случае инструкция INSERT не будет выполнена. Можно использовать дополнительную инструкцию INSERT INTO с предложением VALUES для каждой добавляемой новой записи.

Обновление данных

Инструкция **UPDATE** создает запрос на обновление, который изменяет значения полей указанной таблицы на основе заданного условия отбора.

Синтаксис команды:

UPDATE таблица

SET новое Значение

WHERE условие Отбора;

где таблица — имя таблицы, данные в которой следует изменить;

новое Значение — выражение, определяющее значение, которое должно быть вставлено в указанное поле обновленных записей;

условие Отбора — выражение, отбирающее записи, которые должны быть изменены.

При выполнении этой инструкции будут изменены только записи, удовлетворяющие указанному условию. Инструкцию UPDATE особенно удобно использовать для изменения сразу нескольких записей или в том случае, если записи, подлежащие изменению, находятся в разных таблицах. Одновременно можно изменить значения нескольких полей. Следующая инструкция SQL увеличивает стипендию студентов группы 422-1 на 10 %:

```
UPDATE Студенты
SET Стипендия = стипендия * 1.1
WHERE Номер_группы = '422-1';
```

Запрос на объединение

Операция **UNION** создает запрос на объединение, который объединяет результаты нескольких независимых запросов или таблиц.

Синтаксис команды:

[TABLE] запрос_1 UNION [ALL] [TABLE] запрос_2 [UNION[ALL] [TABLE] запрос_n [...]]

где запрос_1-n — инструкция SELECT или имя сохраненной таблицы, перед которым стоит зарезервированное слово TABLE.

В одной операции UNION можно объединить в любом наборе результаты нескольких запросов, таблиц и инструкций SELECT. В следующем примере объединяется существующая таблица СТУДЕНТЫ и инструкции SELECT:

```
TABLE Студенты UNION ALL
SELECT *
FROM Абитуриенты
WHERE Общий_балл > 22;
```

По умолчанию повторяющиеся записи не возвращаются при использовании операции UNION, однако в нее можно добавить предикат ALL, чтобы гарантировать возврат всех записей. Кроме того, такие запросы выполняются несколько быстрее.

Все запросы, включенные в операцию UNION, должны отбирать одинаковое число полей; при этом типы данных и размеры полей не обязаны совпадать. Псевдонимы необходимо использовать только в первом предложении SELECT, в остальных они пропускаются.

В каждом аргументе «Запрос» допускается использование предложения GROUP BY или HAVING для группировки возвращаемых данных. В конец последнего аргумента «Запрос» можно включить предложение ORDER BY, чтобы отсортировать возвращенные данные.

Основные различия Microsoft Jet SQL и ANSI SQL

Рассмотрим различия двух диалектов языка SQL Microsoft Jet SQL и ANSI SQL, описанные в руководстве разработчика приложений баз данных на СУБД Microsoft Access:

– языки SQL-ядра базы данных Microsoft Jet и ANSI SQL имеют разные наборы зарезервированных слов и типов данных;

– разные правила применимы к оператору Between...And, имеющему следующий синтаксис: выражение [NOT] Between значение_1 And значение_2. В языке SQL Microsoft Jet значение_1 может превышать значение_2; в ANSI SQL, значение_1 должно быть меньше или равно значение_2;

– разные подстановочные знаки используются с оператором Like. Так, в языке SQL Microsoft Jet любой одиночный символ изображается знаком «?», а в ANSI SQL знаком «_», любое число символов в языке SQL Microsoft Jet изображается знаком «*», а в ANSI SQL — знаком «%»;

– язык SQL-ядра Microsoft Jet обычно предоставляет пользователю большую свободу. Например, разрешается группировка и сортировка по выражениям.

Особые средства языка SQL Microsoft Jet

В языке SQL Microsoft Jet поддерживаются следующие дополнительные средства:

– инструкция TRANSFORM, предназначенная для создания перекрестных запросов;

– дополнительные статистические функции, такие как StDev и VarP;

– описание PARAMETERS, предназначенное для создания запросов с параметрами.

Средства ANSI SQL, не поддерживаемые в языке SQL Microsoft Jet

В языке SQL Microsoft Jet не поддерживаются следующие средства ANSI SQL:

– инструкции, управляющие защитой, такие как COMMIT, GRANT и LOCK;

– зарезервированное слово DISTINCT в качестве описания аргумента статистической функции (например, нельзя использовать выражение SUM(DISTINCT имя Столбца);

– предложение LIMIT TO nn ROWS, используемое для ограничения количества строк, возвращаемых в результате выполнения запроса. Для ограничения количества возвращаемых запросом строк можно использовать только предложение WHERE.

Порядок выполнения работы

Все запросы, создаваемые в рамках данной лабораторной работы, должны быть реализованы на языке SQL без помощи построителя запросов.

Для создания нового SQL-запроса в окне базы данных нажмите кнопку Создание запроса в режиме конструктора и не выбирая таблицы для запроса нажмите кнопку Вид  на панели инструментов.

1. Используя инструкцию CREATE TABLE, создайте запрос на создание новой таблицы для выбранной ранее предметной области, содержащую пять полей, различных типов данных, определив в запросе первичный ключ и проиндексировав соответствующие поля, используя предложение CONSTRAINT. Для запуска запроса нажмите кнопку Запуск  на панели инструментов. После чего создайте запрос на создание еще одной таблицы, содержащей внешний ключ по отношению к первичному ключу предыдущей таблицы. Запустите запрос, после чего проверьте, отразились ли изменения в схеме данных.
2. Используя команду SELECT, создайте запрос на выборку записей из двух (или более) таблиц, используя правила внешнего и внутреннего соединения, а также различные условия отбора и сортировки.
3. Используя команду UPDATE, создайте запрос на обновление данных в созданных ранее таблицах.
4. Используя команду INSERT INTO, создайте запросы на добавление группы записей и одной записи в существующую таблицу.
5. Используя команду CREATE INDEX, создайте запрос на создание нового индекса, используя различные условия на значения индексов (IGNORE NULL, DISSALLOW NULL, PRIMARY), а также типы сортировки.
6. Используя команду DROP, создайте запросы на удаление таблицы и индекса, созданных ранее в БД.

Сохраните все созданные запросы в базе данных.

Лабораторная работа № 6 «Создание концептуальной модели данных в среде автоматизированного проектирования»

Тема: Создание концептуальной модели данных в среде автоматизированного проектирования.

Раздел дисциплины: Моделирование данных с помощью ER-диаграмм.

Цель работы: спроектировать концептуальную модель, выбранной ранее предметной области в пакете Power Designer.

Продолжительность: 8 часов.

Сущности и атрибуты

Каждая сущность является множеством подобных индивидуальных объектов, называемых экземплярами. Каждый экземпляр индивидуален и должен отличаться от всех остальных экземпляров. Атрибут выражает определенное свойство объекта. С точки зрения БД (физическая модель) сущности соответствует таблица, экземпляру сущности – строка в таблице, а атрибуту – колонка таблицы.

Построение модели данных предполагает определение сущностей и атрибутов, т.е. необходимо определить, какая информация будет храниться в конкретной сущности или атрибуте. Сущность можно определить как объект, событие или концепцию, информация о которой должна сохраняться. Сущности должны иметь наименование с четким смысловым значением, именоваться существительным в единственном лице, не носить «технических» наименований и быть достаточно важными для того, чтобы их моделировать. Именование сущности в единственном числе облегчает в дальнейшем чтение модели. Фактически имя сущности дается по имени ее экземпляра.

Каждая сущность должна быть полностью определена с помощью текстового описания. Каждый атрибут хранит информацию об определенном свойстве сущности, а каждый экземпляр сущности должен быть уникальным. Атрибут или группа атрибутов, которые идентифицируют сущность, называется первичным ключом. При установлении связей между сущностями атрибуты первичного ключа родительской сущности мигрируют в качестве внешних ключей в дочернюю сущность.

Очень важно дать атрибуту правильное имя. Атрибуты должны именоваться в единственном числе и иметь четкое смысловое значение. Соблюдение этого правила позволяет частично решить проблему нормализации данных уже на этапе определения атрибутов.

Связи

Связь является логическим соотношением между сущностями. Каждая связь должна именоваться глаголом или глагольной фразой. Имя связи выражает некоторое ограничение или бизнес-правило и облегчает чтение построенной модели данных.

Различают зависимые и независимые сущности. Тип сущности определяется ее связью с другими сущностями. Идентифицирующая связь устанавливается между независимой (родительский конец связи) и зависимой (дочерний конец связи) сущностями. При установлении идентифицирующей связи атрибуты первичного ключа родительской

сущности переносятся в состав первичного ключа дочерней сущности. Эта операция дополнения атрибутов дочерней сущности при создании связи называется миграцией атрибутов. В дочерней сущности атрибуты помечаются как внешний ключ (FK).

При установлении неидентифицирующей связи дочерняя сущность остается независимой, а атрибуты первичного ключа родительской сущности мигрируют в состав неключевых компонентов родительской сущности. Неидентифицирующая связь служит для связывания независимых сущностей.

Имя связи – фраза, характеризующая отношение между родительской и дочерней сущностями. Для связи один-ко-многим идентифицирующей или не идентифицирующей достаточно указать имя, характеризующее отношение от родительской к дочерней сущности.

Тип связи (идентифицирующая/неидентифицирующая). Для неидентифицирующей связи можно указать обязательность. В случае обязательной связи атрибут внешнего ключа получит признак NOT NULL, несмотря на то, что внешний ключ не войдет в состав первичного ключа дочерней сущности. В случае необязательной связи внешний ключ может принимать значение NULL. Необязательная неидентифицирующая связь помечается прозрачным ромбиком со стороны родительской сущности.

Правила ссылочной целостности – логические конструкции, которые выражают бизнес-правила использования данных и представляют собой правила вставки, замены и удаления.

Информацию о предметной области суммируют составлением спецификаций по сущностям, атрибутам и отношениям с использованием графических диаграмм, в чем и заключается процесс моделирование данных.

Порядок выполнения работы

Для запуска пакета Power Designer в меню программы (Windows) найдите папку Sybase и запустите файл Power Designer. Для создания концептуальной модели данных необходимо выбрать File/ New или на панели инструментов выбрать значок . Далее появится окно для выбора создаваемой модели (рис. 24), в котором надо выбрать Conceptual Data Model

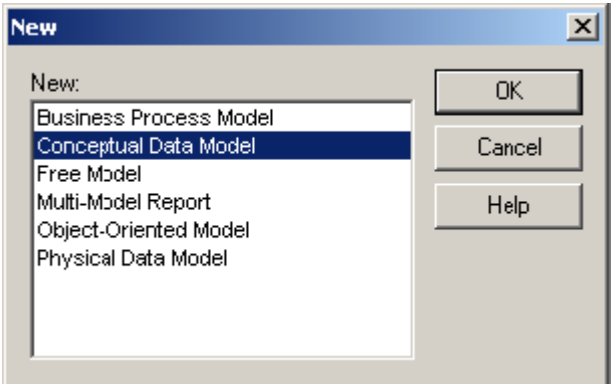

Рис. 24. Окно выбора модели.

После нажатия кнопки OK появиться окно, в котором создается ER-диаграмма.

Создание сущностей

Для создания сущности, в панели TOOLS (рис. 25) нажмите кнопку с белым прямоугольником (с подсказкой Entity).

Рис. 25. Панель элементов с выбранным элементом сущность

Далее, поместите указатель мыши на рабочее поле в нужном месте и щелкните кнопкой мыши. Прямоугольник, изображающий сущность появится в указанном месте. При этом, курсор мыши на рабочем поле выглядит как выбранный элемент, т.о. можно создавать несколь-

ко выбранных элементов одного типа без повторного их выбора на панели элементов.

Для того, чтобы изменить свойства созданной сущности, дважды щелкните на нее левой кнопкой мыши или нажмите правую кнопку и в выпавшем меню, выберите пункт Properties, в результате чего откроется окно свойств сущности (рис. 26) .

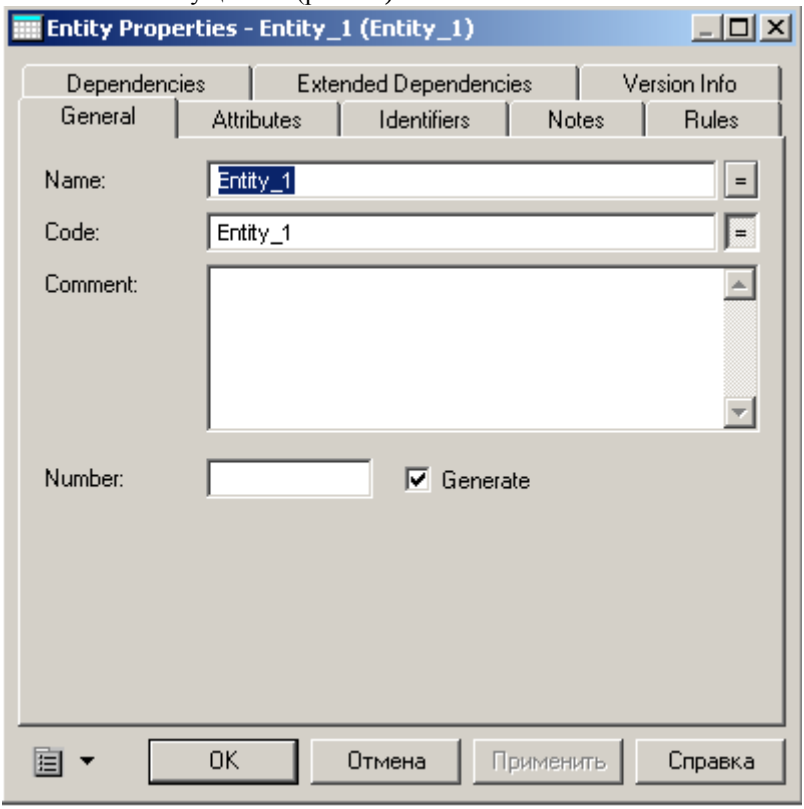

Рис. 26. Окно свойств сущности

В открывшемся окне восемь закладок.

Description и Annotation предназначены для словесного описания сущности (что улучшает понимание модели).

Закладка General позволяет ввести следующие параметры:

- Name — имя сущности, которое будет видеть пользователь;
- Code — имя кода сущности, которое будет использоваться при генерации физической модели;

- Number — ограничение количества записей в таблице после генерации физической модели;
- Comment — комментарий, предназначенный для улучшения понимания модели.

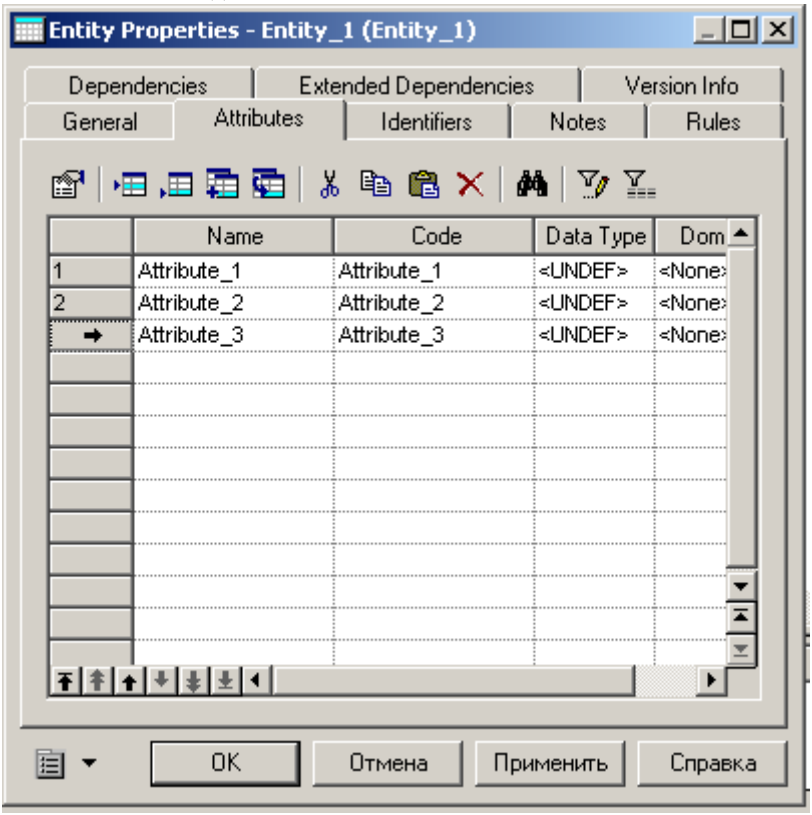

Рис. 27. Окно ввода атрибутов сущностей

Закладка Attributes содержит таблицу (рис. 27) и позволяет определять ***атрибуты*** сущности:

- Name — имя атрибута, которое будет видеть пользователь;
- Code — имя кода атрибута, которое будет использоваться при генерации физической модели;
- Data type — тип данных атрибута, который может быть выбран из выпадающего списка или вручную, щелкнув в поле Data type;

- Domain — принадлежность к домену, если он определен. Использование доменов позволяет, определив один раз пользовательский тип данных, использовать его в дальнейшем при определении типа данных атрибута. О создании домена будет сказано ниже;
- M (mandatory) — обязательный атрибут, указывает может ли данный атрибут принимать неопределенные значения (обязательно ли данное поле для заполнения в таблице БД);
- P (Primary Identifier) — первичный идентификатор сущности (в физической модели данных атрибут будет являться первичным ключом или его составной частью);
- D (Displayed) — отображаемый, т.е. будет ли атрибут показываться в модели.

Более полную информацию по свойствам атрибута можно получить, дважды щелкнув по полю, расположенному слева от поля с именем атрибута. Здесь можно вводить комментарий в поле Comment, задавать список значений для данного атрибута, определять верхние и нижние границы значений

Закладка Identifiers содержит автоматически заполняемую таблицу первичных идентификаторов сущности, но позволяет делать это вручную, когда необходимо создать суррогатный первичный идентификатор.

Закладка Rules позволяет вводить необходимые правила на ввод значений в таблицу.

Остальные закладки Notes, Version info носят описательный характер для улучшения понимания модели.

Для фиксации всех изменений в необходимо нажать кнопку Apply.

Домен

Домен – это множество допустимых значений атрибута определенного типа данных. Домен определяется заданием стандартного типа данных, к которому относятся элементы домена и заданием произвольного логического выражения, применяемому к этому типу данных.

Для создания домена необходимо в главном меню выбрав пункт Model, выбрать пункт Domains. На рис. 28 показано форма определения домена. Данная форма содержит аналогичные поля как для определения свойств атрибутов (в закладке Attributes в свойствах формы), а также дополнительные поля Length и Precision для описания длины и точности значений атрибутов.

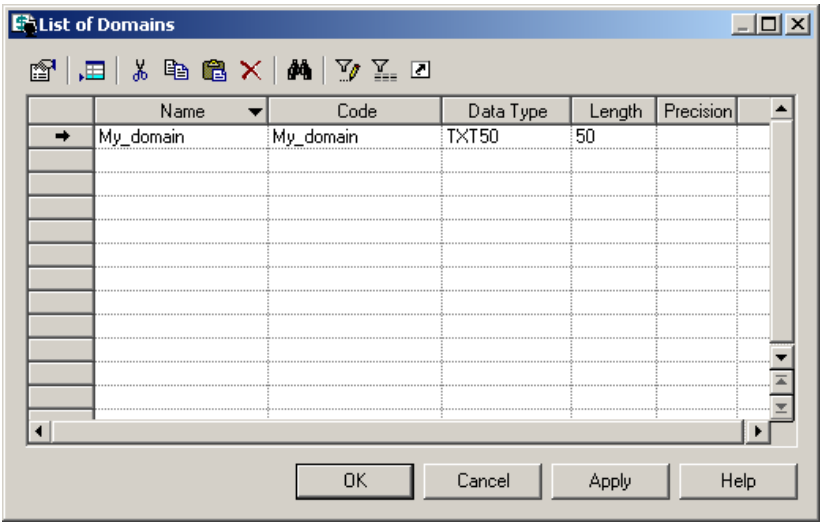

Рис. 28. Окно ввода списка доменов

После создания нового домена его имя появится в списке доменов при задании свойств атрибутов сущностей (рис. 29).

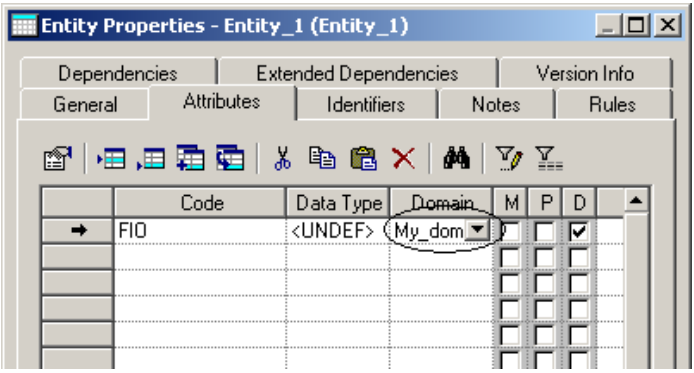

Рис. 29. Определение домена для атрибута сущности

При создании атрибутов сущностей в концептуальной модели не создаются атрибуты, являющиеся внешними ключами сущностей.

После того, как созданы все необходимые сущности и атрибуты, необходимо определить связи между ними.

Установка связей

Для установки связи между двумя сущностями, необходимо нажать кнопку (с двумя белыми прямоугольниками и линией между ними рис. 30).

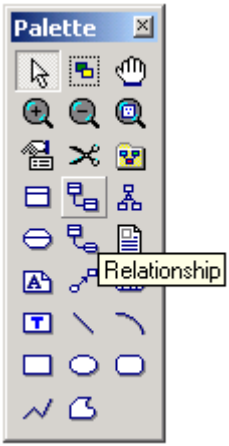

Рис. 30. Панель элементов с выбранным элементом связь

Необходимо перевести курсор мыши на одну сущность и, нажав левую кнопку мыши и, не отпуская ее, перевести курсор на вторую сущность. Далее можно отпустить кнопку мыши – связь установлена.

Для изменения свойств связи, необходимо дважды щелкнуть левой кнопкой мыши на линию связи (или нажать правую кнопку мыши и выпавшем меню выбрать пункт Properties). Откроется окно (рис. 31), с закладками:

- закладки Notes и Version info используются для подробного описания связи. В закладке Rules можно задавать параметры ограничения связи;
- в закладке General указывается имя и код связи, а две кнопки с именами используемых сущностей позволяют вызвать окно со свойствами соответствующей сущности.
- Закладка Detail позволяет указать вид связи (один–к–одному, один–ко–многим, многие–ко–многим и т.д.) и устанавливает свойства связи от Сущности1 к Сущности2 и наоборот:
- Mandatory определяет обязательность связи, показывая, что экземпляр Сущности1 (запись) может существовать

только при наличии соответствующего экземпляра в Сущности2;

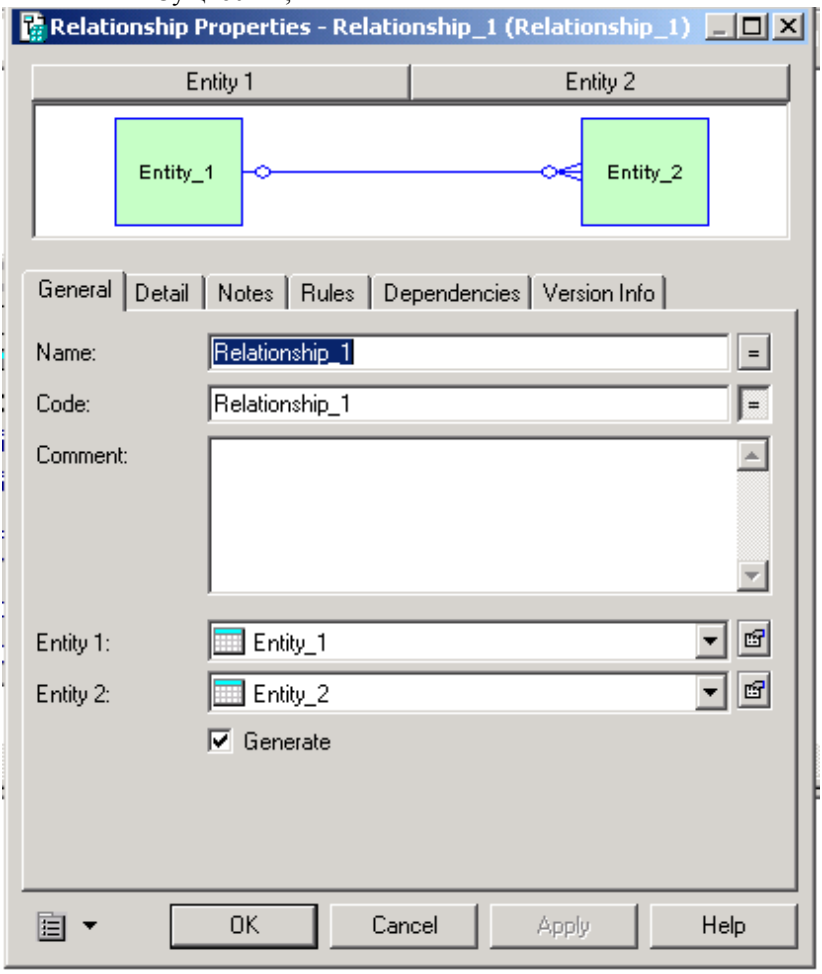

Рис. 31. Окно свойств связи

- Dependent показывает, что каждый экземпляр Сущности1 отождествляется с экземпляром в Сущности2 (первичный ключ на стороне «один» при создании физической модели войдет в состав первичного ключа в таблице на стороне «многие»);
- Role – текст, описывающий связь от Сущности1 к Сущности2.

Связи многие-ко-многим преобразуются в физической модели в промежуточные таблицы.

После того, как созданы все сущности, указаны атрибуты и установлены все связи необходимо проверить концептуальную правильность построения концептуальной модели. Для этого необходимо выбрать в меню Tools/Check Model (или нажать F4). Появится окно (рис. 32), в котором предлагается выбрать объекты для проверки.

Package – система проверит правильность циклических связей.

Domain – система проверит правильность заполнения доменов.

Data items – проверять ли атрибуты.

Entities – система проверит правильность создания сущностей.

Entity attributes – проверка правильности свойств сущности

Entity identifier - проверка правильности идентификаторов сущности.

Relationships – проверка связей.

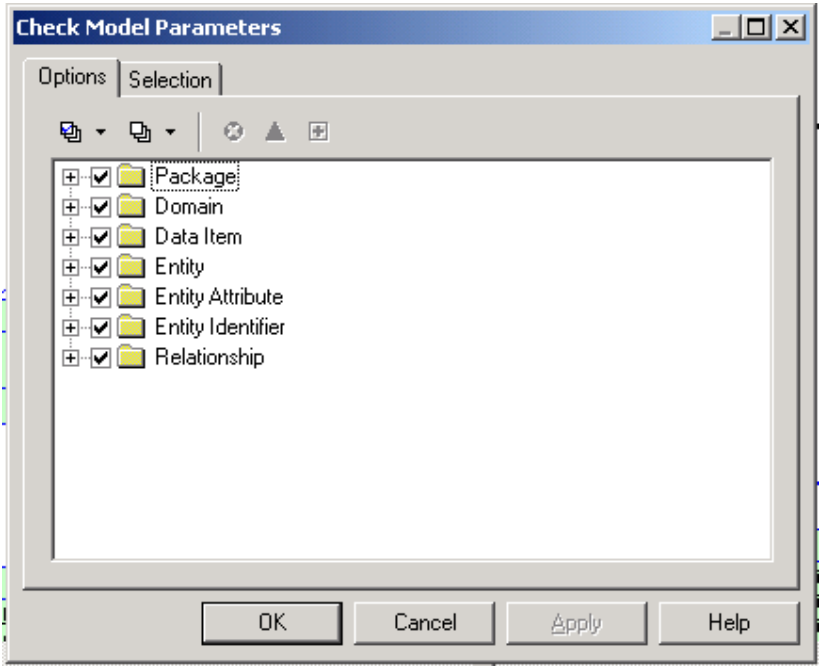

Рис. 32. Окно проверки концептуальной модели

После нажатия кнопки ОК система проверит всю концептуальную модель, выдаст ошибки (или предупреждения), если таковые имеются. Для просмотра сведений об ошибке необходимо дважды «щелкнуть» по ней кнопкой мыши.

Работа считается полностью выполненной, если при проверке модели не выдаются ошибки.

Лабораторная работа № 7 «Генерация физической модели и структуры базы данных»

Тема: Генерация физической модели предметной области.

Раздел дисциплины: Системы управления базами данных.

Цель работы: спроектировать физическую модель, выбранной ранее предметной области на основе созданной концептуальной модели в пакете Power Disigner.

Продолжительность: 4 часа.

Физическая модель данных

На основе спроектированной концептуальной модели создается физическая модель данных, свойственная для конкретной СУБД.

При формировании физической модели данных определяются внешние ключи в связываемых сущностях. Добавляются промежуточные таблицы связи, с целью исключения связей многие-ко-многим (M:M).

Порядок выполнения работы

После того, как проверка концептуальной модели закончится успешно, можно генерировать физическую модель. Для этого необходимо выбрать в меню Tools/Generate Physical Model. Откроется окно генерации физической модели (рис. 33).

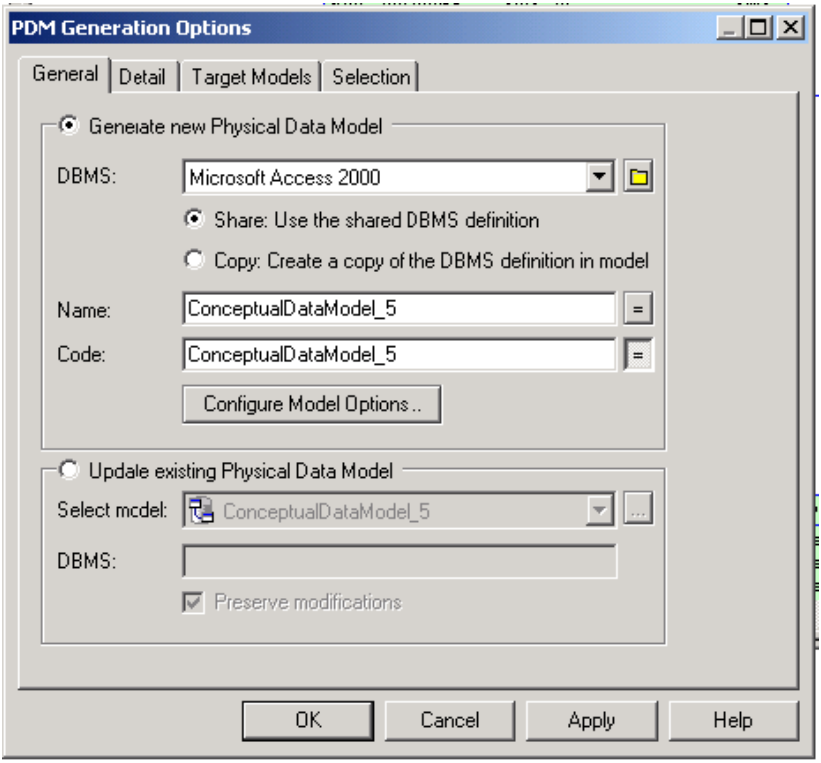

Рис. 33. Окно генерации физической модели

В этом окне необходимо выбрать опцию Generate new Phisical Data Model, и выберать в поле DBMS из выпадающего списка Microsoft Access 2000. *Name* – поле для ввода имени файла для дальнейшей генерации физической модели. Для задания свойств модели щелкните по кнопке Configure Model Options. Откроется окно свойств создаваемой физической модели (рис. 34).

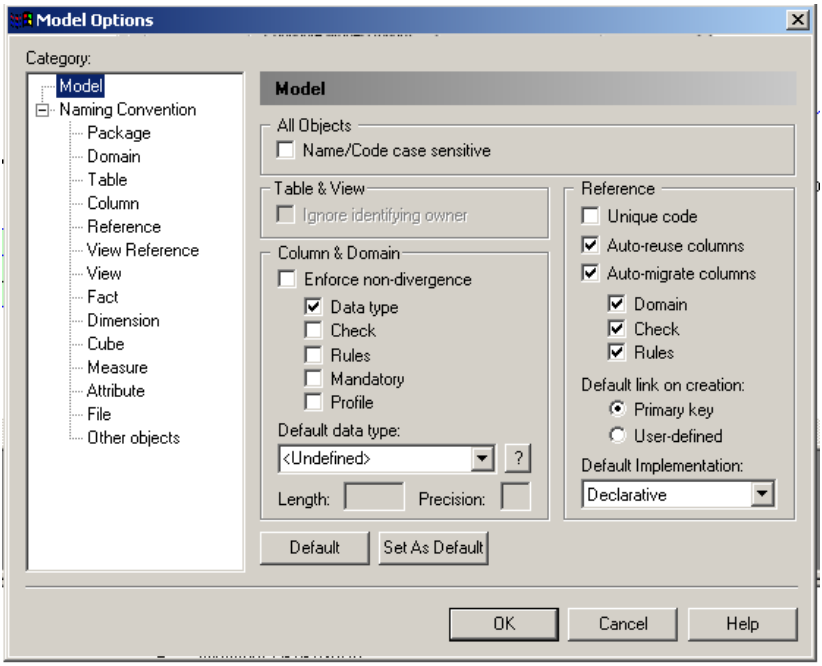

Рис. 34 Окно свойств создаваемой физической модели

Здесь можно выбрать следующие параметры создания физической модели: окно свойств создаваемой физической модели окно свойств создаваемой физической модели

- Data type – учитывать типы данных концептуальной модели при генерации физической модели;
- Check – проверка доменов и соответствие полей атрибутов с выбранным доменом.
- Rules – проверка правил на ввод значений в таблицу.
- Mandatory – учитывать свойства обязательности заполнения.
- Default data type – позволяет установить тип данных по умолчанию для всех не установленных типов данных в атрибутах.
- Auto-reuse columns – автоматическое повторное использование полей таблиц.
- Auto-migrate columns – автоматическая миграция полей таблиц, по которым будет происходить соединение.
- Domain – использовать ссылки на домен.

- Check – использовать ссылки на проверку ограничений.
- Rules – использовать ссылки на проверку правил заполнения таблиц.

В директории Naming convention можно задавать имена шаблонов таблиц, атрибутов и т.д.

Установив необходимые параметры, нажмите кнопку OK. Если концептуальная модель спроецирована корректно, то система создаст физическую модель базы данных для того типа СУБД, который был указан в параметрах.

Сгенерированную физическую модель также необходимо проверить, нажав клавишу F4

Появится окно (рис. 35), в котором предлагается выбрать объекты для проверки.

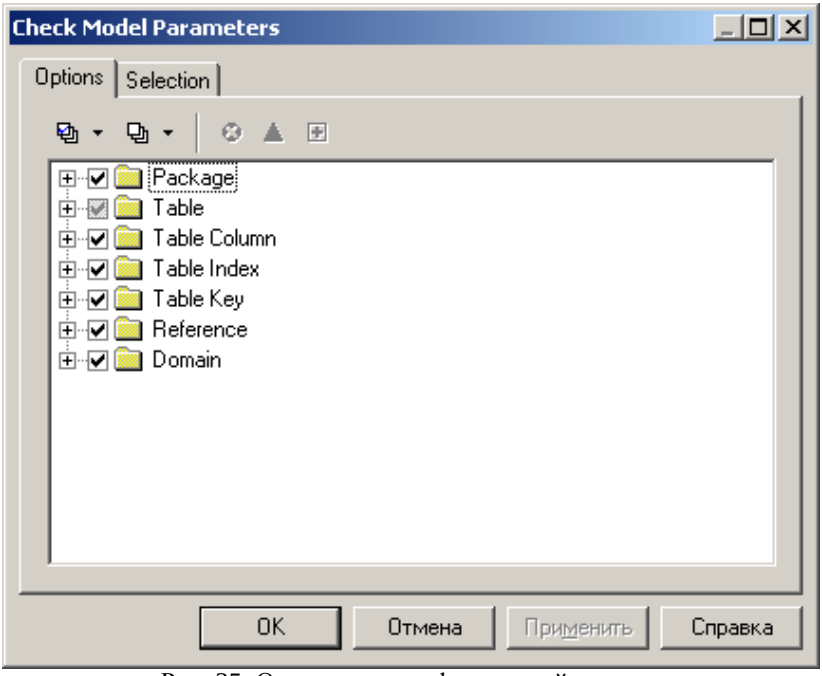

Рис. 35. Окно проверки физической модели

После нажатия кнопки OK система проверит всю физическую модель, выдаст ошибки (или предупреждения), если таковые имеются.

Если ошибки отсутствуют, на основе данной модели необходимо создать новую базу данных. Для создания базы данных в меню Data-

Base выберите пункт Generate DataBase. В появившемся окне (рис. 36) необходимо выбрать опцию ODBC generation, путь к файлу, в котором будет сохранен скрипт на создание БД (набор управляющих SQL-запросов), а также дополнительные характеристики БД, перейдя по вкладкам Keys&Index, Database, Options, Selection. После чего нажмите кнопку OK.

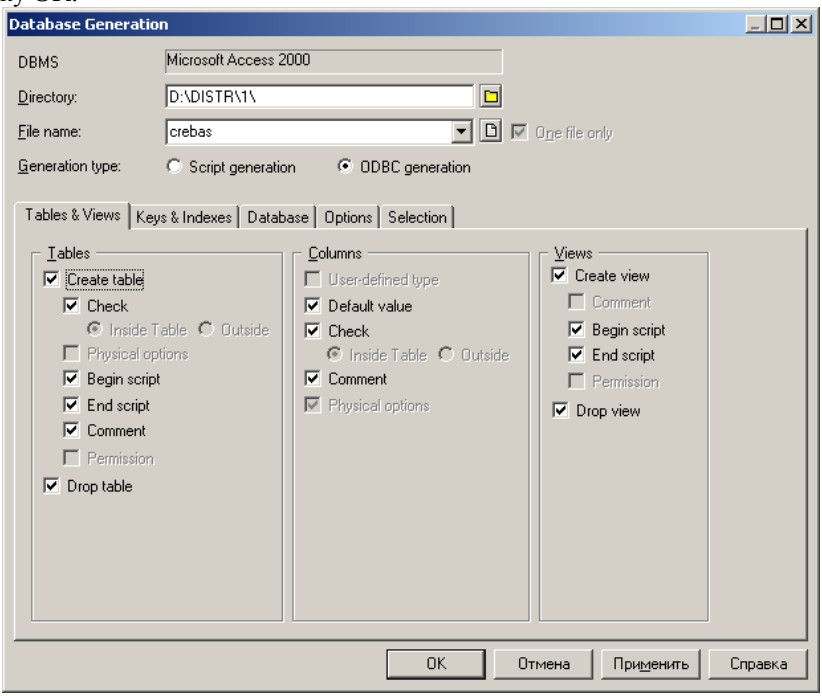

Рис. 36. Окно создания БД

Система попросит указать источник данных рис.37.

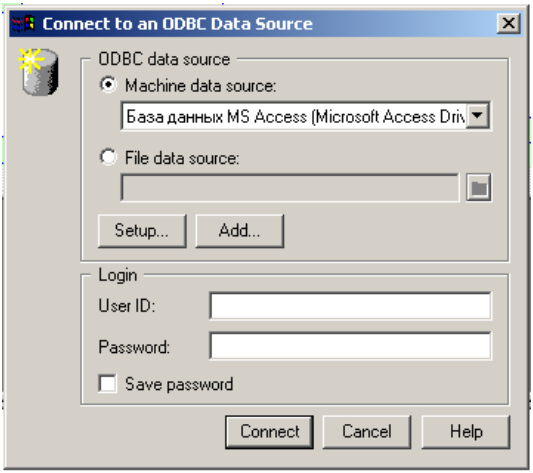

Рис. 37. Окно доступа к БД

Нажмите кнопку Add для создания собственного источника данных. В появившемся окне «Администратор источников данных ODBC» (рис. 38) нажмите кнопку Добавить.

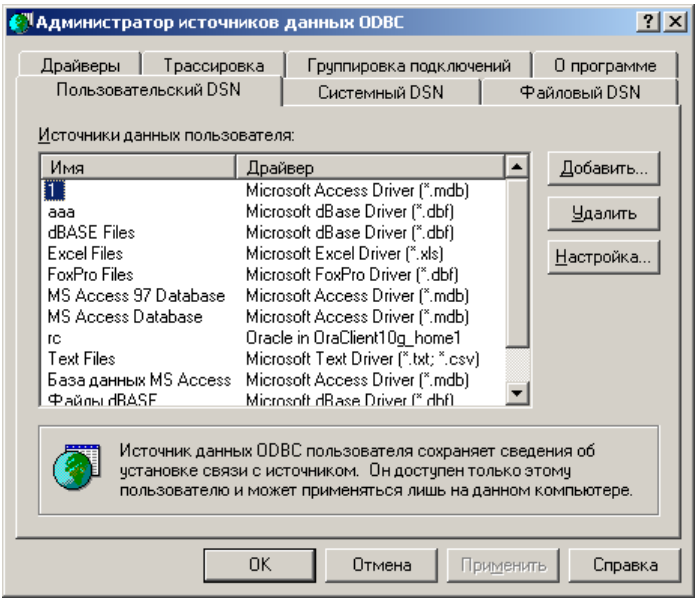

Рис. 38. Администратор источников данных ODBC

В появившемся окне (рис. 39) выберите соответствующий тип драйвера и нажмите кнопку Готово.

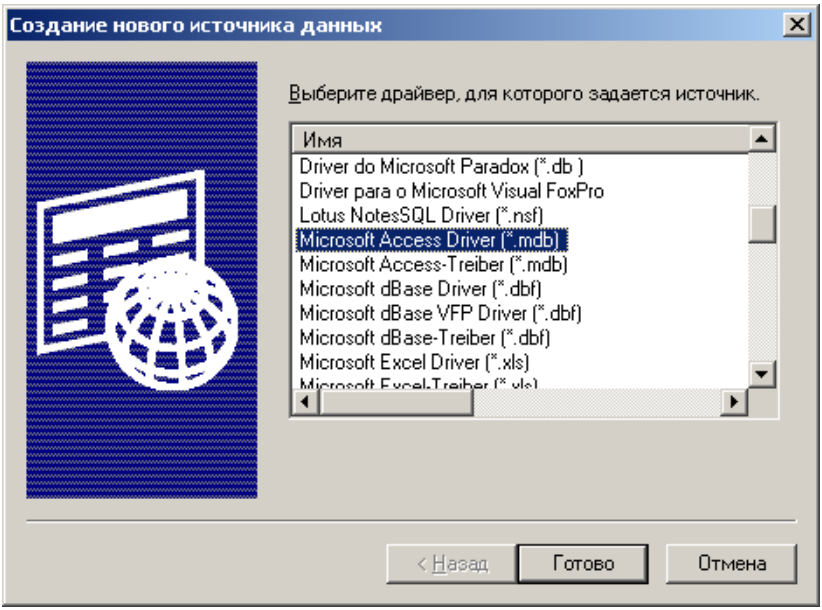

Рис. 39. Выбор драйвера

В появившемся окне (рис. 40) введите имя и описание источника данных и нажмите кнопку Создать из раздела Базы данных для создания нового файла БД или кнопку Выбрать для выбора созданного ранее mdb-файла, настроив тем самым доступ к mdb-файлу, в котором будет создана схема БД.

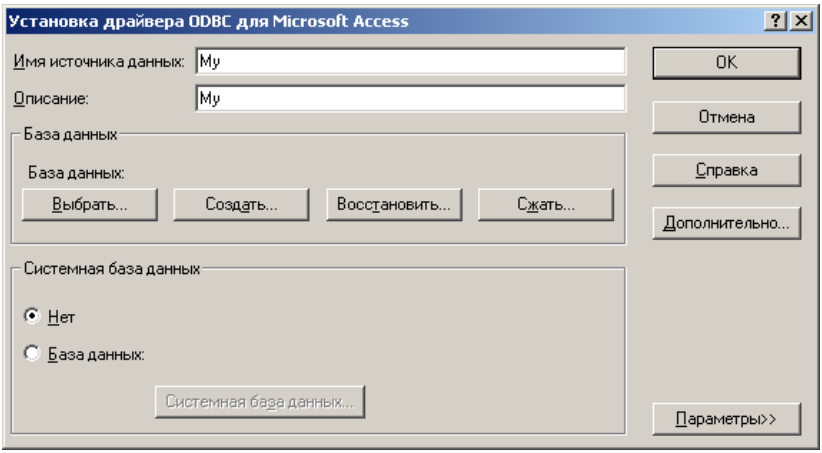

Рис. 40. Установка драйвера ODBC для MS Access

В окне Выбор базы данных (рис. 41) укажите имя базы данных и нажмите OK.

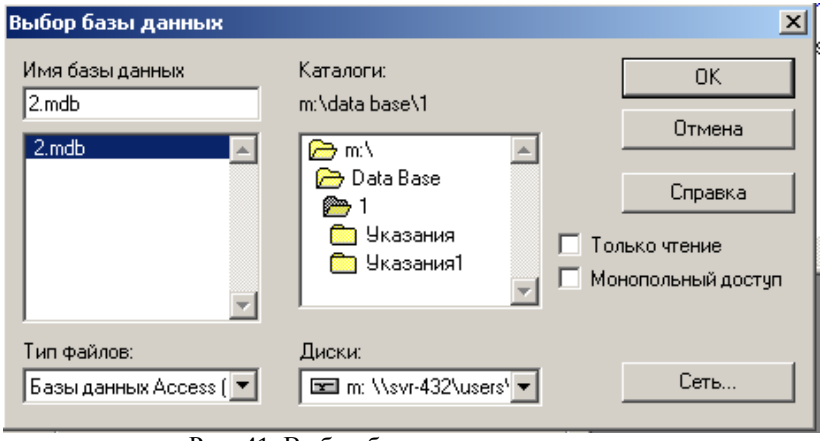

Рис. 41. Выбор базы данных.

Вернитесь в окно доступа к БД (рис. 37) и выберите созданный источник данных (рис. 42).

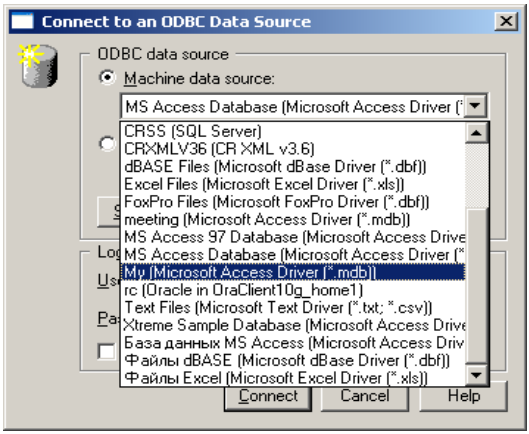

Рис. 42. Выбор своего источника данных

Нажмите кнопку Connect. В появившемся окне (рис 43) будет представлен SQL-скрипт, сгенерированный системой, запуск которого приведет к созданию схемы БД в выбранном mdb-файле.

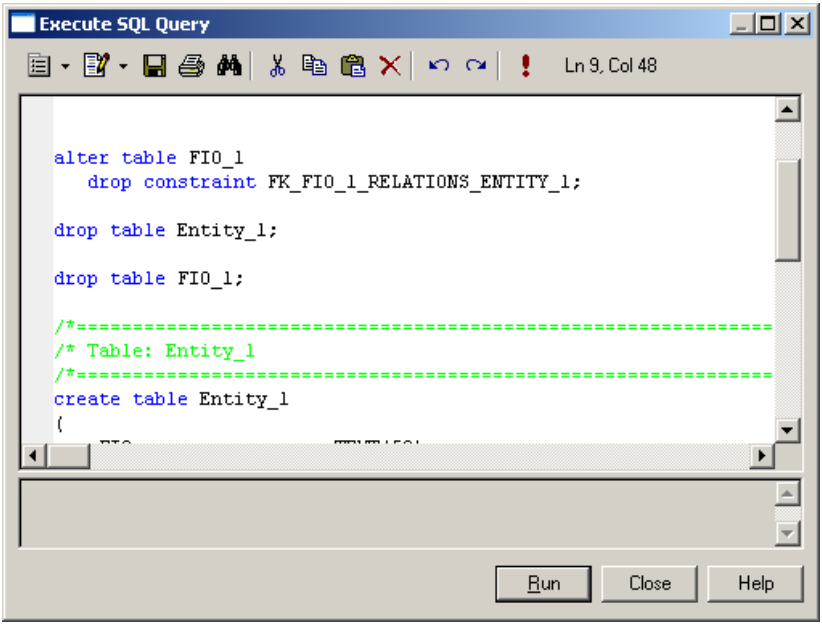

Рис. 43. Окно SQL-скрипта на создание схемы БД

Для запуска SQL-скрипта нажмите кнопку Run. Если система не выдаст сообщений об ошибках, откройте в среде MS Access созданную БД и проверьте ее соответствие физической модели.

Самостоятельная работа

Согласно рабочей программе отводится следующее количество часов на самостоятельную работу:

- подготовка к контрольным работам – 10 часов.
- подготовка к лабораторным работам – 34 часа.
- индивидуальное задание: Подготовка собственного примера нормализации отношений для выбранной предметной области – 10 часов.

Форма контроля и проверка достижения заявленных компетенций (ОК-8, ПК-17, ПК-27, ПК-46): проведение контрольных работ (в том числе тестовых), опрос перед проведением лабораторных работ, проверка отчетов, защита индивидуального задания путем представления презентации – выступление на лекции с демонстрацией примера нормализации.

Для проработки лекционного материала студентам, помимо конспектов лекций, рекомендуются следующие главы учебно-методического пособия [1] по разделам курса:

1. Обоснование концепции баз данных: глава 1.
2. Концепция модели данных: глава 2.
3. Реляционная модель: главы 3-4, глава 5.1.
4. Моделирование данных с помощью ER-диаграмм: глава 5.2, глава 5.3.
5. Реляционные языки: глава 6.
6. Системы управления базами данных: глава 8.

Для подготовки к лабораторным работам рекомендуется повторить соответствующие тематике разделы учебно-методического пособия [1], а также ознакомиться с порядком выполнения лабораторных работ, по настоящему руководству.

Для изучения тем теоретической части курса, отводимых на самостоятельную проработку, рекомендуется ознакомление со всеми разделами [1]. Кроме того, рекомендуется повторить разделы предложенной литературы [2-6], посвященные проектированию данных и построению пользовательских приложений.

Для выполнения индивидуального задания необходимо:

- выбрать предметную область и провести ее анализ на предмет выявления основных объектов и их характеристик;

- создать ненормализованной по 1НФ отношение, содержащее необходимый минимум информации о данной предметной области;
- заполнить значение атрибутов исходного отношения;
- определить первичный ключ исходного отношения;
- провести нормализацию по 1НФ;
- проверить соблюдения целостности сущности;
- определить функциональные зависимости, а также возможные аномалии и коллизии;
- провести последовательную нормализацию отношения до 3НФ;
- заполнить значения атрибутов результирующих отношений;
- определить первичные ключи и функциональные зависимости результирующих отношений;
- доказать, что отношения нормализованы по 3НФ и свободны от выявленных ранее коллизий и аномалий.

Рекомендуемая литература

1. Сенченко, П.В. Организация баз данных: Учебное пособие/ П. В. Сенченко; Федеральное агентство по образованию, Томский государственный университет систем управления и радиоэлектроники. – Томск: ТУСУР, 2004. - 184 с.: ил.. - Библиогр.: с. 183-184. - ISBN 5-86889-224-0: УДК 681.3.016(075.8) (гриф СИБРУМЦ) (Наличие в библиотеке ТУСУР: экземпляры всего: 34, из них: анл: 6 , счз1: 1 , счз5: 1, аул: 26.)

2. Дейт К. Дж. Введение в системы баз данных: Пер. с англ./ К. Дж. Дейт. - 6-е изд. - Киев; М.: Диалектика, 1998. - 784 с.: ил. - (Системное программирование). - (в пер.): Б.ц. (наличие в библиотеке ТУСУР: АНЛ – 1 экз.)

3. Саймон, Алан Р. Стратегические технологии баз данных: менеджмент на 2000 год: Пер. с англ./ Алан Р. Саймон; Ред. М. Р. Когаловский, Пер. М. Р. Когаловский, Пер. Н. И. Вьюкова, Пер. Г. Т. Никитина. - М.: Финансы и статистика, 1999. - 480 с.: ил. (наличие в библиотеке ТУСУР: счз1(1), счз5(1))

4. Кузнецов С.Д. Основы современных баз данных. Информационно-аналитические материалы Центра Информационных технологий. М.– режим доступа к сайту http://citforum.ru/database/osbd/contents.shtml свободный (дата обращения: 28.02.2012)

5. Кириллов В.В. Основы проектирования реляционных баз данных. Учебное пособие режим доступа к сайту http://citforum.ru/database/dbguide/index.shtml свободный (дата обращения: 28.02.2012)

6. Сенченко П.В. Слайды-презентации по дисциплине "Организация баз данных" / "Базы данных", 2012. 270 сл. (учебный сервер каф. АОИ: диск о:\2 курс\Базы данных\Слайды.pdf)

Электронные варианты УМПО находятся в открытом доступе в компьютерных классах.

Для организации работы студентов требуется свободный доступ в компьютерные классы с наличием ОС Windows, MS Office, СУБД MS Access.

Необходимые базы данных, информационно-справочные и поисковые системы: образовательный портал университета (http://edu.tusur.ru), электронный каталог библиотеки http://lib.tusur.ru); электронные информационно-справочные ресурсы вычислительных залов кафедры АОИ.