При этих условиях студент не только хорошо усвоит материал, но и научится применять его на практике, а также получит дополнительный стимул (и это очень важно) для активной проработки лекции.

При самостоятельном решении задач нужно обосновывать каждый этап решения, исходя из теоретических положений курса. Если студент видит несколько путей решения проблемы (задачи), то нужно сравнить их и выбрать самый рациональный. Полезно до начала вычислений составить краткий план решения проблемы (задачи). Решение проблемных задач или примеров следует излагать подробно, вычисления располагать в строгом порядке, отделяя вспомогательные вычисления от основных. Решения при необходимости нужно сопровождать комментариями, схемами, чертежами и рисунками.

Следует помнить, что решение каждой учебной задачи должно доводиться до окончательного логического ответа, которого требует условие, и по возможности с выводом. Полученный ответ следует проверить способами, вытекающими из существа данной задачи. Полезно также (если возможно) решать несколькими способами и сравнить полученные результаты. Решение задач данного типа нужно продолжать до приобретения твердых навыков в их решении.

4.3. Методические рекомендации по самопроверке знаний

После изучения определенной темы по записям в конспекте и учебнику, а также решения достаточного количества соответствующих задач на практических занятиях и самостоятельно студенту рекомендуется, провести самопроверку усвоенных знаний, ответив на контрольные вопросы по изученной теме.

В случае необходимости нужно еще раз внимательно разобраться в материале.

Иногда недостаточность усвоения того или иного вопроса выясняется только при изучении дальнейшего материала. В этом случае надо вернуться назад и повторить плохо усвоенный материал. Важный критерий усвоения теоретического материала - умение решать задачи или пройти тестирование по пройденному материалу. Однако следует помнить, что правильное решение задачи может получиться в результате применения механически заученных формул без понимания сущности теоретических положений.

4.4. Методические рекомендации по написанию научных текстов (докладов, эссе, научных статей и т.д.)

Перед тем, как приступить к написанию научного текста, важно разобраться, какова истинная цель вашего научного текста - это поможет вам разумно распределить свои силы и время.

Во-первых, сначала нужно определиться с идеей научного текста, а для этого необходимо научиться либо относиться к разным явлениям и фактам несколько критически (своя идея – как иная точка зрения), либо научиться увлекаться какими-то известными идеями, которые нуждаются в доработке (идея – как оптимистическая позиция и направленность на дальнейшее совершенствование уже известного). Во-вторых, научиться организовывать свое время, ведь, как известно, свободное (от всяких глупостей) время – важнейшее условие настоящего творчества, для него наконец-то появляется время. Иногда именно на организацию такого времени уходит немалая часть сил и талантов.

Писать следует ясно и понятно, стараясь основные положения формулировать четко и недвусмысленно (чтобы и самому понятно было), а также стремясь структурировать свой текст. Каждый раз надо представлять, что ваш текст будет кто-то читать и ему захочется сориентироваться в нем, быстро находить ответы на интересующие вопросы (заодно представьте себя на месте такого человека). Понятно, что работа, написанная «сплошным текстом» (без заголовков, без выделения крупным шрифтом наиболее важным мест и т, п.), у кулуарного читателя должна вызывать брезгливость и даже жалость к автору (исключения составляют некоторые древние тексты, когда и жанр был иной и к текстам относиться иначе, да и таких текстов было гораздо меньше – не то, что в эпоху «информационного взрыва» и соответствующего «информационного мусора»).

Объем текста и различные оформительские требования во многом зависят от принятых в конкретном учебном заведении порядков.

Доклад - это самостоятельное исследование студентом определенной проблемы, комплекса взаимосвязанных вопросов.

Доклад не должна составляться из фрагментов статей, монографий, пособий. Кроме простого изложения фактов и цитат, в доклад е должно проявляться авторское видение проблемы и ее решения.

Рассмотрим основные этапы подготовки

а студентом.

Выполнение доклада начинается с выбора темы.

Затем студент приходит на первую консультацию к руководителю, которая предусматривает:

- обсуждение цели и задач работы, основных моментов избранной темы;
- консультирование по вопросам подбора литературы;
- составление предварительного плана.

Следующим этапом является работа с литературой. Необходимая литература подбирается студентом самостоятельно.

После подбора литературы целесообразно сделать рабочий вариант плана работы. В нем нужно выделить основные вопросы темы и параграфы, раскрывающие их содержание.

Составленный список литературы и предварительный вариант плана уточняются, согласуются на очередной консультации с руководителем.

Затем начинается следующий этап работы - изучение литературы. Только внимательно читая и конспектируя литературу, можно разобраться в основных вопросах темы и подготовиться к самостоятельному (авторскому) изложению содержания доклада. Конспектируя первоисточники, необходимо отразить основную идею автора и его позицию по исследуемому вопросу, выявить проблемы и наметить задачи для дальнейшего изучения данных проблем.

Систематизация и анализ изученной литературы по проблеме исследования позволяют студенту написать работу.

Рабочий вариант текста доклада предоставляется руководителю на проверку. На основе рабочего варианта текста руководитель вместе со студентом обсуждает возможности доработки текста, его оформление. После доработки доклад сдается на кафедру для его оценивания руководителем.

Требования к написанию доклада

Написание 1 доклада является обязательным условием выполнения плана СРС по любой дисциплине профессионального цикла.

Тема доклада может быть выбрана студентом из предложенных в рабочей программе или фонде оценочных средств дисциплины, либо определена самостоятельно, исходя из интересов студента (в рамках изучаемой дисциплины). Выбранную тему необходимо согласоваться с преподавателем.

Доклад должен быть написан научным языком.

Объем доклада должен составлять 20-25 стр.

Структура доклада:

- Введение (не более 3-4 страниц). Во введении необходимо обосновать выбор темы, ее актуальность, очертить область исследования, объект исследования, основные цели и задачи исследования.
- Основная часть состоит из 2-3 разделов. В них раскрывается суть исследуемой проблемы, проводится обзор мировой литературы и источников Интернет по предмету исследования, в котором дается характеристика степени разработанности проблемы и авторская аналитическая оценка основных теоретических подходов к ее решению. Изложение материала не должно ограничиваться лишь описательным подходом к раскрытию выбранной темы. Оно также должно содержать собственное видение

рассматриваемой проблемы и изложение собственной точки зрения на возможные пути ее решения.

- Заключение (1-2 страницы). В заключении кратко излагаются достигнутые при изучении проблемы цели, перспективы развития исследуемого вопроса
- Список использованной литературы (не меньше 10 источников), в алфавитном порядке, оформленный в соответствии с принятыми правилами. В список использованной литературы рекомендуется включать работы отечественных и зарубежных авторов, в том числе статьи, опубликованные в научных журналах в течение последних 3-х лет и ссылки на ресурсы сети Интернет.
- Приложение (при необходимости).

Требования к оформлению:

- текст с одной стороны листа;
- шрифт Times New Roman;
- кегль шрифта 14;
- межстрочное расстояние 1,5;
- поля: сверху 2,5 см, снизу – 2,5 см, слева - 3 см, справа 1,5 см;
- доклад должен быть представлен в сброшюрованном виде.

Порядок защиты доклада:

Защита доклада проводится на практических занятиях, после окончания работы студента над ним и исправления всех недочетов, выявленных преподавателем в ходе консультаций. На защиту доклада отводится 5-7 минут времени, в ходе которого студент должен показать свободное владение материалом по заявленной теме. При защите доклада приветствуется использование мультимедиа-презентации.

Оценка доклада

Доклад оценивается по следующим критериям:

• соблюдение требований к его оформлению;

• необходимость и достаточность для раскрытия темы приведенной в тексте доклада информации;

• умение студента свободно излагать основные идеи, отраженные в докладе;

• способность студента понять суть задаваемых преподавателем и сокурсниками вопросов и сформулировать точные ответы на них.

Критерии оценки:

Оценка «отлично» выставляется студенту, если в докладе студент исчерпывающе, последовательно, четко и логически стройно излагает материал; свободно справляется с задачами, вопросами и другими видами применения знаний; использует для написания доклада современные научные материалы; анализирует полученную информацию; проявляет самостоятельность при написании доклада.

Оценка «хорошо» выставляется студенту, если качество выполнения доклада достаточно высокое. Студент твердо знает материал, грамотно и по существу излагает его, не допуская существенных неточностей в ответе на вопросы по теме доклада.

Оценка «удовлетворительно» выставляется студенту, если материал доклада излагается частично, но пробелы не носят существенного характера, студент допускает неточности и ошибки при защите доклада, дает недостаточно правильные формулировки, наблюдаются нарушения логической последовательности в изложении материала.

Оценка «неудовлетворительно» выставляется студенту, если он не подготовил доклад или допустил существенные ошибки. Студент неуверенно излагает материал доклада, не отвечает на вопросы преподавателя.

Описание шкалы оценивания

Максимально возможный балл за весь текущий контроль устанавливается равным 55. Текущее контрольное мероприятие считается сданным, если студент получил за него не менее 60% от установленного для этого контроля максимального балла. Рейтинговый балл, выставляемый студенту за текущее контрольное мероприятие, сданное студентом в

установленные графиком контрольных мероприятий сроки, определяется следующим образом:

Уровень выполнения контрольного задания	Рейтинговый балл (в % от максимального балла за контрольное задание)
Отличный	100
Хороший	80
Удовлетворительный	60
Неудовлетворительный	0

4.5. Методические рекомендации по выполнению исследовательских проектов

Исследовательская проектная работа – это групповая работа, для выполнения которой необходим выбор и приложение научной методики к поставленной задаче, получение собственного теоретического или экспериментального материала, на основании которого необходимо провести анализ и сделать выводы об исследуемом явлении. Выполнение проекта – это всегда коллективная, творческая практическая работа, предназначенная для получения определенного продукта или научно-технического результата. Такая работа подразумевает четкое, однозначное формирование поставленной задачи, определение сроков выполнения намеченного, определение требований к разрабатываемому объекту.

Выполнение 1 группового проекта является обязательным условием выполнения самостоятельной работы по любой дисциплине профессионального цикла. Тема проектного задания может быть выбрана студентом из предложенных в рабочей программе или фонде оценочных средств дисциплины, либо определена самостоятельно, исходя из интересов студента (в рамках изучаемой дисциплины). Выбранную тему необходимо согласоваться с преподавателем.

Требования по выполнению и оформлению проекта

При выполнении проекта приветствуется работа в группе (2-3 человека). Проект – это исследовательская работа, в ходе которой студенты должны продемонстрировать владение навыками научного исследования, умения проводить анализ, обобщать информацию, делать выводы, предлагать свои решения проблемы, рассматриваемой в проекте.

При подготовке материалов проекта студенты должны продемонстрировать владение современными методами компьютерной обработки данных.

Критерии оценки работы участника проекта.

Для каждого из участников проекта оцениваются:

• профессиональные теоретические знания в соответствующей области;

• умение работать со справочной и научной литературой, осуществлять поиск необходимой информации в Интернет;

• умение работать с техническими средствами;

• умение пользоваться соответствующими выполняемому проекту информационными технологиями;

• умение готовить материалы проекта для презентации: составлять и редактировать тексты, формировать презентацию проекта;

• умение работать в команде;

• умение публично представлять результаты собственной деятельности;

• коммуникабельность, инициативность, творческие способности.

Критерии выставления оценки участникам проекта

Оценка	Профессиональные компетенции	Компетенции, связанные с использованием соответствующих выполняемому проекту технических средств и информационных технологий	Иные универсальные компетенции (коммуникабельность, инициативность, умение работать в «команде», управленческие навыки и т.д.)	Отчетность
«Отлично»	Работа выполнена на высоком профессиональном уровне. Представленный материал в основном фактически верен, допускаются негрубые фактические неточности. Студент свободно отвечает на вопросы, связанные с проектом.	Технические средства и информационные технологии освоены и использованы для реализации проекта полностью	Студент проявил инициативу, творческий подход, способность к выполнению сложных заданий, навыки работы в коллективе, организационные способности.	Проект представлен полностью и в срок.
«Хорошо»	Работа выполнена на достаточно высоком профессиональном уровне. Допущено до 4–5 фактических ошибок. Студент отвечает на вопросы, связанные с проектом, но недостаточно полно.	Обнаруживаются некоторые ошибки в использовании соответствующих технических средств и информационных технологий	Студент достаточно полно, но без инициативы и творческих находок выполнил возложенные на него задачи.	Проект представлен достаточно полно и в срок, но с некоторыми недоработками.
«Удовлетворительно»	Уровень недостаточно высок. Допущено до 8 фактических ошибок. Студент может ответить лишь на некоторые из заданных вопросов, связанных с проектом.	Обнаруживает недостаточное владение навыками работы с техническими средствами и соответствующими и информационными технологиями	Студент выполнил большую часть возложенной на него работы.	Проект сдан со значительным опозданием (более недели) и не полностью
«Неудовлетворительно»	Работа не выполнена или выполнена на низком уровне. Допущено более 8 фактических ошибок. Ответы на связанные с	Навыков работы с техническими средствами нет, информационные технологии не освоены	Студент практически не работал, не выполнил свои задачи или выполнил лишь отдельные не	Проект не сдан.

Оценка	Профессиональные компетенции	Компетенции, связанные с использованием соответствующих выполняемому проекту технических средств и информационных технологий	Иные универсальные компетенции (коммуникабельность, инициативность, умение работать в «команде», управленческие навыки и т.д.)	Отчетность
	проектом вопросы обнаруживают непонимание предмета и отсутствие ориентации в материале проекта.		существенные поручения в групповом проекте.	

Студенты должны: защитить проект в режиме презентации, предъявить файлы выполненного проекта, уметь рассказать о технологиях, использованных ими при выполнении проекта, дать оценку работы каждого члена группы (*если проект групповой*). Максимально возможный балл за весь текущий контроль устанавливается равным **55.** Текущее контрольное мероприятие считается сданным, если студент получил за него не менее 60% от установленного для этого контроля максимального балла. Рейтинговый балл, выставляемый студенту за текущее контрольное мероприятие, сданное студентом в установленные графиком контрольных мероприятий сроки, определяется следующим образом:

Уровень выполнения контрольного задания	Рейтинговый балл (в % от максимального балла за контрольное задание)
Отличный	100
Хороший	80
Удовлетворительный	60
Неудовлетворительный	0

4.6. Методические рекомендации по подготовке к экзаменам и зачетам

Изучение многих общепрофессиональных и специальных дисциплин завершается экзаменом. Подготовка к экзамену способствует закреплению, углублению и обобщению знаний, получаемых, в процессе обучения, а также применению их к решению практических задач. Готовясь к экзамену, студент ликвидирует имеющиеся пробелы в знаниях, углубляет, систематизирует и упорядочивает свои знания. На экзамене студент демонстрирует то, что он приобрел в процессе обучения по конкретной учебной дисциплине.

Экзаменационная сессия - это серия экзаменов, установленных учебным планом. Между экзаменами интервал 3-4 дня. Не следует думать, что 3-4 дня достаточно для успешной подготовки к экзаменам.

В эти 3-4 дня нужно систематизировать уже имеющиеся знания. На консультации перед экзаменом преподаватель знакомит с основными требованиями, ответят на возникшие у них вопросы. Поэтому посещение консультаций обязательно.

Требования к организации подготовки к экзаменам те же, что и при занятиях в течение семестра, но соблюдаться они должны более строго. Во-первых, очень важно соблюдение

режима дня; сон не менее 8 часов в сутки, занятия заканчиваются не позднее, чем за 2-3 часа до сна. Оптимальное время занятий - утренние и дневные часы. В перерывах между занятиями рекомендуются прогулки на свежем воздухе, неутомительные занятия спортом. Во-вторых, наличие хороших собственных конспектов лекций. Даже в том случае, если была пропущена какая-либо лекция, необходимо вовремя ее восстановить (переписать ее на кафедре), обдумать, снять возникшие вопросы для того, чтобы запоминание материала было осознанным. В-третьих, при подготовке к экзаменам у студента должен быть хороший учебник или конспект литературы, прочитанной по указанию преподавателя в течение семестра. Здесь можно эффективно использовать листы опорных сигналов.
Вначале следует просмотреть весь материал по сдаваемой дисциплине, отметить для себя трудные вопросы. Обязательно в них разобраться. В заключение еще раз целесообразно повторить основные положения, используя при этом листы опорных сигналов.
Систематическая подготовка к занятиям в течение семестра позволит использовать время экзаменационной сессии для систематизации знаний.

Контроль самостоятельной работы студентов

Контроль самостоятельной работы проводится преподавателем в аудитории.
Предусмотрены следующие виды контроля: собеседование, оценка доклада, оценка презентации, оценка участия в круглом столе, оценка выполнения проекта.
Подробные критерии оценивания компетенций приведены в Фонде оценочных средств для проведения текущей и промежуточной аттестации.

Список литературы для выполнения СРС

Основная литература:

1. Гайдамакин Н.А. Автоматизированные информационные системы, базы и банки данных вводный курс: учеб.пособие для вузов / Н. А. Гайдамакин. - М.: Гелиос АРВ, 2013. - 3 с. ил. - Библиогр.: с. 354-355. - Алф.-предм. указ.: с. 356-364. - ISBN 5-85438-035-8
2. Основы проектирования и разработки реляционных баз данных: (Спец. 075200 Компьютерная безопасность): учеб. пособие / авт.-сост.: О. М. Лепешкин, Д. Л. Осипов Федеральное агентство по образованию, Ставроп. гос. ун-т. - Ставрополь : Изд-во СГ 2007. - 203 с. : прил. - Библиогр.: с. 199-200

Дополнительная литература:

3 Гущин, А.Н. Базы данных / А.Н. Гущин. – Москва :Директ-Медиа, 2014. – 266 с.: ил., таб. схем. – Режим доступа: по подписке. – URL: http://biblioclub.ru/index.php?page=book&id=222149
4 Карпова, Т.С. Базы данных: модели, разработка, реализация / Т.С. Карпова. – 2-е из исправ. – Москва : Национальный Открытый Университет «ИНТУИТ», 2016. – 241 с.: – Режим доступа: по подписке. – URL: http://biblioclub.ru/index.php?page=book&id=429003

Методические рекомендации для самостоятельной работы студентов по дисциплине

1. Методические рекомендации по выполнению лабораторных работ по дисциплине «Безопасность баз данных».
2. Методические рекомендации по организации самостоятельной работы студентов по дисциплине «Безопасность баз данных».

Интернет-ресурсы:

1. http://el.ncfu.ru/ – система управления обучением ФГАОУ ВО СКФУ. Дистанционная поддержка дисциплины «Цифровая грамотность и обработка больших данных»
2. http://www.unesco.org – Сайт ООН Информационно-коммуникационные технологии
3. http://www.intuit.ru – Интернет-Университет Компьютерных технологий.

ДОКУМЕНТ ПОДПИСАН
ЭЛЕКТРОННОЙ ПОДПИСЬЮ

Сертификат: 12000002A633E3D113AD425FB50002000002A6

Владелец: Шебзухова Татьяна Александровна

Действителен: с 20.08.2021 по 20.08.2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Пятигорский институт (филиал) СКФУ

Методические указания

по выполнению практических работ
по дисциплине
«БЕЗОПАСНОСТЬ БАЗ ДАННЫХ»
для направления подготовки **10.03.01 Информационная безопасность**
направленность (профиль) **Безопасность компьютерных систем**

Пятигорск
2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

1. Цель и задачи изучения 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, а также приобретение набора общекультурных и общепрофессиональных компетенций будущего бакалавра по направлению подготовки 10.03.01 «Информационная безопасность»

2. Оборудование и материалы

Для проведения практических работ необходимо следующее материально-техническое обеспечение: персональный компьютер; проектор; возможность выхода в сеть Интернет для поиска по образовательным сайтам и порталам; интерактивная доска.

3. Наименование практических занятий

№ Темы дисциплины	Наименование тем дисциплины, их краткое содержание	Объем часов	Из них практическая подготовка, часов
5 семестр			
Тема 1. Базы данных и файловые системы			
1	Безопасность архитектуры «клиент-сервер»	1.5	1.5
Тема 3. Подходы к организации баз данных			
3	Безопасность архитектуры «клиент-сервер»	1.5	1.5
Тема 5. Базисные средства манипулирования реляционными данными			
5	Компиляторы SQL и проблемы безопасности оптимизации	1.5	1.5
Тема 9. Методы сериализации транзакция			
9	Средства манипулирования данными в SQL	1.5	1.5
Тема 11. Язык SQL. Функции и основные возможности			
11	Методы сериализации транзакций	1.5	1.5
Тема 12. Средства манипулирования данными в SQL			
12	Управление параллельностью работы транзакций	1.5	1.5
Тема 13. Безопасность SQL при прикладном программировании			
13	Базисные средства манипулирования реляционными данными	1.5	1.5
Тема 15. Безопасность архитектуры «клиент-сервер»			
15	Общие понятия реляционного подхода к организации баз данных	1.5	1.5
Тема 17. Безопасность объектно-ориентированных баз данных			
17	Подходы к организации баз данных	1.5	1.5
	Итого за семестр	**13.5**	13.5
	Итого	**13.5**	13.5

4. Содержание практических работ

Раздел 1. Архитектура баз данных Практическое занятие №1

Тема 1. Подходы к организации баз данных
Создание свободных таблиц данных

СНОВНЫЕ 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 первое поле столбца *Name* (Имя). При задании названий полей можно использовать буквы, цифры и знак подчёркивания. Все попытки ввести специальные символы Visual FoxPro проигнорирует.

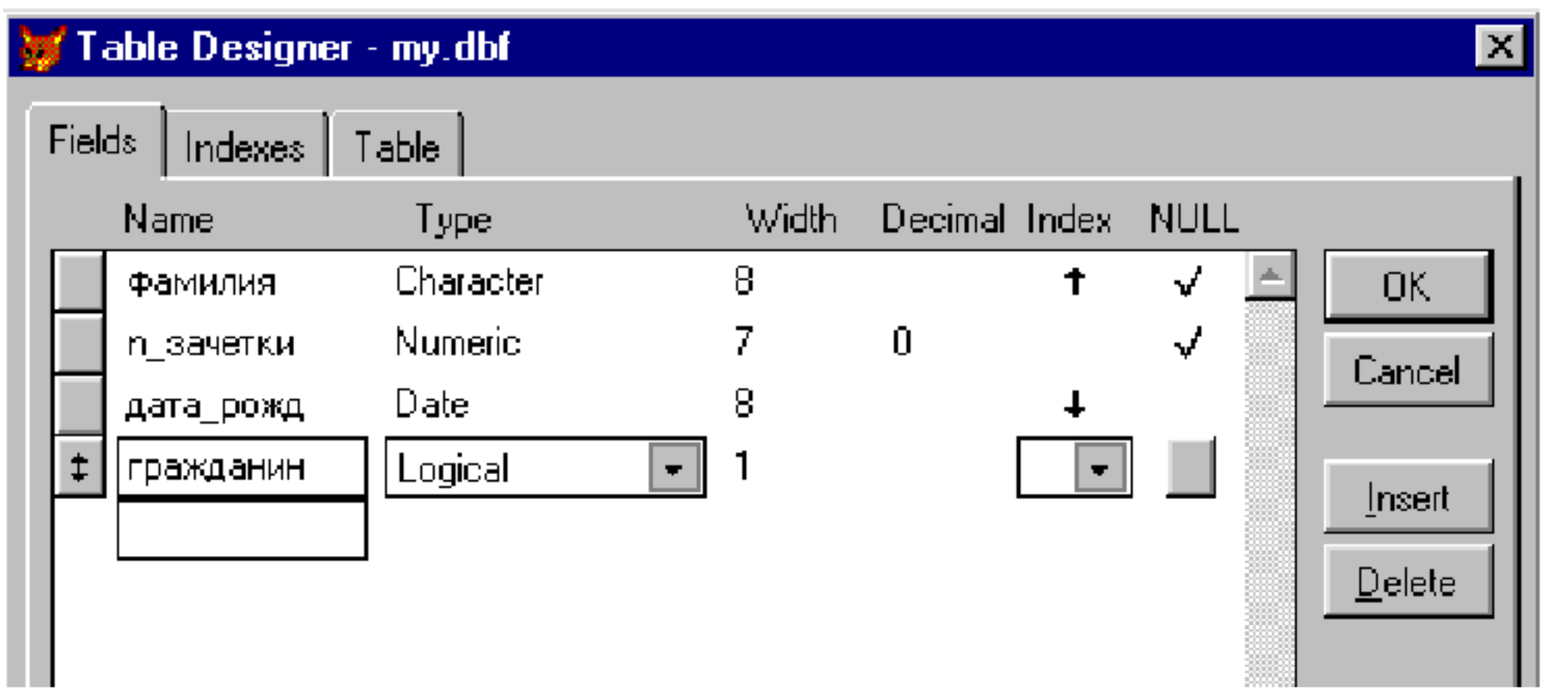

Рисунок 1 - Окно конструктора таблиц Table Designer

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 $-0{,}999999999*10^{+19}$ до $+0{,}999999999*10^{+20}$;

Тип данных Double используется для хранения данных с высокой точностью в диапазоне от $\pm 4{,}940656485412 47*10^{-324}$ до $\pm 1{,}797693134862 32*10^{+308}$

– 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 для ввода 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.

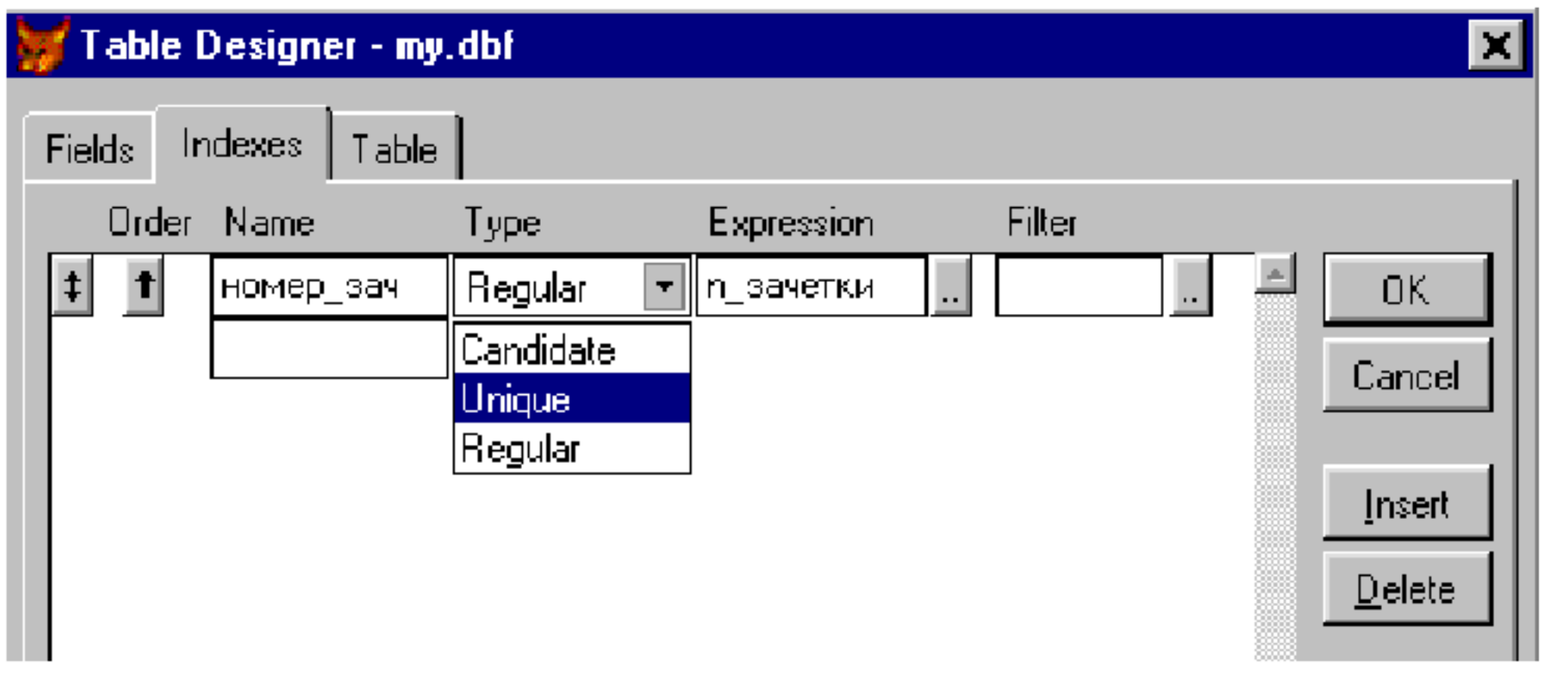

Рисунок 2 - Вкладка Indexes конструктора таблицы.

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 запись 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.

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:

16) Перечислить основные модели данных.
17) Пояснить классификацию моделей данных.
18) Пояснить принцип модели «сущность-связь».

Практическое занятие №2

Тема 2. Подходы к организации баз данных

Создание форм с использованием конструктора форм

- 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).

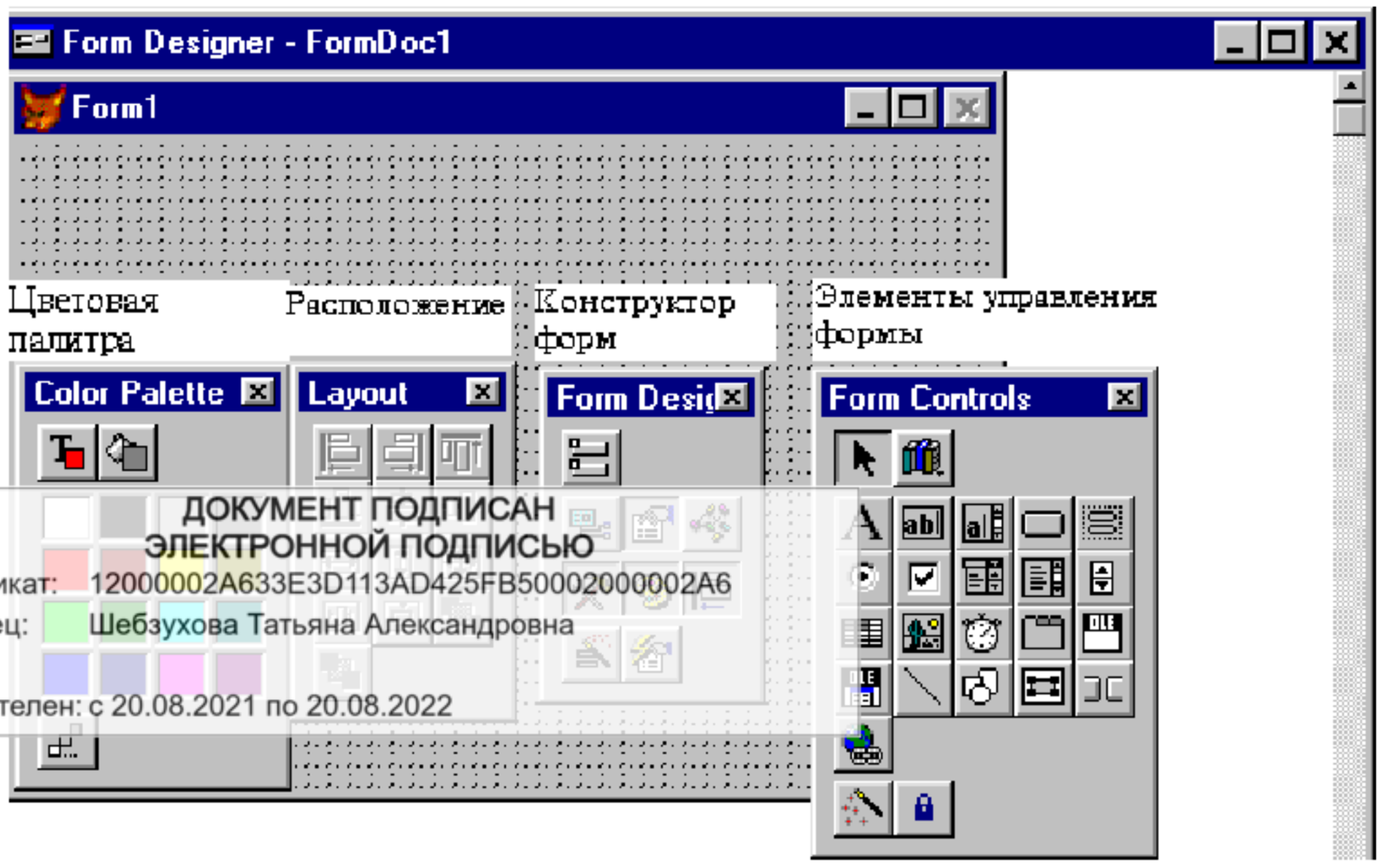

Рисунок 1 - Окно конструктора форм

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 (Запретить 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).

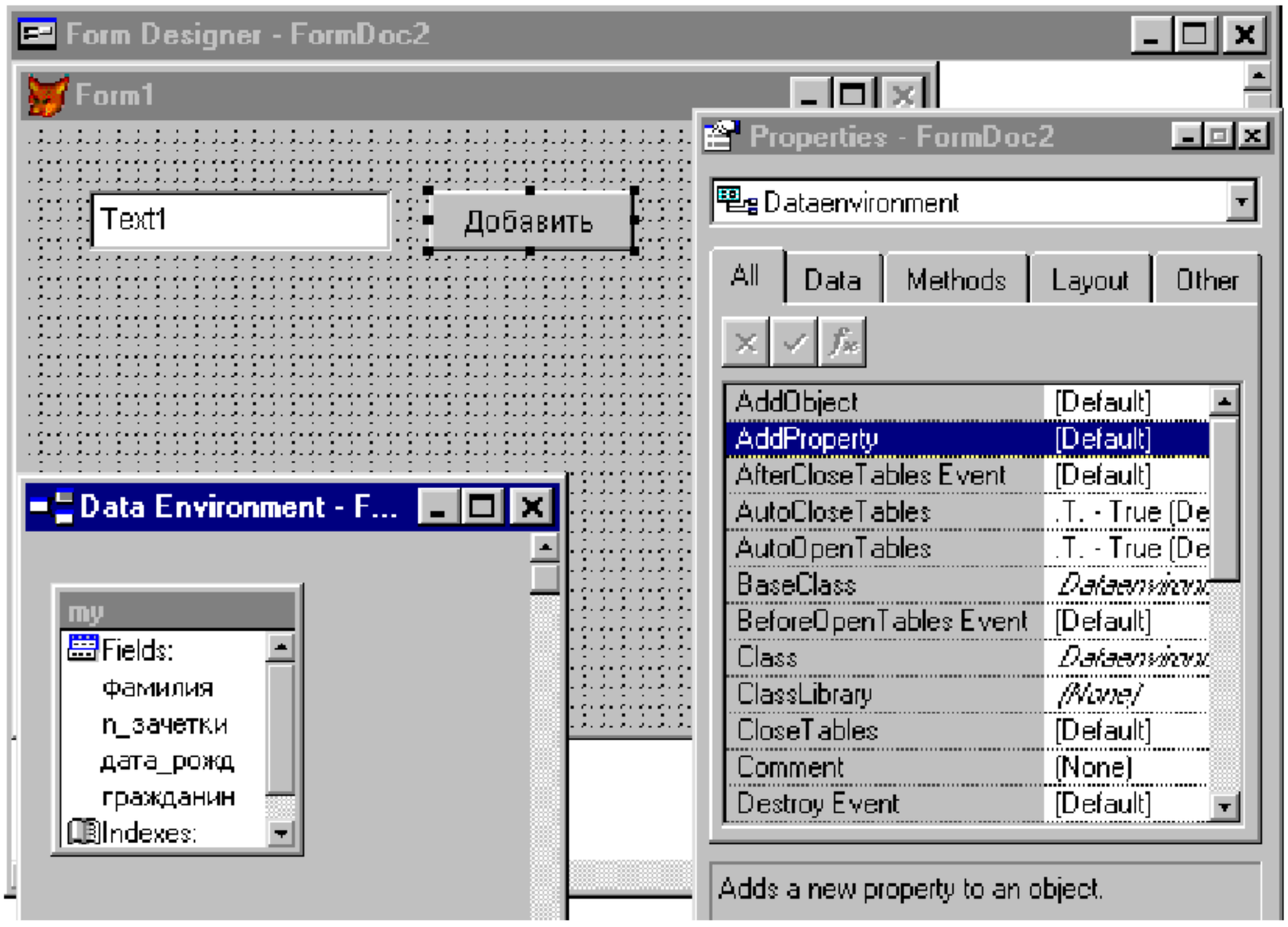

Рисунок 2 - Окно Properties и окно Data Environment.

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 (имя переменной), по которому можно обратиться к данному объекту.

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

0 - *None* (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:

19) Перечислить достоинства и недостатки сетевой модели данных.
20) Дать понятие реляционной модель данных.
21) Перечислить достоинства и недостатки реляционной модель данных.

Практическое занятие №3

Тема 3. Базисные средства манипулирования реляционными данными

Предоставление доступа и привилегий СУБД MySQL

Запуск MySQL

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

Эта команда запустит демон mysql в фоновом режиме. В Windows 95/98 не предусмотрен

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

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).

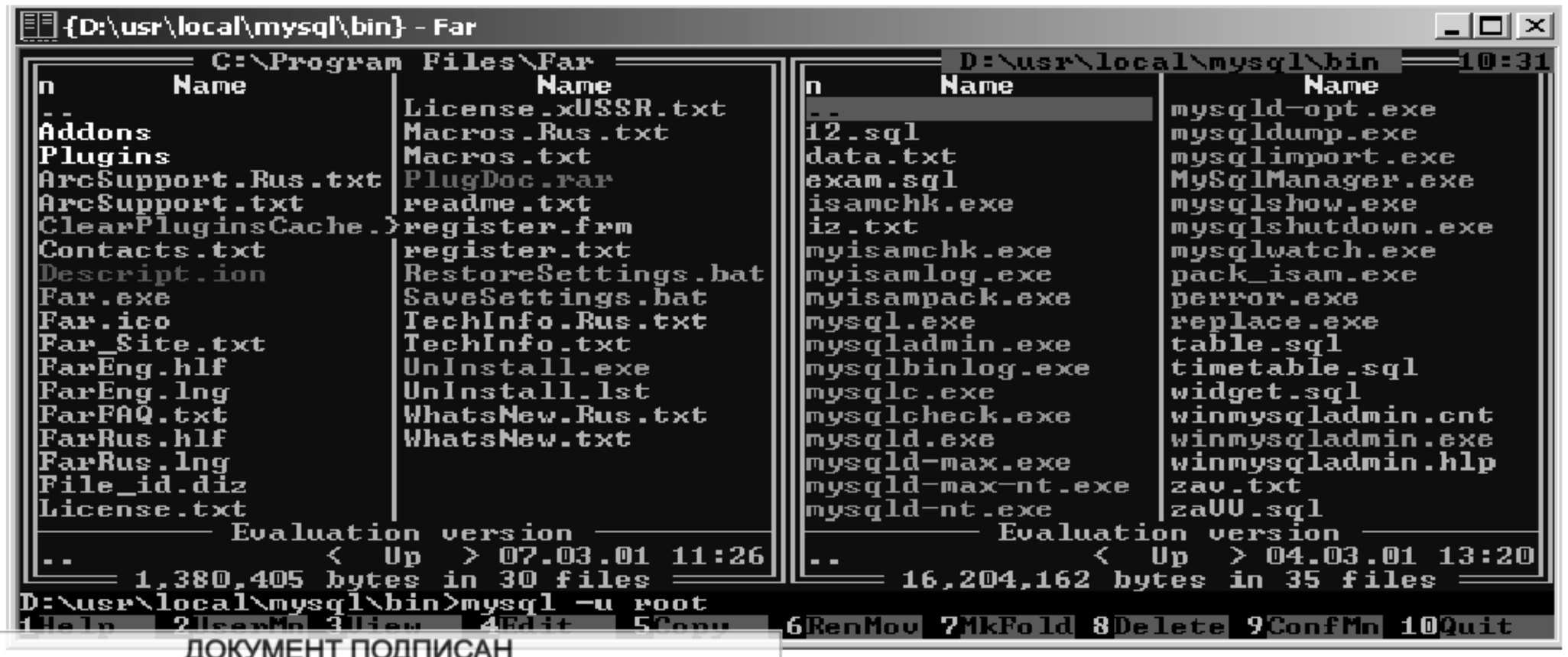

Рисунок 8 - Запуск консоли MYSQL

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

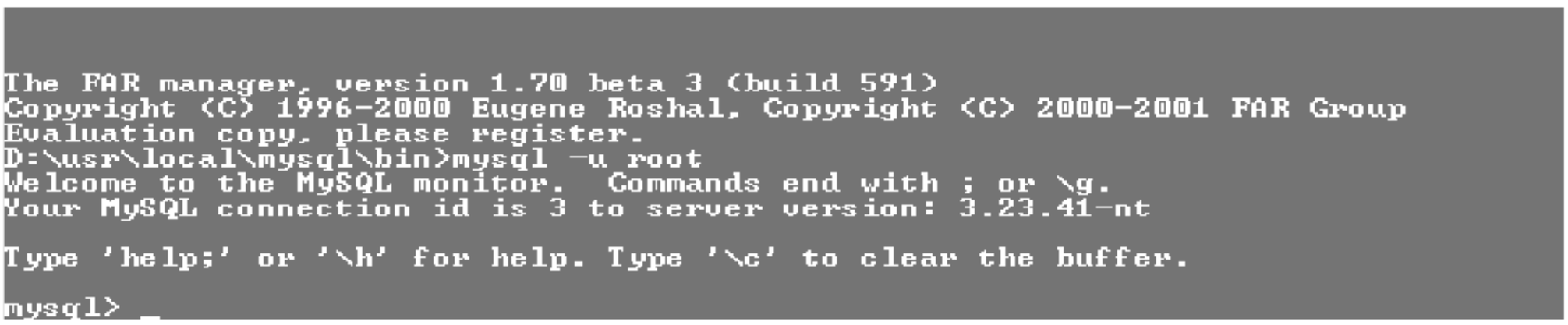

Рисунок 9 - Успешный запуск консоли

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	
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***.

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'@'localhost[7]' 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 данных и назначать привилегии. Если выполнить команду ***show columns from db;*** получим

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
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. Опции описаны

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 полную форму записи):

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.

Раздел 2. Безопасность информации в базах данных

Практическое занятие №4

Тема 4. Проектирование реляционных БД

Моделирование баз данных средствами Erwin

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.

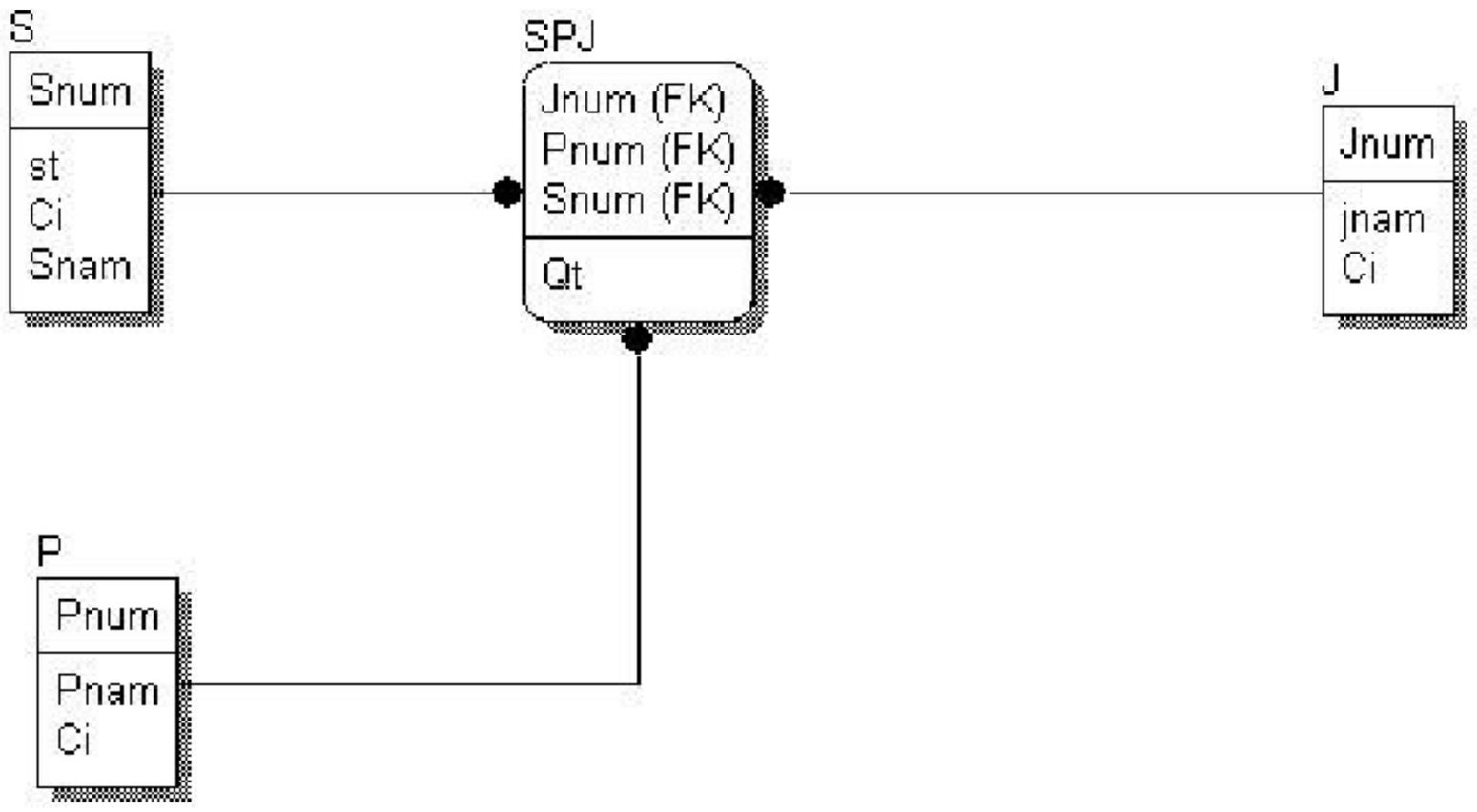

Рисунок 10 - Пример диаграммы с созданными сущностями Построение моделей в ERwin

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.

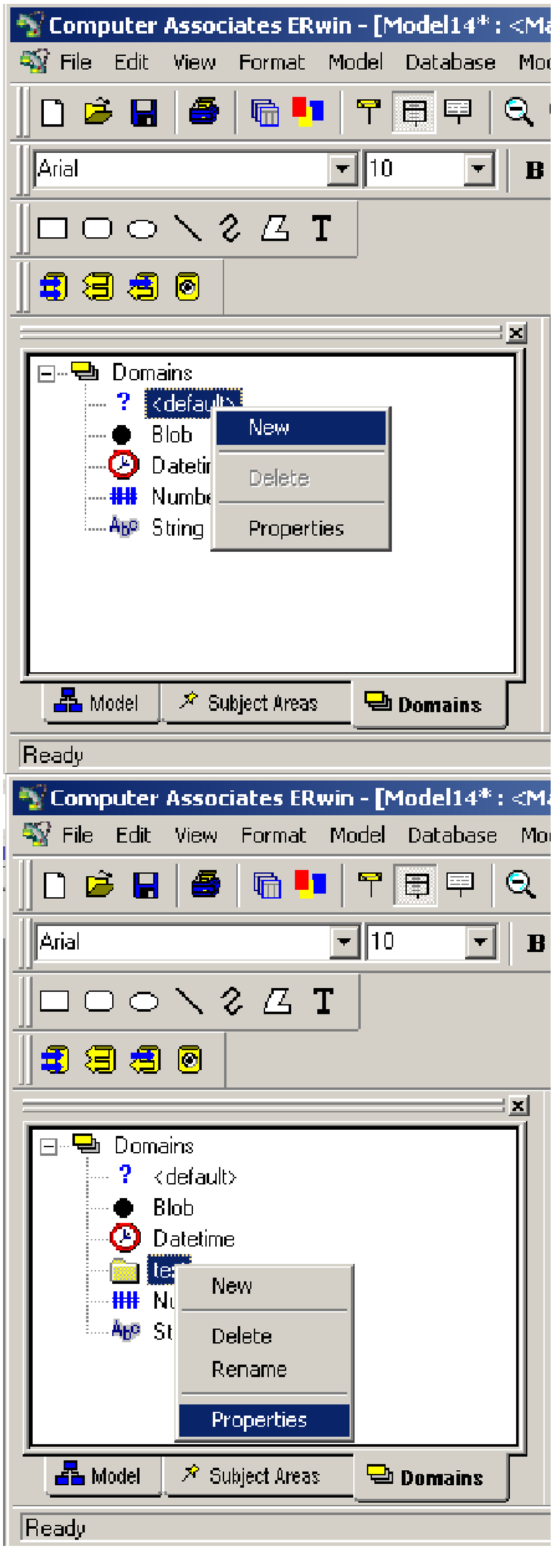

Рисунок 11-Созданиенового домена

Рисунок 12 - Указание свойств нового домена

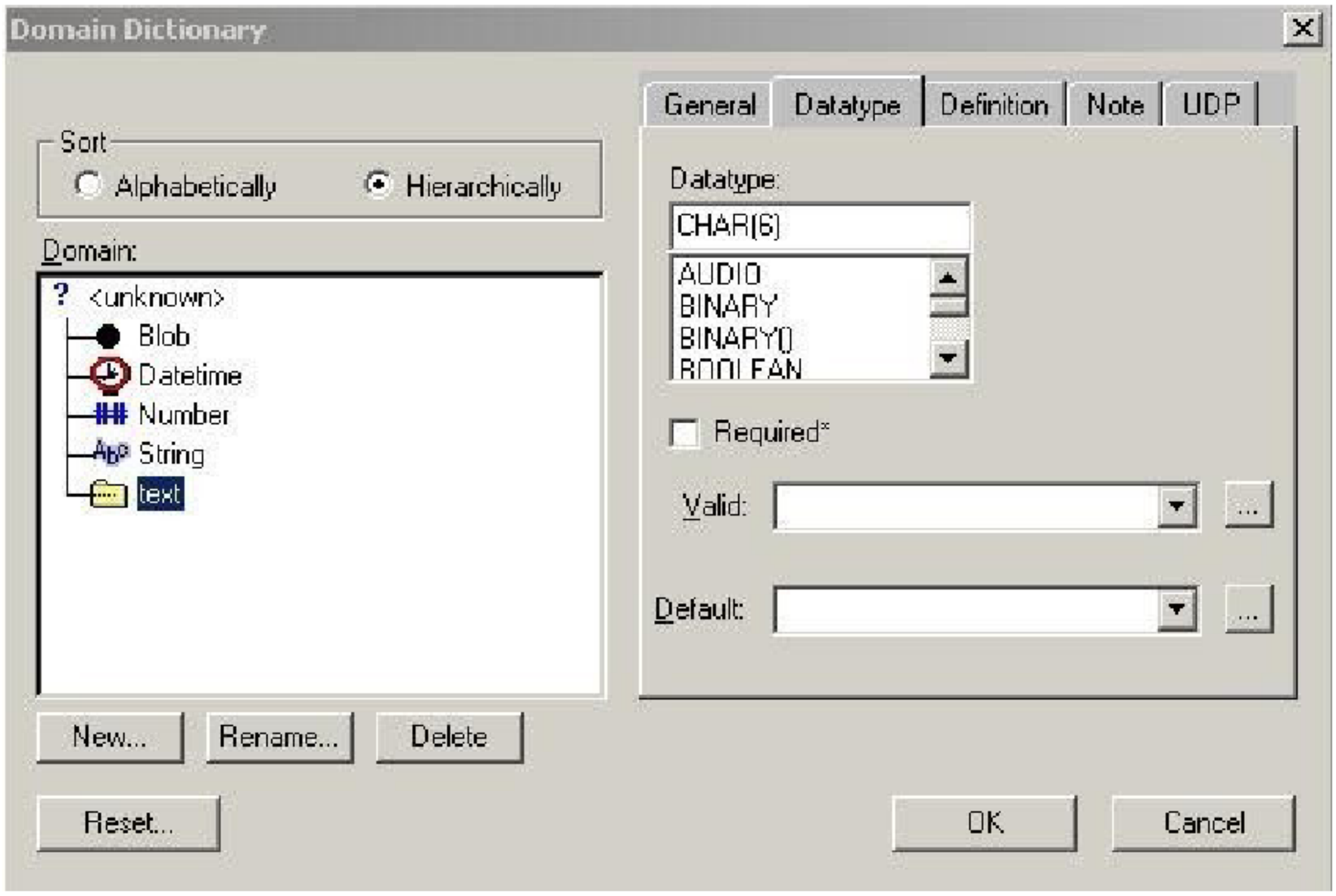

Рисунок 13 - Значение по умолчанию для нового домена

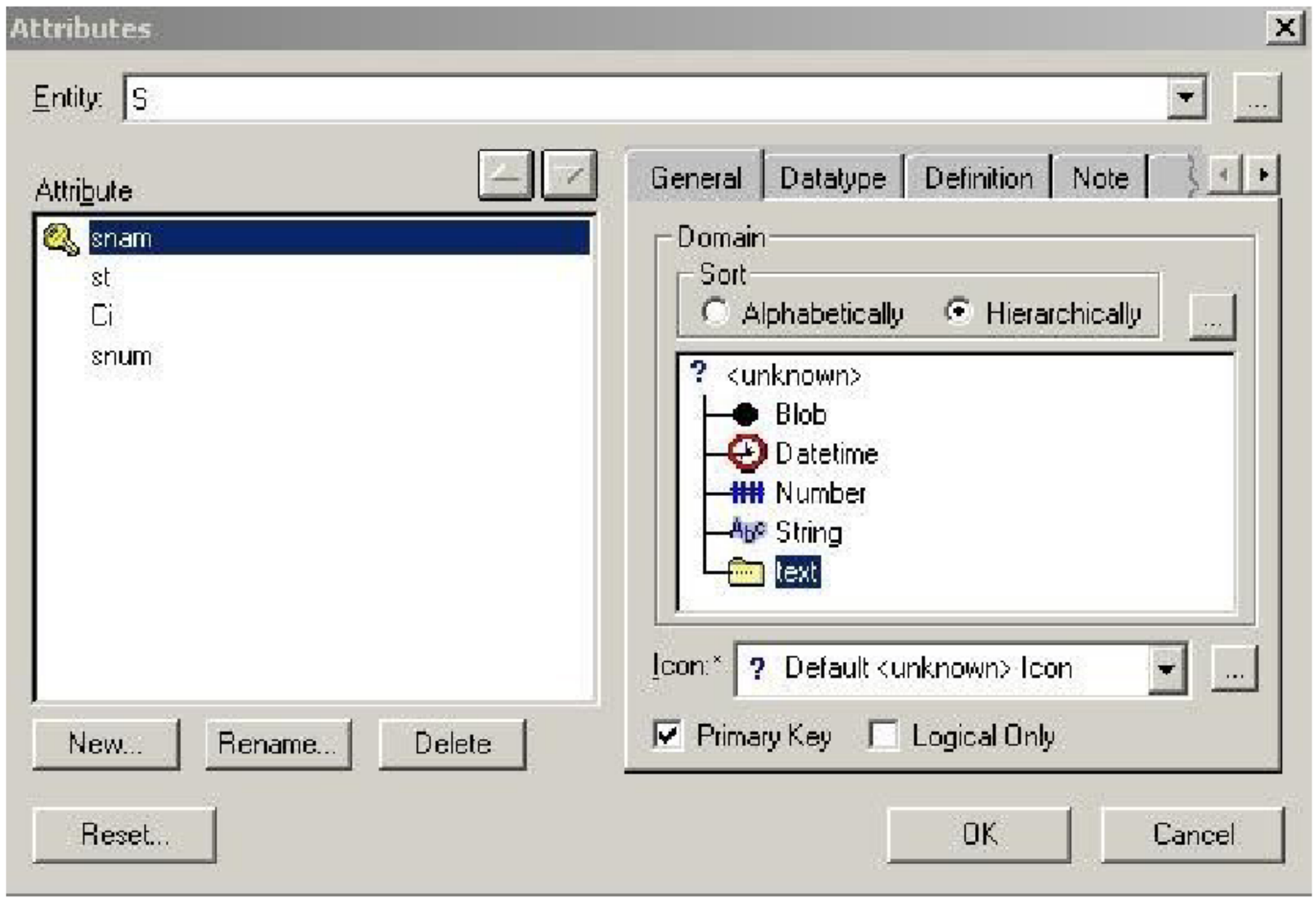

Рисунок 14 - Использование домена для указания типа данных атрибуту.

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 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

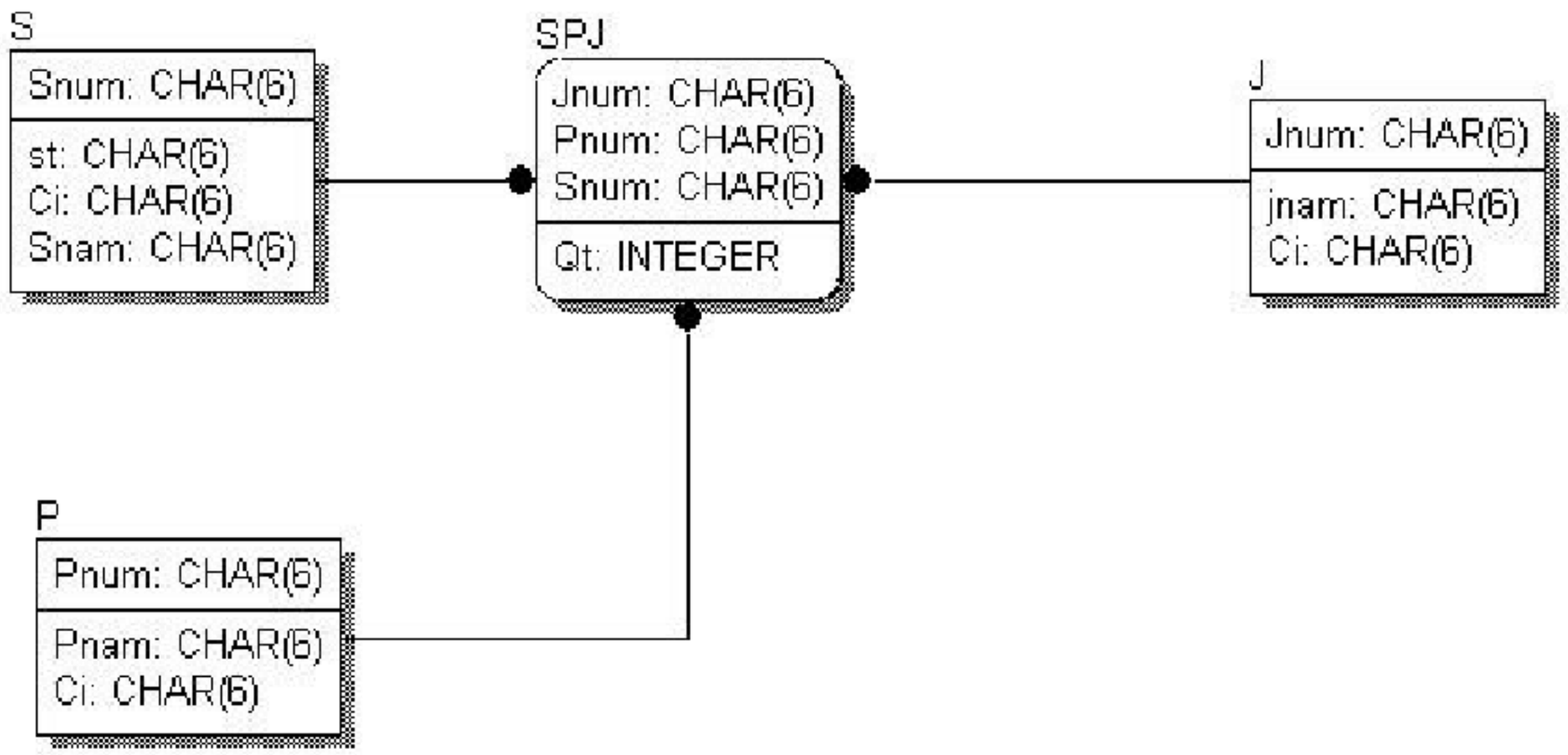

Рисунок 16 - Физическая модель с указанием формата данных.

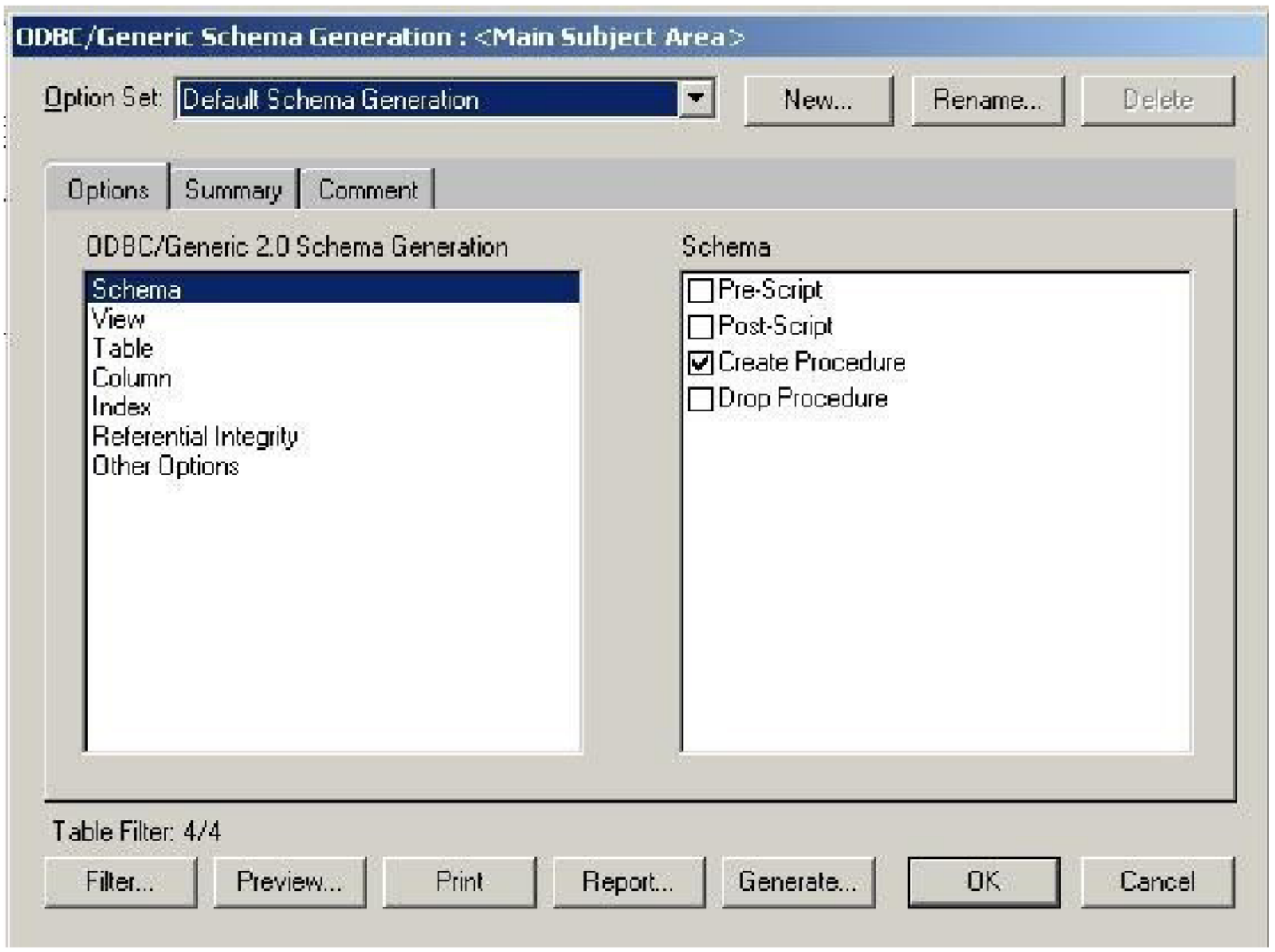

Рисунок 17 - Генерация кода SQL

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?

Практическое занятие№5

Тема 5. Управление параллельностью работы транзакций

Создание запросов и модификация таблиц базы данных.

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 определяют минимально допустимого 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

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.

5. 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?

Практическое занятие №6

Тема 6. Методы сериализации транзакция

Представления и хранимые процедуры

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 (0.00 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 dorepeat(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                       |
```

Контрольные вопросы:

22) Пояснить, каким образом предоставляются права на пользование базой данных и отдельными ее таблицами

23) Пояснить, каким образом изымаются права на пользование базой данных и отдельными ее таблицами.

24) Пояснить понятие внешней базы данных.

25) Пояснить, как идентифицируется таблица внешней базы данных

26) Пояснить, как идентифицируется таблица внешней распределенной базы данных.

Практическое занятие №7

Тема 7. Язык SQL. Функции и основные возможности

Выполнение простейших SQL-операторов с использованием средств ODBC

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 *szAuthStr* - пароль пользователя.

SWORD *cbAuthStr* - 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 *hstmt* - 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.

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.

Вариант 7

– Вставить деталь с заданными параметрами.
– 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?

Практическое занятие 8.

Доступ к базам данных посредством CGI-скрипта, написанного на языке ESQL/C

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-скрипты, 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 программе, 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 этого 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?

Практическое занятие №9

Тема 9. Компиляторы SQL и проблемы безопасности оптимизации

Построение диаграмм работ и диаграмм потоков данных информационной системы
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.
Функциональная зависимость при 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 подразделения (отделы, службы, подсистемы, группы и пр.) согласно выполняемым ими функциям.
3. Определить задачи и функции системы в целом и функции каждого подразделения (подсистемы).
4. Выполнить словесное описание работы каждого подразделения (подсистемы), алгоритмов и сценариев выполнения ими отдельных работ.
5. Построить диаграммы работ и диаграммы потоков данных для всей информационной 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 (подсистемы) 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?

5. Учебно-методическое и информационное обеспечение дисциплины

5.1. Перечень основной и дополнительной литературы, необходимой для освоения дисциплины

5.1.1. Перечень основной литературы:

1. Гайдамакин Н.А. Автоматизированные информационные системы, базы и банки данных вводный курс: учеб.пособие для вузов / Н. А. Гайдамакин. - М.: Гелиос АРВ, 2013. - 3 с. ил. - Библиогр.: с. 354-355. - Алф.-предм. указ.: с. 356-364. - ISBN 5-85438-035-8

2. Основы проектирования и разработки реляционных баз данных: (Спец. 075200 «Компьютерная безопасность»): учеб. пособие / авт.-сост.: О. М. Лепешкин, Д. Л. Осипов

Федеральное агентство по образованию, Ставроп. гос. ун-т. - Ставрополь : Изд-во СГ 2007. - 203 с. : прил. - Библиогр.: с. 199-200

5.1.2. Перечень дополнительной литературы:

5 Гущин, А.Н. Базы данных / А.Н. Гущин. – Москва :Директ-Медиа, 2014. – 266 с.: ил., таб. схем. – Режим доступа: по подписке. – URL: http://biblioclub.ru/index.php?page=book&id=222149

6 Карпова, Т.С. Базы данных: модели, разработка, реализация / Т.С. Карпова. – 2-е из исправ. – Москва : Национальный Открытый Университет «ИНТУИТ», 2016. – 241 с.: – Режим доступа: по подписке. – URL: http://biblioclub.ru/index.php?page=book&id=429003

5.2. Перечень учебно-методического обеспечения самостоятельной работы обучающихся по дисциплине (модулю)

1. Методические рекомендации по выполнению лабораторных работ по дисциплине «Безопасность баз данных».
2. Методические рекомендации по организации самостоятельной работы студентов по дисциплине «Безопасность баз данных».

5.3. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины (модуля)

1. http://el.ncfu.ru/ – система управления обучением ФГАОУ ВО СКФУ. Дистанционная поддержка дисциплины «Цифровая грамотность и обработка данных»
2. http://www.un.org - Сайт ООН Информационно-коммуникационные технологии
3. http://www.intuit.ru – Интернет-Университет Компьютерных технологий.