МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ МОРДОВИЯ

Государственное автономное профессиональное образовательное учреждение Республики Мордовия
«Саранский автомеханический техникум»

У Т В Е Р Ж Д А Ю
Заместитель директора по УМР
ГАПОУ РМ «Саранский
автомеханический техникум»
_____ Е.С. Синичкина
«31» августа 2022 г.

Методические рекомендации по выполнению практических работ ОП.08. Основы проектирования баз данных

09.02.07 Информационные системы и программирование

2022

СОДЕРЖАНИЕ

Пояснительная записка

Методические указания по выполнению лабораторных работ по учебной дисциплине «Основы проектирования баз данных» разработаны в соответствии с рабочей программой учебной дисциплины и предназначены для приобретения необходимых практических навыков и закрепления теоретических знаний, полученных обучающимися при изучении учебной дисциплины, обобщения и систематизации знаний перед экзаменом.

Методические указания предназначены для обучающихся специальности 09.02.07 Информационные системы и программирование.

Учебная дисциплина «Основы проектирования баз данных» является общеобразовательной учебной дисциплиной, изучается на 2 курсе и при ее изучении отводится значительное место выполнению практических работ.

Освоение содержания учебной дисциплины «Основы проектирования баз данных» во время выполнения практических работ обеспечивает достижение обучающимися следующих **результатов:**

Код ПК, ОК	**Умения**	**Знания**
ОК 1, ОК 2, ОК 4, ОК 5, ОК 9, ОК 10, ПК 11.1-11.6	проектировать реляционную базу данных; использовать язык запросов для программного извлечения сведений из баз данных	основы теории баз данных; модели данных; особенности реляционной модели и проектирование баз данных; изобразительные средства, используемые в ER-моделировании; основы реляционной алгебры; принципы проектирования баз данных; обеспечение непротиворечивости и целостности данных; средства проектирования структур баз данных; язык запросов SQL

В соответствии с рабочей программой учебной дисциплины «Основы проектирования баз данных» практические работы сгруппированы в конце четвертого семестра. Целесообразность данной группировки обусловлена необходимостью обобщения и систематизации знаний перед экзаменом.

Рабочая программа учебной дисциплины предусматривает проведение практических работ в объеме 58 часов.

Порядок выполнения практической работы

- записать название работы, ее цель в тетрадь;
- выполнить основные задания в соответствии с ходом работы;

□ выполнить индивидуальные задания.

Рекомендации по оформлению практической работы

Задания выполняются обучающимися по шагам. Необходимо строго придерживаться порядка действий, описанного в практической работе

Результаты выполнения практических заданий необходимо сохранять в своей папке на компьютере или USB – накопителе.

В случае пропуска занятий обучающийся осваивает материал самостоятельно в свободное от занятий время и сдает практическую работу с пояснениями о выполнении.

Критерии оценки практической работы

- наличие цели выполняемой работы, выполнение более половины основных заданий (удовлетворительно);
- наличие цели выполняемой работы, выполнение всех основных и более половины дополнительных заданий (хорошо);
- наличие цели выполняемой работы, выполнение всех основных и индивидуальных заданий (отлично).

Практическая работа № 1 «Преобразование реляционной БД в сущности и связи»

Цель работы: выработать практические навыки моделирования предметной области и построении ER-модели данных, закрепить технологию проектирования БД, закрепить основные понятия теории реляционных баз данных, освоить технологию построения ER-диаграмм, научиться получать реляционные БД из ER-диаграмм

Сущность-связь

Работа с базой данных начинается с построения модели. Наиболее распространенной является ER-модель (entity-relationship model) - модель "Сущность-связь". Для "ручного" построения ER-модели на практике будем использовать простую систему обозначений, предложенную Питером Ченом (обозначения, встречающиеся в разных источниках, могут отличаться от нижеприведенных):

Рисунок 1.1. Сущность - связь

Первичный ключ - атрибут или группа атрибутов, однозначно идентифицирующих объект. Первичный ключ может состоять из нескольких атрибутов, тогда подчеркивается каждый из них.

Связи между объектами могут быть 3-х типов: *Один - к одному*. Этот тип связи означает, что каждому объекту первого вида соответствует не более одного объекта второго вида, и наоборот. Например: сотрудник может руководить только одним отделом, и у каждого отдела есть только один руководитель. *Один - ко многим.* Этот тип связи означает, что каждому объекту первого вида может соответствовать более одного объекта второго вида, но каждому объекту второго вида соответствует не более одного объекта первого вида. Например: в каждом отделе может быть множество сотрудников, но каждый сотрудник работает только в одном отделе. *Многие - ко многим.* Этот тип связи означает, что каждому объекту первого вида может соответствовать более одного объекта второго вида, и наоборот. Например: каждый счет может включать множество товаров, и каждый товар может входить в разные счета.

Реляционная структура данных

В конце 60-х годов появились работы, в которых обсуждались возможности применения различных табличных даталогических моделей данных. Наиболее значительной из них была статья сотрудника фирмы IBM д-ра Эдварда Кодда (Codd E.F., A Relational Model of Data for Large Shared Data Banks. CACM 13: 6, June 1970), где впервые был применен термин «реляционная модель данных».

Будучи математиком по образованию, Э. Кодд предложил использовать для обработки данных аппарат теории множеств (объединение, пересечение, разность, декартово произведение). Он показал, что любое представление данных сводится к совокупности двумерных таблиц особого вида, известного в математике как *отношение* – relation.

Наименьшая единица данных реляционной модели – это отдельное атомарное (неразложимое) для данной модели значение данных.

Так, в одной предметной области фамилия, имя и отчество могут рассматриваться как единое значение, а в другой – как три различных значения.

Доменом называется множество атомарных значений одного и того же типа. Так, на рис. домен пунктов отправления (назначения) – множество названий населенных пунктов, а домен номеров рейса – множество целых положительных чисел.

Таблица 1.1. Рейс

Номер рейса	Дни недели	Пункт отправления	Время вылета	Пункт назначения	Время прибытия	Тип самолета	Стоимость билета
138	2_4_7	Баку	21.12	Москва	0.52	ИЛ-86	115.00
57	3_6	Ереван	7.20	Киев	9.25	ТУ-154	92.00
1234	2_6	Казань	22.40	Баку	23.50	ТУ-134	73.50
242	1 по 7	Киев	14.10	Москва	16.15	ТУ-154	57.00

86	2_3_5	Минск	10.50	Сочи	13.06	ИЛ-86	78.50
137	1_3_6	Москва	15.17	Баку	18.44	ИЛ-86	115.00
241	1 по 7	Москва	9.05	Киев	11.05	ТУ-154	57.00
577	1_3_5	Рига	21.53	Таллин	22.57	АН-24	21.50
78	3_6	Сочи	18.25	Баку	20.12	ТУ-134	44.00
578	2_4_6	Таллин	6.30	Рига	7.37	АН-24	21.50

Смысл доменов состоит в следующем. Если значения двух атрибутов берутся из одного и того же домена, то, вероятно, имеют смысл сравнения, использующие эти два атрибута (например, для организации транзитного рейса можно дать запрос «Выдать рейсы, в которых время вылета из Москвы в Сочи больше времени прибытия из Архангельска в Москву»). Если же значения двух атрибутов берутся из различных доменов, то их сравнение, вероятно, лишено смысла: стоит ли сравнивать номер рейса со стоимостью билета?

Отношение на доменах D1, D2, ..., Dn состоит из заголовка и тела. На рис. 3.4 приведен пример отношения для расписания движения самолетов (таблица 1). Ai - атрибуты, Vi - значения атрибутов.

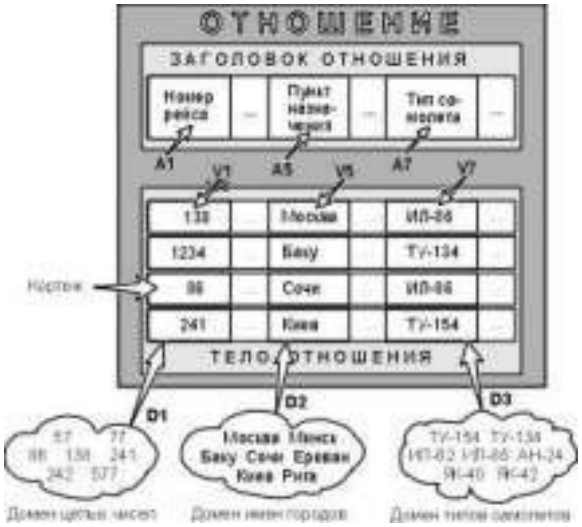

Рисунок 1.2. Отношение с математической точки зрения

Заголовок отношения состоит из такого фиксированного множества атрибутов A1, A2, ..., An, что существует взаимно однозначное соответствие между этими атрибутами Ai и определяющими их доменами Di (i=1,2,...,n).

Тело отношения состоит из меняющегося во времени множества кортежей, где каждый кортеж состоит в свою очередь из множества пар атрибут-значение (Ai:Vi), (i=1,2,...,n), по одной такой паре для каждого атрибута Ai в заголовке.

Для любой заданной пары атрибут-значение (Ai:Vi) Vi является значением из единственного домена Di, который связан с атрибутом Ai.

Степень отношения – это число его атрибутов.

Отношение степени один называют унарным, степени два – бинарным, степени три – тернарным, ..., а степени n – n-арным. Степень отношения «Рейс» (таблица 1) равна 8.

Кардинальное число или мощность отношения – это число его кортежей.

Мощность отношения «Рейс» равна 10. Кардинальное число отношения изменяется во времени в отличие от его степени.

Поскольку отношение – это множество, а множества по определению не содержат совпадающих элементов, то никакие два кортежа отношения не могут быть дубликатами друг друга в любой произвольно заданный момент времени.

Пусть R – отношение с атрибутами A1, A2, ..., An. Говорят, что множество атрибутов K=(Ai, Aj, ..., Ak) отношения R является возможным ключом R тогда и только тогда, когда удовлетворяются два независимых от времени условия:

Уникальность: в произвольный заданный момент времени никакие два различных кортежа R не имеют одного и того же значения для Ai, Aj, ..., Ak.

Минимальность: ни один из атрибутов Ai, Aj, ..., Ak не может быть исключен из K без нарушения уникальности.

Каждое отношение обладает хотя бы одним возможным ключом, поскольку по меньшей мере комбинация всех его атрибутов удовлетворяет условию уникальности. Один из возможных ключей (выбранный произвольным образом) принимается за его первичный ключ. Остальные возможные ключи, если они есть, называются альтернативными ключами.

Вышеупомянутые и некоторые другие математические понятия явились теоретической базой для создания реляционных СУБД, разработки соответствующих языковых средств и программных систем, обеспечивающих их высокую производительность, и создания основ теории проектирования баз данных.

Также на практике широко используются неформальные эквиваленты этих понятий: Отношение – Таблица, Кортеж – Строка таблицы или Запись, Атрибут – Столбец Таблицы или Поле.

При этом принимается, что «запись» означает «экземпляр записи», а «поле» означает «имя и тип поля».

Задание для практической работы

Задание 1. Проектирование реляционных Баз Данных.

Вариант 1.

Построить реляционную таблицу Базы Данных имен родственников студентов вашей группы, содержащую данные об именах родителей, братьев и сестер студентов.

Вариант 2.

Построить реляционную таблицу Базы Данных домов, где живут студенты вашей группы, содержащую данные о районе расположения дома, количестве этажей в нем и номере этажа, где живет студент.

Вариант 3.

Построить реляционную таблицу Базы Данных дней рождения студентов вашей группы, содержащую данные о дате рождения, знаке зодиака и годе по Китайскому календарю.

Вариант 4.

Построить реляционную таблицу Базы Данных Сотовых телефонов студентов вашей группы, содержащую данные о модели телефона, типе корпуса, операторе.

Технология выполнения работы и оформление отчета

1.1. Придумайте заголовок отношения и запишите его в отчет.

1.2. Определите атрибуты отношения. Начертите сетку таблицы в отчет и занесите в нее атрибуты.

1.3. Опросите студентов вашей группы и занесите полученные данные в таблицу.

1.4. На чертеже таблицы укажите чему соответствуют понятия: Заголовок отношения, тело отношения, атрибут отношения, кортеж отношения.

1.5. Определите и запишите в отчет степень отношения и мощность отношения.

1.6. Дайте определение первичного ключа. Укажите Первичный ключ получившегося отношения

1.7. Докажите, что у вас получилась реляционная таблица, для этого укажите типы данных всех атрибутов.

Задание 2. Проектирование Баз Данных. ER-диаграммы.

Формулировка задания. По описанию предметной области построить логическую модель БД методом ER-диаграмм, на основании которой построить набор таблиц БД.

Вариант 1.

Описание предметной области (Ресторан).

Посетители ресторана обслуживаются за столиками. За одним столом может располагаться не более 4 посетителей, каждый из которых может сделать заказ тех или иных блюд. Столики обслуживают официанты. У одного официанта в обслуживании несколько столов.

Задачи для БД:

- Есть ли свободные столы?
- Сколько посетителей обслужил официант за смену?
- Сколько каких блюд было реализовано?

Вариант 2.

Описание предметной области (Колледж).

Студенты колледжа объединены в группы. Набор дисциплин, изучаемых студентом, зависит от номера группы в которой он учится. Преподаватели читают дисциплины и выставляют зачеты студентам. Один преподаватель может читать несколько дисциплин, но каждую дисциплину ведет один преподаватель.

Задачи для БД:

- Какие дисциплины изучает студент?
- Какая оценка у студента по данной дисциплине?
- Кто выставил эту оценку?

Вариант3.

Описание предметной области (Театральная касса).

В театральной кассе продаются билеты на спектакли. Стоимость билета зависит от ряда, театра и спектакля. Каждый день в театре может идти не более одного спектакля. Спектакль характеризуется названием и автором. Каждый покупатель может купить сколько угодно билетов на любые спектакли.

Задачи для БД:

- Какие спектакли идут в определенный день?
- Есть ли билеты на конкретный спектакль?
- Сколько стоит конкретный билет?

Вариант 4.

Описание предметной области (Грузоперевозки).

АТП имеет грузовые автомобили с гос. номерами и организует перевозки для своих заказчиков. Стоимость перевозки зависит от расстояния и грузоподъемности автомобиля, который ее выполняет. Каждый заказчик может сделать заказ нескольких перевозок. Одну перевозку выполняет один грузовик.

Задачи для БД:

- Какие грузовики свободны?
- Какой заказчик сделал самый дорогой заказ?
- Какой грузовик выполнил наибольшее количество заказов?

Технология выполнения работы 1.Построение ER-диаграммы.

1.1. Выберите из описания предметной области все существительные. Продумайте, какие из них будут соответствовать сущностям, а какие атрибутам сущностей. Зарисуйте в отчет все сущности с их атрибутами согласно обозначениям, принятым в ER-диаграммах.

1.2. На рисунке подчеркиванием атрибутов обозначьте для каждой сущности уникальный идентификатор (Ключ). При необходимости добавьте сущностям атрибуты, которые помогут однозначно отличить каждый экземпляр сущности.

1.3. Определите и включите в схему связи сущностей. Подпишите названия связей и пронумеруйте связи. Для первой связи укажите тип и модальность. Для всех связей запишите их прочтение слева направо и справа налево.

1.4. Если в схеме присутствуют связи типа «много-со-многими» уберите их путем ввода дополнительной сущности. Измененную схему зарисуйте в отчет.

2. Получение реляционной схемы из ER-диаграммы.

2.1. Каждая сущность превращается в таблицу. Имя сущности – имя таблицы. Набор всех таблиц – БД. Вспомните, что такое схема БД. Запишите схему вашей БД в отчет.

2.2. Зарисуйте все полученные таблицы с их заголовками и названиями столбцов. Выделите потенциальные и внешние ключи (если есть) для каждой таблицы. Укажите столбцы, допускающие неопределенные значения.

2.3. Докажите, что полученные отношения находятся в Первой нормальной форме.

Практическая работа № 2 «Нормализация реляционной БД, освоение принципов проектирования БД»

Цель работы: выработать практические навыки моделирования предметной области и построения различных видов модели баз данных

Нормализация, функциональные и многозначные зависимости

Нормализация – это разбиение таблицы на две или более, обладающих лучшими свойствами при включении, изменении и удалении данных. Окончательная цель нормализации сводится к получению такого проекта базы данных, в котором каждый факт появляется лишь в одном месте, т.е. исключена избыточность информации. Это делается не столько с целью экономии памяти, сколько для исключения возможной противоречивости хранимых данных.

Каждая таблица в реляционной БД удовлетворяет условию, в соответствии с которым в позиции на пересечении каждой строки и столбца таблицы всегда находится единственное атомарное значение, и никогда не может быть множества таких значений. Любая таблица, удовлетворяющая этому условию, называется нормализованной. Фактически, ненормализованные таблицы, т.е. таблицы, содержащие повторяющиеся группы, даже не допускаются в реляционной БД.

Всякая нормализованная таблица автоматически считается таблицей в первой нормальной форме, сокращенно 1НФ. Таким образом, строго говоря, "нормализованная" и "находящаяся в 1НФ" означают одно и то же. Однако на практике термин "нормализованная" часто используется в более узком смысле – "полностью нормализованная", который означает, что в проекте не нарушаются никакие принципы нормализации.

Теперь в дополнение к 1НФ можно определить дальнейшие уровни нормализации – вторую нормальную форму (2НФ), третью нормальную форму (3НФ) и т.д.

По существу, таблица находится в 2НФ, если она находится в 1НФ и удовлетворяет, кроме того, некоторому дополнительному условию, суть которого будет рассмотрена ниже. Таблица находится в 3НФ, если она находится в 2НФ и, помимо этого, удовлетворяет еще другому дополнительному условию и т.д.

Таким образом, каждая нормальная форма является в некотором смысле более ограниченной, но и более желательной, чем предшествующая. Это связано с тем, что "(N+1)-я нормальная форма" не обладает некоторыми непривлекательными особенностями, свойственным "N-й нормальной форме". Общий смысл дополнительного условия, налагаемого на (N+1)-ю нормальную форму по отношению к N-й нормальной форме, состоит в исключении этих непривлекательных особенностей.

За время развития технологии проектирования реляционных БД были выделены следующие нормальные формы:

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

Обычно на практике применение находят только первые три нормальные формы.

Теория нормализации основывается на наличии той или иной зависимости между полями таблицы. Определены два вида таких зависимостей: функциональные и многозначные.

Определение. Функциональная зависимость. Поле B таблицы функционально зависит от поля A той же таблицы в том и только в том случае, когда в любой заданный момент времени для каждого из различных значений поля A обязательно существует только одно из различных значений поля B. Отметим, что здесь допускается, что поля A и B могут быть составными.

Другими словами, в отношении R атрибут Y функционально зависит от атрибута X в том и только в том случае, если каждому значению X соответствует одно значение Y.

Схематично функциональную зависимость атрибута Y от атрибута X изображают гак:

R.X -> R.Y

R(X -> Y).

ФЗ(X -> Y)

Пример 1.

1. В таблице Блюда (б.д. Пансион) поля Блюдо и В функционально зависят от ключа БЛ.
2. Таблица Поставщики вида:

Таблица 2.1. Поставщики

Поставщик	Статус	Город	Страна	Адрес	Телефон

В таблице Поставщики поле Страна функционально зависит от составного ключа (Поставщик, Город). Однако последняя зависимость не является функционально полной, так как Страна функционально зависит и от части ключа – поля Город. Отсюда определение:

Определение. Полная функциональная зависимость. Поле B находится в полной функциональной зависимости от составного поля A, если оно функционально зависит от A и не зависит функционально от любого подмножества поля A. Пример нормализации

Постановка задачи. Дано отношение.

1) определить первичный ключ отношения и все функциональные зависимости отношения;

2) привести отношение к 3НФ, указать первичные и внешние ключи полученных отношений, построить схему "Таблица-Связь".

R = {НаименованиеЭмитента, ТипЦБ, ДатаЭмиссии, НоминальнаяСтоимость}.

Таблица 2.2. Эмитенты

Наименование Эмитента	ТипЦБ	Дата Эмиссии	Номинальная Стоимость
ОАО ―КрАЗ‖	акция обыкновенная	23.06.1999	100 руб.
ОАО ―КрАЗ‖	акция обыкновенная	23.06.1999	200 руб.
ТОО ―Искра‖	акция привилегирванная	20.06.1999	500 руб.
ТОО ―Искра‖	акция привилегированная	23.06.1999	500 руб.

Решение

1. Функциональные зависимости:

<ДатаЭмиссии, НоминальнаяСтоимость> -> <НаименованиеЭмитента, ТипЦБ>

ТипЦБ ->НаименованиеЭмитента

Первичный ключ отношения R состоит из двух атрибутов:

<ДатаЭмиссии, НоминальнаяСтоимость>.

Таким образом, существует функциональная зависимость между неключевыми атрибутами отношения R, т.е. отношение R не находится в 3НФ.

2. Приведение отношения R к 3НФ состоит в декомпозиции (разбиении отношения R на два отношения):

R1= {НаименованиеЭмитента, ТипЦБ}, где Функциональные зависимости: ТипЦБ -> НаименованиеЭмитента, первичный ключ – атрибут ТипЦБ,

R2={ТипЦБ, ДатаЭмиссии, НоминальнаяСтоимость}, Функциональные зависимости:

<ДатаЭмиссии, НоминальнаяСтоимость> -> ТипЦБ, составной первичный ключ: <ДатаЭмиссии, НоминальнаяСтоимость>, внешний ключ: ТипЦБ.

3. Схема "Таблица-Связь":

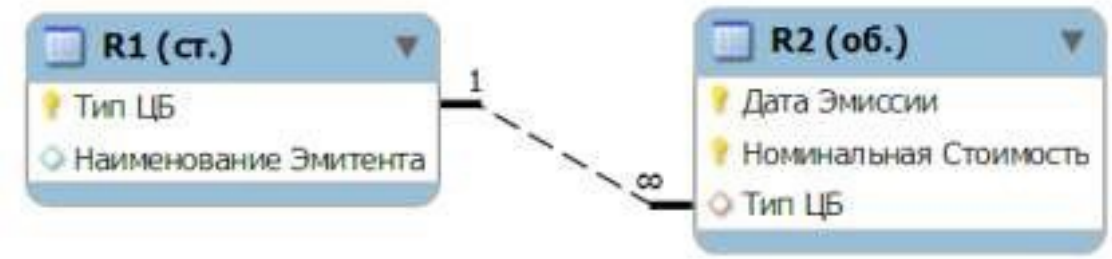

Рисунок 2.1. Схема «Таблица – связь»

Задание для практической работы

Дан фрагмент отношения (таблицы). Предполагается, что функциональные зависимости, имеющиеся во фрагменте, распространяются на все отношение (таблицу). Для вашего варианта:

1. Определить первичный ключ отношения и все функциональные зависимости отношения.
2. Привести отношение к 3НФ, указать первичные и внешние ключи полученных отношений, построить схему "Таблица - Связь".

Индивидуальные задания к практической работе Вариант 1.

Область	Тип	C_2N	Количество
А	С	27	5
А	С	27	6
Б	Д	26	7
Б	Д	27	7

Вариант 2.

Наименование	Свойство	Артикул	Количество
А	С	27	9
А	Д	27	9
Б	С	27	9
Б	Д	25	9

Вариант 3.

Тип	Характеристика	Группа	Количество
А	С	13	100
А	Д	13	100
Б	С	13	200
Б	Д	13	100

Вариант 4.

НаименованиеЭмитента	ТипЦБ	ДатаЭмиссии
ОАО ―КрАЗ‖	акция привилегированная	20.06.1999
ОАО ―КрАЗ‖	акция обыкновенная	20.06.1999
ЗАО ―Агат‖	акция привилегированная	23.06.1999

ТОО ―Искра‖	акция привилегированная	20.06.1999

Вариант 5.

НаименованиеВуза	Команда	Приз
КрасГУ	МатФак	Торт
КГТУ	ЭконФак	Арбуз
СибГТУ	МатФак	Торт
НГУ	ЭконФак	Бананы

Вариант 6.

НаименованиеВуза	Адрес	Команда
КрасГУ	Красноярск, пр-т Свободный,79	МатФак
КрасГУ	Красноярск, пр-т Свободный,79	ЭконФак
КГТУ	Красноярск, ул.Киренского,26	ЭконФак
КГТУ эк. институт	Красноярск, ул.Киренского,26	ФизФак
НГУ	Новосибирск, ул. Пирогова, 2	МатФак
НГУ	Новосибирск, ул. Пирогова, 2	ФизФак

Вариант 7.

НаименованиеВуза	Команда	Приз
КрасГУ	МатФак	Торт
КрасГУ	ЭконФак	Торт
НГУ	ФилФак	Арбуз
НГУ	МатФак	Бананы

Вариант 8.

НаименованиеЭмитента	ТипЦБ	ДатаЭмиссии
ОАО ―КрАЗ‖	акция привилегированная	24.06.1999
ЗАО ―Сибириада‖	акция обыкновенная	25.06.1999
ТОО ―Искра‖	акция привилегированная	23.06.1999
ЗАО ―Агат‖	акция обыкновенная	25.06.1999

Вариант 9.

Наименование Вуза	Адрес	Команда
КрасГУ	Красноярск, пр-т Свободный,79	МатФак
КрасГУ	Красноярск, пр-т Свободный,79	ЭконФак
КГТУ	Красноярск, ул.Киренского,26	ФизФак
КГТУ эк. институт	Красноярск, ул.Киренского,26	ЮрФак
НГУ	Новосибирск, ул. Пирогова, 2	ФилФак

НГУ	Новосибирск, ул. Пирогова, 2	БиоХим

Вариант 10.

НаименованиеЭмитента	ТипЦБ	ДатаЭмиссии
ОАО ―КрАЗ‖	акция привилегированная	20.06.1999
ОАО ―КрАЗ‖	акция обыкновенная	23.06.1999
ЗАО ―Агат‖	акция привилегированная	20.06.1999
ЗАО ―Агат‖	акция привилегированная	24.06.1999

Вариант 11.

P1	P2	Дата
A	D	2000
A	E	2000
B	D	2005
C	D	2000

Вариант 12.

A	B	C
A1	B1	C1
A2	B2	C2
A3	B1	C1
A4	B2	C3

Вариант 13.

A	B	C
A1	B1	C1
A1	B1	C2
A2	B2	C2
A3	B2	C3
A4	B3	C1
A4	B3	C3

Вариант 14.

A	B	C
A1	B1	C1
A1	B2	C1
A2	B3	C2
A2	B1	C3

Вариант 15.

A	B	C
A1	B1	C1
A2	B2	C2
A3	B1	C3
A4	B2	C2

Вариант 16.

A	B	C
A1	B1	C1
A1	B1	C2
A2	B2	C3
A3	B2	C4
A4	B3	C5
A4	B3	C6

Вариант 17.

A	B	C
A1	B1	C1
A1	B2	C2
A2	B1	C1
A2	B1	C3

Вариант 18.

P	F	Q
41	21	11
42	22	12
43	21	11
44	22	13

Вариант 19.

C	D	E
71	61	51
71	61	52
72	62	52
73	62	53
74	63	51
74	63	53

Вариант 20.

F	G	H
App	Beep8	Call7

App	Beep2	Call7
Opp	Beep3	Call2
Opp	Beep8	Call3

Вариант 21

НаименованиеВуза	Адрес	Команда
КрасГУ	Красноярск, пр-т Свободный, 79	Первая
КрасГУ	Красноярск, пр-т Свободный, 79	Вторая
КГТУ	Красноярск, ул. Киренского, 26	Первая
КГТУ	Красноярск, ул. Киренского, 26	Вторая
НГУ	Новосибирск, ул. Пирогова, 2	Первая
НГУ	Новосибирск, ул. Пирогова, 2	Вторая

Вариант 22.

H_2O	Виртуальность	C_2N	Масса
Yes	P	27	45
Yes	P	27	91
No	D	26	NULL
No	D	27	NULL

Вариант 23.

Количество	Свойство	Артикул	Наименование
752	NP	2007	HDR
752	PN	2007	HDR
345	NP	2007	HDR
345	PN	2005	HDR

Вариант 24.

Тип	Признак	Месяц	Век
Второй	Да	10	19
Второй	Нет	10	19
Первый	Да	10	20
Первый	Нет	10	19

Вариант 25.

НаименованиеЭмитента	ТипЦБ	ДатаЭмиссии
ОАО ―Камаз‖	акция привилегированная	20.06.2010

ОАО ―КрАЗ‖	акция обыкновенная	23.06.2010
ЗАО ―Агат‖	акция привилегированная	20.06.2010
ТОО ―Искра‖	акция привилегированная	20.06.2010

Вариант 26.

НаименованиеВуза	Команда	Приз
ТГУ	ФилФак	Сист. блок
ТПУ	ЭконФак	Ноутбук
ТГПУ	МатФак	Монитор
ТУСУР	ЭконФак	Ноутбук

Вариант 27.

НаименованиеВуза	Адрес	Команда
ТГУ	Томск, пр. Ленина, 36	МатФак
ТГУ	Томск, пр. Ленина, 36	ГРФак
ТПУ	Томск, пр. Ленина, 2	ГРФак
ТПУ ФТИ	Томск, пр. Ленина, 2	ФизФак
ТУСУР	Томск, пр. Ленина, 40	ФизФак
ТУСУР	Томск, пр. Ленина, 40	МатФак

Вариант 28.

НаименованиеВуза	Команда	Приз
ТГУ	ЭконФак	Торт
ТГУ	МатФак	Торт
ТПУ	МатФак	Бананы
ТПУ	ФилФак	Арбуз

Вариант 29.

НаименованиеЭмитента	ТипЦБ	ДатаЭмиссии
ЗАО ―Агат‖	акция привилегированная	13.05.2009
ЗАО ―Сибириада‖	акция обыкновенная	14.05.2009
ТОО ―Искра‖	акция привилегированная	12.05.2009
ОАО ―КрАЗ‖	акция обыкновенная	14.05.2009

Вариант 30.

Организация	Адрес	Статус
ТРЕНД	Москва, Кирова, 79	РОА
ТРЕНД	Москва, Кирова, 79	ОУП
ЮРАС, первый	Томск, Кирова, 26	ООО
ЮРАС	Томск, Кирова, 26	ОАО

БЕППО	Юрга, Кирова, 2	БПФ
БЕППО	Юрга, Кирова, 2	ПРС

Вариант 31.

Город	Событие	Дата
Калуга	пожар	11.09.1812
Калуга	наводнение	14.09.1813
Вятка	пожар	11.09.1812
Вятка	пожар	15.09.1825

Вариант 32.

B1	B2	B3
post	cool	stop
post	hot	stop
put	cool	stop
get	cool	nonstop

Вариант 33.

A	B	C
14	362	154
509	412	678
648	362	154
3	412	4

Вариант 34.

Риск	Фреш	Серия
Р1	Ф1	С2
Р1	Ф1	С1
Р2	Ф2	С2
Р3	Ф2	С3
Р4	Ф3	С3
Р4	Ф3	С1

Вариант 35.

Тип	Вес	Цена
HDR	512	13500
HDR	432	13500
GTS	256	14000
GTS	512	16000

Вариант 36.

A	B	C
A11	B7	C2
A21	B7	C1

A17	B9	C4
A3	B9	C4

Вариант 37.

A	B	C
111	31	82
101	31	3
45	77	2
45	77	7
92	91	54
92	91	115

Вариант 38.

Red	Blue	Green
F101	567	706
F101	A812	535
C255	567	706
C255	567	536

Вариант 39.

D	F	T
284	17	78
968	17	89
274	33	56
904	33	56

Вариант 40.

AA	YY	ZZ
72	62	52
71	61	52
73	62	53
74	63	53
74	63	51
71	61	51

Вариант 41.

Наличие	Тип	Число	Масса
Yes	FULL	45	90
No	DEPT	54	130
No	DEPT	45	130
Yes	FULL	45	60

Вариант 42.

Вероятность*1000	Время, с	Мощность, вт	Тип
694	15	2015	HDR
694	17	2015	HDR
604	15	3500	HPQ
604	17	2015	HDR

Вариант 43.

Тип	Наличие	Артикул
Второй	Да	17
Второй	Нет	17
Первый	Да	17
Первый	Нет	17
Второй	Неизвестно	18
Первый	Неизвестно	18

Вариант 44.

A	B	C
17	49	17
35	53	35
24	49	24
19	53	35

Практическая работа №3 «Проектирование реляционной БД. Нормализация таблиц»

Цель работы: выработать практические навыки моделирования предметной области и построения нормальных форм баз данных

Нормализация, функциональные и многозначные зависимости

Нормализация – это разбиение таблицы на две или более, обладающих лучшими свойствами при включении, изменении и удалении данных. Окончательная цель нормализации сводится к получению такого проекта базы данных, в котором каждый факт появляется лишь в одном месте, т.е. исключена избыточность информации. Это делается не столько с целью экономии памяти, сколько для исключения возможной противоречивости хранимых данных.

Каждая таблица в реляционной БД удовлетворяет условию, в соответствии с которым в позиции на пересечении каждой строки и столбца таблицы всегда находится единственное атомарное значение, и никогда не может быть множества таких значений. Любая таблица, удовлетворяющая этому условию, называется нормализованной. Фактически, ненормализованные таблицы, т.е. таблицы, содержащие повторяющиеся группы, даже не допускаются в реляционной
БД.

Всякая нормализованная таблица автоматически считается таблицей в первой нормальной форме, сокращенно 1НФ. Таким образом, строго говоря, "нормализованная" и "находящаяся в 1НФ" означают одно и то же. Однако на практике термин "нормализованная" часто используется в более узком смысле – "полностью нормализованная", который означает, что в проекте не нарушаются никакие принципы нормализации.

Теперь в дополнение к 1НФ можно определить дальнейшие уровни нормализации – вторую нормальную форму (2НФ), третью нормальную форму (3НФ) и т.д.

По существу, таблица находится в 2НФ, если она находится в 1НФ и удовлетворяет, кроме того, некоторому дополнительному условию, суть которого будет рассмотрена ниже. Таблица находится в 3НФ, если она находится в 2НФ и, помимо этого, удовлетворяет еще другому дополнительному условию и т.д.

Таким образом, каждая нормальная форма является в некотором смысле более ограниченной, но и более желательной, чем предшествующая. Это связано с тем, что "(N+1)-я нормальная форма" не обладает некоторыми непривлекательными особенностями, свойственным "N-й нормальной форме". Общий смысл дополнительного условия, налагаемого на (N+1)-ю нормальную форму по отношению к N-й нормальной форме, состоит в исключении этих непривлекательных особенностей.

За время развития технологии проектирования реляционных БД были выделены следующие нормальные формы: первая нормальная форма (1NF); вторая нормальная форма (2NF); третья нормальная форма (3NF); нормальная форма Бойса-Кодда (BCNF); четвертая нормальная форма (4NF); пятая нормальная форма, или нормальная форма проекции-соединения (5NF).

Обычно на практике применение находят только первые три нормальные формы.

Задание для практической работы

Задание 1. Приведение отношения к первой нормальной форме

1. Открыть файл journal.xls. Лист Ученик. Здесь собрана информация об ученике (см. лист Ученик). Анализ таблицы показывает, что в столбце «Родители» и «Место работы» указаны по 2 значения. Кроме того, в столбце адрес слишком много данных. Возможна ситуация, когда РОНО будет интересовать район проживания, при этом конкретная квартира – неинтересна. Из-за неоднозначности значений придется привести таблицу к первой нормальной форме.
2. В таблице скопировать строки и сделать так, чтобы в каждой строке был один родитель. 3. Одновременно нужно создать еще один столбец «Контекст адреса»
4. Получим отношение в первой нормальной форме.
5. Выделим в ней первичный ключ: строки отличаются друг от друга столбцами - «№ билета» и «Фамилии родителей». Зальем желтым цветом.

Задание 2. Приведение отношения ко второй нормальной форме

1. Открыть файл journal.xls. Лист Ученик.
2. Сделайте еще одну таблицу. Будет 2 таблицы. В одной будет все, что определяется только столбцом **«**№ билета**»,** а в другой все, что определяется 2-мя столбцами.
3. Первую таблицу назовем **«**Личные данные ученика**»**. Первичный ключ **«**№ билета**»**
4. Вторую таблицу назовем «Родители». Первичный ключ «№ билета» и «Фамилии родителей».
5. Отношение «Родители» конечно, больше преобразованиям не подлежит. **Задание 3. Приведение отношения к третьей нормальной форме**

1. Открыть файл journal.xls. Лист Ученик.
2. Выделим из отношения «Личные данные ученика» таблицу «Класс».
3. Останется таблица «Личные данные». Однако в ней нужно оставить столбец «Класс».
4. В таблице «Класс» ключом является столбец «Класс».

ЗАМЕЧАНИЕ. Название специализации встречается многократно для разных классов. Со временем формулировка может измениться. Поэтому

целесообразно сделать справочную таблицу «Справка» и сделать столбец «Код специализации».

Задание 4. Нормализация отношения «Преподаватель».

1. Открыть файл journal.xls. Лист «Преподаватель».
2. Отношение «Преподаватель» привести к первой нормальной форме. Для этого скопировать строки, исправить данные так, чтобы в каждой строке был только один класс и предмет.
3. Выделим первичный ключ. Это столбцы «Фамилия преподавателя», «Класс» и «Предмет».
4. Анализируем и видим, что есть атрибуты, которые зависят только от фамилии преподавателя. Значит, нужно привести отношение ко второй нормальной форме.
5. Получим две таблицы. Первую назовем «Преподаватель – класс – предмет». Она уже находится в третьей нормальной форме, т.к. здесь атрибуты ключевые и все не зависят друг от друга. Вернее здесь наблюдаются связи многие-ко-многим. Отставим пока.
6. Рассмотрим вторую таблицу «Личные данные преподавателя». Первичный ключ «Фамилия». Поскольку данные преподавателя могут меняться (замуж вышла), то целесообразнее сделать столбец «Код преподавателя» и сделать этот столбец ключевым. Но тогда в отношении «Класс» нужно вместо поля «Классный руководитель» поставить «Код классного руководителя».
7. Посмотрим на столбец «Классное руководство».
8. Такая информация у нас уже есть, дублировать ее не надо. Поэтому просто вычеркнем этот столбец.

Задание 5. Приведение отношения к четвертой нормальной форме

1. Открыть файл journal.xls. Лист «Преподаватель». Отношение «Преподаватель – класс – предмет».
2. Разделим его на 2 таблицы «Преподаватель – предмет» и «Преподаватель –класс».
3. Видим, что таблица «Преподаватель – класс» осталась «многие – ко – многим». Оставим ее в покое.

Задание 6. Конечный результат

1. Откройте файл journal_ready.xls.
2. Сравните эти таблицы с теми, которые получились у вас.

Задание 7. Постройте сетевую модель по примеру в программе ERModeler

На листе «Конечный результат» приведена реляционная модель той части задачи школьного журнала, нормализацией которой мы занимались на предыдущих страницах. Приведенную модель сложно считать наглядной и удобной для восприятия, однако именно такой вид представления наиболее удобен для проведения дальнейших этапов проектирования. Понимая это,

условимся в дальнейшем проводить этап инфологического проектирования с учетом требований к реляционным моделям.

Практическая работа №4 «Задание ключей. Создание основных объектов БД»

Цель работы: приобретение практических навыков анализа предметной области, информационных задач и построения концептуальной модели базы данных.

Проектирование базы данных (БД)

Проектирование базы данных (БД) – одна из наиболее сложных и ответственных задач, связанных с созданием информационной системы (ИС). В результате еѐ решения должны быть определены содержание БД, эффективный для всех еѐ будущих пользователей способ организации данных и инструментальные средства управления данными.

Основная цель процесса проектирования БД состоит в получении такого проекта, который удовлетворяет следующим требованиям:

- корректность схемы БД, т.е. база должна быть гомоморфным образом моделируемой предметной области (ПО), где каждому объекту предметной области соответствуют данные в памяти ЭВМ, а каждому процессу – адекватные процедуры обработки данных;
- обеспечение ограничений ;
- эффективность функционирования ;
- защита данных (от аппаратных и программных сбоев и несанкционированного доступа);
- простота и удобство эксплуатации;
- гибкость, т.е. возможность развития и адаптации к изменениям предметной области и/или требований пользователей.

Внимание! Базы данных всегда проектируются под конкретное назначение системы.

Техника проектирования баз данных может измениться в целом и в деталях в зависимости от назначения системы. Например, следует различать проектирование систем складирования данных и проектирование так называемых OLTP-систем, ориентируемых на оперативную обработку транзакций. В данном учебном курсе рассматривается проектирование баз данных в основном для OLTP-систем. Именно на таких системах исторически сложилась техника проектирования баз данных.

Этапы проектирования базы данных

Процесс проектирования включает в себя следующие этапы:

Концептуальное проектирование – это процедура конструирования информационной модели, не зависящей от каких-либо физических условий реализации.

Логическое проектирование – это процесс конструирования информационной модели на основе существующих моделей данных, не зависимо от используемой СУБД и других условий физической реализации.

Физическое проектирование – это процедура создания описания конкретной реализации БД с описанием структуры хранения данных, методов доступа к данным.

Концептуальное проектирование

Основными задачами концептуального проектирования являются определение предметной области системы и формирование взгляда на ПО с позиций сообщества будущих пользователей БД, т.е. инфологической модели ПО.

Концептуальная модель ПО представляет собой описание структуры и динамики ПО, характера информационных потребностей пользователей в терминах, понятных пользователю и не зависимых от реализации БД. Это описание выражается в терминах не отдельных объектов ПО и связей между ними, а их типов, связанных с ними ограничений целостности и тех процессов, которые приводят к переходу предметной области из одного состояния в другое.

Рассмотрим основные подходы к созданию концептуальной модели предметной области.

1. Функциональный подход к проектированию БД

   Этот метод реализует принцип "от задач" и применяется тогда, когда известны функции некоторой группы лиц и/или комплекса задач, для обслуживания информационных потребностей которых создаѐтся рассматриваемая БД.

2. Предметный подход к проектированию БД

   Предметный подход к проектированию БД применяется в тех случаях, когда у разработчиков есть чѐткое представление о самой ПО и о том, какую именно информацию они хотели бы хранить в БД, а структура запросов не определена или определена не полностью. Тогда основное внимание уделяется исследованию ПО и наиболее адекватному еѐ отображению в БД с учѐтом самого широкого спектра информационных запросов к ней.

3. Проектирование с использованием метода "сущность-связь"

   Метод "сущность–связь" (entity–relation, ER–method) является комбинацией двух предыдущих и обладает достоинствами обоих. Этап инфологического проектирования начинается с моделирования ПО. Проектировщик разбивает еѐ на ряд локальных областей, каждая из которых (в идеале) включает в себя информацию, достаточную для обеспечения запросов отдельной группы будущих пользователей или решения отдельной задачи (подзадачи). Каждое локальное представление моделируется отдельно, затем они объединяются.

Выбор локального представления зависит от масштабов ПО. Обычно она разбивается на локальные области таким образом, чтобы каждая из них соответствовала отдельному внешнему приложению и содержала 6-7 сущностей.

Сущность – это объект, о котором в системе будет накапливаться информация. Сущности бывают как физически существующие (например, СОТРУДНИК или АВТОМОБИЛЬ), так и абстрактные (например, ЭКЗАМЕН или ДИАГНОЗ). Для сущностей различают тип сущности и экземпляр. Тип характеризуется именем и списком свойств, а экземпляр – конкретными значениями свойств. Типы сущностей можно классифицировать как сильные и слабые. Сильные сущности существуют сами по себе, а существование слабых сущностей зависит от существования сильных. Например, читатель библиотеки – сильная сущность, а абонемент этого читателя – слабая, которая зависит от наличия соответствующего читателя. Слабые сущности называют подчинѐнными (дочерними), а сильные – базовыми (основными, родительскими).

Для каждой сущности выбираются свойства (атрибуты). Различают:

- Идентифицирующие и описательные атрибуты. Идентифицирующие атрибуты имеют уникальное значение для сущностей данного типа и являются потенциальными ключами. Они позволяют однозначно распознавать экземпляры сущности. Из потенциальных ключей выбирается один первичный ключ (ПК). В качестве ПК обычно выбирается потенциальный ключ, по которому чаще происходит обращение к экземплярам записи. Кроме того, ПК должен включать в свой состав минимально необходимое для идентификации количество атрибутов. Остальные атрибуты называются описательными и заключают в себе интересующие свойства сущности.
- Составные и простые атрибуты. Простой атрибут состоит из одного компонента, его значение неделимо. Составной атрибут является комбинацией нескольких компонентов, возможно, принадлежащих разным типам данных (например, ФИО или адрес). Решение о том, использовать составной атрибут или разбивать его на компоненты, зависит от характера его обработки и формата пользовательского представления этого атрибута.
- Однозначные и многозначные атрибуты (могут иметь соответственно одно или много значений для каждого экземпляра сущности).
- Основные и производные атрибуты. Значение основного атрибута не зависит от других атрибутов. Значение производного атрибута вычисляется на основе значений других атрибутов (например,

возраст студента вычисляется на основе даты его рождения и текущей даты).

Спецификация атрибута состоит из его названия, указания типа данных и описания ограничений целостности – множества значений (или домена), которые может принимать данный атрибут. Далее осуществляется спецификация связей внутри локального представления. Связи могут иметь различный содержательный смысл (семантику). Различают связи типа "сущностьсущность", "сущность-атрибут" и "атрибут-атрибут" для отношений между атрибутами, которые характеризуют одну и ту же сущность или одну и ту же связь типа "сущность-сущность". Каждая связь характеризуется именем, обязательностью, типом и степенью. Различают факультативные и обязательные связи. Если вновь порождѐнный объект одного типа оказывается по необходимости связанным с объектом другого типа, то между этими типами объектов существует обязательная связь (обозначается двойной линией). Иначе связь является факультативной. По типу различают множественные связи "один к одному" (1:1), "один ко многим" (1:N) и "многие ко многим" (M:N). Степень связи определяется количеством сущностей, которые охвачены данной связью. Пример бинарной связи – связь между отделом и сотрудниками, которые в нѐм работают. Примером тернарной связи является связь типа экзамен между сущностями ДИСЦИПЛИНА, СТУДЕНТ, ПРЕПОДАВАТЕЛЬ. Из последнего примера видно, что связь также может иметь атрибуты (в данном случае это Дата проведения и Оценка). Пример ER–диаграммы с указанием сущностей, их атрибутов и связей приведен на рис. 1.

Рисунок 1. Пример ER–диаграммы с однозначными и многозначными атрибутами

Пример проектирования реляционной базы данных

В качестве примера возьмем базу данных компании, которая занимается издательской деятельностью. База данных создаѐтся для информационного обслуживания редакторов, менеджеров и других сотрудников компании. БД должна содержать данные о сотрудниках компании, книгах, авторах, финансовом состоянии компании и предоставлять возможность получать

разнообразные отчѐты. В соответствии с предметной областью система строится с учѐтом следующих особенностей:

- каждая книга издаѐтся в рамках контракта;
- книга может быть написана несколькими авторами;
- контракт подписывается одним менеджером и всеми авторами книги;
- каждый автор может написать несколько книг (по разным контрактам);
- порядок, в котором авторы указаны на обложке, влияет на размер гонорара;
- если сотрудник является редактором, то он может работать одновременно над несколькими книгами;
- у каждой книги может быть несколько редакторов, один из них – ответственный редактор;
- каждый заказ оформляется на одного заказчика;
- в заказе на покупку может быть перечислено несколько книг.

Выделим базовые сущности этой предметной области:

1. Сотрудники компании. Атрибуты сотрудников – ФИО, табельный номер, пол, дата рождения, паспортные данные, ИНН, должность, оклад, домашний адрес и телефоны. Для редакторов необходимо хранить сведения о редактируемых книгах; для менеджеров – сведения о подписанных контрактах.
2. Авторы. Атрибуты авторов – ФИО, ИНН (индивидуальный номер налогоплательщика), паспортные данные, домашний адрес, телефоны. Для авторов необходимо хранить сведения о написанных книгах.
3. Книги. Атрибуты книги – авторы, название, тираж, дата выхода, цена одного экземпляра, общие затраты на издание, авторский гонорар.
4. Контракты будем рассматривать как связь между авторами, книгами и менеджерами. Атрибуты контракта – номер, дата подписания и участники.
5. Для отражения финансового положения компании в системе нужно учитывать заказы на книги. Для заказа необходимо хранить номер заказа, заказчика, адрес заказчика, дату поступления заказа, дату его выполнения, список заказанных книг с указанием количества экземпляров.

ER–диаграмма издательской компании приведена на рис. 2 (базовые сущности на рисунках выделены полужирным шрифтом).

Анализ информационных задач и круга пользователей системы

Система создаѐтся для обслуживания следующих групп пользователей:

- администрация (дирекция);

- менеджеры;
- редакторы;
- сотрудники компании, обслуживающие заказы.

Определим границы информационной поддержки пользователей: 1) Функциональные возможности:

- ведение БД (запись, чтение, модификация, удаление в архив);
- обеспечение логической непротиворечивости БД;
- обеспечение защиты данных от несанкционированного или случайного доступа (определение прав доступа);
- реализация наиболее часто встречающихся запросов в готовом виде;
- предоставление возможности сформировать произвольный запрос на языке манипулирования данными.

2) Готовые запросы:

- получение списка всех текущих проектов (книг, находящихся в печати и в продаже);
- получение списка редакторов, работающих над книгами;
- получение полной информации о книге (проекте);
- получение сведений о конкретном авторе (с перечнем всех книг);
- получение информации о продажах (по одному или по всем проектам);
- определение общей прибыли от продаж по текущим проектам;
- определение размера гонорара автора по конкретному проекту.

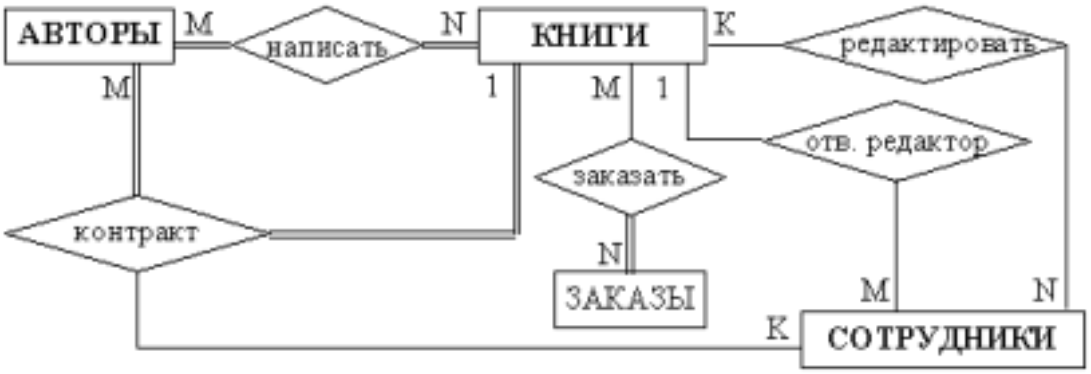

Рисунок 2. ER–диаграмма издательской компании

Задание для практической работы

По заданному описанию предметной области построить концептуальную модель базы данных

Выделите типы сущностей;

- Выделите типы связей и определите для них показатели кардинальности и степень участия сторон;
- Выделите атрибуты и свяжите их типами сущностей и связей;
- Определите потенциальные и первичные ключи сущностей;

- Нарисуйте ER-диаграмму.
- и проанализируйте информационные задачи и группы пользователей.

Индивидуальные задания к практической работе

Вариант 1.

Задача – организация учебного процесса в вузе:

- Студенты: паспортные данные, адрес, дата зачисления, номер приказа, факультет, группа, является ли старостой, кафедра (специализация), изучаемые (изученные) предметы, оценки, задолженности, стипендия.
- Учебные курсы: название, факультет(ы), групп(ы), кафедра, семестр(ы), форма отчѐтности, число часов.
- Преподаватели: паспортные данные, адрес, телефон, фотография, кафедра, должность, учѐная степень, начальник (зав. кафедрой), предмет(ы), число ставок, зарплата.

Вариант 2.

Учет и выдача книг в библиотеке вуза:

- Книги: авторы, название, раздел УДК, раздел (техническая, общественно-политическая и т.п.), место и год издания, издательство, количество страниц, иллюстрированность, цена, дата покупки, номер сопроводительного документа (чек, счѐт/накладная), вид издания (книги, учебники, брошюры, периодические издания), инвентарный номер (есть только для книг и некоторых учебников), длительность использования читателями (год, две недели, день), электронная версия книги или ее реферата (отсканированный текст).
- Читатели: номер читательского билета, ФИО, год рождения, адрес, дата записи, вид (студент, аспирант, преподаватель, сотрудник), курс, номер группы, названия взятых книг и даты их выдачи.

Вариант3.

Отдел кадров некоторой компании.

- Сотрудники: ФИО, паспортные данные, фотография, дом. и моб. телефоны, отдел, комната, раб. телефоны (в т.ч. местный), подчинѐнные сотрудники, должность, тип(ы) работы, задание(я), проект(ы), размер зарплаты, форма зарплаты (почасовая, фиксированная).
- Отделы: название, комната, телефон(ы), начальник, размер финансирования, число сотрудников.
- Проекты: название, дата начала, дата окончания, размер финансирования, тип финансирования (периодический, разовый), задачи и их исполнители, структура затрат и статьи расходов.

Вариант 4.

Отдел поставок некоторого предприятия.

- Поставщики: название компании, ФИО контактного лица, расчѐтный счѐт в банке, телефон, факс, поставляемое оборудование (материалы), даты поставок (по договорам и реальные), метод и стоимость доставки.
- Сырьѐ: тип, марка, минимальный запас на складе, время задержки, цена, продукты, при производстве которых используется, потребляемые объемы (необходимый, реальный, на единицу продукции).

Вариант5.

Пункт проката видеозаписей (внутренний учет).

- Видеокассеты: идентификационный номер видеокассеты, тип видеокассет, дата его создания, компания-поставщик, число штук данного типа (общее, в магазине, выдано в настоящее время, выдано всего, выдано в среднем за месяц), общая длительность записей; записи видеокассет: название, длительность, категория, год выпуска и производитель (оригинала).
- Клиенты: ФИО, паспортные данные, адрес, телефон; заказы, т.е. взятые видеокассеты (сейчас и в прошлом): номер, дата выдачи, дата возвращения, общая стоимость заказа.

Вариант 6.

Пункт проката видеозаписей (информация для клиентов).

- Видеокассеты: краткое описание, внешний вид (этикетка), марка (пустой) видеокассеты, цена за единицу прокатного времени (например: 1 день, 3 дня, неделя), есть ли в наличии, общая длительность записей; записи на видеокассете: название, длительность, жанр (категория), тема, год и страна выпуска (оригинала), кинокомпания, описание, актеры, режиссер.
- Заказы: идентификационные номера и названия выданных видеокассет, дата выдачи, дата возвращения (продления), общая стоимость заказа, возвращены ли кассеты заказа. Вариант 7.

Кинотеатры (информация для зрителей).

- Фильмы: название, описание, жанр (категория), длительность, популярность (рейтинг, число проданных билетов в России и в мире), показывается ли сейчас (сегодня, на текущей неделе), в каких кинотеатрах показывается, цены на билеты (в т.ч. средние).
- Кинотеатры: название, адрес, схема проезда, описание, число мест (в разных залах, если их несколько), акустическая система, широкоэкранность, фильмы и цены на них: детские и взрослые билеты в зависимости от сеанса (дневной, вечерний и т.п.) и от категории мест (передние, задние и т.п.); сеансы показа фильмов (дата и время начала).

Вариант 8.

Ресторан (информация для посетителей).

- Меню: дневное или вечернее, список блюд по категориям.
- Блюда: цена, название, вид кухни, категории (первое, второе и т.п.; мясное, рыбное, салат и т.п.), является ли вегетарианским, компоненты блюда, время приготовления, есть ли в наличии.
- Компоненты блюд: тип (гарнир, соус, мясо и т.п.), калорийность, цена, рецепт, время приготовления, есть ли в наличии, ингредиенты (продукты) и их расходы на порцию.

Вариант 9.

Задача - информационная поддержка деятельности склада.

База данных должна содержать информацию о наименовании товара, его поставщике, количестве, цене товара, конечном сроке реализации, сроке хранения на складе. Торговый склад производит уценку хранящейся продукции. Если продукция хранится на складе дольше 10 месяцев, то она уценивается в 2 раза, а если срок хранения превысил 6 месяцев, но не достиг 10, то в 1,5 раза. Ведомость уценки товаров должна содержать информацию: наименование товара, количество товара(шт.), цена товара до уценки, срок хранения товара, цена товара после уценки, общая стоимость товаров после уценки.

Вариант10.

Задача – информационная поддержка деятельности адвокатской конторы. БД должна осуществлять:

- ведение списка адвокатов; □ ведение списка клиентов; □ ведение архива законченных дел.
- Необходимо предусмотреть:
- получение списка текущих клиентов для конкретного адвоката;
- определение эффективности защиты (максимальный срок минус полученный срок) с учѐтом оправданий, условных сроков и штрафов;
- определение неэффективности защиты (полученный срок минус минимальный срок);
- подсчѐт суммы гонораров (по отдельных делам) в текущем году;
- получение для конкретного адвоката списка текущих клиентов, которых он защищал ранее (из архива, с указанием полученных сроков и статей).

Вариант11.

Задача – информационная поддержка деятельности гостиницы.

БД должна осуществлять:

- ведение списка постояльцев;
- учѐт забронированных мест;
- ведение архива выбывших постояльцев за последний год.

- Необходимо предусмотреть:
- получение списка свободных номеров (по количеству мест и классу);
- получение списка номеров (мест), освобождающихся сегодня и завтра;
- выдачу информации по конкретному номеру;
- автоматизацию выдачи счетов на оплату номера и услуг;
- получение списка забронированных номеров;
- проверку наличия брони по имени клиента и/или названию организации Вариант12.

Описание предметной области:

- В компании несколько отделов.
- В каждом отделе есть некоторое количество сотрудников, занятых в нескольких проектах и размещающихся в нескольких офисах.
- Каждый сотрудник имеет план работы, т.е. несколько заданий, которые он должен выполнить. Для каждого такого задания существует ведомость, содержащая перечень денежных сумм, полученных сотрудником за выполнение этого задания.
- В каждом офисе установлено несколько телефонов.
- В базе данных должна храниться следующая информация.
- Для каждого отдела: номер отдела (уникальный), его бюджет и личный номер сотрудника, возглавляющего отдел (уникальный).
- Для каждого сотрудника: личный номер сотрудника (уникальный), номер текущего проекта, номер офиса, номер телефона, название выполняемого задания вместе с датой и размером выплат, проведенных в качестве оплаты за выполнение данного задания.
- Для каждого проекта: номер проекта (уникальный) и его бюджет.
- Для каждого офиса: номер офиса (уникальный), площадь в квадратных футах, номера всех установленных в нем телефонов.

Вариант13.

Задача – информационная поддержка деятельности спортивного клуба. БД должна осуществлять:

- ведение списков спортсменов и тренеров;
- учѐт проводимых соревнований (с ведением их архива); □ учѐт травм, полученных спортсменами.

Необходимо предусмотреть:

- возможность перехода спортсмена от одного тренера к другому;
- составление рейтингов спортсменов;
- составление рейтингов тренеров;
- выдачу информации по соревнованиям;
- выдачу информации по конкретному спортсмену;

- подбор возможных кандидатур на участие в соревнованиях (соответствующего уровня мастерства, возраста и без травм).

Вариант14.

Задача – информационная поддержка деятельности аптечного склада.

В аптечном складе хранятся лекарства. Сведения о лекарствах содержатся в специальной ведомости: наименование лекарственного препарата; количество (в шт.); цена; срок хранения на складе (в месяцах). Лекарства поступают на склад ежедневно от разных поставщиков, отпускаются два раза в неделю по предварительным заказам аптек. Выяснить, сколько стоит самый дорогой и самый дешевый препарат; сколько препаратов хранится на складе более 3 месяцев; сколько стоят все препараты, хранящиеся на складе, отыскать препараты, остаток которых равен нулю , ниже требуемого по заказам.

Вариант15.

—Электронный журнал посещаемости"

Предметная область представлена следующими документами:

- Список студентов
- Журнал посещаемости
- Расписание занятий
- Предусмотреть учет пропусков по уважительным, неуважительным причинам. Подсчет □ пропусков по каждому студенту, знеделю, месяц, заданный период, по конкретному
- предмету.

Вариант16.

«Итоги сессии»

База данных должна содержать информацию о двух последних сессиях студентов. Источником информации являются экзаменационные ведомости. Необходимо проводить анализ успеваемости по специальностям, формам обучения, курсам, группам, предметам, вычислять средний балл по указанным критериям, а также число каждых оценок .

Практическая работа №5 «Создание проекта БД. Создание БД. Редактирование и модификация таблиц»

Цель работы: освоить основные приемы заполнения и редактирования таблиц; познакомиться с простой сортировкой данных и с поиском записей по образцу; научиться сохранять и загружать базы данных.

Microsoft Office Access

Access является наиболее сложной программой из всех офисных приложений Microsoft Office. Чтобы начать работу с этой программой, вначале необходимо создать структуру базы данных, подробно ее описать, а затем создать различные формы.

ACCESS – это реляционная СУБД. Это означает, что с ее помощью можно работать одновременно с несколькими таблицами базы данных, эти таблицы между собой связаны. Таблицу ACCESS можно связать с данными, хранящимися на другом компьютере. Данные ACCESS очень просто комбинировать с данными EXCEL, WORD и другими программами Office.

Access во многом похож на Excel. Основное различие между таблицей БД и электронной таблицей – в системе адресации: в электронной таблице адресуется каждая ячейка, а в таблице БД – только поля текущей записи.

Таблицы – основные объекты базы данных (БД). В них хранятся данные. Реляционная база данных может иметь много взаимосвязанных таблиц. Сведения по разным вопросам следует хранить в разных таблицах.

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

Формы – Обеспечивают более наглядную работу с таблицами, с помощью форм в базу вводят новые данные или просматривают имеющиеся.

Отчеты – средство представления данных таблиц. Отчеты могут быть оформлены надлежащим образом и распечатаны в том виде, в котором требуется пользователю.

Макросы – набор из одной или более макрокоманд, выполняющих определенные операции

(открытие форм, печать отчетов)

Модули - это программы, написанные на языке программирования Visual Basic.

Задание для практической работы

Заполните таблицы по образцу

1. Вызвать программу Access 2007 (Access 2016).
2. В окне системы управления базы данных щелкнуть по значку «Новая база данных». Справа в появившемся окне дать имя новой базе данных «Анкета ГС-31» и щелкнуть по значку папки, находящемуся справа от окна названия  . Откроется окно сохранения, найдите свою папку и сохраните в нее новый файл базы данных «Анкета ГС-31» (вместо ГС-31 укажите номер вашей группы). Затем нажмите на кнопку «Создать».
3. Появится окно «Таблица» (Рисунок 5.1).

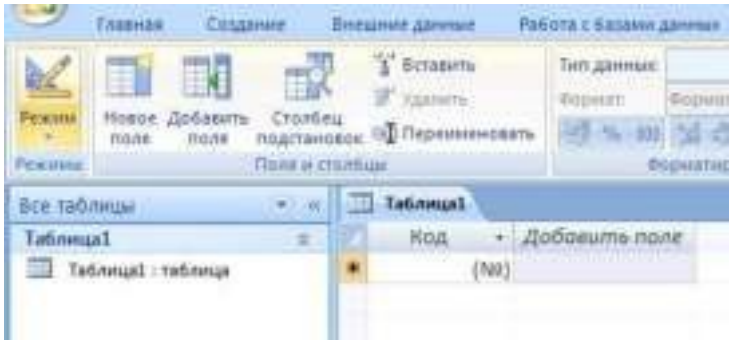

Рисунок 5.1. Окно пустой базы данных

4. В появившемся окне откройте меню команды <Режим> и выберите вариант
   <Конструктор> и сохраните будущую таблицу под названием <Ведомость успеваемости>. Появится окно Конструктора.
5. Заполните поля в Конструкторе данными из *рисунка 5.2*. Тип данных можно выбрать из меню, появившемся при нажатии на кнопку в ячейке справа.

Обратите внимание*: ключевое поле «Счетчик» внесен в таблицу автоматически. Если напротив поля отсутствует значок ключа, то на панели инструментов щелкните по этому значку.*

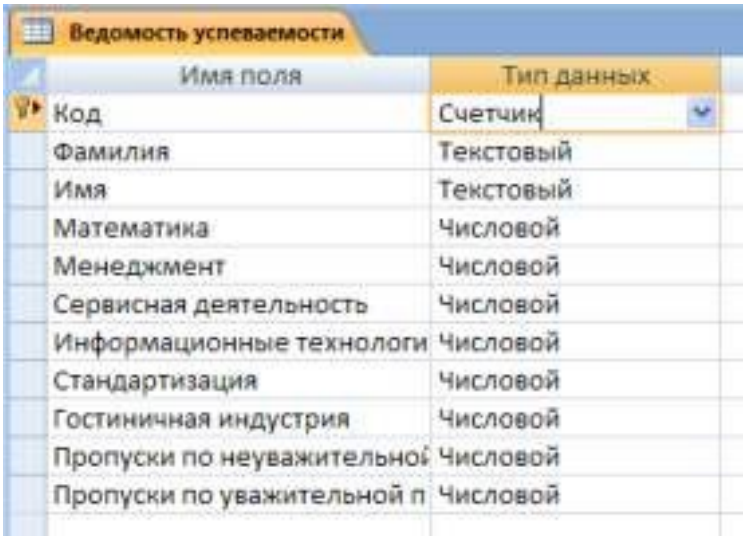

Рисунок 5.2. Создание таблицы через конструктор

6. Перейдите в режим таблицы, щелкнув по кнопке **Режим** на панели инструментов, Введите данные в этом режиме, заполняя клетки таблицы. Значение поля **Код** будет меняться автоматически.
7. Заполните базу данных значениями из *таблицы 5.1*. Напротив каждой фамилии выставьте по всем дисциплинам оценки от 2 до 5

Таблица 5.1.

Код	Фамилия	Имя	Математика	Менеджмент	Сервисная деятельность	Информационные технологии	Стандартизация	Гостиничная индустрия	Пропуски по неуважительной причине	Пропуски уважительной причине
1	Иваникова	Анна								
2	Баранова	Ирина								
3	Корнилова	Ольга								
4	Воробьев	Алексей								
5	Воробьев	Олег								
6	Скоркин	Алекс								
7	Володина	Нина								
8	Новоселов	Алексей								
9	Петрова	Елена								
10	Чернова	Кристина								
11	Терещинка	Инна								
12	Истратов	Максим								
13	Бондарь	Ольга								
14	Ревин	Олег								
15	Шарова	Оксана								

8. Выполните редактирование ячеек:
   – Замените фамилию Иванникова на Иванова.
9. Отсортируйте:
   а) *фамилии* – по алфавиту (поставьте маркер на любую фамилию в столбце

Фамилия и щелкнете мышкой по кнопке  на панели инструментов или произведите сортировку с помощью контекстного меню)

   б) *имя* – по алфавиту
10. Сохраните текущую таблицу, щелкнув по кнопке «крестик» в правом верхнем углу окна таблицы.
11. Откройте снова свою базу данных.
12. Выполните поиск записей по образцу: *найти студентку по фамилии Володина*. Для этого установите курсор в поле фамилия, щелкните на кнопке  Бинокль> на панели инструментов меню Главная и в появившемся диалоговом окне введите в поле <Образец> фамилию *Володина* и щелкните по кнопке <Найти>.

    Примечание: Если требуется найти следующую подобную запись, то щелкните мышкой по кнопке <Найти далее>. По окончании работы щелкните по кнопке <Отмена>.

13. Переименуйте поле «Математика» на «Информатика» с помощью контекстного меню.

    (Верните все как было назад).
14. Скройте столбец Пр н/пр., потом отобразите его назад.
15. Войдите в режим *Конструктора* и назначьте полю Пр н/пр и Пр ув/пр. *Маску ввода* 00 «часов». Заполните эти поля данными от 0 до 99.
16. Завершите работу с Access.

Практическая работа №6 «Создание ключевых полей. Задание индексов. Установление и удаление связей между таблицами»

Цели работы: научиться самостоятельно создавать ключевое поле; закрепить навыки по удалению, добавлению, заполнению и редактированию таблиц; научиться использовать фильтр в таблице. **Microsoft Office Access**

Access является наиболее сложной программой из всех офисных приложений Microsoft Office. Чтобы начать работу с этой программой, вначале необходимо создать структуру базы данных, подробно ее описать, а затем создать различные формы.

ACCESS – это реляционная СУБД. Это означает, что с ее помощью можно работать одновременно с несколькими таблицами базы данных, эти таблицы между собой связаны. Таблицу ACCESS можно связать с данными, хранящимися на другом компьютере. Данные ACCESS очень просто комбинировать с данными EXCEL, WORD и другими программами Office.

Access во многом похож на Excel. Основное различие между таблицей БД и электронной таблицей – в системе адресации: в электронной таблице адресуется каждая ячейка, а в таблице БД – только поля текущей записи.

Таблицы – основные объекты базы данных (БД). В них хранятся данные. Реляционная база данных может иметь много взаимосвязанных таблиц. Сведения по разным вопросам следует хранить в разных таблицах.

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

Формы – Обеспечивают более наглядную работу с таблицами, с помощью форм в базу вводят новые данные или просматривают имеющиеся.

Отчеты – средство представления данных таблиц. Отчеты могут быть оформлены надлежащим образом и распечатаны в том виде, в котором требуется пользователю.

Макросы – набор из одной или более макрокоманд, выполняющих определенные операции (открытие форм, печать отчетов)

Модули - это программы, написанные на языке программирования Visual Basic.

Задание для практической работы

Создайте таблицы и схему данных по заданным критериям 1) Откройте учебную базу данных <Анкета ГС-31>.

2) Создайте таблицу <Преподаватели > в *Режиме таблицы*. Для этого в меню Создание выберите кнопку Таблица. В появившейся таблице сделайте следующее:
   - Добавьте два поля – Поле 1 и Поле 2, выполнив команду через контекстное меню.
   - Переименуйте <Поле 1> на <Предмет>. Для этого поставьте курсор в любую ячейку столбца <Поля 1> и выполните команду *Переименовать столбец* из контекстного меню. Или щелкните два раза по имени поля, удалите старое название и впечатайте новое.
   - Переименуйте аналогично <Поле 2> на <Преподаватель>.
3) Сохраните таблицу с именем <Преподаватели>, щелкнув по кнопке <Сохранить> (дискетка на панели инструментов).
4) Перейдите в режим <Конструктор> и удалите строку с ключевым словом Счетчик. Посмотрите как заданы поля. Сделайте поле <Предмет> ключевым, поместив курсор на имя этого поля и щелкнув по кнопке - *Ключевое поле*. Тип данных поля задайте *текстовым*.
5) Перейдите в *Режим таблицы* и заполните таблицу <Преподаватели> записями из *Рисунок 6.1*.

Ведомость успеваемости | преподаватели | Таблиц

предмет	преподавате
Математика	Бекетова Н.И.
Менеджмент	Казумова Н.С.
Сервисная деятельность	Бессарабова Т.В
Информационные технологии	Бабич О.А.
Стандартизация	Казарян Г.Г.
Гостиничная индустрия	Казарян Г.Г.

Рисунок 6.1. Таблица «Преподаватели»

6) Закройте таблицу <Преподаватели>, сохранив все изменения.
7) Используя <Шаблон таблиц>, создайте таблицу <Личные данные> студентов с ключевым полем. Для этого:
   - Находясь на закладке <Создание> щелкните по кнопке <Шаблоны таблиц>, <Контакты>. Появится таблица уже с готовыми полями.
   - Переименуйте предложенные поля на следующие поля: <Код студента>, <Фамилия>, <Имя>, <Город>, <Адрес>, <Телефон>,

<Дата рождения>, <Фотография>, <Любимый предмет>, лишние поля удалите.

- Сохраните полученную таблицу под названием <Личные данные>. Ключевое поле задано автоматически.

8) Внесите данные в новую таблицу, заполнив поля <Фамилия>, <Имя>, <Город>, <Адрес>, <Телефон>, <Дата рождения>.

ПРИМЕЧАНИЕ. Поля <Фамилия> и <Имя> можно скопировать из таблицы <Ведомость успеваемости>. В поле <Город> внесите четыре разных города (например, Новороссийск, Геленджик, Анапа, Крымск)

9) Перейдите в режим <Конструктор> и назначьте типы данных: для поля <Телефон> - *числовой*, для поля <Дата рождения> - *дата/время*, для поля <Фотография> – *поле объекта OLE, для остальных – текстовый.*

Для поля <Любимый предмет> выполните свойство выбор предмета из списка с помощью *Мастера подстановок*. Для этого в строке <Любимый предмет> в поле *Тип данных – текстовый* щелкните по кнопке и в ниспадающем меню выберите команду <Мастер подстановок>.

- В диалоговом окне <Создание подстановки> поставьте флажок напротив способа <*Будет введен фиксированный набор значений*> и нажмите <Далее>.
- В следующем окне внесите в столбец все предметы (предметы из таблицы <Преподаватели>), нажмите <Далее>.
- В последнем окне, не изменяя имени столбца нажмите <Готово>.

10) Перейдите в режим таблицы и выберите для каждого студента с помощью кнопки из списка любимый предмет.

11) Создайте *схему данных*, т.е. установите связи между таблицами.

- Щелкните по кнопке - *Схема данных* на панели инструментов меню <Работа с базами данных>. В окне <Отобразить таблицу> выделите таблицу <Ведомость успеваемости> и щелкните по кнопке <Добавить>. Также добавьте таблицы <Преподаватели> и <Личные данные>. В окне <Схема данных> появиться условный вид этих таблиц. Закройте окно <Добавление таблицы>.
- Поставьте мышку на имя поля <Предметы> в таблице <Преподаватели> , и не отпуская кнопку мыши перетащите его на поле <Любимый предмет> таблицы <Личные данные>. Отпустите мышку. Появиться диалоговое окно <Связи>, в котором включите значки «*Обеспечение целостности данных*», «*Каскадное обновление связанных полей*» и «*Каскадное удаление связанных полей»*. Щелкните по кнопке <Создать>. Появиться связь «один-ко-многим».

- Поставьте мышку на имя поля <Код студента> в таблице <Личные данные> и перетащите его, не отпуская мышки, на поле <Код> таблицы <Ведомость успеваемости>. В появившемся окне <Связи> включите значок «*Обеспечение целостности данных*» и щелкните по кнопке <Создать>. Появиться связь «один-кодному».
- Закройте схему данных, сохранив ее.

12) Произведите фильтрацию данных в таблице <Личные данные> *по выделенному*.
    - Откройте таблицу в режиме таблицы.
    - Выберите студентов, проживающих в Новороссийске. Для этого поставьте курсор в одну из первых записей, где есть город Новороссийск и щелкните по кнопке - *Фильтр по выделенному* на панели инструментов. Выберите команду <Равно «Новороссийск» >. Access отобразит все записи, удовлетворяющие критерию фильтрации. 
    - Для отображения всех записей выполните команду <Удалить фильтр> для этого щелкните по соответствующей кнопке на панели инструментов  .
13) Закончите работу с базой данных Access.

Практическая работа №7 «Редактирование, добавление и удаление записей в таблице. Применение логических условий к записям. Открытие, редактирование и пополнение табличного файла»

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

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

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

1. поле, которое используется для сравнения;
2. оператор, описывающий тип сравнения;
3. величина, с которой должно сравниваться значение поля. Выражения и операторы, применяемые в условиях отбора.

Выражения и операторы	Описание выражений и операторов
Числа	**Вводятся без ограничений**
Текст	Должен быть заключен в кавычки
Даты	Ограничиваются с двух сторон символами # (например, #01.02.02#)
*; +; -; /; ^	Арифметические операторы, связывающие выражения
<; <=; >; >=; =; <>	Операторы сравнения
And (И); Not (Нет); Or (Или)	Логические операторы
Like	Используется для логики замены в выражениях
In	Для определения, содержится ли элемент данных в списке значений
Between… And…	Для выбора значений из определенного интервала
?	Заменяет один символ (букву или цифру)
*	Заменяет несколько символов

Запросы могут быть простые, сложные перекрестные.

Задание для практической работы

Создайте запросы к вашей базе данных

1) Откройте свою учебную базу данных.
2) Создайте запрос на выборку студентов, у которых по всем предметам только хорошие оценки с помощью *Мастера запросов*.

- На панели инструментов выберите команду <Мастер запросов>.

• В появившемся диалоговом окне выберите <Простой запрос> и щелкните по кнопке <OK>.

• В следующем окне выберите таблицу, по которой строится запрос (<Ведомость успеваемости>), и те поля, которые участвуют в запросе. Перенесите их в правую часть окна с помощью кнопки , нажмите <Далее>. В следующем окне тоже нажмите <Далее>.

• В другом окне дайте название запроса «Хорошисты» и нажмите <Готово>.

• Появится таблица <Хорошисты>, в которой отражены фамилии всех студентов и изучаемые предметы.

• Откройте таблицу «Хорошисты», перейдите в режим <Конструктор>. Здесь в поле <Условия отбора> под каждым предметом поставьте условие >=4 или 4OR5.

Примечание: Галочки в каждом поле означают, что по вашему выбору можно включить или убрать любое поле на выборку.

• Перейдите в режим таблицы, ответив <Да> на вопрос о сохранении запроса. (В таблице должны остаться фамилии «хорошистов»).

3) С помощью <Конструктора запросов> создайте запрос на выборку по таблице <Личные данные>.

• Щелкните по таблице <Личные данные>, зайдите в меню <Создание>, выберите команду <Конструктор запросов >.

• Добавьте нужную таблицу в поле запроса. Выделите еѐ в списке и щелкните по кнопке <Добавить>. Закройте окно <Добавление таблицы>.

• Выберите студентов, чьи фамилии начинаются на букву «В» и которые проживают в Анапе. Для этого:

- добавьте в строку <Поле> два поля <Фамилия> и <Город>;
- в строке <Условия отбора> в первом столбце укажите значение Like ―В * ‖, а во втором столбце с названием <Город> - «Анапа»;
- закройте запрос, сохранив его под названием ―ВВВ‖ (у вас должны остаться в списке студенты, проживающие в Анапе). Рисунок 7.1.

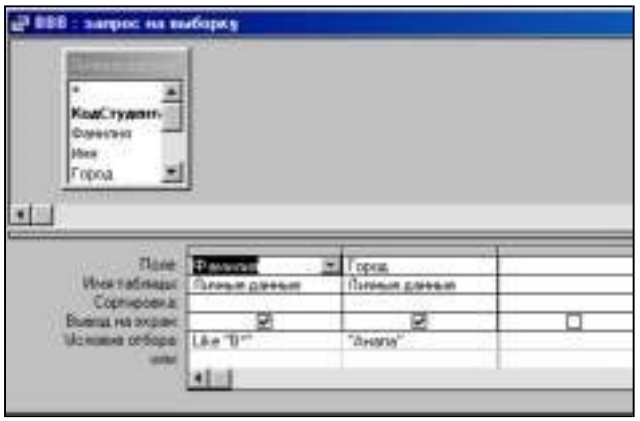

Рисунок 7.1. Запрос на выборку

Индивидуальные задания к практической работе

а) Составьте запрос с названием <Запрос 1> на базе таблицы <Ведомость успеваемости>, в котором будут указаны студенты, имеющие по первым двум предметам оценки не менее «4».

(Выполните запрос или через *Конструктор запросов*, или через *Мастер запросов*)

б) Составьте <Запрос 2> на базе таблицы <Ведомость успеваемости>, в котором будут указаны студенты, имеющие не более 30 часов пропусков по неуважительной причине. Добавьте в этот запрос поле пропуски по уважительной причине в интервале от 30 часов до 45 часов

(используйте оператор *Between… And…*)

в) Составьте <Запрос> на базе таблицы <Личные данные>. Выведите список студентов, которым на данный момент, т.е. на сегодняшнее число, исполнилось уже 17 лет (используйте оператор *Between… And…*)

Примечание: Дата записывается с использованием символа #, например, #01.02.02.#

4) Составьте запрос на базе трех таблиц <Ведомость успеваемости>, <Личные данные> и <Преподаватель>. Выберите студентов, которые проживают в Новороссийске и у которых любимый предмет «Менеджмент». Озаглавьте <Запрос 4>. Используйте <Конструктор запросов>.

- В меню <Создание> выберите <Конструктор запросов>.
- Добавьте все три таблицы в поле запроса. Закройте окно <Добавление таблицы>.
- В первый столбец в строку <Поле> перетащите из первой таблицы с помощью мышки <Фамилия>, из второй таблицы во второй столбец <Город> и из третей таблицы в третий столбец строки <Поле> - <Предмет> (Рисунок 7.2).

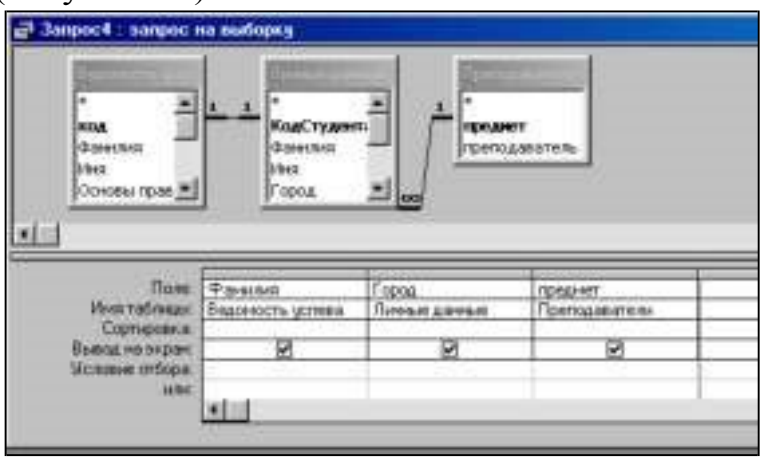

Рисунок 7.2. Запрос на выборку

- В поле <Условия отбора> в столбце <Город> введите город «Новороссийск», в столбец <Предмет> введите «Менеджмент».
- Сохраните запрос под именем <Запрос 4>.

- Откройте запрос и проверьте результат проделанной работы.

5) Выполните запрос на создание новой таблицы, в которой должны быть поля <Фамилия>, <Имя>, <Пропуски по неуважительной причине>, <Город> и <Предмет>.

- В меню <Создание> выберите <Конструктор запросов>.
- Добавьте все три таблицы из списка окна <Добавление таблицы>. Закройте это окно.
- В первую строчку <Поле> из первой таблицы перенесите в первый столбец поля <Фамилия>, во второй <Имя> и в третий <Пропуски по уважительной причине>, в четвертый столбец перетащите поле <Город> из второй таблицы и в последнем столбце будет поле <Предмет> из третьей таблицы.
- Закройте запрос, сохранив его с именем <Запрос 5>.

6) Создайте *перекрестный запрос*.

Допустим, нужно посчитать для ведомости, сколько в группе человек получили по предмету ―троек‖, ―четверок‖ и ―пятерок‖. Для этих целей используется *перекрестный запрос*.

- В меню <Создание> выберите <Мастер запросов>.
- В диалоговом окне выберите <Перекрестный запрос>, щелкните по кнопке <OK>.
- В окне <Создание перекрестных запросов> выделите таблицу <Ведомость успеваемости> и щелкните <Далее>.
- Выберите поля, значения которого будут использоваться в качестве заголовок строк – это <Фамилия> и <Имя>. Щелкните по кнопке <Далее>.
- Выберите поле, значение которого будут использоваться в качестве заголовков столбцов, например <Менеджмент>. Щелкните по кнопке <Далее>.
- Выберите функцию, по которой будут вычисляться значения ячеек на пересечении столбцов и строк (в данном случае Count – количество). Щелкните по кнопке <Далее>.
- Задайте имя запроса <Итог по менеджменту> и щелкните по кнопке <Готово>. Составьте аналогичные запросы для оценок по трем другим предметам.

7) Предъявите преподавателю все запросы своей базы данных на экране дисплея.

8) Завершите работу с Access.

Практическая работа №8 «Проведение сортировки и фильтрации данных. Поиск данных по одному и нескольким полям. Поиск данных в таблице»

Цели работы: Научиться создавать формы ввода-вывода; научиться создавать кнопочные формы.

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

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

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

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

Кнопочное меню создают с помощью Диспетчера кнопочных форм.

Отчет – это гибкое и эффективное средство для организации просмотра и распечатки итоговой информации. В отчете можно получить результаты сложных расчетов, статистических сравнений, а также поместить в него рисунки и диаграммы. Пользователь имеет возможность разработать отчет самостоятельно (в режиме *Конструктора*) или создать отчет с помощью *Мастера*, т.е. полуавтоматически. **Задание для практической работы**

Задание 1. Создайте формы к базе данных 1) Откройте свою базу данных.

2) Создайте форму с помощью <Мастера форм> на базе таблицы <Ведомость успеваемости>.

- Откройте таблицу <Ведомость успеваемости>.
- Выберите закладку <Формы >, щелкните мышкой по кнопке <Другие формы>. 

- В появившемся диалоговом окне выберите <Мастер форм>.
- В поле <Таблицы/Запросы> выберите таблицу <Ведомость успеваемости>, в поле <Доступные поля> выберите поля <Фамилия>, <Имя> и перенесите их стрелкой в поле <Выбранные поля>. Также перенесите поля с названием предметов, щелкните по кнопке <Далее>.
- Выберите внешний вид формы – Табличный, щелкните по кнопке <Далее>.
- Выберите требуемый стиль (н-р, Обычная), щелкните по кнопке <Далее>.
- Задайте имя формы <Успеваемость> и щелкните по кнопке <Готово>. В результате получите форму, в которой можно менять данные и вводить новые значения. □Закройте форму.

3) Создайте форму на основе таблицы <Преподаватели>. □ Откройте таблицу <Преподаватели>.

- Выберите закладку <Формы >, щелкните мышкой по кнопке <Другие формы>. 
- В появившемся диалоговом окне выберите <Мастер форм> .
- Выберите внешний вид формы - < ленточный>.
- Выберите любой стиль.
- Получите готовую форму. Сохраните ее под именем <Преподаватели>. □ Закройте форму.
- Создайте форму <Личные данные> с помощью инструмента <Пустая форма>
- На вкладке Создание в группе Формы щелкните Пустая форма. 
- Access открывает пустую форму в режиме макета и отображает область Список полей.
- В области Список полей щелкните знак плюс (+) рядом с таблицей или таблицами, содержащими поля, которые нужно включить в форму.
- Чтобы добавить поле к форме, дважды щелкните его или перетащите его на форму. Чтобы добавить сразу несколько полей, щелкните их последовательно, удерживая нажатой клавишу CTRL. Затем перетащите выбранные поля на форму. □Закройте окно списка полей.
- Перейдите в режим Конструктора

Примечание 1. Размер окошка для названия поля и для его значений меняются мышкой.

Для этого выделите черный квадратик рамки (рамка станет цветной), установите курсор на границу рамки и с помощью двунаправленной стрелки измените размеры рамки.

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

- Расположите элементы удобно по полю.
- Задайте размер текста поля <Фамилия> равным 24 пт, шрифт - синего цвета.
- Увеличьте в высоту рамку поля <Фотография>.
- Сохраните форму с именем <Данные студентов>.
- Посмотрите все способы представления форм: в режиме Конструктора, режиме Макета и режиме Форм.
- Закройте форму.

4) Добавьте в таблицу <Личные данные> логическое поле <Институт> (т.е., собирается ли в дальнейшем учащийся поступать в институт). Значение этого поля <ДА> или <НЕТ>.

- Откройте таблицу <Личные данные> в режиме Конструктор. Добавьте поле с именем <Институт> и типом Логический. Закройте таблицу.
- Перейдите на закладку Формы и откройте форму <Данные студентов> в режиме Конструктор
- Щелкните по кнопке <Список полей> на панели инструментов, выделите название <Институт> и перетащите его мышкой в область данных, появиться значок и надпись <Институт>.
- Расположите новые элементы по правилам оформления формы (с помощью мыши). ☐Закройте <Список полей>

Примечание 3. Если флажок установлен, поле в таблице имеет значение <ДА>, если снят, то <НЕТ>.

- Перейдите в режим <Раздельная форма> и посмотрите записи. Установите флажки у восьми разных учащихся.
- Закройте форму, ответив утвердительно на вопрос о сохранении.

5) Создайте кнопочную форму <Заставка> с помощью Конструктора. ☐Щелкните по кнопке <Создать>.

- Выберите <Конструктор>. Появиться пустая форма. Задайте мышкой ширину формы, равную 10см, а высоту – 7см.
- Сохраните работу с именем <Заставка>.
- Откройте созданную форму <Заставка> в режиме Конструктора.
- Выберите на панели инструментов <Элементы управления> кнопку Аа – <Надпись>. Курсор мышки примет вид крестика с «приклеенной» буквой А. Щелкните мышкой по месту начала надписи и введите:

(после слов База данных нажмите одновременно комбинацию клавиш Shift+Enter.)

- Нажмите клавишу <Enter>. Выберите размер букв 18, а выравнивание - по центру. Цвет фона – голубой. Растяните мышкой надпись на ширину окна.
- Выберите на панели элементов значок - Кнопка. Щелкните мышкой по тому месту области данных, где должна быть кнопка. Появиться диалоговое окно <Создание кнопок>.
- Выберите категорию <Работа с формой>, а действие <Открыть форму>, и щелкните по кнопке <Далее>.
- Выберите форму <Успеваемость> , открываемую этой кнопкой щелкните по кнопке <Далее>. В следующем окне также щелкните по кнопке <Далее>.
- В следующем окне поставьте переключатель в положение <Текст>, наберите в поле слово <Успеваемость> (Рисунок8.1) и щелкните по кнопке <Далее>.

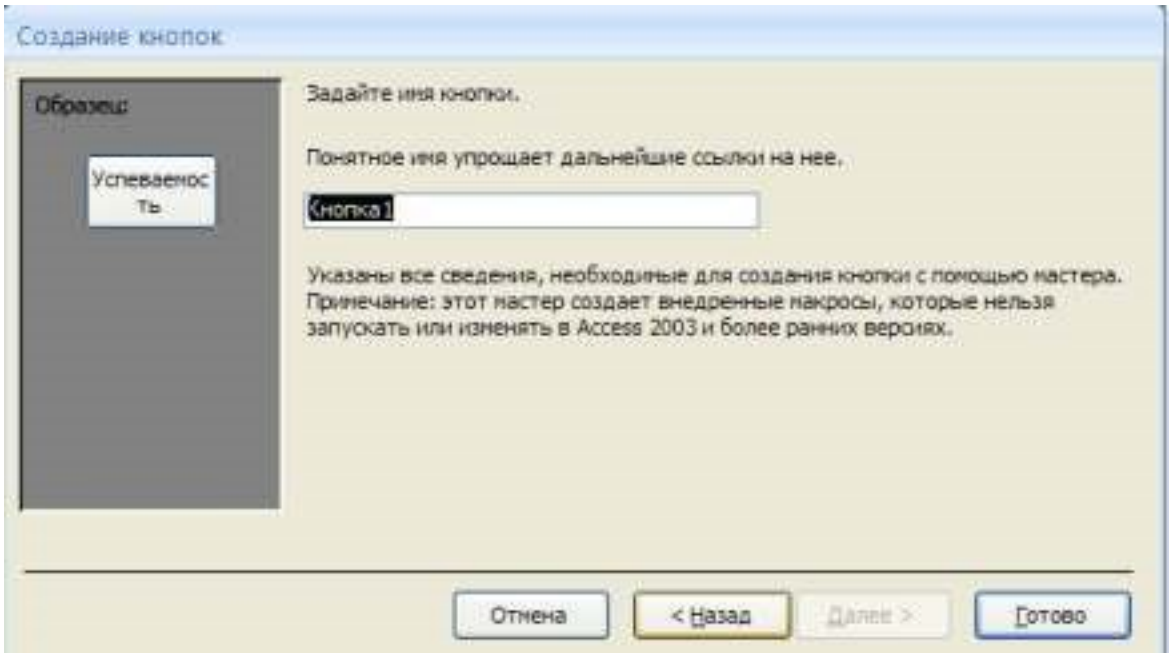

Рисунок 8.1. Создание кнопок

- Задайте имя кнопки <Успеваемость> и щелкните по кнопке <Готово>.

Примечание 4. Размер и расположение кнопок можно менять мышкой в режиме Конструктор.

Самостоятельно создайте кнопки для форм <Личные данные> и <Преподаватели>.

- Перейдите в режим формы (Рисунок 8.2). Теперь при щелчке мышью по соответствующим кнопкам будут открываться соответствующие формы для работы. ☐Закройте форму.

Рисунок 8.2. Окно формы

6) Создайте кнопочную форму при помощи Диспетчера кнопочных форм.

- Откройте вкладку Работа с базами данных, команда - Диспетчер кнопочных форм. Вы получите диалоговое окно, представленное на Рисунке 8.3.

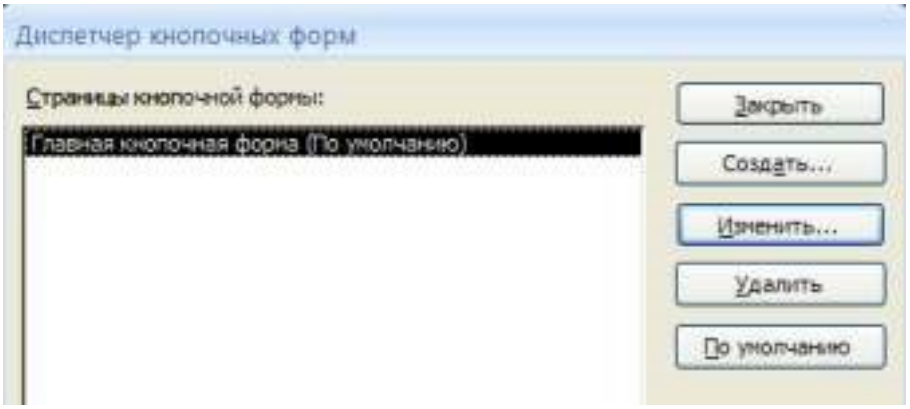

Рисунок 8.3. Диспетчер кнопок ☐ Щелкните в этом окне по кнопке <Изменить>.

- В следующем окне щелкните по кнопке <Создать> и в появившемся окне измените содержимое полей в соответствии с Рисунком 8.4 (Команду и Форму выбирайте из списка, а не набирайте вручную). Щелкните по кнопке <ОК>.

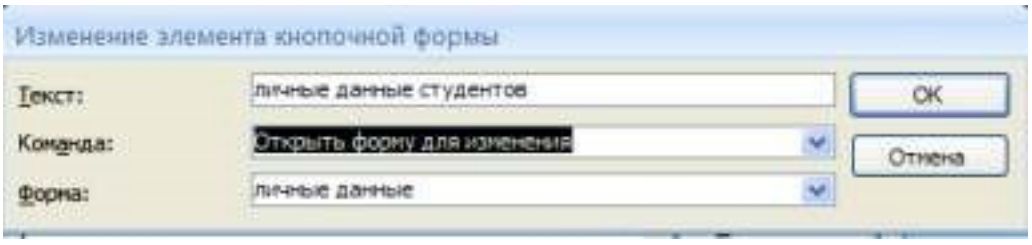

Рисунок 8.4. Изменение элементов кнопочной формы

- Аналогично создайте еще три элемента кнопочной формы: <Успеваемость>, <Преподаватели > и <Заставка>.
- Добавьте кнопку закрытия базы данных. Для этого щелкните по кнопке <Создать>, наберите в поле Текст слово <Выход>, а в поле Команда выберите <Выйти из приложения>. Закройте диалоговые окна.
- Откройте окно <Кнопочная форма> в режиме Конструктора или Макета, измените цвет надписи и название вашей базы данных на ГОСТИНИЦА, сохраните форму.
- Украсьте вашу форму рисунком. Для этого щелкните по значку Эмблема и выберите в открывшемся окне папку с рисунками, выберите понравившийся и вставьте в свою кнопочную форму.
- Перейдите в режим формы, проверьте работу всех кнопок кнопочной формы.

Завершите работу с базой данных, нажав на кнопку <Выход>.

Задание 2. Создайте отчет с помощью *Мастера отчетов.*

- Откройте вкладку *Создание,* меню *Отчеты*.
- Выберите *Мастер отчетов* и таблицу «Личные данные».
- Выберите нужные поля, которые будут участвовать в отчете, нажмите кнопку «Далее».
- В новом окне выберите поля для группировки так, чтобы сначала было указано поле «Фамилия», нажмите кнопку «Далее».
- На этом шаге отсортируйте данные по алфавиту, нажмите кнопку «Далее».
- Выберите вид макета *Ступенчатый* и щелкните по кнопке «Далее».
- Выберите стиль отчета: *Открытая* и щелкните по кнопке «Далее».
- Задайте имя отчета: «Отчет1» и щелкните по кнопке «Готово». Вы попадете в режим просмотра отчета.
- Закройте отчет согласившись с сохранением.

Самостоятельно. Составьте еще два отчета по запросам – «Запрос 3» и «Запрос 5», выбирая из разных макетов: *блок*; *структура*, выбирая из разных стилей. Сохраните отчеты под именами «Отчет 2» и «Отчет 3».

Задание 3. Создайте Пустой отчет в столбец на базе таблицы «Ведомость успеваемости» и сохраните его с именем «Успеваемость».

С помощью Конструктора измените цвет букв заголовка, их размер и шрифт.

Задание 4. Создайте почтовые наклейки.

- Откройте вкладку *Создание,* меню *Отчеты*.
- Выберите таблицу «Личные данные», команда Наклейки.
- В следующем окне щелкните по кнопке «Далее».

- В следующем окне выберите шрифт, размер шрифта, насыщенность и цвет, вновь щелкните по кнопке «Далее».
- В следующем окне создайте прототип наклейки, напечатав слово ЛИЧНОСТЬ и выбрав соответствующие поля, щелкните по кнопке «Далее».
- В следующем окне укажите поля для сортировки (Фамилия, Имя), щелкните по кнопке «Далее».
- Введите имя отчета «Наклейки» и щелкните по кнопке «Готово».
- Просмотрите Наклейки (Рисунок 8.5).

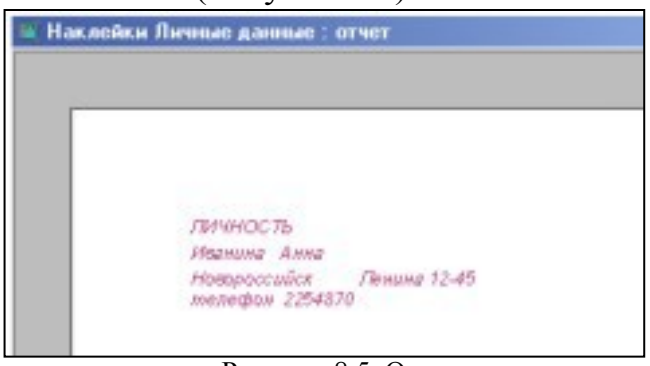

Рисунок 8.5. Отчет

Практическая работа №9 «Работа с переменными. Написание программного файла и работа с табличными файлами. Заполнение массива из табличного файла. Заполнение табличного файла из массива»

Цель работы: овладение навыками работы с переменными, обработка табличных файлов в среде VisualFoxPro.

VisualFoxPro

При создании приложения используется проект, который объединяет элементы приложения VisualFoxPro и группирует их по типам. База данных в VisualFoxPro – это совокупность таблиц, отношений между таблицами, индексов, триггеров, хранимых процедур.

База данных является частью проекта, поэтому еѐ целесообразно создавать в окне проекта.

Структурными элементами базы данных являются:

Поле – элементарная единица логической организации данных, которая соответствует неделимой единице информации – реквизиту. Для описания поля используются следующие характеристики: имя, длина, тип и точность (для числовых данных). В структуре записи файла указываются поля, значения которых являются ключами: первичными и внешними.

Первичный ключ - это одно или несколько полей, однозначно идентифицирующих запись. Если первичный ключ состоит из одного поля, он называется простым, если из нескольких полей - составным ключом.

Внешний ключ - это одно или несколько полей, которые выполняют роль поисковых или группировочных признаков. В отличие от первичного, значение внешнего ключа может повторяться в нескольких записях файла, то есть он не является уникальным. Если по значению первичного ключа может быть найден один единственный экземпляр записи, то по внешнему – несколько.

Файл (таблица) – совокупность экземпляров записей одной структуры. Описание логической структуры записи файла содержит последовательность расположения полей записи и их основные характеристики,

Запись – совокупность логически связанных полей.

Экземпляр записи - отдельная реализация записи, содержащая конкретные значения ее полей.

Таблицы составляют основу вашей базы данных. В них будет храниться вся необходимая информация. В дальнейшем данные в таблице будут дополняться новыми данными, редактироваться или исключаться из таблицы. Поля таблицы предназначены для хранения в них данных. Это могут быть числа, текстовая информация, даты, графические файлы и т.д. В VisualFoxPro допустимыми являются типы полей, указанные в таблице 9.1. Таблица 9.1.Некоторые типы данных, используемых в СУБД FoxPro

Тип	Описание	Пример
Integer	Целые числа	9846
Numeric	Данные, с фиксированной точкой	3.1456
		-56.235
		"01/07/04"
Logical	Поля, содержащие только одно из двух возможных значений (да/нет)	True; False
Date	Дата	01/07/04
DateTime	Дата и время	01/07/04 12:30:00 pm
Memo	Очень длинный текст или комбинация текста и чисел	

Таблица 9.2. Некоторые команды СУБД FoxPro

Команда	Описание
CREATE	создание элементов БД (базы, таблиц, отчётов, запросов и т.д.)
MODIFY (STRUCTURE)	изменение элементов БД (базы, таблиц, отчётов, запросов и т.д.)
BROWSE, EDIT, CHANGE, APPEND	команды редактирования и просмотра содержимого таблиц
OPEN DATABASE	ОткрытиеБД
CLOSE	закрытие элементов БД (базы, таблиц, отчётов, запросов и т.д.)
QUIT	Выходиз Visual FoxPro

Задание для практической работы

Создать в служебной папке Мои документы новую папку и присвоить ей имя, пример:

3курс4группafoxlab (указывать свой курс и группу) Запустить программу MicrosoftVisualFoxPro:

Пуск/программы/VisualFoxPro

Ознакомиться с элементами рабочего окна программы

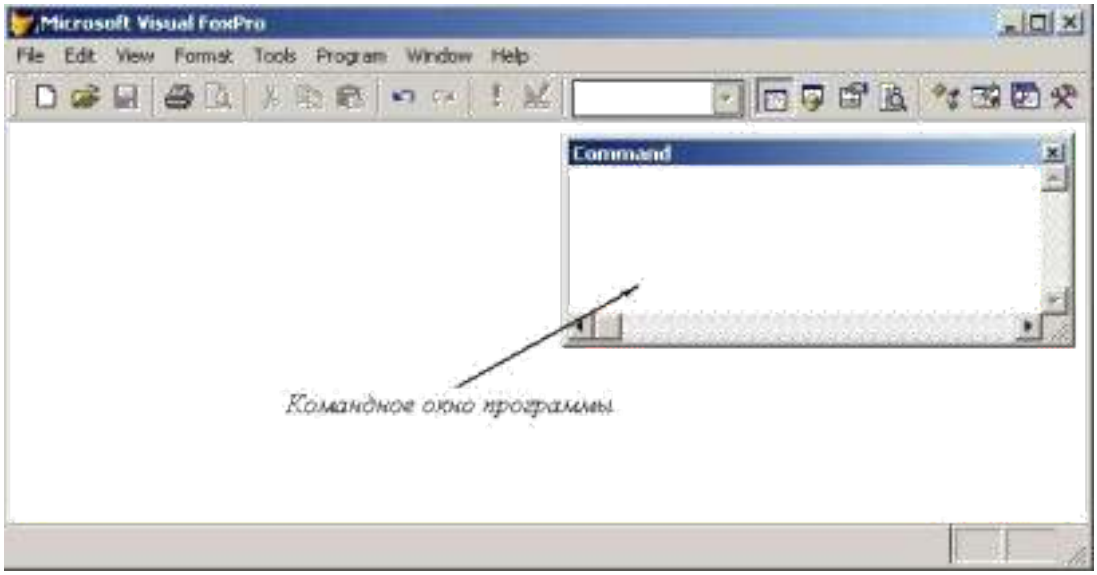

Рисунок 9.1. Главное окно VisualFoxPro 8.0

Создать новый проект: File/New/Project/NewFile, указать созданную нами ранее папку, присвоить имя проекту Информационные системы и сохранить. Все создаваемые в последующем элементы приложения будут храниться в проекте Информационные системы БД создаèтся аналогично: File/New/Database/Newfile, присвоить имя Штат исохранить. Структура проекта и его элементы отражаются в окне программы ProjectManager (Менеджер проекта)

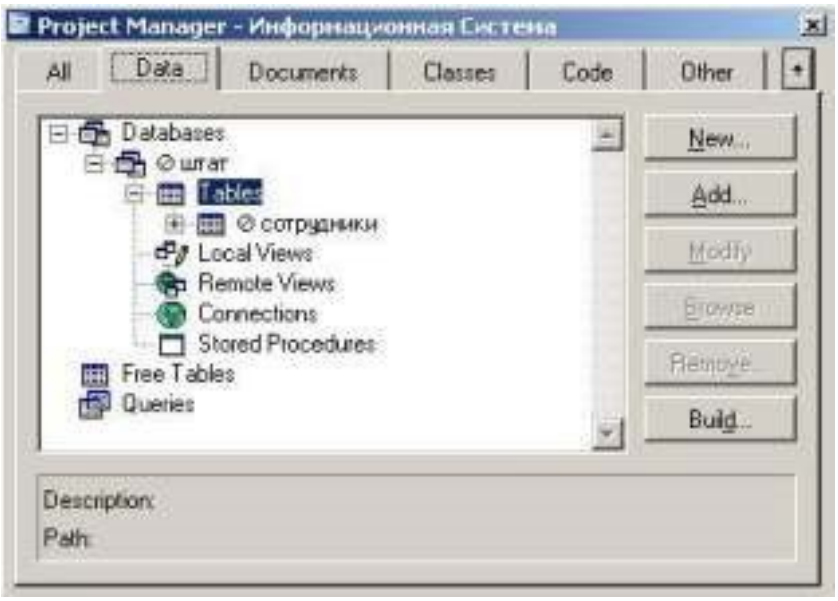

Рисунок 9.2. ОкноProject Manager

Добавить БД в проект: в окне ProjectManager щѐлкнуть на вкладке Data/Databases/Add, в открывшемся диалоговом окне выбрать БД и нажать OK

Создать таблицу в БД штат: в окне ProjectManager (см. рис.9.2.) щѐлкнуть клавишей мыши на вкладке Data/Databases/штат/Tables/New/Newtable, присвоить имя Сотрудники и сохранить

В появившемся окне TableDesigner(Конструктор таблиц) (см. рис.9.3.) на вкладке Fields (Поля) создать структуру таблицы в соответствии с таблицей 3.

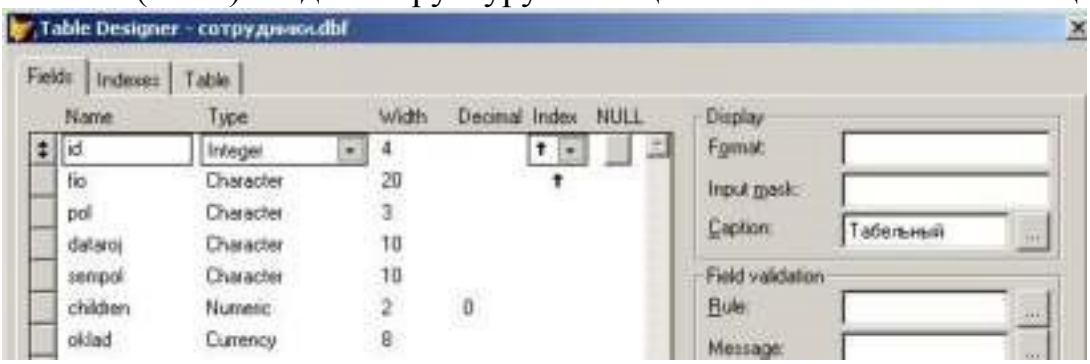

Рисунок 9.3. Окно конструктора таблицы TableDesigner вкладка Fields

Таблица 9.3.Определение полей таблицы Сотрудники в окне конструктора таблицы TableDesigner

(наименование поля)	(тип поля)	(ширина поля)	(индексное поле)	(надпись, определяет заголовок поля)
Name	Type	Width	Index	Caption
id	Integer	4	↑Ascending	Табельный номер
fio	Character	20		ФИО

pol	Character	3		Пол
dataroj	Character	10		Датарождения
sempol	Character	10		Семейное положение
children	Numeric	2		Количестводетей
oklad	Carrency	8		Оклад
doljnost	Character	15		Должность
adress	Character	30		Место жительства

Перейти на вкладку Indexes(Индексы) окна конструктора таблицы TableDesigner и присвоить созданному индексу значения в соответствии с таблицей 9.4. Это необходимо для создания ключевого поля

Order	Name	Type	Expression	Filter	Collate
↑	id	Primary	id		Machine

Для ввода и редактирования данных в таблице *Сотрудники* в командном окне программы введите команду APPEND и нажмите клавишу Enter Командное окно программы **FoxPro** (см. рис.9.1.) предназначено для ввода команд с клавиатуры и последующего их выполнения. Любые действия и операции над элементами приложения в **СУБД FoxPro** могут осуществляться при помощи команд, вводимых в программное окно.

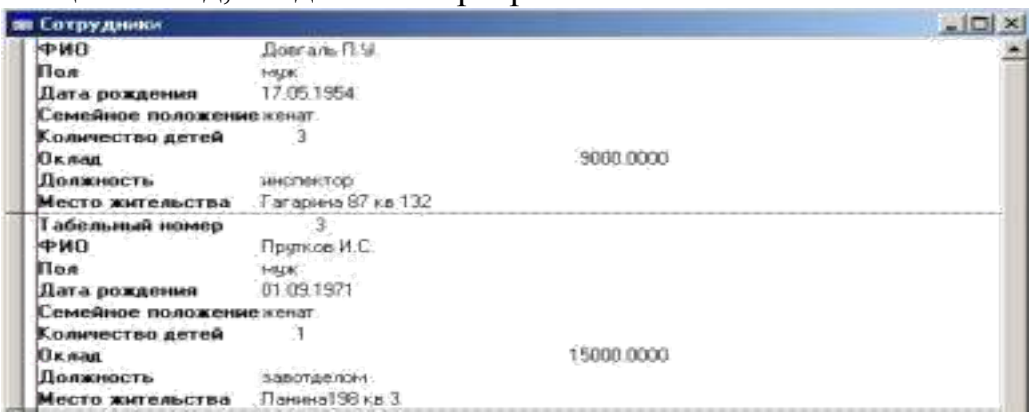

Рисунок 9.4. Просмотр и редактирование содержимого таблицы в режиме APPEND

После ввода данных (см. таблицу 9.4.) в командном окне программы введите команду BROWSE и нажмите клавишу Enter. Результат должен соответствовать рис.9.5.

Рисунок 9.5. Просмотр таблицы в режиме BROWSE

Таблица 9.4. Содержимое таблицы Сотрудники

Таб. номер	ФИО	Пол	Дата рождения	Семейное положение	Количество детей	Оклад	Должность	Место жительства
1	Гуськов Г.Г.	муж.	22.02.1976	холост	0	7000	экономист	Лаптиева 156 кв 43
2	Довгаль П.У.	муж.	17.05.1954	женат	3	9000	инспектор	Гагарина 87 кв 132
3	Прутков И.С.	муж.	01.09.1971	женат	1	15000	завотдело м	Панина198 кв 3
4	Косыгин Н.И.	муж.	28.03.1979	холост	0	7000	экономист	Николаева 30 кв 77
5	Патрико в Е.П.	муж.	14.08.1980	холост	0	4500	консульта нт	Нуралилова 18г кв 84
6	Белкина Б.Ю.	жен	24.11.1959	замуже м	4	12000	бухгалтер	Луначевского 64 кв 9
7	Петрен. о Е.П.	жен	29.09.1981	незаму жем	0	7000	экономист	Ленина 101 кв 15
8	Петросо ва Г.Н.	жен	19.06.1976	замуже м	2	4500	консульта нт	Мира пр. 120 кв 2
9	Орлова К.Д.	жен	31.12.1977	незаму жем	0	7000	экономист	Седова 18 кв 112
10	Косыгин а А.К.	жен	01.10.1969	замуже м	1	6000	менеджер	Панина 198 кв 3

В окне BROWSE установите курсор в нижней части таблицы и нажмите комбинацию клавиш Ctrl + Y на клавиатуре. В таблицу добавится новая строка Прежде чем удалить строку в VisualFoxPro еѐ сначала необходимо пометить на удаление. Для этого щелкнуть клавишей мыши в ячейке слева от удаляемой записи в области узкого, непоименованного столбца, ячейка станет помеченной чѐрным цветом, или выделить необходимую строку и выбрать команду ToggleDeletionMark (Установка метки на удаление) пункта Table

(Таблица) системного меню VisualFoxPro, запись будет помечена на удаление.

Удалить запись, выбрав команду RemoveDeletedRecords (Удалить запись) пункта Table (Таблица) системного меню VisualFoxPro. Запись будет удалена

Для закрытия всех элементов приложения в командном окне программы введите команду

CLOSE ALL

Для завершения работы с программой **VisualFoxPro** в командном окне программы введите команду QUIT.

Создание отношений между таблицами в многотабличной БД

Библиотека

Среди требований, предъявляемых к СУБД, основное место занимает возможность быстрого поиска необходимой информации. Средством, позволяющим решить эту проблему, является применение индексов. В VisualFoxPro для создания первичных ключей, определяющих отношения между таблицами и условия целостности данных также предназначены индексы. В этом случае индексы должны быть уникальными, то есть значения индексированного поля должны быть неповторяющимися (уникальными).

Для создания индекса таблицы используется вкладка Indexes (Индексы) окна конструктора таблиц TableDesigner. Все индексы в VisualFoxPro имеют имена, задаваемые в поле Name (Имя).

Для задания типа создаваемого индекса используется список Type (Тип).

Таблица 9.10.Описание типов индекса

Тип индекса	Описание
Regular (Обычный)	Создаётся индекс, в котором для каждой записи таблицы хранится значение индексного выражения. Если несколько записей имеют одинаковое значение индексного выражения, то каждое значение хранится отдельно и содержит ссылку на связанную с ней запись
Unique(Уникальный)	Создаётся индекс, в котором хранятся только неповторяющиеся значения индексного выражения. Если две или более записей содержат одинаковое значение индексного выражения, то будет храниться только одно значение и ссылка на первую из записей с одинаковым значением индексного выражения. Таблица может иметь несколько уникальных индексов
Candidate (Кандидат)	Создаётся уникальный индекс, который не содержит полей с пустыми значениями. Этот индекс обладает всеми качествами первичного ключа и не является им только по той причине, что таблица не может содержать более одного первичного ключа
Primary (Первичный)	Создаётся уникальный индекс, который используется для связывания таблиц и определения условий целостности данных. Поля, входящие в первичный ключ, не должны допускать ввода пустых значений. В отличие от уникального индекса, таблица может иметь только один первичный ключ

Обычно, в VisualFoxPro при создании форм, отчѐтов и запросов используются несколько таблиц, между которыми установлены постоянные отношения. Такие таблицы называются связанными. Из двух связанных таблиц одна является главной (родительской), а другая подчинѐнной (дочерней). При создании индексов для родительской таблицы должен быть определѐн ключ типа Primary (Первичный) или типа Candidate (Кандидат), а для дочерней таблицы – индекс для связи с родительской таблицей типа Regular (Обычный).

Таблица 9.11. Типы отношений между таблицами

Типы отношений	Описание
Отношение "один-к-одному"	Каждая запись в одной таблице соответствует только одной записи в другой таблице
Отношение "один-ко-многим"	Наиболее распространенный тип отношений, каждой записи в одной таблице может соответствовать несколько записей в другой таблице
Отношение "много-к-одному"	Отношение "много-к-одному" можно сравнить с отношением "один-ко-многим", рассматриваемое с другой точки зрения
Отношение "много-ко-многим"	Нескольким записям одной таблицы может соответствовать несколько записей в другой таблице

Одним из самых важных требований, предъявляемых к базам данных, является целостность данных, которую определяют установленные между таблицами отношения. Для определения целостности данных в VisualFoxPro используется окно построителя условий целостности данных ReferentialIntegrityBuilder (Построитель целостности данных), которое содержит перечень всех установленных отношений между таблицами (см. рис.9.3.).

Таблица 9.12.Описание действий VisualFoxPro, в зависимости от выбранной опции, при изменении значения первичного ключа или ключа типа

Наименование опции	Описание
Cascade (Каскадное изменение)	При изменении значений полей первичного ключа или ключа-кандидата в родительской таблице автоматически осуществляется каскадное изменение всех соответствующих значений в дочерней таблице
Restrict (Запрет изменения)	Не позволяет изменять значения полей первичного ключа или ключа-кандидата в родительской таблице, если в дочерней таблице имеется хотя бы одна запись, содержащая ссылку на изменяемую запись
Ignore(Игнорировать)	Позволяет изменять значения полей первичного ключа или ключа-кандидата в родительской таблице независимо от существования связанных записей в дочерней таблице. Целостность данных при этом не поддерживается

Таблица 9.13.Действия VisualFoxPro, в зависимости от выбранной опции, при удалении записи из родительской таблицы

Наименование опции	Описание
Cascade (Каскад)	При удалении записи из родительской таблицы автоматически осуществляется каскадное удаление всех записей из дочерней таблицы, связанных с удаляемой записью
Restrict (Запрет)	Не позволяет удалять записи в родительской таблице, если в дочерней таблице имеется хотя бы одна запись, содержащая ссылку на удаляемую запись. При попытке удаления записи возникает ошибка, которую вы можете обработать программно
Ignore(Игнорировать)	Позволяет удалять записи в родительской таблице независимо от существования связанных записей в дочерней таблице. Целостность данных при этом не поддерживается

Таблица 9.14.Действия **VisualFoxPro**, в зависимости от выбранной опции, при добавлении новой записи в родительскую таблицу

Наименование опции	Описание
Restrict (Запрет)	Не позволяет вводить запись, если значение индексного выражения дочерней таблицы не соответствует одной из записей в родительской таблице
Ignore(Игнорировать)	При вводе данных в дочернюю таблицу не анализируется значение индексного выражения. Целостность данных при этом не поддерживается

Индивидуальные задания к практической работе

Создать многотабличную БД и присвоить имя Библиотека. Создать следующие таблицы: 1. Таблица Книги

(наименование поля)	(тип поля)	(ширина поля)	(индексное поле)	(надпись, определяет заголовок поля)
Name	Type	Width	Index	Caption
Kodknigi	Integer	4		Код книги
nazvanie	Character	40		Название
razdel	Character	15		Раздел
izdat	Character	20		Издательство
godizdan	Numeric	4		Год издания
mestohran	Character	5		Местохранения

Определение полей таблицы Книги в окне конструктора таблицы **TableDesigner**

Содержимое таблицы Книги

Код	Название	Раздел	Издательство	Год	Место
книги				издания	хранения
1	Практический курс программирования	Информатика	Наука	1983	6-11
2	TURBOPASCAL для школьников	Информатика	Финансы и статистика	1999	6-22
3	Занимательная математика	Математика	Тригон	1998	3-14
4	HTML в действии	Информатика	Питер	1997	5-4
5	Национальное счетоводство	Экономика	Финансы и статистика	1998	4-11
6	Самоучитель VisualFoxPro 8.0	Информатика	БХВ-Петербург	2003	5-34
7	Язык телодвижений	Психология	Наука	1984	2-17
8	Теория машин	Техника	Машиностроение	1957	3-15
9	Теория гипноза	Психология	Наука	1999	2-31
10	Карьера менеджера	Экономика	Парадокс	1998	1-212

2. Таблица Разделы

(наименование поля)	(тип поля)	(ширина поля)	(индексное поле)	(надпись, определяет заголовок поля)
Name	Type	Width	Index	Caption
razdel	Character	15		Раздел

Определение полей таблицы Разделы в окне конструктора таблицы TableDesigner Содержимое таблицы Разделы

Раздел
Экономика
Информатика
Психология
Математика
Техника

3. Таблица *Издательство*

Определение полей таблицы *Издательство* в окне конструктора таблицы **TableDesigner**

(наименование поля)	(тип поля)	(ширина поля)	(индексное поле)	(надпись, определяет заголовок поля)
Name	Type	Width	Index	Caption
izdat	Character	20		Издательство
gorod	Character	15		Город

Содержимое таблицы *Издательство*

Издательство	Город
Финансы и статистика	Москва
Тригон	Санкт-Петербург
Питер	Санкт-Петербург
Наука	Москва
Машиностроение	Москва
Парадокс	Минск
БХВ-Петербург	Санкт-Петербург

4. Таблица *Автор_книги*

Определение полей таблицы Автор_книги в окне конструктора таблицы **TableDesigner**

(наименование поля)	(тип поля)	(ширина поля)	(индексное поле)	(надпись, определяет заголовок поля)
Name	Type	Width	Index	Caption
kodavknigi	Integer	4		Код автора книги
kodknigi	Integer	4		Код книги
kodavtora	Integer	4		Код автора

Содержимое таблицы *Автор_книги*

Код автора книги	Код книги	Код автора
1	1	1
2	1	2
3	2	3
4	3	4
5	4	5
6	5	6
7	6	7
8	7	8
9	8	9
10	9	10
11	10	11

5. Таблица *Авторы*

Определение полей таблицы *Авторы* в окне конструктора таблицы **TableDesigner**

(наименование поля)	(тип поля)	(ширина поля)	(индексное поле)	(надпись, определяет заголовок поля)
Name	Type	Width	Index	Caption
kodavtora	Integer	4		Код автора
familiya	Character	20		Фамилия
imya	Character	15		Имя

Содержимое таблицы *Авторы*

Код автора	Фамилия	Имя
1	Фролов	Геннадий
2	Илюхин	Виктор
3	Попов	Владимир
4	Акимова	Светлана
5	Морис	Брюс
6	Кулагина	Галина
7	Омельченко	Людмила
8	Алан	Пиз
9	Мальцев	Анатолий
10	Горин	Дмитрий
11	Яковка	Ли

6. Открыть БД штат: в окне ProjectManager (Менеджер проекта) вкладка Data/Databases/Библиотека/Open

7. Необходимо модифицировать таблицы БД Библиотека, создать индексированные поля, а также первичные ключи для осуществления связей между таблицами. Для модификации (изменения структуры) таблицы в окне проекта ProjectManager (Менеджер проекта) установить курсор на модифицируемую таблицу Книги и нажать кнопку Modify (Модифицировать) или в окне конструктора БД DatabaseDesigner (Конструктор базы данных) установить курсор в таблицу Книги, вызвать контекстное меню и выбрать команду Modify (Модифицировать). На экране откроется диалоговое окно TableDesigner (Конструктор таблицы)

8. В окне TableDesigner (Конструктор таблицы) перейти на вкладку Indexes (Индексы) и ввести значения в соответствии с таблицей 6., нажать кнопку OK, система попросит подтвердить сохранение изменений, нажать Yes

Таблица 9.15.Определение индексов таблицы Книги на вкладке Indexes (Индексы) окна конструктора таблицы TableDesigner

Order	Name	Type	Expression	Filter	Collate
↑	kodknigi	Primary	kodknigi		Machine
↑	razdel	Regular	razdel		Machine
↑	izdat	Regular	izdat		Machine

Рисунок 9.5. Вкладка Indexes (Индексы) окна TableDesigner таблицы Книги

Индекс, с присвоенным типом Primary является первичным ключом таблицы

9. Аналогично внести изменения в таблицы БД Библиотека: Разделы,

Издательство, Автор_книги, Авторы в соответствии с таблицами 9.16,9.17, 9.18, 9.19

Таблица 9.16.Определение индексов таблицы Разделы на вкладке Indexes (Индексы) окна конструктора таблицы TableDesigner

Order	Name	Type	Expression	Filter	Collate
↑	razdel	Primary	razdel		Machine

Таблица 9.17.Определение индексов таблицы Издательство на вкладке Indexes (Индексы) окна конструктора таблицы TableDesigner

Order	Name	Type	Expression	Filter	Collate
↑	izdat	Primary	izdat		Machine

Таблица 9.18.Определение индексов таблицы Автор_книги на вкладке Indexes (Индексы) окна конструктора таблицы TableDesigner

Order	Name	Type	Expression	Filter	Collate
↑	kodavknigi	Primary	kodavknigi		Machine
↑	kodavtora	Regular	kodavtora		
↑	kodknigi	Regular	kodknigi		

Таблица 10. Определение индексов таблицы Авторы на вкладке Indexes (Индексы) окна конструктора таблицы TableDesigner

Order	Name	Type	Expression	Filter	Collate
↑	kodavtora	Primary	kodavtora		Machine

10. В окне DatabaseDesigner (Конструктор базы данных) выбрать родительскую таблицу Разделы. Таблицы в конструкторе БДобозначаются

прямоугольниками, в нижней части которых после надписи Indexes (Индексы) расположен список индексов, созданных для данной таблицы. Первичный ключ в этом списке выделяется значком ключа, расположенным с левой стороны от наименования индекса. Установить курсор мыши на первичный ключ razdel таблицы Разделы. Нажать кнопку мыши и, не отпуская ее, переместить курсор мыши на индекс razdel дочерней таблицы Книги. Отпустить кнопку мыши. Созданные отношения между таблицами отображаются в виде линий. Аналогичным образом, создать связь между таблицами Издательство и Книги

(Таблица Издательство – родительская, таблица Книги – дочерняя, первичный ключ – izdat)

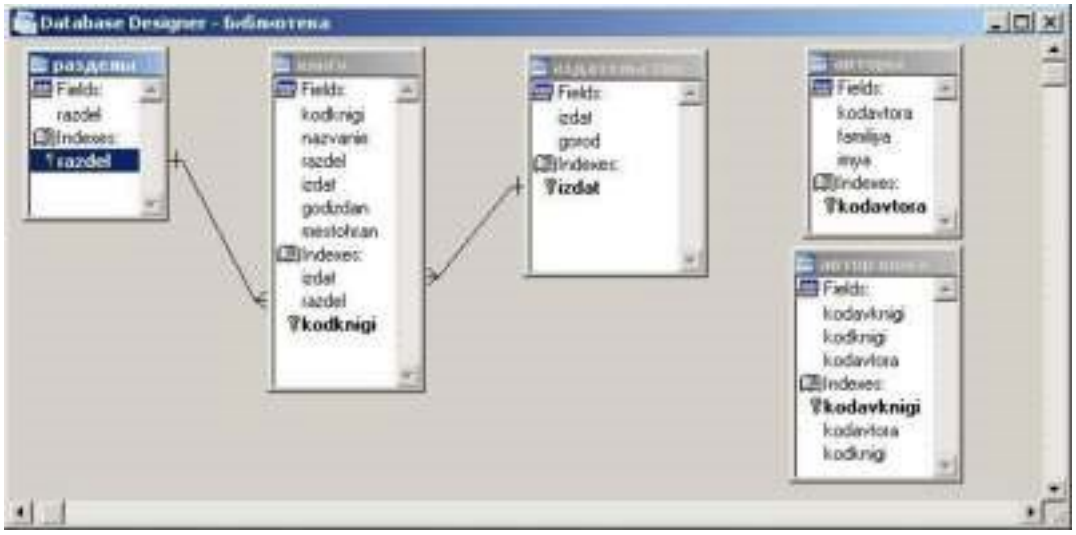

Рисунок 9.6. Отображение связей между таблицами в окне DatabaseDesigner (Конструктор базы данных)

11. Для редактирования отношений между связанными таблицами дважды щелкнуть левой кнопкой мыши на линии, появится диалоговое окно EditRelationship(Редактирование отношения), в котором слева приведено наименование родительской таблицы и расположен раскрывающийся список индексов таблицы, справа размещена аналогичная информация о дочерней таблице. В этом диалоговом окне указан также тип установленного отношения между таблицами.

9. В окне EditRelationship(Редактирование отношения) нажать кнопку ReferentialIntegrity(Целостность данных), в появившемся диалоговом окне

ReferentialIntegrityBuilder (Построитель целостности данных) выбрать отношение издательствокниги. В полях Update (Изменить), Delete (Удалить), Insert (Заменить) установить тип действий Restrict (Запрет изменения). Провести аналогичные действия для отношения разделы-книги. Результат описанных действий, которые необходимы для обеспечения целостности данных, представлен на рис.9.7.

10. Для сохранения выполненных действий нажать кнопку OK, система потребует подтверждение сохранения, нажать кнопку Да

11. Нажать кнопку OK для закрытия диалогового окна EditRelationship

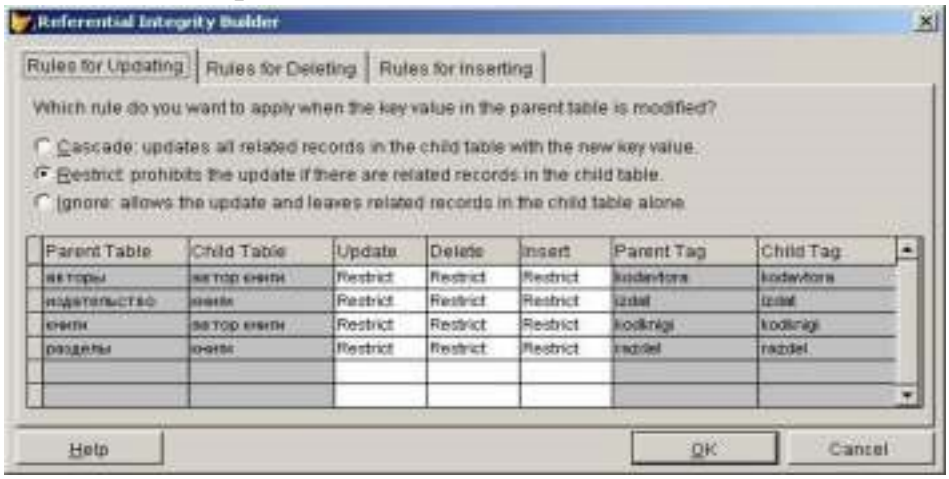

Рисунок 9.7. Диалоговое окно ReferentialIntegrityBuilder(Построитель целостности данных)

12. Создать связь между таблицами Книги и Автор_книги (kodknigi)

13. Создать связь между таблицами Авторы и Автор_книги (kodavtora)

14. Установить целостность данных в созданных отношениях: в полях Update (Изменить),

Delete (Удалить), Insert (Заменить) установить тип действий Restrict (Запрет изменения)

Практическая работа №10 «Создание формы. Управление внешним видом формы»

Цель работы: овладение практическими навыками по созданию фом

Формы в VisualFoxPro

В VisualFoxPro для просмотра, ввода и редактирования данных, хранящихся в таблицах, используются формы, являющиеся более наглядным средством представления информации. Важным преимуществом форм является, то, что они позволяют работать не с одной, а с несколькими связанными таблицами, что, в свою очередь, также увеличивает наглядность.

При создании форм в VisualFoxPro разработчик может использовать следующие средства: FormWizard (мастер форм), FormBuilder (построитель формы), Builder (построитель объектов формы), AutoFormatBuilder (построитель автоформата), FormDesigner (конструктор форм).

Чтобы создать форму для одной или связанных таблиц с возможностью задания отображаемых в форме полей, стиля их отображения и указания типа кнопок управления, можно использовать FormWizard (мастер создания форм).

Для самостоятельной разработки формы с заданными свойствами или изменения формы, созданной с помощью мастера, вам необходимо использовать FormDesigner (конструктор форм).

Для облегчения размещения в конструкторе форм полей и надписей, оформленных в соответствии с выбранным стилем, можно использовать FormBuilder (Построитель формы).

Создание формы с помощью конструктора форм

Любая форма в VisualFoxPro состоит из объектов, каждый из которых имеет характерные свойства. Для любого объекта вы можете указать действия, выполняемые написанной разработчиком программой при наступлении определѐнных событий. Процесс создания формы в конструкторе форм состоит в размещении в форме объектов и определении свойств, а также связанных с ними событий и выполняемых действий.

Процесс создания формы включает следующие действия:

- настройка параметров формы
- определение среды окружения, то есть выбор используемых в форме таблиц и установка связей между ними
- размещение в форме объектов: текста, полей различных типов, линий, рисунков, кнопок управления.

Задание для практической работы

Задание 1. Создание формы средствами мастера форм

Запустить программу MicrosoftVisualFoxPro

Открыть созданный проект: File/Open/Информационная система

Открыть БД штат: в окне ProjectManager (Менеджер проекта) вкладка Data/Databases/штат/Open

Создать форму Сотрудники, щелкнув клавишей мыши на вкладке Documents/Forms/New/FormWizard/FormWizard, нажать клавишуOk

Рисунок 10.1. Диалоговое окно WizardSelection

В появившемся окне FormWizard в области Databasesandtables (Базы данных и таблицы) выбрать необходимую БД штат и указать таблицу Сотрудники, для которой создается форма. Из области

Availablefields (Имеющиеся поля) выбрать поля, которые будут размещены в форме, в соответствии с рис.10.2, используя кнопки

расположенные между списками для перехода к следующему шагу нажать кнопку Next(Далее)

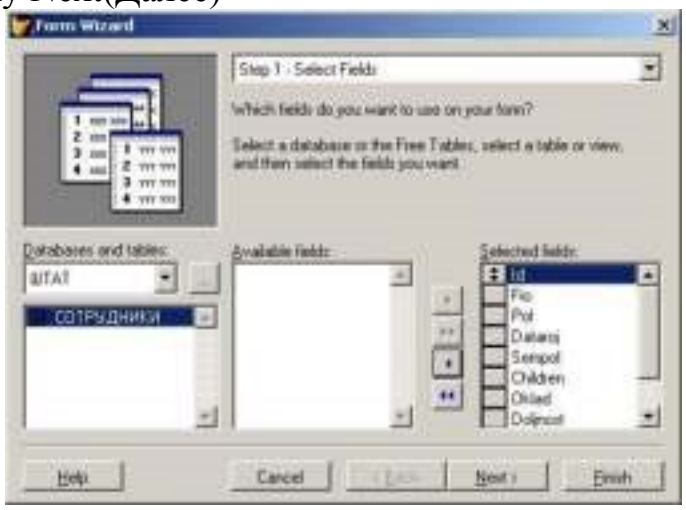

Рисунок 10.2. Диалоговое окно выбора полей для отображения.

В появившемся диалоговом окне в области Style установить стиль отображения объектов и в области Button type типы кнопок управления в соответствии с рис.10.3, нажать кнопку Next

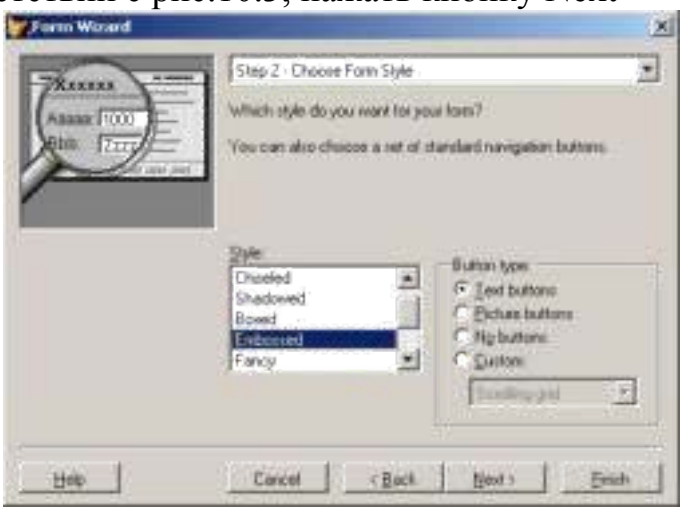

Рисунок 10.3. Окно выбора стиля отображения полей и управляющих кнопок.

В появившемся диалоговом окне задать критерий сортировки данных, указав поля по которым будет осуществляться упорядочивание в соответствии рис.10.4. нажать кнопку Next

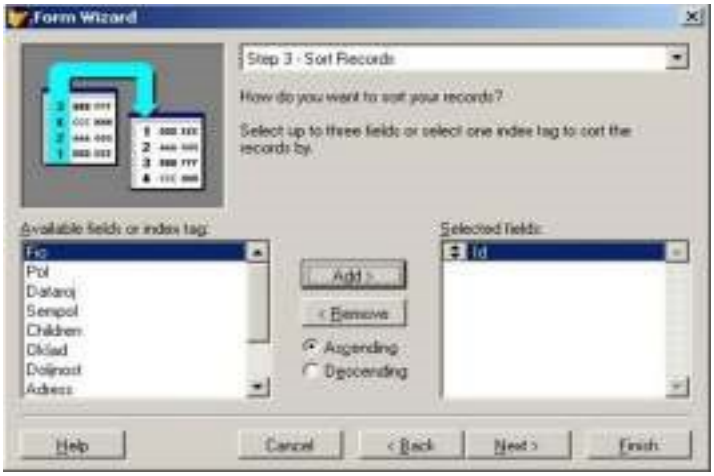

Рисунок 10.4. Установка критерия упорядочения данных

На заключительном шаге создания форм в области Typeatitleforyourform(Тип заголовка формы) указать имя формы

Сотрудники. Воспользовавшись кнопкой Preview (Просмотр) просмотрите, как будет выглядеть создаваемая форма, после просмотра нажмите кнопку ReturntoWizard.

Рисунок 10.5.Окно задания заголовка формы.

Если что-то не так, вернитесь к предыдущим шагам, воспользовавшись кнопкой Back. Нажать кнопку Finish (Готово) и в появившемся диалоговом окне SaveAs (Сохранить как) указать папку, в которой будет размещена форма Сотрудники

Запустить в окне ProjectManager форму Сотрудники:

Documents/Forms/Сотрудники/Run

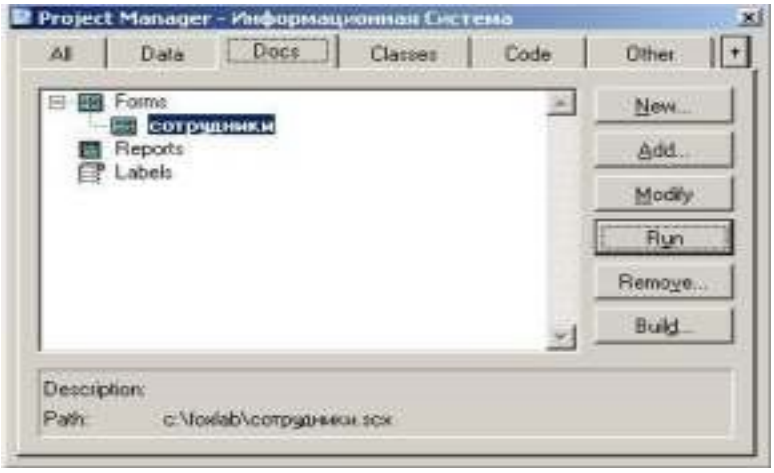

Рисунок 10.6. Окно ProjectManager

Или с помощью меню программы VisualFoxPro: Program/Dозадать имя формыСотрудникии тип файлаForm, нажать кнопкуDo. С помощью кнопок (см. таблицу 10.1.), находящихся в нижней части формы можно вводить или редактировать записи в таблице сотрудники.

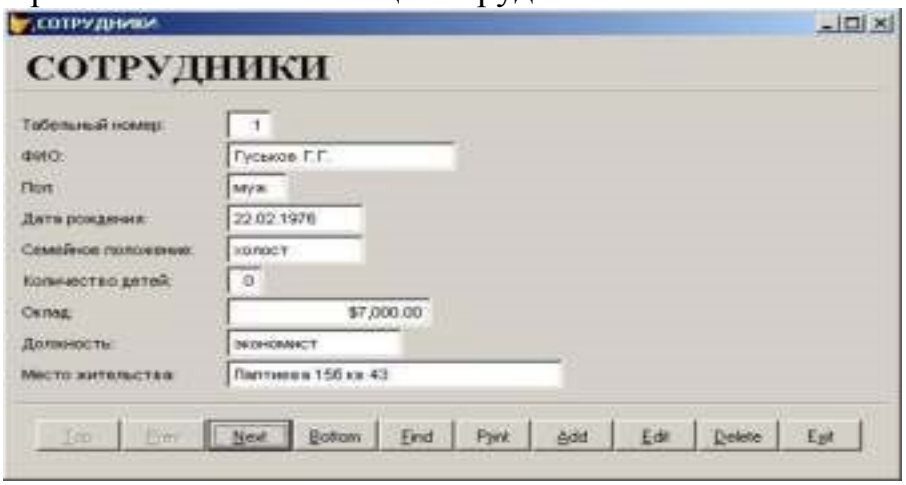

Рисунок 10.7.Форма Сотрудники

Ввести в таблицу Сотрудники дополнительно шесть записей, одну удалить, две отредактировать

Таблица 10.1.Кнопки управления и редактирования данных

Top	Prev	Next	Buttom	Find	Save
Первая запись	Предыдущая запись	Следующая запись	Последняя запись	Найти	Сохранить
Print	Add	Edit	Delete	Exit	Revent
Вывести на печать	Добавить запись	Редактировать запись	Удалить запись	Выход	Отменить

Индивидуальные задания к практической работе Создать форму для таблицы Книги БД Библиотека

1. В диалоговом окне выбора полей для отображения выбрать все поля таблицы Книги

2. В окне выбора стиля отображения полей и управляющих кнопок (см. рис.3.) в области Style (стиль) выбрать стиль Embossed, а в областиButtontype (Типы кнопок управления) выбрать пункт NoButtons
3. Установить критерий упорядочения данных по полю kodknigi
4. В окне задания заголовка формы в области Typeatitleforyourform(Тип заголовка формы) указать имя формы Книги. Просмотреть вид создаваемой формы нажатием кнопки
   Preview (Просмотр).
5. После сохранения запустить форму Книги, она должна иметь вид как на рис.10.8.

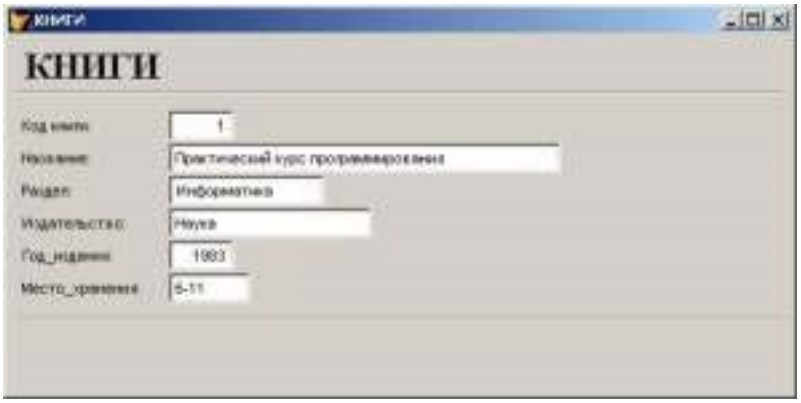

Рисунок 10.8. Форма Книги

Задание 2. Создание формы средствами конструктора форм

Запустить программу MicrosoftVisualFoxPro

Открыть проект Информационная система

Открыть БД штат Выбрать в окне Project Manager вкладку Documents/Forms/New/New Form. Откроется окно FormDesigner(Конструктор форм) для создания новой формы

Открыть окно окружения формы DataEnvironment (Среда окружения): пункт меню View/ DataEnvironment. Для размещения таблицы в среде окружения выбрать команду Add(Добавить) из меню DataEnvironment (Среда окружения) или вызвать контекстное меню на окне окружения формы DataEnvironmentи выбрать пункт Add. В появившемся диалоговом окне AddTableorView (Добавить таблицу или представление данных), выбрать из списка таблиц открытой базы штат таблицу Сотрудники, для которой создаѐтся форма, и нажать кнопку OK

Открыть окно свойств таблицы, размещѐнной в окне окруженияDataEnvironment. Для этого установить на таблицу Сотрудники курсор, вызвать контекстное меню правой клавишей мыши и выбрать команду Properties (Свойства)

В окне Properties (свойства) выделить свойство Order (Упорядочение). Для упорядочения данных в форме в поле коррекции свойства нажмите кнопку раскрытия списка и из списка индексов таблицы выберите индекс (id), по

которому хотите упорядочить данные. Закройте окно определения среды окружения DataEnvironment.

Для задания свойств формы установить курсор в форму, вызвать контекстное меню, выбрать команду Properties (Свойства). Откроется окно Properties(Свойства). В его верхнем списке, указывающем название объекта, для которого осуществляется настройка свойств содержится текст Form1 (Форма1)

В окне Properties (Свойства) скорректировать свойство Caption (Надпись), введя в текстовое поле заголовок формыСотрудники тест

Свойство формы AutoCenter (Автоцентр) должно иметь значение True(Истина), чтобы форма располагалась в центре экрана

Изменить свойства FontName (Наименование шрифта), указав шрифт TimesNewRomanu FontSize (Размер шрифта) указав размер шрифта 12 После того как вы определили параметры формы, разместили в окружении используемые таблицы, можно приступать к размещению объектов в форме. Осуществим размещение полей таблицы Сотрудники и надписей к ним, используя FormBuilder (построитель формы)

Для запуска построителя форм установить курсор в форму, вызвать контекстное меню, выбрать команду Builder (Построитель), откроется диалоговое окноFormBuilder (построитель формы), содержащее две вкладки: FieldSelection (Выбор поля) и Styles (Стиль). В вкладке FieldSelection (Выбор поля) из области Databasesandtables (Базы данных и таблицы) выбрать БД штат и таблицу Сотрудники, затем из списка Availablefields (Имеющиеся поля) перенести в Selectedfields(Выбранные поля) все необходимые поля используя кнопки расположенные между списками. Перейти на вкладку Styles (Стиль), выбрать из списка стилей стиль Embossed и нажать кнопку OK. На форме будут размещены объекты формы (см. рис. 10.9.).

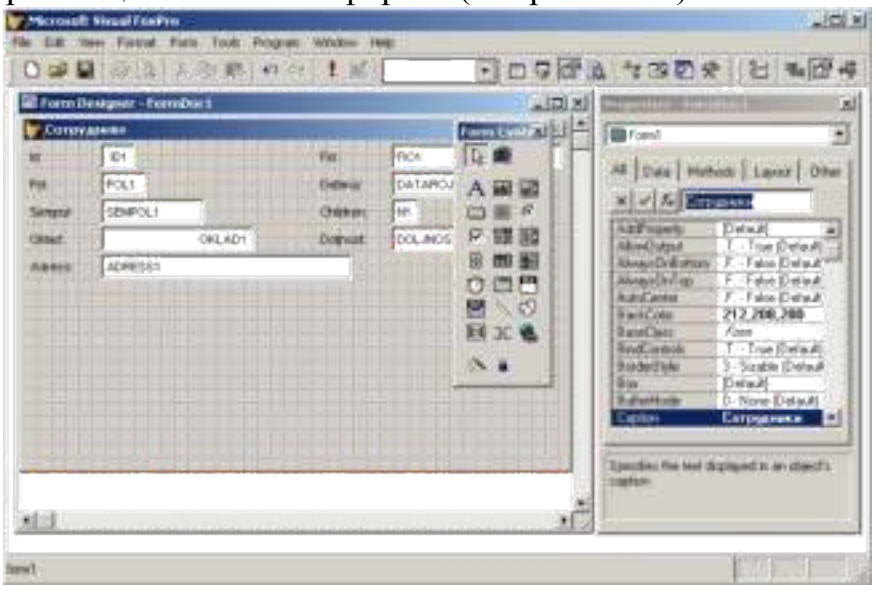

Рисунок 10.9. Окно FormDesigner(Конструктор форм)

Рисунок 10.10. Объект - блок кнопок управления

Для размещения на создаваемой форме кнопок управления воспользуемся ранее созданной формой Сотрудники. Открыть форму Сотрудники в режиме редактирования: окно программы ProjectManager/Documents/Forms/сотрудники/Modify. В появившемся окне FormDesigner формы Сотрудники нажатием левой клавиши мыши выделить объект BUTTONSET1 (блок кнопок управления, см. рис.10.10.) и нажать комбинацию клавиш Ctrl+Сна клавиатуре или установить курсор в форму, вызвать контекстное меню, выбрать команду Copy. Закрыть окно FormDesigner формы Сотрудники.

В окне FormDesigner формы Сотрудники тест установить курсор в форму, вызвать контекстное меню, выбрать команду Paste или нажать комбинацию клавиш Ctrl+V. Блок кнопок управления записями таблицы (см. рис.10.10) в создаваемой форме будет скопирован.

Задать цвет фона формы. Выделить в окнеProperties (Свойства) свойство формы BackColor (цвет фона), нажать расположенную с правой стороны поля редактирования свойства кнопку и в открывшемся диалоговом окне Цвет выберите цвет, который вы хотите использовать для фона

Расположить объекты формы в соответствии с рис.10.11. Для перемещения объектов формы можно использовать метод перетаскивания мышью или кнопками управления курсором на клавиатуре, предварительно выделив необходимый объект, а также комбинацией клавиш Ctrl+[кнопки управления курсором], использовать все выше перечисленные способы, выяснить их различия. Для изменения размеров формы или объектов формы необходимо выделить объект (форму), подвести курсор к углу объекта (формы), когда курсор примет вид разнонаправленной стрелки нажать левую клавишу мыши и удерживая еѐ, изменить размеры объекта (формы)

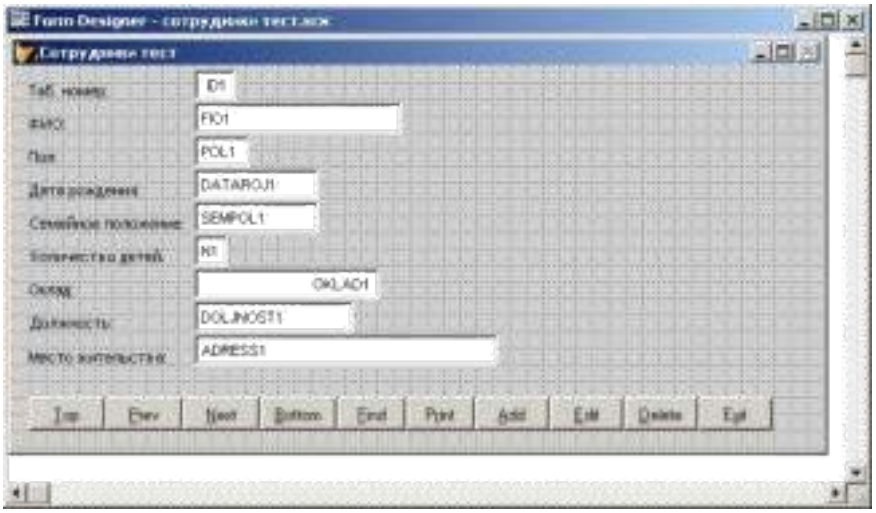

Рисунок 10.11 Вид отредактированной формы

Запустить отредактированную форму: установить курсор в форму, вызвать контекстное меню, выбрать команду RunForm или нажать кнопку [ ] на панели инструментов. Система попросить сохранить созданную форму,

нажать кнопку Yes, в появившемся окне указать папку и имя файла (Сотрудники тест), сохранить.

Индивидуальные задания к практической работе

Создать кнопки навигации в форме Книги

Открыть форму Книги в окне FormDesigner (Конструктор форм)

Нажать кнопку CommandGroup (Группа кнопок)  на панели инструментов FormControls (элементы управления формы) и щѐлкнуть в месте их предполагаемого размещения в форме

В Окне Properties (свойства) размещѐнного объекта выделить свойство ButtonCount (Количество кнопок) и указать необходимое количество размещаемых в форме кнопок, указав число 5

Увеличить с помощью мыши размеры рамки, окружающей данный объект. Перейти в режим редактирования: установить на объект курсор, вызвать контекстное меню и выбрать команду Edit (Редактировать). Выделяя поочерѐдно кнопки, переместить их, расположив горизонтально, в одну линию

Выйти из режима редактирования, щѐлкнув вне области объекта CommandGroup (Группа кнопок)

Скорректировать размер рамки, окружающей объект: в свойстве AutoSize (Авторазмер) установить значение True (Истина)

Открыть окно свойств объекта CommandGroup (Группа кнопок), нажать кнопку раскрытия списка в верхней части данного окна, поочерѐдно выбрать из списка элементы Command1, Command2, Command3, Command4, Command5 и, используя свойство Caption(Надпись) задать названия кнопок соответственно: Первая, Следующая, Предыдущая, Последняя, Выход (см. рис.10.12.)

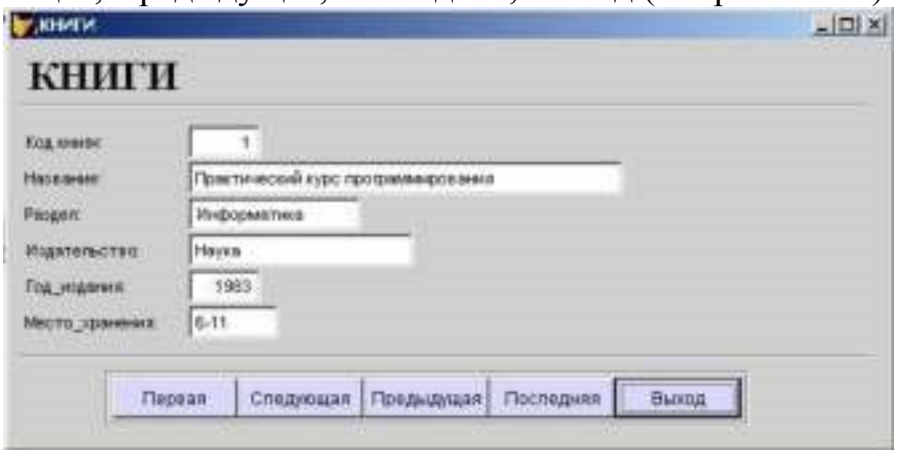

Рисунок 10.12. Форма Книги

Определить команды, которые будут выполняться при нажатии на созданные кнопки. Установить на объект CommandGroup (Группа кнопок) курсор, вызвать контекстное меню и выбрать команду Edit (Редактировать), выделить кнопку Первая, вызвать контекстное меню выбрать команду Code (Код программы), в появившемся окне ввести следующие команды:

Для кнопки Первая:
Переходим на первую запись и обновляем информацию в форме
IF !BOF()
GO TOP
ENDIF
_screen.ActiveForm.Refresh()
После ввода кода программы закрыть окно.
Ввести аналогично команды для остальных кнопок:
Для кнопки Следующая:
Переходим на следующую запись и обновляем информацию в форме
IF !EOF()
SKIP
ENDIF
_screen.ActiveForm.Refres
h() Для кнопки
Предыдущая:
Переходим на предыдущую запись и обновляем информацию в форме
IF !BOF()
SKIP - 1
ENDIF
_screen.ActiveForm.Refres
h() Для кнопки
Последняя:
Переходим на последнюю запись и обновляем информацию в форме
IF !BOF()
GO BOTTOM
ENDIF
_screen.ActiveForm.Refres
h() Для кнопки Выход:
Запрашиваем и выходим, если Да
IFMESSAGEBOX("Выходить из формы?", 4+32+256, "Выход")=6
_screen.ActiveForm.Release()
ELSE
_screen.ActiveForm.Refresh()
ENDIF
После ввода команд закрыть окно процедур
Запустить форму на выполнение. Проверить результат проделанной работы

Практическая работа №11 «Добавление записей в табличный файл из двумерного массива. Работа с командами ввода-вывода. Использование функций для работы с массивами»

Цель работы: овладение навыками изменения табличных файлов при помощи запросов и функций

Обработка запросов в VisualFoxPro

Одним из основных назначений разработанного приложения является быстрый поиск информации в БД и получение ответов на разнообразные вопросы. Для этих целей в VisualFoxPro используются средства, называемые запросами.

При создании запросов в VisualFoxPro разработчик может использовать следующие средства:

- QueryWizard (Мастер запросов) □ QueryDesigner (Конструктор запросов)
- Команда SELECT языкаVisual FoxPro

Результатом запроса является таблица, которую можно сохранить в массиве, создаваемой новой таблице, отобразить на экране в режиме Browse (Просмотр) или вывести в виде отчѐта.

Для создания запросов можно использоватьQueryWizard (Мастер запросов), который последовательно запрашивает наименования таблиц, используемых в запросе, перечень полей таблиц, критерий упорядочения и условия фильтрации данных. С помощью QueryDesigner (Конструктор запросов) можно формировать различной сложности критерии для выбора записей из одной или нескольких таблиц, над полями, выбираемыми из таблиц с помощью запросов, можно выполнять различные вычисления.

В верхней части окна QueryDesigner (Конструктор запросов) расположена панель, на которой отображаются используемые в запросе таблицы. Ниже находятся вкладки, предназначенные для выбора полей запроса и формирования условий выборки. Назначение этих вкладок приведено в таблице 1.. Открывая в окне QueryDesigner (Конструктор запросов) необходимые вкладки, можно выполнять следующие действия:

- Выбирать поля результирующей таблицы запроса
- Формировать вычисляемые поля
- Задавать критерии для выборки, группировки и упорядочения данных
- Указывать, куда выводить результаты выборки

Таблица 1. Назначение вкладок окна QueryDesigner (Конструктор запросов)

Вкладка	Назначение
Fields (Поля)	Позволяет указать поля исходных таблиц, выбираемые в результирующий запрос
Join(Объединение)	Позволяет задать условия объединения таблиц
Filter (Фильтр)	Позволяет определить фильтры, накладываемые для выбора записей
OrderBy (Упорядочение)	Позволяет задать критерии упорядочения данных
GroupBy (Группировка)	Позволяет задать условия группировки данных
Miscellaneous (Разное)	Позволяет задать дополнительные условия, такие как признак выборки повторяющихся значений,
	количество или процент выбора данных

Таблица 2. Назначение команд меню Query и кнопок панели инструментов Query

DesignerокнаQuery Designer (Конструкторзапросов)

Команда меню	Кнопка	Назначение
AddTable (Добавить таблицу)		Добавляет в запрос новую таблицу
RemoveTable (Удалить таблицу)		Удаляет выбранную таблицу из запроса
RemoveJoinCondition (Удалить условие объединения)		Удаляет условие объединения таблиц
OutputFields (Результирующие поля)		Открывает вкладку **Fields** для выбора полей результирующей таблицы
Join(Объединение)		Открывает вкладку**Join**для создания условия объединения таблиц
Filter (Фильтр)		Открывает вкладку**Filter** для задания фильтра
OrderBy (Упорядочение)		Открывает вкладку**OrderBy** для определения критерия упорядочения данных
GroupBy (Группировка)		Открывает вкладку**GroupBy** для определения условия группировки данных
Miscellaneous (Разное)		Открывает вкладку**Miscellaneous** для задания дополнительных параметров запроса
Query destination(Результат запроса)		Открывает диалоговое окно **Querydestination**, в котором указывается куда выводить результат запроса
ViewSQL (Показать SQL)		Открывает диалоговое окно, в котором отображается SQL-оператор, соответствующий данному запросу
Maximizethetableview (Максимизировать панель отображения)		Раскрывает панель отображения используемых в запросе таблиц на весь экран. Повторное нажатие возвращает панели первоначальный размер
AddJoin (Добавить условие объединения)		Открывает диалоговое окно**JoinCondition**длязадание условия объединения таблиц
Comments (Комментарии)		Открывает диалоговое окно, в котором вы можете ввести краткое описание создаваемого запроса
RunQuery (Выполнить запрос)		Запускает запрос на выполнение

Задание для практической работы

Запустить программу MicrosoftVisualFoxPro Открыть проект Информационная система
Открыть БД Штат проекта

Создать запрос о сотрудниках, имеющих более одного ребѐнка и получающих зарплату менее 9000: в окне ProjectManager щѐлкнуть клавишей мыши на вкладке
Data/Queries/New/QueryWizard, в появившемся диалоговом окне WizardSelection (Выбор мастера) выбрать пункт QueryWizard (Мастер запросов) и нажать кнопку OK

В появившемся диалоговом окне выбора исходной таблицы и полей в области Databasesandtables (Базы данных и таблицы) выбрать необходимую БДштат иуказать таблицу Сотрудники. Из области Availablefields (Имеющиеся поля) выбрать поля Fio и Oklad. Для перехода к следующему шагу нажать кнопку Next(Далее)

В появившемся диалоговом окне выбора условий выборки в области Field(Поле) верхней строки выбрать поле Children (Количество детей) таблицы Сотрудники, в области Operator (Оператор Условия) выбрать условие morethen, в области Value (Выражение) установить значение 1, в нижней строке внести следующие данные соответственно: в области Field(Поле) – поле Oklad (Зарплата), в области Operator (Оператор Условия) – lessthan, в области Value (Выражение) – 9000. Нажать кнопку Next(Далее)

В появившемся диалоговом окне выбора условия сортировки данных из области Availablefields (Имеющиеся поля) выбрать поле Fio. Нажать кнопку Next(Далее)

Нажать кнопку Preview (Просмотр) и просмотреть результат запроса (запрос будет выглядеть в виде простой таблицы см. рис.11.1.), закрыть таблицу запроса. Нажать кнопку Finish (Готово), в появившемся диалоговом окне SaveAs (Сохранить как) указать необходимую папку и присвоить запросу имя Зарплата дети

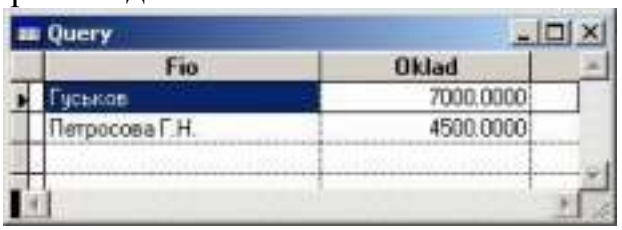

Рисунок 11.1. Запрос Зарплата дети

Запустить запрос: в окне ProjectManager щѐлкнуть клавишей мыши на вкладке Data/Queries/Зарплата дети/Run 10. Закрыть запрос Зарплата дети

Создать новый запрос, показывающий количество сотрудников организации, получающих определѐнную зарплату: в окне ProjectManager щѐлкнуть клавишей мыши на вкладке Data/Queries/New/NewQuery. В появившемся диалоговом окне AddTableorView (Добавить таблицу или представление данных) в области Database (База данных) выбрать БД Штат, в

области TablesinDatabaseтаблицу Сотрудники. Нажать кнопку Add, при этом в верхней области окна
QueryDesigner (конструктор запросов) (см. рис.11.2.), находящегося позади окна AddTableorView (Добавить таблицу или представление данных) появится выбранная таблица. Нажать кнопку Close (Закрыть) для закрытия окна AddTableorView.

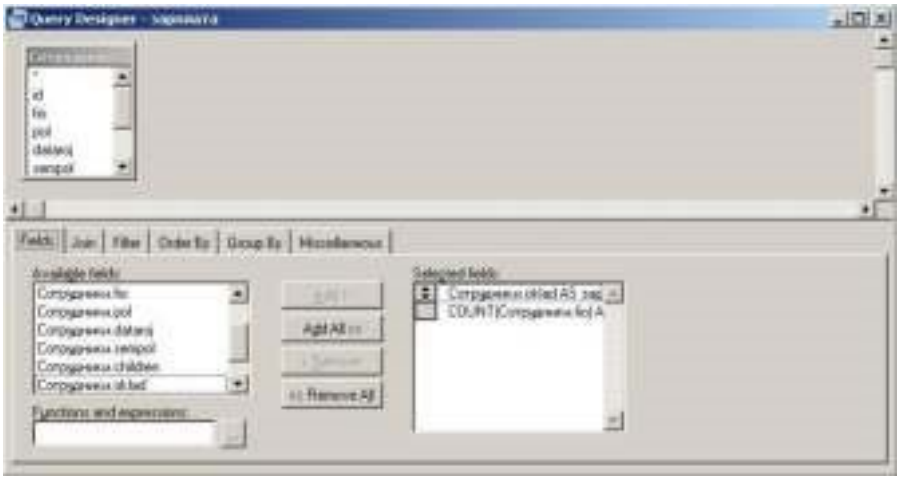

Рисунок 11.2. Окно QueryDesigner (конструктор запросов)

На вкладке Fields (Поля) окна QueryDesigner (конструктор запросов) в области Availablefields (Имеющиеся поля) выделить поле Сотрудники.oklad, оно автоматически перенесётся в область Functionandexpressions (Функции и выражения), нажать расположенную справа от поля кнопку вызова построителя выражения, откроется диалоговое окно
ExpressionBuilder (Построитель выражения) (см. рис.11.3.)

Рисунок 11.3. Диалоговое окно ExpressionBuilder (Построитель выражения)

В поле ввода Expression (Выражение) диалогового окна ExpressionBuilder (Построитель выражения) используя поля таблиц, расположенные в списке Fields (Поля), функции области Functions (Функции),

и ввод данных с клавиатуры сформировать следующее выражение:Сотрудники.oklad AS Зарплата

Для проверки правильности набранного выражения нажать кнопку Verify (Проверить), нажать кнопку OK

Для переноса созданного выражения из области Functionandexpressions (Функции и выражения) в область SelectedFields (Выбранные поля) нажать находящуюся между данными областями кнопку Add (Добавить)

Руководствуясь предыдущими пунктами 12-15 практической части создать следующее выражение:

COUNT(Сотрудники.fio) AS Количество_сотрудников

Выбрать вкладку OrderBy (Упорядочение) окна QueryDesigner (конструктор запросов) и из областиSelectedFields (Выбранные поля) в область Orderingcriteria (Критерий упорядочения) перенести поле Сотрудники.oklad

Выбрать вкладкуGroupBy (Группировка) окна QueryDesigner (конструктор запросов) и из области Availablefields (Имеющиеся поля) в область GroupedFields (Поля группировки) перенести поле Сотрудники.oklad

Выбрать пункт меню Query (Запрос) окна программы VisualFoxPro, выбрать команду Comments (Комментарии) (см. таблицу 2.). В появившемся окне Comment(Комментарий) в поле ввода области AddComment (Добавить комментарий) ввести строку: Запрос, показывающий количество сотрудников организации, получающих определѐнную зарплату. Закрыть окно Comment(Комментарий)

Запустить созданный запрос: установить курсор в окне QueryDesigner (конструктор запросов), вызвать контекстное меню, выбрать команду RunQuery, или нажать кнопку [ ] на панели инструментов. Закрыть запрос

Рисунок 11.4. Запрос Зарплата

Просмотреть SQL-оператор, соответствующий запросу: установить курсор в окне QueryDesigner (конструктор запросов), вызвать контекстное меню, выбрать команду ViewSQL (Показать SQL), откроется диалоговое окно, в котором отображается SQL-оператор, соответствующий данному запросу. Закрыть окно QueryDesigner (конструктор запросов), система предложит сохранить запрос, нажать кнопку Yes (Да), в появившемся диалоговом окне SaveAs

(Сохранить как) указать необходимую папку и присвоить запросу имя Зарплата

Индивидуальные задания к практической работе

Создать многотабличный запрос, выводящий в таблицу названия, год издания и автора книг по информатике

Создать новый запрос, добавить таблицы Авторы, Автор_книги и Книги из БД Библиотека в окно QueryDesigner (конструктор запросов)

Из области Availablefields (Имеющиеся поля) вкладки Fields (Поля) внести в область SelectedFields (Выбранные поля) используя построитель выражения ExpressionBuilder следующие выражения:

Книги.nazvanie AS название_книги

Книги.razdel AS раздел

Книги.godizdan AS год_издания

ALLTRIM(Авторы.familiya)+" "+ALLTRIM(Авторы.imya) AS автор

Необходимо задать условие выбора записей из таблицы (Книги только по информатике) по полю razdel (Раздел). Перейти на вкладку Filter (Фильтр), в области FieldName (Имя поля) нажать кнопку раскрытия списка и выбрать поле Книги.razdel, в области Criteria (Критерии) выбрать значение ==, в области Example (Образец) ввести значение — Информатика‖

Запустить запрос и посмотреть результат (см. рис.11.5.), закрыть запрос, закрыть окноQueryDesigner (конструктор запросов), сохранить запрос, присвоив ему имя Книги по информатике

Название_книги	Раздел	Год_издания	Автор
Практический курс программирования	Информатика	1983	Фролов Геннадий
Практический курс программирования	Информатика	1983	Илюхин Виктор
TURBO PASCAL для школьников	Информатика	1999	Попов Владимир
HTML в действии	Информатика	1997	Морис Брюс
Самоучитель Visual FoxPro 8.0	Информатика	2003	Омельченко Людмила

Рисунок 11.5. Запрос Книги по информатике

Практическая работа №12 «Создание меню различных видов. Модификация и управление меню»

Цель работы: овладение навыками создания и модификации структуры меню

Создание меню в VisualFoxPro

В соответствии со стандартами Windows в любом приложении рекомендуется иметь строку меню, которая в VisualFoxPro содержит команды, предназначенные для вызова форм, формирования отчѐтов, запросов и т.д.

Строкой меню называется горизонтальное меню, располагаемое в верхней части экрана. Примером строки меню является основное меню VisualFoxPro, а также меню программ, работающих в среде Windows. Созданное в конструкторе MenuDesigner (Конструктор меню) меню может замещать основное меню VisualFoxPro или добавляться к нему.

Для создания меню необходимо выполнить следующие действия:

- Открыть окно конструктора меню MenuDesigner (Конструктор меню)
- Описать вид меню, текст, пункты меню и его атрибуты
- Сгенерировать меню. При этом создаѐтся программа, которая в результате запускается на выполнение

VisualFoxPro имеется возможность создания меню в виде строки Menu, или всплывающего меню Shortcut, в котором основные пункты расположены по вертикали.

Таблица 12.1.Назначение кнопок конструктора меню MenuDesigner (Конструктор меню)

Кнопка	Назначение
Insert (Вставить)	Добавляет в меню новый пункт
InsertBar (Вставить команды системного меню)	Открывает диалоговое окно **InsertSystemMenuBar**, содержащее команды системного меню **VisualFoxPro**, позволяя разместить их в создаваемом пользовательском меню
Delete (Удалить)	Удаляет текущий пункт меню
MoveItem (Переместить элемент)	Открывает одноимѐнное диалоговое окно, позволяющее указать пункт меню, в который переносится текущий подпункт
Preview (Просмотр)	Размещает создаваемое меню на экране, позволяя просмотреть его внешний вид

Таблица 12.2. Типы меню

Тип меню	Назначение
Command(Команда)	При выборе пункта меню данного типа будет выполняться связанная с ним команда
PadName (Наименование строки меню)	При выборе пункта меню никаких действий выполняться не будет. Как правило, используется в качестве дополнительного пояснения к меню
Submenu(Подменю)	При выборе пункта меню раскрывается связанное с данным пунктом ниспадающее меню
Procedure (Процедура)	При выборе пункта меню вызывается процедура, определѐнная для данного пункта меню

Таблица 12.3. Типы пункта меню

Тип пункта меню	Действие
Submenu(Подменю)	Раскрывается связанное с данным пунктом меню ниспадающее меню
Procedure (Процедура)	Выполняется процедура, определенная в конструкторе меню
Command(Команда)	Выполняется команда, расположенная в поле рядом с типом пункта меню

Задание для практической работы

Запустить программу MicrosoftVisualFoxPro Открыть проект Информационная система

Открыть окно MenuDesigner (Конструктор меню): в окне ProjectManager щёлкнуть клавишей мыши на вкладке Other/Menus/New, в появившемся диалоговом окне NewMenu (Новое меню) выбрать пункт Menu(Меню), появится окно MenuDesigner (Конструктор меню) (см. рис.12.1.)

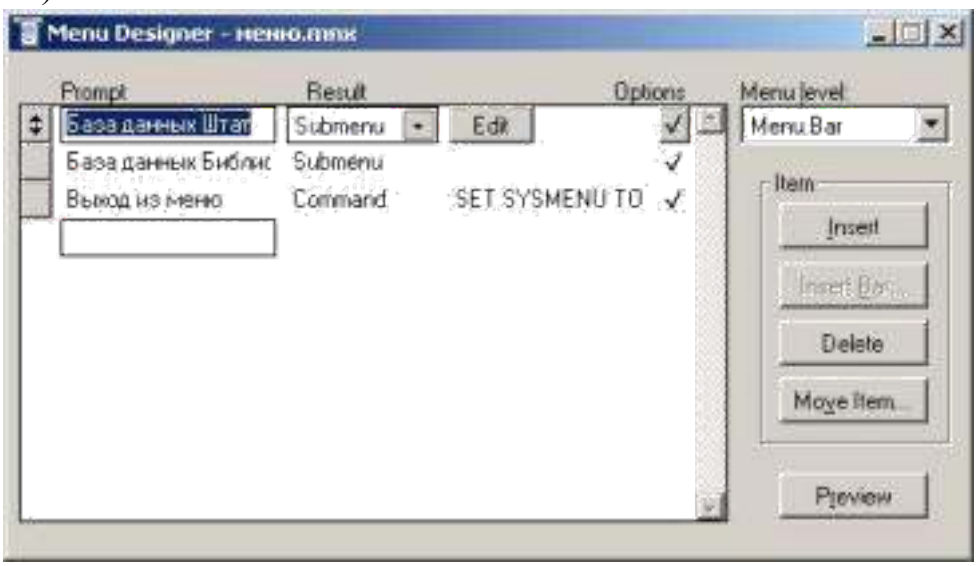

Рисунок 12.1. Окно MenuDesigner (Конструктор меню)

Область конструктора меню, над которой расположены надписи Prompt(Приглашение), Result (Результат) и Options(Опции), предназначена для формирования меню. В поле Prompt(Приглашение) вводят наименования пунктов меню, раскрывающийся список Result (Результат) используется для указания типа пункта меню, а кнопка Options(Опции) открывает диалоговое окно PromptOptions (Опции элемента меню), в котором можно определить дополнительные параметры данного элемента меню. В списке Menulevel (Уровень меню) указывается уровень текущего меню. Слева в конструкторе меню размещены кнопки (см. таблицу 12.1.)

Рисунок 12.2. Схема создаваемого меню Меню

В поле Prompt(Приглашение) ввести наименование первого пункта меню – База данных Штат, нажать для перехода на следующее поле клавишу Enter или Tab на клавиатуре, курсор окажется в списке Result (Результат) (см. рис.1.) Указать тип пункта меню Submenu(Подменю), нажать кнопку Options(Опции), откроется диалоговое окно PromptOptions (Опции элемента меню), в котором, в области Shortcut(Всплывающее меню) установить курсор в поле KeyLabel (Метка) и нажать комбинацию клавиш Alt + F1 на клавиатуре. Мы определили комбинацию клавиш быстрого вызова пункта меню База данных Штат. Нажать кнопку OK

Перейти на следующую строку и ввести наименование второго пункта – База данных

Библиотека, тип пункта меню Submenu(Подменю), определить комбинацию клавиш Alt + F2 быстрого вызова пункта меню База данных Библиотека (см. пункт 4. лабораторной работы)

Перейти на следующую строку и ввести наименование третьего пункта – Выход из меню, в списке Result (Результат) указать тип пункта меню Command(Команда), в появившейся справа строке ввода ввести команду – SET SYSMENU TO DEFAULT. Определить комбинацию клавиш Alt + F4 быстрого вызова действия пункта меню Выход из меню

Пункты меню База данных Библиотека и База данных Штат имеют свои подменю. Для создания подменю перейти на строку База данных Библиотека, нажать кнопку Create (Создать), находящуюся справа от списка Result (Результат), на экране появится пустое окно конструктора меню. Ввести в поле Prompt(Приглашение) наименование подпункта – Таблицы базы данных, в списке Result (Результат) указать тип пункта меню Submenu(Подменю), нажать кнопку Create (Создать), так как он тоже имеет подменю (см.рис.2.)

В появившемся пустом окне конструктора меню ввести в поле Prompt(Приглашение) наименование подпункта – Автор книги, в списке Result (Результат) указать тип пункта меню Procedure (Процедура), нажать кнопку Create (Создать) и в открывшемся окне редактирования процедуры ввести следующие команды:

```
OPEN DATABASE \<Имя вашей папки>\библиотека.dbc
USE библиотека!автор_книги
SELECTавтор_книги
BROWSE
CLOSE
DATABASES
```

Закрыть окно редактирования процедуры, перейти на следующую строку и ввести наименование следующего пункта – Авторы, в списке Result (Результат) указать тип пункта меню Procedure (Процедура), нажать кнопку Create (Создать) и в открывшемся окне редактирования процедуры ввести следующие команды:

```
OPEN DATABASE \<Имя вашей папки>\библиотека.dbc
USE    библиотека!авторы
SELECT авторы
BROWSE
CLOSE
DATABASES
```

Закрыть окно редактирования процедуры, перейти на следующую строку и ввести наименование следующего пункта – Издательство, в списке Result (Результат) указать тип пункта меню Procedure (Процедура), нажать кнопку Create (Создать) и в открывшемся окне редактирования процедуры ввести следующие команды:

```
OPEN DATABASE \<Имя вашей папки>\библиотека.dbc
USE библиотека!издательство
SELECT издательство
BROWSE
CLOSE
DATABASES
```

Закрыть окно редактирования процедуры, перейти на следующую строку и ввести наименование следующего пункта – Книги, в списке Result (Результат) указать тип пункта меню Procedure (Процедура), нажать кнопку Create (Создать) и в открывшемся окне редактирования процедуры ввести следующие команды:

```
OPEN DATABASE \<Имя вашей папки>\библиотека.dbc
USE библиотека!книги
SELECT книги
```

BROWSE
CLOSE
DATABASES

Закрыть окно редактирования процедуры, перейти на следующую строку и ввести наименование следующего пункта – Разделы, в списке Result (Результат) указать тип пункта меню Procedure (Процедура), нажать кнопку Create (Создать) и в открывшемся окне редактирования процедуры ввести следующие команды:

OPEN DATABASE \<Имя вашей папки>\библиотека.dbc
USE библиотека!разделы
SELECT разделы
BROWSE
CLOSE
DATABASES

Закрыть окно редактирования процедуры. Для перехода в меню верхнего уровня использовать список MenuLevel (Уровень меню): выбрать из выпадающего списка MenuLevel (Уровень меню) пункт – Базаданных

Под строкой Таблицы базы данных ввести наименование следующего пункта – Форма Книги, в списке Result (Результат) указать тип пункта меню Command(Команда), в появившейся справа строке ввода ввести команду – DO FORM \< Имя вашей папки>\книги.scx

Перейти на следующую строку и ввести наименование следующего пункта – Отчѐт Книги, в списке Result (Результат) указать тип пункта меню Command(Команда), в появившейся справа строке ввода ввести команду – REPORT FORM \< Имя вашей папки>\книги.frx PREVIEW

Перейти на следующую строку и ввести наименование следующего пункта – Запрос, выводящий на экран названия, год издания и авторов книг по информатике, в списке Result
(Результат) указать тип пункта менюProcedure (Процедура), нажать кнопку Create (Создать) и в открывшемся окне редактирования процедуры ввести следующие команды:

OPEN DATABASE \<Имя вашей папки>\библиотека.dbc
DO "\<Имя вашей папки>\книги по информатике.qpr"
CLOSE DATABASES

Закрыть окно редактирования процедуры. Для просмотра создаваемых пунктов меню, нажать кнопку Preview (Просмотр), при этом основное меню VisualFoxPro будет заменено создаваемым меню. После просмотра нажать кнопку OKдиалового окна Preview (Просмотр)

Для улучшения внешнего вида, а также для объединения в группы схожих по смыслу команд, в меню можно использовать разделительные линии. Разделительные линии представляют собой пункт меню, в котором в поле ввода Prompt(Приглашение) вместо наименования пункта вводятся символы \-. Также VisualFoxPro позволяет справа от команд

пользовательскогоменю расположить графическое изображение, для этого необходимо выполнить следующие действия: нажать кнопку Options(Опции) для необходимого пункта меню, откроется диалоговое окно PromptOptions (Опции элемента меню), в котором, в области Picture(Изображение) указать опциюResource (Ресурс), нажать кнопку, располагаемую справа от поля, находящегося под опцией. Откроется диалоговое окно InsertSystemMenuBar (Вставить из системного меню). Из списка графических изображений, используемых VisualFoxPro в системном меню выбрать наиболее соответствующее пункту меню значение и нажать кнопку OK. Выбранное значение переносится в область просмотра области Picture(Изображение). Нажать кнопку OK для закрытия окна PromptOptions (Опции элемента меню)

Для перехода в меню верхнего уровня выбрать из выпадающего списка MenuLevel

(Уровень меню) пункт – MenuBar

Указать месторасположение создаваемого меню: пункт меню VisualFoxProView/GeneralOptions, откроется диалоговое окно GeneralOptions (Основные параметры), в области Location (Размещение) выбрать пункт Append (Добавить), для добавления создаваемого меню в основное меню VisualFoxPro

Индивидуальные задания к практической работе

Добавить в пункт меню – База данных Штат подпункты, определить действия для этих подпунктов меню в соответствии с таблицей 12.4.

Таблица 12.4. Наименование пунктов меню и определение их действий

Prompt(Приглашение)	**Result** (Результат)	Команды и процедуры

Таблица Сотрудники	Procedure	OPENDATABASE \<*Имя вашей папки*>\штат.dbc USEштат!сотрудники SELECT Сотрудники BROWSELAST CLOSE DATABASES
Форма Сотрудники	Command	DOFORM \<*Имя вашей папки*>\сотрудники.scx
Форма Сотрудники тест	Command	DOFORM \<*Имя вашей папки*>\тест.scx
Отчёт Сотрудники	Command	REPORTFORM \<*Имя вашей папки*>\сотрудники.frxPREVIEW
Запрос, показывающий количество сотрудников организации, получающих определённую зарплату	Procedure	OPEN DATABASE \<*Имявашейпапки*>\штат.dbc DO "\<*Имя вашей папки*>\зарплата.qpr" CLOSE DATABASES
Запрос, показывающий сотрудников, имеющих более одного ребёнка и получающих зарплату менее 9000	Procedure	OPEN DATABASE \<*Имявашейпапки*>\штат.dbc DO "\<*Имя вашей папки*>\зарплата дети.qpr" CLOSE DATABASES

Сохранить созданное меню: пункт меню File/SaveAs, в появившемся диалоговом окне указать имя своей папки, куда следует сохранить файл, присвоить имя Меню и нажать кнопку Save (Сохранить)

Необходимо сгенерировать меню: пункт меню Menu/Generate, откроется диалоговое окно GenerateMenu (Генерация меню), нажать кнопку Generate (Генерация)

После завершения генерации запустить программу меню на выполнение: в окне проекта ProjectManagerустановить курсор на наименование созданного меню Меню и нажать кнопку Run (Запустить)

Практическая работа №13 «Создание рабочих и системных окон. Добавление элементов управления рабочим окном»

Цель работы: овладение практическими навыками работы с системными окнами, создание отчетов по представленным данным

Отчеты в VisualFoxPro

Отчет – форматированное представление данных, выводимое на экран, принтер или в файл. Отчет, создаваемый в VisualFoxPro, может быть представлен в табличном виде или в свободной форме. Табличные отчеты используются для печати данных, представленных в виде списка. При создании отчета в VisualFoxProразработчикможет использовать следующие средства:

- ReportWizard – Мастер отчета. Позволяет быстро создать отчѐт, применяя сортировку, группировку данных и заданный разработчиком стиль оформления;
- ReportDesigner – Конструктор отчета. Позволяет разрабатывать собственные отчѐты или модифицировать отчѐты, созданные с помощью мастера;
- QuickReport – Быстрый отчет. Данное средство предназначено для размещения в конструкторе отчѐта полей и задания среды окружения.

Задание для практической работы

Запустить программу MicrosoftVisualFoxPro Открыть проект Информационная система

Открыть БД проекта. Для этого необходимо на вкладке Data (Данные) установить курсор на еѐ название и нажать кнопку Open (Открыть) окна проекта. При этом на стандартной панели инструментов в списке Databases (Базы данных) появится название открытой БД Перейти на вкладку Documents/Reports/New/New Report /Report Wizard

В появившемся диалоговом окне WizardSelection указать тип создаваемого отчета ReportWizard(Мастер отчетов) и нажать кнопку OK

Появляется первое диалоговое окно мастера отчѐта (см. рис.13.1.). Выбрать из верхнего списка области Databasesandtables (Базы данных и таблицы) базу данных штат, а из нижнего – таблицу сотрудники

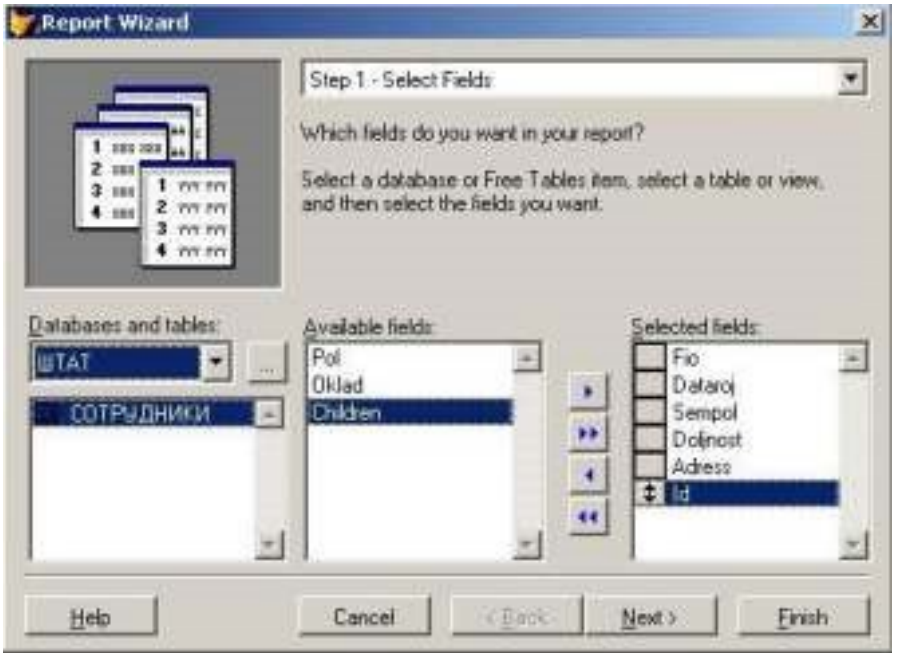

Рисунок 13.1. Диалоговое окно создания однотабличного отчета с помощью мастера. Перенести из списка Availablefields (Имеющиеся поля) в список Selectedfields (Выбранные поля) поля, которые вы хотите разместить в создаваемом отчете (в соответствии с рис.13.1.). После формирования списка отображаемых в отчете полей нажать кнопку Next (Далее) для перехода к следующему шагу в создании отчета

В следующем окне мастера создания отчета указать поля, по которым будут

осуществляться группировка данных в отчете (рис.13.2.)

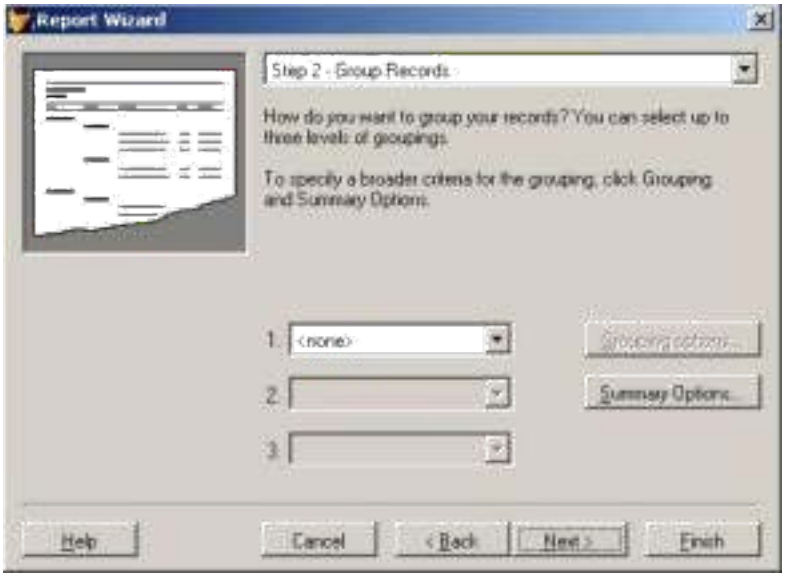

Рисунок 13.2. Диалоговое окно определения полей для группировки данных в отчете.

В центре диалогового окна расположены три раскрывающихся списка, позволяющих задать до трех группировок данных в отчете. Эти списки содержат все поля таблицы. Для осуществления группировки данных в отчете

выбрать нужное поле из раскрывающегося списка 1. При создании второй и третьей группировки используются, соответственно, списки 2 и 3. В нашем примере группировка данных по полям не осуществляется, это диалоговое окно мастера отчѐтов необходимо пропустить и нажать кнопку Next

В следующем диалоговом окне мастера задается стиль отображения объектов в отчете (рис.13.3.). Список Style (Стиль) содержит несколько вариантов отображения объектов (полей, линий, заголовок и т.д.) в отчете. С помощью области просмотра в верхнем левом углу диалогового окна можно просмотреть тот стиль, который мы выбрали. Выбрав стиль, нажать кнопку

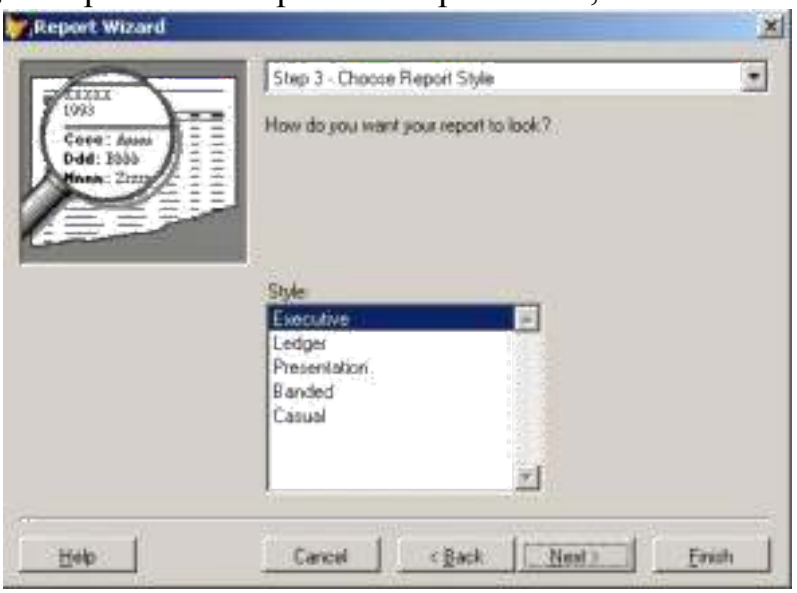

Рисунок 13.3. Диалоговое окно выбора стиля отображения объектов.

Указать порядок размещения объектов в отчете (рис.13.4.) и ориентацию страницы отчета. В области NumberofColumns (Количество колонок) указать 1, в области Orientation (Ориентация) Указать Portrait (Книжная), в области FieldLayout (Расположение полей) указать Rows (Строка). После установки требуемых опций, нажать кнопку Next

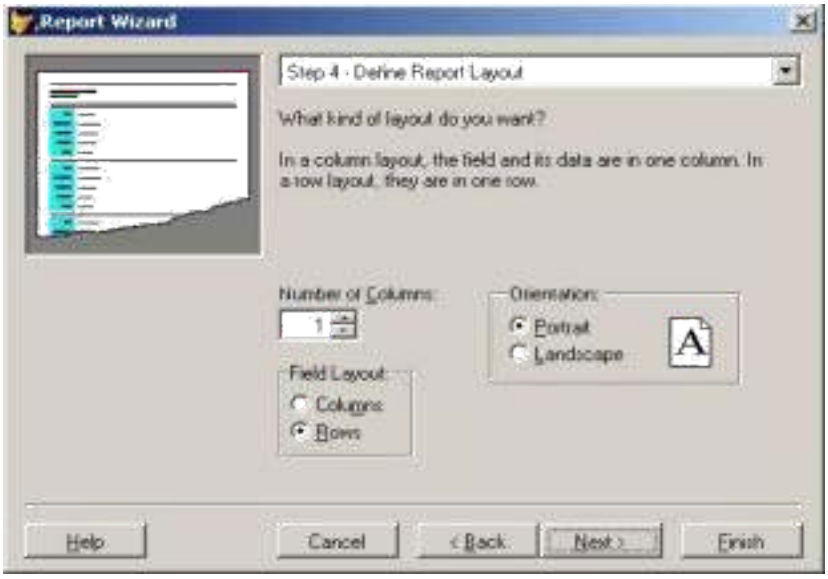

Рисунок 13.4. Диалоговое окно установки порядка размещения объектов.

Задать поле, по которому требуется упорядочение данных в отчете в соответствии с рис.13.5. Из списка Availablefieldsorindextag (Выбранные поля и индексы) перенести в список Selectedfields (Выбранные поля) поле Fio (упорядочивание данных по фамилии). Для переноса полей используйте кнопку Add (Добавить). Сформировав список полей, нажать кнопку Next (Далее);

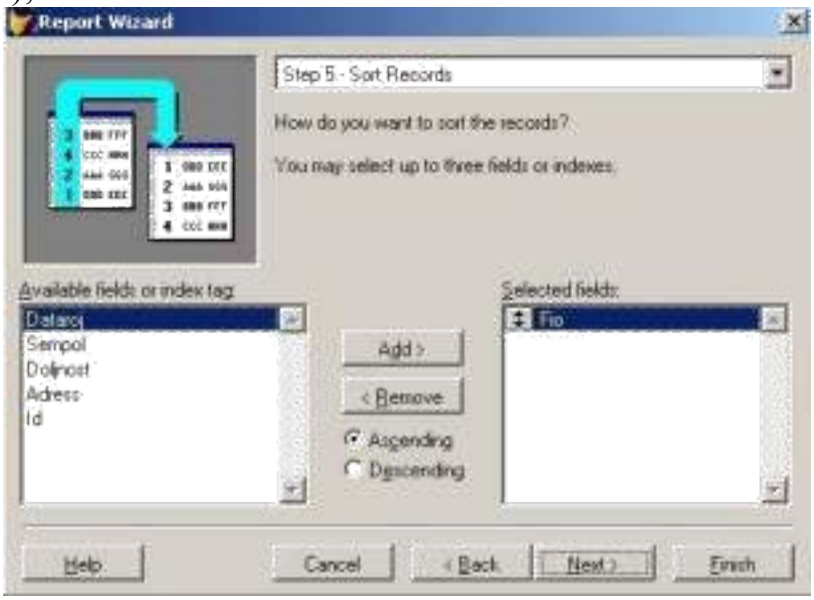

Рисунок 13.5. Диалоговое окно установки критерия упорядочения данных.

На заключительном шаге создания отчета в области Typeatitleforyourreport (Введите имя заголовка отчета) указать имя отчѐта Сотрудники. Воспользовавшись кнопкой Preview (Просмотр) просмотреть, как будет выглядеть создаваемый отчет, после просмотра нажать кнопку ClosePreviewна панелиPrintPreview(см. рис.13.6.),если что-то не так, вернитесь к предыдущим шагам, воспользовавшись кнопкой Back. Нажать кнопку Finish (Готово) и в появившемся диалоговом окне SaveAs (Сохранить как) указать папку, в которой будет размещен отчет Сотрудники.

Рисунок 13.6. Панель просмотра отчѐтаPrintPreview

Просмотреть созданный отчѐт: в окне ProjectManager выбрать вкладку Documents/Reports/Сотрудники/Preview или команду Preview (Просмотр) из меню программы View (Вид)

Открыть отчѐт Сотрудники в окне ReportDesigner (конструктор отчѐта): в окне ProjectManager выбрать вкладку Documents/Reports/Сотрудники/Modify

Таблица 13.1. Типы полос отчѐта

Title (Титул)	В этой полосе размещается информация, появляющаяся перед основным отчётом (имя отчёта, сопроводительное письмо и т. д.) и называется титульной
PageHeader (Верхний колонтитул)	Данные, помещённые в эту полосу, печатаются в начале каждой страницы (название отчёта, текущая дата и т. д.)
Detail (Детали)	Эта полоса содержит данные полей из таблицы или результат вычислений над ними
PageFooter (Нижний колонтитул)	В нижнем колонтитуле печатается название отчёта, дата, номер страницы, итоговые значения по данным текущей страницы

Выделить все объекты в полосе Title (Титул) и перенести их полосу PageHeader (Верхний колонтитул) используя метод перетаскивания мышью или кнопками управления курсором на клавиатуре, предварительно выделив необходимый объект, или комбинацией клавиш Ctrl+[кнопки управления курсором]

В полосе Title (Титул) напечатать следующий текст: — Отчѐт - форматированное представление данных, выводимое на экран, принтер или в файл. Табличный отчѐт используется для печати данных, представленных в виде списка‖

Поместить набранный текст в прямоугольник со скругленными краями: нажать на

панели инструментов ReportControls кнопку  поместить данный объект в отчѐт.

Отредактировать его расположение в соответствии с рис.13.7.

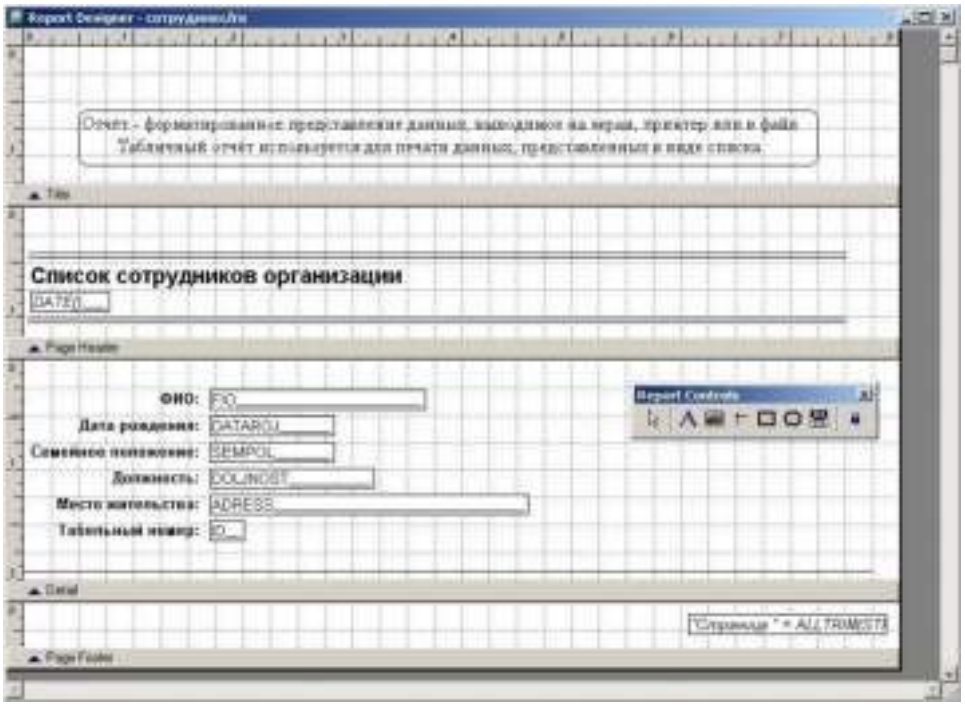

Рисунок 13.7. Окно ReportDesigner (Конструктор отчѐтов)

В процессе работы просматривайте результаты редактирования, используя пункт меню View/Preview

В полосе Detail (Детали), ниже поля Табельный номер провести линию, используя кнопку  панели инструментов ReportControls и отредактировать еѐ расположение в отчѐте в
соответствии с рис.7.

В полосе PageFooter (Нижний колонтитул) переместить элемент  в правый нижний угол в
соответствии с рис.137.

Вызвать свойства элемента  отчѐта двойным щелчком мыши или через контекстное меню команда Properties и в появившемся диалоговом окне ReportExpression (Выражение отчѐта) в поле ввода Expression заменить словоPage на слово Страница, нажать OK

Просмотреть результат выполненной работы: пункт меню View/Preview **Индивидуальные задания к практической работе**

Создать отчет для таблицы Книги БД Библиотека

В диалоговом окне ReportWizard(Мастер отчетов) (см. рис.1.) выбрать следующие поля таблицы Книги: Nazvanie, Razdel, Izdat, Godizdan

В диалоговом окне выбора стиля отображения объектов (см. рис.3.) в поле области Style
(Стиль) выбрать стиль Executive

В диалоговом окне установки порядка размещения объектов (см. рис.4.) в поле областиNumberofColumns (Количество колонок) указать 1, в области FieldLayout (Расположение полей) указать Rows (Строка)

В диалоговом окне установки критерия упорядочения данных (см. рис.5.) из списка
Availablefieldsorindextag (Выбранные поля и индексы) перенести в список Selectedfields
(Выбранные поля) поле Nazvanie

Просмотреть вид создаваемого отчета нажатием кнопки Preview (Просмотр).

После сохранения запустить отчет Книги, она должна иметь вид как на рис.13.8.

Рисунок 13.8. Отчѐт Книги

Практическая работа №14 «Создание файла проекта базы данных. Создание интерфейса входной формы. Использование исполняемого файла проекта БД, приемы создания и управления»

Цель работы: овладеть навыками проектирования предметной области, создания интерфейсов и применения исполняемого файла БД

Проектирование базы данных с использованием ER-технологии

Для заданной предметной области должен быть определен состав реляционных таблиц и логические связи между таблицами. Для каждого атрибута должны быть заданы тип и размер данных, ограничения целостности. Для каждой таблицы – первичный ключ, потенциальные ключи и внешние ключи.

Разработка логической модели методом «сущность-связь» (ER-методом) предусматривает выполнение следующих шагов, детально описанных в работе:

1. построение ER-диаграммы, включающей все сущности и связи, важные с точки зрения интересов предметной области;
2. анализ связей и определение их характеристик – степени связи, мощности и класса принадлежности;
3. построение набора предварительных отношений с указанием предполагаемого первичного ключа для каждого отношения;
4. подготовка списка всех неключевых атрибутов и назначение каждого из этих атрибутов одному из предварительных отношений;
5. проверка нахождения всех полученных отношений в нормальной форме Бойса-Кодда; 6. построение модели данных.

Создание и связывание таблиц базы данных в среде MySQL Рассмотрим следующие вопросы:

- создание и выбор базы данных;
- создание таблиц;
- столбцы и типы данных в MySQL;
- создание индексов;
- удаление таблиц, индексов и баз данных;
- изменение структуры таблиц.

Базы данных, таблицы и индексы легко создаются в рамках графического интерфейса MySQL, но мы будем использовать монитор MySQL (клиент командной строки), чтобы лучше понять структуру БД, таблиц и индексов.

Чувствительность к регистру и идентификаторы.

- Имена БД подчиняются тем же правилам зависимости от регистра символов, каким следуют каталоги операционной системы. Имена таблиц следуют тем же правилам, что и имена файлов. Все остальное не зависит от регистра.
- Все идентификаторы, кроме имен псевдонимов, могут содержать до 64 символов. Имена псевдонимов могут иметь до 255 символов.
- Идентификаторы могут содержать любые допустимые символы, но имена баз данных не могут содержать символы /, \ и . , а имена таблиц – символы . и /.
- Зарезервированные слова можно использовать для идентификаторов, если заключить их в кавычки.

Комментарий в SQL. Начинается с двух дефисов (--), за которыми должен следовать пробел. Кроме того, MySQL содержит ряд собственных комментариев. Shell-комментарий # действует аналогично – все, что расположено правее его, является текстом комментария. Скомментарий /* */ является многострочным – комментарий начинается с /* и заканчивается, когда встретится завершение */.

Создание и выбор базы данных. Осуществляется с помощью оператора

CREATE DATABASE имя_базы_данных;

Убедиться в том, что оператор выполнил задачу, можно с помощью оператора

SHOW DATABASES;

Теперь имеется пустая БД, ожидающая создания таблиц. Прежде чем работать с БД, необходимо выбрать эту БД с помощью оператора

USE имя_базы_данных;

Теперь все действия по умолчанию будут применяться именно к этой БД.

Создание таблиц. Используется оператор *CREATE TABLE*, который в общем виде выглядит следующим образом:

CREATE [TEMPORARY] TABLE [IF NOT EXISTS]
имя_таблицы (определение таблицы)
[TYPE=тип_таблицы];

Ключевое слово *TEMPORARY* используется для создания таблиц, которые будут существовать только в текущем сеансе работы с БД и будут автоматически удалены, когда сеанс завершится.

При использовании выражения *IF NOT EXISTS* таблица будет создана только в том случае, если еще нет таблицы с указанным именем.

Создать таблицу с такой же схемой, как у существующей, позволяет команда *CREATE [TEMPORARY] TABLE [IF NOT EXISTS] имя_таблицы LIKE имя_старой_таблицы;*

После имени таблицы в скобках объявляются имена столбцов, их типы и другая информация. В определение столбца можно добавить следующие описания.

- Объявить для любого столбца *NOT NULL* или *NULL* (столбцу запрещено или не запрещено содержать значения *NULL*). По умолчанию – *NULL*.
- Объявить для столбца значение по умолчанию, используя ключевое слово *DEFAULT*, за которым должно следовать значение по умолчанию.
- Использовать ключевое слово *AUTO_INCREMENT*, чтобы генерировать порядковый номер. Автоматически генерируемое значение будет на единицу большим, чем наибольшее значение в таблице. Первая введенная строка будет иметь порядковый номер 1. В таблице можно иметь не более одного столбца *AUTO_INCREMENT*, и он должен индексироваться.
- Объявить столбец первичным ключом таблицы с помощью выражения *PRIMARY KEY*.
- Объявить столбец внешним ключом, используя выражение *FOREIGN KEY*, с ссылкой на соответствующую таблицу с помощью выражения *REFERENCES*.
- Индексировать столбец с помощью слов *INDEX* или *KEY* (синонимы). Такие столбцы не обязательно должны содержать уникальные значения.
- Индексировать столбец с помощью слова *UNIQUE,* которое используется для указания того, что столбец должен содержать уникальные значения.
- Создать полнотекстовые индексы на основе столбцов типа *TEXT, CHAR* или *VARCHAR* с помощью слова *FULLTEXT* (только с таблицами *MyISAM*).

После закрывающей скобки можно указать тип таблицы:

- *MyISAM* – таблицы этого типа являются «родными» для MySQL, работают очень быстро и поддерживают полнотекстовую индексацию;
- *InnoDB* – ACID-совместимый механизм хранения, поддерживающий транзакции, внешние ключи, каскадное удаление и блокировки на уровне строк;
- *BDB (Berkeley DB)* – является механизмом хранения, который обеспечивает поддержку транзакций и блокировки на уровне страниц;

- *MEMORY (HEAP)* – таблицы целиком хранятся в оперативной памяти и никогда не записываются на диск, поэтому работают очень быстро, но ограничены в размерах и не допускают возможности восстановления в случае отказа системы;
- *MERGE* – тип позволяет объединить несколько таблиц *MyISAM* с одной структурой, чтобы к ним можно было направлять запросы как к одной таблице;
- *NDB Cluster* – тип предназначен для организации кластеров MySQL, когда таблицы распределены между несколькими компьютерами, объединенными в сеть;
- *ARCHIVE* – тип введен для хранения большого объема данных в сжатом формате; таблицы поддерживают только два SQL-оператора: *INSERT* и *SELECT,* причем оператор *SELECT* выполняется по методу полного сканирования таблицы;
- *CSV* – формат представляет собой обычный текстовый фал, записи в котором хранятся в строках, а поля разделены точкой с запятой (широко распространен в компьютерном мире, любая программа, поддерживающая CSV-формат, может открыть такой файл);
- *FEDERATED* – тип позволяет хранить данные в таблицах на другой машине сети (при создании таблицы в локальной директории создается только файл определения структуры таблицы, а все данные хранятся на удаленной машине).

MySQL поддерживает следующие типы данных, допустимые для столбцов:

- числовые;
- строковые;
- календарные;
- *NULL* – специальный тип, обозначающий отсутствие информации.

Числовые типы используются для хранения чисел и представляют два подтипа: □ точные числовые типы;

- приближенные числовые типы.

К точным числовым типам (табл. 1) относятся целый тип *INTEGER* и его вариации, а также вещественный тип *DECIMAL* (синонимы *NUMERIC* и *DEC*). Последний используется для представления денежных данных.

Числовые типы могут характеризоваться максимальной длиной *M*. Для типа *DECIMAL* параметр *M* задает число символов для отображения всего числа, а *D* – для его дробной части. Например: *b_price DECIMAL (5, 2)*. Цифра 5 определяет общее число символов под число, а цифра 2 – количество знаков после запятой (интервал величин от –99.99 до 99.99). Можно не использовать параметры вообще, указать только общую длину или указать длину и число десятичных разрядов.

Объявления точных числовых типов можно завершать ключевыми словами *UNSIGNED* и (или) *ZEROFILL*. Ключевое слово *UNSIGNED* указывает, что столбец содержит только положительные числа или нули. Ключевое слово *ZEROFILL* означает, что число будет отображаться с ведущими нулями.

Таблица 14.1. Числовые типы данных

Тип	**Объем памяти**	**Диапазон**
TINYINT (M) *TINYINT UNSIGNED*	1 байт	от -128 до 127 (от -2^7 до 2^7-1) от 0 до 255 (от 0 до 2^8-1)
SMALLINT (M) *SMALLINT UNSIGNED*	2 байта	от -32 768 до 32 767 (от -2^{15} до $2^{15}-1$) от 0 до 65 535 (от 0 до $2^{16}-1$)
MEDIUMINT (M) *MEDIUMINT UNSIGNED*	3 байта	от -8 388 608 до 8 388 607 (от -2^{23} до $2^{23}-1$) от 0 до 16 777 215 (от 0 до $2^{24}-1$)
INT (INTEGER) (M) INT UNSIGNED	4 байта	от -2 147 683 648 до 2 147 683 647 (от -2^{31} до $2^{31}-1$) от 0 до 4 294 967 295 (от 0 до $2^{32}-1$)
BIGINT (M) *BIGINT UNSIGNED*	8 байт	(от-2^{63} до$2^{63}-1$) (от 0 до $2^{64}-1$)
BIT (M)	(*M*+7)/8 байт	От 1 до 64 битов, в зависимости от значения *M*
BOOL, BOOLEAN	1 байт	0 (*false*) либо 1 (*true*)
DECIMAL (M, D), *NUMERUC (M, D)*	*M* + 2 байта	Повышенная точность, зависит от параметров *M* и *D*

К приближенным числовым типам (табл. 14.2) относятся:

□ *FLOAT* – представление чисел с плавающей запятой с обычной точностью; □ *DOUBLE* – представление чисел с плавающей запятой с двойной точностью.

Таблица 14.2. Приближенные к числовым типы данных

Тип	**Объем памяти**	**Диапазон**
FLOAT (M, D) $1.175494351*10^{-39}$	4 байта	Минимальное по модулю значение
		Максимальное по модулю значение $3.402823466*10^{38}$
DOUBLE (M, D), 8 бай *REAL (M,D),* 2.22507 *DOUBLE PRECISION (M,D)*		Минимальное по модулю значение $38585072014*10^{-308}$ Максимальное по модулю значение $1.797693134862315*10^{308}$

Числовые типы с плавающей точкой также могут иметь параметр *UNSIGNED*. Атрибут предотвращает хранение в столбце отрицательных величин, но максимальный интервал величин столбца остается прежним.

Приближенные числовые данные могут задаваться в обычной форме (например, 45.67) и в форме с плавающей точкой (например, 5.456E-02 или 4.674E+04).

Текстовые типы и строки (табл. 3):

- *CHAR* – хранение строк фиксированной длины;
- *VARCHAR* – хранение строк переменной длины;
- *TEXT, BLOB* и их вариации – хранение больших фрагментов текста;
- *ENUM* и *SET* – хранение значений из заданного списка.

Таблица 14.3. Тестовые типы и строки

Тип	Объем памяти	Максимальный размер
CHAR(M)	*M* символов	*M* символов
VARCHAR(M)	*L*+1 символов	*M* символов
TINYBLOB, TINYTEXT	*L*+1 символов	2^8-1 символов
BLOB, TEXT	*L*+2 символов	2^{16}-1 символов
MEDIUMBLOB, MEDIUMTEXT	*L*+3 символов	2^{24}-1 символов
LONGBLOB, LONGTEXT	*L*+4 символов	2^{32}-1 символов
ENUM('value 1', 'value2 ', ...)	1 или 2 байта	65 535 элементов
SET('value 1', 'value2', ...)	1, 2, 3, 4 или 8 байт	64 элемента

Здесь *L* – длина хранимой в ячейке строки, а приплюсованные к *L* байты – накладные расходы для хранения длины строки.

Для строк *VARCHAR* требуется количество символов, равное длине строки плюс 1 байт, тогда как тип *CHAR(M)*, независимо от длины строки, использует для ее хранения все *M* символов. Тип *CHAR* обрабатывается эффективнее переменных типов. Нельзя смешивать в таблице столбцы *CHAR* и *VARCHAR*. Если есть столбец переменной длины, все столбцы типа *CHAR* будут приведены к типу *VARCHAR*.

Типы *BLOB* и *TEXT* аналогичны и отличаются в деталях. При выполнении операций над столбцами типа *TEXT* учитывается кодировка, а типа *BLOB* – нет. Тип *TEXT* используется для хранения больших объемов текста, тип *BLOB* – для больших двоичных объектов (электронные документы, изображения, звук). Основное отличие *TEXT* от *CHAR* и *VARCHAR* – поддержка полнотекстового поиска.

Строки типов данных *ENUM* и *SET* принимают значения из заданного списка. Значение типа *ENUM* должно содержать точно одно значение из указанного множества, тогда как столбцы *SET* могут содержать любой или все

элементы заданного множества одновременно. Для типа *SET* (как и для *ENUM*) при объявлении задается список возможных значений, но ячейка может принимать любое значение из списка, а пустая строка означает, что ни один из элементов списка не выбран.

Типы *ENUM* и *SET* задаются списком строк, но во внутреннем представлении элементы множеств сохраняются в виде чисел. Элементы типа *ENUM* нумеруются последовательно, начиная с 1. Под столбец может отводиться 1 байт (до 256 элементов в списке) или 2 байта (от 257 до 65536 элементов в списке). Элементы типа *SET* обрабатываются как биты, размер типа определяется числом элементов в списке: 1 байт (от 1 до 8 элементов), 2 байта (от 9 до 16 элементов), 3 байта (от 17 до 24 элементов), 4 байта (от 25 до 32 элементов) и 8 байт (от 33 до 64 элементов).

Календарные типы данных (табл. 14.4):

- *DATE* – для хранения даты (формат *YYYY-MM-DD* для дат вида 2009-10-15 и формат *YY-MM-DD* для дат вида 09-10-15);
- *TIME* – для хранения времени суток (формат *HH:MM:SS*, где *HH* – часы, *MM* – минуты, *SS* – секунды, например, 10:48:56);
- *DATETIME*– для представления и даты, и времени суток;
- *TIMESTAMP* – если в соответствующем столбце строки не указать конкретное значение или *NULL*, там будет записано время, когда соответствующая строка была создана или в последний раз изменена (в формате *DATETIME*); □ *YEAR* – позволяет хранить только год.

Таблица 14.4. Календарные типы данных

Тип	**Объем памяти**	**Диапазон**
DATE	3 байта	от '1000-01-01' до '9999-12-31'
TIME	3 байта	от '-828:59:59' до '828:59:59'
DATATIME	8 байт	от '1000-01-01 00:00:00' до '9999-12-31 00:00:00'
TIMESTAMP (M)	4 байта	от '1970-01-01 00:00:00' до '2038-12-31 59:59:59'
YEAR(2) *YEAR(4)*	1 байт	формат *YY,* диапазон – от 1970 до 2069 формат *YYYY*, диапазон – от 1901 до 2155

Дни, месяцы, часы, минуты и секунды можно записывать как с ведущим нулем, так и без него. Например, все следующие записи идентичны:

'2009-04-06 02:04:08' '2009-4-06 02:04:08' '2009-4-6 02:04:08' '2009-4-6 2:04:08' '2009-4-6 2:4:08' '2009-4-6 2:4:8'

В качестве разделителя между годами, месяцами, днями, часами, минутами, секундами может выступать любой символ, отличный от цифры. Так, следующие значения идентичны:

'09-12-31 11:30:45' '09.12.31 11+30+45' '09/12/31 11*30*45'

При указании времени после секунд через точку можно указать микросекунды, т. е.

использовать расширенный формат вида *HH:MM:SS.FFFFFF*, например '10:25:14.000001'. Кроме того, можно использовать краткие форматы *HH:MM* и *HH* (вместо пропущенных величин будут подставлены нулевые значения).

Если время задается в недопустимом формате, то в поле записывается нулевое значение. Нулевое значение присваивается полям временного типа по умолчанию, когда им не присваивается инициирующее значение (табл. 14.5).

Таблица 14.5. Значения времен полей временного типа

Тип	Нулевое значение
DATE	'0000-00-00'
TIME	'00:00:00'
DATATIME	'0000-00-00 00:00:00'
TIMESTAMP	00000000000000
YEAR	0000

Формат *TIMESTAMP* совпадает с *DATETIME*, но во внутреннем представлении дата хранится как число секунд, прошедших с полуночи 1 января 1970 г. (такое исчисление принято в операционной системе UNIX, а дата 01.01.1970 считается началом эпохи UNIX и днем рождения операционной системы).

Если в таблице несколько столбцов *TIMESTAMP*, при модификации записи текущее время будет записываться только в один из них (первый). Можно явно указать столбец, которому необходимо назначать текущую дату при создании или изменении записи. Чтобы поля принимали текущую дату при создании записи, следует после определения столбца добавить *DEFAULT CURRENT_TIMESTAMP*. Если текущее время должно выставляться при модификации записи, при использовании оператора *UPDATE* следует добавить *ON UPDATE CURRENT_TIMESTAMP*.

Тип данных *NULL* используется, когда информации недостаточно и для части данных нельзя определить, какое значение они примут. Для указания того, что поле может принимать неопределенное значение, в определении столбца после типа данных следует указать ключевое слово *NULL*. Если поле не должно принимать значение *NULL*, следует указать ключевое слово *NOT NULL*.

Рекомендации по выбору типа данных.

- Обработка числовых данных происходит быстрее строковых. Так как типы *ENUM* и *SET* имеют внутреннее числовое представление, им следует отдавать предпочтение перед другими видами строковых данных, если это возможно.
- Производительность можно увеличить за счет представления строк в виде чисел. Пример – преобразование IP-адреса из строки в *BIGINT*.

Это позволит уменьшить размер таблицы и значительно увеличить скорость при сортировке и выборке данных, но потребует дополнительных преобразований.

- Базы данных хранятся на жестком диске, и чем меньше места они занимают, тем быстрее происходит поиск и извлечение. Если есть возможность, следует выбирать типы данных, занимающие меньше места.
- Типы фиксированной длины обрабатываются быстрее типов переменной длины, т. к. в последнем случае при частых удалениях и модификациях таблицы происходит ее фрагментация.
- Если применение столбцов с данными переменной длины неизбежно, для дефрагментации таблицы следует применять команду *OPTIMIZE TABLE.*

Обеспечение ссылочной целостности. Задается конструкцией:

FOREIGN KEY [name_key] (col1, ...) REFERENCES tbl (tbl_col, ...) [ON DELETE {CASCADE | SET NULL | NO ACTION | RESTRICT | SET DEFAULT}] [ON UPDATE {CASCADE | SET NULL | NO ACTION | RESTRICT | SET DEFAULT}]

Конструкция позволяет задать внешний ключ с необязательным именем *name_key* на столбцах, которые задаются в круглых скобках (один или несколько). Ключевое слово *REFERENCES* указывает таблицу *tbl*, на которую ссылается внешний ключ, в круглых скобках указываются имена столбцов. Необязательные конструкции *ON DELETE* и *ON UPDATE* позволяют задать поведение СУБД при удалении и обновлении строк из таблицы-предка. Параметры, следующие за этими ключевыми словами, имеют следующие значения:

- *CASCADE* – при удалении или обновлении записи в таблице-предке, содержащей первичный ключ, записи со ссылками на это значение в таблице-потомке удаляются или обновляются автоматически;
- *SET NULL* – при удалении или обновлении записи в таблице-предке, содержащей первичный ключ, в таблице-потомке значения внешнего ключа, ссылающегося на таблицу-предка, устанавливаются в *NULL*;
- *NO ACTION* – при удалении или обновлении записей, содержащих первичный ключ, с таблицей-потомком никаких действий не производится;
- *RESTRICT* – если в таблице-потомке имеются записи, ссылающиеся на первичный ключ таблицы-предка, при удалении или обновлении записей с таким первичным ключом возвращается ошибка;
- *SET DEFAULT* – согласно стандарту *SQL*, при удалении или обновлении первичного ключа в таблице-потомке для ссылающихся на него записей в поле внешнего ключа должно устанавливаться

значение по умолчанию (в MySQL это ключевое слово зарезервировано, но не обрабатывается).

Создание индексов. Индексы играют большую роль в БД, т. к. это основной способ ускорения их работы. Записи в таблице располагаются хаотически. Чтобы найти нужную запись, необходимо сканировать всю таблицу, на что уходит много времени. Идея индексов состоит в том, чтобы создать для столбца копию, которая постоянно будет поддерживаться в отсортированном состоянии. Это позволяет быстро осуществлять поиск по такому столбцу.

Все необходимые индексы формируются при создании таблицы. Индексированы будут все столбцы, объявленные как *PRIMARY KEY, KEY, UNIQUE* или *INDEX*. Индекс также можно добавить с помощью оператора *CREATE INDEX*. Перед выполнением оператор преобразуется в оператор *ALTER TABLE*. Например, создание индекса с именем *name* на основе поля *u_name* из таблицы *users*:

CREATE INDEX name ON users (u_name);

Перед ключевым словом *INDEX* может присутствовать *UNIQUE*, требующее уникальности ограничения.

Корректность таблиц в БД можно проверить с помощью оператора

SHOW TABLES;

Более подробную информацию о структуре таблицы дает команда

DESCRIBE имя_таблицы;

Переименование БД. Специального оператора переименования БД нет, но можно переименовать каталог БД в системном каталоге (*…\DATA*).

Удаление БД. Удалить всю БД вместе с ее содержимым можно командой:

DROP DATABASE [IF EXISTS] имя_базы_данных;

Удаление таблиц и индексов. Удалить таблицу можно с помощью оператора:

DROP TABLE [IF EXISTS] имя_таблицы; Удалить индекс можнос помощью оператора:

DROP INDEX имя_индекса ON имя_таблицы;

Изменение структуры таблиц. Изменить структуру существующей таблицы можно с помощью оператора *ALTER TABLE*. Например, можно создать индекс *name* для таблицы *users* следующим образом: *ALTER TABLE users ADD INDEX name (u_name);*

Оператор *ALTER TABLE* является исключительно гибким, поэтому он имеет огромное множество дополнительных ключевых слов.

Задание для практической работы

При выполнении практической работы необходимо для заданной предметной области средствами MySQL:

- создать базу данных;
- создать таблицы, определить поля таблиц, индексы;
- определить связи между таблицами и ограничения целостности; □ составить отчет по практической работе.

Создайте базу данных *book* Интернет-магазина, торгующего компьютерной литературой. В базе данных должна поддерживаться следующая информация:

- тематические каталоги, по которым сгруппированы книги;
- предлагаемые книги (название, автор, год издания, цена, имеющееся на складе количество);
- зарегистрированные покупатели (имя, отчество, фамилия, телефон, адрес электронной почты, статус – авторизованный, неавторизованный, заблокированный, активный с хорошей кредитной историей);
- покупки, совершенные в магазине (время совершения покупки, число приобретенных экземпляров книги).

Логическая модель данных предметной области в стандарте IDEF1X представлена на рис. 14.1. Выделены сущности *КАТАЛОГ, КНИГА, КЛИЕНТ, ЗАКАЗ*, между которыми установлены не идентифицирующие связи мощностью один-ко-многим, определенные спецификой предметной области.

Рисунок 14.1. Логическая модель данных предметной области

Физическая модель данных предметной области в стандарте IDEF1X для целевой СУБД MySQL представлена на рис. 14.2.

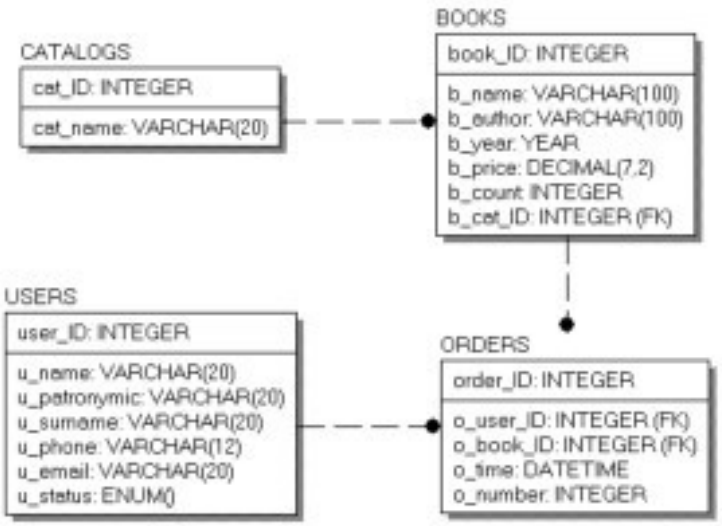

Рисунок 14.2. Физическая модель предметной области

База данных *book* состоит из четырех таблиц:

- *catalogs* – список торговых каталогов;
- *books* – список предлагаемых книг;
- *users* – список зарегистрированных пользователей магазина; □ *orders* – список заказов (осуществленных сделок). Таблица *catalogs* состоит из двух полей: □ *cat_ID* – уникальный код каталога;
- *cat_name* – имя каталога.

Оба поля должны быть снабжены атрибутом *NOT NULL*, поскольку неопределенное значение для них недопустимо.

Таблица *books* состоит из семи полей:

- *book_ID* – уникальный код книги;
- *b_name* – название книги;
- *b_author* – автор книги;
- *b_year* – год издания;
- *b_price* – цена книги;
- *b_count* – количество книг на складе;
- *b_cat_ID* – код каталога из таблицы *catalogs*.

Цена книги *b_price* и количество экземпляров на складе *b_count* могут иметь атрибут *NULL*. На момент доставки часто неизвестны количество товара и его цена, но отразить факт наличия товара в прайс-листе необходимо.

Поле *b_cat_ID* устанавливает связь между таблицами *catalogs* и *books*. Это поле должно быть объявлено как внешний ключ (FK) с правилом каскадного удаления и обновления. Обновление таблицы *catalogs* вызовет автоматическое обновление таблицы *books*. Удаление каталога в таблице *catalogs* приведет к автоматическому удалению всех записей в таблице *books*, соответствующих каталогу.

Таблица *users* состоит из семи полей:

- *user_ID* – уникальный код покупателя;

- *u_name* – имя покупателя;
- *u_patronymic* – отчество покупателя;
- *u_surname* – фамилия покупателя;
- *u_phone* – телефон покупателя (если имеется); ☐ *u_email* – e-mail покупателя (если имеется);
- *u_status* – статус покупателя.

Статус покупателя представлен полем типа *ENUM*, которое может принимать одно из четырех значений:

- *active* – авторизованный покупатель, который может осуществлять покупки через Интернет;
- *passive* – неавторизованный покупатель (значение по умолчанию), который осуществил процедуру регистрации, но не подтвердил ее и пока не может осуществлять покупки через Интернет, однако ему доступны каталоги для просмотра;
- *lock* – заблокированный покупатель, не может осуществлять покупки и просматривать каталоги магазина;
- *gold* – активный покупатель с хорошей кредитной историей, которому предоставляется скидка при следующих покупках в магазине.

Поля *u_phone* и *u_email* могут быть снабжены атрибутом *NULL*. Остальные поля должны получить атрибут *NOT NULL*.

Таблица *orders* включает пять полей:

- *order_ID* – уникальный номер сделки;
- *o_user_ID* – номер пользователя из таблицы *users*;
- *o_book_ID* – номер товарной позиции из таблицы *books*; ☐ *o_time* – время совершения сделки;
- *o_number* – число приобретенных товаров.

Поля таблицы *orders* должны быть снабжены атрибутом *NOT NULL*, т. к. при совершении покупки вся информация должна быть занесена в таблицу.

В таблице *orders* устанавливается связь с таблицами *users* (за счет поля *o_user_ID*) и *books* (за счет поля *o_book_ID*). Эти поля объявлены как внешние ключи (FK) с правилом каскадного удаления и обновления. Обновление таблиц *users* и *books* приведет к автоматическому обновлению таблицы *orders*. Удаление любого пользователя в таблице *users* приведет к автоматическому удалению всех записей в таблице *orders*, соответствующих этому пользователю.

Операторы создания БД *book* имеют следующий вид (целесообразно создать в *Блокноте* текстовый файл и записать туда эти операторы).

```
DROP DATABASE IF EXISTS book;
CREATE DATABASE book;
USE book;
CREATE TABLE catalogs (
```

```
        cat_ID int(6) NOT NULL
    AUTO_INCREMENT,     cat_name varchar(20)
    NOT NULL,
        PRIMARY KEY  (cat_ID)
    ) TYPE=InnoDB;
    CREATE TABLE books (
        book_ID int(6) NOT NULL
    AUTO_INCREMENT,     b_name
    varchar(100) NOT NULL,        b_author
    varchar(100) NOT NULL,        b_year year
    NOT NULL,
        b_price decimal(7,2) NULL
    default '0.00',        b_count int(6)
    NULL default '0', b_cat_ID int(6)
    NOT NULL default '0',
        PRIMARY KEY  (book_ID),
        FOREIGN   KEY   (b_cat_ID)  REFERENCES     catalogs(cat_ID)
    ON    DELETE CASCADE ON UPDATE CASCADE ) TYPE=InnoDB;
    CREATE TABLE users (
        user_ID int(6) NOT NULL
    AUTO_INCREMENT,     u_name
    varchar(20) NOT NULL,
    u_patronymic varchar(20) NOT NULL,
    u_surname varchar(20) NOT NULL,
    u_phone varchar(12) NULL,
        u_email varchar(20) NULL,
     u_status ENUM ('active','passive','lock','gold') default 'passive',
        PRIMARY KEY  (user_ID)
    ) TYPE=InnoDB;
    CREATE TABLE orders (
        order_ID int(6) NOT NULL
    AUTO_INCREMENT,     o_user_ID int NOT NULL,
        o_book_ID int NOT NULL,
     o_time datetime NOT NULL default '0000-00-00 00:00:00',
       o_number int(6) NOT NULL default '0',
        PRIMARY KEY  (order_ID),
        FOREIGN   KEY   (o_book_ID)        REFERENCES
    books(book_ID)    ON    DELETE CASCADE ON UPDATE CASCADE,
     FOREIGN KEY (o_user_ID) REFERENCES users(user_ID) ON DELETE
CASCADE ON UPDATE CASCADE
    )TYPE=InnoDB;
```

Практическая работа №15 «Задание значений и ограничений поля. Проверка введенного в поле значения. Отображение данных числового типа и типа дата»

Цель работы: овладение практическими навыками обработки табличных данных

Вставка, удаление и обновление данных

После создания БД и таблиц перед разработчиком встает задача заполнения таблиц данными. В реляционных БД традиционно применяют три подхода:

- однострочный оператор *INSERT* – добавляет в таблицу новую запись;
- многострочный оператор *INSERT* – добавляет в таблицу несколько записей;
- пакетная загрузка *LOAD DATA INFILE* – добавление данных из файла.

Вставка данных с помощью оператора *INSERT*. Однострочный оператор *INSERT* может использоваться в нескольких формах. Упрощенный синтаксис первой формы: *INSERT [IGNORE] [INTO] имя_таблицы [(имя_столбца, ...)]*

VALUES (выражение, ...);

Оператор вставляет новую запись в таблицу *имя_таблицы*. Значения полей записи перечисляются в списке *(выражение, ...)*. Порядок следования столбцов задается списком *(имя_столбца, ...)*. Список столбцов *(имя_столбца, ...)* позволяет менять порядок следования столбцов при добавлении.

Первичный ключ таблицы является уникальным, и попытка добавить уже существующее значение приведет к ошибке. Чтобы новые записи с дублирующим ключом отбрасывались без генерации ошибки, следует добавить после оператора *INSERT* ключевое слово *IGNORE*.

Другая форма оператора *INSERT* предполагает использование слова *SET*:

INSERT [IGNORE] [INTO] имя_таблицы

SET имя_столбца1 = выражение1, имя_столбца2 = выражение2, ... ;

Оператор заносит в таблицу *имя_таблицы* новую запись, столбец *имя_столбца* в которой получает значение *выражение*.

Многострочный оператор *INSERT* совпадает по форме с однострочным оператором, но после ключевого слова *VALUES* добавляется через запятую несколько списков *(выражение, ...)*.

Практические примеры использования оператора *INSERT* для заполнения учебной БД *book* см. ниже, в пункте «Пример выполнения работы».

Удаление данных. Для удаления записей из таблиц предусмотрены:

- оператор *DELETE;*

- оператор *TRUNCATE TABLE.*

Оператор *DELETE* имеет следующий синтаксис:

DELETE FROM имя_таблицы
[WHERE условие]
[ORDER BY имя_поля]
[LIMIT число_строк];

Оператор удаляет из таблицы *имя_таблицы* записи, удовлетворяющие условию. В следующем примере из таблицы *catalogs* удаляются записи, имеющие значение первичного ключа *catalog_id* больше двух.

```
mysql> DELETE FROM catalogs WHERE cat_ID>2;
Query OK, 3 rows affected (0.05 sec)

mysql> SELECT * FROM catalogs;
+--------+-------------------+
| cat_ID | cat_name          |
+--------+-------------------+
|      1 | Программирование  |
|      2 | Интернет          |
+--------+-------------------+
2 rows in set (0.00 sec)
```

Если в операторе отсутствует условие *WHERE*, удаляются все записи таблицы.

```
mysql> DELETE FROM catalogs;
Query OK, 2 rows affected (0.03 sec)

mysql> SELECT * FROM catalogs;
Empty set (0.00 sec)
```

Ограничение *LIMIT* позволяет задать максимальное число записей, которые могут быть удалены. Следующий запрос удаляет все записи таблицы *orders*, но не более 3 записей.

```
mysql> DELETE FROM orders LIMIT 3;
Query OK, 3 rows affected (0.01 sec)

mysql> SELECT * FROM orders;
+----------+-----------+-----------+---------------------+----------+
| order_ID | o_user_ID | o_book_ID | o_time              | o_number |
+----------+-----------+-----------+---------------------+----------+
|        4 |         4 |        20 | 2009-03-10 18:20:00 |        1 |
|        5 |         3 |        20 | 2009-03-17 19:15:36 |        1 |
+----------+-----------+-----------+---------------------+----------+
2 rows in set (0.00 sec)
```

Конструкция *ORDER BY* обычно применяется вместе с ключевым словом *LIMIT*. Например, если необходимо удалить 20 первых записей таблицы, то производится сортировка по полю типа *DATETIME* – тогда в первую очередь будут удалены самые старые записи.

Оператор *TRUNCATE TABLE* полностью очищает таблицу и не допускает условного удаления. Он аналогичен оператору *DELETE* без условия *WHERE* и ограничения *LIMIT*. Удаление происходит гораздо быстрее, т. к. осуществляется не перебор записей, а полное очищение таблицы.

```
mysql> TRUNCATE TABLE orders;
Query OK, 5 rows affected (0.03 sec)

mysql> SELECT * FROM orders;
Empty set (0.00 sec)
```

Обновление данных. Обновление данных (изменение значений полей в существующих записях) обеспечивают:

□ оператор *UPDATE*; □

оператор
REPLACE.

Оператор *UPDATE* позволяет обновлять отдельные поля в существующих записях. Имеет следующий синтаксис

UPDATE [IGNORE] имя_таблицы
SET имя_столбца1 = выражение1 [, имя_столбца2 = выражение2 ...]
[WHERE условие]
[ORDER BY имя_поля]
[LIMIT число_строк] ;

После ключевого слова *UPDATE* указывается таблица, которая изменяется. В предложении *SET* указывается, какие столбцы обновляются и устанавливаются их новые значения. Необязательное условие *WHERE* позволяет задать критерий отбора строк (обновляться будут только строки, удовлетворяющие условию).

Если указывается необязательное ключевое слово *IGNORE*, то команда обновления не будет прервана, даже если при обновлении возникнет ошибка дублирования ключей. Строки, породившие конфликтные ситуации, обновлены не будут.

Запрос, изменяющий в таблице *catalogs* «Сети» на «Компьютерные сети».

```
mysql> UPDATE catalogs SET cat_name='Компьютерные сети'
    -> WHERE cat_name='Сети';
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM catalogs;
+--------+--------------------+
| cat_ID | cat_name           |
+--------+--------------------+
|      1 | Программирование   |
|      2 | Интернет           |
|      3 | Базы данных        |
|      4 | Компьютерные сети  |
|      5 | Мультимедиа        |
+--------+--------------------+
5 rows in set (0.00 sec)
```

Обновлять можно всю таблицу. Пусть требуется уменьшить на 5 % цену на все книги. Для этого следует старую цену в рублях умножить на 0,95.

```
mysql> UPDATE books SET b_price=b_price*0.95;
Query OK, 30 rows affected (0.03 sec)
Rows matched: 30  Changed: 30  Warnings: 0

mysql> SELECT book_ID, b_name, b_price FROM books;
+---------+-----------------------------------------+---------+
| book_ID | b_name                                  | b_price |
+---------+-----------------------------------------+---------+
|       1 | JavaScript в кармане                    |   39.90 |
|       2 | Visual FoxPro 9.0                       |  627.00 |
|       3 | C++ Как он есть                         |  207.10 |
|       4 | Создание приложений с помощью C#        |  160.55 |
|       5 | Delphi. Народные советы                 |  230.85 |
|       6 | Delphi. Полное руководство              |  475.00 |
|       7 | Профессиональное программирование на PHP |  293.55 |
|       8 | Совершенный код                         |  732.45 |
|       9 | Практика программирования               |  203.30 |
|      10 | Принципы маршрутизации в Internet       |  406.60 |
|      11 | Поиск в Internet                        |  101.65 |
|      12 | Web-конструирование                     |  168.15 |
|      13 | Самоучитель Интернет                    |  114.95 |
|      14 | Популярные интернет-браузеры            |   77.90 |
|      15 | Общение в Интернете                     |   80.75 |
|      16 | Базы данных                             |  309.70 |
|      17 | Базы данных. Разработка приложений      |  179.55 |
|      18 | Раскрытие тайн SQL                      |  190.00 |
|      19 | Практикум по Access                     |   82.65 |
|      20 | Компьютерные сети                       |  598.50 |
|      21 | Сети. Поиск неисправностей              |  412.30 |
|      22 | Безопасность сетей                      |  438.90 |
|      23 | Анализ и диагностика компьютерных сетей |  326.80 |
|      24 | Локальные вычислительные сети           |   77.90 |
|      25 | Цифровая фотография                     |  141.55 |
|      26 | Музыкальный компьютер для гитариста     |  206.15 |
|      27 | Видео на ПК                             |  219.45 |
|      28 | Мультипликация во Flash                 |  200.45 |
|      29 | Запись CD и DVD                         |  158.65 |
|      30 | Запись и обработка звука на компьютере  |   48.45 |
+---------+-----------------------------------------+---------+
30 rows in set (0.00 sec)
```

Инструкции *LIMIT* и *ORDER BY* позволяют ограничить число изменяемых записей. При этом за один запрос можно обновить несколько столбцов таблицы. Например, необходимо в таблице *books* для десяти самых дешевых товарных позиций уменьшить количество книг на складе на единицу, а цену – на 5 %.

```
mysql> UPDATE books SET b_price=b_price*0.95,b_count=b_count-1
    -> ORDER BY b_price LIMIT 10;
Query OK, 10 rows affected (0.05 sec)
Rows matched: 10  Changed: 10  Warnings: 0

mysql> SELECT book_ID, b_name, b_price, b_count FROM books;
+---------+-----------------------------------------+---------+---------+
| book_ID | b_name                                  | b_price | b_count |
+---------+-----------------------------------------+---------+---------+
|       1 | JavaScript в кармане                    |   39.90 |       9 |
|       2 | Visual FoxPro 9.0                       |  660.00 |       2 |
|       3 | C++ Как он есть                         |  218.00 |       4 |
|       4 | Создание приложений с помощью C#        |  169.00 |       1 |
|       5 | Delphi. Народные советы                 |  243.00 |       6 |
|       6 | Delphi. Полное руководство              |  500.00 |       6 |
|       7 | Профессиональное программирование на PHP |  309.00 |       5 |
|       8 | Совершенный код                         |  771.00 |       1 |
|       9 | Практика программирования               |  214.00 |      12 |
|      10 | Принципы маршрутизации в Internet       |  428.00 |       4 |
|      11 | Поиск в Internet                        |  101.65 |       1 |
|      12 | Web-конструирование                     |  177.00 |       6 |
|      13 | Самоучитель Интернет                    |  114.95 |       3 |
|      14 | Популярные интернет-браузеры            |   77.90 |       5 |
|      15 | Общение в Интернете                     |   80.75 |       4 |
|      16 | Базы данных                             |  326.00 |       2 |
|      17 | Базы данных. Разработка приложений      |  189.00 |       6 |
|      18 | Раскрытие тайн SQL                      |  200.00 |       3 |
|      19 | Практикум по Access                     |   82.65 |       5 |
|      20 | Компьютерные сети                       |  630.00 |       6 |
|      21 | Сети. Поиск неисправностей              |  434.00 |       4 |
|      22 | Безопасность сетей                      |  462.00 |       5 |
|      23 | Анализ и диагностика компьютерных сетей |  344.00 |       3 |
|      24 | Локальные вычислительные сети           |   77.90 |       7 |
|      25 | Цифровая фотография                     |  141.55 |      19 |
|      26 | Музыкальный компьютер для гитариста     |  217.00 |      15 |
|      27 | Видео на ПК                             |  231.00 |      10 |
|      28 | Мультипликация во Flash                 |  211.00 |      20 |
|      29 | Запись CD и DVD                         |  158.65 |      11 |
|      30 | Запись и обработка звука на компьютере  |   48.45 |       7 |
+---------+-----------------------------------------+---------+---------+
30 rows in set (0.00 sec)
```

Оператор *REPLACE* работает как оператор *INSERT,* за исключением того, что старая запись с тем же значением индекса *UNIQUE* или *PRIMARY KEY* перед внесением новой будет удалена. Если не используются индексы *UNIQUE* или *PRIMARY KEY,* то применение оператора

REPLACE не имеет смысла.

Синтаксис оператора *REPLACE* аналогичен синтаксису оператора *INSERT*:

REPLACE [INTO] имя_таблицы [(имя_столбца, ...)]
VALUES (выражение, ...)

В таблицу вставляются значения, определяемые в списке после ключевого слова *VALUES*. Задать порядок столбцов можно при помощи необязательного списка, следующего за именем таблицы. Как и оператор *INSERT*, оператор *REPLACE* допускает многострочный формат.

Задание для практической работы

При выполнении практической работы необходимо для заданной предметной области средствами MySQL:

□ заполнить согласованными данными таблицы БД; □ при необходимости исправить введенную информацию; □ составить отчет по практической работе.

Операторы заполнения БД *book* имеют следующий вид.

```
USE book;
SET CHARACTER SET cp1251;
DELETE FROM catalogs;
INSERT INTO catalogs VALUES (1,'Программирование');
INSERT INTO catalogs VALUES (2,'Интернет');
INSERT INTO catalogs VALUES (3,'Базы данных');
INSERT INTO catalogs VALUES (4,'Сети');
INSERT INTO catalogs VALUES (5,'Мультимедиа');
DELETE FROM books;
INSERT INTO books VALUES (1,'JavaScript в кармане','Рева О.Н.', 2008,
42.00, 10, 1);
 INSERT INTO books VALUES (2,'Visual FoxPro 9.0','Клепинин В.Б.',
   2007, 660.00, 2, 1); INSERT INTO books VALUES (3,'C++ Как он
                есть','Тимофеев В.В.',2009, 218.00, 4, 1);
INSERT INTO books VALUES (4,'Создание приложений с помощью
C#','Фаронов В.В.', 2008,
169.00, 1, 1);
INSERT INTO books VALUES (5,'Delphi. Народные советы','Шкрыль
A.A.',2007,243.00,6,1);
INSERT INTO books VALUES (6,'Delphi. Полное руководство','Сухарев
M.',2008,500.00,6,1);
INSERT INTO books VALUES (7,'Профессиональное программирование
на PHP',
'Шлосснейгл Дж.', 2006, 309.00, 5, 1);
```

INSERT INTO books VALUES (8,'Совершенный код','Макконнелл С.', 2007, 771.00, 1, 1);

INSERT INTO books VALUES (9,'Практика программирования','Керниган Б.', 2004, 214.00,

12, 1);

INSERT INTO books VALUES (10,'Принципы маршрутизации в Internet','Хелеби С.', 2001,

428.00, 4, 2);

INSERT INTO books VALUES (11,'Поиск в Internet','Гусев В.С.',2004,107.00,2,2);

INSERT INTO books VALUES (12,'Web-конструирование','Дуванов А.А.', 2003, 177.00, 6, 2);

INSERT INTO books VALUES (13,'Самоучитель Интернет','Константинов Ю.П.', 2009,

121.00, 4, 2);

INSERT INTO books VALUES (14,'Популярные интернет-браузеры','Маринин С.А.', 2007,

82.00, 6, 2);

INSERT INTO books VALUES (15,'Общение в Интернете','Экслер А.', 2006, 85.00, 5, 2);

INSERT INTO books VALUES (16,'Базы данных','Малыхина М.П.', 2006, 326.00, 2, 3);

INSERT INTO books VALUES (17,'Базы данных. Разработка приложений','Рудикова Л.В.',

2006, 189.00, 6, 3);

INSERT INTO books VALUES (18,'Раскрытие тайн SQL','Оппель Э.', 2007, 200.00, 3, 3); INSERT INTO books VALUES (19,'Практикум по Access','Золотова С.И.', 2007, 87.00, 6, 3); INSERT INTO books VALUES (20,'Компьютерные сети','Танненбаум Э.', 2007, 630.00, 6, 4);

INSERT INTO books VALUES (21,'Сети. Поиск неисправностей','Бигелоу С.', 2005, 434.00,

4, 4);

INSERT INTO books VALUES (22,'Безопасность сетей','Брегг Р.', 2006, 462.00, 5, 4);

INSERT INTO books VALUES (23,'Анализ и диагностика компьютерных сетей', 'Хогдал

Дж.', 2001, 344.00, 3, 4);

INSERT INTO books VALUES (24,'Локальные вычислительные сети', 'Епанешников А.' ,

2005, 82.00, 8, 4);

INSERT INTO books VALUES (25,'Цифровая фотография','Надеждин Н.', 2004, 149.00,

```
20,5);
        INSERT INTO books VALUES (26,'Музыкальный компьютер для
        гитариста', 'Петелин
Р.Ю.', 2004, 217.00, 15, 5);
        INSERT INTO books VALUES (27,'Видео на ПК','Федорова
        А.',2003,231.00,10,5);
        INSERT INTO books VALUES (28,'Мультипликация во
        Flash','Киркпатрик Г.', 2006, 211.00,
20, 5);
        INSERT INTO books VALUES (29,'Запись CD и DVD','Гультяев А.К.',
        2003, 167.00, 12, 5);
        INSERT INTO books VALUES (30,'Запись и обработка звука на
        компьютере', 'Лоянич А.А.',
2008, 51.00, 8, 5);
        DELETE FROM users;
       INSERT INTO users VALUES (1,'Александр','Валерьевич','Иванов','58-
98-78',
'ivanov@email.ru', 'active');
        INSERT INTO users VALUES (2,'Сергей','Иванович','Лосев','90-57-77',
        'losev@email.ru',
'passive');
       INSERT INTO users VALUES (3,'Игорь','Николаевич','Симонов','95-
66-61',
'simonov@email.ru', 'active');
        INSERT INTO users VALUES (4,'Максим','Петрович','Кузнецов',NULL,
        'kuznetsov@email.ru',
'active');
        INSERT INTO users VALUES (5,'Анатолий','Юрьевич','Петров', NULL,
        NULL, 'lock');
       INSERT  INTO  users VALUES
(6,'Александр','Александрович','Корнеев','89-78-36',
'korneev@email.ru', 'gold');
        DELETE FROM
        orders;
        INSERT INTO orders VALUES (1,3,8,'2009-01-04 10:39:38',1);
        INSERT INTO orders VALUES (2,6,10,'2009-02-10 09:40:29',2);
        INSERT INTO orders VALUES (3,1,20,'2009-02-18 13:41:05',4);
        INSERT INTO orders VALUES (4,4,20,'2009-03-10
        18:20:00',1); INSERT INTO orders VALUES
        (5,3,20,'2009-03-17 19:15:36',1);
```

Практическая работа №16 «Создание и модификация таблиц БД. Выборка данных из

БД. Модификация содержимого БД»

Цель работы: овладение практическими навыками по модификации таблиц БД, создание запросов к БД

Создание простых запросов на выборку

Для выполнения запросов (извлечения строк из одной или нескольких таблиц БД) используется оператор *SELECT*. Результатом запроса всегда является таблица. Результаты запроса могут быть использованы для создания новой таблицы. Таблица, полученная в результате запроса, может стать предметом дальнейших запросов. Общая форма оператора *SELECT*:

SELECT столбцы FROM таблицы
[WHERE условия]
[GROUP BY группа [HAVING групповые_условия]]
[ORDER BY имя_поля]
[LIMIT пределы];

Оператор *SELECT* имеет много опций. Их можно использовать или не использовать, но они должны указываться в том порядке, в каком они приведены. Если требуется вывести все столбцы таблицы, необязательно перечислять их после ключевого слова *SELECT*, достаточно заменить этот список символом *.

```
mysql> SELECT * FROM orders;
+----------+----------+----------+---------------------+----------+
| order_ID | o_user_ID | o_book_ID | o_time              | o_number |
+----------+----------+----------+---------------------+----------+
|        1 |        3 |        8 | 2009-01-04 10:39:38 |        1 |
|        2 |        6 |       10 | 2009-02-10 09:40:29 |        2 |
|        3 |        1 |       20 | 2009-02-18 13:41:05 |        4 |
|        4 |        4 |       20 | 2009-03-10 18:20:00 |        1 |
|        5 |        3 |       20 | 2009-03-17 19:15:36 |        1 |
+----------+----------+----------+---------------------+----------+
5 rows in set (0.02 sec)
```

Список столбцов в операторе *SELECT* используют, если нужно изменить порядок следования столбцов в результирующей таблице или выбрать часть столбцов.

```
mysql> SELECT cat_name, cat_ID FROM catalogs;
+-------------------+--------+
| cat_name          | cat_ID |
+-------------------+--------+
| Программирование  |      1 |
| Интернет          |      2 |
| Базы данных       |      3 |
| Сети              |      4 |
| Мультимедиа       |      5 |
+-------------------+--------+
5 rows in set (0.01 sec)
```

Условия выборки. Гораздо чаще встречается ситуация, когда необходимо изменить количество выводимых строк. Для выбора записей, удовлетворяющих определенным критериям поиска, можно использовать конструкцию *WHERE*.

```
mysql> SELECT user_ID, u_surname FROM users
    -> WHERE u_status='active';
+---------+-----------+
| user_ID | u_surname |
+---------+-----------+
|       1 | Иванов    |
|       3 | Симонов   |
|       4 | Кузнецов  |
+---------+-----------+
3 rows in set (0.03 sec)
```

В запросе можно использовать ключевое слово *DISTINCT*, чтобы результат не содержал повторений уже имеющихся значений, например:

```
mysql> SELECT DISTINCT u_status FROM users;
+----------+
| u_status |
+----------+
| active   |
| passive  |
| lock     |
| gold     |
+----------+
4 rows in set (0.01 sec)
```

Сортировка. Результат выборки – записи, расположенные в том порядке, в котором они хранятся в БД. Чтобы отсортировать значения по одному из столбцов, необходимо после конструкции *ORDER BY* указать этот столбец, например:

```
mysql> SELECT * FROM orders ORDER BY o_user_ID;
+----------+----------+----------+---------------------+----------+
| order_ID | o_user_ID | o_book_ID | o_time              | o_number |
+----------+----------+----------+---------------------+----------+
|        3 |        1 |       20 | 2009-02-18 13:41:05 |        4 |
|        1 |        3 |        8 | 2009-01-04 10:39:38 |        1 |
|        5 |        3 |       20 | 2009-03-17 19:15:36 |        1 |
|        4 |        4 |       20 | 2009-03-10 18:20:00 |        1 |
|        2 |        6 |       10 | 2009-02-10 09:40:29 |        2 |
+----------+----------+----------+---------------------+----------+
5 rows in set (0.00 sec)
```

Сортировку записей можно производить по нескольким столбцам (их следует указать после слов *ORDER BY* через запятую). Число столбцов, указываемых в конструкции *ORDER BY*, не ограничено.

По умолчанию сортировка производится в прямом порядке (записи располагаются от наименьшего значения поля сортировки до наибольшего). Обратный порядок сортировки реализуется с помощью ключевого слова *DESC*:

```
mysql> SELECT o_time FROM orders ORDER BY o_time DESC;
+---------------------+
| o_time              |
+---------------------+
| 2009-03-17 19:15:36 |
| 2009-03-10 18:20:00 |
| 2009-02-18 13:41:05 |
| 2009-02-10 09:40:29 |
| 2009-01-04 10:39:38 |
+---------------------+
5 rows in set (0.27 sec)
```

Для прямой сортировки существует ключевое слово *ASC*, но так как записи сортируются в прямом порядке по умолчанию, данное ключевое слово опускают.

Ограничение выборки. Результат выборки может содержать тысячи записей, вывод и обработка которых занимают значительное время. Поэтому информацию часто разбивают на страницы и предоставляют ее пользователю частями. Постраничная навигация используется при помощи ключевого слова *LIMIT*, за которым следует число выводимых записей. Следующий запрос извлекает первые 5 записей, при этом осуществляется обратная сортировка по полю *b_count*:

```
mysql> SELECT book_ID, b_count FROM books
    -> ORDER BY b_count DESC
    -> LIMIT 5;
+---------+---------+
| book_ID | b_count |
+---------+---------+
|      28 |      20 |
|      25 |      20 |
|      26 |      15 |
|      29 |      12 |
|       9 |      12 |
+---------+---------+
5 rows in set (0.03 sec)
```

Для извлечения следующих пяти записей используется ключевое слово *LIMIT* с двумя цифрами. Первая указывает позицию, начиная с которой необходимо вернуть результат, вторая цифра – число извлекаемых записей, например:

```
mysql> SELECT book_ID, b_count FROM books
    -> ORDER BY b_count DESC
    -> LIMIT 5,5;
+---------+---------+
| book_ID | b_count |
+---------+---------+
|       1 |      10 |
|      27 |      10 |
|      24 |       8 |
|      30 |       8 |
|      28 |       6 |
+---------+---------+
5 rows in set (0.00 sec)
```

При определении смещения нумерация строк начинается с нуля (поэтому в последнем примере для шестой строки указано смещение 5).

Группировка записей. Конструкция *GROUP BY* позволяет группировать извлекаемые строки. Она полезна в комбинации с функциями, применяемыми к группам строк. Эти функции (табл. 16.1) называются агрегатами (суммирующими функциями) и вычисляют одно значение для каждой группы, создаваемой конструкцией *GROUP BY*. Функции позволяют узнать число строк в группе, подсчитать среднее значение, получить сумму значений столбцов. Результирующее значение рассчитывается для значений, не равных *NULL* (исключение – функция *COUNT(*)*). Допустимо использование этих функций в запросах без группировки (вся выборка – одна группа).

Пример использования функции *COUNT( )*, которая возвращает число строк в таблице, значения указанного столбца для которых отличны от *NULL*:

```
mysql> SELECT COUNT(book_ID) FROM books;
+----------------+
| COUNT(book_ID) |
+----------------+
|             30 |
+----------------+
1 row in set (0.16 sec)
```

Таблица 16.1. Агрегатные функции

Обозначение	Описание
AVG ([DISTINCT] expr)	Возвращает среднее значение аргумента *expr*. В качестве аргумента обычно выступает имя столбца. Необязательное слово *DISTINCT* позволяет обрабатывать только уникальные значения столбца *expr*
COUNT ( )	Подсчитывает число записей и имеет несколько форм. Форма *COUNT (выражение)* возвращает число записей в таблице, поле *выражение* для которых не равно *NULL*. Форма *COUNT(*)* возвращает общее число строк в таблице независимо от того, принимает какое-либо поле значение *NULL* или нет. Форма *COUNT (DISTINCT выражение1, выражение2, ...)* позволяет использовать ключевое слово *DISTINCT*, которое позволяет подсчитать только уникальные значения столбца

MIN ([DISTINCT] expr)	Возвращает минимальное значение среди всех непустых значений выбранных строк в столбце *expr*. Необязательное слово *DISTINCT* позволяет обрабатывать только уникальные значения столбца *expr*
MAX ([DISTINCT] expr)	Возвращает максимальное значение среди всех непустых значений выбранных строк в столбце *expr*. Необязательное слово *DISTINCT* позволяет обрабатывать только уникальные значения столбца *expr*
STD (expr)	Возвращает стандартное среднеквадратичное отклонение в аргументе *expr*
STDDEV_SAMP (expr)	Возвращает выборочное среднеквадратичное отклонение в аргументе *expr*
SUM ([DISTINCT] expr)	Возвращает сумму величин в столбце *expr*. Необязательное слово *DISTINCT* позволяет обрабатывать только уникальные значения столбца *expr*

Использование ключевого слова *DISTINCT* с функцией *COUNT( )* позволяет вернуть число уникальных значений *b_cat_ID* в таблице *books*, например:

```
mysql> SELECT COUNT(DISTINCT b_cat_ID) FROM books;
+----------------------------+
| COUNT(DISTINCT b_cat_ID) |
+----------------------------+
|                          5 |
+----------------------------+
1 row in set (0.02 sec)
```

В *SELECT*-запросе столбцу можно назначить новое имя с помощью оператора *AS*. Например, результату функции *COUNT( )* присваивается псевдоним *total*:

```
mysql> SELECT COUNT(order_ID) AS total FROM orders;
+--------+
| total |
+--------+
|     5 |
+--------+
1 row in set (0.05 sec)
```

Использование функций в конструкции *WHERE* приведет к ошибке. В следующем примере показана попытка извлечения из таблицы *catalogs* записи с максимальным значением поля *cat_ID*:

```
mysql> SELECT * FROM catalogs WHERE cat_ID=MAX(cat_ID);
ERROR 1111 (HY000): Invalid use of group function
```

Решение задачи следует искать в использовании конструкции *ORDER BY*:

```
mysql> SELECT * FROM catalogs ORDER BY cat_ID DESC LIMIT 1;
+--------+--------------+
| cat_ID | cat_name     |
+--------+--------------+
|      5 | Мультимедиа |
+--------+--------------+
1 row in set (0.00 sec)
```

Для извлечения уникальных записей используют конструкцию *GROUP BY* с именем столбца, по которому группируется результат:

```
mysql> SELECT b_cat_ID FROM books
    -> GROUP BY b_cat_ID ORDER BY b_cat_ID;
+-----------+
| b_cat_ID |
+-----------+
|        1 |
|        2 |
|        3 |
|        4 |
|        5 |
+-----------+
5 rows in set (0.03 sec)
```

При использовании *GROUP BY* возможно использование условия *WHERE*:

```
mysql> SELECT b_cat_ID, COUNT(b_cat_ID) FROM books
    -> WHERE b_cat_ID > 2
    -> GROUP BY b_cat_ID
    -> ORDER BY b_cat_ID;
+-----------+-----------------+
| b_cat_ID | COUNT(b_cat_ID) |
+-----------+-----------------+
|        3 |               4 |
|        4 |               5 |
|        5 |               6 |
+-----------+-----------------+
3 rows in set (0.02 sec)
```

Часто при задании условий требуется ограничить выборку по результату функции (например, выбрать каталоги, где число товарных позиций больше 5). Использование для этих целей конструкции *WHERE* приводит к ошибке. Для решения этой проблемы вместо ключевого слова *WHERE* используется ключевое слово *HAVING*, располагающееся за конструкцией *GROUP BY*:

```
mysql> SELECT b_cat_ID, COUNT(b_cat_ID) AS total FROM books
    -> GROUP BY b_cat_ID
    -> HAVING total > 5
    -> ORDER BY b_cat_ID;
+-----------+-------+
| b_cat_ID | total |
+-----------+-------+
|        1 |     9 |
|        2 |     6 |
|        5 |     6 |
+-----------+-------+
3 rows in set (0.00 sec)
```

Запрос, извлекающий уникальные значения столбца *b_cat_ID*, большие двух:

```
mysql> SELECT b_cat_ID, COUNT(b_cat_ID) FROM books
    -> GROUP BY b_cat_ID
    -> HAVING b_cat_ID > 2
    -> ORDER BY b_cat_ID;
+-----------+-----------------+
| b_cat_ID | COUNT(b_cat_ID) |
+-----------+-----------------+
|        3 |               4 |
|        4 |               5 |
|        5 |               6 |
+-----------+-----------------+
3 rows in set (0.00 sec)
```

При этом в случае использования ключевого слова *WHERE* сначала производится выборка из таблицы с применением условия и лишь затем группировка результата, а в случае использования ключевого слова *HAVING* сначала происходит группировка таблицы и лишь затем выборка с применением условия. Допускается использование условия *HAVING* без группировки
GROUP BY.

Использование функций. Для решения специфических задач при выборке удобны встроенные функции MySQL. Большинство функций предназначено для использования в выражениях *SELECT* и *WHERE*. Существуют также специальные функции группировки для использования в выражении *GROUP BY* (см. выше).

Каждая функция имеет уникальное имя и может иметь несколько аргументов (перечисляются через запятую в круглых скобках). Если

аргументы отсутствуют, круглые скобки все равно следует указывать. Пробелы между именем функции и круглыми скобками недопустимы.

Число доступных для использования функций велико, в приложениях приведены наиболее полезные из них.

Пример использования функции, возвращающей версию сервера MySQL:

```
mysql> SELECT VERSION();
+-----------------------+
| VERSION()             |
+-----------------------+
| 5.0.51b-community-nt  |
+-----------------------+
1 row in set (0.00 sec)
```

Отметим также возможность использования оператора *SELECT* без таблиц вообще. В такой форме *SELECT* можно использовать как калькулятор:

```
mysql> SELECT 2+3;
+-----+
| 2+3 |
+-----+
|   5 |
+-----+
1 row in set (0.00 sec)
```

Можно вычислить любое выражение без указания таблиц, получив доступ ко всему разнообразию математических и других операторов и функций. Возможность выполнять математические расчеты на уровне *SELECT* позволяет проводить финансовый анализ значений таблиц и отображать полученные результаты в отчетах. Во всех выражениях MySQL (как в любом языке программирования) можно использовать скобки, чтобы контролировать порядок вычислений.

Операторы. Под операторами подразумеваются конструкции языка, которые производят преобразование данных. Данные, над которыми совершается операция, называются операндами.

В MySQL используются три типа операторов:

- арифметические операторы;
- операторы сравнения;
- логические операторы.

Арифметические операции. В MySQL используются обычные арифметические операции:

сложение (+), вычитание (–), умножение (*), деление (/) и целочисленное деление *DIV* (деление и отсечение дробной части). Деление на 0 дает безопасный результат *NULL*.

Операторы сравнения. При работе с операторами сравнения необходимо помнить о том, что, за исключением нескольких особо оговариваемых случаев, сравнение чего-либо со значением *NULL* дает в результате *NULL*. Это касается и сравнения значения *NULL* со значением *NULL*:

```
mysql> SELECT NULL=NULL;
+-----------+
| NULL=NULL |
+-----------+
|      NULL |
+-----------+
1 row in set (0.02 sec)
```

Корректнее использовать следующий запрос:

```
mysql> SELECT NULL IS NULL;
+--------------+
| NULL IS NULL |
+--------------+
|            1 |
+--------------+
1 row in set (0.00 sec)
```

Поэтому следует быть предельно внимательными при работе с операторами сравнения, если операнды могут принимать значения *NULL*.

Наиболее часто используемые операторы сравнения приведены в табл. 16.2.

Логические операторы. MySQL поддерживает все обычные логические операции, которые можно использовать в выражениях. Логические выражения в MySQL могут принимать значения 1 (истина), 0 (ложь) или *NULL*.

Кроме того, следует учитывать, что MySQL интерпретирует любое ненулевое значение, отличное от *NULL*, как значение «истина». Основные логические операторы приведены в табл. 8.

Таблица 16.2. Наиболее часто используемые операторы

Оператор	**Значение**
=	Оператор равенства. Возвращает 1 (истина), если операнды равны, и 0 (ложь), если не равны
<=>	Оператор эквивалентности. Аналогичен обычному равенству, но возвращает только два значения: 1 (истина) и 0 (ложь). *NULL* не возвращает
<>	Оператор неравенства. Возвращает 1 (истина), если операнды не равны, и 0 (ложь), если равны
<	Оператор «меньше». Возвращает 1 (истина), если левый операнд меньше правого, и 0 (ложь) – в противном случае
<=	Оператор «меньше или равно». Возвращает 1 (истина), если левый операнд меньше правого или они равны, и 0 (ложь) – в противном случае
>	Оператор «больше». Возвращает 1 (истина), если левый операнд больше правого, и 0 (ложь) – в противном случае
>=	Оператор «больше или равно». Возвращает 1 (истина), если левый операнд больше правого или они равны, и 0 (ложь) – в противном случае
n BETWEEN min AND max	Проверка диапазона. Возвращает 1 (истина), если проверяемое значение *n* находится между *min* и *max*, и 0 (ложь) – в противном случае
IS NULL и *IS NOT NULL*	Позволяют проверить, является ли значение значением *NULL* или нет
n IN (множество)	Принадлежность к множеству. Возвращает 1 (истина), если проверяемое значение *n* входит в список, и 0 (ложь) – в противном случае. В качестве множества может

	использоваться список литеральных значений или выражений или подзапрос

Таблица 16.3. Логические операторы

Оператор	Пример	Значение
AND n	*AND m*	Логическое *И*: истина *AND* истина = истина, ложь *AND* любое = ложь. Все остальные выражения оцениваются как *NULL*
OR	*n OR m*	Логическое *ИЛИ*: истина *OR* любое = истина, *NULL OR* ложь = *NULL*, *NULL OR NULL = NULL,* ложь *OR* ложь = ложь
NOT	*NOT n*	Логическое НЕТ: *NOT* истина = ложь, *NOT* ложь = истина. *NOT NULL = NULL*
XOR	*n XOR m*	Логическое *исключающее ИЛИ*: истина *XOR* истина = ложь, истина *XOR* ложь = истина, ложь *XOR* истина = истина, ложь *XOR* ложь = ложь, *NULL XOR* любое = *NULL*, любое *XOR NULL = NULL*

Переменные SQL и временные таблицы. Часто результаты запроса необходимо использовать в последующих запросах. Для этого полученные данные необходимо сохранить во временных структурах. Эту задачу решают переменные SQL и временные таблицы. Объявление переменной начинается с символа @, за которым следует имя переменной. Значения переменным присваиваются посредством оператора *SELECT* С использованием оператора присваивания := . Например:

```
mysql> SELECT @total := COUNT(*) FROM books;
+----------------------+
| @total := COUNT(*) |
+----------------------+
|                   30 |
+----------------------+
1 row in set (0.42 sec)

mysql> SELECT @total;
+---------+
| @total |
+---------+
| 30     |
+---------+
1 row in set (0.02 sec)
```

Объявляется переменная *@total*, которой присваивается число записей в таблице *books*. Затем в рамках текущего сеанса в последующих запросах появляется возможность использования данной переменной. Переменная действует только в рамках одного сеанса соединения с сервером MySQL и прекращает свое существование после разрыва соединения.

Переменные также могут объявляться при помощи оператора *SET*:

```
mysql> SET @last=CURDATE()-INTERVAL 7 DAY;
Query OK, 0 rows affected (0.03 sec)

mysql> SELECT CURDATE(), @last;
+------------+------------+
| CURDATE()  | @last      |
+------------+------------+
| 2009-12-22 | 2009-12-15 |
+------------+------------+
1 row in set (0.00 sec)
```

При использовании оператора *SET* в качестве оператора присваивания может выступать обычный знак равенства =. Оператор *SET* удобен тем, что он не возвращает результирующую таблицу. Не рекомендуется одновременно присваивать переменной некоторое значение и использовать эту переменную в одном запросе.

Переменная SQL позволяет сохранить одно промежуточное значение. Когда необходимо сохранить результирующую таблицу, прибегают к временным таблицам. Создание временных таблиц осуществляется при помощи оператора *CREATE TEMPORARY TABLE,* синтаксис которого ничем не отличается от синтаксиса оператора *CREATE TABLE*.

Временная таблица автоматически удаляется по завершении соединения с сервером, а ее имя действительно только в течение данного соединения. Это означает, что два разных клиента могут использовать временные таблицы с одинаковыми именами без конфликта друг с другом или с существующей таблицей с тем же именем. **Задание для практической работы**

1. Создайте простой запрос на выборку к таблице *books*, который выводит максимальную и минимальную цены товарных позиций, присваивая им соответственно псевдонимы *maximum* и *minimum*:

```
mysql> SELECT MAX(b_price) AS maximum, MIN(b_price) AS minimum
    -> FROM books;
+---------+---------+
| maximum | minimum |
+---------+---------+
|  771.00 |   42.00 |
+---------+---------+
1 row in set (0.00 sec)
```

2. Создавайте простой запрос на выборку к таблице *books*, который выводит количество записей, соответствующих каждому из уникальных значений *b_cat_ID*. Для этого используем функцию *COUNT( )* вместе с выражением *GROUP BY*:

```
mysql> SELECT b_cat_ID, COUNT(b_cat_ID) FROM books
    -> GROUP BY b_cat_ID ORDER BY b_cat_ID;
+----------+-----------------+
| b_cat_ID | COUNT(b_cat_ID) |
+----------+-----------------+
|        1 |               9 |
|        2 |               6 |
|        3 |               4 |
|        4 |               5 |
|        5 |               6 |
+----------+-----------------+
5 rows in set (0.00 sec)
```

3. Создавайте многотабличный запрос на выборку, который выводит фамилии, имена и отчества покупателей магазина, сделавших менее двух покупок:

```
mysql> SELECT users.u_surname,users.u_name,users.u_patronymic,
    -> COUNT(orders.order_ID) AS total
    -> FROM users LEFT JOIN orders ON users.user_ID=orders.o_user_ID
    -> GROUP BY users.user_ID
    -> HAVING total<2
    -> ORDER BY total DESC;
+-----------+-----------+---------------+-------+
| u_surname | u_name    | u_patronymic  | total |
+-----------+-----------+---------------+-------+
| Иванов    | Александр | Валерьевич    |     1 |
| Корнеев   | Александр | Александрович |     1 |
| Кузнецов  | Максим    | Петрович      |     1 |
| Петров    | Анатолий  | Юрьевич       |     0 |
| Лосев     | Сергей    | Иванович      |     0 |
+-----------+-----------+---------------+-------+
5 rows in set (0.02 sec)
```

4. Создайте запрос на выборку с вложенным запросом, выводящим перечень книг, которые не заказывались покупателями:

```
mysql> SELECT book_ID, b_name, b_price FROM books
    -> WHERE NOT EXISTS (SELECT * FROM orders
    -> WHERE orders.o_book_ID=books.book_ID);
+---------+-----------------------------------------+---------+
| book_ID | b_name                                  | b_price |
+---------+-----------------------------------------+---------+
|       1 | JavaScript в кармане                    |   42.00 |
|       2 | Visual FoxPro 9.0                       |  660.00 |
|       3 | C++ Как он есть                         |  218.00 |
|       4 | Создание приложений с помощью C#        |  169.00 |
|       5 | Delphi. Народные советы                 |  243.00 |
|       6 | Delphi. Полное руководство              |  500.00 |
|       7 | Профессиональное программирование на PHP |  309.00 |
|       9 | Практика программирования               |  214.00 |
|      11 | Поиск в Internet                        |  107.00 |
|      12 | Web-конструирование                     |  177.00 |
|      13 | Самоучитель Интернет                    |  121.00 |
|      14 | Популярные интернет-браузеры            |   82.00 |
|      15 | Общение в Интернете                     |   85.00 |
|      16 | Базы данных                             |  326.00 |
|      17 | Базы данных. Разработка приложений      |  189.00 |
|      18 | Раскрытие тайн SQL                      |  200.00 |
|      19 | Практикум по Access                     |   87.00 |
|      21 | Сети. Поиск неисправностей              |  434.00 |
|      22 | Безопасность сетей                      |  462.00 |
|      23 | Анализ и диагностика компьютерных сетей |  344.00 |
|      24 | Локальные вычислительные сети           |   82.00 |
|      25 | Цифровая фотография                     |  149.00 |
|      26 | Музыкальный компьютер для гитариста     |  217.00 |
|      27 | Видео на ПК                             |  231.00 |
|      28 | Мультипликация во Flash                 |  211.00 |
|      29 | Запись CD и DVD                         |  167.00 |
|      30 | Запись и обработка звука на компьютере  |   51.00 |
+---------+-----------------------------------------+---------+
27 rows in set (0.03 sec)
```

Практическая работа №17 «Обработка транзакций. Использование функций защиты для БД»

Цель работы: получение практических навыков обработки транзакций и защиты баз данных при помощи функций.

Обработка транзакций

Изменения БД часто требуют выполнения нескольких запросов, например при покупке в электронном магазине требуется добавить запись в таблицу заказов и уменьшить число товарных позиций на складе. В промышленных БД одно событие может затрагивать большее число таблиц и требовать многочисленных запросов.

Если на этапе выполнения одного из запросов происходит сбой, это может нарушить целостность БД (товар может быть продан, а число товарных позиций на складе не обновлено). Чтобы сохранить целостность БД, все изменения должны выполняться как единое целое. Либо все изменения успешно выполняются, либо, в случае сбоя, БД принимает состояние, которое было до начала изменений. Это обеспечивается средствами обработки транзакций.

Транзакция – последовательность операторов SQL, выполняющихся как единая операция, которая не прерывается другими клиентами. Пока

происходит работа с записями таблицы (обновление или удаление), никто другой не может получить доступ к этим записям, т. к. MySQL автоматически блокирует доступ к ним.

Таблицы *ISAM, MyISAM* и *HEAP* не поддерживают транзакции. В настоящий момент их поддержка осуществляется только в таблицах *BDB* и *InnoDB*.

Транзакции позволяют объединять операторы в группу и гарантировать, что все операторы группы будут выполнены успешно. Если часть транзакции выполняется со сбоем, результаты выполнения всех операторов транзакции до места сбоя отменяются, приводя БД к виду, в котором она была до выполнения транзакции.

По умолчанию MySQL работает в режиме автоматического завершения транзакций, т. е. как только выполняется оператор обновления данных, который модифицирует таблицу, изменения тут же сохраняются на диске. Чтобы объединить операторы в транзакцию, следует отключить этот режим: *SET AUTOCOMMIT=0;*

После отключения режима для завершения транзакции необходимо ввести оператор *COMMIT*, для отката – *ROLLBACK*.

Включить режим автоматического завершения транзакций для отдельной последовательности операторов можно оператором *START TRANSACTION*.

Для таблиц *InnoDB* есть операторы *SAVEPOINT* и *ROLLBACK TO SAVEPOINT*, которые позволяют работать с именованными точками начала транзакции.

```
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO catalogs VALUES(NULL,'Периферия');
Query OK, 1 row affected (0.00 sec)

mysql> SAVEPOINT point1;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO catalogs VALUES(NULL,'Разное');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM catalogs;
+--------+-------------------+
| cat_ID | cat_name          |
+--------+-------------------+
|      1 | Программирование  |
|      2 | Интернет          |
|      3 | Базы данных       |
|      4 | Сети              |
|      5 | Мультимедиа       |
|     12 | Периферия         |
|     13 | Разное            |
+--------+-------------------+
7 rows in set (0.00 sec)

mysql> ROLLBACK TO SAVEPOINT point1;
Query OK, 0 rows affected (0.02 sec)
```

```
mysql> SELECT * FROM catalogs;
+--------+--------------------+
| cat_ID | cat_name           |
+--------+--------------------+
|      1 | Программирование   |
|      2 | Интернет           |
|      3 | Базы данных        |
|      4 | Сети               |
|      5 | Мультимедиа        |
|     12 | Периферия          |
+--------+--------------------+
6 rows in set (0.00 sec)
```

Оператор *SAVEPOINT* устанавливает именованную точку начала транзакции с именем *point1*. Оператор *ROLLBACK TO SAVEPOINT point1* откатывает транзакцию к состоянию, в котором находилась БД на момент установки именованной точки. Все точки сохранения транзакций удаляются, если выполняются операторы *COMMIT* или *ROLLBACK* без указания имени точки сохранения.

Управление правами пользователей

СУБД MySQL является многопользовательской средой, поэтому для доступа к таблицам БД могут быть созданы различные учетные записи с разным уровнем привилегий. Учетной записи редактора можно предоставить привилегии на просмотр таблицы, добавление новых записей и обновление уже существующих. Администратору БД можно предоставить более широкие полномочия (возможность создания таблиц, редактирования и удаления уже существующих). Для пользователя БД достаточно лишь просмотра таблиц.

Рассмотрим следующие вопросы:

□ создание, редактирование и удаление учетных записей пользователей; □ назначение и отмена привилегий.

Учетная запись является составной и принимает форму *'username' @ 'host'*, где *username* – имя пользователя, а *host* – наименование хоста, с которого пользователь может обращаться к серверу. Например, записи *'root' @ '127.0.0.1'* и *'wet' @ '62.78.56.34'* означают, что пользователь с именем *root* может обращаться с хоста, на котором расположен сервер, а *wet* – только с хоста с IPадресом 62.78.56.34.

IP-адрес 127.0.0.1 всегда относится к локальному хосту. Если сервер и клиент установлены на одном хосте, то сервер слушает соединения по этому адресу, а клиент отправляет на него SQLзапросы.

IP-адрес 127.0.0.1 имеет псевдоним *localhost*, поэтому учетные записи вида *'root' @*
'127.0.0.1' можно записывать в виде *'root' @ 'localhost'*.

Число адресов, с которых необходимо обеспечить доступ пользователю, может быть значительным. Для задания диапазона в имени хоста используется специальный символ "%". Так, учетная запись *'wet' @ '%'* позволяет пользователю *wet* обращаться к серверу MySQL с любых компьютеров сети.

Все учетные записи хранятся в таблице *user* системной базы данных с именем *mysql*. После первой инсталляции содержимое таблицы *user* выглядит так, как показано в листинге.

```
mysql> SELECT Host,User,Password FROM mysql.user;
+-------------------------+------+----------+
| Host                    | User | Password |
+-------------------------+------+----------+
| localhost               | root |          |
| production.mysql.com    | root |          |
| 127.0.0.1               | root |          |
| localhost               |      |          |
| production.mysql.com    |      |          |
+-------------------------+------+----------+
5 rows in set (0.27 sec)
```

Создание новой учетной записи. Создать учетную запись позволяет оператор

CREATE USER 'username' @ 'host' [IDENTIFIED BY [PASSWORD] 'пароль'];

Оператор создает новую учетную запись с необязательным паролем. Если пароль не указан, в его качестве выступает пустая строка. Разумно хранить пароль в виде хэш-кода, полученного в результате необратимого шифрования. Чтобы воспользоваться этим механизмом авторизации, необходимо поместить между ключевым словом *IDENTIFIED BY* и паролем ключевое слово *PASSWORD*.

Удаление учетной записи. Удалить учетную запись позволяет оператор

DROP USER 'username' @ 'host';

Изменение имени пользователя в учетной записи. Осуществляется с помощью оператора

RENAME USER старое_имя TO новое_имя;

Назначение привилегий. Рассмотренные выше операторы позволяют создавать, удалять и редактировать учетные записи, но они не позволяют изменять привилегии пользователя – сообщать MySQL, какой пользователь имеет право только на чтение информации, какой на чтение и редактирование, а кому предоставлены права изменять структуру БД и создавать учетные записи.

Для решения этих задач предназначены операторы *GRANT* (назначает привилегии) и *REVOKE* (удаляет привилегии). Если учетной записи, которая показана в операторе *GRANT*, не существует, то она автоматически создается. Удаление всех привилегий с помощью оператора *REVOKE* не приводит к автоматическому уничтожению учетной записи. Для удаления пользователя необходимо воспользоваться оператором *DROP USER*.

В простейшем случае оператор *GRANT* выглядит следующим образом:

```
mysql> GRANT ALL ON *.* TO 'wet'@'localhost' IDENTIFIED BY 'pass';
Query OK, 0 rows affected (0.17 sec)
```

Данный запрос создает пользователя с именем *wet* и паролем *pass*, который может обращаться к серверу с локального хоста (*localhost*) и имеет все права (*ALL*) для всех баз данных (*.*). Если такой пользователь существует, то его привилегии будут изменены на *ALL*.

Вместо ключевого слова *ALL* можно использовать любое из ключевых слов, представленных в табл. 17.1. Ключевое слово *ON* в операторе *GRANT* задает уровень привилегий, которые могут быть заданы на одном из четырех уровней, представленных в табл. 10. Для таблиц можно установить только следующие типы привилегий: *SELECT, INSERT, UPDATE, DELETE, CREATE,*

DROP, GRANT OPTION, INDEX и *ALTER*. Это следует учитывать при использовании конструкции *GRANT ALL*, которая назначает привилегии на текущем уровне. Так, запрос уровня базы данных *GRANT ALL ON db.** не предоставляет никаких глобальных привилегий.

Отмена привилегий. Для отмены привилегий используется оператор *REVOKE*:

```
mysql> REVOKE DELETE, UPDATE ON *.* FROM 'wet'@'localhost';
Query OK, 0 rows affected (0.02 sec)
```

Оператор *REVOKE* отменяет привилегии, но не удаляет учетные записи, для их удаления необходимо воспользоваться оператором *DROP USER*.

Таблица 17.1. Вместо ключевого слова ALL

Привилегия	Операция, разрешенная привилегией
ALL [PRIVILEGES]	Комбинация всех привилегий, за исключением привилегии *GRANT OPTION*, которая задается отдельно
ALTER	Позволяет редактировать таблицу с помощью оператора *ALTER TABLE*
ALTER ROUTINE	Позволяет редактировать или удалять хранимую процедуру
CREATE	Позволяет создавать таблицу при помощи оператора *CREATE TABLE*
CREATE ROUTINE	Позволяет создавать хранимую процедуру
CREATE TEMPORARY TABLES	Позволяет создавать временные таблицы
CREATE USER	Позволяет работать с учетными записями с помощью *CREATE USER, DROP USER, RENAME USER* и *REVOKE ALL PRIVILEGES*
CREATE VIEW	Позволяет создавать представление с помощью оператора *CREATE VIEW*
DELETE	Позволяет удалять записи при помощи оператора *DELETE*
DROP	Позволяет удалять таблицы при помощи оператора *DROP TABLE*
EXECUTE	Позволяет выполнять хранимые процедуры
INDEX	Позволяет работать с индексами, в частности, использовать операторы *CREATE INDEX* и *DROP INDEX*

INSERT	Позволяет добавлять в таблицу новые записи оператором *INSERT*
LOCK TABLES	Позволяет осуществлять блокировки таблиц при помощи операторов *LOCK TABLES* и *UNLOCK TABLES*. Для вступления в действие этой привилегии должна быть установлена привилегия *SELECT*
SELECT	Позволяет осуществлять выборки таблиц оператором *SELECT*
SHOW DATABASES	Позволяет просматривать список всех таблиц на сервере при помощи оператора *SHOW DATABASES*
SHOW VIEW	Позволяет использовать оператор *SHOW CREATE VIEW*
UPDATE	Позволяет обновлять содержимое таблиц оператором *UPDATE*
USAGE	Синоним для статуса «отсутствуют привилегии»
GRANT OPTION	Позволяет управлять привилегиями других пользователей, без данной привилегии невозможно выполнить операторы *GRANT* И *REVOKE*

Таблица 17.2. Вместо ключевого слова ON

Ключевое слово *ON*	**Уровень**
*ON *.**	Глобальный уровень – пользователь с полномочиями на глобальном уровне может обращаться ко всем БД и таблицам, входящим в их состав
*ON db.**	Уровень базы данных – привилегии распространяются на таблицы базы данных *db*
ON db.tbl	Уровень таблицы – привилегии распространяются на таблицу *tbl* базы данных *db*
ON db.tbl	Уровень столбца – привилегии касаются отдельных столбцов в таблице *tbl* базы данных *db*. Список столбцов указывается в скобках через запятую после ключевых слов *SELECT, INSERT, UPDATE*

Задание для практической работы

□ При выполнении практической работы необходимо: □ создать транзакцию, произвести ее откат и фиксацию; □ составить отчет по практической работе.

Задание 1. Обработка транзакций

Для выполнения задания объединим несколько операций по добавлению в таблицу *catalogs* новых каталогов, а затем произведем откат транзакции, т. е. отмену произведенных действий. Отключаем режим автоматического завершения, добавляем новые записи и проверяем, добавились записи или нет.

```
mysql> SET AUTOCOMMIT=0;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO catalogs VALUES(NULL,'Аппаратура');
Query OK, 1 row affected (0.06 sec)

mysql> INSERT INTO catalogs VALUES(NULL,'Безопасность');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM catalogs;
+--------+--------------------+
| cat_ID | cat_name           |
+--------+--------------------+
|      1 | Программирование   |
|      2 | Интернет           |
|      3 | Базы данных        |
|      4 | Сети               |
|      5 | Мультимедиа        |
|      6 | Аппаратура         |
|      7 | Безопасность       |
+--------+--------------------+
7 rows in set (0.02 sec)
```

Откатываем транзакцию оператором *ROLLBACK* (изменения не сохранились).

```
mysql> ROLLBACK;
Query OK, 0 rows affected (0.03 sec)

mysql> SELECT * FROM catalogs;
+--------+--------------------+
| cat_ID | cat_name           |
+--------+--------------------+
|      1 | Программирование   |
|      2 | Интернет           |
|      3 | Базы данных        |
|      4 | Сети               |
|      5 | Мультимедиа        |
+--------+--------------------+
5 rows in set (0.00 sec)
```

Воспроизведем транзакцию и сохраним действия оператором *COMMIT*.

```
mysql> INSERT INTO catalogs VALUES(NULL,'Аппаратура');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO catalogs VALUES(NULL,'Безопасность');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM catalogs;
+--------+--------------------+
| cat_ID | cat_name           |
+--------+--------------------+
|      1 | Программирование   |
|      2 | Интернет           |
|      3 | Базы данных        |
|      4 | Сети               |
|      5 | Мультимедиа        |
|      8 | Аппаратура         |
|      9 | Безопасность       |
+--------+--------------------+
7 rows in set (0.00 sec)

mysql> COMMIT;
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT * FROM catalogs;
+--------+--------------------+
| cat_ID | cat_name           |
+--------+--------------------+
|      1 | Программирование   |
|      2 | Интернет           |
|      3 | Базы данных        |
|      4 | Сети               |
|      5 | Мультимедиа        |
|      8 | Аппаратура         |
|      9 | Безопасность       |
+--------+--------------------+
7 rows in set (0.00 sec)
```

Задание 2. Защита данных

Для работы выберем два компьютера, подключенных к локальной сети. На одном необходимо развернуть сервер MySQL, на другой – скопировать клиент командной строки *mysql.exe*. Определим IP-адрес сервера:

```
C:\Documents and Settings\admin>ipconfig

Настройка протокола IP для Windows


Подключение по локальной сети - Ethernet адаптер:

        DNS-суффикс этого подключения . . :
        IP-адрес  . . . . . . . . . . . . : 192.168.67.6
        Маска подсети . . . . . . . . . . : 255.255.255.0
        Основной шлюз . . . . . . . . . . : 192.168.67.254
```

Создадим новую учетную запись, позволив пользователю *user1* обращаться к серверу MySQL с любых компьютеров сети:

```
mysql> CREATE USER 'user1'@'%' IDENTIFIED BY '123';
Query OK, 0 rows affected (0.01 sec)
```

Назначим этому пользователю привилегии глобального уровня:

```
mysql> GRANT ALL ON *.* TO 'user1'@'%' IDENTIFIED BY '123';
Query OK, 0 rows affected (0.00 sec)
```

На клиентском компьютере в командной строке (например, с помощью FAR), запустим клиент командной строки в следующем формате:

```
The FAR manager, version 1.70 beta 5 (build 1634)
Copyright (C) 1996-2000 Eugene Roshal, Copyright (C) 2000-2003 FAR Group
Зарегистрирован: xUSER регистрация
C:\>mysql -u user1 -p123 -h 192.168.67.6
```

Наблюдаем отклик удаленного сервера и работаем с ним как обычно:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.0.51b-community-nt MySQL Community Edition (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| book               |
| mysql              |
| test               |
+--------------------+
4 rows in set (0.00 sec)
```

Варианты заданий к практическим работам по MySQL

1. *Страховая компания*. Страховая компания имеет филиалы, которые характеризуются наименованием, адресом и телефоном. В филиалы обращаются клиенты с целью заключения договора о страховании. В зависимости от принимаемых на страхование объектов и страхуемых рисков договор заключается по определенному виду страхования (страхование автотранспорта от угона, страхование домашнего имущества, добровольное медицинское страхование). При заключении договора фиксируются: дата заключения, страховая сумма, вид страхования, тарифная ставка и филиал, в котором заключался договор. Договоры заключают страховые агенты. Помимо информации об агентах (фамилия, имя, отчество, адрес, телефон) нужно хранить филиал, в котором они работают. Необходимо иметь возможность рассчитывать заработную плату агентам. Заработная плата составляет некоторый процент от страхового платежа (платеж – страховая сумма, умноженная на тарифную ставку). Процент зависит от вида страхования, по которому заключен договор.

2. *Гостиница*. Гостиница предоставляет номера клиентам. Каждый номер характеризуется вместимостью, комфортностью (люкс, полулюкс, обычный) и ценой. О клиентах собирается определенная информация (фамилия, имя, отчество, паспортные данные, адрес жительства и некоторый

комментарий). Сдача номера клиенту производится при наличии свободных мест в номерах, подходящих клиенту по указанным выше параметрам. При заселении фиксируется дата заселения. При выезде из гостиницы для каждого места запоминается дата освобождения. Необходимо также осуществлять бронирование номеров. Для постоянных клиентов, а также для определенных категорий клиентов предусмотрена система скидок. Скидки могут суммироваться.

3. *Ломбард*. В ломбард обращаются различные лица с целью получения денежных средств под залог товаров. Клиент сообщает фамилию, имя, отчество и другие паспортные данные. После оценивания стоимости принесенного в качестве залога товара работник ломбарда определяет сумму, которую готов выдать на руки клиенту, а также свои комиссионные. Кроме того определяется срок возврата денег. Если клиент согласен, то договоренности фиксируются в виде документа, деньги выдаются клиенту, а товар остается в ломбарде. Если в указанный срок не происходит возврата денег, товар переходит в собственность ломбарда. После перехода прав собственности на товар, ломбард может продавать товары по цене, меньшей или большей, чем была заявлена при сдаче. Цена может меняться несколько раз, в зависимости от ситуации на рынке (например, владелец ломбарда может устроить распродажу зимних вещей в конце зимы). Помимо текущей цены нужно хранить все возможные значения цены для данного товара.

4. *Оптово-розничная продажа товаров*. Компания торгует товарами из определенного спектра. Каждый товар характеризуется наименованием, оптовой ценой, розничной ценой и справочной информацией. В компанию обращаются покупатели, для каждого из которых в базе данных фиксируются стандартные данные (наименование, адрес, телефон, контактное лицо). По каждой сделке составляется документ, в котором наряду с покупателем фиксируются количество купленного им товара и дата покупки. Обычно покупатели в рамках одной сделки покупают не один товар, а сразу несколько. Также компания решила предоставлять скидки в зависимости от количества закупленных товаров и их общей стоимости.

5. *Ведение заказов*. Компания занимается оптовой продажей различных товаров. Каждый из товаров характеризуется ценой, справочной информацией и признаком наличия или отсутствия доставки. В компанию обращаются заказчики. Для каждого из них в базе данных запоминаются стандартные данные (наименование, адрес, телефон, контактное лицо). По каждому заказу составляется документ, в котором наряду с заказчиком фиксируются количество купленного им товара и дата покупки. Доставка товаров может производиться способами, различными по цене и скорости. Нужно хранить информацию о том, какими способами может осуществляться доставка каждого товара, и информацию о том, какой вид доставки (и какую стоимость доставки) выбрал клиент при заключении сделки.

6. *Бюро по трудоустройству*. Бюро готово искать работников для различных работодателей и вакансии для ищущих работу специалистов различного профиля. При обращении в бюро работодателя его стандартные данные (название, вид деятельности, адрес, телефон) фиксируются в базе данных. При обращении в бюро соискателя его стандартные данные (фамилия, имя, отчество, квалификация, профессия, иные данные) также фиксируются в базе данных. По каждому факту удовлетворения интересов обеих сторон составляется документ. В документе указываются соискатель, работодатель, должность и комиссионные (доход бюро). В базе должна фиксироваться не только сделка, но и информация по открытым вакансиям. Кроме того для автоматического поиска вариантов необходимо вести справочник «Виды деятельности».

7. *Нотариальная контора*. Нотариальная контора готова предоставить клиенту определенный комплекс услуг. Услуги формализованы, т. е. составлен их список с описанием каждой услуги. При обращении клиента его стандартные данные (название, вид деятельности, адрес, телефон) фиксируются в базе данных. По каждому факту оказания услуги клиенту составляется документ, в котором указываются дата, услуга, сумма сделки, комиссионные (доход конторы), описание сделки. В рамках одной сделки клиенту может быть оказано несколько услуг. Стоимость каждой услуги фиксирована. Кроме того компания предоставляет в рамках одной сделки различные виды скидок. Скидки могут суммироваться.

8. *Фирма по продаже запчастей*. Фирма продает запасные части для автомобилей. Фирма имеет определенный набор поставщиков, по которым известны название, адрес и телефон. У поставщиков приобретаются детали. Каждая деталь характеризуется названием, артикулом и ценой. Некоторые из поставщиков могут поставлять одинаковые детали (один артикул). Каждый факт покупки запчастей у поставщика фиксируется в базе данных, причем обязательными для запоминания являются дата покупки и количество приобретенных деталей. Цена детали может меняться от поставки к поставке. Поставщики заранее ставят фирму в известность о дате изменения цены и ее новом значении. Нужно хранить не только текущее значение цены, но и всю историю изменения цен.

9. *Курсы по повышению квалификации*. В учебном заведении организованы курсы повышения квалификации. Группы слушателей формируются в зависимости от специальности и отделения. В каждую из них включено определенное количество слушателей. Проведение занятий обеспечивает штат преподавателей, для каждого из которых в базе данных зарегистрированы стандартные анкетные данные (фамилия, имя, отчество, телефон) и стаж работы. В результате распределения нагрузки получена информация о том, сколько часов занятий проводит каждый преподаватель с соответствующими группами. Хранятся также сведения о виде занятий (лекция, практика), дисциплине и оплате за 1 час. Размер почасовой оплаты

зависит от предмета и типа занятия. Кроме того каждый преподаватель может вести не все предметы, а только некоторые.

10. *Определение факультативов для студентов*. Преподаватели кафедры в высшем учебном заведении обеспечивают проведение факультативных занятий по некоторым предметам. Имеются сведения о студентах, включающие стандартные анкетные данные (фамилия, имя, отчество, группа, адрес, телефон). По каждому факультативу существует определенное количество часов и вид проводимых занятий (лекции, практика, лабораторные работы). В результате работы со студентами появляется информация о том, кто из них записался на какие факультативы. Существует некоторый минимальный объем факультативных предметов, которые должен прослушать каждый студент. По окончании семестра в базу данных заносится информация об оценках, полученных студентами на экзаменах. Некоторые из факультативов могут длиться более одного семестра. В каждом семестре для предмета устанавливается объем лекций, практик и лабораторных работ в часах. В качестве итоговой оценки за предмет берется последняя оценка, полученная студентом.

11. *Распределение учебной нагрузки*. Необходимо распределять нагрузку между преподавателями кафедры. Имеются сведения о преподавателях, включающие наряду с анкетными данными сведения об их ученой степени, занимаемой должности и стаже работы. Преподаватели кафедры должны обеспечить проведение занятий по некоторым дисциплинам. По каждой из них существует определенное количество часов. В результате распределения нагрузки необходимо получить информацию следующего рода: «Такой-то преподаватель проводит занятия по такой-то дисциплине с такой-то группой». Все проводимые занятия делятся на лекционные и практические. По каждому виду занятий устанавливается свое количество часов. Кроме того данные по нагрузке нужно хранить несколько лет.

12. *Распределение дополнительных обязанностей*. Кафедра вуза имеет штат сотрудников, каждый из которых получает определенный оклад. Каждый месяц возникает потребность в выполнении некоторой дополнительной работы, не входящей в круг основных обязанностей сотрудников. Для наведения порядка в этой сфере классифицированы все виды дополнительных работ и определена сумма оплаты по факту их выполнения. При возникновении дополнительной работы назначается ответственный, фиксируется дата начала. По факту окончания фиксируется дата и выплачивается дополнительная сумма к зарплате с учетом классификации. Необходимо учесть разделение сотрудников на преподавателей и учебно-вспомогательный персонал. Для первых нужно хранить сведения об ученой степени и ученом звании, для вторых – о должности. Некоторые работы являются трудоемкими и срочными, что требует привлечения к их выполнению нескольких сотрудников. Длительность работ различна. Нужно

заранее планировать длительность работы и количество сотрудников, занятых для выполнения работы.

13. *Техническое обслуживание станков*. Компания занимается ремонтом станков и другого оборудования. Клиентами компании являются промышленные предприятия. Ремонтные работы организованы следующим образом: все станки классифицированы по типам, странампроизводителям, годам выпуска и маркам. Все виды ремонта отличаются названием, продолжительностью в днях, стоимостью. Исходя из этих данных, по каждому факту ремонта фиксируется вид станка, дата начала и дата окончания ремонта. Анализ показал, что нужно не просто подразделять станки по типам, а иметь информацию о том, сколько раз ремонтировался тот или иной станок.

14. *Туристическая фирма*. Фирма продает путевки клиентам. У каждого клиента запрашиваются стандартные данные – фамилия, имя, отчество, адрес, телефон. После этого сотрудники компании выясняют у клиента, куда он хотел бы поехать отдыхать. Ему демонстрируются различные варианты, включающие страну проживания, особенности климата, отель. Обсуждается длительность пребывания и стоимость путевки. Если удалось найти приемлемый вариант, регистрируется факт продажи путевки (или путевок, если клиент покупает сразу несколько), фиксируется дата отправления. Иногда клиенту предоставляется скидка (скидки фиксированы и могут суммироваться). Фирма работает с несколькими отелями (название, категория, адрес) в нескольких странах. Путевки продаются на одну, две или четыре недели. Стоимость путевки зависит от длительности тура и отеля.

15. *Грузовые перевозки*. Компания осуществляет перевозки грузов по различным маршрутам. Необходимо отслеживать стоимость перевозок с учетом заработной платы водителей. Для каждого маршрута определено название, вычислено примерное расстояние и установлена некоторая оплата для водителя. Информация о водителях включает фамилию, имя, отчество и стаж. Для проведения расчетов хранится полная информация о перевозках (маршрут, водитель, даты отправки и прибытия). По факту некоторых перевозок водителям выплачивается премия. Фирма решила ввести гибкую систему оплаты. Оплата водителям должна зависеть не только от маршрута, но и от стажа водителя. Кроме того, нужно учесть, что перевозку могут осуществлять два водителя.

16. *Учет телефонных переговоров*. Телефонная компания предоставляет абонентам телефонные линии для междугородних переговоров. Абонентами компании являются юридические лица, имеющие телефонную точку, ИНН, расчетный счет в банке. Стоимость переговоров зависит от города, в который осуществляется звонок, и времени суток (день, ночь). Каждый звонок абонента автоматически фиксируется в базе данных. При этом запоминаются город, дата, длительность разговора и время суток. Компания решила ввести гибкую систему скидок. Так, стоимость минуты теперь

уменьшается в зависимости от длительности разговора. Размер скидки для каждого города разный.

17. *Учет внутриофисных расходов*. Сотрудники частной фирмы могут осуществлять мелкие покупки для нужд фирмы, предоставляя в бухгалтерию товарный чек. Бухгалтерия отслеживает внутриофисные расходы. Фирма состоит из отделов, каждый из которых имеет название. В каждом отделе работает определенное количество сотрудников. Сотрудники могут осуществлять покупки в соответствии с видами расходов. Каждый вид расходов имеет название, некоторое описание и предельную сумму средств, которые могут быть потрачены по данному виду расходов в месяц. При каждой покупке сотрудник оформляет документ, где указывает вид расхода, дату, сумму и отдел. Нужно хранить данные о расходах не только в целом по отделу, но и по отдельным сотрудникам. Нормативы по расходованию средств устанавливаются не в целом, а по каждому отделу за каждый месяц. Неиспользованные в текущем месяце деньги могут быть использованы позже.

18. *Библиотека*. Библиотека решила зарабатывать деньги, выдавая напрокат книги, имеющиеся в небольшом количестве экземпляров. У каждой книги, выдаваемой в прокат, есть название, автор, жанр. В зависимости от ценности книги для каждой из них определена залоговая стоимость (сумма, вносимая клиентом при взятии книги напрокат) и стоимость проката (сумма, которую клиент платит при возврате книги, получая назад залог). Читатели регистрируются в картотеке, которая содержит стандартные анкетные данные (фамилия, имя, отчество, адрес, телефон). Каждый читатель может обращаться в библиотеку несколько раз. Все обращения читателей фиксируются, при этом по каждому факту выдачи книги запоминаются дата выдачи и ожидаемая дата возврата. Стоимость проката книги должна зависеть не только от самой книги, но и от срока ее проката. Кроме того, необходимо добавить систему штрафов за вред, нанесенный книге, и систему скидок для некоторых категорий читателей.

19. *Прокат автомобилей*. Фирма, занимающаяся прокатом автомобилей, имеет автопарк, содержащий некоторое количество автомобилей различных марок, стоимостей и типов. Каждый автомобиль имеет свою стоимость проката. В пункт проката обращаются клиенты. Клиенты проходят обязательную регистрацию, в ходе которой о них собирается стандартная информация (фамилия, имя, отчество, адрес, телефон). Каждый клиент может обращаться в пункт проката несколько раз. Обращения клиентов фиксируются, при этом по каждой сделке запоминаются дата выдачи и ожидаемая дата возврата. Стоимость проката автомобиля должна зависеть не только от самого автомобиля, но и от срока его проката, а также от года выпуска. Также нужно ввести систему штрафов за возвращение автомобиля в ненадлежащем виде и систему скидок для постоянных клиентов.

20. *Выдача банком кредитов*. Коммерческий банк выдает кредиты юридическим лицам. В зависимости от условий получения кредита,

процентной ставки и срока возврата все кредитные операции делятся на несколько основных видов. Каждый из этих видов имеет свое название. Кредит может получить юридическое лицо (клиент), при регистрации предоставивший следующие сведения: название, вид собственности, адрес, телефон, контактное лицо. Каждый факт выдачи кредита регистрируется банком, при этом фиксируются сумма кредита, клиент и дата выдачи. Чтобы отслеживать динамику возврата кредитов принято решение учитывать в системе еще и дату фактического возврата денег. Нужно еще учесть, что кредит может гаситься частями, и за задержку возврата кредита начисляются штрафы.

21. *Инвестиционная компания*. Компания занимается вложением денежных средств в ценные бумаги, которые характеризуются рейтингом, доходностью за прошлый год, минимальной суммой сделки и некоторой дополнительной информацией. Клиентами компании являются предприятия, которые доверяют ей управлять их свободными денежными средства на определенный период. Необходимо выбрать вид ценных бумаг, которые позволят получить прибыль и компании и клиенту. При работе с клиентом существенной является информация о предприятии – название, вид собственности, адрес и телефон. Каждая инвестиция характеризуется информацией о клиенте, информацией о ценной бумаге, котировкой бумаги, датой ее покупки и датой ее продажи. Необходимо хранить историю котировок каждой ценной бумаги. Кроме того, помимо вложений в ценные бумаги существует возможность вкладывать деньги в банковские депозиты.

22. *Занятость актеров театра*. Коммерческий директор театра организует привлечение актеров и заключение контрактов. Каждый год театр осуществляет постановку различных спектаклей. Каждый спектакль имеет определенный бюджет. Для участия в конкретных постановках в определенных ролях привлекаются актеры. С каждым из актеров заключается персональный контракт на определенную сумму. Каждый актер имеет стаж работы, некоторые из них удостоены различных званий. В рамках одного спектакля на одну и ту же роль привлекается несколько актеров. Контракт определяет базовую зарплату актера, а по итогам реально отыгранных спектаклей актеру назначается премия. В базе данных нужно хранить информацию за несколько лет.

23. *Платная поликлиника*. В поликлинике работают врачи различных специальностей, имеющие разную квалификацию. Каждый день в поликлинику обращаются пациенты. Все пациенты проходят обязательную регистрацию, при которой в базу данных заносятся стандартные анкетные данные (фамилия, имя, отчество, год рождения, адрес). При обращении в поликлинику пациент обследуется и проходит лечение у разных специалистов. Каждый пациент может обращаться в поликлинику несколько раз, нуждаясь в различной медицинской помощи. Все обращения пациентов фиксируются, при этом устанавливается диагноз, определяется стоимость лечения, запоминается

дата обращения. Общая стоимость лечения зависит от стоимости консультаций и процедур, назначенных пациенту. Для определенных категорий граждан предусмотрены скидки.

24. Анализ динамики показателей финансовой отчетности предприятий. Информационно-аналитический центр крупного холдинга отслеживает динамику показателей предприятий холдинга. В структуру холдинга входят несколько предприятий. Каждое предприятие имеет стандартные характеристики (название, реквизиты, телефон, контактное лицо). Работа предприятия может быть оценена следующим образом: в начале каждого отчетного периода на основе финансовой отчетности вычисляется определенный набор показателей. Важность показателей характеризуется некоторыми числовыми константами. Значение каждого показателя измеряется в некоторой системе единиц. Некоторые показатели считаются в рублях, некоторые в долларах, некоторые в евро. Для удобства работы с показателями нужно хранить изменения курсов валют относительно друг друга.

25. Учет телекомпанией стоимости прошедшей в эфире рекламы. Работа коммерческой службы телевизионной компании построена следующим образом: заказчики просят поместить свою рекламу в определенной передаче в определенный день. Каждый рекламный ролик имеет определенную продолжительность. Для каждой организации-заказчика известны банковские реквизиты, телефон и контактное лицо для проведения переговоров. Передачи имеют определенный рейтинг. Стоимость минуты рекламы в каждой передаче определяется, исходя из рейтинга передачи и прочих соображений. Необходимо хранить информацию об агентах, заключивших договоры на рекламу. Зарплата рекламных агентов составляет некоторый процент от общей стоимости рекламы, прошедшей в эфире.

26. IT-компания. Компания оказывает IT-услуги организациям и предприятиям. В компании работают сотрудники, о которых должна сохраняться стандартная информация и данные о квалификации (владение языками и системами программирования, знание СУБД, операционных систем). В компанию обращаются клиенты, о которых собираются стандартные данные (наименование и адрес организации, телефон, адрес электронной почты, фамилия, имя и отчество контактного лица для связи). Задание для клиента выполняет определенный сотрудник, при этом фиксируется дата выдачи задания и трудоемкость выполнения (в часах). При повторном обращении клиент переходит в категорию постоянных и получает скидку. С ростом компании возникла необходимость разделения ее на отделы. Увеличились масштабы проектов, и теперь задание клиента поручается отделу. В рамках одного договора может выполняться несколько заданий разными отделами компании.

27. Ювелирная мастерская. Ювелирная мастерская осуществляет изготовление ювелирных изделий для частных лиц на заказ. Мастерская

работает с определенными материалами (платина, золото, серебро, драгоценные камни). При обращении потенциального клиента выясняется, какое именно изделие ему необходимо. Все изготавливаемые изделия принадлежат к некоторому типу (серьги, кольца, броши, браслеты), выполняются из определенного материала, имеют некоторый вес и цену (включающую стоимость материалов и работы). Ювелирное изделие может состоять из нескольких материалов. Кроме того, постоянным клиентам мастерская предоставляет скидки.

28. *Парикмахерская*. Парикмахерская стрижет клиентов в соответствии с их пожеланиями и некоторым каталогом различных видов стрижки. Для каждой стрижки определены название, категория (мужская, женская, детская), стоимость работы. Для наведения порядка составляется база данных клиентов, где запоминаются их анкетные данные (фамилия, имя, отчество). Начиная с 5-ой стрижки, клиент переходит в категорию постоянных и получает скидку в 3 % при каждой последующей стрижке. После того, как закончена очередная работа, в БД фиксируются стрижка, клиент и дата производства работ. Кроме того, у парикмахерской появился филиал и необходима раздельная статистика по филиалам. Стоимость стрижки может меняться с течением времени. Нужно хранить не только последнюю цену, но и все данные по изменению цены стрижки.

29. *Химчистка*. Химчистка осуществляет прием у населения вещей для выведения пятен. Для наведения порядка составляется база данных клиентов, в которой запоминаются их анкетные данные (фамилия, имя, отчество, адрес, телефон). Начиная с 3-го обращения, клиент переходит в категорию постоянных клиентов и получает скидку в 3 % при чистке каждой последующей вещи. Все оказываемые услуги подразделяются на виды, имеющие название, тип и стоимость, зависящую от сложности работ. Работа с клиентом первоначально состоит в определении объема работ, вида услуги и, соответственно, ее стоимости. Если клиент согласен, он оставляет вещь (при этом фиксируется услуга, клиент и дата приема) и забирает ее после обработки (при этом фиксируется дата возврата). Химчистка заключает с клиентом договор. Клиент может одновременно сдавать в чистку несколько вещей. У химчистки появились филиалы, и необходима раздельная статистика по филиалам. Введены надбавки за срочность и сложность.

30. *Сдача в аренду торговых площадей*. Торговый центр сдает в аренду коммерсантам свои торговые площади. В результате планирования определено некоторое количество торговых точек в пределах здания, которые могут сдаваться в аренду. Для каждой из торговых точек важными данными являются этаж, площадь, наличие кондиционера и стоимость аренды в день. С потенциальных клиентов собираются стандартные данные (название, адрес, телефон, реквизиты, контактное лицо). При появлении потенциального клиента ему показывают имеющиеся свободные площади. При достижении соглашения оформляется договор и в базе данных фиксируется торговая точка,

клиент, период (срок) аренды. Некоторые клиенты в рамках одного договора арендуют сразу несколько торговых точек, причем для каждой точки возможен свой срок аренды. Дата заключения договора может не совпадать с датой начала аренды. Необходимо собирать информацию об ежемесячных платежах, поступающих от арендаторов.

Список литературы

Электронные издания (электронные ресурсы)

1. Голицына, О.Л. Основы проектирования баз данных [Электронный ресурс]: учеб. пособие / О.Л. Голицына, Т.Л. Партыка, И.И. Попов. – 2-е изд., перераб. и доп. – Москва: ФОРУМ: ИНФРА-М, 2019. – 416 с. - Режим доступа: http://znanium.com/catalog/product/1018906 (ЭБС Znanium)
2. Шустова,Л.И. Базы данных [Электронный ресурс]: учебник / Л.И. Шустова, О.В. Тараканов. — М. : ИНФРА-М, 2019. – 304 с. + Доп. материалы .- Режим доступа: http://znanium.com/catalog/product/967755 (ЭБС Znanium)
3. Стружкин, Н. П. Базы данных: проектирование. Практикум : учеб. пособие для СПО / Н. П. Стружкин, В. В. Годин. — Москва: Издательство Юрайт, 2019. – 291 с. - Текст : электронный // ЭБС Юрайт [сайт]. – Режим доступа: https://biblioonline.ru/bcode/442343
4. Илюшечкин, В. М. Основы использования и проектирования баз данных : учебник для СПО / В. М. Илюшечкин. – испр. и доп. – Москва : Издательство Юрайт, 2019. – 213 с. – Текст: электронный // ЭБС Юрайт [сайт]. – Режим доступа:: https://biblioonline.ru/bcode/437670

Методические издания

1. Игнатенко, Е.С. Методические указания по выполнению практических работ по ОП.08 Основы проектирования баз данных .– Нефтеюганск: НИК(филиал) ФГБОУ ВО «ЮГУ», 2019 [Электронный ресурс] Режим доступа: локальная сеть филиала Периодические издания

1. Программные продукты и системы [Электронный ресурс]: журнал. - Тверь: – Научноисследовательский институт «Центрпрограммсистем» Электронно-библиотечная система «Лань»: [сайт]. – Режим доступа: https://e.lanbook.com/journal/2276