ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ОРЕНБУРГСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

Б1.В.ДВ.02.02 Распределенная обработка информации в автоматизированных системах

Направление подготовки (специальность)
09.04.01 Информатика и вычислительная техника

Профиль образовательной программы
“Автоматизированные системы обработки информации и управления**”**

Форма обучения очная

СОДЕРЖАНИЕ

1. Тематическое содержание дисциплины

1.1 Тема 1 «Понятие распределенной системы. Аппаратные и программные средства построения распределенных систем» (12 часов)

1.1.1 Перечень и краткое содержание рассматриваемых вопросов:

1. Понятие распределенной системы. Аппаратные и программные средства построения распределенных систем

В литературе упоминаются разные определения понятия распределенной системы (РС), но все они могут быть сведены к следующим определениям (Таненбаум):

1. Распределенная система – это набор независимых компьютеров, представляющийся их пользователям как единая система.

2. Распределенная информационная система (РИС) – это совокупность взаимодействующих друг с другом программных компонент.

Основная задача – облегчение доступа к удаленным ресурсам и контроль совместного использования этих ресурсов (компьютеров, файлов, данных в БД). Web-страницы и сети также входят в этот список.

Основные технологии: сокрытие времени ожидания связи, распределение и репликация.

1. Сокрытие времени ожидания связи применяется в случае географического масштабирования. Основная идея – постараться избежать ожидания ответа назапрос от удаленного сервера. Это означает разработку приложения-клиента с использованием асинхронной связи. При получении ответа, приложение-клиент прервет свою работу и вызовет специальный обработчик для завершения отправленного ранее запроса. Асинхронная связь используется в системах пакетной обработки и параллельных приложениях, в которых во время ожидания одной задачей завершения связи предполагается выполнение других независимых задач. Во многих задачах приложения не могут эффективно использовать асинхронную связь. Например, в задачах интерактивной обработки с использованием форм. В этих случаях для сокращения объема взаимодействия часть вычислений, которые обычно выполняются на стороне сервера, обычно перемещают на сторону клиента. Особенно это касается проверки данных на форме, т.е. эффективнее переместить на сторону клиента проверку данных всех полей формы.

2. . Распределение. Предполагает разбиение компонентов на более мелкие части с последующим их распределением в системе. Пример – система доменных имен Интернета (DNS). Пространство DNS организовано иерархически в виде дерева доменов (domains) разбитых на зоны (по странам и областям деятельности). Имена каждой зоны обрабатываются отдельным сервером имен. Получается, что служба доменных имен распределена по нескольким серверам, что позволяет избежать обработки всех запросов одним сервером. Второй пример – Web. Каждый документ имеет уникальное имя URL. Физически среда Web разнесена по многим серверам. Имя сервера, содержащего конкретный документ, определяется по его URL-адресу. Это позволяет наращивать количество документов без потери производительности.

3. Репликация (дублирование). Применение репликации повышает доступность ресурсов, а также помогает выравнивать нагрузку компонентов, что ведет к улучшению производительности. Кроме того, в географически распределенных системах репликация (в виде близко лежащей копии) помогает уменьшить проблему ожидания связи. Особую форму репликации представляет собой кеширование (caching). Хотя отличия между ними незначительны. Как и при репликации, результатом кеширования является создание копии ресурса, обычно в непосредственной близости от клиента. В отличие от репликации кеширование – это действие, предпринимаемое со стороны клиента, а не сервера.

2. Изучение палитры инструментов среды разработки Erwin

Оформить шаблон пояснительной записки для курсового проекта по дисциплине «Архитектура» в соответствии с требованиями оформления текстовой документации.

В ведомость технического проекта вносят все документы, входящие в курсовой проект.

Задание на проектирование оформляется на стандартном бланке, выдаваемом преподавателем перед началом проектирования.

Маршрутные и операционные карты выполняются на стандартных бланках (приложения А, Б). Маршрутная карта выполняется в соответствии с разработанным планом технологических операций по восстановлению детали с использованием информации о служебных символах (приложение В). Операционные карты выполняются на те операции, по которым рассчитывались нормы времени.

Ремонтный чертёж выполняется на формате А4 или А3.

3. . Уровни протоколов. Удаленный вызов процедур

При изучении вопроса необходимо обратить внимание на следующие особенности: виды сетевых протоколов.

1.2 Тема 2 «Связь в распределенных системах» (8 часов)

1.2.1 Перечень и краткое содержание рассматриваемых вопросов:

1. Связь в распределенных системах

Цель такой конфигурации состоит в повышении общей производительности сети за счет перераспределения выполняемых процессов между центральным и периферийными процессорами. У каждого из периферийных процессоров нет в распоряжении других локальных периферийных устройств, кроме тех, которые ему нужны для связи с центральным процессором. Файловая система и все устройства находятся в распоряжении центрального процессора. Предположим, что все пользовательские процессы исполняются на периферийном процессоре и между периферийными процессорами не перемещаются; будучи однажды переданы процессору, они пребывают на нем до момента завершения. Периферийный процессор содержит облегченный вариант операционной системы, предназначенный для обработки локальных обращений к системе, управления прерываниями, распределения памяти, работы с сетевыми протоколами и с драйвером устройства связи с центральным процессором.

При инициализации системы на центральном процессоре ядро по линиям связи загружает на каждом из периферийных процессоров локальную операционную систему. Любой выполняемый на периферии процесс связан с процессом-спутником, принадлежащим центральному процессору (см. [Birrell 84]); когда процесс, протекающий на периферийном процессоре, вызывает системную функцию, которая нуждается в услугах исключительно центрального процессора, периферийный процесс связывается со своим спутником и запрос поступает на обработку на центральный процессор. Процесс-спутник исполняет системную функцию и посылает результаты обратно на периферийный процессор. Взаимоотношения периферийного процесса со своим спутником похожи на отношения клиента и сервера

Теперь перейдем к рассмотрению систем с менее сильной связью, которые состоят из машин, производящих обращение к файлам, находящимся на других машинах. В сети, состоящей из персональных компьютеров и рабочих станций, например, пользователи часто обращаются к файлам, расположенным на большой машине. В последующих двух разделах мы рассмотрим такие конфигурации систем, в которых все системные функции выполняются в локальных подсистемах, но при этом имеется возможность обращения к файлам (через функции подсистемы управления файлами), расположенным на других машинах.

Для идентифицирования удаленных файлов в этих системах используется один из следующих двух путей. В одних системах в составное имя файла добавляется специальный символ: компонента имени, предшествующая этому символу, идентифицирует машину, остальная часть имени - файл, находящийся на этой машине.

2. Проектирование базы данных с помощью программы Erwin

Применение относительной и абсолютной адресаций для финансовых расчетов. Сортировка, условное форматирование и копирование созданных таблиц. Работа с листами электронной книги.

Создать таблицы ведомости начисления заработной платы за два месяца на разных листах электронной книги, произвести расчеты, форматирование, сортировку и защиту данных.

Рекомендации. Для удобства работы и формирования навыков работы с абсолютным видом адресации рекомендуется при оформлении констант окрашивать ячейку цветом, отличным от цвета расчетной таблицы. Тогда при вводе формул в расчетную окрашенная ячейка (т.е. ячейка с константой) будет вам напоминанием, что следует установить абсолютную адресацию (набором символов $ с клавиатуры или нажатием клавиши [F4]).

Краткая справка. Каждая рабочая книга Excel может содержать до 255 рабочих листов. Это позволяет, используя несколько листов, создавать понятные и четко структурированные документы, вместо того, чтобы хранить большие последовательные наборы данных на одном листе.

1.3 Тема 3 «Планировщик ОС. Изоляция приложений» (8 часов)

1.3.1 Перечень и краткое содержание рассматриваемых вопросов:

1. Планировщик ОС. Изоляция приложений

Время от времени каждому администратору приходится развертывать Web-приложения, связанные с пересылкой конфиденциальной информации. Приложение необходимо установить на надежно защищенном сервере таким образом, чтобы никакая другая программа Microsoft IIS не имела доступа к файлам приложения. Эффективные меры позволяют защитить конфиденциальную информацию от злоумышленников и опасных программ, размещенных на одном сервере с конфиденциальной информацией. Я всегда считал, что приложения, нуждающиеся в особенной защите, необходимо размещать на отдельном сервере. Однако нередко администраторам приходится разворачивать защищенные приложения на одном сервере со многими другими программами.

Например, после слияния двух компаний перед администратором может быть поставлена задача консолидации центров обработки данных, но в соответствии с политиками безопасности программы одной компании должны быть строго изолированы от приложений другой компании, даже если они размещены на одном сервере. В некоторых случаях ресурсы сервера, на котором хранятся важнейшие приложения с конфиденциальной информацией, недоиспользуются, и в процессе консолидации на сервере размещаются другие программы. Каковы бы ни были причины, предположим, что перед администратором поставлена задача развернуть приложение, требующее надежной защиты, на одном IIS-сервере с другими приложениями. На какие вопросы следует ответить и какие обстоятельства учесть, составляя план развертывания максимально изолированных друг от друга приложений?

Идентификация процесса чрезвычайно важна для изоляции приложений. Рассмотрим два приложения — защищенное и общедоступное. Если оба приложения работают с одним экземпляром процесса, то разработчик общедоступного приложения может загрузить на сервер программу, которая переключает экземпляр процесса. В данном контексте

безопасности программа, работающая в общедоступном приложении, может прочитать весь контент Web-узла из безопасного приложения и даже вызывать размещенные в нем приложения. Самая большая угроза при этом исходит от пользователей, имеющих право загружать на сервер и запускать программы, особенно двоичные исполняемые файлы.

Поэтому главную опасность представляют программисты, имеющие доступ к приложениям. Взломщик извне может добиться таких же результатов, успешно проведя атаку с использованием идентификатора процесса общедоступного приложения. Типичный пример — атаки с переполнением буфера. Злоумышленник также может разместить «троянского коня» на рабочей станции программиста и получить привилегированный доступ к серверу или получить доступ методами «социальной инженерии». По этой причине приложения, которые должны работать в контексте учетной записи System или Administrator, нельзя надежно изолировать друг от друга.

2. Изучение палитры инструментов среды разработки IBExpert.

Изучить способы и методы работы с готовой базой данных, созданной в MSExcel, и составлять отчеты по результатам работы и с помощью встроенных функций и инструментов редактора MSExcel дать ответ на следующие вопросы (ответ на каждый вопрос должен быть на отдельном листе книги):

- ✓ Посчитать выручку каждого продавца за каждый день
- ✓ Определить, какой продавец самый успешный
- ✓ Определить, самый востребованный товар
- ✓ Определить суммарную премию каждого продавца за три дня работы.

1.4 Тема 4 «Механизмы синхронизации процессов» (12 часов)

1.4.1 Перечень и краткое содержание рассматриваемых вопросов:

1. Механизмы синхронизации процессов

Рассмотренные в конце предыдущей лекции алгоритмы хотя и являются корректными, но достаточно громоздки и не обладают элегантностью. Более того, процедура ожидания входа в критический участок предполагает достаточно длительное вращение процесса в пустом цикле, то есть напрасную трату драгоценного времени процессора. Существуют и другие серьезные недостатки у алгоритмов, построенных средствами обычных языков программирования. Допустим, что в вычислительной системе находятся два взаимодействующих процесса: один из них – H – с высоким приоритетом, другой – L – с низким приоритетом. Пусть планировщик устроен так, что процесс с высоким приоритетом вытесняет низкоприоритетный процесс всякий раз, когда он готов к исполнению, и занимает процессор на все время своего CPUburst (если не появится процесс с еще большим приоритетом). Тогда в случае, если процесс L находится в своей критической секции, а процесс H, получив процессор, подошел ко входу в критическую область, мы получаем тупиковую ситуацию. Процесс H не может войти в критическую область, находясь в цикле, а процесс L не получает управления, чтобы покинуть критический участок. Для того чтобы не допустить возникновения подобных проблем, были разработаны различные механизмы синхронизации более высокого уровня.

Описанию ряда из них – семафоров, мониторов и сообщений – и посвящена данная лекция. Семафоры Одним из первых механизмов, предложенных для синхронизации поведения процессов, стали семафоры, концепцию которых описал Дейкстра (Dijkstra) в 1965 году. Концепция семафоров Семафор представляет собой целую переменную, принимающую неотрицательные значения, доступ любого процесса к которой, за исключением момента ее инициализации, может осуществляться только через две атомарные операции: P (от датского слова proberen – проверять) и V (от verhogen – увеличивать). Классическое определение этих операций выглядит следующим образом:

P(S): пока S == 0 процесс блокируется; S = S – 1; V(S): S = S + 1; Эта запись означает следующее: при выполнении операции P над семафором S сначала проверяется его значение. Если оно больше 0, то из S вычитается 1. Если оно меньше или равно 0, то процесс блокируется до тех пор, пока S не станет больше 0, после чего из S вычитается 1. При выполнении операции V над семафором S к его значению просто прибавляется 1.

Хотя решение задачи producer-consumer с помощью семафоров выглядит достаточно изящно, программирование с их использованием требует повышенной осторожности и внимания, чем отчасти напоминает программирование на языке Ассемблера. Допустим, что в рассмотренном примере мы случайно поменяли местами операции P, сначала выполнив операцию для семафора mutex, а уже затем для семафоров full и empty. Допустим теперь, что потребитель, войдя в свой критический участок (mutex сброшен), обнаруживает, что буфер пуст. Он блокируется и начинает ждать появления сообщений. Но производитель не может войти в критический участок для передачи информации, так как тот заблокирован потребителем. Получаем тупиковую ситуацию. В сложных программах произвести анализ правильности использования семафоров с карандашом в руках становится очень непросто. В то же время обычные способы отладки программ зачастую не дают результата, поскольку возникновение ошибок зависит от interleaving атомарных операций, и ошибки могут быть трудновоспроизводимы. Для того чтобы облегчить работу программистов, в 1974 году Хором (Hoare) был предложен механизм еще более высокого уровня, чем семафоры, получивший название мониторов.

2. Создание базы данных в программе IBExpert.

Создание однотабличной базы данных.

1. Создайте новую базу данных.
2. Создайте таблицу базы данных.
3. Определите поля таблицы в соответствии с таблицей 1.
4. Сохраните созданную таблицу.

На странице Приступая к работе с MicrosoftOfficeAccess**в разделе** Новая пустая база данных **выберите команду** Новая база данных.

В области **Новая база данных** в поле **Имя файла** введите имя файла. Если имя файла указано без расширения, расширение будет добавлено автоматически (*.accdb). Чтобы сохранить файл в другой папке, отличной от используемой по умолчанию, нажмите кнопку **Открыть** (рядом с полем **Имя файла),** перейдите к нужной папке (D:\Имя группы, папку создать предварительно), и нажмите кнопку **ОК.**

Имя базы данных задайте *Деканат,* а тип файла оставьте прежним, так как другие типы файлов нужны в специальных случаях.

Нажмите кнопку **Создать.**

2. Для создания таблицы базы данных:

Приложение Accesscоздаст базу данных с пустой таблицей с именем «Таблица1» и

откроет эту таблицу в режиме таблицы. Курсор находится в первой пустой ячейке столбца, **Добавить поле.**

Создание таблицы в режиме конструктора В режиме конструктора сначала создается структура новой таблицы. Затем можно переключиться в режим таблицы для ввода данных или ввести данные, используя другой метод, например вставку или импорт.

Заполнение базы данных

1. Введите ограничения на данные, вводимые в поле «Должность»; должны вводиться только слова *Профессор, Доцент или Ассистент.*
2. Задайте текст сообщения об ошибке, который будет появляться на экране при вводе неправильных данных в поле «Должность».
3. Задайте значение по умолчанию для поля «Должность» в виде слова *Доцент.*

4. Введите ограничения на данные в поле «Код преподавателя»; эти данные не должны повторяться.
5. Создайте столбец подстановок для поля «Дисциплина».
6. Заполните таблицу данными в соответствии с таблицей 4 и проверьте реакцию системы на ввод неправильных данных в поле «Должность».
7. Измените ширину каждого поля таблицы в соответствии с шириной данных.
8. Произведите поиск в таблице преподавателя Миронова.
9. Произведите замену данных: измените заработную плату ассистенту Сергеевой с 450 р. на 470 р.
10. Произведите сортировку данных в поле «Год рождения» по убыванию.
11. Произведите фильтрацию данных по полям «Должность» и «Дисциплина».
12. Просмотрите созданную таблицу, как она будет выглядеть на листе бумаги при печати.

3. Клиенты.Серверы

При изучении вопроса необходимо обратить внимание на следующие особенности: клиент-серверная архитектура локальных сетей.

1.5 Тема 5 «Основные понятия теории реляционных СУБД. Структурированный язык запросов» (16 часов)

1.5.1 Перечень и краткое содержание рассматриваемых вопросов:

1. Основные понятия теории реляционных СУБД. Структурированный язык запросов.

База данных - совокупность связанных данных, организованных по определенным правилам, предусматривающим общие принципы описания, хранения и манипулирования, независимая от прикладных программ. Реляционная база данных - это база данных, в которой информация представлена в виде двумерных таблиц, сохраняемых в файлах. Таблица состоит из строк, называемых записями, записи состоят из столбцов, называемых полями.

Система управления базами данных (СУБД) - это совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования БД многими пользователями. Пользователей СУБД можно разделить на три большие группы:

- разработчики базы данных отвечают за написание прикладных программ, использующих базу данных;
- конечные пользователи работают с базой данных либо через некоторую информационную систему, либо пользовательский интерфейс системы управления базами данных;
- администраторы базы данных обеспечивают поддержание работоспособности базы данных, сохранность и безопасность данных, настройку работы базы данных.

Базовой архитектурой СУБД является трехуровневая модель, в рамках которой СУБД представляется состоящей из внешнего, концептуального и внутреннего уровня. На внешнем уровне содержимое СУБД представляется так, как его видит отдельный пользователь. На концептуальном уровне данные СУБД описываются в обобщенной модели предметной области, для которой создается информационная система. Внутренний уровень - это собственно данные, расположенные в файлах. В ведении СУБД также находится словарь данных, содержащий сведения обо всем, что хранится в базе данных (имена всех таблиц и представлений, хранимых процедур и триггеров, учетные записи пользователей СУБД и т.п.). Сведения такого рода принято называть метаданными.

Реляционная модель данных основывается на математических принципах теории

множеств и математической логике. Эти принципы были впервые использованы для моделирования данных в 60-х годах прошлого века Коддом. Реляционная модель определяет, каким образом данные могут быть представлены (структура данных), каким образом данные могут быть защищены от некорректных изменений (целостность данных) и какие операции могут быть выполнены с данными (операции с данными).

2. Создание серверной части приложения в программе IBConsole. Основные составляющие окна программы IBConsole.

Создание однотабличной базы данных

1. Создайте структуру таблицы *Студенты.*

2. Создайте структуру таблицы *Дисциплины.*

3. Измените структуру таблицы *Преподаватели.*

4. Создайте структуру таблицы *Оценки.*

5. Разработайте схему данных, т.е. создайте связи между таблицами.

Создание форм для ввода данных в таблицы

1. Создайте форму *Студенты.*

2. Заполните данными таблицу *Студенты* посредством формы *Студенты.*

3. Создайте форму *Дисциплины.*

4. Заполните данными таблицу *Дисциплины* посредством формы *Дисциплины.*

5. Создайте форму *Оценки.*

6. Заполните данными таблицу *Оценки* посредством формы *Оценки.*

Формирование сложных запросов

1. Разработайте запрос с параметрами о студентах заданной группы, в котором при вводе в окно параметров номера группы (в примере это 151 или 152) на экран должен выводиться состав этой группы.

2. Создайте запрос, в котором выводятся оценки студентов заданной группы по заданной дисциплине.

3. Создайте перекрестный запрос, в результате которого создастся выборка, отражающая средний балл по дисциплинам в группах.

4. Разработайте запрос на увеличение на 10% заработной платы тех преподавателей, кто получает менее 5000 руб.

5. Создайте запрос на удаление отчисленных студентов.

6. Разработайте запрос на создание базы данных отличников.

7. Дня всех созданных вами запросов разработайте формы.

Создание сложных форм

1. Разработайте сложную форму, в которой с названиями дисциплин была бы связана подчиненная форма *Студенты* и подчиненная форма *Оценки студентов.*

2. Измените расположение элементов в форме в соответствии с рис.4.

3. Вставьте в форму диаграмму, графически отражающую оценки студентов.

4. Отредактируйте вид осей диаграммы.

Создание сложных отчетов

1. Создайте запрос, на основе которого будет формироваться отчет. В запросе должны присутствовать: из таблицы *Студенты* — поля «Фамилия», «Имя», «Отчество» и «Номер группы», из таблицы *Дисциплины* — поле «Название дисциплины», из таблицы *Оценки* — поле «Оценки».

2.Создайте отчет по итогам сессии. В отчете оценки студентов должны быть сгруппированы по номерам групп и дисциплинам. Для каждого студента должна вычисляться средняя оценка в сессию, а для каждой группы — среднее значение

3. Архитектура удаленных баз данных. Примеры использования триггера

При изучении вопроса необходимо обратить внимание на следующие

особенности:хранилища удаленных баз данных.

1.6 Тема 6 « Распределенные транзакции» (12 часов)

1.6.1 Перечень и краткое содержание рассматриваемых вопросов:

1. Распределенные транзакции

При разработке программного обеспечения достаточно часто возникает потребность получать извещения о каких-либо событиях, возникающих асинхронно, то есть в некоторые произвольные моменты времени. В распределенных системах так же может возникнуть необходимость использования таких извещений, получаемых от удаленной системы. Можно выделить два подхода к обработке событий – тесно связанные и слабо связанные события. При тесно связанном событии происходит прямое уведомление одной стороны другой стороной. Хотя этот метод можно использовать, например, вместе с однонаправленным асинхронным вызовом, ему свойственен ряд недостатков, ограничивающих его применение в распределенных системах:

- обе компоненты системы должны выполняться одновременно;
- для уведомления нескольких компонент об одном событии уведомляющей стороной должны использоваться механизмы для ведения списка получателей событий;
- затруднена фильтрация или протоколирование событий.

Транзакция – последовательность операций с какими-либо данными, которая либо успешно выполняется полностью, либо не выполняется вообще. В случае невозможности успешно выполнить все действия происходит возврат к первоначальным значениям всех измененных в течение транзакции данных (*откат* транзакции). *Транзакция* должна обладать следующими качествами.

- **Атомарность**. Транзакция выполняется по принципу "все или ничего".
- **Согласованность**. После успешного завершения или отката транзакции все данные находятся в согласованном состоянии, их логическая целостность не нарушена.
- **Изоляция**. Для объектов вне транзакции не видны промежуточные состояния, которые могут принимать изменяемые в транзакции данные. С точки зрения "внешних" объектов, до успешного завершения транзакции они должны иметь то же состояние, в котором находились до ее начала.
- **Постоянство**. В случае успешности транзакции сделанные изменения должны иметь постоянный характер (т.е. сохранены в энергонезависимой памяти).

Транзакции являются основой приложений, работающих с базами данных, однако в распределенной системе может быть недостаточно использования только транзакций систем управления базами данных. Например, в распределенной системе в транзакции может участвовать несколько распределенных *компонент*, работающих с несколькими независимыми базами.

.

2. Работа с программой Interactive SQL

Найти материал для доклада по дисциплинам «Строительные конструкции» и «Технологические основы строительного производства» и оформить средствами MSWord в соответствии с требованиям оформления творческих работ.

Настройка MSOutlook. Запуск программы. Окно Outlook и назначение информационных служб. Настройка параметров MSOutlook. Настройка порядка ввода имен для контактов. Настройка вида окна Outlook при запуске программы. Настройки для работы с почтой. Настройка параметров отправки/получения почты.

Работа с почтовым клиентом

Задание2 Создание нового сообщения. Ввод адресов. Вложение файлов. Вставка гиперссылок. Отправка сообщений. Подписи в исходящих сообщениях. Отправка копий сообщений. Работа с полученными сообщениями. Использование заметок. Создание новой

папки для сообщений. Перемещение сообщений в папку.

Задание 3. Управление контактами

Импорт контактной информации из другой программы. Создание контактов. Сохранение контактов. Дополнительные возможности для создания контактов. Просмотр и печать контактных данных. Создание нового сообщения из папки Контакты. Создание новой папки для контактов. Перемещение контактов в папку. Список рассылки. Использование цветовых категорий для элементов Outlook. Создание сообщения контактам, организованным по категориям. Создание сообщения контактам из одной папки.

Задание 4. Планирование персональной деятельности. Календарь MSOutlook

Основные понятия Календаря. Просмотр календаря. Создание встречи. Создание события и оповещения. Повторяющиеся встречи и события. Приглашение на собрание. Просмотр назначенных встреч и событий. Печать Календаря. Планирование персональной деятельности.

Задание 5. Организация поиска информации

Поиск с помощью упорядочения. Поиск контактов. Расширенный поиск. Создание папки поиска для сообщений. Мгновенный поиск. Создание правил для работы с сообщениями.

Задание 6. Копирование и восстановление данных

Сохранение и копирование данных. Экспорт данных. Автоархивация устаревших данных. Резервное копирование файлов Outlook. Ручная архивация. Восстановление данных.

3. Классификация транзакций

При изучении вопроса необходимо обратить внимание на следующие особенности: понятие транзакций.

1.7 Тема 7 «Технология DCOM. Развитие модели COM» (14 часов)

1.7.1 Перечень и краткое содержание рассматриваемых вопросов:

1. Технология DCOM. Развитие модели COM

Для того чтобы различные фрагменты сложного приложения могли работать вместе через Internet, необходимо обеспечить между ними надежные и защищенные соединения, а также создать специальную систему, которая направляет программный трафик. С начала 1990-х годов над решением этой задачи трудятся две конкурирующие группы разработчиков.

Одну из них - консорциум Object Management Group (OMG) - поддерживают компании IBM, Sun Microsystems и ряд других производителей. В **1991** году OMG предложил архитектуру распределенных вычислений, получившую название Common Object Request Broker Architecture (CORBA). Сегодня она применяется многими крупными организациями для развертывания распределенных вычислений в масштабах предприятия на базе Unix-серверов и мэйнфреймов.

Другое технологическое направление представляет Microsoft, создавшая распределенную компонентную объектную модель Distributed Component Object Model (DCOM), которая встраивается в операционные системы Windows.

DCOM - программная архитектура, разработанная компанией Microsoft для распределения приложений между несколькими компьютерами в сети. Программный компонент на одной из машин может использовать DCOM для передачи сообщения (его называют удаленным вызовом процедуры) к компоненту на другой машине. DCOM автоматически устанавливает соединение, передает сообщение и возвращает ответ удаленного компонента.

COM и DCOM - технологии, обеспечивающие взаимодействие между компонентами приложения и позволяющие развертывать распределенное приложение на платформе Windows. COM является моделью программирования на основе объектов, которая упрощает взаимодействие различных приложений и компонентов, а DCOM - это своего рода "клей", связывающий воедино разнообразные технологии, применяемые в распределенных приложениях. DCOM дает возможность двум или нескольким компонентам легко взаимодействовать друг с другом независимо от того, когда и на каком языке программирования они были написаны, а также где именно они находятся и в какой операционной системе работают.

Обе системы DCOM компании Microsoft и CORBA консорциума Object Management Group поддерживают распределенные вычисления. Однако, две эти технологии развиваются в разных направлениях.

Microsoft расширила DCOM, добавив службы обработки транзакций, упростив программирование распределенных приложений и усовершенствовав поддержку Unix и других платформ.

OMG расширяет свою компонентную модель за счет служб, ориентированных на конкретные отрасли, то есть телекоммуникации, производство, электронную коммерцию, финансы, медицину, транспорт и коммунальные услуги.

2. Проектирование и разработка клиентской части приложения. Применение компонент страниц BDE и IBX

С помощью редактора MSPowerPoint создать презентацию о программном обеспечении информационных технологий.

В качестве темы первой презентации возьмем электронную иллюстрацию выступления, касающегося структур построения курса лекций по изучению MicrosoftOffice.

Этот процесс подготовки презентации придется разбить на два этапа:

1. непосредственная разработка презентации, т. е. оформление каждого слайда;
2. *демонстрация, т. е. процесс показа готовых слайдов, который может сопровождаться пояснениями лектора, некоторыми графическими пометками по ходу демонстрации.*

1. Откройте созданный вами ранее файл (PP_Иванов)
2. Демонстрация:

3. Отказоустойчивость

При изучении вопроса необходимо обратить внимание на следующие особенности: способы определения отказоустойчивости.

1.8 Тема 8 «Управление жизненным циклом объекта» (14 часов)

1.8.1 Перечень и краткое содержание рассматриваемых вопросов:

1. Управление жизненным циклом объекта

Сложные инженерные системы (атомные электростанции, оффшорные буровые платформы, вертолёты и т.д.) проходят жизненный цикл, занимающий десятки лет – от замысла до вывода из эксплуатации. За это время инженерная система проходит множество различных состояний: существует как набор презентационных документов для инвесторов и потенциальных пользователей, многотомных детальных требований, часть из которых существует в виде обязательного отраслевого регулирования, архитектуры (эскизного проекта), рабочей документации типового проекта, свежеизготовленных комплектующих и жидкого бетона, эксплуатируемой и обслуживаемой затем десятки лет системы «в металле и бетоне», но и после этого система продолжает существовать -- в виде мусора и лома.

«Управление жизненным циклом» -- это термин, которым люди пытаются обозначить практику обеспечения связности всех этих состояний системы, как в прямом направлении (например, передача рабочей документации на стадию сооружения), так и в обратном направлении (например, учет данных по надёжности аналогичных эксплуатируемых уже систем на стадии проектирования новых).

Слово "продукт" из PLM в СУЖЦ пропало не случайно, ибо речь идет о сложных иненерных объектах – если вертолёт еще можно с натяжкой назвать «продуктом» в силу его серийности, то атомная станция «продуктом» обычно уже не называется. Поэтому "жизненным циклом" какой именно системы тут «управляется», нужно уточнять в каждом конкретном случае. «Система УЖЦ системы» немного запутывает, но именно это и имеется ввиду. Чтобы не слишком путаться, в название этой статьи я вынес «СУЖЦ управления сложным инженерным объектом», но «сложный инженерный объект» -- это просто расшифровка слова «система».

Коллизии -- это противоречия (несоответствия) одних частей целевой системы другим, независимо от того, в каком состоянии по мере прохождения по жизненному циклу находится целевая система. Противоречия неизбежно появляются при коллаборативной разработке проекта и сооружении, ибо разные участники разработки и сооружения целевой системы действуют в ситуации неполной информации как о целевой системе и ее текущем (или прогнозируемом далее по жизненному циклу) состоянии, так и о действиях и целях друг друга, так и об ожиданиях пользователей и систем в операционном окружении уже развёрнутой для эксплуатации системы.

Коллизии могут найтись на любой стадии жизненного цикла системы, например:

- на стадии замысла оценки стоимости могут не соответствовать текущей предполагаемой конструкции, а реализуемые разработчиками-контракторами функции противоречить действительным нуждам заказчика-пользователя;
- на стадии проектирования предполагаемое комплектующее оборудование может не соответствовать имеющимся типам из имеющегося у закупщиков каталога, или комплектующее оборудование может быть пропущенным на части чертежей, ведомостей, инженерных обоснований, или оказаться запроектированным внутри стены -- ибо все эти чертежи, ведомости, обоснования делаются разными людьми, часто в разных организациях и в разное время,
- на стадии строительства арматура может оказаться еще не закуплена, или закуплена неправильная, или установлена на не своё место, или опоры не выдержат нагрузки после заполнения трубопроводов, ибо были выбраны неправильно, или три бригады выйдут на работу одновременно в одном тесном помещении из-за ошибки в планировании работ.
- на стадии эксплуатации может оказаться, что забыли запроектировать входную дверь. Увы, это даже не шутка: один из классических примеров ошибок в системной инженерии – это реально сданный в эксплуатацию высокотехнологичный почтамт, в котором въезд для погрузки-разгрузки грузовиков с почтой был невозможен: при проектировании забыли узнать, какой высоты эти грузовики, а они оказались слишком большие для оставленной им въездного проёма в железобетонной конструкции почтамта.

2. Управление доступом к данным в таблицах InterBase с помощью SQL-запросов**»**

Получение навыков создания графической документации строительного цикла.

Выполните сборочный чертеж «Виды соединений» и спецификацию.

1. Создайте документ-чертеж и сохраните его в свою папку под именем *ПР13.cdw.*

2. По предварительно выполненному чертежу начертите заданные пластины, рис. 13.3. Проверьте подключение спецификации (***Сервис – Объекты спецификации – Включить работу со спецификацией***).

3. Выполните болтовое соединение. По данному варианту номинальный диаметр болта **М20.** Толщина пакета по чертежу равна **35 мм** (толщину пакета можно измерить). Подключите ***Конструкторскую библиотеку***, если она не подключена (***Сервис***). Выберите раздел ***Крепежный элемент***.

Появится окно с параметрами пакетаАктивизируйте **Набор элементов**. Выберите набор болтового соединения. Если отсутствует данный набор, активизируйте **Все элементы** и создайте набор самостоятельно, введя в верхнее окно болт, а в нижнее – сначала шайбу, а затем гайку.

Создание спецификации.

- В составе системы поставляются стили спецификаций, правила которых соответствуют ГОСТ2.108-68 и ГОСТ2.113-75.
- Спецификация может быть составлена на основе уже готового сборочного чертежа или создаваться независимо от сборочного чертежа (например, параллельно вычерчиванию сборки). Составленную таким образом спецификацию на любом этапе работы с ней можно синхронизировать со сборочным чертежом.
- При вставке в чертеж стандартных изделий из конструкторской библиотеки их обозначение формируется и вносится в спецификацию автоматически.
- Осуществляется двунаправленная ассоциативная связь между спецификацией и соответствующими ей чертежами. Благодаря наличию этой связи изменения в сборочном чертеже или деталировке автоматически отражаются в спецификации.
- Если спецификация и связанный с ней сборочный чертеж открыты одновременно, возможен режим работы, в котором при выделении строки спецификации подсвечиваются соответствующие ей геометрические объекты и линии – выноски. Расположив рядом окна спецификации и сборочного чертежа, можно быстро найти на чертеже изображение любого внесенного в спецификацию объекта.
- Система проектирования спецификаций предполагает два режима работы: ручной и полуавтоматический.

В полуавтоматическом режиме модуль проектирования спецификаций устанавливает связи между спецификацией, листом или листьями сборочного чертежа и рабочими чертежами деталей. Основная идея, реализованная в модуле проектирования спецификаций при работе в полуавтоматическом режиме, заключается в том, что конструктор не создает сразу спецификацию, как отдельный документ. Конструктор создает новую пустую спецификацию и устанавливает связи между ней и всеми листами чертежей, относящихся к сборочной единице. При этом информация об объектах спецификации из подключенных листов чертежей передается в спецификацию, разносится по ее разделам и сортируется (**объект спецификации** – строка или несколько следующих друг за другом строк спецификации, относящихся к одному материальному объекту). Объекты спецификации бывают **базовые и вспомогательные**. Для базовых объектов предусмотрена возможность автоматического заполнения колонок, сортировка объектов внутри раздела, подключение графических объектов из сборочного чертежа и т.д.

3. Разделение приложений по уровням

При изучении вопроса необходимо обратить внимание на следующие особенности: виды приложений.

1.9 Тема 9 «Распределенные файловые системы. Файловая система NFS. Семантика совместного использования файлов. Проблема отказов» (10 часов)

1.9.1 Перечень и краткое содержание рассматриваемых вопросов:

1. Распределенные файловые системы. Файловая система NFS. Семантика совместного использования файлов. Проблема отказов

Появившаяся в 70-х годах возможность объединения компьютеров в единую сеть произвела революцию в компьютерной промышленности. Эта возможность прежде всего вызвала желание организовать разделение доступа к файлам между различными компьютерами. Первые достижения в этой области были ограничены возможностью копирования целых файлов из одной машины в другую. В качестве примера можно указать программу UNIX-to-UNIX copy (uucp) и File Transfer Protocol (ftp). Однако эти решения не позволяли даже близко подойти к реализации доступа к файлам на удаленной машине, по своим возможностям напоминающего доступ к файлам на локальных дисках.

Только в середине 80-х годов появилось несколько распределенных файловых систем, которые обеспечили прозрачный доступ по сети к удаленным файлам. Это были Network File System (NFS) компании Sun Microsystems (1985), Remote File Sharing system (RFS) компании AT&T (1986) и Andrew File System (AFS) университета Карнеги-Меллона (1995). Эти три системы резко отличались друг от друга по целям разработки, архитектуре и семантике, хотя все они пытались решить одну и ту же фундаментальную проблему. Сегодня RFS доступна практически на всех системах, базирующихся на UNIX System V. Разработка ASF перешла корпорации Transarc, в которой она была развита и превращена в Distributed File System (DFS) - компоненнт распределенной вычислительной среды DSE (Distributed Computing Environment) Open Software Foundation. Но наибольшее распространение получила NFS, которая поддерживается на всех UNIX и многих "не UNIX" системах.

Компания Sun Microsystems представила NFS в 1985 году как средство обеспечения прозрачного доступа к удаленным файловым системам. Помимо публикации протокола Sun лицензировала его базовую реализацию, которая была использована различными поставщиками для портирования NFS на разные операционные системы. С тех пор NFS стала фактически промышленным стандартом, который поддерживается действительно всеми вариантами системы UNIX, а также некоторыми другими системами, например, VMS и MS-DOS.

Архитектура NFS базируется на модели клиент-сервер. Файл-сервер представляет собой машину, которая экспортирует некоторый набор файлов. Клиентами являются машины, которые имеют доступ к этим файлам. Одна машина может для различных файловых систем выступать как в качестве сервера, так и в качестве клиента. Однако программный код NFS разделен на две части, что позволяет иметь только клиентские или только серверные системы.

Клиенты и серверы взаимодействуют с помощью удаленных вызовов процедур (rpc - remoteprocedurecall), которые работают как синхронные запросы. Когда приложение на клиенте пытается обратиться к удаленному файлу, ядро посылает запрос в сервер, а процесс клиента блокируется до получения ответа. Сервер ждет приходящие запросы, обрабатывает их и отсылает ответы назад клиентам.

2. Выборка данных. Команда Select

Найти материал, необходимые для выполнения дипломного задания и оформить средствами MSWord в соответствии с требованиям оформления творческих работ.

1. Загрузить персональный компьютер
2. Загрузить браузер
3. Зайти на Сайт-поисковик.
4. Путешествуя по ссылкам в поисковике и по другим сайтам, выбрать интересующую информацию и скопировать в MSWord.
5. Подготовленный к оформлению материал оформить в соответствии с требованиями и распечатать на принтере.

Произвести поиск сайтов в наиболее популярных поисковых системах общего назначения в русскоязычном Интернете (Рунете).

Краткая справка. Наиболее популярными русскоязычны ми поисковыми

системами являются:

Rambler — www.rambler.ru;

Апорт — www.aport.ru;

Япс1ех — www.yandex.ru.

Англоязычные поисковые системы:

Yahoo — www.yahoo.com.

Специализированные поисковые системы позволяют искать информацию в специализированных слоях Интернета. К ним можно отнести поиск файлов на серверах FTP и систему поиска адресов электронной почты WhoWhere.

Произвести поиск в интернет-энциклопедии Кирилла и Мефодия.

Произвести поиск по нескольким поисковым серверам.

Краткая справка. Мегапоисковый инструмент — это программа, которая посылает ваш запрос сразу на несколько серверов, а затем собирает наиболее вероятные источники необходимой вам информации на одной странице. Один из лучших поисковых инструментов в WWW — сервер SavvySearch (в переводе с англ. — Поиск здравого смысла) (http:/www.savvysearch.com).