МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению лабораторных работ
по дисциплине «Безопасность баз данных»
для студентов направления подготовки
10.03.01 «Информационная безопасность»
Направленность (профиль)
«Организация и технологии защиты информации (по отрасли или в сфере профессиональной деятельности)»

Ставрополь
2026

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

1. Цель и задачи освоения дисциплины

Целью дисциплины «Безопасность баз данных» является формирование у обучающихся понимания основ информационной безопасности систем баз данных для последующего практического использования в науке и образовании.

Задачами дисциплины являются:

- Теоретическая и практическая подготовка бакалавров к настройке параметров безопасности баз данных;
- Объяснение принципов построения подсистем защиты в СУБД, средств и методов несанкционированного доступа к ресурсам СУБД;
- Привитие студентам системного подхода к проблеме защиты информации в СУБД; механизмов защиты информации и возможностей по их преодолению.

2. Перечень планируемых результатов обучения по дисциплине (модулю), соотнесённых с планируемыми результатами освоения образовательной программы

Код, формулировка компетенции	Код, формулировка индикатора	Планируемые результаты обучения по дисциплине (модулю), характеризующие этапы формирования компетенций, индикаторов
ПК-1 Способен обеспечивать защиту от несанкционированного доступа сооружений и средств связи сетей электросвязи (за исключением сетей связи специального назначения) в процессе их эксплуатации	ИД-1 Способен проводить мониторинг функционирования СССЭ, защищённости от НСД сооружений и СССЭ	- Применяет методы контроля функционирования СССЭ, их защищенности от НСД - Анализирует средства мониторинга работоспособности и эффективности применяемых программных, программно-аппаратных (в том числе криптографических) и технических средств защиты СССЭ от НСД
	ИД-2. Способен к управлению функционированием СССЭ, защищённостью от НСД сооружений СССЭ	- определяет необходимый состав, особенности размещения СССЭ, а также программных, программно-аппаратных (в том числе криптографических) и технических средств и систем защиты СССЭ от НСД; - организовывает и проводит монтаж и настройку СССЭ, а также

		программных, программно-аппаратных (в том числе криптографических) и технических средств и систем защиты СССЭ от НСД; - устанавливает и настраивает программное обеспечение, необходимое для управления СССЭ и средствами их защиты от НСД
	ИД-1 Способен проводить мониторинг функционирования СССЭ, защищённости от НСД сооружений и СССЭ	- Применяет методы контроля функционирования СССЭ, их защищенности от НСД - Анализирует средства мониторинга работоспособности и эффективности применяемых программных, программно-аппаратных (в том числе криптографических) и технических средств защиты СССЭ от НСД
ПК-3 Способен обеспечивать защиту информации в автоматизированных системах в процессе их эксплуатации	ИД-1 Способен администрировать системы защиты информации автоматизированных систем	Устанавливает и настраивает операционные системы, системы управления базами данных, компьютерные сети и программные системы с учетом требований по обеспечению защиты информации

Раздел 1. Архитектура баз данных

Лабораторная работа №1-2

Тема 1 - 2. Базы данных и файловые системы. Функции СУБД

1.Цель работы: изучение технологии создания базы данных с использованием конструктора базы данных

Формируемые компетенции: ПК-1, ПК-3

2. Подготовка к занятию

1. Изучить (повторить) теоретический материал.
2. Ознакомиться с программой работы.
3. Подготовить отчет о работе.
4. Ответить на контрольные вопросы.

3. Распределение времени занятия:

Всего: 90 мин

Вступительная часть 2 мин

Проверка готовности студентов к занятию 5 мин

Основная часть – 70 мин

Проверка выполнения практического занятия 10 мин

Заключительная часть 3 мин

4. Правила работы в лаборатории

К работе в лаборатории допускаются лица, изучившие правила и меры безопасности, сдавшие зачет по ним и усвоившие порядок выполнения лабораторной работы.

4.1. Требования безопасности перед началом работ

- ЗАПРЕЩАЕТСЯ: переодеваться, пользоваться огнем, курить, принимать пищу в лаборатории.

- Убедиться в целостности электрических розеток и разъемов. В лаборатории необходимо быть в сменной обуви.

- Включение компьютера производить только после получения допуска по выполняемой работе и разрешения преподавателя или лаборанта.

4.2. Требования безопасности во время работы

- выполняя практическое занятие, студенты обязаны использовать только вычислительную технику, периферийное оборудование, соединительные кабели, измерительное оборудование и носители информации, непосредственно относящиеся к данному лабораторному занятию;

- подключение и отключение составляющих вычислительного комплекса производить только при полном снятии напряжения со всех составляющих вычислительного комплекса;

- при обнаружении неисправностей в оборудовании немедленно отключить источники питания и доложить об этом руководителю занятий или лаборанту.

4.3. Требования безопасности по окончании работы

- доложить руководителю занятий или лаборанту о завершении работ;
- привести в порядок и сдать рабочее место лаборанту, и доложить руководству

1 ОСНОВНЫЕ СВЕДЕНИЯ О БАЗАХ ДАННЫХ

В системах автоматизации обработки информации значительная часть затрат на разработку падает на проектирование базы данных как основного средства структурного хранения информации и организации доступа к данным. База данных в Visual FoxPro – это совокупность *таблиц* (Table), *представлений* (View) и *отношений* между ними. В БД входят также *хранимые процедуры* (Stored Procedures), которые используются для описания сложных правил проверки целостности данных в БД, а также *соединения* (Connections) для связи с внешними источниками данных. Файлы БД имеют расширение *dbc* и хранят информацию о входящих в нее компонентах.

Создание базы данных в Visual FoxPro осуществляется в интерактивном режиме с помощью визуального инструментария создания базы данных - *конструктора БД* (Database Designer), позволяющего:

- создавать и модифицировать таблицы, хранимые процедуры, представления данных;
- добавлять свободные таблицы;
- определять для таблиц индексы;
- устанавливать отношения между таблицами, которые будут поддерживаться при создании форм и отчётов.

Для создания базы данных необходимо открыть окно конструктора БД воспользовавшись системным меню Visual FoxPro. Для этого в меню *File* (Файл) выбирается команда *New* (Новый). В открывшемся диалоговом окне New необходимо установить опцию *Database* (База данных) и нажать кнопку *New file* (Новый файл). В окне *Create* в поле *Enter* задать имя создаваемой базы данных. Убедившись, что в поле Тип файла установлен тип сохраняемого файла *dbc*, а в раскрывающемся списке Сохранить в правильно указана папка, в которой будет располагаться создаваемая база данных, нажать кнопку Сохранить.

Откроется пустое окно базы данных *Database Designer* (Конструктор базы данных) (Рисунок 1). Используя панель инструментов *Database Designer*, команды меню *Database* и контекстного меню, в окне конструктора базы данных можно добавлять имеющиеся свободные таблицы, создавать новые таблицы и представления, модифицировать и удалять существующие, создавать для них индексы, устанавливать отношения между таблицами.

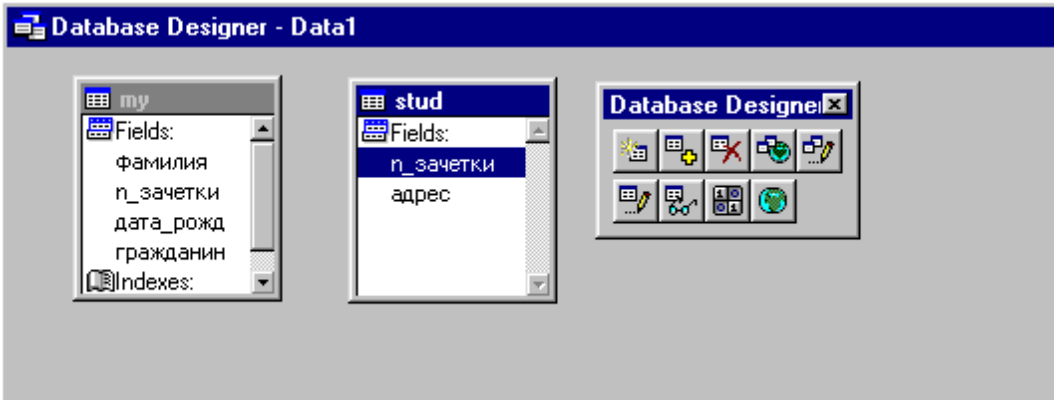


Рисунок 1 - Окно Database Designer

В начале создания базы данных окно конструктора базы данных пусто. Для создания в конструкторе базы данных новых таблиц и модификации существующих, можно использовать команды меню *Database*:

- New Table (Новая таблица) – создаёт новую таблицу;
- Add Table (Добавить таблицу) – добавляет созданную ранее свободную таблицу в базу данных;
- New Remote View (Новое удаленное представление) – создаёт удалённое представление данных;
- New Local View (Новое локальное представление) – создаёт локальное представление данных;
- Modify (Модифицировать) – открывает таблицу в конструкторе таблиц;
- Browse (Обзор таблицы) – показывает содержимое таблицы в режиме Browse;
- Remove (Удалить) – удаляет таблицу из базы данных;
- Find Object (Найти объект) – находит указанный объект в окне конструктора базы данных ;
- Rebuild Table Indexes (Перестроить индексы) – перестраивает индексы;
- Remove Deleted Records (Удалить помеченные записи) – физически удаляет из таблицы помеченные для удаления записи;
- Edit Relationship (Редактирование отношений) – редактирует отношения между таблицами;
- Edit Referential Integrity (Редактирование условия целостности) – определяет условия целостности данных;
- Edit Stored Procedures (Редактирование хранимых процедур) – открывает окно редактирования хранимых процедур;
- Connections (Соединения) – выводит на экран диалоговое окно Connections, в котором создаются или модифицируются соединения с удалёнными данными;

— Arrange (Упорядочить) – упорядочивает объекты по имени или типу и выравнивает их по горизонтали или вертикали;
— Refresh (Обновить) – обновляет информацию в окне конструктора базы данных;
— Properties (Свойства) – выводит на экран диалоговое окно Database Properties;
— Clean Up Database (Очистка базы данных) – очищает базу данных от помеченных на удаление объектов.

Для работы в окне конструктора базы данных можно использовать контекстное меню, вызываемое нажатием правой кнопки мыши. Оно содержит наиболее часто используемые команды меню *Database*, команду вызова справочной системы, а также команды *Expand All* (Развернуть все) и *Collapse All* (Свернуть все), предназначенные для раскрытия и свёртывания уровней вложенности объектов в окне конструктора базы данных.

Панель инструментов *Database Designer* содержит кнопки для выполнения наиболее часто используемых операций над базой данных. Если панель инструментов *Database Designer* не видна на экране, в меню *View* (Вид) нужно выбрать команду *Toolbars* (Панель инструментов). Откроется диалоговое окно *Toolbars*, в котором необходимо установить флажок напротив пункта *Database Designer*. Вид панели инструментов *Database Designer* приведён на рисунке 2.

Рисунок 2 - Панель инструментов Database Designer

Ниже приведено описание кнопок панели *Database Designer* (слева направо):
— New Table (Новая таблица) - создаёт новую таблицу;
— Add Table (Добавить таблицу) – добавляет ранее созданную таблицу в базу данных;
— Remove Table (Удалить таблицу) – удаляет таблицу из базы данных;
— New Remote View (Новое удалённое представление) – создаёт удалённое представление данных;
— New Local View (Новое локальное представление) – создаёт локальное представление данных;
— Modify Table (Модифицировать таблицу) – открывает таблицу в конструкторе таблиц;
— Browse Table (Обзор таблицы) – показывает содержимое таблицы в режиме Browse;
— Edit Stored Procedures (Редактирование хранимых процедур) – открывает окно для редактирования хранимых процедур;

— Connections (Соединение) – создает связь с удалёнными данными.

Для создания таблицы из конструктора базы данных нужно выполнить одно из следующих действий:

- выбрать команду *New Table* (Новая таблица) из меню *Database* (База данных);
- выбрать команду *New Table* контекстного меню;
- нажать кнопку *New Table* на панели инструментов *Database Designer* (Конструктор базы данных).

В появившемся диалоговом окне *New Table* (Новая таблица) нажать кнопку *Table Wizard* (для запуска Мастера таблиц) или *New File* (для запуска Конструктора таблиц).

При создании таблиц в базе данных есть дополнительные возможности для придания таблице большей функциональности. Окно конструктора таблиц в этом случае содержит дополнительные области для задания параметров, свойств, функций таблицы.

При определении полей таблицы используется вкладка *Fields* (Поля), позволяющая помимо основных параметров указать для каждого поля дополнительные параметры, которые будут определять условия ввода данных в него, а также краткое описание, которое поможет разработчику при модификации таблицы в процессе создания приложения или его сопровождения (Рисунок 3).

В нижней части вкладки *Fields* конструктора таблиц расположены области, позволяющие задать для каждого поля таблицы свойства, которые будут использоваться при вводе данных в эти поля.

Область *Display* (Отображение) содержит поля, позволяющие задать форматы ввода и отображения данных:

— Format (Формат) – задаёт формат отображения данных в формах, отчётах и окне Browse;
— Input mask (Маска ввода) задаёт формат ввода данных;
— Caption (Надпись) – определяет выводимый заголовок поля.

Область *Map field type to classes* (Используемые типы полей для классов) предназначена для указания библиотеки и имени класса, который будет использоваться для создания объектов при размещении данного поля таблицы в форме:

— Display library (Показывать библиотеку) – задаёт местоположение и имя файла библиотеки классов;
— Display class (Показывать класс) – задаёт имя класса из выбранной библиотеки.

Область Field Validation (Проверка правильности ввода) позволяет задать следующие параметры:

— Rule (Условие) – условие правильности ввода данных;

— Message (Сообщение) – сообщение, выводимое при неправильном вводе данных в поле;

— Default Value (Значение по умолчанию) – значение, вводимое в поле по умолчанию.

В текстовом поле *Field comment* (Комментарий) можно ввести краткое описание поля, которое может потребоваться при последующих модификациях структуры таблицы и сопровождении проекта.

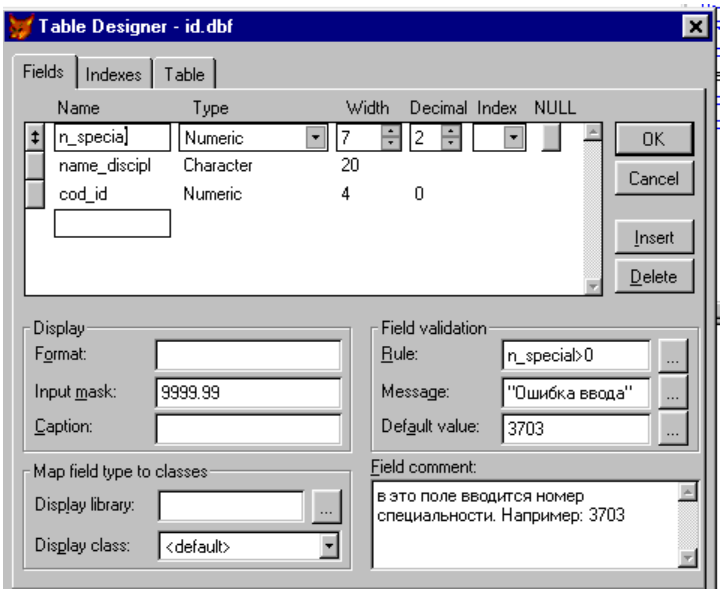


Рисунок 3 -Вкладка Fields Конструктора таблиц с дополнительными областями.

На вкладке *Indexes* (Индексы) можно задать новые и переопределить имеющиеся индексы в таблице. В таблице БД в отличие от свободной таблицы можно задать первичный ключ - *Primary Key*. При этом нужно помнить, что поле может использоваться для задания первичного ключа таблицы только в том случае, если оно содержит неповторяющиеся значения. С помощью кнопок *Insert* и *Delete* добавляют новые индексы в таблицу и удаляют существующие.

Для определения свойств самой таблицы предназначена вкладка *Table* (Таблица) конструктора. Для таблиц, входящих в базу данных, можно задать два имени. Одно вводится в диалоговом окне *Create*, а второе – на вкладке *Table* окна конструктора таблицы. Имя, вводимое в диалоговом окне *Create* при создании таблицы, и будет именем файла, в котором таблица сохраняется на диске. При задании этого имени необходимо придерживаться ограничений, накладываемых операционной системой на количество символов в имени файла. Наименование таблицы может содержать буквы, цифры и знак подчёркивания. Создавая новую таблицу, необходимо помнить, что в базе данных не может быть двух таблиц с одинаковыми именами. Если в базе данных уже имеется таблица с таким именем, на экране появляется запрос, заменить ли существующую таблицу новой.

Второе имя таблицы является внутренним и хранится в базе данных. Внутреннее имя таблицы может содержать до 128 символов. Оно вводится в поле *Name* на вкладке *Table* в верхней части окна конструктора таблицы. Это имя будет отображаться в окне проекта, а также использоваться при создании форм, запросов и отчётов. Используя поле *Table Comment* вкладки *Table*, можно ввести описание таблицы. Для определения условия проверки правильности ввода информации на уровне записей, гарантирующих достоверность данных, вводимых в таблицу, используются поля области *Record validation*. Для проверки значений при добавлении, изменении и удалении записей таблицы предназначены поля области *Triggers: Insert trigger, Update trigger, Delete trigger*.

2 ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

Лабораторная работа выполняется в среде реляционной СУБД Visual FoxPro. Для выполнения лабораторной работы необходимо:

1) Запустить СУБД Visual FoxPro;

2) Повторить основной материал по технологии создания базы данных (см. список литературы, конспект лекций, справочную систему Visual FoxPro);

3) Изучить описание данной лабораторной работы, выполняя все указанные действия на компьютере в среде СУБД Visual FoxPro;

4) Проверить в системе наличие таблиц данных и других компонентов, которые будут включены в базу данных;

5) Используя командное окно и конструктор БД создать базу данных, включающую две таблицы и одно представление (одну таблицу добавить в БД, вторую таблицу создать с учетом дополнительных возможностей - задать триггеры, индексы и т.д.);

6) Сохранить БД и поработать с ней, выполняя просмотр, редактирование, модификацию структуры таблиц и т.д.;

7) Оформить отчет по работе.

Контрольные вопросы:

1) Дать понятие иерархической модели данных.
2) Перечислить достоинства и недостатки иерархической модели данных.
3) Дать понятие сетевой модели данных.

Список литературы, рекомендуемый к использованию по данной теме:
Основная: 1-11
Дополнительная: 1-11
Интернет-ресурсы: 1-7
Программное обеспечение: 1-4

Лабораторная работа № 3-4

Тема 3 - 4. Подходы к организации баз данных. Общие понятия реляционного подхода к организации БД

1. Цель работы: изучение технологии создания отчетов с использованием конструктора отчетов

Формируемые компетенции: ПК-1, ПК-3

2. Подготовка к занятию

1. Изучить (повторить) теоретический материал.
2. Ознакомиться с программой работы.
3. Подготовить отчет о работе.
4. Ответить на контрольные вопросы.

3. Распределение времени занятия:

Всего: 90 мин

Вступительная часть 2 мин

Проверка готовности студентов к занятию 5 мин

Основная часть – 70 мин

Проверка выполнения практического занятия 10 мин

Заключительная часть 3 мин

4. Правила работы в лаборатории

К работе в лаборатории допускаются лица, изучившие правила и меры безопасности, сдавшие зачет по ним и усвоившие порядок выполнения работы.

4.1. Требования безопасности перед началом работ

- ЗАПРЕЩАЕТСЯ: переодеваться, пользоваться огнем, курить, принимать пищу в лаборатории.

- Убедиться в целостности электрических розеток и разъемов. В лаборатории необходимо быть в сменной обуви.

- Включение компьютера производить только после получения допуска по выполняемой работе и разрешения преподавателя или лаборанта.

4.2. Требования безопасности во время работы

- выполняя практическое занятие, студенты обязаны использовать только вычислительную технику, периферийное оборудование, соединительные кабели, измерительное оборудование и носители информации, непосредственно относящиеся к данному лабораторному занятию;

- подключение и отключение составляющих вычислительного комплекса производить только при полном снятии напряжения со всех составляющих вычислительного комплекса;

- при обнаружении неисправностей в оборудовании немедленно отключить источники питания и доложить об этом руководителю занятий или лаборанту.

4.3. Требования безопасности по окончании работы

- доложить руководителю занятий или лаборанту о завершении работ;
- привести в порядок и сдать рабочее место лаборанту, и доложить руководству

1 ОСНОВНЫЕ СВЕДЕНИЯ О ТЕХНОЛОГИИ СОЗДАНИЯ ОТЧЕТОВ

Отчет — это гибкое и эффективное средство для организации данных при выводе на печать. С помощью отчета имеется возможность вывести необходимые сведения в том виде, в котором требуется. Отчет является объектом Visual FoxPro, предназначенным для представления данных, отбираемых и форматируемых в соответствии с заданными пользователем спецификациями. С помощью отчетов печатаются сводки продаж, накладные, различные ведомости, статистические отчеты, телефонные справочники и другие документы

Отчеты можно создавать на основе таблиц и представлений, также в отчет можно направить результат выполнения запроса SQL.

Существуют следующие типы отчетов:

- табличный (по столбцам);
- отчет в свободной форме (например, письмо которое рассылается разным адресатам, взятым из БД);
- стандартный отчет;
- отчет с упорядочением данных по возрастанию или убыванию;
- отчет с группировкой данных по категориям;
- многоколоночный отчет.

В качестве визуального инструментария при создании отчета используется *Конструктор отчетов* (Report Designer).

Для создания связи между отчетом и его исходными данными применяются следующие элементы управления:

- заголовок отчета;
- заголовок каждой страницы отчета;
- номер страницы отчета, используя системную переменную _PAGENO( );
- дату распечатки отчета, используя системную функцию DATE( );
- верхний и нижний колонтитулы столбца;
- значения полей, переменных, функций, элементов массива, вычисляемых полей;
- любой текст, рисунки, линии, прямоугольники.

Для создания отчета необходимо открыть окно конструктора отчетов, воспользовавшись системным меню Visual FoxPro. Для этого в меню *File* (Файл) выбирается команда *New* (Новый). В открывшемся диалоговом окне New необходимо установить опцию *Report* (Отчет) и нажать кнопку *New file*. После выполнения данных действий откроется *окно конструктора отчета* (Report Designer), которое содержит панель инструментов *Report Controls* (Элементы управления отчета).

Панель инструментов *Report Controls* (Элементы управления отчета) используется для размещения следующих объектов:

 -позволяет разместить в отчете любой текст;

 -позволяет отображать в отчете данные из полей таблиц, вычисляемые значения, значения переменных, элементов массива;

 - позволяет представить в отчете горизонтальные/вертикальные линии;

 -позволяет выделить информацию в отчете прямоугольной рамкой;

 -позволяет выделить информацию в отчете рамкой с закругленными краями;

 -позволяет вставить в отчет OLE объект или содержимое поля General.

Каждый *объект* помещается в Конструктор отчетов на ту или иную полосу путем выбора этого объекта на панели инструментов щелчком мыши. Щелкнув мышкой в окне Конструктора отчетов и не отпуская ее, нужно провести по диагонали, чтобы определить месторасположение объекта. Объекты можно перетаскивать, изменять размеры, подписи. Когда объект находится на полосе *Report Designer*, двойным щелчком можно вызвать дополнительное окно, где можно задать свойства этого объекта.

Рабочая область Конструктора отчетов по умолчанию разделена на три полосы. При использовании в отчете группирования данных, итоговых данных и добавления в него титульной страницы появляются дополнительные полосы. Типы возможных полос:

- Title (Титул) – служит для размещения информации, появляющейся перед основным отчетом;
- Page Header (Верхний колонтитул) – служит для задания верхнего колонтитула;
- Group Header (Группа сверху) – служит для размещения информации, используемой при группировке;
- Detail (Детали) – служит для размещения полей из таблиц или результат вычисления над ними;
- Group Footer (Группа снизу) – служит для размещения итоговой информации по группе;

— Page Footer (Нижний колонтитул) – служит для указания нижнего колонтитула страницы;

— Summary (Итоги) - служит для размещения информации, появляющейся после основного отчета.

Для создания стандартного отчета (Рисунок 1) необходимо выполнить команду *Quick Report* из меню *Report*. В появившемся окне стандартного запроса файла нужно выбрать ту таблицу или представление, из которого будут взяты данные.

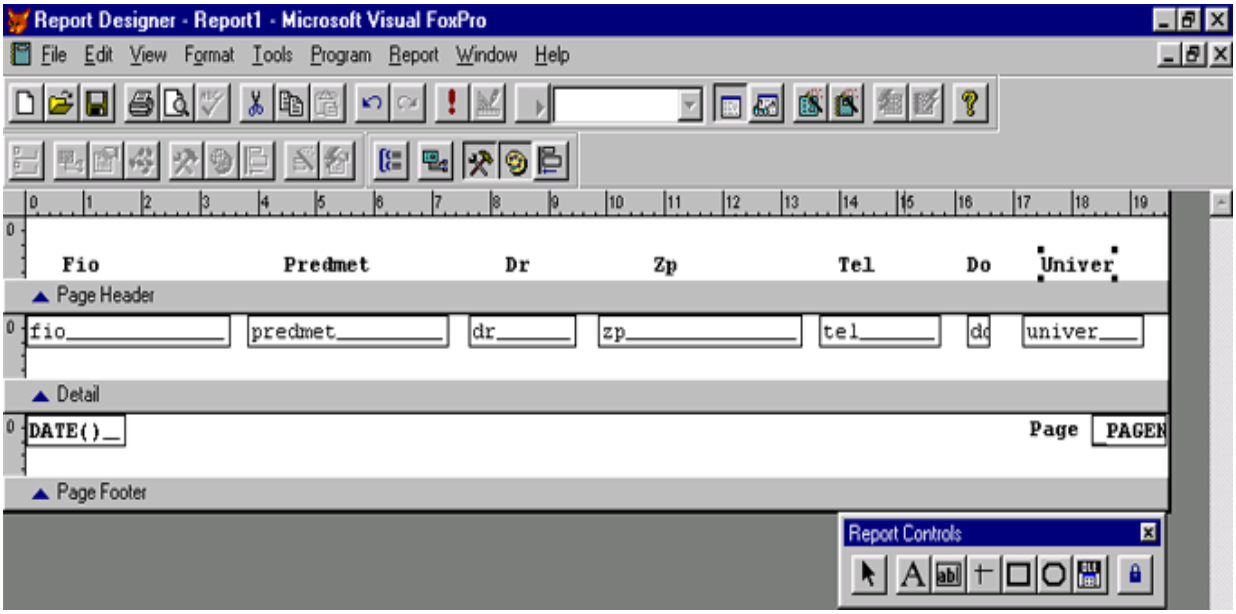


Рисунок 1 – Окно конструктора отчетов

Открывшееся окно *Quick Report* (Рисунок 2) позволяет выбирать расположение полей и опции:

☑*Titles* - добавить заголовки полей;

☑*Add alias* - добавить псевдонимы полей;

☑*Add table to data environment* - добавить таблицу в среду окружения;

Fields…- выбрать нужные поля из таблицы(по умолчанию все).

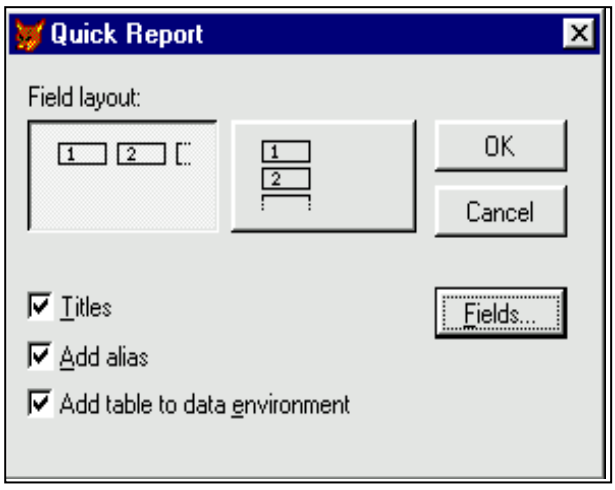


Рисунок 2 – Окно Quick Report (Стандартный отчет)

Quick Report содержит в полосе *Detail* выбранные поля из таблицы, в полосе *Page Header*-заголовки этих полей, а в полосе *Page Footer* - поле с функцией *DATE()* и поле с переменной *_PAGE( )* и словом *Page* (номер страницы). Этот отчет можно модифицировать.

Для добавления в отчет суммарного итога необходимо использовать команду *Title/Summary* из меню *Report*. Опции окна *Title/Summary* (Рисунок 3) позволяют:

☑*Title band* – печатать полосу заголовка;

☑*New page* - печатать полосу заголовка на каждой странице;

☑*Summary band* - печатать полосу суммарного итога;

☑*New page* - печатать полосу суммарного итога на каждой странице.

Рисунок 3 - Окно Title/Summary (Заголовок/Суммарный итог)

После этого в *окне Конструктора Отчетов* появляются две полосы:

— Title – здесь с помощью кнопки  пишется заголовок;

— Summary – - здесь с помощью кнопки  выбирается то поле, по которому будет подводиться итог. В появившемся окне *Report Expression* (Рисунок 4) нужно выбрать необходимое поле суммирования и с помощью кнопки *Calculations…* , в открывшемся окне *Calculate Field* (Рисунок 5) назначить необходимую стандартную функцию

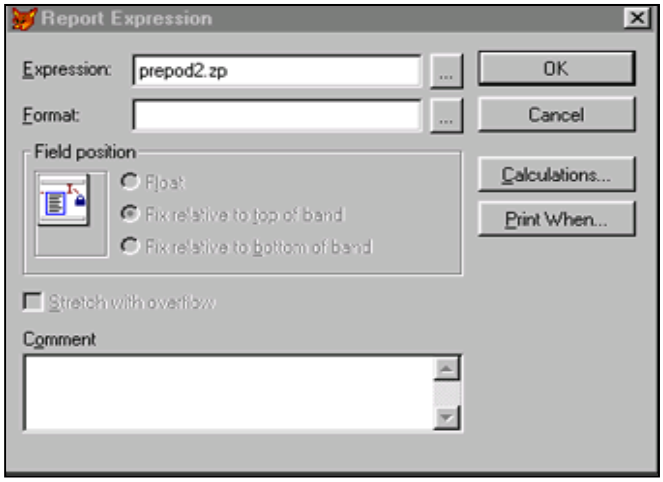


Рисунок 4- Окно Report Expression(Выражение отчета)

Окно Report Expression (Рисунок 4) позволяет:

— Expression (Выражение) – определить выражение, результат вычисления которого будет выводиться в данное поле;

— Format (Формат) - задать формат отображения данных в поле;

— Field position (Положение поля) – задать расположение поля в полосе.

Используя опции окна *Calculate Field* (Вычисляемое поле), открываемого при нажатии кнопки *Calculations* (Вычисления) (Рисунок 4) в отчет можно поместить статистические данные, размещенные в полях данных (Рисунок 5).

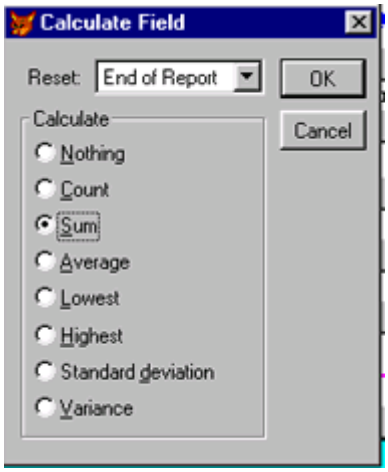


Рисунок 5 - Окно Calculate Field(Вычисляемое поле)

Для вывода упорядоченных данных можно воспользоваться командой *Data Grouping* в меню *Report* или в контекстно-зависимом меню, щелкнув правой кнопкой мыши на полосе отчета.

Чтобы выводить информацию в упорядоченном виде, нужно предварительно проиндексировать или отсортировать таблицу по соответствующему полю. Можно осуществить вложенную группировку до 128 уровней. В открывшемся окне *Data Grouping* (Рисунок 6) нужно выбрать параметры поля группировки.

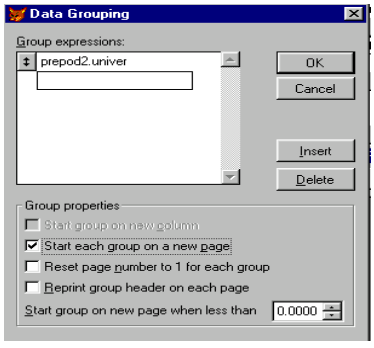


Рисунок 6 – Окно Data Grouping (Группировка данных)

Область *Group expressions* (Выражение группировки) *Data Grouping* позволяет задать выражение, по которому будут группироваться данные.

Область *Group properties* (Свойства группировки) окна *Data Grouping* позволяет задать параметры группировки:

☑ *Start each group on new page* – начинать каждую группу на новой странице;

☑ *Reset page number to 1 for each group* – сбросить номер страницы к 1 для каждой группы;

☑ *Reprint group header on each page* – печатать заголовок группы на каждой странице;

☑ *Start group on new page when less than* – печатать группу на новой странице, если под заголовком группы остается расстояние меньше указанного в данном поле.

Кнопки *Insert/Delete* – позволяют вставить/удалить поля группировки.

После задания всех параметров появляются две полосы: *Group Header* – здесь задается заголовок каждой группы и/или имя поля, по которому группируются данные, *Group Footer* – здесь задается итог по каждой группе и/или имя поля, по которому группируются данные.

После того как отчет сформирован можно отправить его на предварительный просмотр командой *Preview* меню *View* . Для просмотра многостраничного отчета, установления масштаба, вывода на принтер, выхода в режим Конструктора служит панель предварительного просмотра (Рисунок 7). Также для печати отчета на принтере можно воспользоваться командой *Run Report* из меню *Report*.

Рисунок 7-Панель предварительного просмотра

2 ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

Лабораторная работа выполняется в среде реляционной СУБД Visual FoxPro. Для выполнения лабораторной работы необходимо:

1) Запустить СУБД Visual FoxPro;

2) Повторить основной материал по технологии создания отчета (см. список

литературы, конспект лекций, справочную систему Visual FoxPro);

3) Изучить описание данной лабораторной работы, выполняя все указанные действия на компьютере в среде СУБД Visual FoxPro;

4) Проверить в системе наличие таблиц данных, которые будут использованы при создании отчетов;

5) Создать отчет в свободной форме (в виде письма, представленного на рисунке 8):

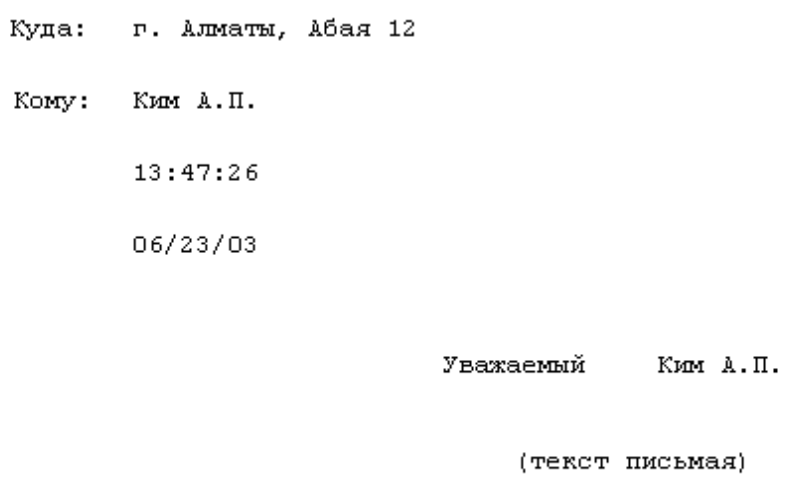


Рисунок 8 – Образец оформления письма

При создании отчета использовать стандартные функции *xBase*.

Например: alltrim(city)+' '+alltrim(address).

Данные о получателе письма взять из таблицы, каждое письмо на новой странице;

6) Создать табличный отчет с заголовком и итогами, используя команду Title/Summary. В поле Summary разместить объекты TextBox, для вывода суммы значения числового поля и среднего значения числового поля, наименьшего и наибольшего общего значений полей таблицы, общего числа записей;

7) Создать отчет с вычисляемым полем, используя объект TextBox, для которого задается выражение, вычисляющее значение функции, например:

Price*Count или Price1- Price2 (Price, Price1, Price2, Count – названия полей таблицы);

8) Создать многоколоночный отчет с двумя колонками: создается образец отчета, затем выполняется команда *PageSetup* из меню *File*, в появившемся диалоговом окне *Page Setup* в области *Columns* (Колонки) определяются размеры колонок и их количество *Number=2* ;

9) Оформить отчет по работе.

Контрольные вопросы:

1)Перечислить операции реляционной алгебры.

2)Пояснить принцип реляционного исчисления с переменными-кортежами.

3)Пояснить принцип реляционного исчисления с переменными на доменах.

4)Пояснить понятие функциональных зависимостей.

5)Пояснить правила вывода функциональных зависимостей.

6)Пояснить понятие избыточных функциональных зависимостей.

Список литературы, рекомендуемый к использованию по данной теме:
Основная: 1-11
Дополнительная: 1-11
Интернет-ресурсы: 1-7
Программное обеспечение: 1-4

Лабораторная работа № 5-6

Тема 5 - 6. Базисные средства манипулирования реляционными данными. Проектирование реляционных БД

1. Подготовка к занятию

1. Изучить (повторить) теоретический материал.
2. Ознакомиться с программой работы.
3. Подготовить отчет о работе.
4. Ответить на контрольные вопросы.

Формируемые компетенции: ПК-1, ПК-3

2. Распределение времени занятия:

Всего: 90 мин

Вступительная часть 2 мин

Проверка готовности студентов к занятию 5 мин

Основная часть – 70 мин

Проверка выполнения практического занятия 10 мин

Заключительная часть 3 мин

4. Правила работы в лаборатории

К работе в лаборатории допускаются лица, изучившие правила и меры безопасности, сдавшие зачет по ним и усвоившие порядок выполнения лабораторной работы.

4.1. Требования безопасности перед началом работ

- ЗАПРЕЩАЕТСЯ: переодеваться, пользоваться огнем, курить, принимать пищу в лаборатории.

- Убедиться в целостности электрических розеток и разъемов. В лаборатории необходимо быть в сменной обуви.

- Включение компьютера производить только после получения допуска по выполняемой работе и разрешения преподавателя или лаборанта.

4.2. Требования безопасности во время работы

- выполняя практическое занятие, студенты обязаны использовать только вычислительную технику, периферийное оборудование, соединительные кабели, измерительное оборудование и носители информации, непосредственно относящиеся к данному лабораторному занятию;

- подключение и отключение составляющих вычислительного комплекса производить только при полном снятии напряжения со всех составляющих вычислительного комплекса;

- при обнаружении неисправностей в оборудовании немедленно отключить источники

питания и доложить об этом руководителю занятий или лаборанту.

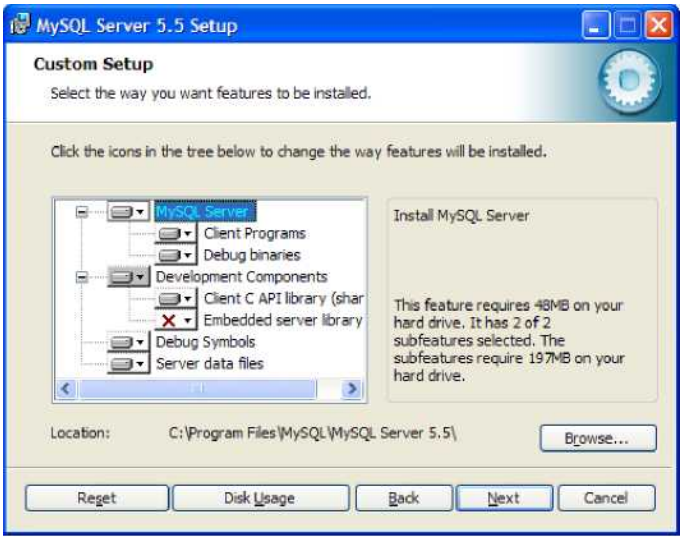
4.3. Требования безопасности по окончании работы

- доложить руководителю занятий или лаборанту о завершении работ;
- привести в порядок и сдать рабочее место лаборанту, и доложить руководству

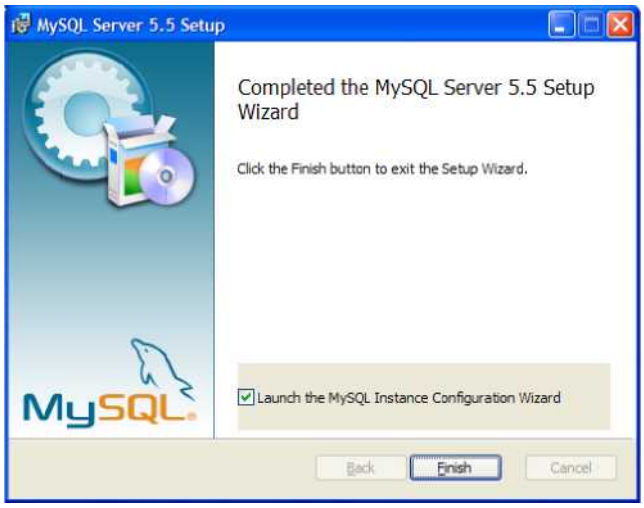
Установка и настройка MySQL Server

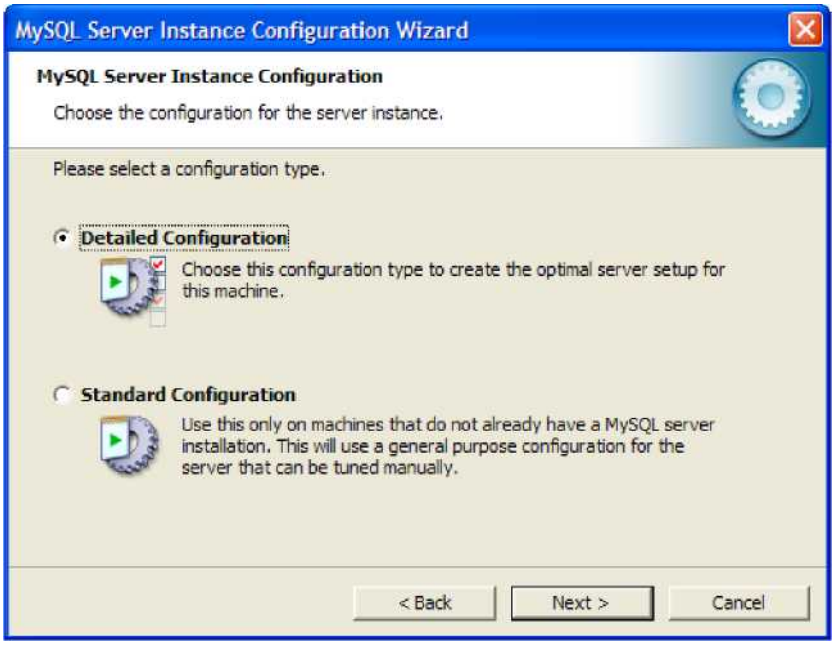
Для установки программы необходимо запустить инсталляционный файл программы.

Нажмите в данном окне выборочную установку компонентов "Custom"



Здесь вы можете выбрать дополнительные компоненты и сменить установочную директорию программы.





Выбираем детализированную настройку - "Detailed Configuration".

MySQL Server Instance Configuration

Choose the configuration for the server instance.

:r type, This will influence Plecmemory, disk and CPU usage.

(I Developer Machine

1 d ^'s's a development machine, and many other applications will be I run on it. MySQL Server should only use a minimal amount of memory.

C" Server Machine

Теперь приступим к настройке MySQL сервера.

d Several server applications will be running on this machine. Choose this option for web/application servers. MySQL will have medium memory usage,

C Dedicated MySQL Server Machine

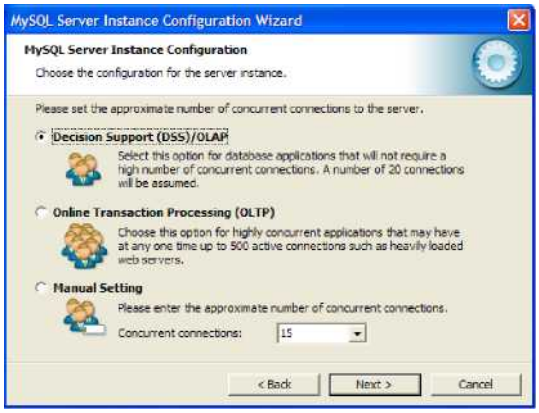
d This machine is dedicated to run the MySQL Database Server. No other servers, such as a web or mail server, will be run. MySQL will utilize up to all available memory.

Отмечаем пункт "Developer Machine

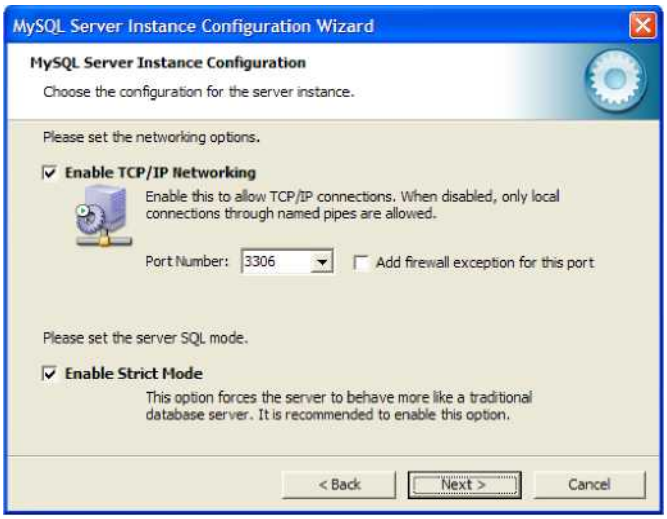
Выбрав пункт "Multifunctional Database", вы сможете работать как с таблицами типа InnoDB (с возможностью использования транзакций), так и с высокоскоростной MyISAM (как правило для веб-разработок используется именно этот тип таблиц).

Далее необходимо осуществить выбор диска и директории для хранения таблиц типа InnoDB.

В данном диалоговом окне выбирается максимально возможное количество подключений к серверу MySQL.



При выборе "Decision Support (DSS)/OLAP", максимальное количество подключений будет ограничено двадцатью, чего более чем достаточно при установке сервера на домашнем компьютере и отсутствии большого количества одновременных подключений.

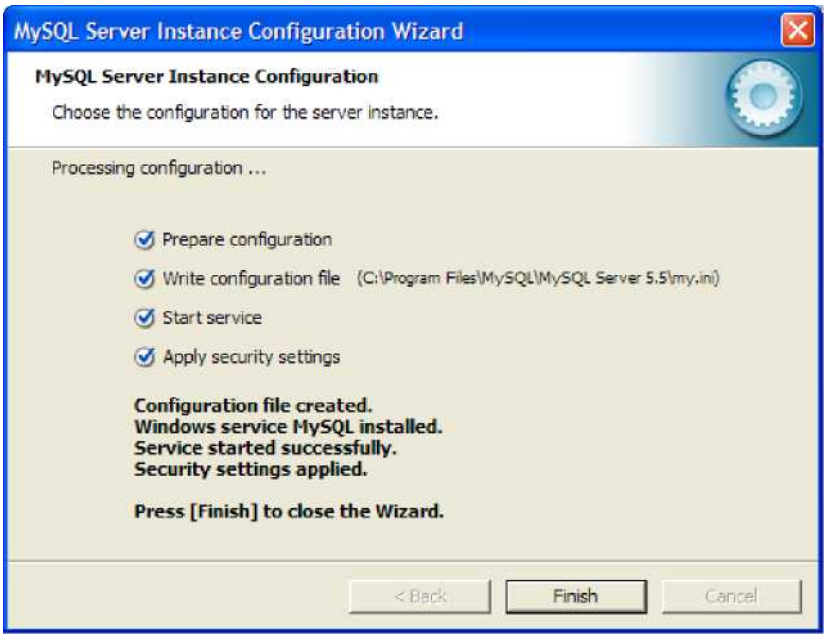


Отметив "Enable TCP/IP Networking" мы включаем поддержку TCP/IP соединений и выбираем порт, через который они будут осуществляться. Стандартным для сервера MySQL является порт 3306. Отметив "Enable Strict Mode", мы задаем режим строгого соответствия стандарту SQL (данную опцию рекомендуется оставлять включенной).

Обратите внимание на выставление настроек данного окна. Отметив "Manual Selected Default Character Set / Collation" и выбрав из ниспадающего меню "cp1251" определяем, что изначально для таблиц будет использоваться кодировка Cyrillic Windows (cp1251), что означает корректную работу с русским языком в данной кодировке.

Если отметить "Install As Windows Service", сервер будет запускаться в виде сервиса, что является рекомендуемым способом запуска. Ниже, в ниспадающем списке, задается имя сервиса. Далее, уберите галочку рядом с "Launch the MySQL Server automatically" - мы будем запускать сервер вручную. Также поставьте галочку рядом с "Include Bin Directory in Windows PATH" - это позволит установить видимость директории "bin", для командной строки.

Установите пароль пользователя "root". Не оставляйте поле пустым, это убережёт вас от возможных неприятностей в дальнейшем.



В данном окне обратите внимание на строку "Write configuration file", которая указывает на месторасположение конфигурационного файла MySQL - "my.ini", далее, его необходимо будет немного отредактировать.

Откройте для редактирования файл "my.ini".

В раздел [client], после строки: port=3306 добавьте строку определяющую каталог содержащий файлы описания кодировок:

character-sets-dir="C:/Program Files/MySQL/MySQL Server 5.5/share/charsets"

В раздел [mysqld], после строки: port=3306 добавьте следующие две строки, первая из которых вам уже известна, вторая - устанавливает кодировку в которой данные передаются MySQL:

character-sets-dir="C:/Program Files/MySQL/MySQL Server 5.5/share/charsets" init-connect-'SET NAMES cp1251"

Далее, найдите строку: default-storage-engine-INNODB

Замените изначально устанавливаемый тип таблиц на MYISAM: default-storage-engine-MYISAM.

Сохраните изменения и закройте файл "my.ini". Установка и настройка сервера MySQL - завершена.

Контрольные вопросы:

1) Пояснить, каким способом возможен запуск серверной части СУБД MySQL.
2) Дать определение привилегии и пояснить её предназначение.

3) Перечислить основные утилиты, входящие в состав СУБД, какие функции они выполняют.

4) Перечислить принципы использования кортежных переменных

5) Дать понятие минимального покрытия и декомпозиции отношений.

6) Перечислить принципы использования переменных в доменах

Задания творческого уровня:

7) Запустите сервер MySQL. Зарегистрируйте своего пользователя в консольном приложении, задайте ему права.

8) С помощью утилиты Mysqlshow выполните команду на просмотр структуры и состав таблиц базы Mysql.

9) С помощью утилиты Mysqldump получите полный дамп базы Mysql (данные и таблицы), а также отдельные дампы таблиц и данных.

Список литературы, рекомендуемый к использованию по данной теме:
Основная: 1-11
Дополнительная: 1-11
Интернет-ресурсы: 1-7
Программное обеспечение: 1-4

Раздел 2. Безопасность информации в базах данных

Лабораторная работа № 7-8

Тема 7-8. Структуры внешней памяти в реляционных СУБД. Управление параллельностью работы транзакций

1. Цель работы: ознакомиться с возможностями СУБД MySQL и создать с его помощью базу данных, набор таблиц в ней и заполнить таблицы данными для последующей работы.

Формируемые компетенции: ПК-1, ПК-3

2. Подготовка к занятию

1. Изучить (повторить) теоретический материал.
2. Ознакомиться с программой работы.
3. Подготовить отчет о работе.
4. Ответить на контрольные вопросы.

3. Распределение времени занятия:

Всего: 90 мин

Вступительная часть 2 мин

Проверка готовности студентов к занятию 5 мин

Основная часть – 70 мин

Проверка выполнения практического занятия 10 мин

Заключительная часть 3 мин

4. Правила работы в лаборатории

К работе в лаборатории допускаются лица, изучившие правила и меры безопасности, сдавшие зачет по ним и усвоившие порядок выполнения лабораторной работы.

4.1. Требования безопасности перед началом работ

- ЗАПРЕЩАЕТСЯ: переодеваться, пользоваться огнем, курить, принимать пищу в лаборатории.

- Убедиться в целостности электрических розеток и разъемов. В лаборатории необходимо быть в сменной обуви.

- Включение компьютера производить только после получения допуска по выполняемой работе и разрешения преподавателя или лаборанта.

4.2. Требования безопасности во время работы

- выполняя практическое занятие, студенты обязаны использовать только вычислительную технику, периферийное оборудование, соединительные кабели, измерительное оборудование и носители информации, непосредственно относящиеся к данному лабораторному занятию;

- подключение и отключение составляющих вычислительного комплекса производить только при полном снятии напряжения со всех составляющих вычислительного комплекса;

- при обнаружении неисправностей в оборудовании немедленно отключить источники питания и доложить об этом руководителю занятий или лаборанту.

4.3. Требования безопасности по окончании работы

- доложить руководителю занятий или лаборанту о завершении работ;
- привести в порядок и сдать рабочее место лаборанту, и доложить руководству

Содержание работы и методические указания к ее выполнению

1. Ознакомиться с возможностями работы клиентского приложения MySQL .

2. Изучить набор команд языка SQL, связанный с созданием базы данных, созданием, модификацией структуры таблиц и их удалением, вставкой, модификацией и удалением записей таблиц.

Функция	**Описание**
CREATE DATABASE DB_NAME	создание базы данных
USE DATABASE	выбор существующей базы данных
CLOSE DATABASE	закрытие файлов текущей базы данных
DROP DATABASE	удаление базы данных
CREATE TABLE	создание таблицы базы данных
ALTER TABLE	модификация структуры базы данных
DROP TABLE	удаление таблицы базы данных
INSERT	добавление одной или нескольких строк в таблицу
DELETE	удаление одной или нескольких строк из таблицы
UPDATE	модификация одной или нескольких строк таблицы
LOAD DATA INFILE	загрузка данных в таблицы из файла

3. Создать базу данных.

Создание базы данных в MySQL производится с помощью утилиты mysqladmin. Изначально существует только БД mysql для администратора и БД test, в которую может войти любой пользователь и которая по умолчанию пуста. Приведенный ниже пример иллюстрирует создание базы данных.

```
Mysql/bin>mysqladmin -u root -p create data_name
Enter password:******
Database "data_name" created.
```

```
mysqlbin>
```

Где data_name – имя создаваемой БД. Проверить, что БД создана можно ранее рассмотренной командой ***Show databases*** или утилитой **mysqlshow.**

По умолчанию, **root** имеет доступ ко всем базам данных и таблицам. Перейти в созданную базу данных можно, используя команду **mysql**. Use database

```
Mysql/bin>mysql -u root -p data1
Enter password:******

Welcome to MySQL monitor.
```

Или, находясь в другой базе данных, например в mysql ввести команду:

```
mysql>use data1

Database changed.
```

Создать базу данных можно непосредственно находясь в клиентском приложении MySQL, вводом команды:

CREATE DATABASE Base_name

Где **Base_name** имя создаваемой базы данных. В созданной базе можно создавать таблицы и вводить информацию. Указанные операции можно выполнить, используя специализированное программное обеспечение, например MySQL-Front, запуск которого осуществляется из меню ПУСК/ПРОГРАММЫ (см. рис 1, 2).

Необходимо указать:

- Имя;
- Хост;
- Пароль;
- Порт;
- Имя БД (при необходимости).

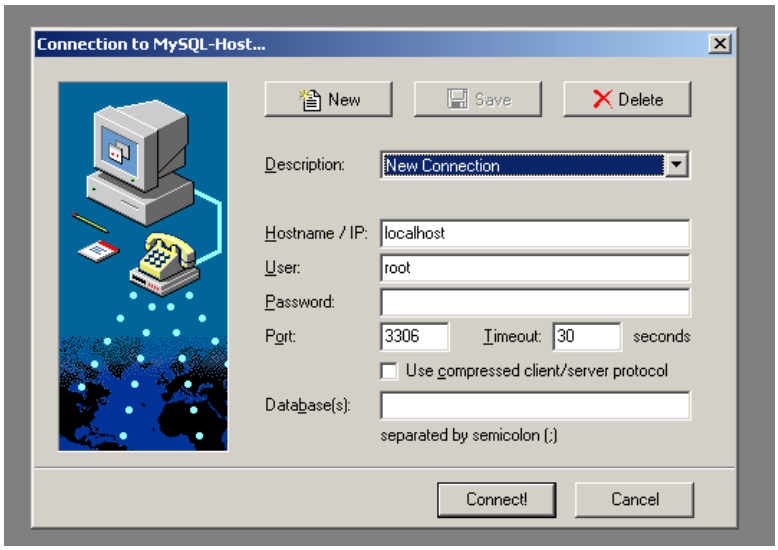


Рисунок 1 - Запуск MySQL-front

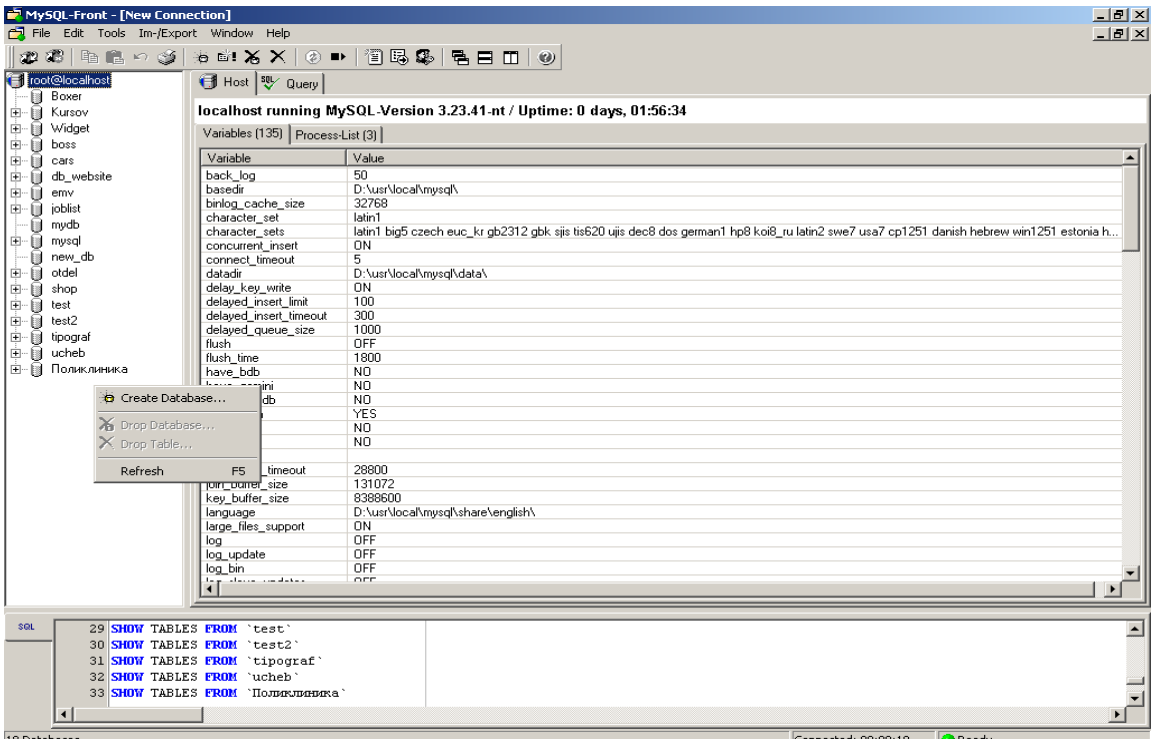


Рисунок 2 - Создание БД в среде MySQL-front

После задания активной БД можно с помощью средств, предоставляемых программой изменять структуру БД, вводить данные, задавать ключевые поля. Помимо этого можно в специально отведенном окне напрямую вводить инструкции, используя синтаксис языка SQL, как показано на рисунке:

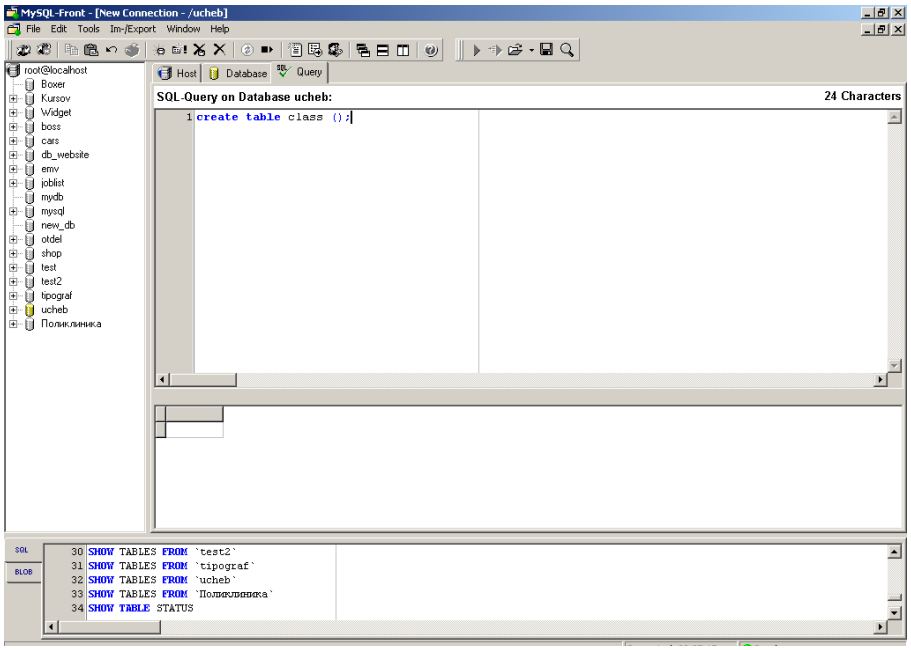


Рисунок 3 - Использование синтаксиса SQL

4. Средствами языка SQL необходимо создать четыре таблицы в базе данных, используя команду CREATE TABLE, синтаксис которой приведен в приложении. Для таблицы J:

```
CREATE TABLE j (
  Jnum varchar(6) NOT NULL default '',
  Jnam varchar(20) default NULL,
  Ci varchar(20) default NULL,
  PRIMARY KEY  (Jnum)
) TYPE=MyISAM;
```

Значками /* */ - выделяются комментарии в тексте запроса.

При создании таблиц выполнить такую реализацию, чтобы она отражала структуру таблиц, указанную ниже (таблице S, P, J, SPJ) и должны быть наложены следующие ограничения:

- поля номер_поставщика, номер_детали, номер_изделия во всех таблицах имеет символьный тип и длину 6 (**varchar**(6));

- поля рейтинг, вес и количество имеют целочисленный тип (**integer)**;

- поля фамилия, город (поставщика, детали или изделия), название (детали или изделия) имеют символьный тип и длину 20 (**varchar**(20));

- ни для одного поля не предусматривается использование индексов;

-для всех полей допускаются значения NULL и значения-дубликаты, кроме полей первичного и внешнего ключей.

После создания пустых таблиц их необходимо наполнить данными. Вводить данные в нее можно несколькими способами:

а)Вручную, используя команду **insert into**;

Пример ввода данных вручную (команда INSERT):

mysql>***insert into J (Jnum, Jnam, Ci)values ('J1','Жесткий диск','Париж')***;
или
mysql>***insert into J values ('J1','Жесткий диск','Париж');***

//т.е в случае если вы вставляете данные во все поля таблицы то их перечислять не обязательно.

Таким образом SQL инструкция имеет следующий вид

INSERT INTO table_name (id, name) VALUES ('id_value', 'name_value');

Записать и выполнить совокупность запросов для занесения нижеприведенных данных в созданные таблицы

insert into имя_таблицы [(поле [,поле]...)] values (константа [,константа]...)

б)Загрузить данные из текстового файла, что является более предпочтительным, особенно если нужно ввести несколько тысяч записей.

Синтаксис команды LOAD DATA INFILE.

DATA [LOW_PRIORITY] [LOCAL] INFILE 'file_name.txt' [REPLACE | IGNORE]
INTO TABLE tbl_name
[FIELDS
[TERMINATED BY 't']
[OPTIONALLY] ENCLOSED BY '']
[ESCAPED BY '']]
[LINES TERMINATED BY 'n']
[IGNORE number LINES]
[(col_name,...)]

Пример:

LOAD DATA LOCAL INFILE '/MyDocs/categories.txt' REPLACE INTO TABLE category FIELDS TERMINATED BY ';' OPTIONALLY ENCLOSED BY '\"' LINES TERMINATED BY '\n'

В данном случае файл ***categories.txt*** находится на машине под управлением MS Windows, в каталоге ***C:\MyDocs***.

Обратите внимание на UNIX стиль написания пути. Слово

REPLACE

В SQL запросе означает, что необходимо замещать записи с совпадающими значениями ключей.

INTO TABLE[1]

указывает имя таблицы, куда будут импортированы данные.

FIELDS TERMINATED BY ';' [2]

указывает разделители полей, порядок полей должен быть таким же, как и в таблице назначения,

OPTIONALLY ENCLOSED BY '\"'

указывает, что поля VARCHAR взяты в двойные кавычки, и

LINES TERMINATED BY '\r' [3]

в) Использовать утилиту mysqlimport также для загрузки данных из текстового файла.

Эти и другие операции можно выполнить также и в программе MySQL-Front.

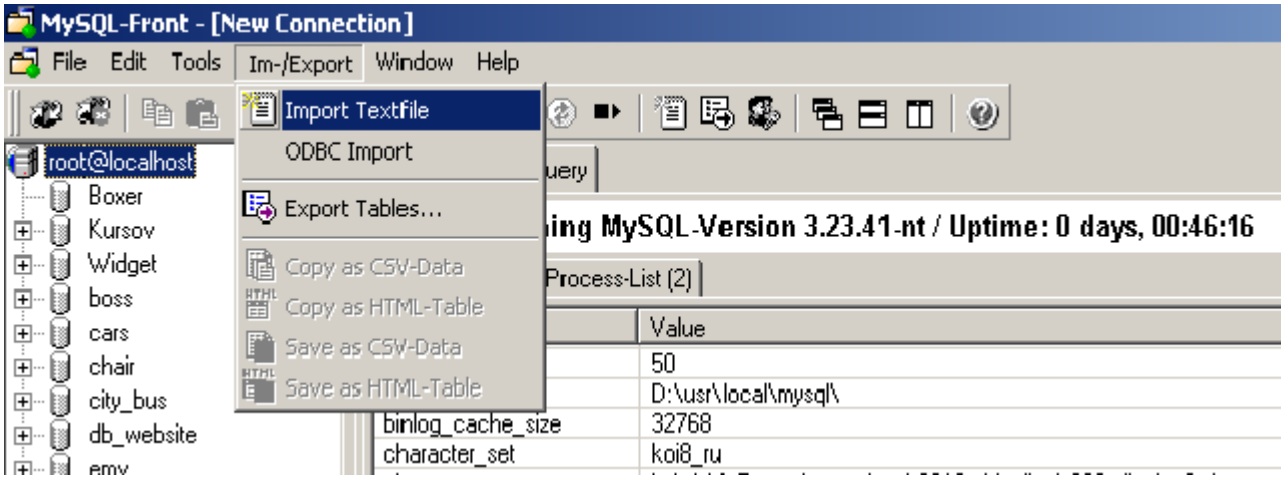


Рисунок 4 - Использование программы MySQL-front для заполнения таблиц данными из файла

---

[1] В случае если данные вставляются не во все ячейки таблицы то это указывается при формировании инструкции tablename(id, id2), где tablename – имя таблицы, а id, id2 наименования атрибутов таблицы.

[2] В случае использования табулятора \t

[3] В случае Enter \r\n

Таблица поставщиков (S)

Номер поставщика	**Фамилия**	**Рейтинг**	**Город**
S1	Смит	20	Лондон
S2	Джонс	10	Париж
S3	Блейк	30	Париж
S4	Кларк	20	Лондон
S5	Адамс	30	Афины

Таблица деталей (P)

Номер детали	**Название**	**Цвет**	**Вес**	**Город**
P1	Гайка	Красный	12	Лондон
P2	Болт	Зеленый	17	Париж
P3	Винт	Голубой	17	Рим
P4	Винт	Красный	14	Лондон
P5	Кулачок	Голубой	12	Париж
P6	Блюм	Красный	19	Лондон

Таблица изделий (J)

Номер изделия	**Название**	**Город**
J1	Жесткий диск	Париж
J2	Перфоратор	Рим
J3	Считыватель	Афины
J4	Принтер	Афины
J5	Флоппи-диск	Лондон
J6	Терминал	Осло
J7	Лента	Лондон

Таблица поставок (SPJ)

Номер поставщика	Номер детали	Номер изделия	Количество
S1	P1	J1	200
S1	P1	J4	700
S2	P3	J1	400
S2	P3	J2	200
S2	P3	J3	200
S2	P3	J4	500
S2	P3	J5	600
S2	P3	J6	400
S2	P3	J7	800
S2	P5	J2	100
S3	P3	J1	200
S3	P4	J2	500
S4	P6	J3	300
S4	P6	J7	300
S5	P2	J2	200
S5	P2	J4	100
S5	P5	J5	500
S5	P5	J7	100
S5	P6	J2	200
S5	P1	J4	100
S5	P3	J4	200
S5	P4	J4	800
S5	P5	J4	400
S5	P6	J4	500

Убедиться в успешности выполненных действий. При необходимости исправить ошибки. Для ускорения процесса ввода данных рекомендуется воспользоваться командой LOAD DATA (синтаксис см. в приложении), предварительно скопировав содержимое перечисленных таблиц сначала в Excel, а оттуда в текстовые файлы. Такой порядок необходим, для того, чтобы текстовый файл был с табуляцией.

5. Выполнить модификацию структуры таблицы SPJ, добавив в SPJ поле с датой поставки. Убедиться в успешности выполненных действий. При необходимости исправить ошибки (команда Alter table).

6. Уничтожить созданные таблицы, предварительно сохранив инструкции для восстановления структуры БД и информационного наполнения, используя средства работы СУБД[4]. Убедиться в успешности выполненных действий.

7. Выполнить необходимые действия, написав и выполнив соответствующие запросы для модификации таблиц, чтобы структура соответствовала концептуальной модели учебной базы данных (рисунок 5). Убедиться в успешности выполненных действий. При необходимости исправить ошибки.

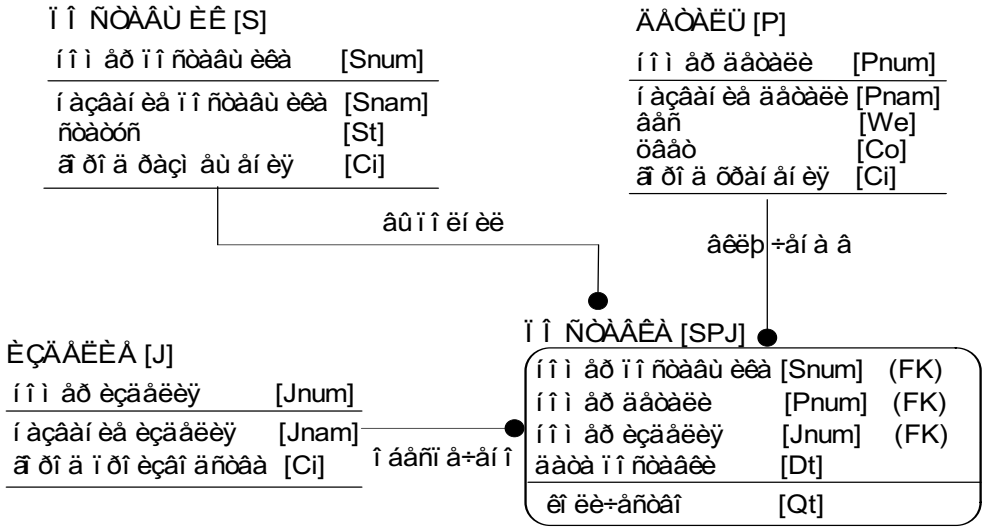


Рисунок 5 - Концептуальная модель учебной базы данных

Проверить результат заполнения таблиц, написав и выполнив простейший запрос:

select * from имя_таблицы

При наличии ошибок выполнить корректировку, исправив либо удалив ошибочные строки таблиц

[4] См утилиту Mysqldump

Контрольные вопросы

1. В каких режимах возможно создание базы данных?
2. Какие типы данных допустимы при создании таблицы?
3. Как выполнить создание таблицы средствами СУБД?
4. Как выполнить создание таблицы средствами языка SQL?
5. Как разделяются операторы SQL в случае нескольких операторов в запросе?
6. Каким образом выполнить простейшие операции вставки строк данных в таблицу средствами SQL?
7. Каким образом выполнить простейшие операции модификации строк таблицы средствами SQL?
8. Каким образом выполнить просмотр таблицы?
9. Как получить информацию о структуре таблицы в рамках СУБД MySQL?

Список литературы, рекомендуемый к использованию по данной теме:
Основная: 1-11
Дополнительная: 1-11
Интернет-ресурсы: 1-7
Программное обеспечение: 1-4

Лабораторная работа№9-10

Тема 9 - 10. Методы сериализации транзакция. Журнализация изменений БД

1. Цель работы: ознакомиться со средствами предоставления полномочий на использование баз данных и таблиц и основами работы с внешними базами данных.

Формируемые компетенции: ПК-1, ПК-3

2. Подготовка к занятию

1. Изучить (повторить) теоретический материал.
2. Ознакомиться с программой работы.
3. Подготовить отчет о работе.
4. Ответить на контрольные вопросы.

3. Распределение времени занятия:

Всего: 90 мин

Вступительная часть 2 мин

Проверка готовности студентов к занятию 5 мин

Основная часть – 70 мин

Проверка выполнения практического занятия 10 мин

Заключительная часть 3 мин

4. Правила работы в лаборатории

К работе в лаборатории допускаются лица, изучившие правила и меры безопасности, сдавшие зачет по ним и усвоившие порядок выполнения лабораторной работы.

4.1. Требования безопасности перед началом работ

- ЗАПРЕЩАЕТСЯ: переодеваться, пользоваться огнем, курить, принимать пищу в лаборатории.

- Убедиться в целостности электрических розеток и разъемов. В лаборатории необходимо быть в сменной обуви.

- Включение компьютера производить только после получения допуска по выполняемой работе и разрешения преподавателя или лаборанта.

4.2. Требования безопасности во время работы

- выполняя практическое занятие, студенты обязаны использовать только вычислительную технику, периферийное оборудование, соединительные кабели, измерительное оборудование и носители информации, непосредственно относящиеся к данному лабораторному занятию;

- подключение и отключение составляющих вычислительного комплекса производить только при полном снятии напряжения со всех составляющих вычислительного комплекса;

- при обнаружении неисправностей в оборудовании немедленно отключить источники питания и доложить об этом руководителю занятий или лаборанту.

4.3. Требования безопасности по окончании работы

- доложить руководителю занятий или лаборанту о завершении работ;
- привести в порядок и сдать рабочее место лаборанту, и доложить руководству

Предоставление доступа к базам данных

СУБД MySQL использует специальную базу данных для предоставления прав доступа к своим базам данных. Эти права могут базироваться на именах серверов и/или пользователей и предоставляться для одной или нескольких баз данных

Пользовательские учетные записи могут быть снабжены паролями. При обращении к базе данных, пароль шифруется. Поэтому он не может быть перехвачен и использован посторонним (это мнение автора СУБД...).

СУБД MySQL имеет три таблицы, а именно:

База данных: mysql Таблица: db

База данных: mysql Таблица: db					
Поле	**Тип**	**Null**	**Ключ**	**Умолчание**	**Extra**
Хост	char(60)		PRI		
Db	char(32)		PRI		
Пользователь	char(16)		PRI		
Select_priv	char(1)			N	
Insert_priv	char(1)			N	
Update_priv	char(1)			N	
Delete_priv	char(1)			N	
Create_priv	char(1)			N	
Drop_priv	char(1)			N	

База данных: mysql Таблица: host

Поле	**Тип**	**Null**	**Ключ**	**Умолчание**	**Extra**
Хост	char(60)		PRI		
Db	char(32)		PRI		
Select_priv	char(1)			N	
Insert_priv	char(1)			N	

Update_priv	char(1)			N	
Delete_priv	char(1)			N	
Create_priv	char(1)			N	
Drop_priv	char(1)			N	

База данных: mysql Таблица: user

База данных: mysql Таблица: user

Поле	**Тип**	**Null**	**Key**	**Умолчание**	**Extra**
Хост	char(60)		PRI		
Пользователь	char(16)		PRI		
Пароль	char(8)				
Select_priv	char(1)			N	
Insert_priv	char(1)			N	
Update_priv	char(1)			N	
Delete_priv	char(1)			N	
Create_priv	char(1)			N	
Drop_priv	char(1)			N	
Reload_priv	char(1)			N	
Shutdown_priv	char(1)			N	
Process_priv	char(1)			N	
File_priv	char(1)			N	

Текущей базой данных называется база данных, открытая с помощью операторов use Database или с помощью утилиты mysqladmin. Любая другая база данных называется внешней. Для ссылки на таблицу во внешней базе данных необходимо указать имя этой базы данных как часть имени таблицы, например, salesdb:contracts, где salesdb - имя внешней базы данных, contracts - имя таблицы. К имени базы данных можно добавить имя сервера, т.е. сетевой машины, где запущен еще один сервер баз данных mysql, и таким образом в случае распределенной базы

данных обращение к таблице contracts базы данных salesdb, размещенной на сервере central, будет выглядеть следующим образом: salesdb@central:contracts.

В программе MYSQL-FRONT также существует механизм, обеспечивающий наделение пользователей определенными правами (см. Рисунок 6).

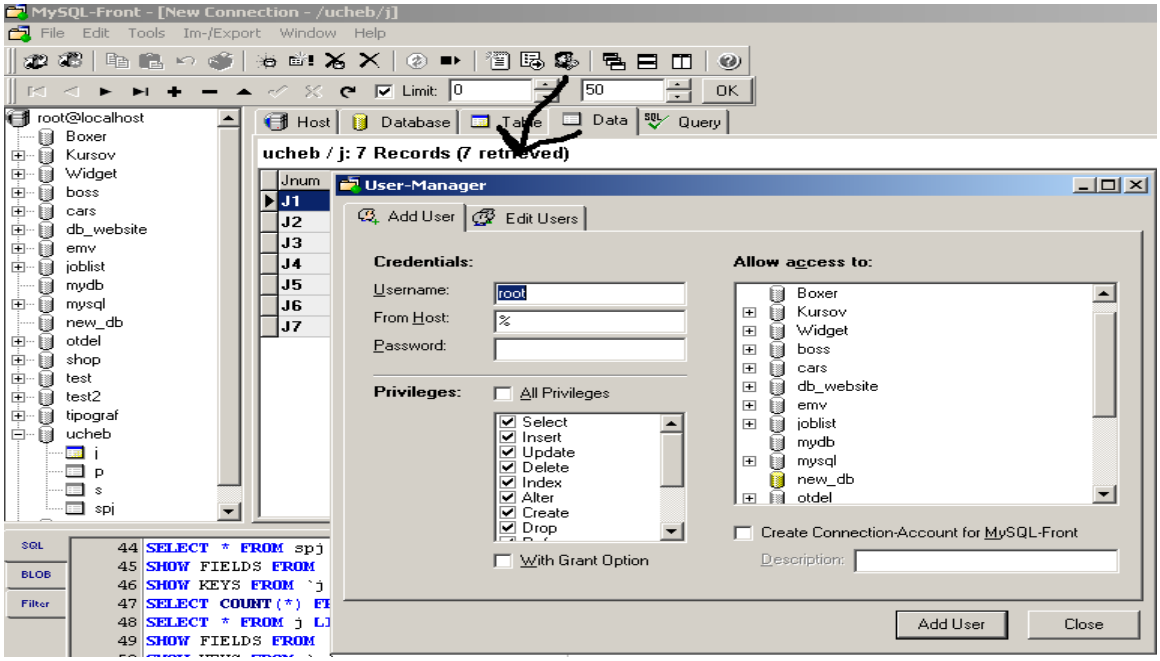


Рисунок 6 - Редактирование прав пользователя

При наличии сетевого соединения с сервером выполните приведенную последовательность выполнения лабораторной работы, при отсутствии соединения создать еще 1го пользователя в вашей БД, наделив его привилегиями лишь для просмотра таблиц, в этом случае все приведенные операции осуществлять от имени созданного пользователя.

Задание (общее):

1. Убедиться, что в таблице поставщиков S имеются строки с Вашими фамилиями (задание выполнялось в третьей лабораторной работе).
2. Откорректировать экранную форму, созданную в третьей лабораторной работе для работы с таблицей поставок SPJ, обеспечив возможность ввода и модификации данных. Занести произвольным образом несколько строк (5-10 строк) о поставках, связанных с Вашими фамилиями.
3. Выполнить два запроса к базе данных, согласно номера Вашего варианта. При выполнении запроса данные должны выбираться из таблиц Вашей базы данных.

4. Повторить задание п.3 с той разницей, что сведения о номенклатуре деталей и изделий (P и J) должна браться из собственной базы данных, а сведения о поставщиках и поставках (S и SPJ) должны браться из базы данных соседней бригады. Предварительно необходимо узнать имя этой базы данных. Убедитесь в невозможности выполнения задания.

5. Обеспечьте, чтобы владелец внешней используемой Вами базы данных предоставил Вам полномочия на просмотр используемых Вами таблиц в его базе данных, дав соответственно ему такие же полномочия для выполнения аналогичных действий.

6. Повторите задание п.4. Сравните результаты с результатами, полученными в п.3.

7. Сделайте попытку изменить информацию о поставщиках-владельцах базы данных (город, рейтинг и т.д.) в таблице S внешней базы данных. Убедитесь в невозможности выполнения задания.

8. Обеспечьте, чтобы владелец внешней используемой Вами базы данных предоставил Вам полномочия на модификацию данных из используемых Вами таблиц в его базе данных, дав соответственно ему такие же полномочия для выполнения аналогичных действий.

9. Повторите задание п.7. Проверьте успешность выполнения действий.

10. Дождавшись, когда владелец внешней базы данных закончит выполнение п.9, сделайте попытку удалить из таблицы S используемой Вами внешней базы данных поставщиков с именами, принадлежащими владельцам базы данных, и связанные с ними поставки из таблицы SPJ. Убедитесь в невозможности выполнения задания.

11. Обеспечьте, чтобы владелец используемой Вами внешней базы данных предоставил Вам полномочия на удаление из используемых Вами таблиц в его базе данных, дав соответственно ему такие же полномочия для выполнения аналогичных действий.

12. Повторите задание п.10. Проверьте успешность выполнения действий.

13. Отнимите предоставленные Вами права на пользование Вашей базой данных.

Контрольные вопросы

1. Кто является владельцем базы данных?
2. Какими правами обладают другие пользователи по отношению к Вашей базе данных?
3. Какими правами обладает администратор базы данных по отношению к Вашей базе данных?
4. Каким образом предоставляются права на пользование базой данных и отдельными ее таблицами?
5. Каким образом изымаются права на пользование базой данных и отдельными ее таблицами?

6. Что такое внешняя база данных?
7. Как идентифицируется таблица внешней базы данных?
8. Как идентифицируется таблица внешней распределенной базы данных?

Список литературы, рекомендуемый к использованию по данной теме:
Основная: 1-11
Дополнительная: 1-11
Интернет-ресурсы: 1-7
Программное обеспечение: 1-4

Лабораторная работа№11-12

Тема 11 - 12. Язык SQL. Функции и основные возможности. Средства манипулирования данными в SQL

1. Цель работы: приобретение практических навыков работы со средствами динамического SQL при написании программ на ESQL/C.

Формируемые компетенции: ПК-1, ПК-3

2. Подготовка к занятию

1. Изучить (повторить) теоретический материал.
2. Ознакомиться с программой работы.
3. Подготовить отчет о работе.
4. Ответить на контрольные вопросы.

3. Распределение времени занятия:

Всего: 90 мин

Вступительная часть 2 мин

Проверка готовности студентов к занятию 5 мин

Основная часть – 70 мин

Проверка выполнения практического занятия 10 мин

Заключительная часть 3 мин

4. Правила работы в лаборатории

К работе в лаборатории допускаются лица, изучившие правила и меры безопасности, сдавшие зачет по ним и усвоившие порядок выполнения лабораторной работы.

4.1. Требования безопасности перед началом работ

- ЗАПРЕЩАЕТСЯ: переодеваться, пользоваться огнем, курить, принимать пищу в лаборатории.

- Убедиться в целостности электрических розеток и разъемов. В лаборатории необходимо быть в сменной обуви.

- Включение компьютера производить только после получения допуска по выполняемой работе и разрешения преподавателя или лаборанта.

4.2. Требования безопасности во время работы

- выполняя практическое занятие, студенты обязаны использовать только вычислительную технику, периферийное оборудование, соединительные кабели, измерительное оборудование и носители информации, непосредственно относящиеся к данному лабораторному занятию;

- подключение и отключение составляющих вычислительного комплекса производить только при полном снятии напряжения со всех составляющих вычислительного комплекса;

- при обнаружении неисправностей в оборудовании немедленно отключить источники питания и доложить об этом руководителю занятий или лаборанту.

4.3. Требования безопасности по окончании работы

- доложить руководителю занятий или лаборанту о завершении работ;

- привести в порядок и сдать рабочее место лаборанту, и доложить руководству

Содержание работы и методические указания к ее выполнению

С помощью динамического SQL программа-клиент выполняет программное формирование оператора SQL для его последующего исполнения, делая это в три этапа:

— программа собирает текст оператора SQL в виде символьной строки, хранящейся в программной переменной; в общем случае это может быть не один, а несколько операторов SQL, разделенных точкой с запятой;

— программа выполняет оператор *Prepare*, который обращается к серверу баз данных для анализа текста оператора и подготовки его к выполнению;

— программа использует оператор Execute для выполнения подготовленного оператора.

Динамический оператор SQL по форме напоминает любой другой оператор SQL, записанный в программе, с тем ограничением, что он не может содержать имена главных переменных. Поэтому

— в динамический оператор *Select* нельзя включать спецификатор *Into*;

— в любом месте, где главная переменная могла бы появиться в выражении, ей соответствует знак вопроса.

Оператор *Prepare* создает структуру данных, имеющую имя и отображающую строку символов с текстом оператора SQL.

Подготовленный по оператору *Prepare* динамический оператор (группу операторов) можно многократно выполнять. С помощью оператора *Execute* выполняются операторы, отличные от операторов Select, а также операторы Select, которые возвращают в качестве результата одну строку. Если оператор *Select* возвращает более одной строки, динамический оператор *Select* выполняется не с помощью оператора *Execute*, а подключается к курсору и в дальнейшем используется обычным образом с помощью курсорных средств. В обоих случаях при выполнении динамического оператора с помощью спецификатора *Using* ему передаются главные переменные, участвующие в выражениях и принимающие возвращаемые значения и, по сути, играющие роль фактических параметров. Как ограничение, следует отметить, что знак вопроса нельзя использовать вместо идентификаторов SQL, таких как имя базы данных, таблицы или столбца: эти идентификаторы должны указываться в тексте оператора при его подготовке, возможно, как полученные из пользовательского ввода.

Последовательность выполнения лабораторной работы:

1. Изучить синтаксис и правила использования операторов *Prepare, Execute* (см. Приложение 2), а также особенности работы с курсором при выполнении динамического оператора SQL.

2. Разработать и отладить набор ESQL/C-программ, решающих задачи из соответствующего варианта заданий. Результатом работы программ является одна или несколько строк, которые подлежат выводу на экран с соответствующими пояснительными заголовками.

Требования к разрабатываемой программе

Разрабатываемые ESQL/C-программы должна удовлетворять следующим требованиям:

— обеспечивать необходимую обработку ошибок;

— использовать аппарат транзакций;

— все используемые в программах операторы SQL, включая операторы, реализующие аппарат транзакций, должны быть динамически подготовлены;

— необходимые параметры, определяющие условия задачи, вводятся с клавиатуры и передаются в строку с текстом динамического оператора SQL;

— все параметры, специфицирующие выполняемые действия, должны передаваться через главные переменные; такими параметрами в условиях задач являются название города, название детали, номер поставщика и т.д.;

— должен быть предусмотрен вывод сообщений обо всех шагах выполнения программы, в том числе и о возможных ошибках;

— программа должна быть достаточно документирована.

Варианты заданий

Вариант 1.

1. Выдать полную информацию о поставщике, имеющим максимальный рейтинг (с использованием оператора *Execute*).

2. Получить номера изделий, для которых детали полностью поставляет поставщик с указанным номером (параметр - номер поставщика (S1)).

3. Выдать номера и фамилии поставщиков, поставляющих детали для какого-либо изделия с деталью, номер которой указывается, в количестве, большем, чем средний объем поставок данной детали для этого изделия (параметр - номер детали (P1)).

Вариант 2.

1. Выдать полную информацию об изделии, изготавливаемом в городе, в котором проживает поставщик с максимальным рейтингом (с использованием оператора *Execute*).

2. Получить общее количество деталей с указанным номером, поставляемых некоторым поставщиком (параметры - номер детали (P1), номер поставщика (S1)).

3. Выдать номера изделий, использующих только детали, поставляемые некоторым поставщиком (параметр - номер поставщика (S1)).

Вариант 3.

1. Выдать полную информацию о детали, имеющей максимальный вес (с использованием оператора *Execute*).

2. Получить общее число изделий, для которых поставляет детали поставщик с указанным номером (параметр - номер поставщика (S1)).

3. Выдать номера изделий, детали для которых поставляет каждый поставщик, поставляющий какую-либо деталь указанного цвета (параметр - цвет детали (красный)).

Вариант 4.

1. Выдать общий объем поставок деталей красного цвета (с использованием оператора *Execute*).

2. Получить полный список деталей для всех изделий, изготавливаемых в некотором городе (параметр - название города (Лондон)).

3. Выдать номера деталей, поставляемых каким-либо поставщиком из указанного города (параметр - название города (Лондон)).

Вариант 5.

1. Выдать полную информацию об изделии, имеющем максимальный объем поставок деталей (с использованием оператора *Execute*).

2. Получить список всех поставок, в которых количество деталей находится в некотором диапазоне (параметры - границы диапазона (от 300 до 750)) .

3. Выдать номера и названия деталей, поставляемых для какого-либо изделия из указанного города (параметр - название города (Лондон)).

Вариант 6.

1. Выдать общий объем поставок деталей для изделия J2 (с использованием оператора *Execute*).

2. Получить цвета деталей, поставляемых некоторым поставщиком (параметр - номер поставщика (S1)).

3. Выдать номера и фамилии поставщиков, поставляющих некоторую деталь для какого-либо изделия в количестве, большем среднего объема поставок данной детали для этого изделия (параметр - номер детали (P1)).

Вариант 7.

1. Выдать общий объем поставок деталей для изделия с максимальным объемом поставок (с использованием оператора *Execute*).

2. Получить названия изделий, для которых поставляются детали некоторым поставщиком (параметр - номер поставщика (S1)).

3. Выдать номера изделий, для которых средний объем поставки некоторой детали больше максимального объема поставки любой детали для указанного изделия (параметры - номер детали (P1), номер изделия (J1)).

Контрольные вопросы

1. Каково назначение и синтаксис оператора *Prepare*?
2. Каково назначение и синтаксис оператора *Execute?*
3. Каковы особенности использования динамических операторов SQL?
4. Что такое динамические главные переменные? Для чего они используются?
5. С какими операторами связано использование динамических главных переменных?
6. Каково назначение оператора *Execute Immediate*?

Список литературы, рекомендуемый к использованию по данной теме:
Основная: 1-11
Дополнительная: 1-11
Интернет-ресурсы: 1-7
Программное обеспечение: 1-4

Лабораторная работа№13-14

Тема 13 - 14. Безопасность SQL при прикладном программировании. Компиляторы SQL и проблемы безопасности оптимизации

1. Цель работы: ознакомиться с базовыми функциями выборки данных в ODBC и разработать с их использованием программу получения данных из результирующего множества.

2. Подготовка к занятию

1. Изучить (повторить) теоретический материал.
2. Ознакомиться с программой работы.
3. Подготовить отчет о работе.
4. Ответить на контрольные вопросы.

3. Распределение времени занятия:

Всего: 90 мин

Вступительная часть 2 мин

Проверка готовности студентов к занятию 5 мин

Основная часть – 70 мин

Проверка выполнения практического занятия 10 мин

Заключительная часть 3 мин

4. Правила работы в лаборатории

К работе в лаборатории допускаются лица, изучившие правила и меры безопасности, сдавшие зачет по ним и усвоившие порядок выполнения лабораторной работы.

4.1. Требования безопасности перед началом работ

- ЗАПРЕЩАЕТСЯ: переодеваться, пользоваться огнем, курить, принимать пищу в лаборатории.

- Убедиться в целостности электрических розеток и разъемов. В лаборатории необходимо быть в сменной обуви.

- Включение компьютера производить только после получения допуска по выполняемой работе и разрешения преподавателя или лаборанта.

4.2. Требования безопасности во время работы

- выполняя практическое занятие, студенты обязаны использовать только вычислительную технику, периферийное оборудование, соединительные кабели, измерительное

оборудование и носители информации, непосредственно относящиеся к данному лабораторному занятию;

- подключение и отключение составляющих вычислительного комплекса производить только при полном снятии напряжения со всех составляющих вычислительного комплекса;

- при обнаружении неисправностей в оборудовании немедленно отключить источники питания и доложить об этом руководителю занятий или лаборанту.

4.3. Требования безопасности по окончании работы

- доложить руководителю занятий или лаборанту о завершении работ;

- привести в порядок и сдать рабочее место лаборанту, и доложить руководству

Содержание работы и методические указания к ее выполнению

Для выполнения работы необходимо

- изучить базовые функции выборки данных **SQLBindCol, SQLFetch(), SQLGetData()**;
- ознакомиться с алгоритмами извлечения данных из результирующего множества с использованием средств ODBC;
- настроить среду выполнения, разработать и отладить ODBC-программу выборки данных.

В ODBC существует две функции базового уровня для выборки результатов - **SQLBindCol()** и **SQLFetch()**. Функция **SQLBindCol()** определяет область хранения данных результирующего множества, функция **SQLFetch()** осуществляет выборку данных в области хранения.

Каждый столбец, который требуется выбрать, связывается с помощью отдельного вызова функции **SQLBindCol()**. Функция **SQLBindCol()** назначает область хранения в памяти и тип данных для столбца результирующего множества. Она определяет:

- буфер хранения для получения содержимого столбца данных в результирующем множестве.
- длину указанного буфера хранения.
- область памяти для хранения длины столбца выборки.
- преобразование типа данных.

Алгоритм программы, использующей **SQLFetch()** и **SQLBindCol()** для возвращения данных из результирующего множества предполагает выполнения следующих шагов:

1. Вызывается функция **SQLBindCol()** один раз для каждого столбца, который должен быть возвращен из результирующего множества.

2. Вызывается функция **SQLFetch()** для перемещения курсора на следующую строку и возврата данных из связанных столбцов.

3. Повторяется шаг 2 до тех пор, пока функция **SQLFetch()**не возвратит SQL_NO_DATA_FOUND. Это указывает на то, что был достигнут конец результирующего множества. Если результирующее множество является пустым, то SQL_NO_DATA_FOUND будет возвращен при первом вызове **SQLFetch()**.

RETCODE **SQLBindCol** (*hstmt, icol, fcType, rgbValue, cbValueMax, pcbValue*)
HSTMT *stmt* - идентификатор оператора;
UWORD *icol* - идентификатор окружения;
SWORD *fCtype* - С-тип данных результирующего множества;
PTR *rgbValue* - указатель области хранения данных. Если rgbValue является нулевым указателем, то драйвер "отвязывает" столбец; для отвязывания всех столбцов прикладная программа вызывает **SQLFreeStmt()** с опцией SQL_UNBIND;
SDWORD *cbValueMax* - максимальная длина буфера rgbValue; для символьных данных, rgbValue должен также включать в себя место для байта нулевого окончания;
SDWORD *pcbValue* - число байт, которое может возвращаться в rgbValue при вызове **SQLFetch()**.
Как только все требуемые столбцы были связаны, вызывается **SQLFetch** для выборки данных в определенной области хранения.
RETCODE **SQLFetch** (*hstmt*)
HSTMT *stmt* - идентификатор оператора.
Функция **SQLGetData()** позволяет выполнить выборку данных из столбцов, для которых область хранения не была заранее подготовлена с помощью функции **SQLBindCol()**. Функция **SQLGetData()** вызывается после вызова функции**SQLFetch()** для выборки данных из текущей строки.
Алгоритм программы, использующей **SQLFetch()** и **SQLGetData()** для извлечения данных из каждой строки результирующего множества предполагает выполнения следующих шагов:

1. Вызывается функция **SQLFetch()** для продвижения на следующую строку.

2. Вызывается функция **SQLGetData()** для каждого столбца, который должен быть возвращен из результирующего множества.

3. Повторяется шаги 1 и 2 до тех пор, пока функция **SQLFetch()** не возвратит SQL_NO_DATA_FOUND. Это указывает на то, что был достигнут конец

результирующего множества. Если результирующее множество является пустым, то SQL_NO_DATA_FOUND будет возвращен при первом вызове **SQLFetch()**.

RETCODE **SQLGetData** (*hstmt, icol, fcType, rgbValue,*
cbValueMax, pcbValue);

HSTMT *stmt;* - идентификатор оператора;

UWORD *icol;* - номер столбца результирующего множества, упорядоченный слева направо, начиная с 1;

SWORD *fCtype;* - С-тип данных результирующего множества;

PTR *rgbValue;* - указатель области хранения данных. Если rgbValue является нулевым указателем, то драйвер "отвязывает" столбец. Для отвязывания всех столбцов прикладная программа вызывает функцию **SQLFreeStmt()** с опцией SQL_UNBIND.

SDWORD *cbValueMax;* - максимальная длина буфера rgbValue; для символьных данных, rgbValue должен также включать в себя место для байта нулевого окончания.

SDWORD FAR* *pcbValue;* - число байт, которое может возвращаться в rgbValue при вызове **SQLGetData()**.

Замечание. Прикладная программа может использовать **SQLBindCol()** для некоторых столбцов, а **SQLGetData()** - для других столбцов в пределах той же самой строки.

Последовательность выполнения лабораторной работы

1. Убедиться в наличии и заполненности базы данных поставщиков, деталей, изделий, поставок.
2. Разработать ODBC-программу для решения задачи из соответствующего варианта с помощью функций **SQLBindCol(), SQLFetch()**.
3. Разработать ODBC-программу для решения задачи из соответствующего варианта с помощью функций **SQLFetch(), SQLGetData()**.
4. После выполнения лабораторной работы привести базу данных в исходное состояние.

Требования к разрабатываемой программе

Разрабатываемая программа должна удовлетворять следующим требованиям:

— все используемые функции ODBC должны анализироваться на корректность кода возврата;

— в программе должен быть предусмотрен вывод сообщений обо всех шагах ее выполнения, в том числе и о возможных ошибках;

— при выполнении запросов должно быть предусмотрено использование параметров; параметры варианта задания должны быть введены в ходе выполнения программы и переданы в SQL-запрос;

— при выполнении программы должна контролироваться целостность базы данных;

— программа должна быть достаточно документирована.

Варианты заданий

1. Вывести информацию о поставщиках, поставивших детали для изделий из указанного города.
2. Вывести информацию о деталях, поставки которых были осуществлены для указанного изделия.
3. Вывести информацию о поставщиках, которые осуществляли поставки деталей из заданного города в указанный период.
4. Вывести информацию о поставщиках, поставивших указанную деталь в заданный период.
5. Вывести информацию обо всех деталях, поставляемых для указанного изделия более чем одним поставщиком.
6. Вывести информацию о деталях, поставки которых были осуществлены для указанного изделия всеми поставщиками.
7. Вывести информацию об изделиях, для которых была поставлена указанная деталь.
8. Вывести информацию о поставщиках, которые осуществляли поставки деталей для указанного изделия.

Контрольные вопросы

1. Какие существуют базовые функции выборки данных? Охарактеризуйте каждую из них.
2. Как "отвязать" отдельный столбец результирующего множества?
3. Каким образом можно "отвязать" все столбцы результирующего множества?
4. Что возвращает функция **SQLFetch,** если в результирующем множестве больше не осталось данных?
5. Как определить область хранения данных?
6. Каков алгоритм выборки данных с помощью курсора?

7. Как работает функция **SQLFetch**?

8. Можно ли использовать **SQLFetch** для продвижения курсора в обратном направлении?

Список литературы, рекомендуемый к использованию по данной теме:
Основная: 1-11
Дополнительная: 1-11
Интернет-ресурсы: 1-7
Программное обеспечение: 1-4

Лабораторная работа№15-16

Тема 15 - 16. Безопасность архитектуры «клиент-сервер». Безопасность распределенных баз данных

1. Цель работы: ознакомиться с базовыми конструкциями языка PHP с целью написания с их использованием простейших PHP-программ доступа к базам данных.

Формируемые компетенции: ПК-1, ПК-3

2. Подготовка к занятию

1. Изучить (повторить) теоретический материал.
2. Ознакомиться с программой работы.
3. Подготовить отчет о работе.
4. Ответить на контрольные вопросы.

3. Распределение времени занятия:

Всего: 90 мин

Вступительная часть 2 мин

Проверка готовности студентов к занятию 5 мин

Основная часть – 70 мин

Проверка выполнения практического занятия 10 мин

Заключительная часть 3 мин

4. Правила работы в лаборатории

К работе в лаборатории допускаются лица, изучившие правила и меры безопасности, сдавшие зачет по ним и усвоившие порядок выполнения лабораторной работы.

4.1. Требования безопасности перед началом работ

- ЗАПРЕЩАЕТСЯ: переодеваться, пользоваться огнем, курить, принимать пищу в лаборатории.

- Убедиться в целостности электрических розеток и разъемов. В лаборатории необходимо быть в сменной обуви.

- Включение компьютера производить только после получения допуска по выполняемой работе и разрешения преподавателя или лаборанта.

4.2. Требования безопасности во время работы

- выполняя практическое занятие, студенты обязаны использовать только вычислительную технику, периферийное оборудование, соединительные кабели, измерительное

оборудование и носители информации, непосредственно относящиеся к данному лабораторному занятию;

- подключение и отключение составляющих вычислительного комплекса производить только при полном снятии напряжения со всех составляющих вычислительного комплекса;

- при обнаружении неисправностей в оборудовании немедленно отключить источники питания и доложить об этом руководителю занятий или лаборанту.

4.3. Требования безопасности по окончании работы

- доложить руководителю занятий или лаборанту о завершении работ;

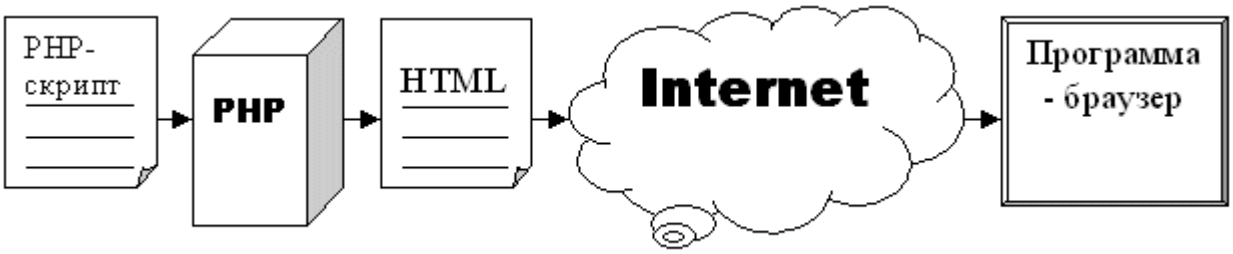
- привести в порядок и сдать рабочее место лаборанту, и доложить руководству

Содержание работы и методические указания к ее выполнению

Для выполнения работы необходимо

- ознакомиться с синтаксисом языка PHP;
- изучить особенности передачи значений переменных HTML-формы в переменные PHP;
- ознакомиться с набором функций для общения с СУБД Informix;
- с использованием средств языка PHP разработать и отладить программу доступа к базе данных.

Язык PHP - это действующий на стороне сервера встраиваемый в HTML язык, имеющий синтаксис, близкий к языку Си. Язык PHP дает возможность вставлять в файлы HTML инструкции языка PHP для создания динамического содержания. Эти инструкции обрабатывает препроцессор-интерпретатор PHP и заменяет их тем содержимым, которое производит этот код. PHP-программа может целиком состоять из конструкций языка PHP, а может быть смесью конструкций языков PHP и HTML. Стандартное расширение файла с PHP-программой - *php.*



Одним из распространенных применений PHP является работа с базами данных. Для целого ряда баз данных PHP имеет собственную поддержку, а другие доступны через ODBC-функции PHP. При вызове PHP-программы URL-адрес должен содержать номер порта, через который работает PHP:

html://fpm.ami.nstu.ru:81/~pmxxyy/t1.php

К особенностям языка PHP относятся:

— возможность встраивать конструкции языка PHP в HTML-документ;

— возможность включать в PHP-программу файлы;

— наличие достаточного набора встроенных функций;

— возможность определять собственные переменные, строки, массивы, объекты;

— наличие необходимого набора управляющих структур;

— возможность вводить собственные функции.

Одна из наиболее удобных особенностей PHP - это способность автоматически передавать значения переменных из HTML-форм в переменные PHP. PHP автоматически генерирует набор переменных, имена которых совпадают с именами объектов в HTML-форме и содержащих значения данных объектов. В результате отпадает необходимость в выполнении рутинного преобразования, связанного с разбором последовательности

имя=значение&имя1=значение1&...&имяN=значениеN

Для связи с любой из СУБД PHP в своем наборе имеет ряд функций, которые очень похожи между собой и имеют одинаковую логику работы и аналогичные параметры.

В приведенной ниже таблице представлен минимальный набор функций, необходимых для написания PHP-программ, общающихся с СУБД Informix.

1.	**int ifx_connect (string database, string user, string password)** - создать соединение с сервером Informix	
.	Входные параметры:	*Database* - имя базы данных; *user* - имя пользователя *password* - пароль.
.	Возвращаемое значение:	идентификатор соединения, если соединение прошло успешно, и равен 0 в противном случае.
2.	**int ifx_query (string query, int link_id, int cursor_type)** - выполнить запрос к базе	
.	Входные параметры:	*query* - строка SQL-запроса; *link_id* - идентификатор соединения; *cursor type* - тип курсора

.	Возвращаемое значение:	значение 1 или 0 в зависимости от успеха выполнения операции.
3.	**array ifx_fetch_row (int result_id, mixed [position])**	- получить строку запроса как массив
.	Входные параметры:	*result_id* - идентификатор результата, возвращенный функцией ifx_query() (только для запросов типа select); *[position]* - параметр из списка: "NEXT", "PREVIOUS", "CURRENT", "FIRST", "LAST" или номер.
.	Возвращаемое значение:	строка таблицы базы данных, возвращаемая как массив.
4.	**string current (array row)**	- получить очередное поле из строки таблицы базы данных.
.	Входные параметры:	*array row*- строка таблицы базы данных, возвращенная функцией **ifx_fetch_row().**
.	Возвращаемое значение:	очередное поле строки таблицы.
5.	**string next (array row)**	- получить следующее поле из строки таблицы базы данных.
.	Входные параметры:	*array row* - строка таблицы базы данных, возвращенная функцией **ifx_fetch_row().**
.	Возвращаемое значение:	следующее поле строки таблицы.
6.	**int reset(array$row)**	- перейти в начало строки.
.	Входные параметры:	*array row* - строка таблицы базы данных, возвращенная функцией **ifx_fetch_row().**
.	Возвращаемое значение:	нулевая позиция строки результата.
7.	**string key(array$row)**	- перейти в начало строки.
.	Входные параметры:	*array row* - строка таблицы базы данных, возвращенная функцией **ifx_fetch_row().**
.	Возвращаемое значение:	имя очередного поля строки результата.
8.	**int ifx_affected_rows (int result_id)**	- получить число столбцов, обработанных запросом
.	Входные параметры:	*result_id* - результат, возвращенный функцией **ifx_query().**
.	Возвращаемое значение:	Возвращается число столбцов, обработанных запросом, ассоциированных с *result_id.* Для вставок, обновлений и удалений - это реальное количество обработанных столбцов. Для выборок - ожидаемое количество.

9.	**int ifx_free_result (int result_id)**	- освободить ресурсы запроса
.	Входные параметры:	*result_id* - результат, возвращенный функцией **ifx_query().**
.	Возвращаемое значение:	Освобождает ресурсы, занятые запросом с идентификатором результата *result_id.* Возвращает 0 в случае ошибки.
10.	**int ifx_close (int [link_identifier])**	- закрыть соединение с Informix
.	Входные параметры:	*link_id* - идентификатор соединения;
.	Возвращаемое значение:	закрывает соединение с Informix. Если идентификатор ссылки не указан, предполагается последнее установленное соединение.

Общая схема написания PHP-программы, выполняющей взаимодействие с базой данных, мало отличается от структуры CGI-скрипта, написанного любыми другими средствами, разница состоит лишь в используемых средствах:

— подключиться к серверу баз данных и зарегистрироваться;

— выбрать базу данных, которая будет использоваться;

— отправить запрос SQL на сервер и получить данные;

— отключиться от сервера баз данных.

При этом остаются актуальными все замечания, сделанные в предыдущей лабораторной работе относительно установки переменных окружения и обеспечения мер безопасности при работе с базой данных.

Последовательность выполнения лабораторной работы

1. Убедиться в наличии и заполненности базы данных поставщиков, деталей, изделий, поставок.
2. Разработать и отладить HTML-формы для ввода данных пользователя согласно варианту задания (можно модифицировать HTML-формы из предыдущей лабораторной работы).
3. Разработать и отладить PHP-программы для обработки данных HTML-форм и доступа к базе данных.
4. После выполнения лабораторной работы привести базу данных в исходное состояние.

Требования к разрабатываемой программе

Разрабатываемые программы должна удовлетворять следующим требованиям:

- разрабатываемое программное приложение должно содержать HTML-документ с формой для ввода данных и PHP-программу, вызываемую по окончании работы с HTML-формой;
- ввод параметров задания в HTML-форме может быть осуществлен либо путем ввода значений в текстовом виде, либо посредством выбора значений из предлагаемого списка;
- программа должна быть написана в предположении, что любой пользователь без ограничений может иметь доступ к данным;
- в программе должен быть предусмотрен вывод сообщений обо всех шагах ее выполнения, в том числе и о возможных ошибках;
- программа должна быть достаточно документирована.

Варианты заданий

Вариант 1

1. Вывести информацию о поставщиках, поставивших детали для изделий из указанного города.
2. Увеличить рейтинг поставщика, выполнившего наибольшую поставку некоторой детали, на указанную величину.

Вариант 2

1. Вывести информацию о деталях, поставки которых были осуществлены для указанного изделия.
2. Изменить цвет самой тяжелой детали на указанный.

Вариант 3

1. Вывести информацию о поставщиках, которые осуществляли поставки деталей из заданного города в указанный период.
2. Вставить поставщика с заданными параметрами.

Вариант 4

1. Вывести информацию о поставщиках, поставивших указанную деталь в заданный период.
2. Удалить поставщика, выполнившего меньше всего поставок.

Вариант 5

1. Вывести информацию обо всех деталях, поставляемых для указанного изделия более чем одним поставщиком.
2. В таблице поставок изменить номер поставщика при заданном номере детали и изделия.

Вариант 6

1. Вывести информацию о деталях, поставки которых были осуществлены для указанного изделия всеми поставщиками.
2. Увеличить рейтинг поставщика, выполнившего больший суммарный объем поставок, на указанную величину.

Вариант 7

1. Вывести информацию об изделиях, для которых была поставлена указанная деталь.
2. Изменить название и город детали с максимальным весом на указанные значения.

Вариант 8

1. Вывести информацию о поставщиках, которые осуществляли поставки деталей для указанного изделия.
2. Увеличить рейтинг поставщика, выполнившего большее число поставок, на указанную величину.

Контрольные вопросы

1. Каким образом вставить конструкции PHP в HTML-документ?
2. Каким образом обеспечить, чтобы встречающиеся в строке переменные были заменены их значениями?
3. Каковы правила определения функций в языке PHP?
4. Каковы особенности передачи значений переменных из HTML-формы в переменные PHP?
5. Каким образом осуществляется взаимодействие с базами данных в языке PHP?

Список литературы, рекомендуемый к использованию по данной теме:
Основная: 1-11
Дополнительная: 1-11
Интернет-ресурсы: 1-7
Программное обеспечение: 1-4

Лабораторная работа№17-18

Тема 17 - 18. Безопасность объектно-ориентированных баз данных. Язык для взаимодействия с БД

1. Подготовка к занятию

1. Изучить (повторить) теоретический материал.
2. Ознакомиться с программой работы.
3. Подготовить отчет о работе.
4. Ответить на контрольные вопросы.

Формируемые компетенции: ПК-1, ПК-3

3. Распределение времени занятия:

Всего: 90 мин

Вступительная часть 2 мин

Проверка готовности студентов к занятию 5 мин

Основная часть – 70 мин

Проверка выполнения практического занятия 10 мин

Заключительная часть 3 мин

4. Правила работы в лаборатории

К работе в лаборатории допускаются лица, изучившие правила и меры безопасности, сдавшие зачет по ним и усвоившие порядок выполнения лабораторной работы.

4.1. Требования безопасности перед началом работ

- ЗАПРЕЩАЕТСЯ: переодеваться, пользоваться огнем, курить, принимать пищу в лаборатории.

- Убедиться в целостности электрических розеток и разъемов. В лаборатории необходимо быть в сменной обуви.

- Включение компьютера производить только после получения допуска по выполняемой работе и разрешения преподавателя или лаборанта.

4.2. Требования безопасности во время работы

- выполняя практическое занятие, студенты обязаны использовать только вычислительную технику, периферийное оборудование, соединительные кабели, измерительное оборудование и носители информации, непосредственно относящиеся к данному лабораторному занятию;

- подключение и отключение составляющих вычислительного комплекса производить только при полном снятии напряжения со всех составляющих вычислительного комплекса;

- при обнаружении неисправностей в оборудовании немедленно отключить источники питания и доложить об этом руководителю занятий или лаборанту.

4.3. Требования безопасности по окончании работы

- доложить руководителю занятий или лаборанту о завершении работ;

- привести в порядок и сдать рабочее место лаборанту, и доложить руководству

Работа с phpmyadmin

1. Установите веб-сервер Denwer.

2. Запуск phpmyadmin.

Откройте браузер и введите адрес http://localhost/phpmyadmin/, после чего в браузер загрузится следующая страница (Рисунок 1):

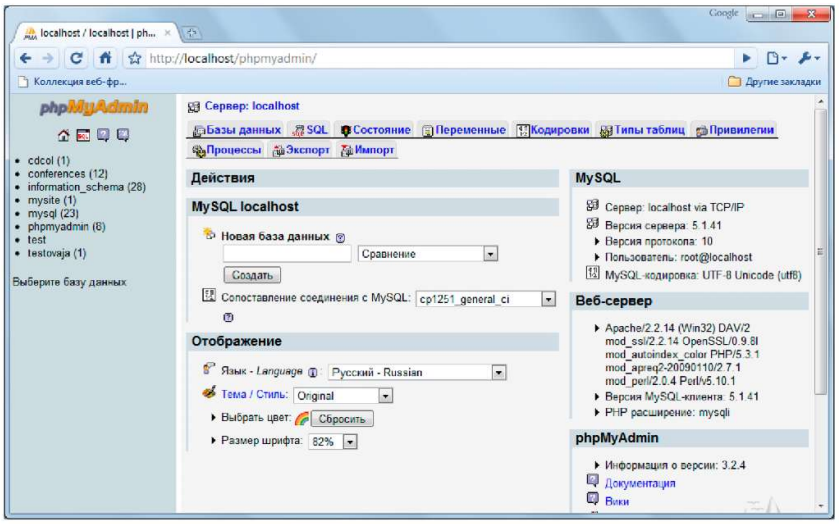


Рисунок 1. Интерфейс главной страницы phpmyadmin

3. Если используете систему в первый раз, и не изменяли данные пользователя root, то будет выведено соответствующее предупреждение.

4. Изменение пароля для root

Чтобы добавить нового пользователя для СУБД MySQL или изменить данные существующих пользователей, необходимо активировать вкладку **«Привилегии»** (Рисунок 2):

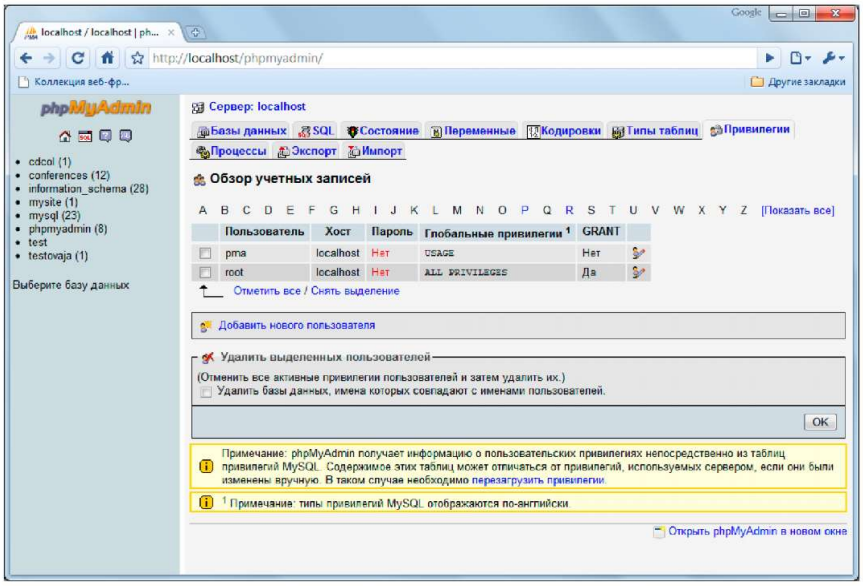


Рисунок 2. Интерфейс вкладки «Привилегии» системы phpmyadmin

5. На данной странице отображается список пользователей MySQL, в том числе и главной пользователь (суперпользователь) - root (по аналогии с операционными системами семейства UNIX). По умолчанию, данный пользователь не имеет пароля после установки системы. Вам необходимо задать для него пароль.

6. Изменение пароля для пользователя root.

7. В списке учетных записей СУБД MySQL отметьте необходимого пользователя (в данном случае root).

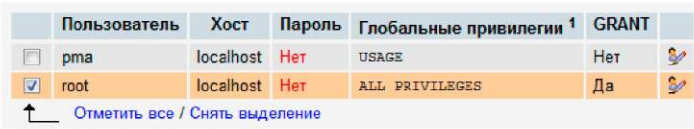


Рисунок 3. Выбор пользователя для редактирования

Дальше нажмите в последней колонке таблицы с учетными записями пользователей на пиктограмму, которая вызывает интерфейс для редактирования данных выбранного пользователя (Рисунок 4).

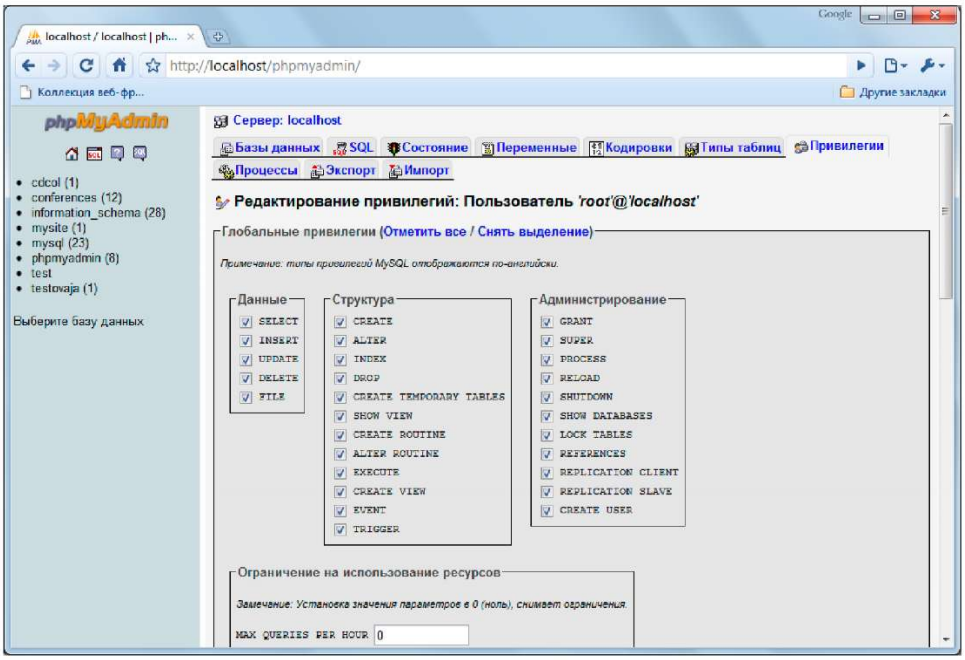


Рисунок 4. Интерфейс страницы для редактирования данных учетной записи

В блоке **«Изменить пароль»** введите новый пароль и подтвердите его, введя в поле «Подтверждение» тот же самый текст. Если вы не хотите самостоятельно определять пароль для пользователей, то можете воспользоваться пунктом «Создать пароль» и нажать кнопку «Генерировать», после чего система предложит свой вариант пароля, сгенерированный случайным образом.

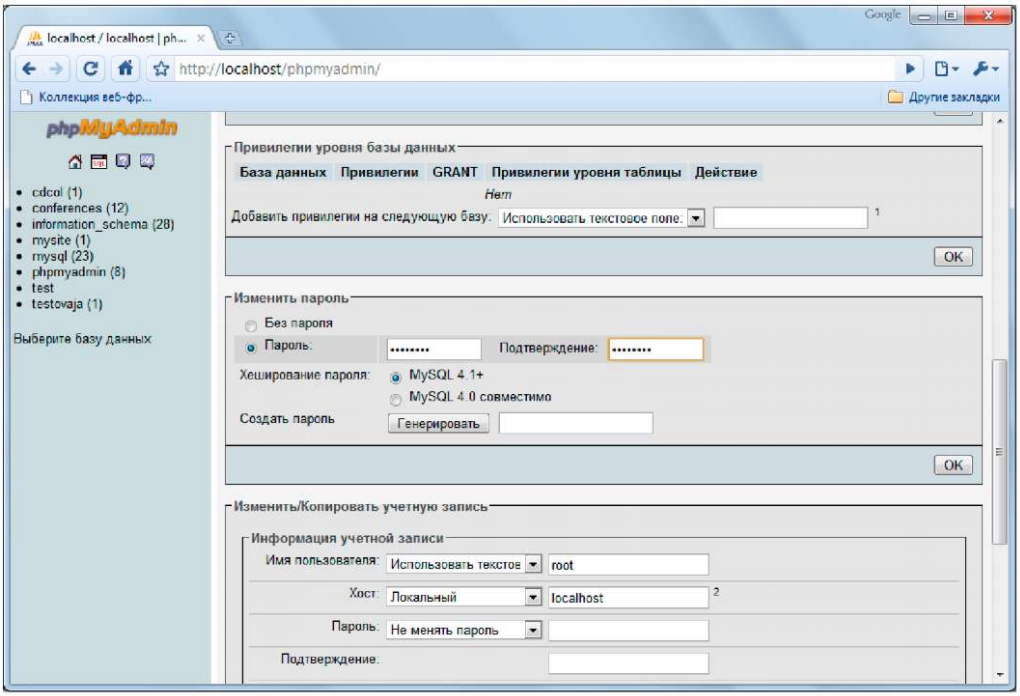


Рисунок 5. Изменение пароля пользователя

После ввода необходимых данных нажмите на кнопку **«ОК».** На странице появится надпись «Пароль для 'root'@'localhost' был успешно изменен».

8. Вернитесь в обзор учетных записей пользователей СУБД MySQL. Обновите страницу. Так как вы изменили пароль пользователя root, а именно от него Вы работаете в данный момент в системе phpmyadmin, то будет выведено сообщение следующего вида и содержания:

phpMyAdmin не смог установить соединение с сервером MySQL. Проверьте хост, имя пользователя и пароль установленные в конфигурационном файле config.inc.php и удостоверьтесь, что они соответствуют данным полученным от администратора сервера MySQL.

Это означает, что доступ для пользователя с именем root и с пустым паролем запрещен. Для устранения данной проблемы необходимо в конфигурационном файле phpmyadmin также изменить пароль для root, так как по умолчанию там стоит пустая строка.

Откройте файл config.inc.php из директории с установленным phpmyadmin. Найдите строки

$cfg['Servers'][$i]['user']= 'root';

$cfg['Servers'][$i]['password'] = '';

Измените значение переменной $cfg['Servers'][$i]['password'] на то значение, которое вы указали как новый пароль для пользователя root: $cfg['Servers'][$i]['password'] = 'password';

9. Обновите страницу Phpmyadmin **(F5).**

10. Активируйте вкладку **«База данных».** На данной странице будет отображен список созданных баз данных в СУБД MySQL.

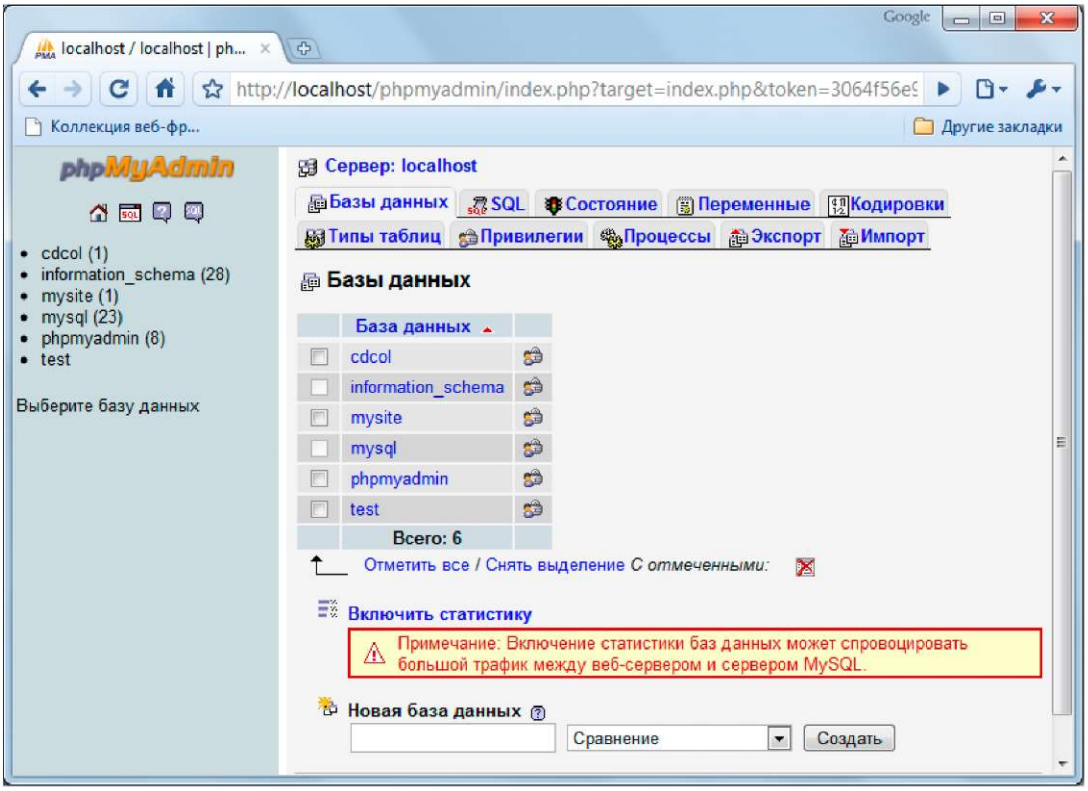


Рисунок 7. Интерфейс страницы «Базы данных»

С помощью данной страницы вы можете редактировать параметры существующих БД, удалять БД и создавать новые.

11. В блоке «Новая база данных» на текущей странице введите в текстовое поле «webProject», а в выпадающем меню с доступными кодировками для БД выберите CP1251-bin (аналогично Windows-1251), нажмите на кнопку «Создать» (Рисунок 8).

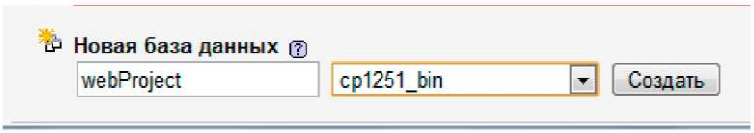


Рисунок 8. Ввод параметров для новой БД

12. Страница обновится, и появится сообщение следующего вида (Рисунок 9):

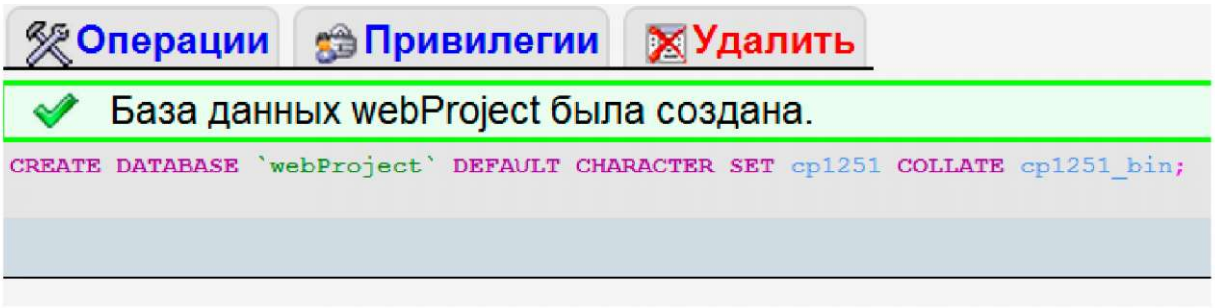


Рисунок 9. Сообщение об успешном создании БД

13. Обратите внимание на то, что практически при любых манипуляциях с БД и пользователями БД система phpmyadmin выводит на страницу соответствующий SQL-запрос. В данном случае это SQL- запрос для создания новой БД.

Для удобства разработчика система предлагает также генерацию PHP-кода для соответствующего запроса. Т.е. при разработке конкретно системы, которая обрабатывает данные из БД, вы можете сначала протестировать необходимые запросы в phpmyadmin, а затем скопировать PHP-код с SQL-запросом в свою систему.

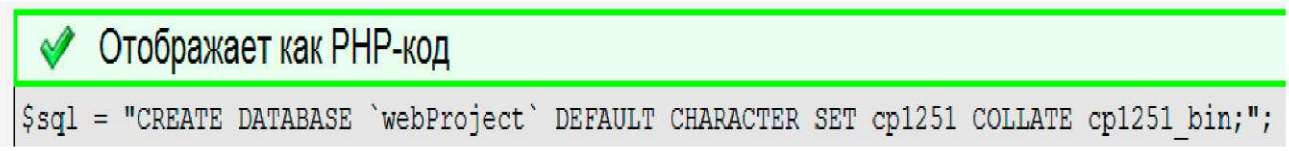


Рисунок 10. PHP-код с SQL-запросом для создания БД

14. После создания новой БД, система автоматически активирует работу с ней. Перейдите во вкладку **«Структура».** Здесь находится список таблиц выбранной БД (вновь созданная БД является пустой).

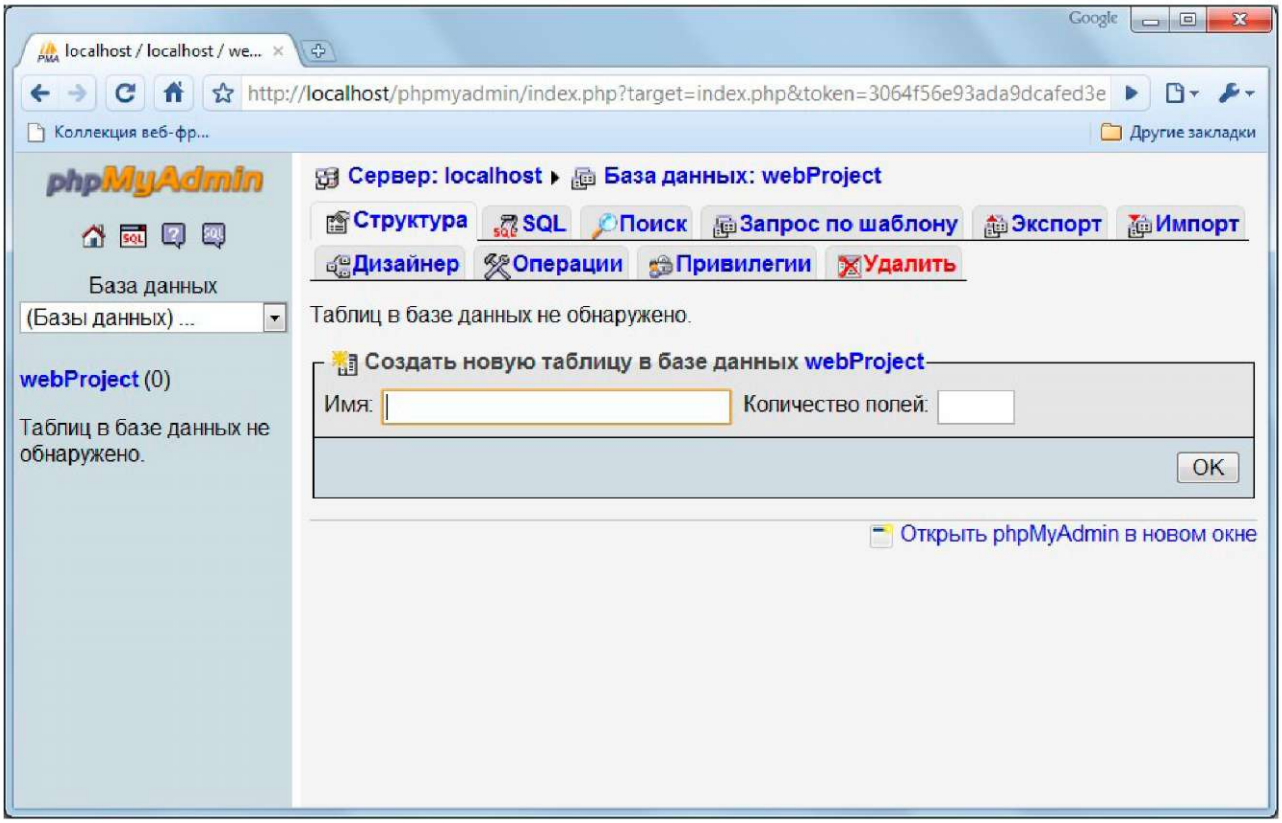


Рисунок 13. Ввод параметров для новой таблицы БД 17. Заполняем форму (водите только те параметры, которые указаны далее):

Имя поля - id (данное поле будет идентификатором записи и должно содержать уникальные значения);

тип - INT;

A_I (Auto increment) - ON (что означает, что при добавлении новой записи в таблицу данное поле будет автоматически увеличено на единицу);

Индекс - PRIMARY (что значит, что это поле - первичный ключ)

Рисунок 11. Интерфейс страницы, содержащей структуру БД (список таблиц

БД)

15. Создайте новую таблицу, введя строку «sections» в текстовое поле «Имя» и «5» в поле «Количество полей». Нажмите «ОК».

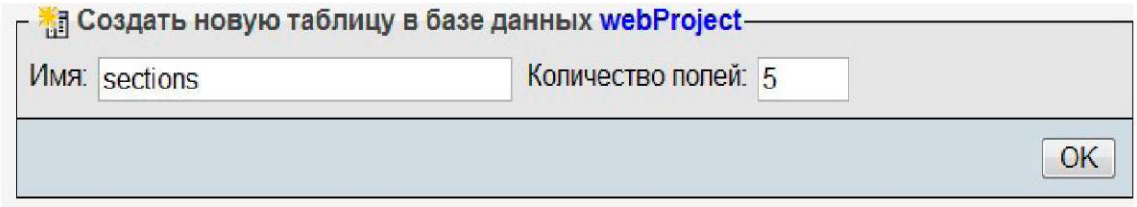
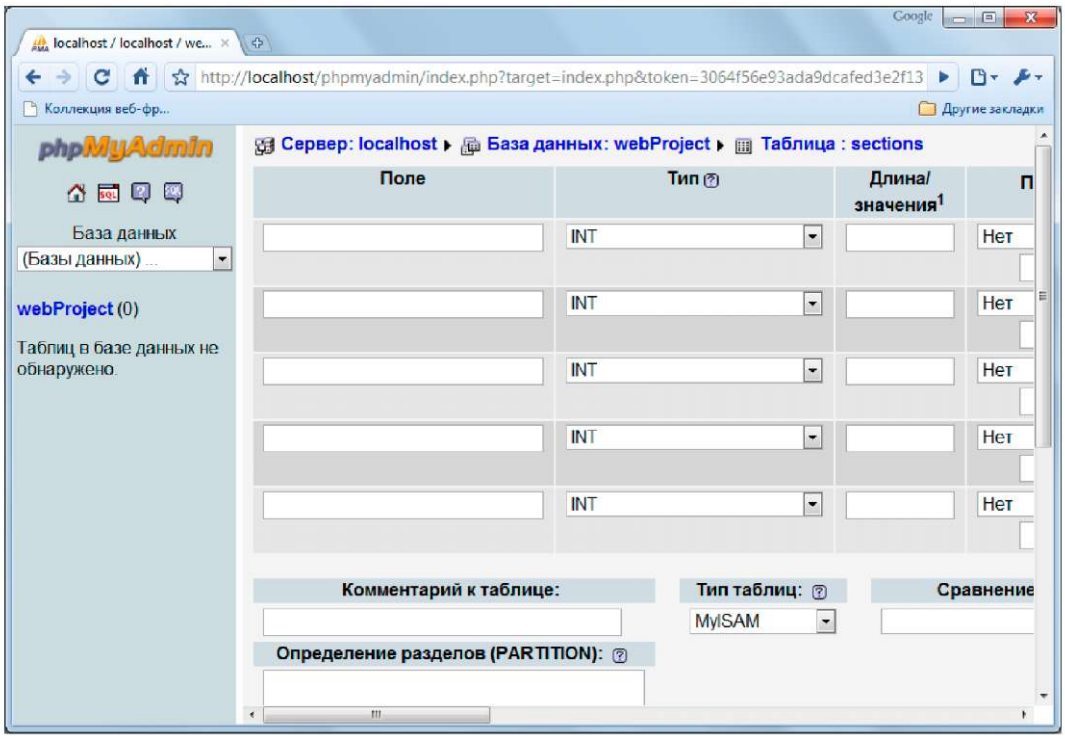


Рисунок 12. Ввод параметров для новой таблицы БД 16. Далее страница обновится и появится форма для добавления параметров каждого из пяти полей нашей таблицы.



Имя поля: caption; **тип** - VARCHAR, **длина** - 50;

Имя поля: text; **тип** - TEXT;

Имя поля: module; **тип** - VARCHAR, **длина** - 30; Allow **NULL** - ON (что означает, что мы разрешаем пустое значение для данного поля)

Имя поля: position; **тип** - INT; **По умолчанию** - Как определено **(Ю).**

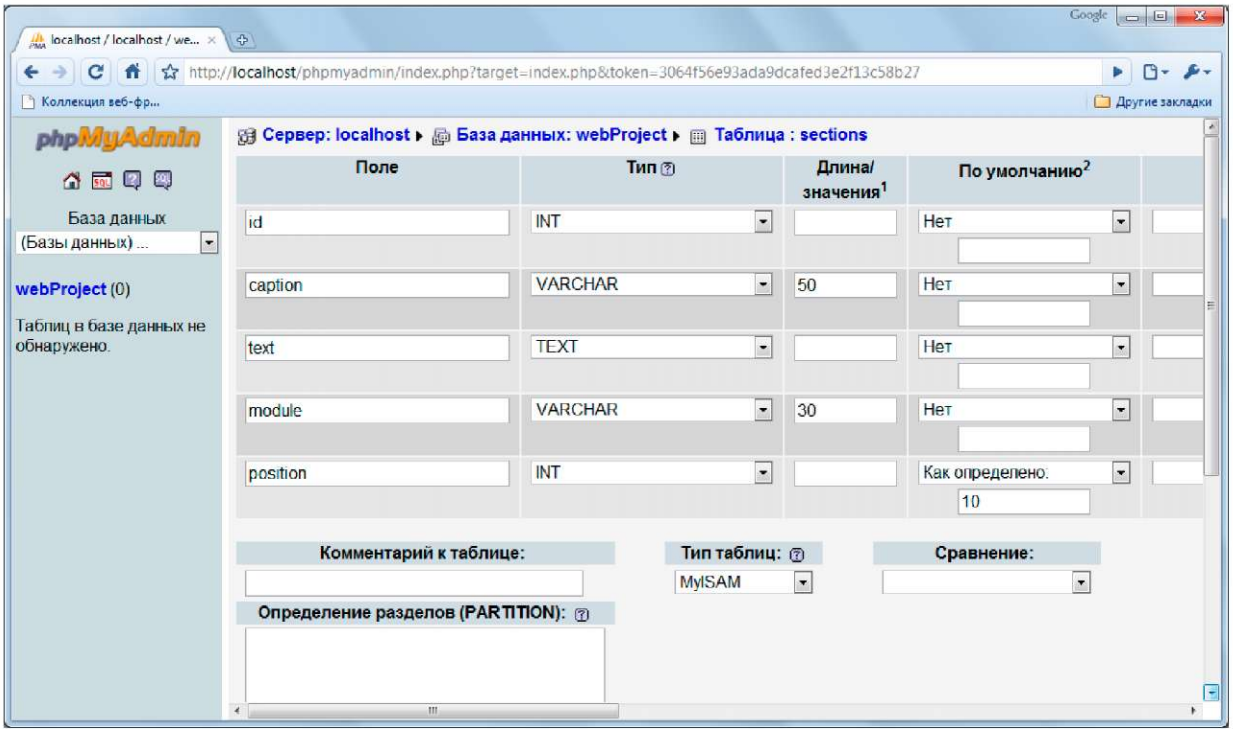


Рисунок 14. Добавление полей в БД 18. Обратите внимание на то, что в любой момент вы можете добавить новое поле, указав, какое количество полей вы хотите добавить и нажав на кнопку ОК:

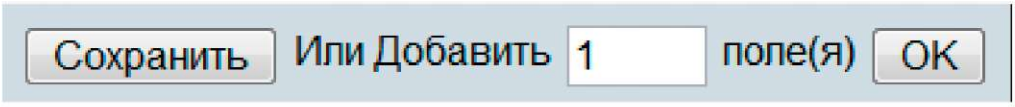


Рисунок 15. Добавление полей в БД 19. Нажмите на кнопку «Сохранить». Страница после обновления примет следующий вид:

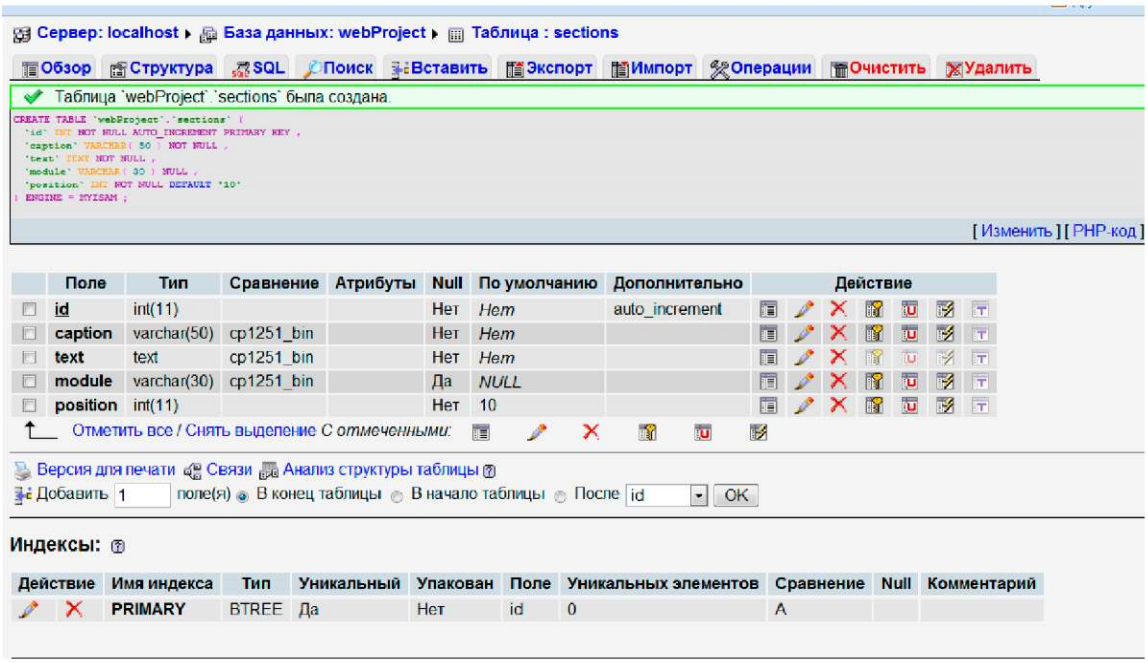


Рисунок 16. Интерфейс страницы «Структура» после сохранения таблицы

```
CREATE TABLE `webProject`.`sections` (
  `id` INT NOT NULL AUTO_INCREMENT PRIMARY KEY ,
  `caption` VARCHAR( 50 ) NOT NULL ,
  `text` TEXT NOT NULL ,
  `module` VARCHAR( 30 ) NULL ,
  `position` INT NOT NULL DEFAULT '10'
) ENGINE = MYISAM ;
```

Рисунок 17. SQL-запрос, выполненный для создания таблицы

20. После создания таблицы вы можете в любой момент изменить ее структуру (добавлять, редактировать или удалять поля и т.д.)

21. Добавление нового поля в таблицу осуществляется с использованием следующей формы (Рисунок 18):

Рисунок 18. Вид формы для добавления нового поля в существующую таблицу

22. Чтобы добавить запись, перейдите во вкладку «Вставить» и заполните форму согласно следующему изображению:

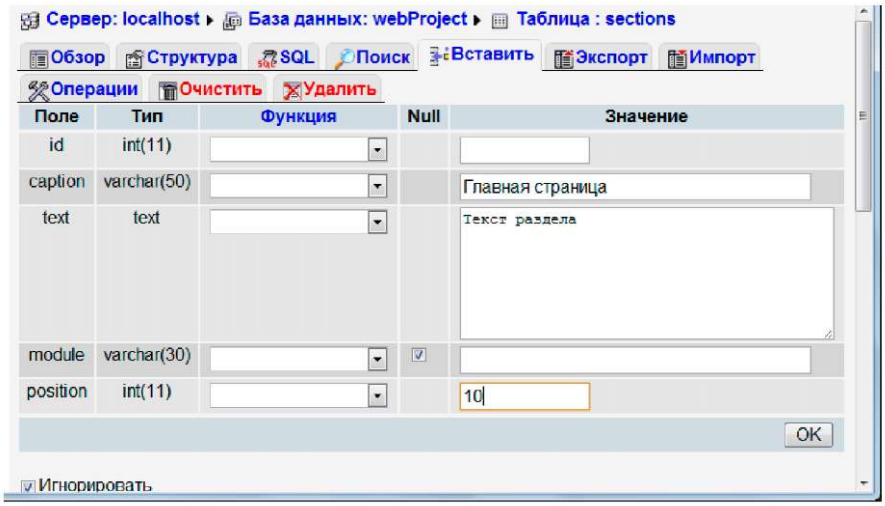


Рисунок 19. Добавление новых записей Нажмите ОК. После обновления страницы появится сообщение (Рисунок 20):

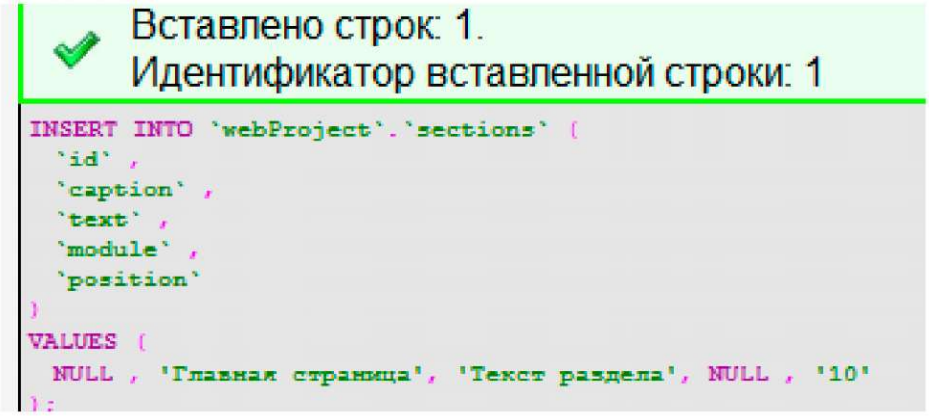


Рисунок 20. SQL-запрос, выполненный для вставки новой записи в таблицу

23. Самостоятельно

24. Добавьте новое поле «idParent» в таблицу (целый тип). Данное поле будет указывать на «родителя» данной записи и позволит организовать древовидную структуру на сайте.

25. Спроектируйте и создайте таблицу в этой же БД для хранения новостей и организации новостной ленты на сайте.

26. Добавьте нового пользователя для созданной БД и установите ему привилегии, позволяющие работать только с данной БД (webProject).

Контрольные вопросы:

1) Пояснить особенности модели «клиент-сервер» в технологии баз данных.
2) Пояснить особенности модели файлового сервера. Достоинства и недостатки.
3) Пояснить особенности модели удаленного доступа к данным. Достоинства и недостатки.
4) Пояснить особенности модели сервера баз данных. Достоинства и недостатки.
5) Пояснить особенности модели сервера приложений. Достоинства и недостатки.
6) Перечислить свойства транзакций и способы завершений транзакций.

Список литературы, рекомендуемый к использованию по данной теме:
Основная: 1-11
Дополнительная: 1-11
Интернет-ресурсы: 1-7
Программное обеспечение: 1-4

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению практических работ
по дисциплине «Безопасность баз данных»
для студентов направления подготовки
10.03.01 «Информационная безопасность»
Направленность (профиль)
«Организация и технологии защиты информации (по отрасли или в сфере профессиональной деятельности)»

Ставрополь
2026

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

1. Цель и задачи освоения дисциплины

Целью дисциплины «Безопасность баз данных» является формирование у обучающихся понимания основ информационной безопасности систем баз данных для последующего практического использования в науке и образовании.

Задачами дисциплины являются:

- Теоретическая и практическая подготовка бакалавров к настройке параметров безопасности баз данных;
- Объяснение принципов построения подсистем защиты в СУБД, средств и методов несанкционированного доступа к ресурсам СУБД;
- Привитие студентам системного подхода к проблеме защиты информации в СУБД; механизмов защиты информации и возможностей по их преодолению.

2. Перечень планируемых результатов обучения по дисциплине (модулю), соотнесённых с планируемыми результатами освоения образовательной программы

Код, формулировка компетенции	Код, формулировка индикатора	Планируемые результаты обучения по дисциплине (модулю), характеризующие этапы формирования компетенций, индикаторов
ПК-1 Способен обеспечивать защиту от несанкционированного доступа сооружений и средств связи сетей электросвязи (за исключением сетей связи специального назначения) в процессе их эксплуатации	ИД-1 Способен проводить мониторинг функционирования СССЭ, защищённости от НСД сооружений и СССЭ	- Применяет методы контроля функционирования СССЭ, их защищенности от НСД - Анализирует средства мониторинга работоспособности и эффективности применяемых программных, программно-аппаратных (в том числе криптографических) и технических средств защиты СССЭ от НСД
	ИД-2. Способен к управлению функционированием СССЭ, защищённостью от НСД сооружений СССЭ	- определяет необходимый состав, особенности размещения СССЭ, а также программных, программно-аппаратных (в том числе криптографических) и технических средств и систем защиты СССЭ от НСД; - организовывает и проводит монтаж и настройку СССЭ, а также

		программных, программно-аппаратных (в том числе криптографических) и технических средств и систем защиты СССЭ от НСД; - устанавливает и настраивает программное обеспечение, необходимое для управления СССЭ и средствами их защиты от НСД
	ИД-1 Способен проводить мониторинг функционирования СССЭ, защищённости от НСД сооружений и СССЭ	- Применяет методы контроля функционирования СССЭ, их защищенности от НСД - Анализирует средства мониторинга работоспособности и эффективности применяемых программных, программно-аппаратных (в том числе криптографических) и технических средств защиты СССЭ от НСД
ПК-3 Способен обеспечивать защиту информации в автоматизированных системах в процессе их эксплуатации	ИД-1 Способен администрировать системы защиты информации автоматизированных систем	Устанавливает и настраивает операционные системы, системы управления базами данных, компьютерные сети и программные системы с учетом требований по обеспечению защиты информации

Раздел 1. Архитектура баз данных

Практическое занятие №1

Тема 1 - 2. Базы данных и файловые системы. Функции СУБД.

1. Цель работы: изучение технологии создания таблиц данных с использованием конструктора таблиц данных

Формируемые компетенции: ПК-1, ПК-3

2. Подготовка к занятию

1. Изучить (повторить) теоретический материал.
2. Ознакомиться с программой работы.
3. Подготовить отчет о работе.
4. Ответить на контрольные вопросы.

3. Распределение времени занятия:

Всего: 90 мин

Вступительная часть 2 мин

Проверка готовности студентов к занятию 5 мин

Основная часть – 70 мин

1. Исследование мандатной политики безопасности 70 мин

Проверка выполнения практического занятия 10 мин

Заключительная часть 3 мин

4. Правила работы в лаборатории

К работе в лаборатории допускаются лица, изучившие правила и меры безопасности, сдавшие зачет по ним и усвоившие порядок выполнения лабораторной работы.

4.1. Требования безопасности перед началом работ

- ЗАПРЕЩАЕТСЯ: переодеваться, пользоваться огнем, курить, принимать пищу в лаборатории.

- Убедиться в целостности электрических розеток и разъемов. В лаборатории необходимо быть в сменной обуви.

- Включение компьютера производить только после получения допуска по выполняемой работе и разрешения преподавателя или лаборанта.

4.2. Требования безопасности во время работы

- выполняя практическое занятие, студенты обязаны использовать только вычислительную технику, периферийное оборудование, соединительные кабели, измерительное оборудование и носители информации, непосредственно относящиеся к данному лабораторному занятию;

- подключение и отключение составляющих вычислительного комплекса производить только при полном снятии напряжения со всех составляющих вычислительного комплекса;

- при обнаружении неисправностей в оборудовании немедленно отключить источники питания и доложить об этом руководителю занятий или лаборанту.

4.3. Требования безопасности по окончании работы

- доложить руководителю занятий или лаборанту о завершении работ;
- привести в порядок и сдать рабочее место лаборанту, и доложить руководству

1 ОСНОВНЫЕ СВЕДЕНИЯ О ТАБЛИЦАХ ДАННЫХ

Таблицы составляют основу баз данных, в них хранится вся необходимая информация. Таблицы могут дополняться новыми данными, редактироваться, их можно включать в БД или исключать из БД. Можно просматривать данные таблиц с помощью форм, упорядочивать данные по заданному критерию, осуществлять поиск требуемых данных. Информация, содержащаяся в таблицах, может быть использована для создания отчётов. Кроме того, используя диаграммы, можно графически представить информацию, содержащуюся в таблице данных.

Таблица состоит из строк и столбцов и имеет уникальное имя. В каждой из таблиц содержится информация о каких-либо объектах одного типа. В Visual FoxPro можно создавать как таблицы, входящие в базу данных, так и отдельные таблицы, называемые свободными таблицами, аналогичные создаваемым в предыдущих версиях FoxPro.

Таблицы данных FoxPro хранятся в файлах с расширением *dbf* и как и все объекты в Visual FoxPro, имеют имена. При задании имени необходимо придерживаться ограничений, накладываемых операционной системой на количество символов в имени файла. Наименование таблицы может содержать буквы, цифры и знак подчёркивания. Создавая новую таблицу, необходимо помнить, что имя таблицы должно быть уникально. Если в системе уже имеется таблица с таким именем, на экране появляется запрос, заменить ли существующую таблицу новой. Свободную таблицу данных можно создать с использованием мастера или конструктора таблиц данных.

Для создания таблиц с помощью конструктора таблиц данных необходимо запустить конструктор таблиц из системного меню Visual FoxPro. Для этого в меню *File* (Файл) выбирается команда *New* (Новый). В открывшемся диалоговом окне New необходимо установить опцию *Table* (Таблица) и нажать кнопку *New file*. В появившемся окне Create в поле Enter задать имя создаваемой таблицы данных. Убедившись, что в поле Тип файла установлен тип сохраняемого файла **.dbf*, а в раскрывающемся списке Сохранить в правильно указана папка, в которой будет располагаться создаваемая таблица, нажать кнопку Сохранить. В результате выполнения этих действий откроется окно конструктора таблиц *Table Designer*.

Окно конструктора таблиц *Table Designer* (Рисунок 1) содержит три вкладки, предназначенные для определения следующих параметров:

— Fields (Поля) – для описания характеристик полей таблицы;

— Indexes (Индексы) – для указания индексов таблицы;

— Table (таблица) – для задания условия достоверности вводимых данных, а также триггеров добавления, удаления и модификации.

Поля таблицы предназначены для хранения данных: чисел, текстов, дат, графики и т. д. При определении полей таблицы используется вкладка *Fields*, позволяющая ввести наименование поля, тип размещаемых данных, ширину поля. Для числовых полей необходимо также задать количество десятичных знаков. Наименования полей таблицы вводятся в строке ввода столбца *Name* (Имя). При задании названий полей можно использовать буквы, цифры и знак подчёркивания. Все попытки ввести специальные символы Visual FoxPro проигнорирует.

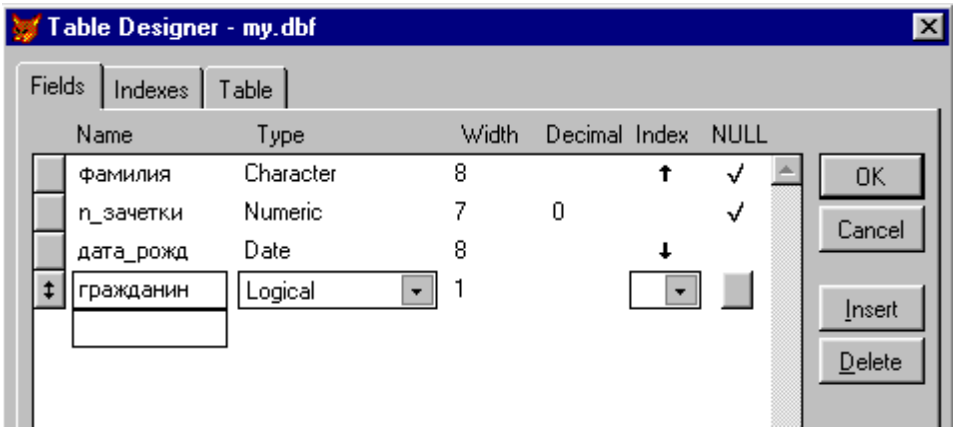


Рисунок 1 - Окно конструктора таблиц Table Designer

Для определения типа данных, размещаемых в поле, используется тип поля, его ширина и количество знаков после запятой. Для их ввода предназначены столбцы *Type* (Тип), *Width* (Ширина) и *Decimal* (Десятичные) вкладки *Fields* конструктора таблиц.

В Visual FoxPro допустимыми являются типы полей, указанные ниже:

— Текстовый (Character, Character binary) – текстовые поля могут содержать буквы, цифры и специальные символы. Максимальная ширина поля составляет 254 символа. Тип Character binary используется в том случае, если не требуется учитывать кодовую страницу отображаемых данных;

— Числовой (Integer, Numeric, Float, Double) – числовые поля содержат числа. Integer отображает целые числа от –2147483647 до + 2147483647. Числовые поля типа Numeric и Float отображают данные с фиксированной точкой в диапазоне от $-0{,}9999999999*10^{+19}$ до $+0{,}9999999999*10^{+20}$;

Тип данных Double используется для хранения данных с высокой точностью в диапазоне от ±4,94065648541247*10^{-324} до ±1,79769313486232*10^{+308}

— Денежный (Currency) – в поле денежного типа могут содержаться числа от -922337203685477,5807 до +922337203685477,5807;

— Дата (Date) – в поле типа Date может содержаться любая дата от 01.01.0001 до 31.12.9999 года;

— Дата и время (DateTime) – в поле типа DateTime может содержаться любая дата от 01.01.0001 до 31.12.9999 г. и время от 00:00:00 a.m. (до полудня) до 11:59:59 p.m. (после полудня);

— Логический (Logical) – содержит логическое значение True, обозначаемое как .T. (Истина), или False, обозначаемое как .F. (Ложь);

— Текстовое поле (Memo, Memo binary) – Memo – поле содержит символьные данные большого объёма;

— Двоичное поле произвольной длины (General) – поле данного типа предназначено для хранения в таблицах изображений и других двоичных данных.

Для каждого поля можно определить признак, разрешающий при вводе данных оставлять это поле пустым. Для этого используется опция *NULL* в описании поля таблицы. Столбец Index используется для указания упорядочения записей по данным этого поля (по возрастанию или убыванию).

Ввод полей в окне конструктора таблицы осуществляется последовательно. После определения всех необходимых параметров первого поля осуществляется переход на новую строку и определяются характеристики следующего поля таблицы.

На вкладке *Fields* справа расположены четыре кнопки. Кнопка *OK* предназначена для закрытия окна конструктора таблицы и сохранения всех изменений, внесённых в структуру таблицы. Если структура таблицы была изменена, но от этого нужно отказаться, необходимо воспользоваться кнопкой *Cancel* (Отмена). Для добавления в таблицу нового поля нужно установить курсор на поле, выше которого предполагается разместить новое поле, и нажать кнопку *Insert* (Вставить). Будет добавлена пустая строка, в которую можно ввести параметры нового поля. Для удаления поля таблицы необходимо перейти на строку с описанием данного поля и нажать кнопку *Delete* (Удалить).

Для создания индексов таблицы используется вкладка *Indexes* (Индексы) (Рисунок 2) окна конструктора таблиц Table Designer.

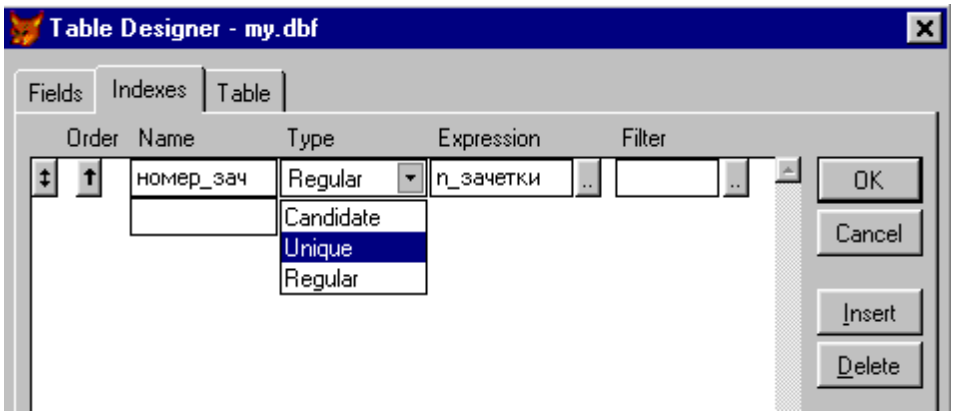


Рисунок 2 - Вкладка Indexes конструктора таблицы.

Все индексы в Visual FoxPro имеют имена, задаваемые в поле *Name* (Имя). Слева от имени индекса в столбце *Order* (Упорядочение) располагается переключатель, определяющий, как будут упорядочены значения индексного выражения. По умолчанию при создании индекса в данном поле появляется стрелка, направленная вверх, которая показывает, что значения индексного выражения упорядочены по возрастанию. Если стрелка направлена вниз, это говорит о том, что значения упорядочены по убыванию. Для изменения способа упорядочения можно использовать клавишу *SpaceBar* или щелчок мыши.

Список *Type* (Тип) используется для задания типа создаваемого индекса и содержит следующие значения:

— Primary (Первичный) – создаёт уникальный индекс, который используется для связывания таблиц и определения условий целостности данных. Поля, входящие в первичный ключ, не должны допускать ввода пустых значений. Таблица может иметь первичный ключ, если она входит в состав БД. Первичный ключ в таблице может быть только один;

— Candidate (Кандидат) – создаётся уникальный индекс, в том числе и в свободной таблице, 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. Количество индексов типа Candidate в одной таблице неограниченно;.

— 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 индексов Unique.

Значение индекса или индексного выражения вводится в поле *Expression* (Выражение). Можно ввести индексное выражение непосредственно в поле ввода или для формирования выражения использовать диалоговое окно конструктора выражений *Expression Builder* (Построитель выражения), для запуска которого необходимо нажать кнопку, расположенную справа от поля *Expression.*

С помощью кнопок *Insert* и *Delete* на вкладке *Indexes* (Индексы) можно добавлять в таблицу новые индексы и удалять существующие.

Вкладка *Table* используется только для таблиц, входящих в БД.

Структуру таблицы, созданную с помощью мастера или конструктора таблиц, можно модифицировать, то есть изменить наименование любого поля и его тип, вставить новое поле или удалить существующее, изменить порядок следования полей в таблице. Чтобы модифицировать таблицу, её нужно открыть в конструкторе таблиц.

Для этого в меню *File* (Файл) выбирается команда *Open* (Открыть) и в появившемся окне *Open* в поле Тип файлов устанавливается тип открываемого файла **.dbf*. Убедившись, что в раскрывающемся списке Папка правильно указана папка, в которой располагается таблица, необходимо выделить модифицируемую таблицу и нажать кнопку *OK*. В результате выполнения этих действий откроется указанная таблица данных, а в системном меню *View* появится команда *Table Designer*. Выбор этой команды приведет к открытию окна конструктора таблиц *Table Designer* со структурой данной таблицы.

Открыть окно конструктора таблиц *Table Designer* со структурой активной (предварительно открытой) таблицы можно также и через командное окно, используя команду *MODIFY STRUCTURE*. В результате на экране открывается диалоговое окно *Table Designer* (Конструктор таблицы), содержащее структуру модифицируемой таблицы.

Ошибки, допущенные при задании имени поля или его типа легко устраняются. Для этого нужно установить курсор на имени поля, которое нужно изменить, и, используя клавишу *Backspace* или *Delete*, удалить ошибочные символы. После этого ввести правильное имя поля. Для изменения типа поля установить курсор в столбец *Type* и выбрать из списка требуемое значение. Следует учесть, что изменение типов полей таблицы, содержащих данные, может привести к потере информации.

Порядок расположения полей, заданный при создании структуры таблицы, можно изменить. Для этого необходимо выполнить следующие действия:

1) Установить курсор на поле, расположение которого нужно изменить. На кнопке слева от имени поля появится значок перемещения поля в виде двунаправленной стрелки  ;

2) Установить курсор на значок перемещения;

3) Нажать кнопку мыши и, удерживая её нажатой, переместить значок вверх или вниз на требуемое место в структуре;

4) Отпустить кнопку мыши. Поле изменит своё местоположение;

5) После создания или изменения структуры таблицы, используя команду Append можно ввести данные в таблицу;

6) Команда *Browse* используется для просмотра и редактирования данных таблицы.

2 ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

Лабораторная работа выполняется в среде реляционной СУБД Visual FoxPro. Для выполнения лабораторной работы необходимо:

1) Запустить СУБД Visual FoxPro;

2) Повторить основной материал по технологии создания таблиц данных с использованием мастера и конструктора таблиц данных (см. список литературы, конспект лекций, справочную систему Visual FoxPro);

3) Изучить описание данной лабораторной работы, выполняя все описанные действия на компьютере в среде СУБД Visual FoxPro;

4) Выбрать предметную область, определить для данной предметной области два объекта и их свойства, представить эти объекты в терминах реляционных отношений - в виде таблиц с атрибутами (столбцами);

5) Используя командное окно и конструктор таблиц данных, создать таблицы данных (см. пункт 4), ввести в таблицы данные;

6) Поработать с таблицами данных, выполняя просмотр, редактирование, модификацию структуры таблиц и данных;

7) Оформить отчет по лабораторной работе.

Контрольные вопросы:

1. Перечислить основные модели данных.
2. Пояснить классификацию моделей данных.
3. Пояснить принцип модели «сущность-связь».

Список литературы, рекомендуемый к использованию по данной теме:
Основная: 1-11
Дополнительная: 1-11
Интернет-ресурсы: 1-7
Программное обеспечение: 1-4

Практическое занятие №2

Тема 3 - 4. Подходы к организации баз данных. Общие понятия реляционного подхода к организации БД.

1. Цель работы: изучение технологии создания форм с использованием конструктора форм

Формируемые компетенции: ПК-1, ПК-3

2. Подготовка к занятию

1. Изучить (повторить) теоретический материал.
2. Ознакомиться с программой работы.
3. Подготовить отчет о работе.
4. Ответить на контрольные вопросы.

3. Распределение времени занятия:

Всего: 90 мин

Вступительная часть 2 мин

Проверка готовности студентов к занятию 5 мин

Основная часть – 70 мин

Проверка выполнения практического занятия 10 мин

Заключительная часть 3 мин

4. Правила работы в лаборатории

К работе в лаборатории допускаются лица, изучившие правила и меры безопасности, сдавшие зачет по ним и усвоившие порядок выполнения лабораторной работы.

4.1. Требования безопасности перед началом работ

- ЗАПРЕЩАЕТСЯ: переодеваться, пользоваться огнем, курить, принимать пищу в лаборатории.

- Убедиться в целостности электрических розеток и разъемов. В лаборатории необходимо быть в сменной обуви.

- Включение компьютера производить только после получения допуска по выполняемой работе и разрешения преподавателя или лаборанта.

4.2. Требования безопасности во время работы

- выполняя практическое занятие, студенты обязаны использовать только вычислительную технику, периферийное оборудование, соединительные кабели, измерительное оборудование и носители информации, непосредственно относящиеся к данному лабораторному занятию;

- подключение и отключение составляющих вычислительного комплекса производить только при полном снятии напряжения со всех составляющих вычислительного комплекса;

- при обнаружении неисправностей в оборудовании немедленно отключить источники питания и доложить об этом руководителю занятий или лаборанту.

4.3. Требования безопасности по окончании работы

- доложить руководителю занятий или лаборанту о завершении работ;
- привести в порядок и сдать рабочее место лаборанту, и доложить руководству

1 ОСНОВНЫЕ СВЕДЕНИЯ О ТЕХНОЛОГИИ СОЗДАНИЯ ФОРМ

Для просмотра, ввода и редактирования данных, хранящихся в таблицах, используются *формы*, являющиеся наглядным средством представления информации. Формы, предназначенные для ввода документов, должны выглядеть на экране точно так же, как стандартные бланки этих документов. Важным преимуществом форм является и то, что они позволяют работать не с одной, а с несколькими связанными таблицами.

Пользователю приложения нет необходимости знать, что такое СУБД, какие команды используются для добавления или удаления записей в таблицах. Он может даже вообще не знать, с использованием каких программных средств создавалось приложение. Для него главным является перемещение по таблице, добавление новых записей, редактирование и удаление имеющихся. Все эти возможности предоставляют формы.

При создании форм в Visual FoxPro можно использовать следующие средства:

- AutoForm Wizard – мастер автоформы;
- Form Wizard – мастер форм;
- Form Builder – построитель формы;
- Builder – построитель объектов формы;
- Form Designer – конструктор формы.

Форма сохраняется в файле с заданным именем и расширением **.scx*.

Процесс создания формы в конструкторе форм состоит в размещении в форме объектов и определении свойств, а также связанных с ними событий и выполняемых действий. Для открытия окна конструктора форм при создании формы необходимо выполнить команду *New* (Новый) из меню *File* (Файл). В открывшемся диалоговом окне *New* выбрать опцию *Form* (Форма) и нажать кнопку *New File* (Новый файл).

Окно конструктора форм показано на рисунке 1, оно содержит панели инструментов: *Color Palette* (Цветовая палитра), *Layout* (Расположение), *Form Designer* (Конструктор форм) и *Form Controls* (Элементы управления формы), используемые при работе в конструкторе. В окне конструктора находится и новая форма, с которой можно начинать работать. Если необходимые панели инструментов отсутствуют, для их отображения на экране нужно установить метки в соответствующих опциях меню *View* (Вид) или установить флажки выбора панелей инструментов в диалоговом окне *Toolbars* (Панели инструментов).

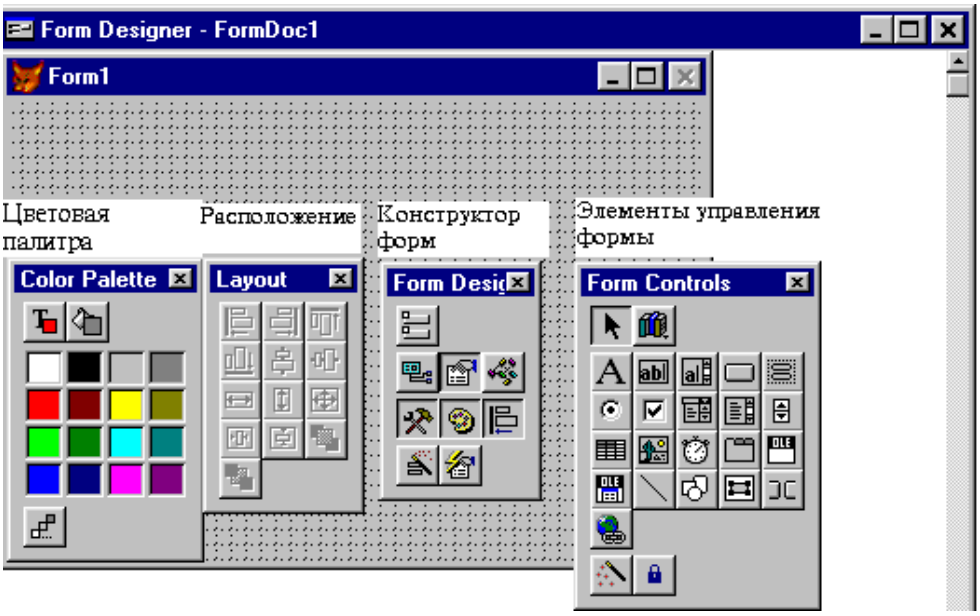


Рисунок 1 - Окно конструктора форм

Панель инструментов конструктора форм (Form Designer) содержит кнопки вызова панелей инструментов: *Form Controls* (Элементы управления формы), *Color Palette* (Цветовая палитра) и *Layout* (Расположение). С помощью этой панели можно выполнять и некоторые дополнительные действия по управлению формой. Краткое назначение кнопок данной панели инструментов приводится ниже:

 - Set Tab Order (Порядок объектов), переключает конструктор форм в режим установления порядка обхода объектов формы;

 - Data Environment (Окружение данных), открывает окно определения среды окружения формы;

 - Code Window (Окно кода), открывает окно просмотра исходного кода формы;

 - Color Palette Toolbar (Панель инструментов Цветовая палитра), отображает на экране панель инструментов Color Palette;

 - Form Builder (Построитель форм), вызывает построитель формы;

 - Properties Window (Окно свойств), открывает на экране окно свойств объектов формы;

 - Forms Controls Toolbar (Панель инструментов Элементы управления формы) , вызывает на экран панель инструментов Form controls;

 - Layout (Расположение) , вызывает на экран панель инструментов Layout;

 - Auto Format (Автоформат), вызывает построитель автоформата для выбранных объектов формы.

Панель инструментов *Form Controls* (Элементы управления формы) используется для размещения в форме объектов. Краткое описание кнопок этой панели приводится ниже:

 - Select Objects (Выбор объектов), указатель выделения, позволяет выбирать в форме объекты;

 - View Classes (Просмотр классов), позволяет выбрать класс для создаваемых в форме объектов;

 - Label (Метка), создаёт в форме текстовый объект;

 - Text Box (Поле ввода), создаёт в форме поле ввода;

 - Edit Box (Поле редактирования) - создаёт в форме поле редактирования;

 - Command Button (Кнопка) - создаёт в форме кнопку управления;

 - Option Group (Переключатель) - создаёт в форме зависимый переключатель;

 - Check Box (Флажок) - создаёт в форме флажок;

 - Grid (Таблица) - создаёт в форме объект в виде таблицы для размещения полей таблицы данных;

 - Combo Box (Раскрывающийся список) - создаёт в форме раскрывающийся список;

 - List Box (Список) - создаёт в форме список;

 - Spinner (Счётчик) - создаёт в форме поле ввода значения в виде счётчика;

 - Line (Линия) - создаёт в форме линию;

 - Shape (Контур) - создаёт в форме контур;

 - Container (Контейнер) - создаёт в форме контейнер;

 - Image (Изображение), размещает в форме рисунок;

 - Command Group (Группа кнопок), размещает в форме группу кнопок;

 - Timer (Таймер) - создаёт в форме объект типа таймера;

 - Page Frame (Вкладка), размещает в форме страницы с вкладками;

 - ActiveX Bound Control (OleBoundControl – ActiveX объект), отображает содержимое OLE-объекта, хранящего в поле типа General;

 - ActiveX Control (OleControl - ActiveX объект), создаёт OLE-объект;

 - Separator (Разделитель), размещает на панели инструментов разделитель кнопок;

 - Builder Lock (Закрепитель построителя), закрепляет выбор построителя;

 - Button Lock (Закрепитель кнопки), закрепляет выбранную кнопку на панели инструментов.

Для выравнивания объектов, размещённых в форме, удобно использовать панель инструментов *Layout* (Расположение) со следующими инструментами:

 - Align Left Sides (По левому краю), выравнивает выбранные объекты по левому краю самого левого объекта;

 - Align Top Edges (По верхнему краю), выравнивает выбранные объекты по верхнему краю самого верхнего объекта;

 - Align Right Sides (По правому краю) - выравнивает выбранные объекты по правому краю самого правого объекта;

 - Align Bottom Edges (По нижнему краю) - выравнивает выбранные объекты по нижнему краю самого нижнего объекта;

 - Align Vertical Center (По вертикали) - выравнивает выбранные объекты по вертикали;

 - Align Horizontal Center (По горизонтали) - выравнивает выбранные объекты по горизонтали;

 - Center Vertically (Относительно вертикали, проходящей через центр), центрирует выбранные объекты относительно вертикальной средней линии формы;

 - Center Horizontally (Относительно горизонтали, проходящей через центр), центрирует выбранные объекты относительно горизонтальной средней линии формы;

 - Same Width (Одинаковая ширина), устанавливает одинаковую ширину для выбранных объектов формы;

 - Same Size (Одинаковый размер), устанавливает одинаковую ширину и высоту для выбранных объектов формы;

 - Same Height (Одинаковая высота), устанавливает одинаковую высоту для выбранных объектов формы;

- Send to Back (Позади), направляет выбранный объект на самый нижний слой формы;

- Bring to Form (Поверх), направляет выбранный объект на самый верхний слой формы.

Процесс создания формы состоит из следующих действий:

— 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 и т.д.;

— Настройка свойств размещённых в форме объектов.

Форма, как и все располагаемые в ней объекты, имеет свойства, используя которые можно задать её размер, координаты верхнего левого угла, стиль рамки обрамления, заголовок, цвет и т. д.

Настройка параметров формы осуществляется в окне свойств *Properties* (Свойства) (Рисунок 2), для открытия которого нужно установить курсор на свободную от объектов поверхность формы и выбрать команду *Properties* (Свойства) из меню *View* (Вид).

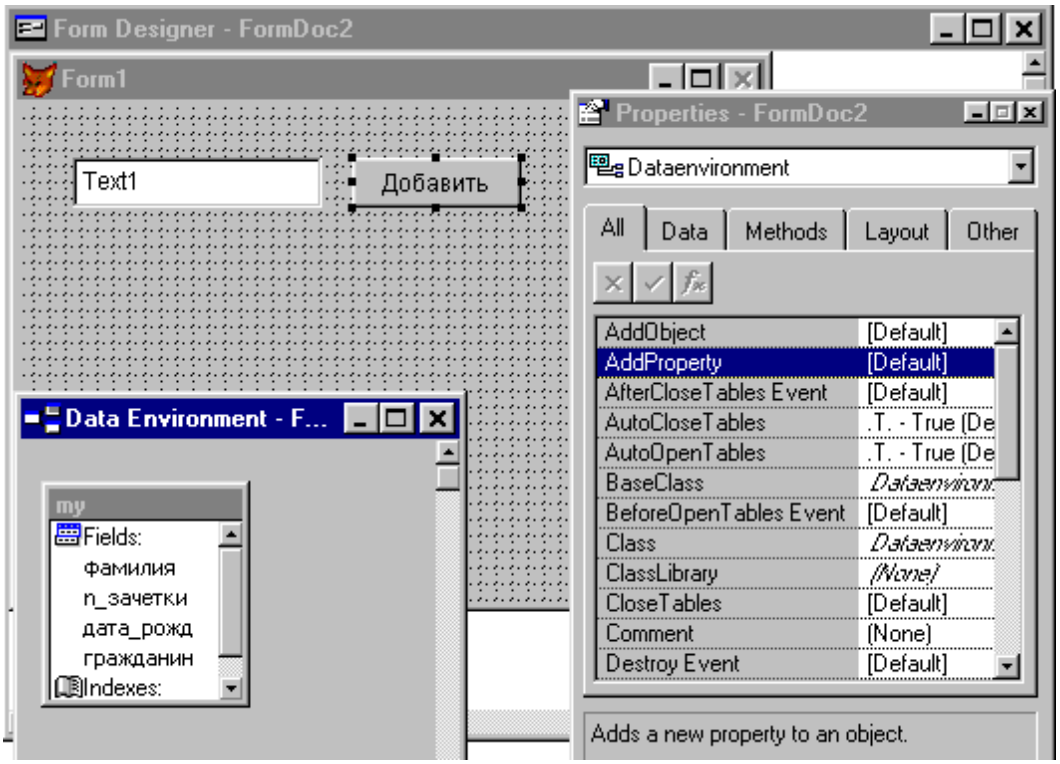


Рисунок 2 - Окно Properties и окно Data Environment.

После создания форма, по умолчанию, располагается в верхнем левом углу главного окна Visual FoxPro. Для изменения её положения можно использовать свойства *Left* (Левый) и *Top* (Верхний), указывающие расстояние в пикселях от левого и верхнего края, соответственно, а также мышь. Если использовать мышь для изменения положения формы, то нужно установить курсор на заголовок формы, нажать левую кнопку мыши и, удерживая её, переместить форму в окне конструктора в нужное место расположения. Для размещения формы в центре главного окна Visual FoxPro необходимо в окне свойств установить для свойств *AutoCenter* (Автоцентр) значение *True* (Истина).

Для задания текста заголовка, располагающегося в верхней части формы, предназначено свойство *Caption* (Надпись) окна свойств. Чтобы отредактировать заголовок, нужно открыть окно *Properties*, выделить свойство *Caption* и в поле ввода, ставшее активным, ввести с клавиатуры заголовок формы. Если появится необходимость, чтобы форма вообще не содержала заголовок, то необходимо установить для свойства *Title Bar* (Строка заголовка) значение *Off*. Свойство *Name* используется для задания программного имени объекта (в том числе и формы), по которому можно обратиться к данному объекту.

Стиль обрамления формы можно задать с помощью свойства *BorderStyle* (Стиль рамки), он может принимать следующие значения:

— 0 – No Border (Нет рамки), форма не имеет рамки;

— 1 – Fixed Single (Одинарная рамка), неизменяемая одинарная рамка;

— 2 – Fixed Dialog (Двойная рамка), неизменяемая двойная рамка;

— 3 – Sizable (Изменяемая), изменяемая рамка (размеры формы можно изменять при выполнении).

При помощи свойства *BackColor* (Цвет фона) задается цвет фона формы.

Свойство *WindowState* (Состояние окна) определяет размер формы при её вызове и может принимать одно из следующих значений:

— Normal – форма имеет размеры, определённые её свойствами;

— Minimized (Windows only) – форма сворачивается в значок;

— Maximized (Максимизированное) – форма разворачивается на весь экран.

При создании формы, предназначенной для редактирования или просмотра данных таблиц, в конструкторе форм необходимо определить среду окружения, т. е. задать таблицы, используемые в форме, и установить связи между ними.

Следует учесть, что при создании форм с помощью мастера и размещении объектов в форме с помощью построителя, среда окружения создаётся Visual FoxPro автоматически без участия разработчика.

При определении среды окружения нужно выполнить следующие действия:

— Добавить таблицы и представления, используемые в форме;

— Установить индексы для таблиц;

— Установить для таблицы отношения, необходимые для создания формы.

Вся эта информация, относящаяся к среде окружения, хранится в файле описания формы. Для создания среды окружения формы предназначено диалоговое окно *Data Environment* (Среда окружения) (Рисунок 2), открыть которое можно одним из следующих способов:

— Выбрать команду *Data Environment* (Среда окружения) из меню *View* (Вид);

— Нажать кнопку *Data Environment*  на панели инструментов *Form Designer* (Конструктор форм);

— Выбрать команду контекстного меню формы *Data Environment*.

При открытии окна среды окружения *Data Environment* в основное меню добавляется пункт *Data Environment*. Для работы в окне *Data Environment* можно использовать команды из меню *Data Environment* или контекстного меню, позволяющие добавить в окружение таблицы, просмотреть их в режиме *Browse*, открыть окно свойств окружения для задания различных параметров.

Для добавления таблицы в среду окружения можно выполнить одно из следующих действий:

1. Выбрать команду контекстного меню *Add*;
2. Выбрать команду *Add* из меню *Data Environment*.

При этом откроется диалоговое окно *Add Table or View* (Добавить таблицу или представление данных), содержащее список таблиц открытой базы данных. В поле *Database* выбирается база данных, а в поле *Tables in database* находится таблица, которую следует выделить и нажимается кнопка Add. При этом выбранная таблица размещается в среде окружения формы. Кнопка *Close* позволяет закрыть диалоговое окно. Кнопка *Other* открывает окно *Open*, с помощью которого можно добавить отсутствующую базу данных и таблицы. Опция *Views* (Представления данных) области *Select* (Выбор) позволяет разместить в среде окружения созданные в базе данных представления данных.

Объекты из среды окружения (таблицы, представления, их поля) можно разместить на форме путем перетаскивания объектов при нажатой правой кнопке мыши.

Для запуска формы используется команда *do Form <имя формы>*.

2 ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

Лабораторная работа выполняется в среде реляционной СУБД Visual FoxPro Для выполнения лабораторной работы необходимо:

1) Запустить СУБД Visual FoxPro;

2) Повторить основной материал по технологии создания форм (см. список литературы, конспект лекций, справочную систему Visual FoxPro);

3) Изучить описание данной работы, выполняя все указанные действия на компьютере в среде СУБД Visual FoxPro;

4) Проверить в системе наличие таблиц данных, представлений, запросов и других компонентов, которые будут использованы при работе с формой;

5) Используя командное окно и конструктор форм создать форму для работы с одной таблицей (просмотр, редактирование, добавление, удаление данных) и запустить ее на выполнение;

6) Используя командное окно и конструктор форм создать форму для работы с двумя таблицами данных и запустить ее на выполнение;

7) Оформить отчет по работе.

Контрольные вопросы:

- Перечислить достоинства и недостатки сетевой модели данных.
- Дать понятие реляционной модель данных.
- Перечислить достоинства и недостатки реляционной модель данных.

Список литературы, рекомендуемый к использованию по данной теме:

Основная: 1-11
Дополнительная: 1-11
Интернет-ресурсы: 1-7
Программное обеспечение: 1-4

Практическое занятие №3

Тема 5 - 6. Базисные средства манипулирования реляционными данными. Проектирование реляционных БД

1. Цели работы

1. Ознакомиться с приложениями, включенными в состав СУБД MySQL.

2. Получить навыки управления учетными записями пользователей и определения привилегий.

3. Ознакомиться с утилитами, входящими в состав СУБД MySQL, получить навыки работы с ними.

Формируемые компетенции: ПК-1, ПК-3

2. Подготовка к занятию

1. Изучить (повторить) теоретический материал.
2. Ознакомиться с программой работы.
3. Подготовить отчет о работе.
4. Ответить на контрольные вопросы.

3. Распределение времени занятия:

Всего: 90 мин

Вступительная часть 2 мин

Проверка готовности студентов к занятию 5 мин

Основная часть – 70 мин

Проверка выполнения практического занятия 10 мин

Заключительная часть 3 мин

4. Правила работы в лаборатории

К работе в лаборатории допускаются лица, изучившие правила и меры безопасности, сдавшие зачет по ним и усвоившие порядок выполнения лабораторной работы.

4.1. Требования безопасности перед началом работ

- ЗАПРЕЩАЕТСЯ: переодеваться, пользоваться огнем, курить, принимать пищу в лаборатории.

- Убедиться в целостности электрических розеток и разъемов. В лаборатории необходимо быть в сменной обуви.

- Включение компьютера производить только после получения допуска по выполняемой работе и разрешения преподавателя или лаборанта.

4.2. Требования безопасности во время работы

- выполняя практическое занятие, студенты обязаны использовать только вычислительную технику, периферийное оборудование, соединительные кабели, измерительное оборудование и носители информации, непосредственно относящиеся к данному лабораторному занятию;

- подключение и отключение составляющих вычислительного комплекса производить только при полном снятии напряжения со всех составляющих вычислительного комплекса;

- при обнаружении неисправностей в оборудовании немедленно отключить источники питания и доложить об этом руководителю занятий или лаборанту.

4.3. Требования безопасности по окончании работы

- доложить руководителю занятий или лаборанту о завершении работ;
- привести в порядок и сдать рабочее место лаборанту, и доложить руководству

Запуск MySQL

Управление сервером обычно осуществляется из командной строки. Запуск в Windows 95/98/2000/XP осуществляется через сеанс DOS выполнением следующей команды:

```
D:\usr\local\Mysql\bin\mysqld --standalone
```

Эта команда запустит демон mysql в фоновом режиме. В Windows 95/98 не предусмотрен запуск mysqld в виде службы. В Windows 2000 демон mysql запускается в виде службы.

Можно осуществить запуск winmysqladmin.exe, в этом случае все настройки перечисляются в файле my.ini

При запуске mysqld можно указывать следующие опции:

Таблица 1- Опции команды MySQLD

-?, --help	Справка
-b, --basedir=[path]	Путь к каталогу в котором установлен mysql
-h, --datadir [homedir]	Путь к каталогу, в котором хранятся базы данных.
-l, --log=[filename]	Имя журнала транзакций
-L, --language=[language]	Язык по умолчанию(обычно English).
-P, --port=[port]	Порт для соединения.
--skip-grant-tables	Игнорировать таблицы привилегий. Это дает любому ПОЛНЫЙ доступ ко всем таблицам. Не следует предоставлять обычным пользователям разрешений на запуск mysqld.
--skip-name-resolve	Позволяет предоставлять доступ только тем хостам, чьи IP-адреса указаны в таблицах привилегий. Ипользуется для более высокого уровня защиты.

--skip-networking	Использовать подключения только через интерфейс localhost.
-V, --version	Вывести информацию о версии.

Наличие в статусной строке иконки светофора с активным зеленым цветом указывает на то, что сервер запущен (см. рис 1).

Рисунок 7 - Приложение winmysqladmin запущено

Теперь можно попытаться войти в сервер. В случае, если предполагается управление сервером через консоль, то необходимо использовать команду **mysql**. Изначально существует единственный пользователь, которому предоставляется право входа - **root**, которая не имеет пароля. Первое, что нужно сделать войти под именем **root** и зарегистрировать нового пользователя и установить для него пароль. Команда **mysql** может использовать следующие опции:

Таблица 2 - Опции команды MySQL

-?, --help	Справка
-h,--hostname=[hostname]	Имя сервера mysql.
-u, --user=[user]	Имя пользователя для доступа к mysql.
-p, --password=[password]	Пароль пользователя для доступа к mysql.
-P, --port=[port]	Порт для соединения с сервером.
-V, --version	Информация о версии

Примечание. Команды mysqld и mysql имеют еще некоторые опции, но в данный момент они особого интереса не представляют.

Запуск из сеанса ДОС осуществляется как показано на Рисунок 8 (в указанном случае осуществляется подключение к БД mysql).

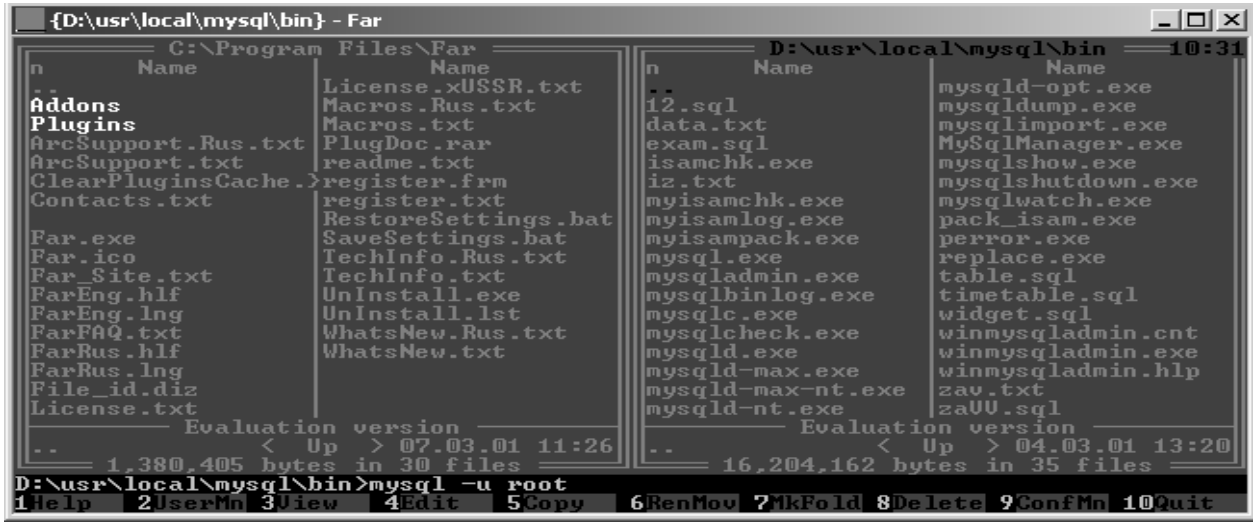


Рисунок 8 - Запуск консоли MYSQL

Для выполнения в строке наберите команду: **mysql –u root**

```
The FAR manager, version 1.70 beta 3 (build 591)
Copyright (C) 1996-2000 Eugene Roshal, Copyright (C) 2000-2001 FAR Group
Evaluation copy, please register.
D:\usr\local\mysql\bin>mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3 to server version: 3.23.41-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> _
```

Рисунок 9 - Успешный запуск консоли

Если вы это получили, значит вы успешно вошли в консоль mysql, которая используется для администрирования сервера.

Для составления отчета вам понадобятся приведение команд, которые вы будете посылать на сервер. В MySQL имеется возможность ведение протокола выполняемых команд, чтобы запустить ведение протокола необходимо выполнить команду

\T filename

!!! обязательно в верхнем регистре. Filename – имя файла, в который будут записываться команды (создается автоматически при выполнении команды, и действует во время жизни сеанса, т.е. в случае отключения от сервера лог прерывается и для возобновления необходимо повторить команду с выводом в новый файл, так как команда затирает имеющиеся в файле данные).

Просмотр списка БД, доступных на сервере осуществляется командой ***SHOW DATABASES.***

Для выполнения в строке наберите команду: **show databases.**

Командой: **USE MYSQL;** – выбираем текущую БД где MYSQL имя БД.

Система привилегий и безопасность в MySQL

— User

— Db

— Host

— Пользовательские привилегии

База данных mysql и таблицы привилегий.

Итак, вы успешно вошли в базу данных mysql, которая используется для администрирования сервера. Что же здесь находится? А находятся здесь 5 таблиц, которые ничем не отличаются от других таблиц баз данных, за исключением того, что эти таблицы используются для предоставления доступа к базам данных и таблицам в них пользователям. Рассмотрим каждую из них.

Введите следующую команду, **show tables,** которая покажет таблицы в базе данных mysql.

Кратко рассмотрим функции каждой из таблиц:

Таблица User

Определяет, разрешено ли пользователю, пытающемуся подключиться к серверу делать это. Содержит имя пользователя, пароль а также привилегии. Если ввести команду show columns from user; то получим следующее:

Таблица 3- Структура таблицы User

Field	Type	Null	Key	Default	Extra
Host	char(60)		PRI		
User	char(16)		PRI		
Password	char(41)				
Select_priv	enum('N','Y')			N	
Insert_priv	enum('N','Y')			N	
Update_priv	enum('N','Y')			N	
Delete_priv	enum('N','Y')			N	
Create_priv	enum('N','Y')			N	
Drop_priv	enum('N','Y')			N	
Reload_priv	enum('N','Y')			N	
Shutdown_priv	enum('N','Y')			N	
Process_priv	enum('N','Y')			N	
File_priv	enum('N','Y')			N	
Grant_priv[5]	enum('N','Y')			N	
References_priv	enum('N','Y')			N	
Index_priv	enum('N','Y')			N	
Alter_priv	enum('N','Y')			N	
Show_db_priv	enum('N','Y')			N	
Super_priv	enum('N','Y')			N	
Create_tmp_table_priv	enum('N','Y')			N	
Lock_tables_priv	enum('N','Y')			N	
Execute_priv	enum('N','Y')			N	
Repl_slave_priv	enum('N','Y')			N	
Repl_client_priv	enum('N','Y')			N	
Create_view_priv	enum('N','Y')			N	
Show_view_priv	enum('N','Y')			N	
Create_routine_priv	enum('N','Y')			N	
Alter_routine_priv	enum('N','Y')			N	
Create_user_priv	enum('N','Y')			N	
Event_priv	enum('N','Y')			N	

[5]Эта и все, описанные ниже команды добавлены начиная с версии 5.12

Trigger_priv	enum('N','Y')			N	
ssl_type	enum('','ANY','X509','SPECIFIED')				
ssl_cipher	blob			NULL	
x509_issuer	blob			NULL	
x509_subject	blob			NULL	
max_questions	int(11) unsigned			0	
max_updates	int(11) unsigned			0	
max_connections	int(11) unsigned			0	
max_user_connections	int(11) unsigned			0	

Изначально эта таблица содержит пользователя root без пароля. По умолчанию root может входить с любого хоста, имеет все привилегии и доступ ко всем базам данных.Также в таблице содержится запись для пользователя '%'.

В БД MYSQL содержатся таблицы, называемых таблицами привилегий. Система привилегий будет подробно рассмотрена в следующих работах, а пока вы можете выполнить команды на добавления своего пользователя:

Для добавления нового пользователя **your_name**, можно выполнить следующие операторы языка (Insert):

Insert into user (host, user, password, ssl_cipher[6], x509_issuer, x509_subject) values ('localhost', 'your_name', password('your_pass'), '', '', '');

Выполнением команды

Select host, user, password from user;

Мы выводим перечисленные поля в виде таблицы

Host	User	Password
%	root	456g879k34df9

Если необходимо выделить все столбцы таблицы, то необходимо набрать * в качестве аргумента команды ***select***.

---

[6] ***Атрибуты ssl_cipher[6], x509_issuer, x509_subject обязательны для заполнения для версии сервера 5.12***

Чтобы изменения вступили в силу нужно перегрузить сервер, предварительно закончив текущий сеанс работы командой ***quit***.

mysqladmin -u root reload (эта команда перегружает сервер)

После установки пароля для пользователя нужно перезагрузить сервер командой mysqladmin reload, чтобы изменения вступили в силу. После этого можно попробовать войти снова:

```
Mysql/bin/mysql -u your_name -p mysql
Enter password:*******
```

Если же после этой операции вы не получите приглашение ко входу, то необходимо будет повторить вход в сервер под учетной записью **ROOT** и назначить необходимые права**.** Т.о., недостаточно добавить сведения о пользователе в системную БД, дополнительно необходимо назначить права пользователю, после чего можно начинать настраивать таблицы привилегий, вводить новых пользователей, создавать базы данных и таблицы, то есть делать все то, что называется администрированием. Назначить права можно указанием инструкцией ***INSERT*** для заполнения соответствующие привилегии (перечень привилегий см.

Таблица 3)

Mysql/bin/***mysql -u root***

И выполнить следующий запрос к БД:

Mysql>***USE MYSQL;***

Mysql>***GRANT ALL PRIVILEGES ON *.* TO 'your_name'@'localhost7' IDENTIFIED BY 'your_pass' WITH GRANT OPTION;***

Mysql>***FLUSH PRIVILEGES;***

Если пароль был случайно забыт, чтобы его задать по новой, придется стереть файлы mysql.frm mysql.MYI и mysql.MYD из папки с базами данных, затем запустить скрипт mysql_install_db и повторить все по новой. Можно воспользоваться ключом MYSQL и ввести **--skip-grant-tables,** при этом все пароли будут имеет пустое поле.

Команда имеет вид ***mysqld --skip-grant-tables.***

Пояснения:

1.Команда insert вставляет данные в таблицу, не забывайте завершать команды ';'.

2.При вводе пароля используйте функцию password(), иначе пароль работать не будет!

3.Все пароли шифруются mysql, поэтому в поле Password вы видите абракадабры. Это делается в целях безопасности.

4.Не есть хорошей практикой назначать привилегии пользователям в таблице user, так как в этом случае они являются глобальными и распространяются на все базы данных. Предоставляйте привилегии каждому пользователю к конкретной базе данных в таблице db, которая будет рассмотрена далее.

5.При задании имени хоста для входа через сеть рекомендуется явно указывать полное имя хоста, а не '%'. В приведенном выше примере пользователю mary разрешается вход на сервер со всех машин домена tomsk.ru. Можно также указывать IP-адреса машин и маски подсетей для большей безопасности.

Таблица Db

Определяет к каким базам данных каким пользователям и с каких хостов разрешен доступ. В этой таблице можно предоставлять каждому пользователю доступ к базам

[7] Для случая, если работаете на том же компьютере где запущен сервер БД

данных и назначать привилегии. Если выполнить команду ***show columns from db;*** получим следующее:

Таблица 4 - Структура таблицы Db

Field	Type	Null	Key	Default	Extra
Host	char(60)		PRI		
Db	char(32)		PRI		
User	char(16)		PRI		
Select_priv	char(1)			N	
Insert_priv	char(1)			N	
Update_priv	char(1)			N	
Delete_priv	char(1)			N	
Create_priv	char(1)			N	
Drop_priv	char(1)			N	

— По умолчанию, все привилегии установлены в 'N'. Например, предоставим юзеру mary доступ к базе данных mysql и дадим ему привилегии **select**, **insert** и **update** (описание основных команд mysql будет дано в следующих лабораторных работах, сейчас ваша цель увидеть, как работают таблицы привилегий).

— Для справки:

```
Insert into db (host, user, db, select_priv, insert_priv, update_priv)
Values (''localhost', 'your_name', mysql, 'Y', 'Y', 'Y');
```

— Привилегии, устанавливаемые в таблице db, распространяются только на базу данных library. Если же установить эти привилегии в таблице user, то они будут распространяться и на другие базы данных, даже если доступ к ним и не установлен явно.

Таблица Host

Таблица host используется для расширения диапазона доступа в таблице db. К примеру, если доступ к какой-либо базе данных должен быть предоставлен более чем одному хосту, тогда следует оставить пустой колонку host в таблице db, и внести в таблицу host необходимые имена хостов. Выполним команду

show columns from host;

Таблица 5 - Структура таблиц Host

Field	Type	Null	Key	Default	Extra
Host	char(60)		PRI		
Db	char(32)		PRI		
Select_priv	char(1)			N	
Insert_priv	char(1)			N	
Update_priv	char(1)			N	
Delete_priv	char(1)			N	
Create_priv	char(1)			N	
Drop_priv	char(1)			N	

Как видно из таблицы, здесь также можно задавать привилегии для доступа к базе данных.Они обычно редко используются без необходимости. Все привилегии доступа нужно задавать в таблице db для каждого пользователя, а в таблице host только перечислить имена хостов.Сервер читает все таблицы, проверяет имя пользователя, пароль, имя хоста, имя базы данных, привилегии.Если в таблице db привилегии select, insert установлены в 'Y', а в таблице host в 'N', то в итоге юзер все равно получит 'Y'.Чтобы не вносить путаницы, лучше назначать привилегии в таблице db.

Эти 3 таблицы являются основными.В новых версиях MySQL, начиная с 3.22 добавлены еще 2 таблицы- tables_priv и columns_priv, которые позволяют задать права доступа к определенной таблице в базе данных и даже к определенной колонке. Они работают подобно таблице db, только ссылаются на таблицы и колонки. Также, начиная с версии 3.22 можно использовать команду GRANT для предоставления доступа к базам данных, таблицам и колонкам таблиц, что избавляет от необходимости вручную модифицировать таблицы db, tables_priv и columns_priv. Команда GRANT будет подробно рассмотрена в следующих разделах.

Привилегии, предоставляемые MySQL

Таблица 6 - Привилегии пользователя[8]

Привилегия	Колонка	Где используется
select	Select_priv	таблицы
insert	Insert_priv	таблицы
Update	Update_priv	таблицы

[8] Даны для справки, для текущей версии сервера может быть существенно расширены

delete	Delete_priv	таблицы
index	Index_priv	таблицы
alter	Alter_priv	таблицы
create	Create_priv	БД, таблицы, индексы
drop	Drop_priv	БД или таблицы
grant	Grant_priv	БД или таблицы
References	References_priv	БД или таблицы
reload	Reload_priv	администрирование сервера
Shutdown	Shutdown_priv	администрирование сервера
Process	Process_priv	администрирование сервера
file	File_priv	доступ к файлам на сервере

Основные утилиты MySQL.

В состав дистрибутива MySQL входят следующие утилиты:

- mysqld
- mysql
- mysqladmin
- mysqlaccess
- mysqlshow
- mysqldump
- isamchk

Утилиты **mysqld** и **mysql** были подробно рассмотрены ранее, поэтому возвращаться к ним не будем. Кратко рассмотрим остальные.

Mysqladmin

Утилита для администрирования сервера. Может использоваться администратором, а также некоторыми пользователями, которым предоставлены определенные привилегии, например – **Reload_priv, Shutdown_priv, Process_priv** и **File_priv**. Данная команда может использоваться для создания баз данных, изменения пароля пользователя(администратор может изменить пароль любому пользователю, а рядовой пользователь – только свой собственный), перезагрузки и остановки сервера, просмотра списка процессов, запущенных на сервере. Mysqladmin поддерживает следующие команды:

Таблица 7 - Опции команды MySQLadmin

Create [database_name]	Создает базу данных
Drop [database_name]	Удаляет базу данных и все таблицы в ней
Reload	Перезагружает сервер
Shutdown	Останавливает работу сервера MySQL
Processlist	Выводит список процессов на сервере
Status	Выводит сообщение о статусе сервера

Пример использования mysqladmin для изменения пароля:

mysqladmin -u your_name password your_pass

Следует заметить, что в случае использования mysqladmin для установки пароля, не требуется использование функции password().Mysqladmin сам заботится о шифровании пароля.

Mysqlaccess

Используется для проверки привилегий пользователя для доступа к конкретной базе данных. Общий синтаксис:

mysqlaccess [host] [user] [db] опции

Полезная утилита для проверки прав доступа пользователя, если он получает сообщение Access denied, при попытке соединиться с базой данных.

Опции:

Таблица 8 - Опции команды MySQLAccess

-?, --help	Справка
-u, --user=[username]	Имя пользователя
-p, --password=[password]	Пароль пользователя
-h, --host=[hostname]	Имя хоста для проверки прав доступа
-d, --db=[dbname]	Имя базы данных для проверки прав доступа
-U, --superuser=[susername]	Имя суперпользователя(root)
-P, --spassword=[spassword]	Пароль администратора
-b, --brief	Выводит краткие сведения о таблице

Mysqlshow

Используется, чтобы показать, с какими базами данных работает сервер, какие таблицы содержит каждая БД и какие колонки есть в каждой таблице.Синтаксис:

mysqlshow [опции] [database [table [field]]]

Mysqlshow может использовать следующие параметры:

Таблица 9 - Параметры команды Mysqlshow

-?, --help	Справка
-h, --host=[hostname]	Имя сервера
-k, --key	Показать ключи для таблицы
-p, --password=[password]	Пароль пользователя
-u, --user=[username]	Имя пользователя
-p, --port=[port]	Порт для связи
-V, --version	Вывести информацию о версии

Если ввести mysqlshow без аргументов, будут показаны все базы данных, если указать имя БД, будут показаны все таблицы в ней.

Команды

mysqlshow

mysqlshow mysql

Mysqldump

Программа mysqldump используется для создания дампа содержания базы данных MySQL. Она пишет инструкции SQL в стандартный вывод. Эти инструкции SQL могут быть переназначены в файл. Можно резервировать базу данных MySQL, используя mysqldump, но при этом Вы должны убедиться, что в этот момент с базой данных не выполняется никаких других действий. А то mysqldump Вам такого нарезервирует...

Программа mysqldump поддерживает следующие параметры (Вы можете использовать короткую или подробную версию):

Таблица 10 - Опции команды MySQLdump

-#, --debug=[options]	**Вывести в протокол отладочную информацию. В общем виде 'd:t:o, filename`.**
-?, --help	Справка.
-c, --compleat-insert	Генерируйте полные инструкции insert (не исключая значений, которые соответствуют значениям столбца по умолчанию).
-h, --host=[hostname]	Соединиться с сервером hostname.
-d, --no-data	Экспорт только схемы информации (исключая данные).
-t, --no-create-info	Экспорт только данных, исключая информацию для создания таблицы. Противоположность -d.
-p, --password=[password]	Пароль пользователя, для соединения с сервером MySQL. Обратите внимание, что не должно быть пробела между -p и паролем.
-q, --quick	Не буферизовать результаты запроса, дамп выдать непосредственно к STDOUT.
-u, --user=[username]	Имя пользователя. Если не задано, используется текущий логин.
-v, --verbose	Вывести подробную информацию относительно различных стадий выполнения mysqldump.
-P, --port=[port]	Порт для связи.
-V, --version	Информация о версии.

Вы можете направить вывод mysqldump в клиентскую программу MySQL, чтобы копировать базу данных. ПРИМЕЧАНИЕ: Вы должны убедиться, что база данных не изменяется в это время, иначе Вы получите противоречивую копию!

Для справки:

mysqldump -u root -p mysql user>mysql-1.sql
mysqldump -u root mysql>mysql-2.sql

Примечание флаг –p используется в случае, если пользователь наделен паролем.

После выполнения этой команды у нас появился файл mysql-1.sql и mysql-2.sql. Загрузим их в текстовый редактор, чтобы поподробнее изучить, и, возможно, немного поправить.

Задание

Запустите сервер MySQL. Зарегистрируйте своего пользователя в консольном приложении, задайте ему права.

С помощью утилиты Mysqlshow выполните команду на просмотр структуры и состав таблиц базы Mysql. Приведите в отчете её схему. С помощью утилиты Mysqldump получите полный дамп базы Mysql (данные и таблицы), а также отдельные дампы таблиц и данных.

Контрольные вопросы:

1. Каким способом возможен запуск серверной части СУБД.
2. Что такое привилегия. Каково её предназначение.
3. Какие основные утилиты входят в состав СУБД, какие функции они выполняют.

Список литературы, рекомендуемый к использованию по данной теме:
Основная: 1-11
Дополнительная: 1-11
Интернет-ресурсы: 1-7
Программное обеспечение: 1-4

Раздел 2. Безопасность информации в базах данных

Практическое занятие №4

Тема 7 - 8. Структуры внешней памяти в реляционных СУБД. Управление параллельностью работы транзакций

1. Цель работы: приобретение студентами практических навыков создания логических и физических моделей данных с помощью CASE – средств разработки информационных систем.

Формируемые компетенции: ПК-1, ПК-3

2. Подготовка к занятию

1. Изучить (повторить) теоретический материал.
2. Ознакомиться с программой работы.
3. Подготовить отчет о работе.
4. Ответить на контрольные вопросы.

3. Распределение времени занятия:

Всего: 90 мин

Вступительная часть 2 мин

Проверка готовности студентов к занятию 5 мин

Основная часть – 70 мин

Проверка выполнения практического занятия 10 мин

Заключительная часть 3 мин

4. Правила работы в лаборатории

К работе в лаборатории допускаются лица, изучившие правила и меры безопасности, сдавшие зачет по ним и усвоившие порядок выполнения лабораторной работы.

4.1. Требования безопасности перед началом работ

- ЗАПРЕЩАЕТСЯ: переодеваться, пользоваться огнем, курить, принимать пищу в лаборатории.

- Убедиться в целостности электрических розеток и разъемов. В лаборатории необходимо быть в сменной обуви.

- Включение компьютера производить только после получения допуска по выполняемой работе и разрешения преподавателя или лаборанта.

4.2. Требования безопасности во время работы

- выполняя практическое занятие, студенты обязаны использовать только вычислительную технику, периферийное оборудование, соединительные кабели, измерительное оборудование и носители информации, непосредственно относящиеся к данному лабораторному занятию;

- подключение и отключение составляющих вычислительного комплекса производить только при полном снятии напряжения со всех составляющих вычислительного комплекса;

- при обнаружении неисправностей в оборудовании немедленно отключить источники питания и доложить об этом руководителю занятий или лаборанту.

4.3. Требования безопасности по окончании работы

- доложить руководителю занятий или лаборанту о завершении работ;
- привести в порядок и сдать рабочее место лаборанту, и доложить руководств

Основные сведения

Система ERwin поддерживает прямое и обратное моделирование баз данных. При прямом моделировании схема базы данных описывается в прямом виде с использованием диаграммы сущность-связь. Сущности на диаграмме представляются прямоугольниками. Каждый прямоугольник может иметь различные визуальные атрибуты. Каждой сущности должно быть присвоено уникальное имя. Имена сущностей необходимо задавать в единственном числе. Это определяется тем, что система всегда оперирует отдельными экземплярами сущности. При этом отдельные экземпляры сущности рассматриваются как объекты, а сущности – как класс объектов. Если сущности были описаны при моделировании в BPwin, то их можно просто импортировать в ERwin. Пример диаграммы с созданными сущностями приведен на рисунке.

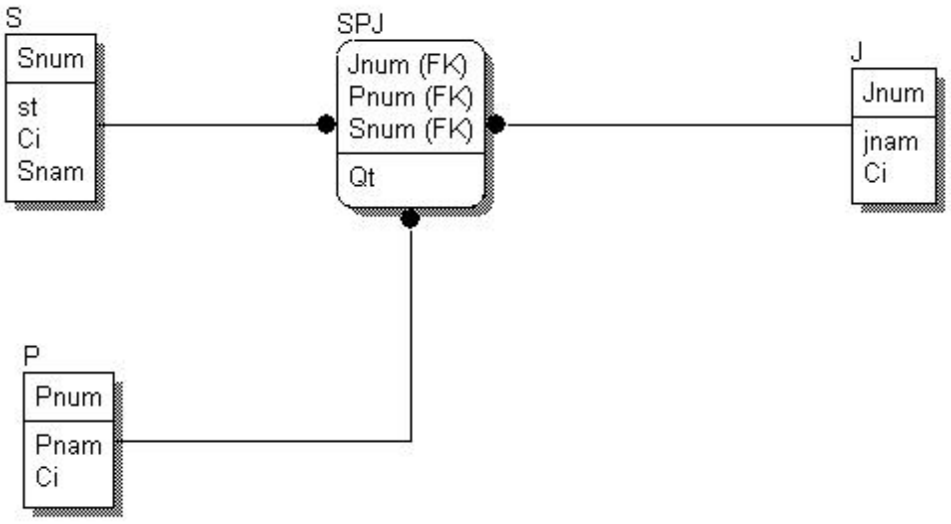


Рисунок 10 - Пример диаграммы с созданными сущностями

Построение моделей в ERwin

Возможны две точки зрения на информационную модель и, соответственно, два уровня модели. Первый - логический уровень (точка зрения пользователя) означает прямое отображение фактов из реальной жизни. Например, люди, столы, отделы, собаки и компьютеры являются реальными объектами. Они именуются на естественном языке, с

любыми разделителями слов (пробелы, запятые и т.д.). На физическом уровне модели рассматривается использование конкретной СУБД, определяются типы данных (например, целое или вещественное число), индексы для таблиц.

ERwin предоставляет возможности создавать и управлять этими двумя различными уровнями представления одной диаграммы (модели), равно как и иметь много вариантов отображения на каждом уровне. Термин "логический уровень" в ERwin соответствует концептуальной модели.

Этапы построения информационной модели.

- определение сущностей;
- определение зависимостей между сущностями;
- задание первичных и альтернативных ключей;
- определение атрибутов сущностей;
- приведение модели к требуемому уровню нормальной формы;
- переход к физическому описанию модели: назначение соответствий имя сущности - имя таблицы, атрибут сущности - атрибут таблицы;
- задание триггеров, процедур и ограничений;
- генерация базы данных.

Erwin создает визуальное представление (модель данных) для решаемой задачи. Это представление может использоваться для детального анализа, уточнения и распространения документации, необходимой в цикле разработки. Однако ERwin далеко не только инструмент для рисования. ERwin автоматически создает базу данных (таблицы, индексы, хранимые процедуры, триггеры для обеспечения ссылочной целостности и другие объекты, необходимые для управления данными).

Создание сущности.

Для внесения сущности в модель необходимо щелкнуть по кнопке сущности на панели инструментов (Erwin Toolbox)  , затем - по тому месту на диаграмме, где необходимо расположить новую сущность. Щелкнув правой кнопкой мыши по сущности и выбрав из всплывающего меню пункт Entity Editor, можно вызвать диалог Entity Editor, в котором определяются имя, описание и комментарии сущности.

Каждая сущность должна быть полностью определена с помощью текстового описания в закладке Definition. Эти определения полезны как на логическом уровне,

поскольку позволяют понять, что это за объект, так и на физическом уровне, поскольку их можно экспортировать как часть схемы и использовать в реальной БД (**CREATE COMMENT on entity_name**). Закладки Note, Note2, Note3, UDP (User Defined Properties - Свойства, определенные пользователем) служат для внесения дополнительных комментариев и определений к сущности.

В закладке Icon каждой сущности можно поставить в соответствие изображение, которое будет отображаться в режиме просмотра модели на уровне иконок и изображение, которое будет отображаться на всех других уровнях.

Закладка UDP диалога Entity Editor служит для определения свойств, определяемых пользователем (User - Defined Properties). При нажатии на кнопку  этой закладки вызывается диалог User - Defined Property Editor (также вызывается из меню Edit/UDPs). В нем необходимо указать вид объекта, для которого заводится UDP (диаграмма в целом, сущность, атрибут и т.д.) и тип данных. Для внесения нового свойства следует щелкнуть в таблице по кнопке  и внести имя, тип данных, значение по умолчанию и определение.

Создание атрибутов.

Следующий этап создания модели состоит в задании атрибутов для каждой сущности. При задании типа атрибута имеется возможность использовать домены. Домен – это абстрактный пользовательский тип, который присваивается любому физическому типу данных. При этом каждый домен может иметь свои значения по умолчанию и правила проверки вводимых данных. ERwin предоставляет возможность документировать все действия по созданию собственных типов данных. С использованием концепции домена обеспечивается переносимость базы данных на различные аппаратные платформы.

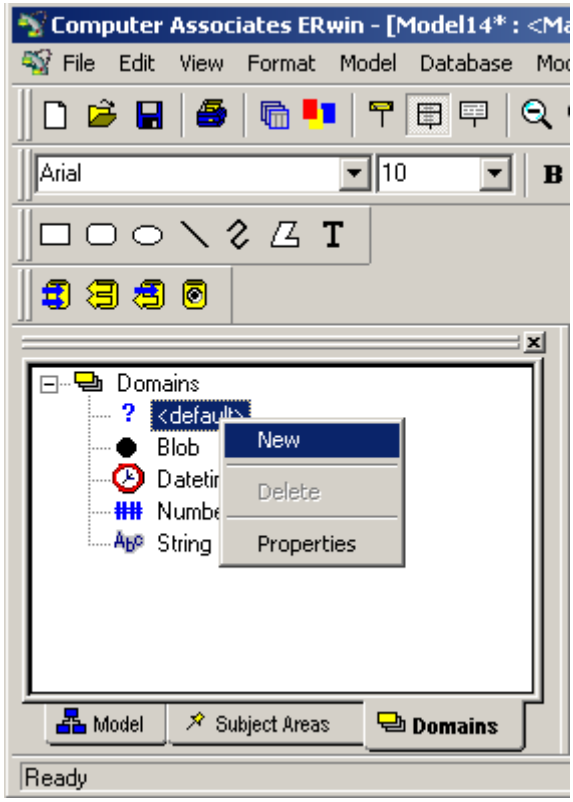


Рисунок 11 - Создание нового домена

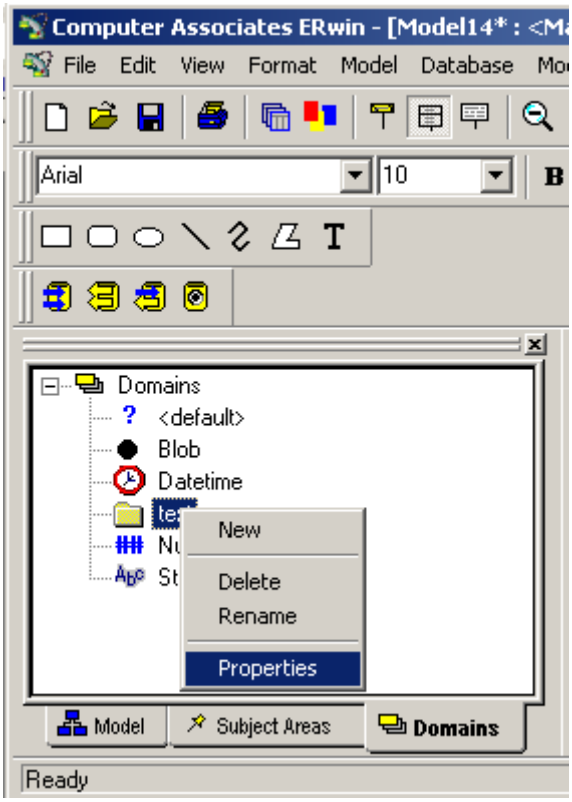


Рисунок 12 - Указание свойств нового домена

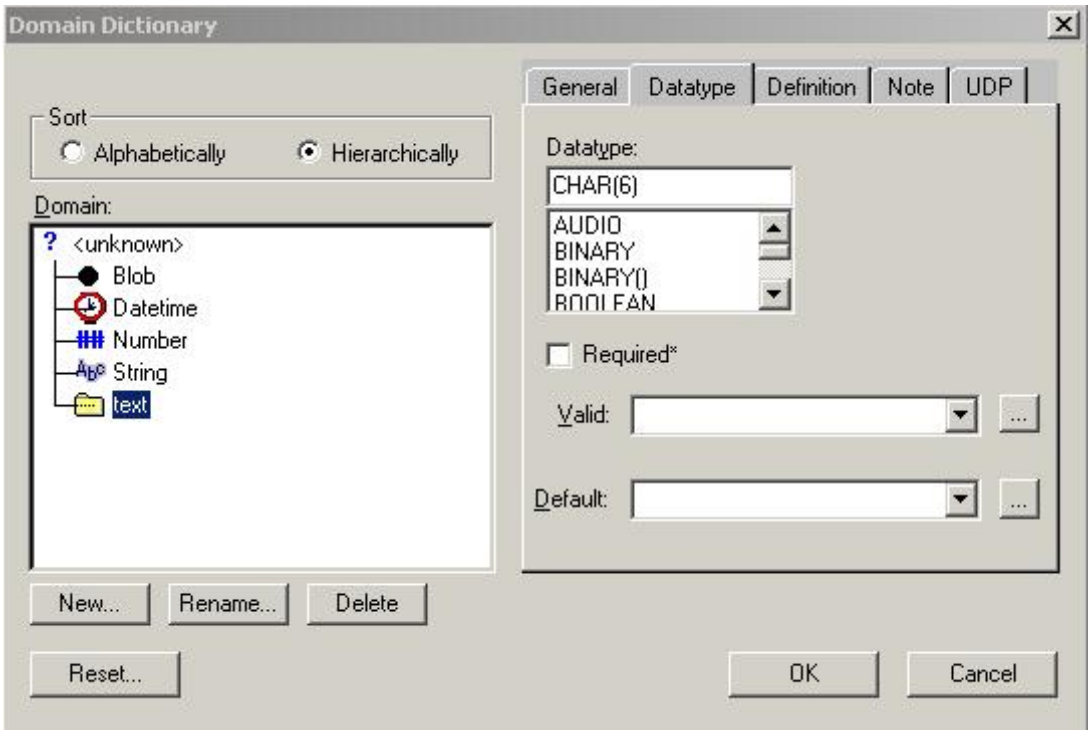


Рисунок 13 - Значение по умолчанию для нового домена

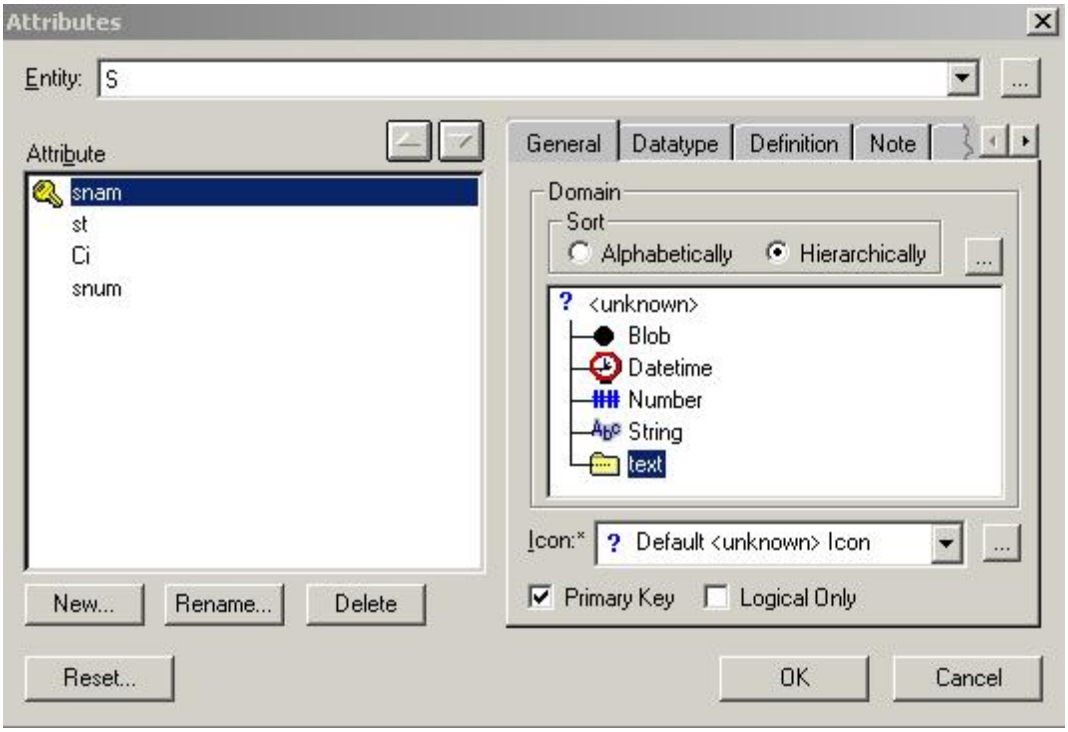


Рисунок 14 - Использование домена для указания типа данных атрибуту.[9]

Для описания атрибутов следует, щелкнув правой кнопкой по сущности, выбрать в появившемся меню пункт Attribute Editor. Появится диалог Attribute Editor.

Если щелкнуть по кнопке New, то в появившемся диалоге New Attribute можно указать имя атрибута, имя соответствующей ему в физической модели колонки и домен. Домен атрибута будет использоваться при определении типа колонки на уровне физической модели.

Для атрибутов первичного ключа в закладке General диалога Attribute Editor необходимо сделать пометку в окне выбора Primary Key. Закладки Definition, Note и UDP несут те же функции, что и при определении сущности, но на уровне атрибутов.

Для большей наглядности диаграммы каждый атрибут можно связать с иконкой. Это можно сделать при помощи списка выбора Icon в закладке General.

Очень важно дать атрибуту правильное имя. Атрибуты должны именоваться в единственном числе и иметь четкое смысловое значение.

Согласно синтаксису IDEF1X, имя атрибута должно быть уникальным в рамках модели (а не только в рамках сущности!). По умолчанию при попытке внесения уже существующего имени атрибута ERwin переименовывает его. Например, если атрибут Комментарий уже существует в модели, другой атрибут (в другой сущности) будет назван Комментарий/2, затем Комментарий/3 и т.д.

[9] Перечень типов данных, поддерживаемых СУБД необходимо уточнить у производителя

При переносе атрибутов внутри и между сущностями можно воспользоваться техникой drag&drop, выбрав кнопку  в палитре инструментов.

Для создания новой связи следует выбрать идентифицирующую или неидентифицирующую связь в палитре инструментов (ERwin Toolbox), щелкнуть сначала по родительской, а затем по дочерней сущности. В палитре инструментов кнопка  соответствует идентифицирующей связи, кнопка  связи многие-ко-многим и кнопка  соответствует неидентифицирующей связи. Для редактирования свойств связи следует щелкнуть правой кнопкой мыши по связи и выбрать на контекстном меню пункт Relationship Editor.

В закладке General появившегося диалога можно задать мощность, имя и тип связи.

Связи на диаграмме представляются линиями, идущими от одной сущности (таблицы) к другой. Каждой связи присваивается уникальное имя. Связанные таблицы разделяют на родительские и дочерние. Родительские таблицы отображаются прямоугольниками с прямыми углами, дочерние – со скругленными.

Мощность связи (Cardinality) - служит для обозначения отношения числа экземпляров родительской сущности к числу экземпляров дочерней. Различают четыре типа мощности:

- общий случай, когда одному экземпляру родительской сущности соответствуют 0, 1 или много экземпляров дочерней сущности, не помечается каким-либо символом;
- символом P помечается случай, когда одному экземпляру родительской сущности соответствуют 1 или много экземпляров дочерней сущности (исключено нулевое значение);
- символом Z помечается случай, когда одному экземпляру родительской сущности соответствуют 0 или 1 экземпляр дочерней сущности (исключены множественные значения);
- цифрой помечается случай, когда одному экземпляру родительской сущности соответствует заранее заданное число экземпляров дочерней сущности.

По умолчанию символ, обозначающий мощность связи, не показывается на диаграмме. Для отображения имени следует в контекстном меню, которое появляется, если щелкнуть правой кнопкой мыши по любому месту диаграммы, не занятому объектами модели, выбрать пункт Display Options/Relationship и затем включить опцию Cardinality.

Тип связи (идентифицирующая/неидентифицирующая).

В IDEF1X различают зависимые и независимые сущности. Тип сущности определяется ее связью с другими сущностями. Идентифицирующая связь устанавливается между независимой (родительский конец связи) и зависимой (дочерний конец связи) сущностями. Когда рисуется идентифицирующая связь, ERwin автоматически преобразует дочернюю связь в зависимую. Зависимая сущность изображается прямоугольником со скругленными углами.

Экземпляр зависимой сущности определяется только через отношение к родительской сущности. При установлении идентифицирующей связи атрибуты первичного ключа родительской сущности автоматически переносятся в состав первичного ключа дочерней сущности. Эта операция дополнения атрибутов дочерней сущности при создании связи называется миграцией атрибутов. В дочерней сущности новые атрибуты помечаются как внешние ключи - (FK).

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

Идентифицирующая связь показывается на диаграмме сплошной линией с жирной точкой на дочернем конце связи, неидентифицирующая - пунктирной.

Для неидентифицирующей связи можно указать обязательность (Nulls в закладке General диалога Relationship Editor). В случае обязательной связи (No Nulls) при генерации схемы БД атрибут внешнего ключа получит признак NOT NULL, несмотря на то, что внешний ключ не войдет в состав первичного ключа дочерней сущности. В случае необязательной связи (Nulls Allowed) внешний ключ может принимать значение NULL. Необязательная неидентифицирующая связь помечается прозрачным ромбом со стороны родительской сущности

Имя связи (Verb Phrase) - фраза, характеризующая отношение между родительской и дочерней сущностями. Для связи один-ко-многим идентифицирующей или неидентифицирующей достаточно указать имя, характеризующей отношение от родительской к дочерней сущности (Parent-to-Child). Для связи многие-ко-многим следует указывать имена как Parent-to-Child, так и Child-to-Parent. Для отображения имени следует в контекстном меню, которое появляется, если щелкнуть правой кнопкой мыши по любому месту диаграммы, не занятому объектами модели, выбрать пункт Display Options/Relationship и затем включить опцию Verb Phrase.

Имя роли или функциональное имя (Rolename) - это синоним атрибута внешнего ключа, который показывает, какую роль играет атрибут в дочерней сущности. Задать имя роли можно в закладке Rolename/RI Actions диалога Relationship Editor.

Создание ключей.

Каждый экземпляр сущности должен быть уникален и отличаться от других атрибутов.

Первичный ключ (primary key) - это атрибут или группа атрибутов, однозначно идентифицирующие экземпляр сущности. Атрибуты первичного ключа на диаграмме не требуют специального обозначения - это те атрибуты, которые находятся в списке атрибутов выше горизонтальной линии. При внесении нового атрибута в диалоге Attribute Editor для того, чтобы сделать его атрибутом первичного ключа, нужно включить флажок Primary Key в нижней части закладки General. На диаграмме ключевой атрибут можно внести в состав первичного ключа, воспользовавшись режимом переноса атрибутов (кнопка  в палитре инструментов).

В одной сущности может оказаться несколько атрибутов или наборов атрибутов, претендующих на роль первичного ключа. Такие претенденты называются **потенциальными ключами (candidate key).**

Ключи могут быть сложными, т.е. содержащими несколько атрибутов. Сложные первичные ключи не требуют специального обозначения - это список атрибутов выше горизонтальной линии. При выборе первичного ключа предпочтение должно отдаваться более простым ключам, т.е. ключам, содержащим меньшее количество атрибутов.

Многие сущности имеют только один потенциальный ключ. Такой ключ становится первичным. Некоторые сущности могут иметь более одного возможного ключа. Тогда один из них становится первичным, а остальные - альтернативными ключами.

Альтернативный ключ (Alternative Key) - это потенциальный ключ, не ставший первичным.

Каждому ключу соответствует индекс, имя которого также присваивается автоматически. Имена ключа и индекса при желании можно изменить вручную.

На диаграмме атрибуты альтернативных ключей обозначаются как (Akn.m.), где n - порядковый номер ключа, m - порядковый номер атрибута в ключе. Когда альтернативный ключ содержит несколько атрибутов, (Akn.m.) ставится после каждого.

Внешние ключи (Foreign Key) создаются автоматически, когда связь соединяет сущности: связи образуют ссылку на атрибуты первичного ключа в дочерней сущности и эти

атрибуты образуют внешний ключ в дочерней сущности (миграция ключа). Атрибуты внешнего ключа обозначаются символом (FK) после своего имени.

Зависимая сущность может иметь один и тот же ключ из нескольких родительских сущностей. Сущность может также получить один и тот же внешний ключ несколько раз от одного и того же родителя через несколько разных связей. Когда ERwin обнаруживает одно из этих событий, он распознает, что два атрибута одинаковы, и помещает атрибуты внешнего ключа в зависимой сущности только один раз. Это комбинирование или объединение идентичных атрибутов называется унификацией.

Есть случаи, когда унификация нежелательна. Например, когда два атрибута имеют одинаковые имена, но на самом деле они отличаются по смыслу, и необходимо, что бы это отличие отражалось в диаграмме. В этом случае необходимо использовать имена ролей внешнего ключа.

Связи на диаграмме представляются линиями, идущими от одной сущности (таблицы) к другой. Каждой связи присваивается уникальное имя. Связанные таблицы разделяют на родительские и дочерние. Родительские таблицы отображаются прямоугольниками с прямыми углами, дочерние – со скругленными.

После указания всем атрибутам формата данных необходимо созданную логическую модель преобразовать в физическую. Для этого необходимо в **Tools** выбрать **Derive New Model**, где в качестве Target Databases выберите **ODBC/Generic** (для использования в СУБД MySQL) см. Рисунок 15. Наша модель (см Рисунок 10) будет преобразована к виду см.Рисунок 17.

Далее выбрав в меню **Tools/Forward Engineer/Shema Generation** и задав необходимые настройки, получим в меню Preview код на языке SQL для реализации схемы БД в СУБД MySQL.

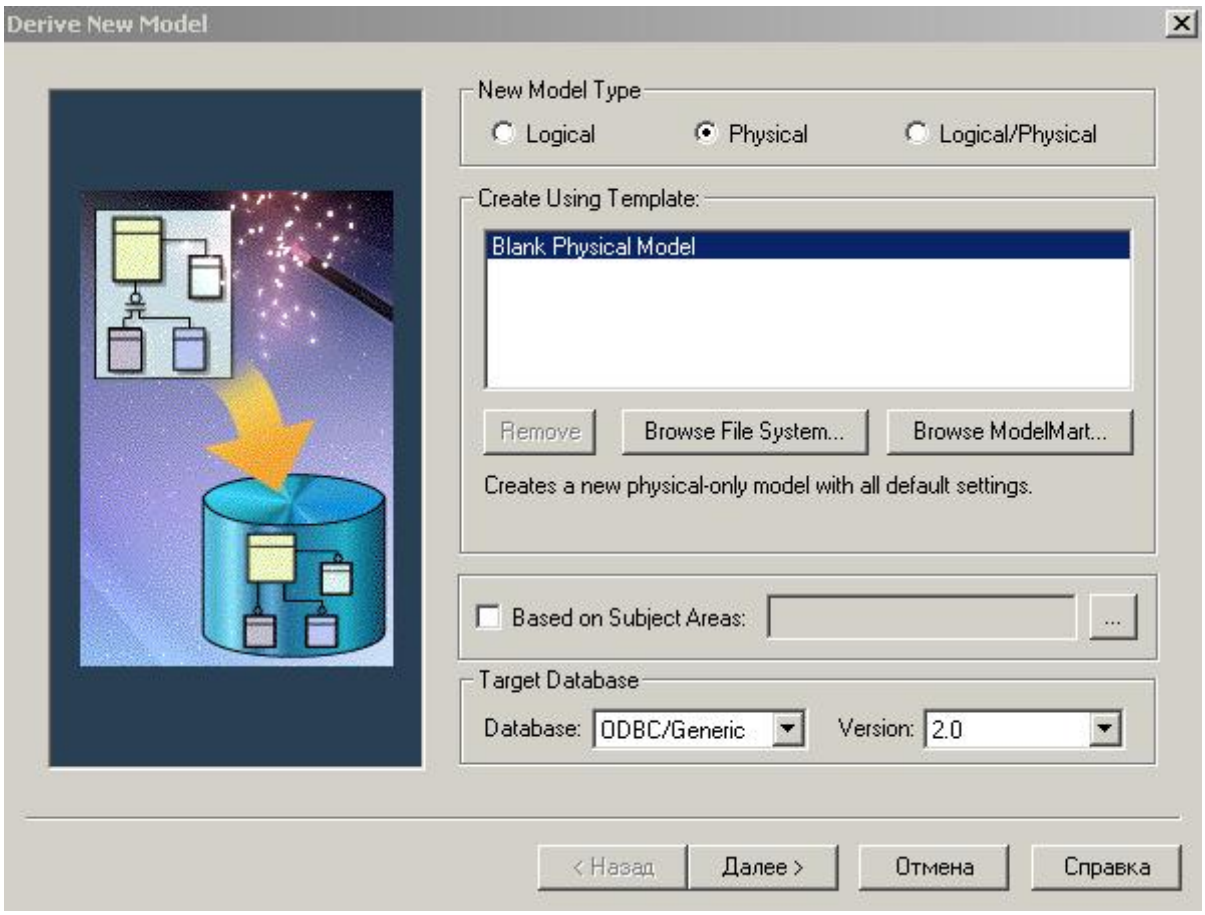


Рисунок 15 - Преобразование логической модели в физическую

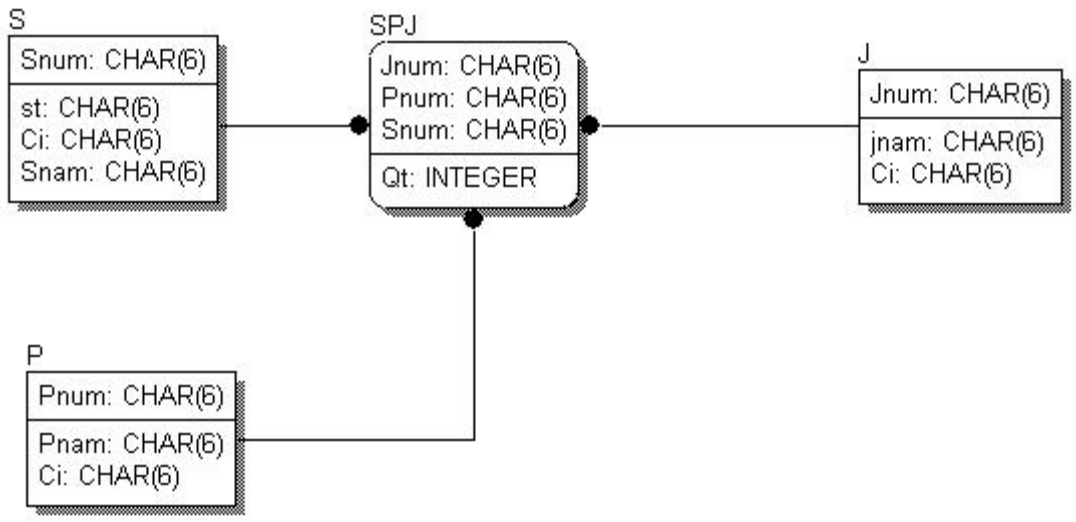


Рисунок 16 - Физическая модель с указанием формата данных.

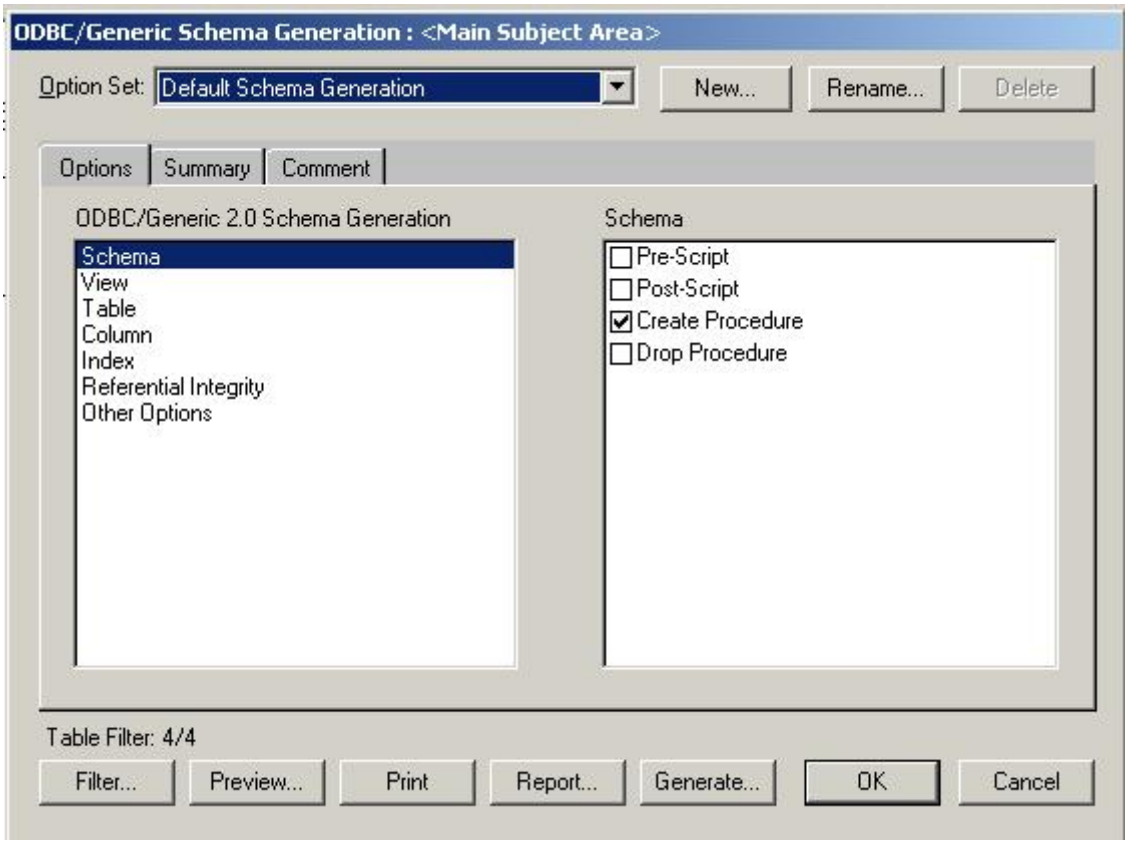


Рисунок 17 - Генерация кода SQL

Задание

1. Выполните построение диаграммы с заданными сущностями (прямое моделирование) для заданной предметной области.
2. Задайте атрибуты для каждой определенной сущности. При задании атрибутов используйте домены.
3. Введите связи между сущностями. Присвойте связям уникальные имена.
4. Используя СУБД MYSQL, решите прямую генерацию базы данных для проектируемой информационной.
5. Отчет должен содержать концептуальную модель и физическую базу данных в СУБД MYSQL.

Контрольные вопросы

1. В чем состоит различие логического и физического уровней представления моделей данных с помощью ERwin?
2. В чем различие между моделями данных, представленных в форме диаграммы сущность-связь, на основе ключей и в виде полной атрибутивной модели?

3. Какие основные компоненты содержат модели данных, представленные по методологии IDEF1X?

Список литературы, рекомендуемый к использованию по данной теме:

Основная: 1-11

Дополнительная: 1-11

Интернет-ресурсы: 1-7

Программное обеспечение: 1-4

Практическое занятие№5

Тема 9 - 10. Методы сериализации транзакция. Журнализация изменений БД

1. Цель работы: используя данные базы данных, подготовленной в предыдущей лабораторной работе, подготовить и реализовать серию запросов, связанных с выборкой информации и модификацией данных таблиц.

Формируемые компетенции: ПК-1, ПК-3

2. Подготовка к занятию

1. Изучить (повторить) теоретический материал.
2. Ознакомиться с программой работы.
3. Подготовить отчет о работе.
4. Ответить на контрольные вопросы.

3. Распределение времени занятия:

Всего: 90 мин

Вступительная часть 2 мин

Проверка готовности студентов к занятию 5 мин

Основная часть – 70 мин

Проверка выполнения практического занятия 10 мин

Заключительная часть 3 мин

4. Правила работы в лаборатории

К работе в лаборатории допускаются лица, изучившие правила и меры безопасности, сдавшие зачет по ним и усвоившие порядок выполнения лабораторной работы.

4.1. Требования безопасности перед началом работ

- ЗАПРЕЩАЕТСЯ: переодеваться, пользоваться огнем, курить, принимать пищу в лаборатории.

- Убедиться в целостности электрических розеток и разъемов. В лаборатории необходимо быть в сменной обуви.

- Включение компьютера производить только после получения допуска по выполняемой работе и разрешения преподавателя или лаборанта.

4.2. Требования безопасности во время работы

- выполняя практическое занятие, студенты обязаны использовать только вычислительную технику, периферийное оборудование, соединительные кабели, измерительное оборудование и носители информации, непосредственно относящиеся к данному лабораторному занятию;

- подключение и отключение составляющих вычислительного комплекса производить только при полном снятии напряжения со всех составляющих вычислительного комплекса;

- при обнаружении неисправностей в оборудовании немедленно отключить источники питания и доложить об этом руководителю занятий или лаборанту.

4.3. Требования безопасности по окончании работы

- доложить руководителю занятий или лаборанту о завершении работ;
- привести в порядок и сдать рабочее место лаборанту, и доложить руководству

Содержание работы и методические указания к ее выполнению

1. Изучить набор команд языка SQL, связанный с созданием запросов, добавлением, модификацией и удалением строк таблицы:

select - осуществление запроса по выборке информации из таблиц базы данных;

insert - добавление одной или нескольких строк в таблицу;

delete - удаление одной или нескольких строк из таблицы;

update - модификация одной или нескольких строк таблицы;

union - объединение запросов в один запрос.

2. Изучить состав, правила и порядок использования ключевых фраз оператора select:

select - описание состава данных, которые следует выбрать по запросу (обязательная фраза);

from - описание таблиц, из которых следует выбирать данные (обязательная фраза);

where - описание условий поиска и соединения данных при запросе;

group by - создание одной строки результата для каждой группы (группой называется множество строк, имеющих одинаковые значения в указанных столбцах);

having - наложение одного или более условий на группу;

order by - сортировка результата выполнения запроса по одному или нескольким столбцам;

into outfile - создание файла, в который будет осуществлен вывод результатов соответствующего запроса.

Порядок следования фраз в команде select должен соответствовать приведенной выше последовательности. Для лучшего понимания механизма функционирования выполните следующие упражнения:

I. Простые запросы на языке SQL

Запрос на языке SQL формируется с использованием оператора Select. Оператор Select используется

- для выборки данных из базы данных;
- для получения новых строк в составе оператора Insert;
- для обновления информации в составе оператора Update.

В общем случае оператор Select содержит следующие семь спецификаторов, расположенных в операторе в следующем порядке:

- спецификатор Select;
- спецификатор From;
- спецификатор Where;
- спецификатор Group by;
- спецификатор Having;
- спецификатор Order by;

Обязательными являются только спецификаторы Select и From. Эти два спецификатора составляют основу каждого запроса к базе данных, поскольку они определяют таблицы, из которых выбираются данные, и столбцы, которые требуется выбрать.

Спецификатор Where добавляется для выборки определенных строк или указания условия соединения. Спецификатор Order by добавляется для изменения порядка получаемых данных. Спецификатор Into temp добавляется для сохранения этих результатов в виде таблицы с целью выполнения последующих запросов. Два дополнительных спецификатора оператора Select - Group by (спецификатор группирования) и Having (спецификатор условия выборки группы) - позволяют выполнять более сложные выборки данных.

У п р а ж н е н и я

1. Выбор всех строк и столбцов таблицы.

Пример.

Выдать полную информацию о поставщиках.

Select * from S

Результат: таблица S в полном объеме.

Подготовьте запрос и проверьте полученный результат.

2. Измение порядка следования столбцов.

Пример.

Выдать таблицу S в следующем порядке: фамилия, город, рейтинг, номер_поставщика.

Select фамилия, город, рейтинг, номер_поставщика from S

Результат: таблица S в требуемом порядке.

Подготовьте запрос и проверьте полученный результат.

3. Выбор заданных столбцов.

Пример.

Выдать номера всех поставляемых деталей.

Select номер_детали from SPJ

Результат: столбец номер_детали таблицы SPJ

Подготовьте запрос и проверьте полученный результат.

4. Выбор без повторения.

Пример.

Выдать номера всех поставляемых деталей, исключая дублирование.

Select distinct номер_детали from SPJ

Результат:	**номер_детали**
	P1
	P2
	P3
	P4
	P5
	P6

Подготовьте запрос и проверьте полученный результат.

5. Использование в запросах констант и выражений.

Пример.

Select номер_детали, "вес в граммах", вес*454 from P

Результат:	P1 вес в граммах=5448

	P6 вес в граммах=8226

Подготовьте запрос и проверьте полученный результат.

6.Ограничение в выборке.

Пример.

Выдать номера всех поставщиков, находящихся в Париже с рейтингом > 20.

Select номер_поставщика from S where город="Париж" and рейтинг>20

Результат:	номер_поставщика
	S3

Подготовьте запрос и проверьте полученный результат.

7. Выборка с упорядочиванием.

Пример.

Выдать номера поставщиков, находящихся в Париже в порядке убывания рейтинга.

Select номер_поставщика, рейтинг from S where город="Париж" order by рейтинг desc

Результат:	номер_поставщика	рейтинг
	S3	30
	S2	10

Подготовьте запрос и проверьте полученный результат.

8. Упорядочивание по нескольким столбцам.

Пример.

Выдать список поставщиков, упорядоченных по городу, в пределах города - по рейтингу.

Select * from S order by 4, 3

Результат:	Номер_поставщика	Фамилия	Рейтинг	Город
	S5	Адамс	30	Атенс
	S1	Смит	20	Лондон
	S4	Кларк	20	Лондон
	S2	Джонс	10	Париж
	S3	Блейк	30	Париж

Подготовьте запрос и проверьте полученный результат.

9. Фраза in (not in).

Пример.

Выдать детали, вес которых равен 12, 16 или 17.

Select номер_детали, название, вес from P where вес in (12, 16, 17)

Результат:	номер_детали	Название	вес
	P1	Гайка	12
	P2	Болт	17
	P3	Винт	17
	P5	Кулачок	12

Подготовьте запрос и проверьте полученный результат.

12. Выбор по шаблону.

Для запросов с поиском по шаблону, основанных на поиске подстрок в полях типа CHARACTER, используются ключевое слова LIKE.

Включение в выражение ключевого слова NOT порождает условие с обратным смыслом. Ключевое слово LIKE соответствует стандарту ANSI.

СИМВОЛ	**ЗНАЧЕНИЕ**
LIKE	
%	Заменяет последовательность символов
-	Заменяет любой одиночный символ
\	Отменяет специальное назначение следующего за ним символа

Примеры.

а) Выбрать список деталей, начинающихся с буквы "Б"[10]

[10] Примечание. Корректно работает только при задании кодировки по умолчанию. Задается в разделе MYSQLD default_character_set=win1251

Select номер_детали, название, вес from P where название like "Б%"

Результат:	номер_детали	название	вес
	P5	Болт	12
	P6	Блюм	19

II. Использование функций

1. Агрегатные функции.

Примеры.

а) Выдать общее количество поставщиков.

Select count (*) from S

Результат: 5

Подготовьте запрос и проверьте полученный результат.

б) Выдать общее количество поставщиков, поставляющих в настоящее время детали.

Select count (distinct номер_поставщика) from SPJ

Результат: 4

Подготовьте запрос и проверьте полученный результат.

в) Выдать количество поставок для детали P2.

Select count (*) from SPJ where номер_детали='P2'

Результат: 5

Подготовьте запрос и проверьте полученный результат.

г) Выдать общее количество поставляемых деталей 'P2'.

Select sum (количество) from SPJ where номер_детали='P2'

Результат: 1000

Подготовьте запрос и проверьте полученный результат.

д) Выдать средний, минимальный и максимальный объем поставок для поставщика S1 с соответствующим заголовком.

Select avg(количество) average, min(количество) minimum,

max(количество) maximum from SPJ where номер_поставщика='S1'

Результат:	average	minimum	maximum
	216.6	100	400

Подготовьте запрос и проверьте полученный результат.

2. Ниже привиден перечень всех функций, используемых в операторе Select

Функции

select_expression может содержать следующие функции и операторы:

+ - * /	Арифметические действия.
%	Остаток от деления (как в C)
| &	Битовые функции (используется 48 бит).
- C	Мена знака числа.
( )	Скобки.
BETWEEN(A, B, C)	(A >= B) AND (A <= C).
BIT_COUNT()	Количество бит.
ELT(N, a, b, c, d)	Возвращает a, если N == 1, b, если N == 2 и т. д. a, b, c, d строки. **ПРИМЕР:** ELT(3, "First", "Second", "Third", "Fourth") вернет "Third".
FIELD(Z, a, b, c)	Возвращает a, если Z == a, b, если Z == b и т. д. a, b, c, d строки. **ПРИМЕР:** FIELD("Second", "First", "Second", "Third", "Fourth") вернет "Second".
IF(A, B, C)	Если A истина (!= 0 and != NULL), то вернет B, иначе вернет C.
IFNULL(A, B)	Если A не null, вернет A, иначе вернет B.
ISNULL(A)	Вернет 1, если A == NULL, иначе вернет 0. Эквивалент ('A == NULL').
NOT !	NOT, вернет TRUE (1) или FALSE (0).
OR, AND	Вернет TRUE (1) или FALSE (0).

SIGN()	Вернет -1, 0 или 1 (знак аргумента).
SUM()	Сумма столбца.
= <> <= < >= >	Вернет TRUE (1) или FALSE (0).
expr LIKE expr	Вернет TRUE (1) или FALSE (0).
expr NOT LIKE expr	Вернет TRUE (1) или FALSE (0).
expr REGEXP expr	Проверяет строку на соответствие регулярному выражению expr.

select_expression может также содержать один или большее количество следующих математических функций.

ABS()	Абсолютное значение (модуль числа).
CEILING()	()
EXP()	Экспонента.
FORMAT(nr, NUM)	Форматирует число в формат '#, ###, ###.##' с NUM десятичных цифр.
LOG()	Логарифм.
LOG10()	Логарифм по основанию 10.
MIN(), MAX()	Минимум или максимум соответственно. Должна иметь при вызове два или более аргументов, иначе рассматривается как групповая функция.
MOD()	Остаток от деления (аналог %).
POW()	Степень.
ROUND()	Округление до ближайшего целого числа.
RAND([integer_expr])	Случайное число типа float, 0 <= x <= 1.0, используется integer_expr как значение для запуска генератора.
SQRT()	Квадратный корень.

select_expression может также содержать одну или больше следующих строковых функций.

CONCAT()	Объединение строк.
INTERVAL(A, a, b, c, d)	Возвращает 1, если A == a, 2, если A == b... Если совпадений нет, вернет 0. A, a, b, c, d... строки.
INSERT(org, strt, len, new)	Заменяет подстроку org[strt...len(gth)] на new. Первая позиция строки=1.
LCASE(A)	Приводит A к нижнему регистру.
LEFT()	Возвращает строку символов, отсчитывая слева.

LENGTH()	Длина строки.
LOCATE(A, B)	Позиция подстроки B в строке A.
LOCATE(A, B, C)	Позиция подстроки B в строке A, начиная с позиции C.
LTRIM(str)	Удаляет все начальные пробелы из строки str.
REPLACE(A, B, C)	Заменяет все подстроки B в строке A на подстроку C.
RIGHT()	Get string counting from right.
RTRIM(str)	Удаляет хвостовые пробелы из строки str.
STRCMP()	Возвращает 0, если строки одинаковые.
SUBSTRING(A, B, C)	Возвращает подстроку из A, с позиции B до позиции C.
UCASE(A)	Переводит A в верхний регистр.

Еще несколько просто полезных функций, которые тоже можно применить в select_expression.

CURDATE()	Текущая дата.
DATABASE()	Имя текущей базы данных из которой выполняется выбор.
FROM_DAYS()	Меняет день на DATE.
NOW()	Текущее время в форматах YYYYMMDDHHMMSS или "YYYY-MM-DD HH:MM:SS". Формат зависит от того в каком контексте используется NOW(): числовом или строковом.
PASSWORD()	Шифрует строку.
PERIOD_ADD(P:N)	Добавить N месяцев к периоду P (в формате YYMM).
PERIOD_DIFF(A, B)	Возвращает месяцы между A и B. Обратите внимание, что PERIOD_DIFF работает только с датами в форме YYMM или YYYMM.
TO_DAYS()	Меняет DATE (YYMMDD) на номер дня.
UNIX_TIMESTAMP([date])	Возвращает метку времени unix, если вызвана без date (секунды, начиная с GMT 1970.01.01 00:00:00). При вызове со столбцом TIMESTAMP вернет TIMESTAMP. date может быть также строкой DATE, DATETIME или числом в формате YYMMDD (или YYYMMDD).
USER()	Возвращает логин текущего пользователя.
WEEKDAY()	Возвращает день недели (0 = понедельник, 1 = вторник, ...).

Групповые функции в операторе ***select*:**

Следующие функции могут быть использованы в предложении GROUP:

AVG()	Среднее для группы GROUP.
SUM()	Сумма элементов GROUP.
COUNT()	Число элементов в GROUP.
MIN()	Минимальный элемент в GROUP.
MAX()	Максимальный элемент в GROUP.

Задание:

1. Выполнить проверку запросов из:

   - 1го раздела (2, 6, 7, 9, 12), 2го раздела (а, б, д)

2. Подготовить 3 запроса с использованием различных функций работа с полем дата, со строковыми данными (в том числе групповых).

3. Подготовить и выполнить средствами СУБД MySQL 4 запроса по выборке информации из таблиц базы данных с использованием агрегатных функций..

4. Подготовить и выполнить средствами СУБД MySQL 2 запроса по модификации информации (вставка, удаление, замещение) из таблиц базы данных для решения нижеприведенных задач. При этом в тех заданиях, где речь идет о создании таблиц, предполагается формировании постоянной таблицы базы данных.

Контрольные вопросы

1. Что такое коррелированный запрос? Чем отличается коррелированный запрос от некоррелированного?
2. Какие существуют ограничения на формирование коррелированного запроса?
3. Каким образом сохранить результаты запроса в таблице?
4. Какими средствами SQL реализуются следующие операции реляционной алгебры: ограничение, декартово произведение, проекция, пересечение, объединение, разность, соединение?
5. Что такое внешнее соединение?
6. В каких случаях вместо фразы IN можно использовать операцию сравнения?
7. Какие существуют средства группирования в SQL? Как они используются?

Список литературы, рекомендуемый к использованию по данной теме:
Основная: 1-11
Дополнительная: 1-11
Интернет-ресурсы: 1-7
Программное обеспечение: 1-4

Практическое занятие №6

Тема 11 - 12. Язык SQL. Функции и основные возможности. Средства манипулирования данными в SQL

1. Цель работы: познакомиться с возможностями MySQL по работе с хранимыми процедурами, функциями, триггерами, представлениями.

Формируемые компетенции: ПК-1, ПК-3

2. Подготовка к занятию

1. Изучить (повторить) теоретический материал.
2. Ознакомиться с программой работы.
3. Подготовить отчет о работе.
4. Ответить на контрольные вопросы.

3. Распределение времени занятия:

Всего: 90 мин

Вступительная часть 2 мин

Проверка готовности студентов к занятию 5 мин

Основная часть – 70 мин

Проверка выполнения практического занятия 10 мин

Заключительная часть 3 мин

4. Правила работы в лаборатории

К работе в лаборатории допускаются лица, изучившие правила и меры безопасности, сдавшие зачет по ним и усвоившие порядок выполнения лабораторной работы.

4.1. Требования безопасности перед началом работ

- ЗАПРЕЩАЕТСЯ: переодеваться, пользоваться огнем, курить, принимать пищу в лаборатории.
- Убедиться в целостности электрических розеток и разъемов. В лаборатории необходимо быть в сменной обуви.
- Включение компьютера производить только после получения допуска по выполняемой работе и разрешения преподавателя или лаборанта.

4.2. Требования безопасности во время работы

- выполняя практическое занятие, студенты обязаны использовать только вычислительную технику, периферийное оборудование, соединительные кабели, измерительное оборудование и носители информации, непосредственно относящиеся к данному лабораторному занятию;
- подключение и отключение составляющих вычислительного комплекса производить только при полном снятии напряжения со всех составляющих вычислительного комплекса;
- при обнаружении неисправностей в оборудовании немедленно отключить источники питания и доложить об этом руководителю занятий или лаборанту.

4.3. Требования безопасности по окончании работы

- доложить руководителю занятий или лаборанту о завершении работ;
- привести в порядок и сдать рабочее место лаборанту, и доложить руководству

Представления

Представления (views) можно сравнить с временными таблицами, наполненными динамически формируемым содержимым.. В настоящей реализации есть две возможности создания представлений: с использованием алгоритма временных таблиц MySQL и с созданием самостоятельной таблицы. Нас интересует именно второй способ (первый был реализован, скорее всего, исходя из соображений совместимости и унификации). Такие представления позволяют значительно снизить объём кода, в котором часто повторялись простые объединения таблиц. К ним (после создания) применимы любые запросы, возвращающие результат в виде набора строк. То есть команды SELECT, UPDATE, DELETE, можно применять так же, как и к реальным таблицам. Важно и то, что посредством представлений можно более гибко распоряжаться правами пользователей базы данных, так как в этом случае есть возможность предоставлять доступ на уровне отдельных записей различных таблиц.

Создание представлений

Для создания представлений используется команда CREATE VIEW

Синтаксис команды CREATE VIEW

```
CREATE
    [OR REPLACE]
    [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
    [DEFINER = { user | CURRENT_USER }]
    [SQL SECURITY { DEFINER | INVOKER }]
    VIEW view_name [(column_list)]
    AS select_statement
    [WITH [CASCADED | LOCAL] CHECK OPTION]
```

Пример создания и работы простейшего представления:

Create View v as Select column 1 from T

Insert into v Values (1)

Select * from v

Результат

```
+---------+
| column1 |
+---------+
| 1       |
```

```
+---------+
1 row in set (0.00 sec)
```

Представление может быть создано на основе различных параметров предложения SELECT, при этом можно ссылаться на другие таблицы и представления. Конструкция может использовать оператор UNION и другие подзапросы.

Синтаксис команды ALTER VIEW

Для внесения изменений в представление используется команда ALTER VIEW

```
ALTER
    [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
    [DEFINER = { user | CURRENT_USER }]
    [SQL SECURITY { DEFINER | INVOKER }]
    VIEW view_name [(column_list)]
    AS select_statement
    [WITH [CASCADED | LOCAL] CHECK OPTION]
```

Синтаксис команды DROP VIEW

Для удаления представления используется команда DROP VIEW

```
VIEW [IF EXISTS]
    view_name [, view_name] ...
    [RESTRICT | CASCADE]
```

ПРИМЕР

```
mysql> CREATE TABLE t (qty INT, price INT);
mysql> INSERT INTO t VALUES(3, 50);
mysql> CREATE VIEW v AS SELECT qty, price, qty*price AS value FROM
t;
mysql> SELECT * FROM v;
+------+-------+-------+
| qty  | price | value |
+------+-------+-------+
|    3 |    50 |   150 |
```

Хранимые процедуры и функции

В СУБД MySQL появилась возможность создания и хранения функций и процедур. Объявление и работа с процедурами и функциями отличаются в следующем:

- в заголовке функции помимо описания формальных параметров обязательно указывается тип возвращаемого ею результата;
- для возврата функцией значения в точку вызова среди ее операторов должен быть хотя бы один, в котором имени функции или переменной Result присваивается значение результата;
- вызов процедуры выполняется отдельным оператором;
- вызов функции может выполняться там, где допускается ставить выражение, в частности, в правой части оператора присваивания.

Пользовательские функции по функциональности похожи на хранимые процедуры. Разница заключается в том, что возможностей у них меньше (в частности, они должны возвращать только одно значение, например, скалярное или табличное), но их удобнее использовать с точки зрения синтаксиса.

Как процедуры, так и функции могут возвращать значения (в виде набора записей). Различие состоит в том, что функция вызывается из запроса, а процедура из отдельной команды.

На настоящий момент реализация хранимых процедур не поддерживает никаких внешних языков, но (по крайней мере, так заявляется) соответствует стандарту SQL:2003, позволяющему применять условные конструкции, итерации и обработку ошибок.

Пример создания хранимой процедуры в MySQL 5:

```
CREATE PROCEDURE p ()
LANGUAGE SQL
NOT DETERMINISTIC
SQL SECURITY DEFINER
COMMENT 'A Procedure' <--
SELECT CURRENT_DATE, RAND() FROM t
```

В данном случае мы создали процедуру с именем p, которая возвращает текущую дату и псевдослучайное число из таблицы t. Пример ее вызова и возвращаемого результата:

```
mysql> call p2()
+--------------+-----------------+
| CURRENT_DATE | RAND()          |
+--------------+-----------------+
| 2005-06-27   | 0.7822275075896 |
+--------------+-----------------+
1 row in set (0.26 sec)
Query OK, 0 rows affected (0.26 sec)
```

Чуть более сложный пример создания и использования функции:

```
CREATE FUNCTION factorial (n DECIMAL(3,0))
RETURNS DECIMAL(20,0)
DETERMINISTIC
BEGIN
DECLARE factorial DECIMAL(20,0) DEFAULT 1;
DECLARE counter DECIMAL(3,0);
SET counter = n;
factorial_loop: REPEAT
SET factorial = factorial * counter;
SET counter = counter - 1;
UNTIL counter = 1
END REPEAT;
RETURN factorial;
END
```

В приложении:

```
INSERT INTO t VALUES (factorial(pi))
SELECT s1, factorial (s1) FROM t
UPDATE t SET s1 = factorial(s1)
WHERE factorial(s1) < 5
```

Разумеется эффективность применения хранимых процедур существенно возрастает при вызове их с параметрами (аргументами). Ниже дан пример процедуры с обработкой переданных ей параметров:

```
CREATE PROCEDURE p1 (IN parameter1 INT)
BEGIN
DECLARE variable1 INT;
SET variable1 = parameter1 + 1;
IF variable1 = 0 THEN
INSERT INTO t VALUES (17);
END IF;
IF parameter1 = 0 THEN
UPDATE t SET s1 = s1 + 1; <--
ELSE
UPDATE t SET s1 = s1 + 2;
END IF;
```

END;

Вызов процедуры теперь будет таким:

mysql> CALL p2(0) // Query OK, 2 rows affected (0.28 sec)

и в результате запроса мы получим:

```
mysql> SELECT * FROM t
+----+
| s1 |
+----+
| 6  |
| 6  |
+------+
2 rows in set (0.01 sec)
```

Кроме условных, возможны и любые циклические конструкции:

CREATE PROCEDURE p3 ()

BEGIN

DECLARE v INT;

SET v = 0;

WHILE v < 5 DO

INSERT INTO t VALUES (v);

SET v = v + 1;

END WHILE;

END;

Вызов процедуры:

```
mysql> CALL p3()
+------+
| s1   |
+------+
……

|    0 |
|    1 |
|    2 |
|    3 |
|    4 |
+------+
```

```
Query OK, 1 row affected (0.00 sec)
```

Также применимы итерации, переходы, словом, всё, что предполагает стандарт.

Внутри функций и хранимых процедур осуществлена реализация курсоров, но, к сожалению, она пока ограничена (ASESITIVE, READ ONLY и NONSCROLL):

CREATE PROCEDURE p25 (OUT return_val INT)
BEGIN
DECLARE a,b INT;
DECLARE cur_1 CURSOR FOR SELECT s1 FROM t;
DECLARE CONTINUE HANDLER FOR NOT FOUND
SET b = 1;
OPEN cur_1;
REPEAT
FETCH cur_1 INTO a;
UNTIL b = 1
END REPEAT;
CLOSE cur_1;
SET return_val = a;
END;

Создание процедур и функций

```
CREATE
    [DEFINER = { user | CURRENT_USER }]
    PROCEDURE sp_name ([proc_parameter[,...]])
    [characteristic ...] routine_body

CREATE
    [DEFINER = { user | CURRENT_USER }]
    FUNCTION sp_name ([func_parameter[,...]])
    RETURNS type
    [characteristic ...] routine_body

proc_parameter:
    [ IN | OUT | INOUT ] param_name type
```

```
func_parameter:
    param_name type

type:
    Any valid MySQL data type

characteristic:
    LANGUAGE SQL
  | [NOT] DETERMINISTIC
  | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA
}
  | SQL SECURITY { DEFINER | INVOKER }
  | COMMENT 'string'

routine_body:
```

Внесение изменений

```
ALTER {PROCEDURE | FUNCTION} sp_name [characteristic ...]

characteristic:
    { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA
}
  | SQL SECURITY { DEFINER | INVOKER }
  | COMMENT 'string'
```

Удаление процедур и функций

```
DROP {PROCEDURE | FUNCTION} [IF EXISTS] sp_name
```

Вызов процедур и функций

```
CALL sp_name([parameter[,...]])
CALL sp_name[()]
```

Оператор `CALL` позволяет вызвать ранее определенную процедуру.

Пример1

```
CREATE PROCEDURE p1 (OUT ver_param VARCHAR(25), INOUT incr_param INT)
BEGIN
 # Set value of OUT parameter
 SELECT VERSION() INTO ver_param;
 # Increment value of INOUT parameter
 SET incr_param = incr_param + 1;
END;
```

Перед вызовом процедуры инициализируйте переменную указанные в параметрах `INOUT` .

После вызова процедуры значения будут установлены или изменены.

```
mysql> SET @increment = 10;
mysql> CALL p(@version, @increment);
mysql> SELECT @version, @increment;
+-----------------+------------+
| @version        | @increment |
+-----------------+------------+
| 5.1.12-beta-log | 11         |
```

Пример2

```
CREATE PROCEDURE `p2`(IN param1 CHAR(2) )
   NOT DETERMINISTIC
   SQL SECURITY DEFINER
   COMMENT ''
BEGIN
select * from s where snum=param1;
END;
```

```
Вызов процедуры
call p2 ('S1')
```

Пример3

```
CREATE PROCEDURE `My_proc2`(IN param1 CHAR(2) )

BEGIN                              /* start of block */
 DECLARE variable1 CHAR(10);              /* variables */
 IF param1 = 17 THEN                  /* start of IF */
  SET variable1 = 'birds';               /* assignment */
 ELSE
  SET variable1 = 'beasts';              /* assignment */
 END IF;                             /* end of IF */
select variable1;   /* statement */
END
```

Вызов процедуры

```
call p3 (10)
```

Триггеры

Три́ггер (англ. trigger) — это хранимая процедура особого типа, которую пользователь не вызывает непосредственно, а исполнение которой обусловлено наступлением определенного события (действием) — по сути добавлением INSERT или удалением DELETE строки в заданной таблице, или модификаци UPDATE данных в определенном столбце заданной таблицы реляционной базы данных. Триггеры применяются для обеспечения целостности данных и реализации сложной бизнес-логики. Триггер запускается сервером автоматически при попытке изменения данных в таблице, с которой он связан. Все производимые им модификации данных рассматриваются как выполняемые в транзакции, в которой выполнено действие, вызвавшее срабатывание триггера. Соответственно, в случае обнаружения ошибки или нарушения целостности данных может произойти откат этой транзакции. Момент запуска триггера определяется с помощью ключевых слов BEFORE (триггер запускается до выполнения связанного с ним события; например, до добавления записи) или AFTER (после события). В случае, если триггер вызывается до события, он может внести изменения в модифицируемую событием запись (конечно, при условии, что событие — не удаление записи). Некоторые СУБД накладывают ограничения на операторы, которые могут быть использованы в триггере (например, может быть запрещено вносить изменения в таблицу, на которой «висит» триггер, и т. п.)

Кроме того, триггеры могут быть привязаны не к таблице, а к представлению (VIEW). В этом случае с их помощью реализуется механизм «обновляемого представления». В этом случае ключевые слова BEFORE и AFTER влияют лишь на последовательность вызова триггеров, так как собственно событие (удаление, вставка или обновление) не происходит.

```
CREATE
   [DEFINER = { user | CURRENT_USER }]
   TRIGGER trigger_name trigger_time trigger_event
   ON tbl_name FOR EACH ROW trigger_stmt
```

Пример создания и работы триггера:

```
CREATE TABLE t22 (s1 INTEGER)

CREATE TRIGGER t22_bi
BEFORE INSERT ON t22
FOR EACH ROW
BEGIN
```

SET @x = 'Trigger was activated!';

SET NEW.s1 = 55;

END;

После этого при выполнении запросов получим:

```
mysql> INSERT INTO t22 VALUES (1)
mysql> SELECT @x, t22.* FROM t22 // вызывается триггер
+-----------------------+------+
| @x                    | s1   |
+-----------------------+------+
| Trigger was activated! | 55   |
+-----------------------+------+
1 row in set (0.00 sec)
```

Словарь данных

Иметь доступ к значениям метаданных – совершенно необходимое требование к современной СУБД. Ранее такая возможность в MySQL достигалась различными SHOW-командами, но такой подход имеет очевидные недостатки. Эти команды нельзя использовать в простых запросах с соединениями, и, что существенно, они не соответствовали стандартам, будучи специфичными для MySQL.

В новой версии СУБД появилась новая служебная база данных – INFORMATION_SCHEMA. Её наличие продиктовано тем же стандартом SQL:2003, и именно она решает задачу реализации словаря данных (data dictionary). INFORMATION_SCHEMA содержит таблицы, описывающие состояние и параметры сервера, в том числе определения и сущности таблиц. Это виртуальная база данных – физически (в виде файлов на диске) она не существует, вся информация динамически предоставляется сервером. Пример использования этой таблицы:

```
mysql> SELECT table_name, table_type, engine
    -> FROM INFORMATION_SCHEMA.tables
    -> WHERE table_schema = 'tp'
    -> ORDER BY table_type ASC, table_name DESC;
+------------+------------+--------+
| table_name | table_type | engine |
+------------+------------+--------+
| t2         | BASE TABLE | MyISAM |
| t1         | BASE TABLE | InnoDB |
| v1         | VIEW       | NULL   |
```

```
+------------+-----------+--------+
```

Другой пример работы со словарём данных – просмотр привелегий:

```
mysql> SELECT * FROM
    -> INFORMATION_SCHEMA.COLUMN_PRIVILEGES\G
*************************** 1. row ***************************
GRANTEE: 'peter'@'%'
TABLE_CATALOG: NULL
TABLE_SCHEMA: tp
TABLE_NAME: t1
COLUMN_NAME: col1
PRIVILEGE_TYPE: UPDATE
IS_GRANTABLE: NO
*************************** 2. row ***************************
GRANTEE: 'trudy'@'%'
TABLE_CATALOG: NULL
TABLE_SCHEMA: tp
TABLE_NAME: t2
COLUMN_NAME: col1
PRIVILEGE_TYPE: SELECT
IS_GRANTABLE: YES
```

Объявление переменных

Объявление. DECLARE Local Variables

Следующая команда позволяет объявлять локальные переменные, содержит возможность задания значения по умолчанию. Переменная может быть объявлена как выражения, не обязательна константа. Если значение по умолчанию не определено то равно NULL.

DECLARE *var_name*[,...] *type* [DEFAULT *value*]

Присваивание Variable SET Statement

SET *var_name* = *expr* [, *var_name* = *expr*] ...

SELECT ... INTO Statement

Оператор SELECT может перенаправить результат в переменные. Таким образом может быть преобразована только одна строка.

ПРИМЕР

```
SELECT col_name[,...] INTO var_name[,...] table_expr
SELECT id,data INTO x,y FROM test.t1 LIMIT 1;
```

Условия и ограничения

Объявление условий

```
DECLARE condition_name CONDITION FOR condition_value

condition_value:
    SQLSTATE [VALUE] sqlstate_value
  | mysql_error_code
```

Объявление ограничений

```
DECLARE handler_type HANDLER FOR condition_value[,...] statement

handler_type:
    CONTINUE
  | EXIT
  | UNDO

condition_value:
    SQLSTATE [VALUE] sqlstate_value
  | condition_name
  | SQLWARNING
  | NOT FOUND
  | SQLEXCEPTION
  | mysql_error_code
```

Пример

```
mysql> CREATE TABLE test.t (s1 int,primary key (s1));
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter //

mysql> CREATE PROCEDURE handlerdemo ()
    -> BEGIN
    ->   DECLARE CONTINUE HANDLER FOR SQLSTATE '23000' SET @x2 =
1;
    ->   SET @x = 1;
    ->   INSERT INTO test.t VALUES (1);
    ->   SET @x = 2;
    ->   INSERT INTO test.t VALUES (1);
    ->   SET @x = 3;
    -> END;
    -> //
Query OK, 0 rows affected (0.00 sec)

mysql> CALL handlerdemo()//
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @x//
    +------+
    | @x   |
    +------+
    | 3    |
    +------+
    1 row in set (0.00 sec)
```

Если вы хотите игнорировать условие вы должны сгенерировать ограничение и ассоциировать его с пустым блоком .

```
DECLARE CONTINUE HANDLER FOR SQLWARNING BEGIN END;
```

Пример

```
CREATE PROCEDURE p ()
BEGIN
  DECLARE i INT DEFAULT 3;
  retry:
    REPEAT
      BEGIN
        DECLARE CONTINUE HANDLER FOR SQLWARNING
          BEGIN
            ITERATE retry;  # illegal
          END;
      END;
      IF i < 0 THEN
        LEAVE retry;        # legal
      END IF;
      SET i = i - 1;
    UNTIL FALSE END REPEAT;
END;
```

Курсоры

Курсор — в некоторых реализациях языка программирования SQL (Oracle, Microsoft SQL Server) — получаемый при выполнении запроса результирующий набор и связанный с ним указатель текущей записи.

Курсор может возвращать одну строку, несколько строк или ни одной строки. Для запросов, возвращающих более одной строки, можно использовать только явный курсор. Для повторного создания результирующего набора для других значений параметров курсор следует закрыть, а затем повторно открыть.

Курсор может быть объявлен в секциях объявлений любого блока PL/SQL, подпрограммы или пакета.

Операторы управления явным курсором

- Оператор CURSOR выполняет объявление явного курсора.
- Оператор OPEN открывает курсор, создавая новый результирующий набор на базе указанного запроса.
- Оператор FETCH выполняет последовательное извлечение строк из результирующего набора от начала до конца.
- Оператор CLOSE закрывает курсор и освобождает занимаемые им ресурсы

Курсоры поддерживают хранимые процедуры и функции. Сейчас курсоры имеют три свойства:

- Asensitive: The server may or may not make a copy of its result table
- Read only: Not updatable
- Non-scrollable: Can be traversed only in one direction and cannot skip rows

Курсоры должны быть объявлены перед объявлением ограничений. Переменные и условия должны быть объявлены перед курсором.

Объявление курсоров

Оператор объявления курсора. М программе можно объявлять несколько курсоров, каждый курсор в блоке должен иметь уникальное имя.

```
DECLARE cursor_name CURSOR FOR select_statement
```

Условие открытия Cursor OPEN Statement

Оператор открывает ранее объявленный курсор.

```
OPEN cursor_name
```

Выполнение курсора Cursor FETCH Statement

```
FETCH cursor_name INTO var_name [, var_name] ...
```

Условия закрытия Cursor CLOSE Statement

```
CLOSE cursor_name
```

Пример

```
CREATE PROCEDURE curdemo()
BEGIN
  DECLARE done INT DEFAULT 0;
  DECLARE a CHAR(16);
  DECLARE b,c INT;
  DECLARE cur1 CURSOR FOR SELECT id,data FROM test.t1;
```

```
  DECLARE cur2 CURSOR FOR SELECT i FROM test.t2;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

  OPEN cur1;
  OPEN cur2;

  REPEAT
    FETCH cur1 INTO a, b;
    FETCH cur2 INTO c;
    IF NOT done THEN
       IF b < c THEN
          INSERT INTO test.t3 VALUES (a,b);
       ELSE
          INSERT INTO test.t3 VALUES (a,c);
       END IF;
    END IF;
  UNTIL done END REPEAT;

  CLOSE cur1;
  CLOSE cur2;
END
```

Задание

Представления

1. Составить представление, возвращающее объем поставок деталей для изделий за заданный календарный месяц
2. Добавить столбец стоимость детали в таблицу SPJ. Создать соответствующее представление (наименование поставщика, наименование детали, наименование изделия, стоимость детали, количество, стоимость поставки.
3. Добавить столбец стоимость детали в таблицу P. Создать представление отражающее стоимость поставки.

Процедуры

1. Составить процедуру, отражающую состав изделия (детали изделия).
2. Составить процедуру, возвращающую расчетную стоимость изделия, учитывая, что для изделия требуется K деталей каждого требуемого наименования (см. табл 1).
3. Составить процедуру , отражающую вес изделия (п4) учитывая что для изделия требуется K деталей каждого требуемого наименования (см. табл 1).
4. С помощью условных операторов разделить всех поставщиков на три категории по количеству поставляемых деталей (ABC анализ) 20 40 60 %
5. Тоже, но по стоимости поставки
6. Определить оптимального поставщика для изделия (см табл 1) для производства максимального количества изделий за период
7. Определить оптимального поставщика для изделия (см табл 1) для производства максимального количества изделий по минимальной стоимости
8. При условии, что поставщик может поставлять не более одной поставки в неделю, а максимальное количество деталей в поставке не выше среднего за период

Функции

1. С помощью функций получить таблицу, отражающую информацию о перечне изделий, для которого выполняется поставка

S1	J1 J2 J4
S2	J5

2. Тоже, но с наименованиями изделий.
3. С помощью функций получить таблицу, отражающую информацию о перечне деталей из которых состоит дневная поставка
4. Тоже , но с наименованиями деталей
5. Получить наименование поставщика поставляемого самое большое количество деталей

6. Получить наименование поставщика поставляемого самое большое количество деталей, для какого либо изделия

Работа с текстовым файлом

1. Создать текстовый файл, содержащий информацию о поставщике, поставившего за последний месяц деталей на большую сумму и меньшего веса.

Курсоры

2. При заполнении поставки поле дата всегда заполнять текущей датой.

Таблица 11 -Варианты

задание	Вар 1	Вар 2	Вар 3	Вар 4	Вар 5
1,9,10	J1	J2	J3	J4	J5
	ЯНВАРЬ	ФЕФРАЛЬ	МАРТ	АПРЕЛЬ	МАЙ
5,6	15	20	25	30	35
18	Max;Min	>AVG;Min	MIN;>AVG	>AVG;>AVG	Max;Max

Контрольные вопросы:

— Пояснить, каким образом предоставляются права на пользование базой данных и отдельными ее таблицами

— Пояснить, каким образом изымаются права на пользование базой данных и отдельными ее таблицами.

— Пояснить понятие внешней базы данных.

— Пояснить, как идентифицируется таблица внешней базы данных

— Пояснить, как идентифицируется таблица внешней распределенной базы данных.

Список литературы, рекомендуемый к использованию по данной теме:
Основная: 1-11
Дополнительная: 1-11
Интернет-ресурсы: 1-7
Программное обеспечение: 1-4

Практическое занятие №7

Тема 13 - 14. Безопасность SQL при прикладном программировании. Компиляторы SQL и проблемы безопасности оптимизации

1. Цель работы: ознакомиться со структурой программ, использующие средства ODBC, реализовать и выполнить с использованием простейших функций ODBC запросы, связанные с модификацией таблиц.

Формируемые компетенции: ПК-1, ПК-3

2. Подготовка к занятию

1. Изучить (повторить) теоретический материал.
2. Ознакомиться с программой работы.
3. Подготовить отчет о работе.
4. Ответить на контрольные вопросы.

3. Распределение времени занятия:

Всего: 90 мин

Вступительная часть 2 мин

Проверка готовности студентов к занятию 5 мин

Основная часть – 70 мин

Проверка выполнения практического занятия 10 мин

Заключительная часть 3 мин

4. Правила работы в лаборатории

К работе в лаборатории допускаются лица, изучившие правила и меры безопасности, сдавшие зачет по ним и усвоившие порядок выполнения лабораторной работы.

4.1. Требования безопасности перед началом работ

- ЗАПРЕЩАЕТСЯ: переодеваться, пользоваться огнем, курить, принимать пищу в лаборатории.

- Убедиться в целостности электрических розеток и разъемов. В лаборатории необходимо быть в сменной обуви.

- Включение компьютера производить только после получения допуска по выполняемой работе и разрешения преподавателя или лаборанта.

4.2. Требования безопасности во время работы

- выполняя практическое занятие, студенты обязаны использовать только вычислительную технику, периферийное оборудование, соединительные кабели, измерительное оборудование и носители информации, непосредственно относящиеся к данному лабораторному занятию;

- подключение и отключение составляющих вычислительного комплекса производить только при полном снятии напряжения со всех составляющих вычислительного комплекса;

- при обнаружении неисправностей в оборудовании немедленно отключить источники питания и доложить об этом руководителю занятий или лаборанту.

4.3. Требования безопасности по окончании работы

- доложить руководителю занятий или лаборанту о завершении работ;

- привести в порядок и сдать рабочее место лаборанту, и доложить руководству

Содержание работы и методические указания к ее выполнению

Для выполнения работы необходимо

— ознакомиться со структурой программы ODBC;

— изучить функции выполнения подготовительных операций в ODBC-программе;

— ознакомиться со средствами обработки ошибок в ODBC-программе;

— изучить функции непосредственного и подготавливаемого выполнения SQL-операторов, передачи параметров;

— настроить среду выполнения, разработать и отладить ODBC-программу.

1. Структура ODBC-программы и функции инициализации

Общая структура ODBC-программы имеет вид:

— Идентификатор окружения каждого приложения ODBC описывается функцией **SQLAllocEnv,** который должен быть освобожден в конце приложения с помощью функции **SQLFreeEnv.** Тип идентификатора окружения **HENV.**

RETCODE **SQLAllocEnv** (*env*)

HENV *env* - указатель области хранения в памяти идентификатора окружения.

RETCODE **SQLFreeEnv** (*env*)

HENV *env* - имя идентификатора окружения, который должен быть освобожден.

— Идентификатор соединения представляет собой соединение между источником данных и прикладной программой. Для каждого источника данных, с которым приложение предполагает соединиться должен быть назначен идентификатор соединения **SQLAllocConnect** и освобожден **SQLFreeConnect.** Приложение может соединиться с источником данных, используя **SQLConnect** и разъединиться, используя **SQLDisconnect.** Тип идентификатора соединения **HDBC.**

RETCODE **SQLAllocConnect** (env, dbc)

HENV *env* - указатель на идентификатор окружения прикладной программы.

HDBC *dbc* - указатель области хранения памяти для идентификатора соединения.

RETCODE **SQLFreeConnect** (dbc)

HDBC *dbc* - указатель области памяти для освобождаемого идентификатора соединения.

RETCODE **SQLConnect**(*dbc, szDSN, sbDSN, szUID, sbUID, szAuthStr, cbAuthStr*)

HDBC *dbc* - идентификатор соединения.

UCHAR *szDSN* - строка с именем источника данных, с которым прикладная программа собирается соединиться.

SWORD *sbDSN* - длина строки источника данных, если это имя имеет нулевое окончание, то этот параметр можно установить в SQL_NTS, который является константой ODBC и используется вместо длины параметра, если параметр содержит строку с нулевым окончанием.

UCHAR *szUID* - имя пользователя.

SWORD *sbUID* - длина имени пользователя или SQL_NTS.

UCHAR *szAut*h*Str* - пароль пользователя.

SWORD *cbAut*h*Str* - длина пароля.

RETCODE **SQLDisconnect** (*dbc*)

HDBC *dbc* - идентификатор доступа для отсоединения.

— Идентификатор оператора аналогичен идентификатору окружения или соединения за исключением того, что он ссылается на SQL-оператор. Идентификатор соединения может быть связан несколькими идентификаторами операторов, но каждый идентификатор оператора связан только со своим идентификатором соединения. Чтобы назначить

идентификатор оператора, приложение вызывает функцию **SQLAllocStmt**, а для освобождения**SQLFreeStmt.** Тип идентификатора оператора **HSTMT.**

RETCODE **SQLAllocStmt** (*dbc, stmt*)

HDBC *dbc* - идентификатор соединения.

HSTMT *stmt* - указатель области хранения в памяти для идентификатора оператора.

RETCODE **SQLFreeStmt** (*stmt, fOption*)

HSTMT *stmt* -идентификатор оператора.

UWORD *fOption* - одна из следующих опций:

SQL_CLOSE - закрывает курсор, связанный с hstmt, (если он был определен) и отбрасывает все ожидаемые результаты. Прикладная программа может вновь открыть этот курсор позднее, вновь выполнить оператор SELECT с теми же самыми или другими значениями параметров. Если курсор не открыт, то эта опция не повлияет на программу.

SQL_DROP - освобождает hstmt, освобождает все ресурсы, связанные с ним, закрывает курсор, если он открыт, и отбрасывает все ожидаемые строки. Эта опция завершает все обращения к hstmt. hstmt обязательно должен быть переназначен для повторного использования. Эта опция освобождает все ресурсы, которые были определены с помощью функции **SQLFreeStmt**.

SQL_UNBIND - освобождает все буферы столбцов, которые повторно используются функцией **SQLBindCol** для данного идентификатора оператора.

SQL_RESET_PARAMS - освобождает все буферы параметров, которые были установлены функцией **SQLBindCol** для данного идентификатора оператора.

2. Средства отслеживания ошибок

Для отслеживания ошибок в ODBC используется функция **SQLError**, которая возвращает сообщение об ошибке при неудачном завершении какой-либо функции ODBC.

Каждая ODBC-функция возвращает RETCODE, который принимает одно из нижеследующих значений:

— *SQL_SUCCESS* Операция выполнена без ошибки.;

— *SQL_SUCCESS_WITH_INFO* Функция завершена, но при вызове **SQLError** указывает на ошибку. В большинстве случаев это возникает, когда значение, которое должно быть возвращено очень большого размера, чем это предусмотрено буфером прикладной программы;

— *SQL_ERROR* Функция не была завершена из-за возникшей ошибки. При вызове **SQLError** можно будет получить больше информации о сложившейся ситуации;

— *SQL_INVALID_HANDLE* Не правильно определен идентификатор окружения, соединения или оператора. Это часто случается, когда идентификатор используется после того, как он был освобожден или если был определен нулевой указатель;

— *SQL_NO_DATA_FOUND* Больше нет подходящей информации. Фактически это не является ошибкой. Чаще всего такой статус возникает при использовании курсора, когда больше нет строк для его продвижения;

— *SQL_NEED_DA*TA Необходимы данные для параметра.

RETCODE **SQLError** (*henv, hdbc, hsmt, szSqlState, pfNativeError, szErrorMsg, cbErrorMsgMax, cbErrorMsg*)

HENV *henv* - идентификатор окружения.

HDBC *hdbc* - идентификатор соединения.

HSTMT *hsmt* - идентификатор оператора.

UCHAR *szSqlState* - SQLSTATE в качестве строки завершения.

SDWORD *pfNativeError* - в этом параметре и будет возвращена ошибка, возникшая в СУБД, а также ее собственный код. Если соответствующего собственного кода ошибки не существует, то возвращается ноль.

UCHAR *szErrorMsg* - указатель на буфер, куда будет возвращен текст ошибки (строка с нулевым окончанием).

SWORD *cbErrorMsgMax* - максимальный размер вышеописанного буфера, должен быть меньше или равен SQL_MAX_MESSAGE_LENGTH-1.

SWORD *cbErrorMsg* - сюда возвращается число байт, скопированных в буфер.

3. Непосредственное и подготавливаемое выполнение операторов SQL

Непосредственное выполнение используется в тех случаях, когда

— SQL-операторы, которые должны быть выполнены, будут выполняться только один раз;

— не требуется информации о результирующем множестве до выполнения оператора;

SQLExecDirect представляет собой самый быстрый способ запустить SQL-оператор при одноразовом выполнении.

RETCODE **SQLExecDirect** (*hstmt, szSqlStr, cbSqlStr*)

HSTMT *hstmt*- идентификатор оператора.

UCHAR *szSqlStr*- строка с SQL-оператором.

SDWORD *cbSqlStr* - длина строки szSqlStr.

Подготавливаемое выполнение предпочтительнее использовать, когда необходима информация о результирующем множестве до выполнения оператора или когда требуется выполнить SQL-оператор более одного раза. Для этого необходимы две функции **SQLPrepare** и **SQLExecute.**

SQLPrepare подготавливает SQL-строку для выполнения:

RETCODE **SQLPrepare** (*hstmt, szSqlStr, cbSqlStr*)

HSTMT *hstmt* - идентификатор оператора

UCHAR *szSqlStr*- строка с SQL-оператором

SDWORD *cbSqlStr* - длина строки szSqlStr

SQLExecute выполняет подготовленный оператор:

RETCODE **SQLExecute**(*hstmt*)

HSTMT *hstmt* - идентификатор оператора

SQLPrepare и **SQLExecDirect** отличаются тем, что при вызове **SQLPrepare** оператор SQL в действительности не выполняется, вместо этого определяется путь доступа к данным источника данных. Использование подготавливаемого выполнения удобно для операторов, которые будут выполняться более одного раза. Так как путь доступа к данным уже определен, то выполнение может осуществляться несколько быстрее, чем при использовании **SQLExecDirect**. Кроме того, каждый вызов **SQLExecute** передает базе данных только идентификатор для планирования обращения, а не весь SQL-оператор.

4. Использование параметров при выполнении

Параметры используются при непосредственном и подготавливаемом выполнении. Маркеры параметров определяются в SQL-операторах с помощью знаков "?". Например, SELECT n_post FROM s WHERE town=? или INSERT INTO p(name, town)

VALUES (?,?). Для того, чтобы связать буфер с маркерами параметров, прикладная программа должна вызвать **SQLBindParameter**:

RETCODE **SQLBindParameter** (*hsmt, ipar, fParamType, fCType, fSqlType, cbColDef, ibScale, rgbValue, cbValueMax, pcValue*)

HSTMT *hsmt* - идентификатор оператора, он должен быть точно тем же идентификатором оператора, с которым этот оператор подготавливается и выполняется.

UWORD *ipar* - номер параметра для связи. В операторе SQL операторы нумеруются слева направо, начиная с 1. Например, для следующего SQL-оператора используется три параметрических маркера : INSERT INTO j (n_izd, name, town) VALUES (?,?,?). Чтобы связать эти параметры, **SQLBindParameter** вызывается три раза с ipar, установленным в 1, 2 и 3 соответственно.

SWORD *fParamType* - является типом параметра для связи и может принимать одно из трех значений: SQL_PARAM_INPUT, SQL_PARAM_INPUT_OUTPUT или SQL_ PARAM_OUTPUT. SQL_PARAM_INPUT используется для процедур, использующих параметры ввода. SQL_PARAM_INPUT_OUTPUT маркирует параметр ввода/вывода в процедуре. SQL_PARAM_OUTPUT маркирует значение возврата или параметр вывода в процедуре.

SWORD *fCType* - является С-типом данных для параметра. Это тип данных из которого необходимо конвертировать данные.

SWORD *fSqlType* - является ODBC типом данных для параметра. Это тип данных в которых конвертируются данные и он должен совпадать с SQL-типом столбца, соответствующего этому параметрическому маркеру.

UDWORD *cbColDef* - точность столбца или выражения соответствующего маркера параметра.

SWORD *ibScale* - размер столбца или выражения соответствующего маркера параметра

PTR *rgbValue* - указатель буфера для данных параметра, который при вызове **SQLExecute** или **SQLExecuteDirect** содержит действительные значения параметра.

SDWORD *cbValueMax* - максимальная длина буфера rgbValue.

SDWORD *pcbValue* - указатель буфера для длины параметра.

Ниже приведены основные значения С- и SQL-типов параметров.

C-тип		SQL-тип	
SQL_C_BINARY	SQL_C_FLOAT	SQL_BINARY	SQL_DOUBLE
SQL_C_BIT	SQL_C_TIME	SQL_BIT	SQL_FLOAT
SQL_C_CHAR	SQL_C_DEFAULT	SQL_CHAR	SQL_INTEGER
SQL_C_DATE	SQL_C_SLONG	SQL_DATE	SQL_REAL
SQL_C_DOUBLE	SQL_C_SSHORT	SQL_DECIMAL	SQL_SMALLINT
		SQL_TIME	SQL_VARCHAR

5. Настройка доступа к источнику данных

Настройка доступа к источнику данных включает:

- редактирование файла .odbc.ini.;
- определение переменной ODBCINI;
- установка переменных окружения СУБД;
- включение необходимых заголовочных файлов.

Для настройки источника данных, необходимо проверить находится ли системный текстовый файл ***.odbc.ini*** в домашней директории пользователя и отредактировать его, настроив на требуемые источники данных.

В файле .odbc.ini должен быть под некоторым идентификатором описан требуемый источник данных, имя которого используется функцией **SQLConnect()**, и далее должен присутствовать раздел с данным именем, в котором содержатся атрибуты, описывающие источник данных. Файл ***.odbc.ini*** должен содержать имя источника данных, имя сервера баз данных, имя базы данных и другие атрибуты.

В переменной окружения ODBCINI необходимо указать полное имя системного файла ***.odbc.ini*** из домашней директории. Сделать это можно, либо введя с консоли соответствующую команду, либо поместив эту команду в файл загрузки .login. Форма записи команды зависит от используемой программы Shell-интерпретатора.

Системные требования зависят от той СУБД, с которой работает пользователь. Например, при работе с СУБД Informix это обеспечивается переменными окружения, выставленными в файле ***.cshrc*** в домашней директории пользователя.

Для получения доступа до ODBC-функций, в программе должен быть описан заголовочный файл sqlext.h. В файле sqltypes.h находятся описание типов данных используемых в ODBC.

Как и любой исходный файл, написанный на языке Си, файл с программой, вызывающей ODBC-функции, должен иметь расширение .c. При компиляции необходимо подключить необходимые библиотеки, используемые в вызываемых функциях. Поскольку в этом случае командная строка принимает достаточно сложный вид, целесообразно на уровне системного администратора оформить командный файл, в котором подключаются необходимые библиотеки.

Последовательность выполнения лабораторной работы

- Убедиться в наличии и заполненности базы данных поставщиков, деталей, изделий, поставок.
- Разработать ODBC-программу для решения задачи 1 из соответствующего варианта с помощью функций непосредственного выполнения.
- Разработать ODBC-программу для решения задачи 2 из соответствующего варианта с помощью функций подготавливаемого выполнения.
- После выполнения лабораторной работы привести базу данных в исходное состояние.

Требования к разрабатываемой программе

Разрабатываемая программа должна удовлетворять следующим требованиям:

- все используемые функции ODBC должны анализироваться на корректность кода возврата;
- в программе должен быть предусмотрен вывод сообщений обо всех шагах ее выполнения, в том числе и о возможных ошибках;
- при выполнении запросов должно быть предусмотрено использование параметров; параметры варианта задания должны быть введены в ходе выполнения программы и переданы в SQL-запрос;
- при выполнении программы должна контролироваться целостность базы данных;
- программа должна быть достаточно документирована.

Варианты заданий

Вариант 1

— Из таблицы поставок удалить поставки при заданных параметрах номера поставщика (имени поставщика) и номера детали.

— Увеличить рейтинг поставщика, выполнившего наибольшую поставку некоторой детали, на указанную величину.

Вариант 2

— Удалить всех поставщиков из указанного города.

— Изменить цвет самой тяжелой детали на указанный.

Вариант 3

— Вставить поставщика с заданными параметрами.

— Удалить самую легкую деталь.

Вариант 4

— Удалить поставщика, выполнившего меньше всего поставок.

— Изменить название детали указанного цвета и веса.

Вариант 5

— Удалить изделие из заданного города.

— В таблице поставок изменить номер поставщика при заданном номере детали и изделия.

Вариант 6

— Увеличить рейтинг поставщика, выполнившего больший суммарный объем поставок, на указанную величину.

— Вставить деталь с заданными параметрами.

Вариант 7

— Изменить название и город детали с максимальным весом на указанные значения.

— Удалить из таблицы поставок все поставки конкретного поставщика.

Вариант 8

— Увеличить рейтинг поставщика, выполнившего большее число поставок, на указанную величину.

— Увеличить вес деталей из Лондона на некоторую величину.

Контрольные вопросы

- Какова структура ODBC-программы? Перечислите ее основные компоненты.
- С помощью каких средств ODBC можно отследить наличие ошибки?
- В каких случаях непосредственное выполнение операторов является наиболее эффективным?
- Когда используется подготавливаемое выполнение?
- Как описываются маркеры параметров, и какая для этого предусмотрена функция? Каким образом можно связать несколько параметров?
- С помощью какого параметра можно освободить буферы всех столбцов?
- Как описать доступ до необходимой базы данных?
- С помощью какой функции описывается соединение с необходимым источником данных? Каковы ее параметры?

Список литературы, рекомендуемый к использованию по данной теме:
Основная: 1-11
Дополнительная: 1-11
Интернет-ресурсы: 1-7
Программное обеспечение: 1-4

Практическое занятие №8

Тема 15 - 16. Безопасность архитектуры «клиент-сервер». Безопасность распределенных баз данных

1. Цель работы: ознакомиться с основными понятиями разработки CGI-скриптов с целью написания простейших CGI-программ доступа к базам данных с использованием языка ESQL/C.

Формируемые компетенции: ПК-1, ПК-3

2. Подготовка к занятию

1. Изучить (повторить) теоретический материал.
2. Ознакомиться с программой работы.
3. Подготовить отчет о работе.
4. Ответить на контрольные вопросы.

3. Распределение времени занятия:

Всего: 90 мин

Вступительная часть 2 мин

Проверка готовности студентов к занятию 5 мин

Основная часть – 70 мин

Проверка выполнения практического занятия 10 мин

Заключительная часть 3 мин

4. Правила работы в лаборатории

К работе в лаборатории допускаются лица, изучившие правила и меры безопасности, сдавшие зачет по ним и усвоившие порядок выполнения лабораторной работы.

4.1. Требования безопасности перед началом работ

- ЗАПРЕЩАЕТСЯ: переодеваться, пользоваться огнем, курить, принимать пищу в лаборатории.

- Убедиться в целостности электрических розеток и разъемов. В лаборатории необходимо быть в сменной обуви.

- Включение компьютера производить только после получения допуска по выполняемой работе и разрешения преподавателя или лаборанта.

4.2. Требования безопасности во время работы

- выполняя практическое занятие, студенты обязаны использовать только вычислительную технику, периферийное оборудование, соединительные кабели, измерительное оборудование и носители информации, непосредственно относящиеся к данному лабораторному занятию;

- подключение и отключение составляющих вычислительного комплекса производить только при полном снятии напряжения со всех составляющих вычислительного комплекса;

- при обнаружении неисправностей в оборудовании немедленно отключить источники питания и доложить об этом руководителю занятий или лаборанту.

4.3. Требования безопасности по окончании работы

- доложить руководителю занятий или лаборанту о завершении работ;

- привести в порядок и сдать рабочее место лаборанту, и доложить руководству

Содержание работы и методические указания к ее выполнению

Для выполнения работы необходимо

- изучить основы языка разметки гипертекста HTML;
- ознакомиться со структурой спецификации CGI и CGI-скрипта;
- изучить необходимые конструкции HTML-формы;
- ознакомиться с переменными CGI-окружения;
- изучить алгоритм обработки данных HTML-формы с использованием методов GET и POST;
- с использованием средств языка ESQL/C разработать и отладить программу доступа к базе данных.

Общая схема доступа к базам данных с использованием CGI-скриптов имеет вид:

<="" p="">

Спецификация CGI описывает формат и правила обмена данными между программным обеспечением WWW-сервера и запускаемой программой и представляет собой общую среду и набор протоколов для внешних приложений, которые используются при взаимодействии с Web-сервером.

CGI-программа (CGI-скрипт) представляет собой программу локальной операционной системы сервера (в двоичном виде или в виде программы для интерпретатора), которая может быть вызвана из среды WWW. Для инициирования CGI достаточно, чтобы в URL-адресе был указан путь до запускаемой программы. Программное обеспечение WWW-

сервера исполняет эту программу, передает ей входные параметры и возвращает результаты ее работы, как результат обработки запроса, клиенту.

С целью облегчения администрирования CGI-программ, а также для удовлетворения требованиям безопасности CGI-программы группируются в одном или нескольких явно указанных серверу каталогах. При выполнении лабораторной работы в качестве таких каталогов выступают каталоги *cgi-bin* в домашней директории пользователя или в директории *public_html*. При этом

— права доступа для каталога, в котором хранятся CGI-скприпты, должны быть самые широкие, иначе скрипт не сможет создавать файлы, нужные ему для работы (кроме того, такие же права должны быть и у файлов, к которым обращается скрипт);

— файлы CGI-скриптов (как, правило, с расширением cgi) должны быть обязательно исполняемыми.

Общепринятым средством организации интерфейса с пользователем в WWW-среде является HTML-форма. Если целью формы является сбор данных для последующей передачи их серверу, то такая форма должна обязательно содержать:

1. Адрес CGI-скрипта (программы-обработчика, расположенной на некотором сервере), которому будут пересылаться данные из формы. При этом выполняется пересылка не всей страницы целиком, а только значений, соответствующих элементам управления формы, которая инициировала запуск программы-обработчика.

2. Метод передачи данных (наиболее применяемые POST и GET).

3. Некоторый объект формы, при нажатии на который произойдет пересылка данных.

Форма задается в HTML-документе с помощью тега *FORM*. Все элементы управления находятся внутри контейнера *<FORM>...</FORM>*.

<FORM action="http://.....cgi" method="GET"|"POST"
enctype="encodingType" name="formName" target="windowName"
onSubmit="Handler">
... Поля формы ...
</FORM>

Среди атрибутов HTML-формы для целей CGI-программирования наиболее важны

— *action.* Этот атрибут задает URL-адрес программы (CGI-скрипта), которая будет обрабатывать данные формы. Если он опущен, используется URL-адрес текущего документа;

— *method.* Задается метод. По умолчанию предполагается GET.

Основными методами являются методы **GET** и **POST.** В зависимости от метода (GET или POST) данные формы будут помещены в переменную окружения *QUERY_STRING* или поданы на стандартный ввод *STDIN*. В случае метода GET,используемого по умолчанию, данные добавляются в конец URL-адреса и отделяются знаком "?". С помощью метода POST информация, введенная пользователем, посылается непосредственно в сценарий с помощью выходного потока сервера STDOUT и входного потока STDIN вашего сценария. Преимуществом использования метода POST является неограниченность длины передаваемого сообщения и безопасность передачи по сравнению с GET.

При нажатии некоторой кнопки на форме HTML документа происходит передача данных броузером серверу, на котором находится программа-обработчик данных. Основная работа такой программы заключается в:

— получении данных формы HTML-документа в виде строки, в которой перечислены значения всех элементов HTML формы, инициировавшей запуск программы (данная строка записана в соответствие с четко определенным форматом);

— разборе полученной строки;

— обработке данных (например, занесение полученных значений в базу данных или выборка некоторых записей из базы данных по полученным значениям и т.д. и т.п.);

— формировании HTML-документа, который будет передан сервером программе-броузеру.

Обработка CGI-скриптом данных формы составляет шаблонную часть скрипта.

Данные, веденные в форме, передаются CGI-модулю в виде последовательности символов через входной поток (метод **POST**) или переменную CGI-окружения (метод **GET**) в формате:

имя=значение&имя1=значение1&...&имяN=значениеN,

где имяi - значение параметра name из элемента HTML-формы, значениеi - введенное или выбранное значение независимо от его типа.

Алгоритм обработки передаваемой в CGI-скрипт строки данных состоит в разделении ее на пары **имя=значение** и декодирования каждой пары, учитывая, что все пробелы в введенных значениях в форме сервером были заменены символом "+" и символы с десятичным кодом больше 128 преобразованы в символ "%" и следующим за ним шестнадцатеричным кодом символа.

После шаблонной части CGI-скрипта следует содержательная часть, в которой можно анализировать полученные данные, обращаясь, при необходимости, к различным таблицам баз данных. При написании CGI-скрипта на языке ESQL/C работа с базой данных ведется средствами встроенного языка SQL и главных переменных, через которые передаются и возвращаются данные к серверу баз данных. Результаты обработки данных, полученных из базы данных, а также другие информационные сообщения передаются функцией *printf()* в выходной поток с учетом форматирования HTML-документа.

CGI-скрипт, запускаемый из среды WWW, должен содержать информацию о сервере баз данных в своем теле. Эта информация состоит в задании в CGI-скрипте системных переменных, определяющих каталог с программным обеспечением сервера, имя сервера, параметры перекодировки и прочие параметры.

Важной особенностью работы с базой данных посредством CGI-скрипта является обеспечение достаточных условий безопасности в отношении данных.

В том случае, когда любой пользователь без ограничений может иметь доступ к данным, средствами программы *dbaccess*, либо средствами иного программного приложений владелец базы данных SQL-оператором

Grant connect to nobody

предоставляет пользователю операционной системы **nobody** минимальные права, в том числе в отношении баз данных. Это позволяет при написании CGI-скрипта использовать простейший вариант подключения к серверу баз данных без проверки индивидуальных полномочий пользователя.

$Connect to 'database@server_name';

После окончания работы с приложением владелец базы данных должен отобрать полномочия у пользователя **nobody**.

Revoke connect from nobody

В том случае, когда необходимо контролировать доступ того или иного пользователя к базе данных или когда разные пользователи обладают разными полномочиями в отношении доступа к данным, следует завести пользователя на сервере баз данных, дать ему требуемые полномочия в отношении тех или иных действий над используемыми таблицами посредством SQL-оператора *Grant*, после чего каждый раз при обращении проверять эти полномочия. При этом CGI-приложение должно знать и каким-либо образом хранить все имена и пароли и полномочия пользователей, которым разрешен доступ к данным.

$Connect to 'database@server_name' user 'name' using $parol;

Последовательность выполнения лабораторной работы

1. Убедиться в наличии и заполненности базы данных поставщиков, деталей, изделий, поставок.
2. Разработать и отладить HTML-формы для ввода данных пользователя согласно варианту задания.
3. Средствами языка ESQL/C разработать и отладить CGI-скрипты для обработки данных HTML-формы и доступа к базе данных.
4. После выполнения лабораторной работы привести базу данных в исходное состояние.

Требования к разрабатываемой программе

Разрабатываемые программы должна удовлетворять следующим требованиям:

— разрабатываемое программное приложение должно содержать HTML-документ с формой для ввода данных и CGI-скрипт, вызываемый по окончании работы с HTML-формой;

— CGI-скрипт должен быть написан на языке ESQL/C;

— ввод параметров задания в HTML-форме может быть осуществлен либо путем ввода значений в текстовом виде, либо посредством выбора значений из предлагаемого списка;

— программа должна быть написана в предположении, что любой пользователь без ограничений может иметь доступ к данным;

— в программе должен быть предусмотрен вывод сообщений обо всех шагах ее выполнения, в том числе и о возможных ошибках;

— все действия в отношении базы данных должны выполняться в рамках транзакций;

— программа должна быть достаточно документирована.

Варианты заданий

Вариант 1

1. Вывести информацию о поставщиках, поставивших детали для изделий из указанного города.

2. Увеличить рейтинг поставщика, выполнившего наибольшую поставку некоторой детали, на указанную величину.

Вариант 2

1. Вывести информацию о деталях, поставки которых были осуществлены для указанного изделия.
2. Изменить цвет самой тяжелой детали на указанный.

Вариант 3

1. Вывести информацию о поставщиках, которые осуществляли поставки деталей из заданного города в указанный период.
2. Вставить поставщика с заданными параметрами.

Вариант 4

1. Вывести информацию о поставщиках, поставивших указанную деталь в заданный период.
2. Удалить поставщика, выполнившего меньше всего поставок.

Вариант 5

1. Вывести информацию обо всех деталях, поставляемых для указанного изделия более чем одним поставщиком.
2. В таблице поставок изменить номер поставщика при заданном номере детали и изделия.

Вариант 6

1. Вывести информацию о деталях, поставки которых были осуществлены для указанного изделия всеми поставщиками.
2. Увеличить рейтинг поставщика, выполнившего больший суммарный объем поставок, на указанную величину.

Вариант 7

1. Вывести информацию об изделиях, для которых была поставлена указанная деталь.
2. Изменить название и город детали с максимальным весом на указанные значения.

Вариант 8

1. Вывести информацию о поставщиках, которые осуществляли поставки деталей для указанного изделия.

2. Увеличить рейтинг поставщика, выполнившего большее число поставок, на указанную величину.

Контрольные вопросы

1. Какова общая схема доступа к базам данных посредством CGI-скриптов?
2. Каково назначение пустой строки, генерируемой CGI-скриптов?
3. Каковы основные элементы HTML-формы?
4. Каково назначение элемента action HTML-формы?
5. В каком виде данные, введенные в форме, передаются CGI-модулю?
6. В чем состоит особенность формата данных, передаваемых из HTML-формы CGI-модулю?
7. Какова общая схема работы CGI-скрипта, вводящего данные посредством HTML-формы?
8. В чем разница методов GET и POST?
9. Как в CGI-скрипте задать системные переменные, определяющие параметры сервера базы данных?
10. Каковы способы предоставления всем пользователям одинаковых полномочий на доступ к данным из CGI-скрипта?
11. Каковы способы предоставления пользователям индивидуальных привилегий на доступ к данным из CGI-скрипта?
12. В чем сильные и в чем слабые стороны CGI-технологии?

Список литературы, рекомендуемый к использованию по данной теме:
Основная: 1-11
Дополнительная: 1-11
Интернет-ресурсы: 1-7
Программное обеспечение: 1-4

Практическое занятие №9

Тема 17 - 18. Безопасность объектно-ориентированных баз данных. Язык для взаимодействия с БД

1. Цель работы: ознакомиться с методологиями функционального моделирования работ, диаграмм потоков данных для проектирования информационной системы.

Формируемые компетенции: ПК-1, ПК-3

2. Подготовка к занятию

1. Изучить (повторить) теоретический материал.
2. Ознакомиться с программой работы.
3. Подготовить отчет о работе.
4. Ответить на контрольные вопросы.

3. Распределение времени занятия:

Всего: 90 мин

Вступительная часть 2 мин

Проверка готовности студентов к занятию 5 мин

Основная часть – 70 мин

Проверка выполнения практического занятия 10 мин

Заключительная часть 3 мин

4. Правила работы в лаборатории

К работе в лаборатории допускаются лица, изучившие правила и меры безопасности, сдавшие зачет по ним и усвоившие порядок выполнения лабораторной работы.

4.1. Требования безопасности перед началом работ

- ЗАПРЕЩАЕТСЯ: переодеваться, пользоваться огнем, курить, принимать пищу в лаборатории.

- Убедиться в целостности электрических розеток и разъемов. В лаборатории необходимо быть в сменной обуви.

- Включение компьютера производить только после получения допуска по выполняемой работе и разрешения преподавателя или лаборанта.

4.2. Требования безопасности во время работы

- выполняя практическое занятие, студенты обязаны использовать только вычислительную технику, периферийное оборудование, соединительные кабели, измерительное оборудование и носители информации, непосредственно относящиеся к данному лабораторному занятию;

- подключение и отключение составляющих вычислительного комплекса производить только при полном снятии напряжения со всех составляющих вычислительного комплекса;

- при обнаружении неисправностей в оборудовании немедленно отключить источники питания и доложить об этом руководителю занятий или лаборанту.

4.3. Требования безопасности по окончании работы

- доложить руководителю занятий или лаборанту о завершении работ;

- привести в порядок и сдать рабочее место лаборанту, и доложить руководству

Содержание работы и методические указания к ее выполнению

Разработка базы данных невозможна без ее тщательного проектирования: слишком велико влияние этого шага на последующие этапы жизненного цикла информационной системы, в основе которой лежит создаваемая база данных.

Для успешной реализации базы данных объект проектирования должен быть прежде всего адекватно описан, должны быть построены полные и непротиворечивые функциональные модели базы данных. Опыт проектирования информационных систем показывает, что это логически сложная, трудоемкая и длительная по времени работа, требующая высокой квалификации участвующих в ней специалистов.

При проектировании информационной системы необходимо провести анализ целей этой системы и выявить требования к ней отдельных пользователей. Информация для построения модели информационной системы берется на основе проведения всестороннего обследования организации, для которой выполняется разработка информационной системы. Сбор данных начинается с изучения сущностей предметной области, процессов, использующих эти сущности, и связей между ними

Для целей проектирования информационной системы могут быть использованы следующие виды моделей:

— методология функционального моделирования работ SADT (Structured Analysis and Design Technique);

— диаграммы потоков данных DFD (Data Flow Diagrams);

— методология объектного проектирования на языке UML (UML-диаграммы).

Методология SADT (Structured Analisys and Design Technique - технология структурного анализа и проектирования) разработана Дугласом Т. Россом и является одной из самых известных и широко используемых методик проектирования. Новое название методики, принятое в качестве стандарта, -IDEF0 (Icam DEFinition) является частью программы ICAM (Integrated Computer -Aided Manufacturing - интегрированная компьютеризация производства).

Процесс моделирования в SADT включает сбор информации об исследуемой области, документирование полученной информации, представление ее в виде модели и уточнение модели. Кроме того, этот процесс подсказывает вполне определенный путь выполнения согласованной и достоверной структурной декомпозиции, что является ключевым моментом в квалифицированном анализе системы.

В IDEF0 система представляется как совокупность взаимодействующих работ (или функций). Связи между работами определяют технологический процесс или структуру взаимосвязи внутри организации. Модель SADT представляет собой серию диаграмм, разбивающих сложный объект на составные части.

Основными понятиями методологии функционального моделирования работ являются:

Работы (activity) - поименованные процессы, функции или задачи, которые происходят в течение определенного времени и имеют распознаваемые результаты. На диаграмме работы изображаются прямоугольниками.

Вход (Input) - материал или информация, которые используются работой для получения результата (стрелка, входящая в левую грань).

Управление (Control) - правила, стратегии, стандарты, которыми руководствуется работа (стрелка, входящая в верхнюю грань). В отличие от входной информации управление не подлежит изменению.

Выход (Output) - материал или информация, которые производятся работой (стрелка, исходящая из правой грани). Каждая работа должна иметь хотя бы одну стрелку выхода, так как работа без результата не имеет смысла и не должна моделироваться.

Механизм (Mechanism) - ресурсы, которые выполняют работу (персонал, станки, устройства - стрелка, входящая в нижнюю грань).

Вызов (Call) представляет собой взаимодействие одной модели работ с другой (стрелка, исходящая из нижней грани).

Различают в IDEF0 пять типов связей работ.

Связь по входу (input-output) имеет место, когда выход вышестоящей работы направляется на вход следующей работы.

Связь по управлению (output-control) обозначает ситуацию, когда выход вышестоящей работы направляется на управление следующей работы. Связь показывает доминирование вышестоящей работы.

Обратная связь по входу (output-input feedback) имеет место, когда выход нижестоящей работы направляется на вход вышестоящей. Используется для описания циклов.

Обратная связь по управлению (output-control feedback) обозначает ситуацию, когда выход нижестоящей работы направляется на управление вышестоящей. Является показателем эффективности бизнес-процесса.

Связь выход-механизм (output-mechanism) имеет место, когда выход одной работы направляется на механизм другой и показывает, что работа подготавливает ресурсы для проведения другой работы.

Диаграммы потоков данных (Data Flow Diagrams - DFD) используются для описания движения документов и обработки информации как дополнение к IDEF0. В отличие от IDEF0, где система рассматривается как взаимосвязанные работы, стрелки в DFD показывают лишь то, как объекты (включая данные) движутся от одной работы к другой.

Диаграмма потоков данных содержит:

— процессы, которые преобразуют данные;

— потоки данных, переносящие данные;

— активные объекты, которые производят и потребляют данные;

— хранилища данных, которые пассивно хранят данные.

Процесс DFD преобразует значения данных и изображается в виде эллипса, внутри которого помещается имя процесса.

Поток данных соединяет выход объекта (или процесса) с входом другого объекта (или процесса) и представляет собой промежуточные данные вычислений. Поток данных изображается в виде стрелки между производителем и потребителем данных, помеченной именами соответствующих данных. Дуги могут разветвляться или сливаться, что означает соответственно разделение потока данных на части либо слияние объектов.

Активным объектом является объект, который обеспечивает движение данных, поставляя или потребляя их. Хранилище данных - это пассивный объект в составе DFD, в котором данные сохраняются для последующего доступа.

Функция, принимающая решение о запуске процесса, будучи включенной в DFD, порождает в диаграмме поток управления и изображается пунктирной стрелкой.

Из перечисленных блоков строятся диаграммы работ и диаграммы потоков данных, описывающие принципы функционирования системы.

Последовательность выполнения лабораторной работы:

1. Ознакомиться с предложенным вариантом описания предметной области. Проанализировать предметную область, уточнив и дополнив ее, руководствуясь собственным опытом, консультациями и другими источниками.
2. Выполнить структурное разбиение предметной области на отдельные под-разделения (отделы, службы, подсистемы, группы и пр.) согласно выполняемым ими функциям.
3. Определить задачи и функции системы в целом и функции каждого под-разделения (подсистемы).
4. Выполнить словесное описание работы каждого подразделения (подсис-темы), алгоритмов и сценариев выполнения ими отдельных работ.
5. Построить диаграммы работ и диаграммы потоков данных для всей ин-формационной системы в целом и для отдельных сценариев работ, отражающие логику и взаимоотношение подразделений (подсистем).
6. Оформить следующие разделы отчета:
   - исходное задание;
   - состав подразделений (подсистем) информационной системы;
   - перечень функций и задач системы в целом и каждого подразделения (под-системы) в отдельности; подробное описание работы каждого подразделения (подсистемы), отношения их между собой, описание отдельных сценариев работ;
   - диаграммы работ и диаграммы потоков данных для всей информационной системы в целом и для входящих в нее подразделений (подсистем).

Контрольные вопросы

1. Каковы задачи методологии структурного анализа данных?
2. Каковы виды связей в методологии IDEF0.
3. Каково назначение методологии диаграмм потоков данных?
4. Что такое поток данных в методологии DFD?
5. Какова функция хранилища данных в DFD?
6. В чем сходство и в чем различие методологии структурного анализа дан-ных и диаграмм потоков данных?

Список литературы, рекомендуемый к использованию по данной теме:
Основная: 1-11
Дополнительная: 1-11
Интернет-ресурсы: 1-7
Программное обеспечение: 1-4

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

для обучающихся по организации и проведению самостоятельной работы
по дисциплине «Безопасность баз данных»
для студентов направления подготовки
10.03.01«Информационная безопасность»
Направленность (профиль):
«Организация и технологии защиты информации (по отрасли или в сфере профессиональной деятельности)»

Ставрополь,
2026

186

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель дисциплины “Безопасность баз данных” заключается в сформировании у обучающихся понимания основ информационной безопасности систем баз данных для последующего практического использования в науке и образовании.

Задачами дисциплины являются:

1. Теоретическая и практическая подготовка бакалавров к настройке параметров безопасности баз данных;
2. Объяснение принципов построения подсистем защиты в СУБД, средств и методов несанкционированного доступа к ресурсам СУБД;
3. Привитие студентам системного подхода к проблеме защиты информации в СУБД; механизмов защиты информации и возможностей по их преодолению.

1. Общая характеристика самостоятельной работы студента

Самостоятельная работа включает те разделы курса, которые не получили достаточного освещения на лекциях по причине ограниченности лекционного времени и большого объема изучаемого материала.

Методическое обеспечение самостоятельной работы по дисциплине состоит из:

1) Определения учебных вопросов, которые студенты должны изучить самостоятельно;
2) Подбора необходимой учебной литературы, обязательной для проработки и изучения;
3) Поиска дополнительной научной литературы, к которой студенты могут обращаться по желанию, если у них возникает интерес в данной теме;
4) Определения контрольных вопросов, позволяющих студентам самостоятельно проверить качество полученных знаний;
5) Организации консультаций преподавателя со студентами для разъяснения вопросов, вызвавших у обучающихся затруднения при самостоятельном освоении учебного материала.

Самостоятельная работа студента – это особым образом организованная деятельность, включающая в свою структуру такие компоненты, как:

- уяснение цели и поставленной учебной задачи;
- четкое и системное планирование самостоятельной работы;
- поиск необходимой учебной и научной информации;
- освоение собственной информации и ее логическая переработка;
- использование методов исследовательской, научно-исследовательской работы для решения поставленных задач;
- выработка собственной позиции по поводу полученной задачи;
- представление, обоснование и защита полученного решения;
- проведение самоанализа и самоконтроля

К основным видам самостоятельной работы студентов относится:

- работа с понятийным аппаратом;
- конспектирование первоисточников;
- проектирование фрагментов исследовательской деятельности;
- анализ научных исследований по психологической проблематике;
- подготовка научного проекта по теме.

Контроль знаний студентов включает формы текущего и итогового контроля. Текущий контроль осуществляется в процессе изучения курса и включает в себя оценку работы студентов на практических занятиях (круглый стол, собеседование, групповые научные проекты, тестирование), а также подготовку домашнего задания.

2. Технологическая карта самостоятельной работы студента

Коды реализуемых копетенций	Вид деятельности студентов	Итоговый продукт самостоятельной работы	Средства и технологии оценки	Обьем часов, в том числе		
				СРС	Контактная работа с преподавателем	Всего
5 семестр						
ИД-1 ПК-1 ИД-2 ПК-1 ИД-1 ПК-3	Подготовка к лекции	Конспект лекций	Собеседование	4.00	2.00	6.00
ИД-1 ПК-1 ИД-2 ПК-1 ИД-1 ПК-3	Подготовка к лабораторному занятию	Отчет	Собеседования	4.00	2.00	6.00
ИД-1 ПК-1 ИД-2 ПК-1 ИД-1 ПК-3	Подготовка к экзамену	Экзамен	Вопросы к экзамену	4.00	2.00	6.00
Итого за 5 семестр				12.00	6.00	18.00
Итого				12.00	6.00	18.00

3. Порядок выполнения самостоятельной работы

3.1 Методические рекомендации по изучению теоретического материала

Комплексное изучение предлагаемой студентам учебной дисциплины «Безопасность баз данных» предполагает овладение материалами лекций, учебников, программы, творческую работу студентов в ходе проведения лабораторных занятий, а также систематическое выполнение заданий для самостоятельной работы.

В ходе лекций раскрываются основные вопросы в рамках рассматриваемых тем, делаются акценты на наиболее сложные и интересные положения изучаемого материала, которые должны быть приняты студентами во внимание. Материалы лекций являются основой для подготовки студентов к лабораторным занятиям.

Работа с конспектом лекций

Просмотрите конспект сразу после занятий. Отметьте материал конспекта лекций, который вызывает затруднения для понимания. Попытайтесь найти ответы на затруднительные вопросы, используя предлагаемую литературу. Если самостоятельно не

удалось разобраться в материале, сформулируйте вопросы и обратитесь на текущей консультации или на ближайшей лекции за помощью к преподавателю.

Каждую неделю отводите время для повторения пройденного материала, проверяя свои знания, умения и навыки по контрольным вопросам и тестам.

Выполнение лабораторных работ по дисциплине «Безопасность баз данных»

На первом занятии получите у преподавателя задания по курсу, планы подготовки к лабораторным занятиям. Обзаведитесь всем необходимым методическим обеспечением.

Перед лабораторными занятиями изучите теорию вопроса, предполагаемого к исследованию, ознакомьтесь с опытом других исследователей в этом направлении.

Целью лабораторных работ является:

- закрепление методов приложения теории к решению практических задач анализа и синтеза психологического знания;
- проверка уровня понимания студентами вопросов, рассмотренных на лекциях и по учебной литературе, степени и качества усвоения материала;
- обучение навыкам освоения методики эксперимента и работы с нормативно-справочной литературой;
- восполнение пробелов в пройденной теоретической части курса и оказание помощи в его усвоении.

3.2 Методические рекомендации по конспектированию первоисточников

Изучение и анализ литературных источников является обязательным видом самостоятельной работы студентов. Изучение литературы по избранной теме имеет своей задачей проследить характер постановки и решения определенной проблемы различными авторами, аргументацию их выводов и обобщений, провести анализ и систематизировать полученный материал на основе собственного осмысления с целью выяснения современного состояния вопроса.

Проработка отобранного материала обязательно должна идти с одновременным ведением записей прочитанного и своих замечаний. Запись может иметь как форму конспекта, так и выписок, а также картотеку положений, тезисов, идей, методик, что в дальнейшем облегчит классификацию и систематизацию полученного материала. Такого рода записи являются лучшим способом накопления и первичной обработки материал, одной из обязательных форм организации умственного труда.

Планом удобно пользоваться при подготовке к устному выступлению по выбранной теме. Каждый пункт плана должен раскрывать одну из сторон избранной темы, а весь план должен охватывать ее целиком.

Тезисы предполагают сжатое изложение основных положений текста в форме утверждения или отрицания. Они являются более совершенной формой записей и представляют основу для дискуссии. К тому же их легко запомнить.

Конспектирование – процесс мысленной переработки и письменной фиксации информации, в виде краткого изложения основного содержания, смысла какого-либо текста.

Результат конспектирования – запись, позволяющая конспектирующему немедленно или через некоторый срок с нужной полнотой восстановить полученную информацию. Конспект в переводе с латыни означает «обзор». По существу его и составлять надо как обзор, содержащий основные мысли текста без подробностей и второстепенных деталей. Конспект носит индивидуализированный характер: он рассчитан на самого автора и поэтому может оказаться малопонятным для других.

Для того чтобы осуществлять этот вид работы, в каждом конкретном случае необходимо грамотно решить следующие задачи:

1. Сориентироваться в общей композиции текста (уметь определить вступление, основную часть, заключение).
2. Увидеть логико-смысловую канву сообщения, понять систему изложения автором информации в целом, а также ход развития каждой отдельной мысли.
3. Выявить «ключевые» мысли, т. е. основные смысловые вехи, на которые «нанизано» все содержание текста.
4. Определить детализирующую информацию.
5. Лаконично сформулировать основную информацию, не перенося на письмо все целиком и дословно.

Во всяком научном тексте содержится информация 2-х видов: основная и вспомогательная. Основной является информация, имеющая наиболее существенное значение для раскрытия содержания темы или вопроса. К ней относятся: определения научных понятий, формулировки законов, теоретических принципов и т. д. Назначение вспомогательной информации – помочь читателю лучше усвоить предлагаемый материал. К этому типу информации относятся разного рода комментарии. Основную – записываем как можно полнее, вспомогательную, как правило, опускаем. Содержание конспектирования составляет переработка основной информации в целях ее обобщения и сокращения. Обобщить – значит представить ее в более общей, схематической форме, в виде тезисов, выводов, отдельных заголовков, изложения основных результатов и т. п. Читая, мы интуитивно используем некоторые слова и фразы в качестве опорных. Такие опорные слова и фразы называются ключевыми. Ключевые слова и фразы несут основную смысловую и эмоциональную нагрузку содержания текста.

Выбор ключевых слов – это первый этап смыслового свертывания, смыслового сжатия материала.
Важными требованиями к конспекту являются наглядность и обозримость записей и такое их расположение, которое давало бы возможность уяснить логические связи и иерархию понятий.
По форме *конспекты подразделяются* на формализованные и графические.
1. Формализованные (все записи вносятся в заранее подготовленные таблицы).
Это удобно, во-первых, при конспектировании материалов, когда перечень характеристик описываемых предметов или явлений более или менее постоянен, во-вторых, при подготовке единого конспекта по нескольким источникам. Особенно если есть необходимость сравнения отдельных данных. Разновидностью формализованного конспекта является запись, составленная в форме ответов на заранее подготовленные вопросы, обеспечивающие исчерпывающие характеристики однотипных предметов или явлений.
2. Графические (элементы конспектируемой работы располагаются в таком виде, при котором видна иерархия понятий и взаимосвязь между ними).
По каждой работе может быть не один, а несколько графических конспектов, отображающих книгу в целом и отдельные ее части. Ведение графического конспекта - наиболее совершенный способ изображения внутренней структуры книги, а сам этот процесс помогает усвоению ее содержания.
Можно выделить следующие основные **типы конспектов: плановый, текстуальный, сводный, тематический.**
Плановый – легко получить с помощью предварительно сделанного плана произведения, каждому вопросу плана отвечает определенная часть конспекта:
а) вопросно-ответный (на пункты плана, выраженные в вопросительной форме, конспект дает точные ответы);
б) схематичный плановый конспект (отражает логическую структуру и взаимосвязь отдельных положений).

Текстуальный – это конспект, созданный в основном из цитат.
Сводный конспект – сочетает выписки, цитаты, иногда тезисы; часть его текста может быть снабжена планом.
Тематический – дает более или менее исчерпывающий ответ (в зависимости из числа привлеченных источников и другого материала, например, своих же записей) на поставленный вопрос – тему: обзорный; хронологический.
Роль конспекта – чисто учебная: он помогает зафиксировать основные понятия и положения первичного текста и в нужный момент их воспроизвести, например, при написании реферата или подготовке к экзамену.
Способы конспектирования.
- *Тезисы* – это кратко сформулированные основные мысли, положения изучаемого материала. Тезисы лаконично выражают суть читаемого, дают возможность раскрыть содержание. Приступая к освоению записи в виде тезисов, полезно в самом тексте отмечать места, наиболее четко формулирующие основную мысль, которую автор доказывает (если, конечно, это не библиотечная книга). Часто такой отбор облегчается шрифтовым выделением, сделанным в самом тексте.
- *Линейно-последовательная запись текста.* При конспектировании линейно – последовательным способом целесообразно использование плакатно-оформительских средств, которые включают в себя следующие:
· сдвиг текста конспекта по горизонтали, по вертикали;
· выделение жирным (или другим) шрифтом особо значимых слов;
· использование различных цветов;
· подчеркивание;
· заключение в рамку главной информации.
- С***пособ «вопросов – ответов».*** Он заключается в том, что, поделив страницу тетради пополам вертикальной чертой, конспектирующий в левой части страницы самостоятельно формулирует вопросы или проблемы, затронутые в данном тексте, а в правой части дает ответы на них.
Одна из модификаций способа «вопросов – ответов» - таблица, где место вопроса занимает формулировка проблемы, поднятой автором (лектором), а место ответа – решение данной проблемы. Иногда в таблице могут появиться и дополнительные графы: например, «мое мнение» и т. п.
- С***хема с фрагментами –*** способ конспектирования, позволяющий ярче выявить структуру текста, - при этом фрагменты текста (опорные слова, словосочетания, пояснения всякого рода) в сочетании с графикой помогают созданию рационально - лаконичного конспекта.
- ***Простая схема –*** способ конспектирования, близкий к схеме с фрагментами, объяснений к которой конспектирующий не пишет, но должен уметь давать их устно. Этот способ требует высокой квалификации конспектирующего. В противном случае такой конспект нельзя будет использовать.
- ***Параллельный способ*** конспектирования. Конспект оформляется на двух листах параллельно или один лист делится вертикальной чертой пополам и записи делаются в правой и в левой части листа.

Однако лучше использовать разные способы конспектирования для записи одного и того же материала.

Комбинированный конспект – вершина овладения рациональным конспектированием. При этом умело используются все перечисленные способы, сочетая их в одном конспекте (один из видов конспекта свободно перетекает в другой в зависимости от конспектируемого текста, от желания и умения конспектирующего). Именно при комбинированном конспекте более всего проявляется уровень подготовки и индивидуальность студента.

Принципы составления конспекта прочитанного.

1. Записать все выходные данные источника: автор, название, год и место издания. Если текст взят из периодического издания (газеты или журнала), то записать его название, год, месяц, номер, число, место издания.

2. Выделить поля слева или справа, можно с обеих сторон. Слева на полях отмечаются страницы оригинала, структурные разделы статьи или книги (названия параграфов, подзаголовки и т. п.), формулируются основные проблемы. Справа – способы фиксации прочитанной информации.

Один из видов чтения – углубленное – предполагает глубокое усвоение прочитанного и часто сохранение информации в целях последующего обращения к ней. Эффективность такого чтения повышается, если прочитанное зафиксировано не только в памяти, но и на бумаге. Психологи утверждают, что записанное лучше и полнее усваивается, прочнее откладывается в памяти. Установлено, что если прочитать 1000 слов и затем записать 50, подытоживающих прочитанное, то коэффициент усвоения будет выше, чем, если прочитать 10000 слов, не записав ни одного. Кроме того, при записи прочитанного формируется навык свертывания информации. И, наконец, чередование чтения и записывания уменьшает усталость, повышает работоспособность и производительность умственного труда.

1. Резюмирование.

Резюме – краткий итог прочитанного, содержащий его оценку. Резюме характеризует основные выводы книги, главные итоги.

Выбор языковых средств для построения резюме-выводов подчинен основной задаче свертывания информации: минимум языковых средств – максимум информации. Это обычно одно – три четких, кратких, выразительных предложения, раскрывающих, по мнению автора, самую суть описываемого объекта.

2. Аннотация.

Аннотация – краткая обобщенная характеристика печатной работы (книги, статьи), включающая иногда и его оценку. Это наикратчайшее изложение содержания первичного документа, дающее общее представление о теме.

Основное ее назначение – дать некоторое представление о книге (статье, научной работе) с тем, чтобы рекомендовать ее определенному кругу читателей или воспользоваться своими записями при выполнении работы исследовательского, реферативного характера. Поэтому аннотации не требуется изложения содержания произведения, в ней лишь перечисляются вопросы, которые освещены в первоисточнике (содержание этих вопросов не раскрывается). Аннотация отвечает на вопрос: «О чем говорится в первичном тексте?», дает представление только о главной теме и перечне вопросов, затрагиваемых в тексте первоисточника.

Текст аннотации не стандартизирован. В научной литературе можно встретить различные требования к составлению аннотаций. Например, текст справочной аннотации может включать следующие сведения:

· тип и название аннотируемого документа (монография, диссертация, сборник, статья и т. п.)

· задачи, поставленные автором аннотируемого документа

· метод, которым пользовался автор (эксперимент, сравнительный анализ, компиляция других источников)
· принадлежность автора к определенной научной школе или направлению
· структуру аннотируемого документа
· предмет и тему произведения, основные положения и выводы автора
· характеристику вспомогательных иллюстративных материалов, дополнений, приложений, справочного аппарата, включая указатели и библиографию.

3.3 Методические указания по подготовке к экзамену

Контроль и оценка знаний студентов является неотъемлемой составной частью учебно-воспитательного процесса в вузе. Экзаменационная сессия – самая ответственная пора в студенческой жизни: она является итогом большой работы студентов в семестре. Экзамен – это метод проверки знаний студентов по части или полному курсу учебной дисциплины, произведенный путем постановки устных или письменных вопросов. Он дает объективную официально фиксируемую оценку успехов студентов за определенный отрезок времени.

Экзамен преследует многогранную цель: во-первых, это – проверка знаний студента, во-вторых, они сами по себе являются важным звеном в овладении наукой, в-третьих, это продолжение учебного процесса; наконец, он имеют большое значение как фактор стимулирования глубокого изучения предмета.

Подготовка к экзамену состоит из двух взаимосвязанных этапов. Первый – систематический труд на протяжении семестра, учебного года, охватывающий все формы учебного процесса: лекции, изучение и конспектирование рекомендованной литературы, активное участие в семинарских (практических) и лабораторных занятиях.

Второй – подготовка непосредственно перед экзаменом. Она позволяет студентам за сравнительно короткий отрезок времени охватить всю перспективу изученного и лучше понять основные закономерности и явления.

Ограниченность времени (3-4 дня) для непосредственной подготовки к экзамену по предмету требует от студентов спокойно, без нервозной суеты и спешки еще раз внимательно продумать изученный в течение семестра материал, тщательно отработать вопросы, недостаточно изученные или плохо понятые, с тем, чтобы по возможности устранить все пробелы в своих знаниях.

Готовиться надо по строго продуманному графику, последовательно переходя от темы к теме, не пропуская ни одну из них.

Сложные вопросы, недостаточно уясненные в процессе подготовки к экзамену, необходимо записать и получить на них разъяснения у преподавателей во время предэкзаменационных консультаций.

Следует заметить, что студенты, которые не ходят на консультации из-за отсутствия, по их мнению, сложных вопросов, поступают опрометчиво. На консультациях очень часто лектор не только отвечает на заданные вопросы, но и по собственной инициативе разъясняет наиболее трудные разделы курса.

Итак, основной задачей подготовки студентов к экзамену является систематизация знаний учебного материала, его творческое осмысливание. Важнейшим учебным пособием на этом этапе работы студента является собственный конспект прослушанных лекций и самостоятельно проработанных тем курса.

В соответствии с положением об экзаменах их, как правило, принимают преподаватели, читающие курс лекций.

Получив билет, студент должен хорошо продумать содержание поставленных вопросов. Значительное число неудачных ответов объясняется неясным пониманием поставленной проблемы. Правильное понимание вопроса обеспечит успех при ответе на

него. При подготовке к ответу на билет нужно составить развернутый план по каждому вопросу

Отвечая на вопросы экзаменационного билета, не следует спешить. Надо излагать материал спокойно, не торопясь, владеть собой, следить за построением фраз. Следует избегать подходов издалека, общих рассуждений. Рекомендуется строить ответы четко, последовательно, конкретно, по возможности исчерпывающе. Вместе с тем весьма желательно быстро и правильно иллюстрировать свой ответ примерами.

От экзаменующегося требуется: определение понятий, обоснование выдвинутых положений, свободное оперирование фактическим материалом, дать альтернативные подходы по отдельным проблемам. Логичность, стройность, литературная грамотность изложения являются неотъемлемыми чертами полноценного ответа. Нельзя при ответе допускать ни излишней краткости, переходящей в схематизм, ни многословия. И то, и другое не оправдано. Краткость не дает преподавателю возможности понять, владеет ли студент учебным материалом, а многословие может показать, что студент не умеет акцентировать внимание на главном и говорит слишком расплывчато.

Вопросы для экзамена

Знать:

1. Понятия и определения реляционной модели
2. Проектирование реляционных баз данных
3. Манипулирование реляционными базами данных. Реляционная алгебра
4. Некоторые особенности логической архитектуры современных реляционных баз данных.
5. Клиент-серверная архитектура современных реляционных СУБД и АИС
6. Технологии и модели клиент-серверной архитектуры
7. СУБД Microsoft SQL Server
8. Теоретические основы безопасности БД и СУБД
9. Модели данных. Классификация моделей данных.
10. Модель «сущность-связь». Основные понятия. Область применения.
11. Иерархическая модель данных. Основные понятия. Область применения. Достоинства и недостатки.
12. Сетевая модель данных. Основные понятия. Область применения. Достоинства и недостатки.
13. Реляционная модель данных. Основные понятия. Область применения. Достоинства и недостатки.
14. Операции реляционной алгебры.
15. Реляционное исчисление с переменными-кортежами.
16. Реляционное исчисление с переменными на доменах.
17. 12. Функциональные зависимости. Аксиомы. Правила вывода функциональных зависимостей.
18. Избыточные функциональные зависимости. Минимальное покрытие. Декомпозиция отношений.
19. Нормальные формы схем отношений. Первая нормальная форма. Вторая нормальная форма.
20. Нормальные формы схем отношений. Третья нормальная форма.
21. Нормальные формы схем отношений. Нормальная форма Бойса-Кодда.
22. Многозначные зависимости. Аксиомы многозначных зависимостей.
23. Нормальные формы схем отношений. Четвертая нормальная форма.
24. Нормальные формы схем отношений. Пятая нормальная форма.
25. Структурированный язык запросов SQL. Операторы манипулирования данными. Курсор.

26. Структурированный язык запросов SQL. Типы связывания.
27. Структурированный язык запросов SQL. Многотабличные запросы.
28. Структурированный язык запросов SQL. Операции изменения и обновления базы данных.
29. Структурированный язык запросов SQL. Индексы.
30. Структурированный язык запросов SQL. Определение пользовательских представлений.
31. Структурированный язык запросов SQL. Использование UNION для объединения результатов инструкций SELECT.
32. Структурированный язык запросов SQL. Формирование запросов.
33. Структурированный язык запросов SQL. Использование псевдонимов.
34. Три уровня представления данных в автоматизированных информационных системах.
35. Логическая и физическая независимость данных.
36. Основные функции СУБД.
37. Виды аномалий в базе данных.
38. Обобщенный алгоритм декомпозиции.
39. Правила преобразования ER - модели в реляционную модель данных.
40. Файловые структуры, используемые для хранения информации в базах данных.

Уметь, Владеть:

Задачи для проверки умений и навыков

1. Хранимые процедуры. Назначение. Правила создания.
2. Основные требования к средствам реализации систем оперативной и аналитической обработки данных.
3. Многомерная модель данных.
4. Гиперкубические и поликубические модели данных.

Повышенный уровень:

Знать:

1. Программа ISS Database Scanner
2. Мониторинг активности пользователей на уровне СУБД
3. Организация местного аудита в базах данных с использованием триггеров
4. Распределенные базы данных
5. Понятия распределенных БД и СУБД
6. Компонентная архитектура СУРБД
7. Распределенные транзакции
8. Репликация данных
9. Методы и механизмы обеспечения целостности информации в реляционных базах данных
10. Обработка транзакций
11. Управление параллельностью работы транзакций
12. Реализация ограничений в базах данных
13. Методы и механизмы обеспечения конфиденциальности информации в системах баз данных

Уметь, Владеть:

Задачи для проверки умений и навыков

1. Постреляционная модель данных.
2. Объектно-ориентированные СУБД.
3. Понятие безопасности БД.
4. Угрозы безопасности БД
5. Меры защиты БД и СУБД

4.2 Вопросы для собеседования по самостоятельному изучению дисциплины

6 семестр:

Раздел 1. Основные понятия и определения изучаемой дисциплины.

1. Правила преобразования ER - модели в реляционную модель данных.
2. Файловые структуры, используемые для хранения информации в базах данных.
3. Файлы прямого доступа.
4. Файлы последовательного доступа.
5. Хэширование. Стратегия разрешения коллизий.
6. Файлы с плотным индексом. Пример организации файла.
7. Файлы с неплотным индексом. Пример организации файла.
8. Организация индексов в виде B-деревьев.

4.3 Комплект разноуровневых задач и заданий

Базовый уровень:

1 Задачи репродуктивного уровня

1. Файлы прямого доступа.
2. Файлы последовательного доступа.
3. Хэширование. Стратегия разрешения коллизий.

Повышенный уровень:

4. Файлы с плотным индексом. Пример организации файла.
5. Файлы с неплотным индексом. Пример организации файла.

2 Задачи базового уровня

Задание 4-7.

6. Организация индексов в виде B-деревьев.
7. Моделирование отношений «один-ко-многим» с использованием одно-направленных указателей.
8. Инвертированные списки.
9. Модели «клиент-сервер» в технологии баз данных.

Задание 8.

10. Модель файлового сервера. Достоинства и недостатки.
11. Модель удаленного доступа к данным. Достоинства и недостатки.
12. Модель сервера баз данных. Достоинства и недостатки.
13. Модель сервера приложений. Достоинства и недостатки.

3 Задачи повышенного уровня

Задание 15.

14. Транзакции. Свойства транзакций. Способы завершений транзакций.
15. Транзакции. Журнал транзакций.
16. Транзакции. Типы синхронизационных захватов. Правила применения.
17. Триггеры. Назначение. Правила создания архивных документов».

4.4 Вопросы для собеседования по практическим занятиям

1. Тема 5. Подходы к организации баз данных

Типовые задачи (всего 11 заданий):

Задания базового уровня:

— Перечислить основные модели данных.
— Пояснить классификацию моделей данных.
— Пояснить принцип модели «сущность-связь».
— Дать понятие иерархической модели данных.
— Перечислить достоинства и недостатки иерархической модели данных.
— Дать понятие сетевой модели данных.
— Перечислить достоинства и недостатки сетевой модели данных.
— Дать понятие реляционной модель данных.
— Перечислить достоинства и недостатки реляционной модель данных.

Задания повышенного уровня:

— Используя командное окно и конструктор форм FoxPro, создать форму для работы с одной таблицей (просмотр, редактирование, добавление, удаление данных) и запустить ее на выполнение.

— Используя командное окно и конструктор БД создать базу данных, включающую две таблицы и одно представление (одну таблицу добавить в БД, вторую таблицу создать с учетом дополнительных возможностей - задать триггеры, индексы и т.д.)

Тема 4. Общие понятия реляционного подхода к организации БД

Типовые задачи (всего 8 заданий):

Задания базового уровня:

— Перечислить операции реляционной алгебры.
— Пояснить принцип реляционного исчисления с переменными-кортежами.
— Пояснить принцип реляционного исчисления с переменными на доменах.
— Пояснить понятие функциональных зависимостей.
— Пояснить правила вывода функциональных зависимостей.
— Пояснить понятие избыточных функциональных зависимостей.

Задания повышенного уровня:

— Отобразить переменные среды в Windows двумя способами: из командной оболочки и окна свойств системы

— Задать переменную среды с различными вариантами динамически формируемых значений.

3. Тема 5. Базисные средства манипулирования реляционными данными.

Типовые задачи (всего 9 заданий):

Задания базового уровня:

— Пояснить, каким способом возможен запуск серверной части СУБД MySQL.

— Дать определение привилегии и пояснить её предназначение.

— Перечислить основные утилиты, входящие в состав СУБД, какие функции они выполняют.

— Перечислить принципы использования кортежных переменных

— Дать понятие минимального покрытия и декомпозиции отношений.

— Перечислить принципы использования переменных в доменах

Задания повышенного уровня:

— Запустите сервер MySQL. Зарегистрируйте своего пользователя в консольном приложении, задайте ему права.

— С помощью утилиты Mysqlshow выполните команду на просмотр структуры и состав таблиц базы Mysql.

— С помощью утилиты Mysqldump получите полный дамп базы Mysql (данные и таблицы), а также отдельные дампы таблиц и данных.

4. Тема 6. Проектирование реляционных БД

Типовые задачи (всего 9 заданий):

Задания базового уровня:

— Пояснить, в чем состоит различие логического и физического уровней представления моделей данных с помощью ERwin?

— Пояснить, в чем различие между моделями данных, представленных в форме диаграммы сущность-связь и на основе ключей

— Пояснить, в чем различие между моделями данных, представленных в форме диаграммы сущность-связь и в виде полной атрибутивной модели.

— Пояснить, какие основные компоненты содержат модели данных, представленные по методологии IDEF1X

— Перечислить и пояснить правила преобразования ER - модели в реляционную модель данных.

Задания повышенного уровня:

— Выполнить построение диаграммы с заданными сущностями (прямое моделирование) для заданной предметной области.

— Задать атрибуты для каждой определенной сущности. При задании атрибутов использовать домены.

— Ввести связи между сущностями и присвоить связям уникальные имена.

— Используя СУБД MYSQL, решить прямую генерацию базы данных.

5. Тема 8. Управление параллельностью работы транзакций.

Типовые задачи (всего 12 заданий):

Задания базового уровня:

— Пояснить, в каких режимах в MySQL возможно создание базы данных.

— Перечислить допустимые типы данных при создании таблицы.

— Пояснить, как разделяются операторы SQL в случае нескольких операторов в запросе.

— Пояснить, каким образом выполнить простейшие операции вставки строк данных в таблицу средствами SQL.

— Пояснить, каким образом выполнить простейшие операции модификации строк таблицы средствами SQL.

— Получить информацию о структуре таблицы в рамках СУБД MySQL.

— Перечислить существующие ограничения на формирование коррелированного запроса.

— Какими средствами SQL реализуются следующие операции реляционной алгебры: ограничение, декартово произведение, проекция, пересечение, объединение, разность, соединение?

Задания повышенного уровня:

— Выполнить создание таблицы средствами СУБД

— Выполнить создание таблицы средствами языка SQL

— Подготовить и выполнить средствами СУБД MySQL 2 запроса по модификации информации (вставка, удаление, замещение) из таблиц базы данных для решения нижеприведенных задач.

— Подготовить и выполнить средствами СУБД MySQL 4 запроса по выборке информации из таблиц базы данных с использованием агрегатных функций.

6. Тема 9. Тема 9. Методы сериализации транзакций

Типовые задачи (всего 7 заданий):

Задания базового уровня:

— Пояснить, каким образом предоставляются права на пользование базой данных и отдельными ее таблицами

— Пояснить, каким образом изымаются права на пользование базой данных и отдельными ее таблицами.

— Пояснить понятие внешней базы данных.

— Пояснить, как идентифицируется таблица внешней базы данных

— Пояснить, как идентифицируется таблица внешней распределенной базы данных.

Задания повышенного уровня:

— Обеспечьте, чтобы владелец внешней используемой базы данных предоставил полномочия на модификацию данных из используемых таблиц в его базе данных, дав соответственно ему такие же полномочия для выполнения аналогичных действий.

— Обеспечьте, чтобы владелец используемой внешней базы данных предоставил полномочия на удаление из используемых таблиц в его базе данных, дав соответственно ему такие же полномочия для выполнения аналогичных действий

7. Тема 11. Язык SQL. Функции и основные возможности

Типовые задачи (всего 10 заданий):

Задания базового уровня:

— Перечислить особенности использования динамических операторов SQL

— Дать понятие динамическим главным переменным, для чего они используются.

— Перечислить операторы, с которыми связано использование динамических главных переменных.

— Перечислите основные компоненты ODBC-программы.

— Пояснить, с помощью каких средств ODBC можно отследить наличие ошибки.

— Пояснить, в каких случаях непосредственное выполнение операторов является наиболее эффективным.

— Пояснить, как описываются маркеры параметров, и какая для этого предусмотрена функция.

— Пояснить, с помощью какого параметра можно освободить буферы всех столбцов.

Задания повышенного уровня:

— Разработать ODBC-программу для решения задачи из соответствующего варианта с помощью функций SQLBindCol(), SQLFetch().

— Разработать ODBC-программу для решения задачи из соответствующего варианта с помощью функций SQLFetch(), SQLGetData().

8. Тема 12. Средства манипулирования данными в SQL.

Типовые задачи (всего 8 заданий):

Задания базового уровня:

— Пояснить, в каком виде данные, введенные в форме, передаются CGI-модулю

— Пояснить, в чем состоит особенность формата данных, передаваемых из HTML-формы CGI-модулю

— Пояснить общую схема работы CGI-скрипта, вводящего данные посредством HTML-формы.

— Перечислить способы предоставления всем пользователям одинаковых полномочий на доступ к данным из CGI-скрипта

— Перечислить способы предоставления пользователям индивидуальных привилегий на доступ к данным из CGI-скрипта.

— Перечислить сильные и слабые стороны CGI-технологии

Задания повышенного уровня:

— С помощью CGI-скрипта задать системные переменные, определяющие параметры сервера базы данных.

— . Средствами языка ESQL/C разработать и отладить CGI-скрипты для обработки данных HTML-формы и доступа к базе данных

9. Тема 13. Безопасность SQL при прикладном программировании.

Типовые задачи (всего 8 заданий):

Задания базового уровня:

— Пояснить, каким образом вставить конструкции PHP в HTML-документ.

— Каким образом обеспечить, чтобы встречающиеся в строке переменные были заменены их значениями?

— Перечислить правила определения функций в языке PHP.

— Перечислить особенности передачи значений переменных из HTML-формы в переменные PHP.

— Пояснить, каким образом осуществляется взаимодействие с базами данных в языке PHP.

— Пояснить, в чем заключается технология "cookies".

Задания повышенного уровня:

— Разработать и отладить HTML-формы для ввода данных пользователя согласно варианту задания (можно модифицировать HTML-формы из предыдущей лабораторной работы).

— Разработать и отладить PHP-программы для обработки данных HTML-форм и доступа к базе данных.

9. Тема 14. Компиляторы SQL и проблемы безопасности оптимизации.

Типовые задачи (всего 8 заданий):

Задания базового уровня:

— Пояснить задачи методологии структурного анализа данных.

— Перечислить виды связей в методологии IDEF0.

— Пояснить назначение методологии диаграмм потоков данных.

— Дать понятие потока данных в методологии DFD.

— Пояснить функцию хранилища данных в DFD.

— Пояснить, в чем сходство и в чем различие методологии структурного анализа дан-ных и диаграмм потоков данных.

Задания повышенного уровня:

— Построить диаграммы работ и диаграммы потоков данных для всей информационной системы в целом и для отдельных сценариев работ, отражающие логику и взаимоотношение подразделений (подсистем).

— Выполнить структурное разбиение предметной области на отдельные подразделения (отделы, службы, подсистемы, группы и пр.) согласно выполняемым ими функциям.

9. Тема 15. Безопасность архитектуры «клиент-сервер».

Типовые задачи (всего 8 заданий):

Задания базового уровня:

— Пояснить особенности модели «клиент-сервер» в технологии баз данных.

— Пояснить особенности модели файлового сервера. Достоинства и недостатки.

— Пояснить особенности модели удаленного доступа к данным. Достоинства и недостатки.

— Пояснить особенности модели сервера баз данных. Достоинства и недостатки.

— Пояснить особенности модели сервера приложений. Достоинства и недостатки.

— Перечислить свойства транзакций и способы завершений транзакций.

Задания повышенного уровня:

— Спроектируйте и создайте таблицу в этой же БД для хранения новостей и организации новостной ленты на сайте.

— Добавить нового пользователя для созданной БД и установить ему привилегии, позволяющие работать только с данной БД (webProject)

9. Тема 16. Безопасность распределенных баз данных.

Типовые задачи (всего 8 заданий):

Задания базового уровня:

— Пояснить, в чем состоит отличие понятия типа сущности и элемента сущности.

— Перечислить способы представления сущности.

— Перечислить правила атрибутов.

— Как классифицируются атрибуты?

— Дать понятие безусловной, условной, биусловной, рекурсивной связи.

— Каковы фундаментальные виды связей?

Задания повышенного уровня:

— Выполнить процедуру построения реляционной модели данных из ER-модели, построив необходимый набор отношений.

— Средствами имеющейся СУБД создать спроектированную базу данных, ее таблицы, задать необходимые ограничения целостности.

Список рекомендуемой литературы

1. Основная литература

1. Васильева, М. А. Фильтрация набора данных : учебно-методическое пособие / М. А. Васильева, О. А. Тимофеева, К. М. Филипченко. — Москва : РУТ (МИИТ), 2020. — 31 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/175827
2. Фешина, Е. В. Базы данных : учебник / Е. В. Фешина, В. В. Ткаченко. — Краснодар : КубГАУ, 2020. — 172 с. — ISBN 978-5-907402-36-2. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/254261

2. Дополнительная литература

1. Гущин, А.Н. Базы данных / А.Н. Гущин. – Москва :Директ-Медиа, 2014. – 266 с. : ил.,табл., схем. – Режим доступа: по подписке. – URL: http://biblioclub.ru/index.php?page=book&id=222149
2. Карпова, Т.С. Базы данных: модели, разработка, реализация / Т.С. Карпова. – 2-е изд., исправ. – Москва : Национальный Открытый Университет «ИНТУИТ», 2016. – 241 с. : ил. – Режим доступа: по подписке. – URL: http://biblioclub.ru/index.php?page=book&id=429003

3. Интернет-ресурсы

1. Audio Hi-Fi http://www.audiohi-fi.narod.ru/
2. Сайт любителей радио http://manimaks.ru/
3. Сайт по ламповой технике http://www.radiolamp.ru/links/2.shtml
4. Портал «Радиотехник» http://shema.org.ua/
5. Портал для радиолюбителей http://www.radioman-portal/ru/
6. Dinistor http://www.dinistor.net.ru/
7. RadioSovet http://www.radiosovet.ru/

4. Программное обеспечение

1. Программы электронного ведения проектов (ОС Windows 2000 XP\ Vista \7)
2. Пакет MS Office
3. MS Project
4.MS Internet Explorer.