Введение 3

1 Исследовательский раздел 5

Введение 5

1.1 Теория построения Автоматизированных Рабочих Мест 6

1.1.1 Анализ принципов АРМ на базе ПК. 6

1.1.2 Языковые средства АРМ 9

1.1.3 Классификация АРМ-ов. 13

1.2 Теория современных систем управления базами даных 17

1.2.1 Концепция баз данных 17

1.2.2 Архитектура СУБД 18

1.2.3 Инфологическая модель данных "Сущность-связь" 20

1.2.4 Реляционная структура данных 21

1.2.5 О нормализации, функциональных и многозначных зависимостях 23

1.3 Технология Клиент-Сервер. 26

1.3.1 Основные понятия 26

1.3.2 Преимущества архитектуры клиент-сервер 26

1.3.3 Компоненты архитектуры Клиент-сервер 28

1.4 Язык структурированных запросов SQL 31

1.4.1 Назначение и принцип работы SQL 31

1.4.2 Достоинства языка SQL 32

1.4.3 Запрос на языке SQL 35

1.4.4 Агрегатные функции 36

1.5 Локальные вычислительные сети 38

1.5.1 Файл сервер и рабочие станции 38

1.5.2 Операционная система рабочей станции 39

1.5.3 Преимущества локальных вычислительных сетей 40

1.5.4 Стандарт передачи информации 40

1.5.5 Сетевые устройства и средства коммуникаций. 44

1.5.6 Витая пара. 45

1.5.7 Широкополосный коаксиальный кабель. 45

1.5.8 Еthernet-кабель. 46

1.5.9 Оптоволоконные линии. 46

1.5.10 Топологии вычислительной сети. 46

1.5.11 Топология типа звезда. 47

1.5.12 Кольцевая топология. 48

1.5.13 Шинная топология. 49

1.5.14 Методы доступа и протоколы передачи данных 51

1.5.15 Локальная сеть Token Ring 51

1.5.16 Локальная сеть Arknet. 52

1.5.17 Локальная сеть Ethernet 53

1.6 Сетевые операционные системы для локальных сетей. 54

1.6.1 NetWare 3.11, Nowell Inc. 56

1.6.2 LAN Server, IВМ Согр. 56

1.6.3 Windows NT Advanced Server 3.1, Microsoft Corp. 57

1.6.4 NetWare 4, Nowell Inc. 58

1.7 Среда Delphi как средство разработки ПО баз даных. 60

1.7.1 Высокопроизводительный компилятор в машинный код 63

1.7.2 Мощный объектно-ориентированный язык 64

1.7.3 Объектно-ориентированная модель программных компонент 65

1.7.4 Библиотека визуальных компонент 66

1.7.5 Формы, модули и метод разработки “Two-Way Tools” 71

1.7.6 Масштабируемые средства для построения баз данных 72

1.7.7 Настраиваемая среда разработчика 73

1.7.8 SQL сервер InterBase 5.1.1 75

Заключение 76

2 Методический раздел 77

Введение 77

2.1.1 Назначение АРМ «Отдел Кадров» 78

2.1.2 Составные части программы 79

2.1.3 Права доступа. Вход в программу 79

2.1.4 Справочники 81

2.1.5 Личные карточки 82

2.1.6 Архив 83

2.1.7 Отчеты 83

2.1.8 Дополнительные функции 84

2.1.9 Установка и настройка программы 84

2.1.10 Особенности реализации поставленной задачи 87

2.1.11 Экранные формы 89

Заключение 90

3 Безопасность жизнедеятельности 91

Введение 91

3.1 Анализ опасных и вредных факторов, возникающих при работе с компьютером 92

3.2 Мероприятия по предотвращению и уменьшению влияния вредных факторов 94

3.2.1 Нормирование искусственного и естественного освещения. 94

3.2.2 Основные требования к искусственному освещению в производственном помещении. 94

3.2.3 Расчёт искусственного освещения 95

3.2.4 Рациональная планировка рабочих мест. 97

3.3 Утилизация и переработка ртути в люминесцентных лампах 98

3.3.5 Химический состав и удельный расход демеркуризационных растворов 101

Заключение 102

4 Приложения 104

4.1.1 Листинг основного файла-проекта 104

4.1.2 Листинг созданного VCL-компонента. 106

4.1.3 Модуль блокировки программы от несанкционированного доступа 108

5 Список использованных источников 110

# 

# Введение

Темой данной дипломной работы является Автоматизированное Рабочее Место (АРМ) отдела кадров предприятия. Результатом работы является программа «АРМ Отдел кадров», применяемая до настоящего времени на Московском Заводе Радиоаппаратуры (МЗР) «ОРБИТА».

Работа отдела кадров достаточно крупного предприятия (в данном случае более 500 человек) связана с накоплением большого количества информации о личных данных сотрудников. Традиционно информация храниться на бумажных носителях. При этом трудно осуществить быстрый отбор нужных данных при приеме на работу, уходе в отпуск, увольнении, переходе на другую должность или других перемещениях сотрудника.

Немаловажен вопрос надежности хранения и конфиденциальности личных данных о работающих на предприятии. При указанной численности штата предприятия отдел кадров состоит из начальника отдела кадров и нескольких сотрудников отдела кадров. Фактически, только начальник отдела кадров должен иметь полный доступ к архиву хранящейся информации.

Таким образом, автоматизация процесса работы отдела кадров является нужным и перспективным процессом. До написания данного диплома на МЗР ОРБИТА использовалась написанная ранее на Cliper-e база данных АРМ-ОК-94. Однако к моменту преддипломной практики накопились замечания по недостаткам программы. Вот список основных недостатков, которые отмечались как критические и требовали устранения в первую очередь:

1. Программа не разграничивала права доступа. Фактически любой, имеющий доступ к компьютеру с установленной программой мог просмотреть данные о личных сотрудников. Единственным препятствием являлось расположение компьютеров в комнате отдела кадров.

2. Программа использовала большое количество файлов для хранения данных, которые хранились на сервере. При этом при одновременной работе нескольких пользователей скорость падала ниже разумно допустимой. Передача других данных по локальной сети также существенно снижалась.

3. Интерфейс программы абсолютно не рассчитан на длительную работу оператора с ПЭВМ. Слишком контрастные цветовые настройки диалогов ввода информации, а также выдача звукового сигнала после удачного ввода информации сильно утомляло и вызывало раздражение пользователей.

4. Вид распечатываемых отчетов не соответствовал виду стандартных документов.

5. Общая методологическая организация связей между разными формами ввода и подразумеваемым результатом плохо продумана. Т.е. интерфейс нельзя назвать «интуитивно-понятным».

Устранить вышеописанные недостатки и было целью дипломной работы. Более кратко требования к новой реализации АРМ можно обозначить так:

1. Разграничение прав доступа к информации.
2. Минимальная нагрузка на локальную вычислительную сеть.
3. При выводе на печать отчеты должны быть максимально похожими на стандартные бланки.
4. Удобный интерфейс, интуитивно-понятные связи между диалогами ввода информации.

С учетом имеющихся знаний относительно построения баз данных с помощью SQL технологии и архитектуры «клиент-сервер» было принято решение написать новую программу с учетом приведенных требований. Данные средства реализации являются перспективными и позволили решить поставленную задачу в соответствии с требованиям к программе.

Созданная программа «АРМ Отдел кадров» является перспективной разработкой, так как на основе уже хранящейся информации о сотрудниках работают отдел бухгалтерии и учета материальных ресурсов завода МЗР ОРБИТА. Таким образом, не исключено создание другими разработчиками новых версий данной программы для автоматизации отделов бухгалтерии и материальной части.

# Исследовательский раздел

## Введение

После анализа поставленной задачи стало необходимо изучить предметную область, в которой решалась задача создания АРМ Отдела Кадров. Для это первым этапом стало изучение и понимание основных принципов построения автоматизированных рабочих мест как таков. Как выяснилось в ходе изучения этой области, проработке интерфейса стоит уделить немалое внимание. Плохо продуманный интерфейс может свети не нет желание работать с программой независимо от примененных аппаратно-программных средств для ее реализации.

С учетом применения программы на уже существующей локальной вычислительной сети следующим шагом стало изучения принципов их построения. Понимание принципов организации сетей в настоящее время можно отнести к элементам компьютерной грамотности. Необходимость данных знаний усиливается при применении технологии клиент-сервер и языка SQL.

Далее возникла задача правильно выбрать конкретное средство реализации поставленной цели. С учетом имеющихся знаний относительно системы разработки программ с среде Delphi, было решено создать программу именно на ее основе. Однако ряд вопросов требовал более глубокого изучения, что и было проделано при исследовании задачи.

## Теория построения Автоматизированных Рабочих Мест

### Анализ принципов АРМ на базе ПК.

Автоматизированное рабочее место (АРМ) , или, в зарубежной терминологии, "рабочая станция" (work-station), представляет собой место пользователя-специалиста той или иной профессии, оборудованное средствами, необходимыми для автоматизации выполнения им определенных функций. Такими средствами, как правило , является ПК, дополняемый по мере необходимости другими вспомогательными электронными устройствами, а именно: дисковыми накопителями, печатающими устройствами, оптическими читающими устройствами или считывателями штрихового кода, устройствами графики, средствами сопряжения с другими АРМ и с локальными вычислительными сетями и т.д.

Hаибольшее распространение в мире получили АРМ на базе профессиональных ПК с архитектурой IBM PC.

АРМ в основном ориентированы на пользователя, не имеющего специальной подготовки по использованию вычислительной техники. Основным назначением АРМ можно считать децентрализованную обработку информации на рабочих местах, использование соответствующих "своих" баз данных при одновременной возможности вхождения в локальные сети АРМ и ПК, а иногда и в глобальные вычислительные сети, включающие мощные ЭВМ.

В настоящее время на очень многих предприятиях реализуется концепция распределенных систем управления народным хозяйством. В них предусматривается локальная, достаточно полная и в значительной мере законченная обработка информации на различных уровнях иерархии. В этих системах организуется передача снизу вверх только той части информации, в которой имеется потребность на верхних уровнях. При этом значительная часть результатов обработки информации и исходные данные должны храниться в локальных банках данных.

Для реализации идеи распределенного управления потребовалось создание для каждого уровня управления и каждой предметной области автоматизированных рабочих мест на базе профессиональных персональных компьютеров. Например, в сфере экономики на таких АРМ можно осуществлять планирование, моделирование, оптимизацию процессов, принятие решений в различных информационных системах и для различных сочетаний задач. Для каждого объекта управления необходимо предусматривать АРМ, соответствующие их значению. Однако принципы создания любых АРМ должны быть общими:

* системность.
* гибкость.
* устойчивость.
* эффективность.

Поясним смыл каждого из указанных понятий.

*Системность.* АРМ следует рассматривать как системы, структура которых определяется функциональным назначением.

*Гибкость.* система приспособлена к возможным перестройкам, благодаря модульности построения всех подсистем и стандартизации их элементов.

*Устойчивость.*  Принцип заключается в том, что система АРМ должна выполнять основные функции независимо от воздействия на нее внутренних и внешних возмущающих факторов. Это значит, что неполадки в отдельных ее частях должны быть легко устраняемы, а работоспособность системы быстро восстанавливаема.

*Эффективность* АРМ следует рассматривать как интегральный показатель уровня реализации приведенных выше принципов, отнесенного к затратам на создание и эксплуатацию системы.

Функционирование АРМ может дать желаемый эффект при условии правильного распределения функций и нагрузки между человеком и машинными средствами обработки информации, ядром которой является компьютер.

Создание такого "гибридного" интеллекта в настоящее время является проблемой. Однако реализация этого подхода при разработке и функционировании АРМ может принести ощутимые результаты - АРМ станет средством повышения не только производительности труда и эффективности управления, но и социальной комфортности специалистов. При этом человек в системе АРМ должен оставаться ведущим звеном.

Hа производственных предприятиях АРМ являются важной структурной составляющей АСУ как персональное средство планирования, управления, обработки данных и принятия решений. АРМ - это всегда специализированния система, набор технических средств и программного обеспечения, ориентированного на конкретного специалиста - администратора, экономиста, инженера, конструктора, проектанта, архитектора, дизайнера, врача, орга­низатора, исследователя, библиотекаря, музейного работника и множество других. ­ ­

В то же время к АРМ любой "профессии" можно предъявить и ряд общих требований, которые должны обеспечиваться при его создании, а именно:

непосредственное наличие средств обработки информации;

возможность работы в диалоговом (интерактивном) режиме;

выполнение основных требований эргономики: рациональ­ное распределение функций между оператором, элементами комп­лекса АРМ и окружающей средой, создание комфортных условий ра­боты, удобство конструкций АРМ, учет психологических факторов человека-оператора, привлекательность форм и цвета элементов АРМ и др.;

достаточно высокая производительность и надежность ПК, работающего в системе АРМ;

адекватное характеру решаемых задач программное обеспечение;

максимальная степень автоматизации рутинных процессов;

оптимальные условия для самообслуживания специалистов как операторов АРМ;

другие факторы, обеспечивающие максимальную комфортность и удовлетворенность специалиста использованием АРМ как рабочего инструмента.

Структура АРМ включает совокупность подсистем - технической, информационной, программной и организационной.

О технической подсистеме уже было сказано выше. К указанному ранее набору технических средств, непосредственно образующему АРМ, надо еще добавить средства связи с другими АРМ, работающими в общей сети объекта, а также другие средства связи (телефон, телекс, телефакс).

К информационной подсистеме относятся массивы информации, хранящейся в локальных базах данных, как правило, на дисковых накопителях. Сюда же относится и системы управления базами данных.

Программное обеспечение включает операционные системы, сервисные программы, стандартные программы пользователей и пакеты прикладных программ, выполненные по модульному принципу и ориентированные на решение определенного класса задач, обусловленного назначением АРМ. По мере необходимости в программное обеспечение включаются также пакеты программ для работы с графической информацией.

Организационное обеспечение АРМ имеет своей целью организацию их функционирования, развития, подготовки кадров, а также администрирования. К последнему относятся: планирование работы, учет, контроль, анализ, регулирование, документальное оформление прав и обязанностей пользователей АРМ.

Если устройство АРМ достаточно сложно, а пользователь не имеет специальных навыков, возможно применение специальных обучающих средств, которые позволяют постепенно ввести пользователя в среду его основного автоматизированного рабочего места. При реализации функций АРМ (т.е. собственно его функционировании) необходимы методики определения цели текущей деятельности , информационной потребности, всевозможных сценариев для описания процессов ее реализации.

Методика проектирования АРМ не может не быть связанной с методикой его функционирования, так как функционирование развитого АРМ предусматривает возможность его развития самими пользователями. Языковые средства АРМ являются реализацией методических средств с точки зрения конечного пользователя, а программные реализуют языковые средства пользователя и дают возможность конечному пользователю выполнять все необходимые действия.

### Языковые средства АРМ

Языковые средства АРМ необходимы прежде всего для однозначного смыслового соответствия действий пользователя и реакции ПЭВМ. Без них невозможен процесс обучения, организация диалога, обнаружение и исправление ошибок. Сложность разработки таких языков заключается в том, что они должны быть преимущественно непроцедурными. Если процедурный язык указывает, как выполняется задаваемое действие, то непроцедурный - что необходимо выполнить без детализации, какие действия для этого требуются. Так как конечные пользователи не знают и не должны знать в деталях процесс реализации информационной потребности, чем выше интеллектуальность АРМ, тем больше непроцедурных возможностей должно быть предусмотрено в его языках.

Языки АРМ должны быть и пользовательски-ориентированными, в том числе и профессионально-ориентированными. Это связано с различиями в классификации пользователей, которые разделяются не только по профессиональной принадлежности, но и по иерархии служебного положения, мере обученности, виду потребляемых данных и др. Следует учесть, что использование естественного языка, несмотря на кажущуюся простоту такого подхода, не может дать сколько-нибудь ощутимых преимуществ из-за необходимости введения через клавиатуру громоздких конструкций ради получения иногда несложных результатов.

Как и во всяком языке, основу языков АРМ должны составлять заранее определяемые термины, а также описания способов с помощью которых могут устанавливаться новые термины, заменяя или дополняя существующие. Это приводит к необходимости при проектировании АРМ определенным образом классифицировать терминологическую основу АРМ , т.е. определить все основные синтаксические конструкции языка и семантические отношения между терминами и их совокупностями. В связи с этим может возникнуть необходимость в простейшей классификации АРМ, например, по возможностям представления данных в некоторых пользовательских режимах обработки: числовые, текстовые, смешанные. В более сложных случаях классификация АРМ может определяться уже организацией баз данных. Возможности языка во многом определяют и список правил, по которым пользователь может строить формальные конструкции, соответствующие реализации информационной потребности. Hапример, в некоторых АРМ все данные и конструкции фиксируются в табличной форме (табличные АРМ) или в виде операторов специального вида (функциональные АРМ).

Языки пользователя разделяют АРМ также по видам диалога. Средства поддержки диалога в конечном счете определяют языковые конструкции, знание которых необходимо пользователю.

Конструкцией одного и того же АРМ может быть предусмотрено не один, а несколько возможных типов диалога в зависимости от роста активности пользователя в процессе обучения или работы, а также необходимости развития АРМ средствами пользователя. Из существующих диалогов при разработке АРМ наиболее употребимы: диалог, инициируемый пЭВМ, диалог заполнения форм, гибридный диалог, диалог необученного пользователя и диалог с помощью фиксированных кадров информации. При диалоге, инициируемом пЭВМ, пользователь АРМ освобождается практически полностью от изучения мнемоники и конструкций языка. Одной из модификаций этого метода является метод меню, при котором выбирается один или несколько из предложенных пЭВМ вариантов.

При диалоге заполнения форм, который также инициируется пЭВМ, пользователь заполняет специально подобранные формы на дисплее с их последующим анализом и обработкой.

Гибридный диалог может быть инициированы и пользователем, и ПЭВМ.

При диалоге необученного пользователя должна быть обеспечена полная ясность ответов пЭВМ, которые не могут оставлять у пользователя сомнений относительно того, что ему нужно делать.

В случае диалога с помощью фиксированных кадров информации пЭВМ выбирает ответ из списка имеющихся. В этом случае пользователь вводит только очень короткие ответы, а основная информация выдается автоматически.

Тип диалога также может определять классификацию АРМ, например АРМ с диалоговыми средствами необученного пользователя. Классификация АРМ по такому признаку связана с классификаццией по профессиональной ориентации пользователя. Hапример, АРМ с диалогом по методу меню вряд ли целесообразно для пользователя-экономиста, относящегося в то же время к персоналу руководителя, вследствие большого числа повторяющихся операций.

Если рассматривать автоматизированниые рабочие места с точки зрения программных средств, их реализующих, то классификация АРМ может быть весьма обширна. Они могут быть классифицированы по языку программирования, возможности предоставления пользователю процедурных средств программирования, возможности достраивания программной системы в процессе эксплуатации, наличию систем управления базами данных, транслятора или интерпретатора с языков пользователей, средств обнаружения и исправления ошибок и т.д. Пакеты прикладных программ (ППП), применяемые в АРМ, могут быть параметризованы для обеспечения привязки системы к конкретному приложению. Могут использоваться генераторы самих ППП.

В состав АРМ обязательно входят различные программные компоненты, обеспечивающие основные расчетные функции и организацию диалога, а также система управления базой данных, трансляторы, справочные системы, собственно база данных, содержащая, например, основные данные, сценарии диалога, инструкции, управляющие параметры, перечни ошибок и др. Основные компоненты АРМ определяют его состав и обеспечивать возможность классификации АРМ по различным признакам.

В зависимости от применения в рамках АРМ средств, обеспечивающих развитие АРМ конечным пользователем, будем разделять АРМ на два больших класса : *обслуживащюие* и *интеллектуальные*. И те и другие могут предназначаться для различных пользователей. Hо в то же время существуют такие пользователи, о которых можно сказать заранее, что он не может быть пользователем того или другого АРМ. Hапример, обслуживающий персонал (делопроизводители, секретари) в силу специфики выполняемых ими функций не нуждаются в интеллектуальных АРМ (в своей непосредственной деятельности).

Обслуживающие АРМ в сферах организационного управления могут быть :

* информационно-справочными.
* вычислительными.
* текстообрабатывающими.

Интеллектуальные АРМ можно прежде всего разделить на ориентированные на данные и ориентированные на занания (*даталогические и фактологические*).Информационно-справочные АРМ обслуживают какой-либо процесс управления. Вычислительные АРМ разнообразны по своему содержанию и могут применяться многочисленными категориями пользователей. С их помощью могут ставиться и решаться организационно-экономические задачи, связанные и не связанные друг с другом, поиск и обработка данных в которых заранее определена или определяется в процессе функционирования АРМ. Текстообразующие АРМ предназначены для обработки и генерации текстовой информации различной структуры и предположении, что текст семантически не анализируется.

Интеллектуальные АРМ даталогического типа основаны на широком использовании баз данных и языков пользователей. При этом пользователь способен самостоятельно модифицировать базы данных и языки, варьировать диалоговыми возможностями. В этих АРМ отсутствует база знаний, т.е. невозможно накопление правил, обеспечивающих объяснение того или иного свойства управляемого объекта. База знаний как составной компонент входит в АРМ фактологического типа. Фактологические АРМ полезны там, где работа в условиях АРМ определяется преимущественно накапливаемым опытом и логическим выводом на его основе.

Выделим несколько основных функций, которые должны быть реализованы в рамках автоматизации организационного управления:

интерпретация (анализ и описание данных и фактов из предметной области для установления их взаимосвязей и систем);

диагностика (поиск, определение и описание состояния управляемого объекта);

мониторинг (непрерывное отслеживание функционирования АРМ и фиксирование получаемых результатов);

планирование (обеспечение заданной последовательности действий);

проектирование (обеспечение пользовательских интерфейсов и развития).

### Классификация АРМ-ов.

АРМ могут быть индивидуальными, групповыми, коллективными. Применительно к групповым и коллективным АРМ в целях эффективного функционирования системы ЭВМ - специалистам (коллективу) необходимо ужесточить требования к организации работы АРМ и чётко определить функции администрирования в такой системе. Система АРМ, являющаяся «человеком – машиной», должна быть открытой, гибкой, приспособленной к постоянному развитию и совершенствованию. В такой системе должны быть обеспечены:

* максимальная приближённость специалистов к машинным средствам обработки информации;
* работа в диалоговом режиме;
* оснащение АРМ в соответствии с требованиями эргономики;
* высокая производительность компьютера;
* максимальная автоматизация рутинных процессов;

моральная удовлетворенность специалистов условиями труда,стимулирующая их творческую активность, в частности, в дальнейшем развитии системы;

* возможность самообучения специалистов.

Задачи, решаемые на АРМ, условно можно разделить на информационные и вычислительные.

К информационным задачам относятся кодирование, классификация, сбор, структурная организация, корректировка, хранение, поиск и выдача информации. Часто информационные задачи включают несложные вычислительные и логические процедуры арифметического и текстового характера и отношения (связи). Информационные задачи являются, как правило, наиболее трудоемкими и занимают большую часть рабочего времени специалистов.

Вычислительные задачи являются как формализуемыми , так и не полностью формализуемыми. Формализуемые задачи решаются на базе формальных алгоритмов и делятся на две группы: задачи прямого счета и задачи на основе математических моделей. Задачи прямого счета решаются с помощью простейших алгоритмов. Для более сложных задач требуется применять различные математические модели.

В последнее время большое внимание выделяется разработке средств решения не полностью формализуемых задач, называемых сематическими. Такие задачи возникают очень часто в ходе оперативного управления экономическими объектами, особенно при принятии решений в условиях неполной информации. Структура АРМ - это совокупность его подсистем и элементов. К обеспечивающим системам в первую очередь следует отнести: техническое, информационное, программное и организационное. Кроме того, существует целый ряд подсистем.

Техническое обеспечение представляет собой комплекс технических средств, основой которого служит профессиональный персональный компьютер, предусматривающий работу специалиста без посредников (программистов, операторов и др.). У групповых АРМ таким компьютером могут пользоваться 4 - 6 человек. В комплект профессионального персонального компьютера входят процессор, дисплей, клавиатура, магнитные накопители информации, печатающие устройства и графопостроители.

К комплексу технических средств следует отнести и средства коммуникаций для связи различных АРМ в сетях, а также средства телефонной связи.

Информационное обеспечение - это массивы информации, хранящиеся в локальных базах данных. Информация организуется и хранится, в основном, на магнитных дисках. Управление ею осуществляется с помощью программной системы управления базами данных, которая производит запись информации, поиск, считывание, корректировку и решение информационных задач. В АРМ может быть несколько баз данных.

Организационное обеспечение включает средства и методы организации функционирования, совершенствования и развития АРМ, а также подготовки и повышения квалификации кадров.

Для групповых и коллективных АРМ в подсистему организационного обеспечения включаются функции администри-рования АРМ: проектирование, планирование, учет, контроль, анализ, регулирование, организационные связи с инфрасистемами и др.

Организационное обеспечение предусматривает определе-ние и документальное оформление прав и обязанностей пользователей АРМ.

Программное обеспечение состоит из системного программного обеспечения и прикладного. Основой систем-ного обеспечения является операционная система и системы программирования, например, алгоритмический язык БЕЙСИК. Системные программы обеспечивают рациональную технологию обработки информации. Так называемые сервисные программы, которыми АРМ комплектуется в зависимости от потребности в них, расширяют возможности операционной системы. Для обеспечения информационной связи в сетях АРМ и связи АРМ по различным каналам также применяются программные средства, которые можно отнести к системному программированию.

Прикладное программное обеспечение составляют программы пользователей и пакеты прикладных программ разного назначения. Стандартные программы пользователей представляют собой программные решения определённых задач на алгоритмическом языке, чаще всего Бейсик.

ППП выполнены по модульному принципу и ориентированны на решение определенного класса задач. ППП являются основным видом проблемного программного обеспечения. Они позволяют формировать алгоритмы, изменять условия решения задач данного класса, контролировать ход решений, вносить коррективы в алгоритмы и др. При работе на АРМ ППП реализуются в диалоговом режиме.

Примерами ППП являются: ППП для формирования различных документов с выполнением расчётных операций, ППП для задач оптимизаций планов, ППП балансовых задач. Особое место уделяется ППП для создания автоматизированных информационных систем, которые могут иметь различное назначение: справочные, для обработки таблиц, ведения массивов информации, создания и ведения баз данных, документальные. Пакеты для работы с графической информацией позволяют представить в наглядном и компактном виде состояние и процессы, свойственные объектам, проиллюстрировать результаты прогнозного анализа.

## Теория современных систем управления базами даных

### Концепция баз данных

Активная деятельность по отысканию приемлемых способов обобществления непрерывно растущего объема информации привела к созданию в начале 60-х годов специальных программных комплексов, называемых "*Системы управления базами данных*" (СУБД). Этому предшествовал первый опыт использования файловых систем для организации баз данных. Файловые системы выявили различные проблемы обработки большого количества информации и заложили основные направления развития теории баз данных. Вот список лишь нескольких потребностей, которые не покрывались возможностями систем управления файлами:

* поддержание логически согласованного набора файлов
* обеспечение языка манипулирования данными
* восстановление информации после разного рода сбоев
* реально параллельная работа нескольких пользователей.

Можно считать, что если прикладная информационная система опирается на некоторую систему управления данными, обладающую этими свойствами, то эта система управления данными является *системой управления базами данных (СУБД).* Основная особенность СУБД – это наличие процедур для ввода и хранения не только самих данных, но и описаний их структуры. Файлы, снабженные описанием хранимых в них данных и находящиеся под управлением СУБД, стали называть банки данных, а затем "*Базы данных*" (БД). Приведем типовую схемы организации работы с СУБД.

Рис. 1.1 Связь программ и данных при использовании СУБД



### Архитектура СУБД

СУБД должна предоставлять доступ к данным любым пользователям, включая и тех, которые практически не имеют и (или) не хотят иметь представления о:

* физическом размещении в памяти данных и их описаний;
* механизмах поиска запрашиваемых данных;
* проблемах, возникающих при одновременном запросе одних и тех же данных многими пользователями (прикладными программами);
* способах обеспечения защиты данных от некорректных обновлений и (или) несанкционированного доступа;
* поддержании баз данных в актуальном состоянии и множестве других функций СУБД.

При выполнении основных из этих функций СУБД должна использовать различные описания данных. Отметим, что проектирование этих описании обычно поручается человеку (группе лиц) – *администратору базы данных* (АБД).

Объединяя частные представления о содержимом базы данных, полученные в результате опроса пользователей, и свои представления о данных, которые могут потребоваться в будущих приложениях, АБД сначала создает обобщенное неформальное описание создаваемой базы данных. Это описание, выполненное с использованием естественного языка, математических формул, таблиц, графиков и других средств, понятных всем людям, работающих над проектированием базы данных, называют *инфологической моделью данных*(рис. 1.2).

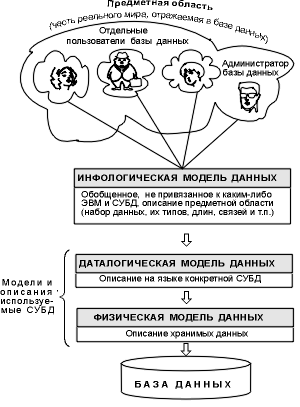


Рис. 1.2 Уровни моделей данных

Такая человеко-ориентированная модель полностью независима от физических параметров среды хранения данных. В конце концов этой средой может быть память человека, а не ЭВМ. Поэтому инфологическая модель не должна изменяться до тех пор, пока какие-то изменения в реальном мире не потребуют изменения в ней некоторого определения, чтобы эта модель продолжала отражать предметную область.

Остальные модели, показанные на рис. 1.2, являются компьютеро-ориентированными. С их помощью СУБД дает возможность программам и пользователям осуществлять доступ к хранимым данным лишь по их именам, не заботясь о физическом расположении этих данных. Нужные данные отыскиваются СУБД на внешних запоминающих устройствах по *физической модели данных*.

Так как указанный доступ осуществляется с помощью конкретной СУБД, то модели должны быть описаны на *языке описания данных* этой СУБД. Такое описание, создаваемое АБД по инфологической модели данных, называют *даталогической моделью данных*.

Трехуровневая архитектура (инфологический, даталогический и физический уровни) позволяет обеспечить *независимость хранимых данных* от использующих их программ. АБД может при необходимости переписать хранимые данные на другие носители информации и (или) реорганизовать их физическую структуру, изменив лишь физическую модель данных. АБД может подключить к системе любое число новых пользователей (новых приложений), дополнив, если надо, даталогическую модель. Указанные изменения физической и даталогической моделей не будут замечены существующими пользователями системы (окажутся "прозрачными" для них), так же как не будут замечены и новые пользователи. Следовательно, независимость данных обеспечивает возможность развития системы баз данных без разрушения существующих приложений.

### Инфологическая модель данных "Сущность-связь"

Цель инфологического моделирования – обеспечение наиболее естественных для человека способов сбора и представления той информации, которую предполагается хранить в создаваемой базе данных. Поэтому инфологическую модель данных пытаются строить по аналогии с естественным языком (последний не может быть использован в чистом виде из-за сложности компьютерной обработки текстов и неоднозначности любого естественного языка). Основными конструктивными элементами инфологических моделей являются сущности, связи между ними и их свойства (атрибуты).

*Сущность* – любой различимый объект (объект, который мы можем отличить от другого), информацию о котором необходимо хранить в базе данных. Сущностями могут быть люди, места, самолеты, рейсы, вкус, цвет и т.д. Необходимо различать такие понятия, как *тип сущности* и *экземпляр сущности*. Понятие тип сущности относится к набору однородных личностей, предметов, событий или идей, выступающих как целое. Экземпляр сущности относится к конкретной вещи в наборе. Например, типом сущности может быть ГОРОД, а экземпляром – Москва.

*Атрибут* – поименованная характеристика сущности. Его наименование должно быть уникальным для конкретного типа сущности, но может быть одинаковым для различного типа сущностей (например, ЦВЕТ может быть определен для многих сущностей: СОБАКА, АВТОМОБИЛЬ, ДЫМ и т.д.). Атрибуты используются для определения того, какая информация должна быть собрана о сущности.

Абсолютное различие между типами сущностей и атрибутами отсутствует. Атрибут является таковым только в связи с типом сущности. В другом контексте атрибут может выступать как самостоятельная сущность. Например, для автомобильного завода цвет – это только атрибут продукта производства, а для лакокрасочной фабрики цвет – тип сущности.

*Ключ* – минимальный набор атрибутов, по значениям которых можно однозначно найти требуемый экземпляр сущности. Минимальность означает, что исключение из набора любого атрибута не позволяет идентифицировать сущность по оставшимся.

*Связь* – ассоциирование двух или более сущностей. Если бы назначением базы данных было только хранение отдельных, не связанных между собой данных, то ее структура могла бы быть очень простой. Однако одно из основных требований к организации базы данных – это обеспечение возможности отыскания одних сущностей по значениям других, для чего необходимо установить между ними определенные связи. А так как в реальных базах данных нередко содержатся сотни или даже тысячи сущностей, то теоретически между ними может быть установлено более миллиона связей. Наличие такого множества связей и определяет сложность инфологических моделей.

### Реляционная структура данных

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

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

Наименьшая единица данных реляционной модели – это отдельное *атомарное* (неразложимое) для данной модели значение данных. Так, в одной предметной области фамилия, имя и отчество могут рассматриваться как единое значение, а в другой – как три различных значения.

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

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

*Тело* состоит из меняющегося во времени множества *кортежей*, где каждый кортеж состоит в свою очередь из множества пар атрибут-значение (Ai:Vi), (i=1,2,...,n), по одной такой паре для каждого атрибута Ai в заголовке. Для любой заданной пары атрибут-значение (Ai:Vi) Vi является значением из единственного домена Di, который связан с атрибутом Ai.

*Степень отношения* – это число его атрибутов. Отношение степени один называют унарным, степени два – бинарным, степени три – тернарным, ..., а степени n – n-арным.

*Кардинальное число* или *мощность отношения* – это число его кортежей. Кардинальное число отношения изменяется во времени в отличие от его степени.

Поскольку отношение – это множество, а множества по определению не содержат совпадающих элементов, то никакие два кортежа отношения не могут быть дубликатами друг друга в любой произвольно-заданный момент времени. Пусть R – отношение с атрибутами A1, A2, ..., An. Говорят, что множество атрибутов K=(Ai, Aj, ..., Ak) отношения R является возможным ключом R тогда и только тогда, когда удовлетворяются два независимых от времени условия:

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

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

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

Вышеупомянутые и некоторые другие математические понятия явились теоретической базой для создания реляционных СУБД, разработки соответствующих языковых средств и программных систем, обеспечивающих их высокую производительность, и создания основ теории проектирования баз данных. Однако для массового пользователя реляционных СУБД можно с успехом использовать неформальные эквиваленты этих понятий:

Отношение–Таблица(иногдаФайл),Кортеж – Строка (иногда Запись), Атрибут – Столбец, Поле. При этом принимается, что "запись" означает "экземпляр записи", а "поле" означает "имя и тип поля".

*Реляционная база данных* – это совокупность отношений, содержащих всю информацию, которая должна храниться в БД. Однако пользователи могут воспринимать такую базу данных как совокупность таблиц

### О нормализации, функциональных и многозначных зависимостях

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

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

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

Таблица находится в *первой нормальной форме* (*1НФ*) тогда и только тогда, когда ни одна из ее строк не содержит в любом своем поле более одного значения и ни одно из ее ключевых полей не пусто

Таблица находится во *второй нормальной форме* (*2НФ*), если она удовлетворяет определению 1НФ и все ее поля, не входящие в первичный ключ, связаны полной функциональной зависимостью с первичным ключом.

Таблица находится в *третьей нормальной форме* (*3НФ*), если она удовлетворяет определению 2НФ и не одно из ее неключевых полей не зависит функционально от любого другого неключевого поля.

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

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

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

*Полная функциональная зависимость*. Поле В находится в полной функциональной зависимости от составного поля А, если оно функционально зависит от А и не зависит функционально от любого подмножества поля А.

*Многозначная зависимость*. Поле А многозначно определяет поле В той же таблицы, если для каждого значения поля А существует хорошо определенное множество соответствующих значений В.

## Технология Клиент-Сервер.

### Основные понятия

Главная мысль, заложенная в эту технологию - минимизировать объем данных, передаваемых по сети, поскольку основные потери времени и сбои происходят именно из-за недостаточно высокой пропускной способности сети.

Очень многие СУБД разделяют свою работу на два уровня по системе "Клиент-Сервер". С точки зрения исполнения программа разделена на 2 части - клиентскую и серверную. На клиентской части (компьютере) происходит контакт с внешним миром. На компьютере-сервере расположены общие для всех клиентов данные и работает специальная программа - сервер баз данных, оптимизирующая выполнение запросов клиентов.

Итак, двухуровневая система "Клиент-Сервер" это:

*Клиент* - Программа обработки, она же пользовательская, она же прикладная программа. Занимается обычно интерфейсом с пользователем, а всю фактическую работу с базой данных возлагает на плечи БД-сервера.

*Сервер Базы Данных* - базис (database engine), он же ядро базы данных. Отдельная программа, выполняемая как отдельный процесс. Передает выбранную из базы информацию по межпроцессному каналу клиенту. Именно он, и только он фактически работает с данными, занимается их размещением на диске.

### Преимущества архитектуры клиент-сервер

В первый момент может возникнуть вопрос, а зачем такие сложности? Вот несколько соображений в пользу такого подхода.

Несомненным преимуществом является приближенность данных к процессам вычисления. Практически, все расчеты выполняются на сервере, что увеличивает быстродействие в десятки и сотни раз.

Технология клиент-сервер в отличие от файл-серверной дает пользователю большую безопасность, устойчивость, согласованность, масштабируемость, повышенную конфиденциальность и надежность обработки и хранения информации. Развитие систем с архитектурой клиент-сервер в немалой степени обязано проверенному факту: подключение к недорогим серверам недорогих ПК позволяет получить оптимальное соотношение цены и производительности.

В большинстве случаев программа обработки (клиентская часть) расположена на одном компьютере, а сама база данных хранится на другом. Тут разделение выглядит совершенно естественным: Программа - клиент (точнее, та ее часть, которая отвечает за интерфейс с пользователем), передает по сети запросы на обработку самих данных на другой компьютер, а там БД-сервер их прочитывает, выполняет требуемое, и по сети возвращает готовые ответы клиенту. При этом по сети передается только полезная информация.

Другое соображение: постоянно идет работа по совершенствованию самого метода хранения и обработки информации, и если его реализация (т.е. БД-сервер) сменилась, то не потребуется перекомпилировать с новыми библиотеками все разработанные программы, а достаточно будет инсталлировать новый БД-сервер взамен старого и перевести базы данных в формат нового сервера (применив для этого прилагаемую к нему утилиту). Естественно, все это можно проделать, если новый сервер придерживается тех же правил обмена между ним и пользовательской программой, что и старый, что, впрочем, наверняка имеет место.

Используя множество небольших компьютеров, разработчики систем клиент-сервер могут эмулировать вычислительную мощность больших ЭВМ, распределяя прикладную задачу по различным микрокомпьютерам и серверам. Каждый из них берет на себя свою часть вычислительной нагрузки, используя информацию совместно с другими процессорами сети. Суть идеи в том, чтобы повысить мощность системы, не наращивая производительность одного компьютера, а суммируя средства многих.

Быстродействие - основной фактор целесообразности разработки систем для архитектуры клиент-сервер . Применение средств быстрой разработки программ (Rapid Application Development - RAD), таких, как Delphi компании Borland, PowerBuilder фирмы PowerSoft и Visual Basic корпорации Microsoft, позволяет разработчикам "штамповать" прикладные системы для архитектуры клиент-сервер в рекордно короткие сроки. Технология серверов баз данных также становится проще в использовании и сочетается в одних системах со средствами RAD. Таким образом, с помощью этой скоростной и практически защищенной от "дурака" платформы разработки сокращается время, необходимое для подготовки и передачи прикладной программы пользователю.

В заключение стоит отметить что архитектура клиент-сервер предоставляет разработчикам ПО исключительную свободу выбора и согласования различных типов компонентов для клиента, сервера и всех промежуточных звеньев.

Наиболее простая (и рассматриваемая в контексте данной дипломной работы) форма архитектуры клиент-сервер - это разделение вычислительной нагрузки между двумя отдельными процессами: клиентом и сервером. Хотя и клиент, и сервер могут находиться на одном и том же компьютере, большинство систем этой архитектуры запускают клиентский процесс на одном компьютере, а процесс-сервер на другом, используя для обмена информацией сетевые связи. В этой модели один процесс может работать независимо от другого, выполнять определенные задания и разделять вычислительную нагрузку.

Обычно *клиентом* служит настольный ПК, выполняющий программное обеспечение конечного пользователя. *программное обеспечение* (ПО) *конечного пользователя* (front-end software) - это любая прикладная программа или пакет, способные направлять запросы по сети серверу и обрабатывать получаемую в ответ информацию. *Сервер*, в свою очередь, получает запросы и предпринимает действия от имени клиента.

ПК, работающий под управлением Windows 95 и выполняющий программу клиент-сервер Delphi, например, может представить на рассмотрение запрос серверу баз данных (скажем, программе InterBase 5.1.1, запущенной на сервере Windows NT). Обычно клиент посылает запросы базе данных в виде предложений на языке структурированных запросов (SQL), используя понятный серверу базы данных диалект.

### Компоненты архитектуры Клиент-сервер

Существуют три основных программных компонента архитектуры клиент-сервер :

ПО конечного пользователя.

промежуточное обеспечение.

ПО сервера.

К *ПО конечного пользователя* относятся средства разработки программ и генераторы отчетов, в том числе электронные таблицы и текстовые процессоры. С помощью этого ПО пользователи устанавливают связь с серверами, отправляют на рассмотрение серверу запросы и получают ответную информацию.

*Промежуточное обеспечение* (middleware) предоставляет общий интерфейс для ПО конечного пользователя и сервера, проникающий сквозь сквозь слои GUI (графический интерфейс пользователя), ОС, вычислительной сети и собственных драйверов базы данных с помощью общих вызовов. Для завершения операции сервер базы данных выполняет запрос и передает клиенту затребованные данные для обработки их программой клиента.

Под *ПО сервера* подразумевается операционная система и конкретный сервер БД, используемый для обработки запросов клиентской части информационной системы.

Серверы баз данных занимаются не только обслуживанием данных. В них предусмотрены также механизмы блокировок и элементы управления многопользовательским доступом, которые обеспечивают защиту данных от опасности параллельного доступа. Кроме этого, серверу баз данных приходится ограждать данные от несанкционированного доступа, оптимизировать запросы к базе данных, обеспечивать кэширование и предоставлять место для размещения словаря данных.

Две другие важные особенности, на которые стоит обратить внимание, - способность сервера обеспечивать целостность ссылочных данных и обоюдный контроль завершения транзакции. *Ссылочная целостность данных* (referential integrity) - это механизм, обеспечивающий каждому внешнему ключу соответствующий первичный ключ. *Обоюдный контроль завершения транзакций* (two-phase commit) - гарантия того, что ваши данные не будут повреждены даже при аппаратном сбое.

С помощью хранимых процедур, триггеров и правил разработчики могут составить программу непосредственно самого сервера баз данных и, таким образом, появляется еще одно место для размещения логики программы. *Хранимые процедуры* (stored procedures) - это группа предложений на языке SQL и процедурная логика, которые разработчики могут компилировать и хранить на сервере баз данных в качестве объектов. Программы-клиенты способны выполнять хранимые процедуры, также как и другой вид хранимых процедур или триггеров, путем посылки сообщений серверу баз данных.

Реальное распространение архитектуры "клиент-сервер" стало возможным благодаря развитию и широкому внедрению в практику концепции открытых систем. Поэтому мы начнем с краткого введения в открытые системы.

Основным смыслом подхода открытых систем является упрощение комплексирования вычислительных систем за счет международной и национальной стандартизации аппаратных и программных интерфейсов. Главной побудительной причиной развития концепции открытых систем явились повсеместный переход к использованию локальных компьютерных сетей и те проблемы комплексирования аппаратно-программных средств, которые вызвал этот переход. В связи с бурным развитием технологий глобальных коммуникаций открытые системы приобретают еще большее значение и масштабность.

Ключевой фразой открытых систем, направленной в сторону пользователей, является независимость от конкретного поставщика. Ориентируясь на продукцию компаний, придерживающихся стандартов открытых систем, потребитель, который приобретает любой продукт такой компании, не попадает к ней в рабство. Он может продолжить наращивание мощности своей системы путем приобретения продуктов любой другой компании, соблюдающей стандарты. Причем это касается как аппаратных, так и программных средств и не является необоснованной декларацией. Реальная возможность независимости от поставщика проверена в отечественных условиях.

Использование подхода открытых систем выгодно и производителям, и пользователям. Прежде всего открытые системы обеспечивают естественное решение проблемы поколений аппаратных и программных средств. Производители таких средств не вынуждаются решать все проблемы заново; они могут по крайней мере временно продолжать комплексировать системы, используя существующие компоненты.

## Язык структурированных запросов SQL

### Назначение и принцип работы SQL

SQL (часто произносится как "сиквэл", сокращенное название от *Structured Query Language*) символизирует собой Структурированный Язык Запросов. *SQL* является инструментом, предназначенным для обработки и чтения данных, содержащихся в компьютерной базе данных. Это язык который дает возможность эффективно создавать реляционные базы данных и работать с ними.

Мир баз данных становится все более и более единым, что привело к необходимости создания стандартного языка, который мог бы функционировать в большом количестве различных видов компьютерных сред. Стандартный язык позволит пользователям знающим один набор команд, использовать их чтобы создавать, отыскивать, изменять, и передавать информацию независимо от того работают ли они на персональном компьютере, сетевой рабочей станции, или на универсальной ЭВМ. Во все более и более взаимосвязанном компьютерном мире, пользователь, снабженый таким языком, имеет огромное преимущество в использовании и обобщении информации из ряда источников с помощью большого количества способов.

Как следует из названия, SQL является *языком программирования*, который применяется для организации взаимодействия пользователя с базой данных. На самом деле SQL работает только с базами данных реляционноготипа*.* На рис. 8 изображена схема работы SQL.

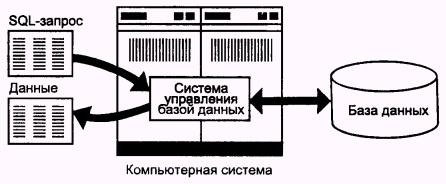


Рис. 1.3. Применение SQL для доступа к базе данных

Согласно этой схеме, в вычислительной системе имеется *база данных,* в которой хранится важная информация. Если вычислительная система относится к сфере бизнеса, то в базе данных может храниться информация о материальных ценностях, выпускаемой продукции, объемах продаж и зарплате. В базе данных на персональном компьютере может храниться информация о выписанных чеках, телефонах и адресах или информация, извлеченная из более крупной вычислительной системы.

Если пользователю необходимо прочитать данные из базы данных, он запрашивает их у СУБД с помощью SQL. СУБД обрабатывает запрос, находит требуемые данные и посылает их пользователю. Процесс запрашивания данных и получения результата называется *запросом* к базе данных: отсюда и название — *структурированный язык запросов.*

Вот основные моменты, характеризующие удобство применения SQL:

Организация данных. SQL дает пользователю возможность изменять структуру представления данных, а также устанавливать отношения между элементами базы данных.

Чтение данных. SQL дает пользователю или приложению возможность читать из базы данных содержащиеся в ней данные и пользоваться ими.

Обработка данных. SQL дает пользователю или приложению возможность изменять базу данных, т.е. добавлять в нее новые данные, а также удалять или обновлять уже имеющиеся в ней данные.

Управление доступом. С помощью SQL можно ограничить возможности пользователя по чтению и изменению данных и защитить их от несанкционированного доступа.

Совместное использование данных. SQL координирует совместное использование данных пользователями, работающими параллельно, чтобы они не мешали друг другу.

Целостность данных. SQL позволяет обеспечить целостность базы данных, защищая ее от разрушения из-за несогласованных изменений или отказа системы.

Таким образом, SQL является достаточно мощным языком для взаимодействия с СУБД.

### Достоинства языка SQL

Язык запросов дает несомненные преимущества. Во первых он продолжает идеологию архитектуры Клиент-Сервер. Клиентская часть приложения подготавливает запрос на обработку информации и отсылает запрос на сервер базы данных. Сервер , выполнив (обработав) полученный запрос возвращает клиентской программе готовый результат.

Основные преимущества напрямую вытекают из преимуществ клиент-серверного подхода. Например, простое суммирование значений всех полей без использования SQL приведет к пересылки всей таблицы по сети на машину клиента. После суммирования таблица фактически уже не нужна и такое использование сети как минимум не рационально. В случае же с SQL по сети уйдет запрос на сервер, сервер проведет суммирование и вернет обратно по сети только полученную сумму!

Элегантность и независимость от специфики компьютерных технологий (аппаратных платформ), а также его поддержка лидерами промышленности в области технологии реляционных баз данных, сделало SQL, и вероятно в течение обозримого будущего оставит его, основным стандартным языком. По этой причине, любой кто хочет работать с базами данных 90-х годов должен знать SQL.

Стандарт SQL определяется ANSI (Американским Национальным Институтом Стандартов) и в данное время также принимается ISO (МЕЖДУНАРОДНОЙ ОРГАНИЗАЦИЕЙ ПО СТАНДАРТИЗАЦИИ). Однако, большинство коммерческих СУБД расширяют SQL без уведомления ANSI, добавляя разные другие особенности в этот язык, которые, как они считают, будут весьма полезны. Иногда они несколько нарушают стандарт языка, хотя хорошие идеи имеют тенденцию развиваться и вскоре становиться стандартами "рынка" сами по себе в силу полезности своих качеств.

Здесь следует отметить, что несмотря на достаточно большой набор нестандартных дополнительных фукций *SQL InterBase*, программа Отдел Кадров использует только стандартные операторы и конструкции. Такое решение принято для возможности легкого переноса программы на другой SQL сервер. Например, при дальнейшем развитии можно перенести программу на Microsoft SQL Server, если InterBase по каким либо причинам перестанет удовлетворять запросы разработчиков. Также следует отметить, что многие нестандартные , дополнительные возможности разных SQL серверов зачастую похожи между собой и при выходе нового стандарта, как правило переносятся в группу стандартизованных. Таким образом происходит дальнейшее развитие языка SQL.

В целом, список преимуществ, на которые стоит обратить внимание в первую очередь, можно представить в следующем виде:

* независимость от конкретных СУБД;
* переносимость с одной вычислительной системы на другую;
* наличие стандартов;
* поддержка со стороны компании Microsoft (протокол ODBC);
* реляционная основа;
* высокоуровневая структура, напоминающая английский язык;
* возможность выполнения специальных интерактивных запросов:
* обеспечение программного доступа к базам данных;
* возможность различного представления данных;
* полноценность как языка, предназначенного для работы с базами данных;
* возможность динамического определения данных;
* поддержка архитектуры клиент/сервер.

Все перечисленные выше факторы явились причиной того, что SQL стал стандартным инструментом для управления данными на персональных компьютерах, мини-компьютерах и больших ЭВМ. Ниже эти факторы рассмотрены более подробно.

*Независимость от конкретных СУБД*

Все ведущие поставщики СУБД используют SQL, и ни одна новая СУБД, не поддерживающая SQL, не может рассчитывать на успех. Реляционную базу данных и программы, которые с ней работают, можно перенести с одной СУБД на другую с минимальными доработками и переподготовкой персонала. Программные средства, входящие в состав СУБД для персональных компьютеров, работают с реляционными базами данных многих типов.

*Переносимость с одной вычислительной системы на другие*

Поставщики СУБД предлагают программные продукты для различных вычислительных систем: от персональных компьютеров и рабочих станций до локальных сетей, мини-компьютеров и больших ЭВМ. Приложения, созданные с помощью SQL и рассчитанные на однопользовательские системы, по мере своего развития могут быть перенесены в более крупные системы. Информация из корпоративных реляционных баз данных может быть загружена в базы данных отдельных подразделений или в личные базы данных. Наконец, приложения для реляционных баз данных можно вначале смоделировать на экономичных персональных компьютерах, а затем перенести на дорогие многопользовательские системы.

*Стандарты языка SQL*

Официальный стандарт языка SQL был опубликован Американским институтом национальных стандартов (American National Standards Institute — ANSI) и Международной организацией по стандартам (International Standards Organization — ISO) в 1986 году и значительно расширен в 1992 году. Кроме того, SQL является федеральным стандартом США по обработке информации (FIPS — Federal Information Processing Standard) и, следовательно, соответствие ему является одним из основных требований, содержащихся в больших правительственных контрактах, относящихся к области вычислительной техники. В Европе стандарт X/OPEN для переносимой среды программирования на основе операционной системы UNIX включает в себя SQL в качестве стандарта для доступа к базам данных. SQL Access Group — консорциум поставщиков компьютерного оборудования и баз данных — определил для SQL стандартный интерфейс вызовов функций, который является основой протокола ODBC компании Microsoft и входит также в стандарт X/OPEN. Эти стандарты служат как бы официальной печатью, одобряющей SQL, и они ускорили завоевание им рынка.

*Протокол ODBC и компания Microsoft*

Компания Microsoft рассматривает доступ к базам данных как важную часть своей операционной системы Windows. Стандартом этой компании по обеспечению доступа к базам данных является ODBC (Open Database Connectivity — взаимодействие с открытыми базами данных) — программный интерфейс, основанный на SQL. Протокол ODBC поддерживается наиболее распространенными приложениями Windows (электронными таблицами, текстовыми процессорами, базами данных и т.п.), разработанными как самой компанией Microsoft, так и другими ведущими поставщиками. Поддержка ODBC обеспечивается всеми ведущими реляционными базами данных. Кроме того, ODBC опирается на стандарты, одобренные консорциумом поставщиков SQL Access Group, что делает ODBC как стандартом де-факто компании Microsoft, так и стандартом, независимым от конкретных СУБД.[13, 8, 17].

### Запрос на языке SQL

Как подчеркивалось ранее, SQL символизирует собой Структурированный Язык Запросов. Запросы - вероятно наиболее часто используемый аспект SQL. Фактически, для категории SQL пользователей, маловероятно чтобы кто-либо использовал этот язык для чего-то другого. Итак :

Запрос - команда которую дается прикладной программе базы данных, и которая сообщает ей чтобы она вывела определенную информацию из таблиц в память. Эта информация обычно посылается непосредственно на экран компьютера или терминала хотя, в большинстве случаев, ее можно также послать принтеру, сохранить в файле ( как объект в памяти компьютера ), или представить как вводную информацию для другой команды или процесса.

Одна из наиболее важных особенностей запросов SQL - это их способность определять связи между многочисленными таблицами и выводить информацию из них в терминах этих связей, всю внутри одной команды. Этот вид операции называется - объединением, которое является одним из видов операций в реляционных базах данных.

Как установлено ранее, главное в реляционном подходе это связи которые можно создавать между позициями данных в таблицах. Используя обьединения, мы непосредственно связываем информацию с любым номером таблицы, и таким образом способны создавать связи между сравнимыми фрагментами данных. При обьединении, таблицы представленые списком в предложении FROM запроса, отделяются запятыми. Предикат запроса может ссылаться к любому столбцу любой связанной таблицы и, следовательно, может использоваться для связи между ими.

Язык допускает три типа синтаксических конструкций, начинающихся с ключевого слова SELECT: спецификация курсора (cursor specification), оператор выборки (select statement) и подзапрос (subquery). Основой всех них является синтаксическая конструкция "табличное выражение (table expression)". Семантика табличного выражения состоит в том, что на основе последовательного применения разделов from, where, group by и having из заданных в разделе from таблиц строится некоторая новая результирующая таблица, порядок следования строк которой не определен и среди строк которой могут находиться дубликаты (т.е. в общем случае таблица-результат табличного выражения является мультимножеством строк). На самом деле именно структура табличного выражения наибольшим образом характеризует структуру запросов языка SQL/89.

### Агрегатные функции

Запросы могут производить обобщенное групповое значение полей точно также как и значение одного поля. Это делает с помощью агрегатых функций. Агрегатные функции производят одиночное значение для всей группы таблицы. Имеется список этих функций:

* COUNT производит номера строк или не-NULL значения полей которые выбрал запрос.
* SUM производит арифметическую сумму всех выбранных значений данного поля.
* AVG производит усреднение всех выбранных значений данного поля.
* MAX производит наибольшее из всех выбранных значений данного поля.
* MIN производит наименьшее из всех выбранных значений данного поля.

## Локальные вычислительные сети

На сегодняшний день в мире существует более 150 миллионов компьютеров, бо­лее 80 % из них объединены в различные информационно-вычислительные сети от малых локальных сетей в офисах до глобальных сетей типа Internet

Автоматизированное рабочее место «Отдел Кадров» является программой, активно использующей сетевое соединение отдельных компьютеров в локальную вычислительную сеть. Только при этом становится возможной передача информации с любого рабочего места пользователя на сервер и обратно. Скорость передачи данных по сети естественным образом влияет на общую скорость работы всего АРМ. В свою очередь, скорость прохождения информации от сервера к локальному компьютеру пользователя определяется комплексом программно-аппаратных средств, которые и составляют локальную вычислительную сеть.

В настоящее время существуют различные способы связи разрозненных компьютеров в единое целое (т.е. в сеть). Спектр аппаратных средств (и программных для управления ими) более чем широк. Иногда это приводит к некоторому затруднению при выборе типа сети и её программного обеспечения. Неправильный выбор может в дальнейшем привести к невозможности функционирования программ в случае увеличения парка машин или возрастания требований к скорости и объемам передаваемой информации. С учетом сказанного становится ясно, что необходимо в достаточной степени понимать принципы организации ЛВС, грамотно выбрать аппаратные и программные средства для её построения.

В данном разделе описаны основные, базовые принципы ЛВС, приведены различные схемы соединения машин. Дано описание достоинств и недостатков каждой схемы. Теоретическая информация подкреплена описанием реально используемыми в настоящее время аппаратными и программными средствами для построения ЛВС. Наряду с давно применяемыми и хорошо изученными способами построения сетей приводится описание современного способа соединения с помощью оптоволоконного кабеля.

### Файл сервер и рабочие станции

ЛВС могут состоять из одного файл-сервера, поддерживающего небольшое число рабочих станций, или из многих файл-серверов и коммуникационных серверов, соединенных с сотнями рабочих станций. Некоторые сети спроектированы для оказания сравнительно простых услуг, таких, как совместное пользование прикладной программой и файлом и обеспечение доступа к единственному принтеру. Другие сети обеспечивают связь с большими и мини-ЭВМ, модемами коллективного пользования, разнообразными устройствами ввода/вывода (графопостроителями, принтерами и т. д.) и устройствам памяти большой емкости (диски типа WORM).

Файл-сервер является ядром локальной сети. Этот компьютер (обычно высокопроизводительный мини-компьютер) запускает операционную систему и управляет потоком данных, передаваемых по сети. Отдельные рабочие станции и любые совместно используемые периферийные устройства, такие, как принтеры, - все подсоединяются к файл-серверу.

Каждая рабочая станция представляет собой обычный персональный компьютер, работающий под управлением собственной дисковой операционной системы (такой, как DOS или OS/2). Однако в отличие от автономного персонального компьютера рабочая станция содержит плату сетевого интерфейса и физически соединена кабелями с файлом-сервером. Кроме того, рабочая станция запускает специальную программу, называемой оболочкой сети, которая позволяет ей обмениваться информацией с файл-сервером, другими рабочими станциями и прочими устройствами сети. Оболочка позволяет рабочей станции использовать файлы и программы, хранящиеся на файл-сервере, так же легко, как и находящиеся на ее собственных дисках.

### Операционная система рабочей станции

Каждый компьютер рабочей станции работает под управлением своей собственной операционной системы (такой, как DOS или OS/2). Чтобы включить каждую рабочую станцию с состав сети, оболочка сетевой операционной системы загружается в начало операционной системы компьютера.

Оболочка сохраняет большую часть команд и функций операционной системы, позволяя рабочей станции в процессе работы выглядеть как обычно. Оболочка просто добавляет локальной операционной системе больше функций и придает ей гибкость.

### Преимущества локальных вычислительных сетей

Понятие локальная вычислительная сеть - ЛВС (англ. LAN - Lokal Area Network) относится к географически ограниченным ( территориально или производственно) аппаратно-программным реализациям, в которых несколько компьютерных систем связаны друг с другом с помощью соответствующих средств коммуникаций. Благодаря такому со­единению пользователь может взаимодействовать с другими рабочими станциями, подключенными к этой ЛВС.

В производственной практики ЛВС играют очень большую роль. Посредством ЛВС в систему объединяются персональные компьютеры, расположенные на многих удален­ных рабочих местах, которые используют совместно оборудование, программные средства и информацию. Рабочие места сотрудников перестают быть изолированными и объеди­няются в единую систему. Рассмотрим преимущества, получаемые при сетевом объединении персональных компьютеров в виде внутрипроизводственной вычислительной сети.

*Разделение ресурсов.*

Разделение ресурсов позволяет экономно использовать ресурсы, например, управлять периферийными устройствами, такими как лазерные печатающие устройства, со всех присоединенных рабочих станций.

*Разделение данных.*

Разделение данных предоставляет возможность доступа и управле­ния базами данных с периферийных рабочих мест, нуждающихся в инфор­мации.

*Разделение программных средств.*

Разделение программных средств предоставляет возможность одновременного использования централизованных, ранее установленных

программных средств.

*Разделение ресурсов процессора.*

При разделение ресурсов процессора возможно использование вычислительных мощностей для обработки данных другими системами, входящими в сеть. Предоставляе­мая возможность заключается в том, что на имеющиеся ресурсы пользовательские программы не “набрасываются” моментально, а только лишь че­рез специальный процессор, доступный каждой рабочей станции.

*Многопользовательский режим.*

Многопользовательские свойства системы содействуют одновременному использованию централизованных прикладных программных средств, ранее установленных и управляемых. Например, если пользователь системы работает с другим заданием, то те­кущая выполняемая работа отодвигается на задний план.

### Стандарт передачи информации

Все ЛВС работают в одном стандарте принятом для компьютерных сетей - в стандарте OSI (англ. Open Systems Interconnection). В данном разделе описана базовая модель OSI.

Для того чтобы взаимодействовать, люди используют общий язык. Если они не могут разговаривать друг с другом непосредственно, они применяют соответствующие вспомогательные средства для передачи сообще­ний.

Показанные выше стадии общения необходимы, когда сообщение передается от отправителя к получателю.

Для того чтобы привести в движение процесс передачи данных, используются машины с одинаковым кодированием данных и связанные одна с другой. Для единого представления данных в линиях связи, по которым передается информация, сформи­рована Международная организация по стандартизации ISO (англ. ISO - International Standards Organization).

ISO предназначена для разработки модели международного коммуникационного протокола, в рамках которой можно разрабатывать международные стандарты. Для наглядного по­яснения расчленим ее на семь уровней.

ISO разработала указанную базовую модель взаимодействия открытых систем OSI. Модель содержит семь отдельных уровней:

1. физический - битовые протоколы передачи информации;
2. канальный - формирование кадров, управление доступом к среде;
3. сетевой - маршрутизация, управление потоками данных;
4. транспортный - обеспечение взаимодействия удаленных процес­сов;
5. сеансовый - поддержка диалога между удаленными про­цессами;
6. представлении данных - интерпретация передаваемых данных;
7. прикладной - пользовательское управление данными.

Основная идея этой модели заключается в том, что каждому уровню отводится кон­кретная ролью в том числе и транспортной среде. Благодаря этому общая задача передачи дан­ных расчленяется на отдельные легко обозримые задачи. Необходимые соглашения для связи одного уровня с выше- и нижерасположенными называют про­токолом.

Так как пользователи нуждаются в эффективном управлении, система вычис­лительной сети представляется как комплексное строение, которое координирует взаимодействие задач пользователей.

С учетом вышеизложенного можно вывести следующую уровневую модель с админи­стративными функциями, выполняющимися в пользова­тельском прикладном уровне.

Отдельные уровни базовой модели проходят в направлении вниз от источника данных (от уровня 7 к уровню 1) и в направлении вверх от прием­ника данных (от уровня 1 к уровню 7). Пользовательские данные переда­ются в нижерасположенный уровень вместе со специфическим для уровня заголовком до тех пор, пока не будет достигнут последний уровень.

На приемной стороне поступающие данные анализируются и, по мере надоб­ности, передаются далее в вышерасположенный уровень, пока ин­формация не будет передана в пользо­вательский прикладной уровень.

*Уровень 1. Физический.*

На физическом уровне определяются электрические, механические, функ­циональные и процедурные параметры для физической связи в системах. Физическая связь и неразрывная с ней экс­плуатационная готовность явля­ются основной функцией 1-го уровня. Стандарты физического уровня вклю­чают рекомендации V.24 МККТТ (CCITT), EIA RS232 и Х.21. Стандарт ISDN ( Integrated Services Digital Network) в будущем сыграет определяющую роль для функций передачи данных. В качестве среды передачи данных исполь­зуют трехжильный медный провод (экранированная витая пара), коакси­аль­ный кабель, оптоволоконный проводник и радиорелейную линию.

*Уровень 2. Канальный.*

Канальный уровень формирует из данных, передаваемых 1-м уров­нем, так на­зываемые "кадры" последовательности кадров. На этом уровне осуществляются управление доступом к передающей среде, используемой несколькими ЭВМ, синхро­низация, обнаружение и исправле­ние ошибок.

*Уровень 3. Сетевой.*

Сетевой уровень устанавливает связь в вычислительной сети между двумя абонентами. Соединение происходит благодаря функциям маршрутизации, которые требуют наличия сете­вого адреса в пакете. Сетевой уровень должен также обеспечи­вать обработку ошибок, мультиплексирование, управление потоками данных. Самый известный стандарт, относящийся к этому уровню, - рекомендация Х.25 МККТТ (для сетей общего пользования с коммутацией пакетов).

*Уровень 4. Транспортный.*

Транспортный уровень поддерживает непрерывную передачу данных между двумя взаимодействующими друг с другом пользовательскими процессами. Качество транспорти­ровки, безошибочность передачи, независимость вычислительных сетей, сервис транспорти­ровки из конца в конец, ми­нимизация затрат и адресация связи га­рантируют непрерывную и безоши­бочную передачу данных.

*Уровень 5. Сеансовый.*

Сеансовый уровень координирует прием, передачу и выдачу одного сеанса связи. Для координации необходимы контроль рабочих параметров, управление потоками данных промежуточных накопителей и диалоговый контроль, гарантирующий передачу, имеющихся в распоряжении данных. Кроме того, сеансовый уровень содержит дополнительно функции управления паролями, подсчета платы за пользование ресурсами сети, управления диалогом, синхрони­зации и отмены связи в сеансе передачи после сбоя вследствие ошибок в нижерасположенных уровнях.

*Уровень 6. Представления данных.*

Уровень представления данных предназначен для интерпретации данных; а также под­готовки данных для пользовательского прикладного уровня. На этом уровне происходит преоб­разование данных из кадров, используемых для передачи данных в экранный формат или фор­мат для печатающих устройств оконечной системы.

*Уровень 7. Прикладной.*

В прикладном уровне необходимо предоставить в распоряжение пользовате­лей уже пе­реработанную информацию. С этим может спра­виться системное и пользовательское приклад­ное программное обеспече­ние.

Для передачи информации по коммуникационным линиям данные преобразу­ются в це­почку следующих друг за другом битов (двоичное кодирование с помощью двух состояний:"0" и "1").

Передаваемые алфавитно-цифровые знаки представляются с помо­щью бито­вых комби­наций. Битовые комбинации располагают в определен­ной кодовой таблице, содержащей 4-, 5-, 6-, 7- или 8-битовые коды.

Количество представленных знаков в ходе зависит от количества би­тов, ис­пользуемых в коде: код из четырех битов может представить макси­мум 16 значений, 5-битовый код - 32 зна­чения, 6-битовый код - 64 значения, 7-битовый - 128 значений и 8-битовый код - 256 алфавитно-цифровых зна­ков.

При передаче информации между одинаковыми вычислительными системами и разли­чающимися типами компьютеров применяют следующие коды:

На международном уровне передача символьной информации осуществляется с помо­щью 7-битового кодирования, позволяющего закодировать заглавные и строч­ные буквы англий­ского алфавита, а также некоторые спец­символы.

Национальные и специальные знаки с помощью 7-битово кода пред­ставить нельзя. Для представления национальных знаков применяют наиболее употребимый 8-битовый код.

Для правильной и, следовательно, полной и безошибочной передачи данных необхо­димо придерживаться согласованных и установленных пра­вил. Все они огово­рены в протоколе передачи данных.

Протокол передачи данных требует следующей информации:

* *Синхронизация.*Под синхронизацией понимают механизм распознавания начала блока данных и его конца.
* *Инициализация*.Под инициализацией понимают установление соединения между взаимодейст­вующими партнерами.
* *Блокирование.*Под блокированием понимают разбиение передаваемой информации на блоки данных строго определенной максимальной длины (включая опо­знава­тельные знаки начала блока и его конца).
* *Адресация*. Адресация обеспечивает идентификацию различного используемого оборудо­вания дан­ных, которое обменивается друг с другом информацией во время взаимодей­ствия.
* *Обнаружение ошибок.* Под обнаружением ошибок понимают установку битов четности и, следова­тельно, вы­числение контрольных битов.
* *Нумерация блоков*. Текущая нумерация блоков позволяет установить ошибочно переда­ваемую или поте­рявшуюся информацию.
* *Управление потоком данных.* Управление потоком данных служит для распределения и синхрони­зации ин­формаци­онных потоков. Так, например, если не хватает места в бу­фере устройства данных или данные не достаточно быстро обрабатыва­ются в периферийных устройст­вах (например, принтерах), со­общения и / или за­просы накапливаются.
* *Методы восстановления*. После прерывания процесса передачи данных используют методы восстанов­ления, чтобы вернуться к определенному положению для повтор­ной передачи инфор­мации.
* *Разрешение доступа.*Распределение, контроль и управление ограничениями доступа к данным вме­няются в обязанность пункта разрешения доступа (например, "только передача" или "только прием" ).

### Сетевые устройства и средства коммуника­ций.

В качестве средств коммуникации наиболее часто используются витая пара, коаксиальный кабель и оптоволоконные линии. При выборе типа кабеля учитывают сле­дующие показатели:

• стоимость монтажа и обслуживания,

• скорость передачи информации,

• ограничения на величину расстояния передачи информации (без дополни­тельных усилителей-повторителей(репитеров)),

• безопасность передачи данных.

Главная проблема заключается в одновременном обеспечении этих показате­лей, например, наивысшая скорость передачи данных ограничена максимально воз­можным расстоянием передачи данных, при котором еще обеспечивается требуемый уровень защиты данных. Легкая наращивае­мость и простота расширения кабельной системы влияют на ее стоимость.

### Витая пара.

Наиболее дешевым кабельным соединением является витое двух­жильное про­водное соединение часто называемое "витой парой" (twisted pair). Она позволяет пе­редавать информацию со скоростью до 10 Мбит/с, легко наращивается, однако явля­ется помехонезащищенной. Длина кабеля не может превышать 1000 м при скорости передачи 1 Мбит/с. Преимущест­вами являются низкая цена и бес проблемная уста­новка. Для повышения помехозащищенности информации часто используют экраниро­ванную ви­тую пару, т.е. витую пару, помещенную в экранирующую оболочку, подобно экрану коаксиального кабеля. Это увеличивает стоимость витой пары и при­ближает ее цену к цене коаксиального кабеля.

### Широкополосный коаксиальный кабель.

Широкополосный коаксиальный кабель невосприимчив к помехам, легко на­ращива­ется, но цена его высокая. Скорость передачи информации равна 500 Мбит/с. При пе­редачи информации в базисной полосе частот на рас­стояние более 1,5 км требуется усилитель, или так называемый репитер (повторитель). Поэтому суммарное расстоя­ние при передаче информации увеличивается до 10 км. Для вычислительных сетей с топологией шина или дерево коаксиальный кабель должен иметь на конце согласую­щий резистор (терминатор).

### Еthernet-кабель.

Ethernet-кабель также является коаксиальным кабелем с волновым сопротив­лением 50 Ом. Его называют еще толстый Ethernet (thick) или жел­тый кабель (yellow ca­ble). Он использует 15-контактное стандартное включе­ние. Вследствие помехоза­щищенности является дорогой альтернативой обычным коаксиальным кабелям. Мак­симально доступное расстояние без повторителя не превышает 500 м, а общее рас­стояние сети Ethernet - около 3000 м. Ethernet-кабель, благодаря своей магистральной топологии, ис­пользует в конце лишь один нагрузочный резистор.

### Оптоволоконные линии.

Наиболее дорогими являются оптопроводники, называемые также стекловоло­конным кабелем. Скорость распространения информации по ним достигает нескольких гигабит в секунду. Допустимое удаление более 50 км. Внешнее воздействие помех практически отсутствует. На данный момент это наиболее дорогостоящее соединение для ЛВС. Применяются там, где возникают электромагнитные поля помех или требу­ется передача информа­ции на очень большие расстояния без использования повтори­телей. Они обладают противоподслушивающими свойствами, так как техника ответв­ле­ний в оптоволоконных кабелях очень сложна. Оптопроводники объединя­ются в JIBC с помощью звездообразного соединения.

Показатели трех типовых сред для передачи приведены в таблице.

|  |  |  |  |
| --- | --- | --- | --- |
| **Показатели** | **Среда передачи данных** | | |
|  | **Двух жильный кабель - витая пара** | **Коаксиальный ка­бель** | **Оптоволо­кон­ный кабель** |
| **Цена** | Невысокая | Относительно высо­кая | Высокая |
| **Наращивание** | Очень простое | Проблематично | Простое |
| **Защита от про­слушивания** | Незначительная | Хорошая | Высокая |
| **Показатели** | **Среда передачи данных** | | |
|  | **Двух жильный кабель - витая пара** | **Коаксиальный ка­бель** | **Оптоволо­кон­ный кабель** |
| **Проблемы с заземлением** | Нет | Возможны | Нет |
| **Восприимчи­вость к поме­хам** | Существует | Существует | Отсутствует |

### Топологии вычислительной сети.

Термин "топология сети" относится к пути, по которому данные перемещаются по сети. Существуют три основных вида топологий: "общая шина", "звезда" и "кольцо".

### Топология типа звезда.

Концепция топологии сети в виде звезды пришла из области больших ЭВМ, в которой головная машина получает и обрабатывает все данные с пе­риферийных устройств как активный узел обработки данных. Вся информация между двумя периферийными рабочими мес­тами проходит через центральный узел вычислительной сети.



Топология в виде звезды

Пропускная способность сети определяется вычислительной мощностью узла и гарантируется для каждой рабочей станции. Коллизий (столкновений) данных не возникает.

Кабельное соединение довольно простое, так как каждая рабочая станция связана с узлом. Затраты на прокладку кабелей высокие, особенно когда центральный узел географически расположен не в центре топологии.

При расширении вычислительных сетей не могут быть использованы ранее выполненные кабельные связи: к новому рабочему месту необходимо прокладывать отдельный кабель из центра сети.

Топология в виде звезды является наиболее быстродействующей из всех топологий вычислительных сетей, поскольку передача данных между рабочими станциями проходит через центральный узел (при его хорошей производительности) по отдельным линиям, используемым только этими рабочими станциями. Частота запросов передачи информации от одной станции к другой невысокая по сравнению с достигаемой в других топологиях.

Производительность вычислительной сети в первую очередь зависит от мощности центрального файлового сервера. Он может быть узким ме­стом вычислительной сети. В случае выхода из строя центрального узла нарушается работа всей сети.Центральный узел управления - файловый сервер мотает реализо­вать оптимальный механизм защиты против несанкционированного доступа к информации. Вся вычислительная сеть может управляться из ее центра.

### Кольцевая топология.

При кольцевой топологии сети рабочие станции связаны одна с дру­гой по кругу, т.е. рабочая станция 1 с рабочей станцией 2, рабочая станция 3 с рабочей станцией 4 и т.д. Последняя рабочая станция связана с первой. Коммуникационная связь замыкается в кольцо.



Кольцевая топология

Прокладка кабелей от одной рабочей станции до другой может быть довольно сложной и дорогостоящей, особенно если географически рабочиестанции расположены далеко от кольца (например, в линию).

Сообщения циркулируют регулярно по кругу. Рабочая станция посы­лает по определенному конечному адресу информацию, предварительно получив из кольца запрос. Пересылка сообщений является очень эффектив­ной, так как большинство сообщений можно отправлять “в дорогу” по ка­бельной системе одно за другим. Очень просто можно сделать кольцевой запрос на все станции. Продолжительность передачи информации увеличи­вается пропорционально количеству рабочих станций, входящих в вычисли­тельную сеть.

Основная проблема при кольцевой топологии заключается в том, что каждая рабочая станция должна активно участвовать в пересылке информа­ции, и в случае выхода из строя хотя бы одной из них вся сеть парализуется. Неисправности в кабельных соединениях локализуются легко.

Подключение новой рабочей станции требует краткосрочного выключения сети, так как во время установки кольцо должно быть разомкнуто. Ограниче­ния на протяженность вычислительной сети не существует, так как оно, в конечном счете, определяется исключительно расстоянием между двумя рабочими станциями.

Специальной формой кольцевой топологии является логическая кольцевая сеть. Физически она монтируется как соединение звездных топо­логий. Отдельные звезды включаются с помощью специальных коммутато­ров (англ. Hub -концентратор), которые по-русски также иногда называют “хаб”. В зависимости от числа рабочих станций и длины кабеля между рабо­чими станциями применяют активные или пассивные концентраторы. Актив­ные концентраторы дополнительно содержат усилитель для подключения от 4 до 16 рабочих станций. Пассивный концентратор является исключи­тельно разветвительным устройством (максимум на три рабочие станции). Управление отдельной рабочей станцией в логической кольцевой сети про­исходит так же, как и в обычной кольцевой сети. Каждой рабочей станции присваивается соответствующий ей адрес, по которому передается управ­ление (от старшего к младшему и от самого младшего к самому старшему). Разрыв соединения происходит только для нижерасположенного (ближайшего) узла вычислительной сети, так что лишь в редких случаях мо­жет нарушаться работа всей сети.

### Шинная топология.

При шинной топологии среда передачи информации представляется в форме коммуникационного пути, доступного дня всех рабочих станций, к которому они все должны быть подключены. Все рабочие станции могут не­посредственно вступать в контакт с любой рабочей станцией, имеющейся в сети.



Шинная топология

Рабочие станции в любое время, без прерывания работы всей вычис­лительной сети, могут быть подключены к ней или отключены. Функциони­рование вычислительной сети не зависит от состояния отдельной рабочей станции.

В стандартной ситуации для шинной сети Ethernet часто используют тонкий кабель или Cheapernet-кaбeль с тройниковым соединителем. Выклю­чение и особенно подключение к такой сети требуют разрыва шины, что вы­зывает нарушение циркулирующего потока информации и зависание сис­темы.

Новые технологии предлагают пассивные штепсельные коробки, че­рез которые можно отключать и / или включать рабочие станции во время работы вычислительной сети.

Благодаря тому, что рабочие станции можно включать без прерыва­ния сетевых процессов и коммуникационной среды, очень легко прослуши­вать информацию, т.е. ответвлять информацию из коммуникационной среды.

В ЛВС с прямой (не модулируемой) передачей информации всегда может существовать только одна станция, передающая информацию. Для предот­вращения коллизий в большинстве случаев применяется временной метод разделения, согласно которому для каждой подключенной рабочей станции в определенные моменты времени предоставляется исключительное право на использование канала передачи данных. Поэтому требования к пропуск­ной способности вычислительной сети при повышенной нагрузке снижа­ются, например, при вводе новых рабочих станций. Рабочие станции при­соединяются к шине посредством устройств ТАР (англ. Terminal Access Point - точка подключения терминала). ТАР представляет собой специальный тип подсоединения к коаксиальному кабелю. Зонд игольчатой формы внедря­ется через наружную оболочку внешнего проводника и слой диэлектрика к внутреннему проводнику и присоединяется к нему.

В ЛВС с модулированной широкополосной передачей информации различные рабочие станции получают, по мере надобности, частоту, на ко­торой эти рабочие станции могут отправлять и получать информацию. Пе­ресылаемые данные модулируются на соответствующих несущих частотах, т.е. между средой передачи информации и рабочими станциями находятся соответственно модемы для модуляции и демодуляции. Техника широкопо­лосных сообщений позволяет одновременно транспортировать в коммуни­кационной среде довольно большой объем информации. Для дальнейшего развития дискретной транспортировки данных не играет роли, какая перво­начальная информация подана в модем (аналоговая или цифровая), так как она все равно в дальнейшем будет преобразована.

Характеристики топологий вычислительных сетей приведены в таб­лице.

|  |  |  |  |
| --- | --- | --- | --- |
| **Характери­стики** | **Топология** | | |
|  | **Звезда** | **Кольцо** | **Шина** |
| **Стоимость расширения** | Незначительная | Средняя | Средняя |
| **Присоединение абонентов** | Пассивное | Активное | Пассивное |
| **Защита от от­казов** | Незначительная | Незначительная | Высокая |
| **Характери­стики** | **Топология** | | |
|  | **Звезда** | **Кольцо** | **Шина** |
| **Размеры сис­темы** | Любые | Любые | Ограниченны |
| **Защищенность от прослуши­вания** | Хорошая | Хорошая | Незначительная |
| **Стоимость подключения** | Незначительная | Незначительная | Высокая |
| **Поведение системы при высоких на­грузках** | Хорошее | Удовлетворитель­ное | Плохое |
| **Возможность работы в ре­альном режиме времени** | Очень хорошая | Хорошая | Плохая |
| **Разводка ка­беля** | Хорошая | Удовлетворитель­ная | Хорошая |
| **Обслуживание** | Очень хорошее | Среднее | Среднее |

### Методы доступа и протоколы передачи данных

В различных сетях существуют различные процедуры обмена информацией в сети. Эти процедуры называются протоколами передачи данных, которые описывают методы доступа к сетевым каналам данных.

Наибольшее распространение получили конкретные реализации методов доступа: Ethernet, Arcnet и Token-Ring.

### Локальная сеть Token Ring

Этот стандарт разработан фирмой IBM. В качестве передающей среды применяется неэкранированная или экранированная витая пара (UPT или SPT) или оптоволокно. Скорость передачи данных 4 Мбит/с или 16Мбит/с. В качестве метода управле­ния доступом станций к передающей среде используется метод - маркерное кольцо (Тоken Ring). Основные положения этого метода:

1. устройства подключаются к сети по топологии кольцо;
2. все устройства, подключенные к сети, могут передавать данные, только получив разрешение на передачу (маркер);
3. в любой момент времени только одна станция в сети обладает таким правом.

***Типы пакетов.***

В IВМ Тоkеn Ring используются три основных типа пакетов:

1. пакет управление/данные (Data/Соmmand Frame);
2. маркер (Token);
3. пакет сброса (Аbort).

*Пакет Управление/Данные*. С помощью такого пакета выполняется передача данных или команд управления работой сети.

*Маркер.* Станция может начать передачу данных только после получения такого пакета, В одном кольце может быть только один маркер и, соответственно, только одна станция с правом передачи данных.

*Пакет Сброса.* Посылка такого пакета называет прекращение любых передач. В сети можно подключать компьютеры по топологии звезда или кольцо.

### Локальная сеть Arknet.

Arknet (Attached Resource Computer NETWork ) - простая, недорогая, надежная и достаточно гибкая архитектура локальной сети. Разработана корпорацией Datapoint в 1977 году. Впоследствии лицензию на Аrcnet приобрела корпорация SМС (Standard Microsistem Corporation), которая стала основным разработчиком и производителем оборудования для сетей Аrcnet. В качестве передающей среды используются витая пара, коаксиальный кабель (RG-62) с волновым сопротивлением 93 Ом и оптоволоконный кабель. Скорость передачи данных - 2,5 Мбит/с. При подключении устройств в Аrcnet применяют топологии шина и звезда.

Этот метод получил широкое распространение в основном благодаря тому, что оборудование Arcnet дешевле, чем оборудование Ethernet или Token - Ring. Arcnet используется в локальных сетях с топологией "звезда". Один из компьютеров создает специальный маркер (сообщение специального вида), который последовательно передается от одного компьютера к другому.

Если станция желает передать сообщение другой станции, она должна дождаться маркера и добавить к нему сообщение, дополненное адресами отправителя и назначения. Когда пакет дойдет до станции назначения, сообщение будет "отцеплено" от маркера и передано станции.

Метод управления доступом станций к передающей среде - маркерная шина (Тоken Bus). Этот метод предусматривает следующие правила:

1. Все устройства, подключенные к сети, могут передавать данные
2. только получив разрешение на передачу (маркер);
3. В любой момент времени только одна станция в сети обладает таким правом;
4. Данные, передаваемые одной станцией, доступны всем станциям сети.

**Основные принципы работы.**

Передача каждого байта в Аrcnet выполняется специальной посылкой ISU(Information Symbol Unit - единица передачи информации), состоящей из трех служебных старт/стоповых битов и восьми битов данных. В начале каждого пакета передается начальный разделитель АВ (Аlегt Вurst), который состоит из шести служебных битов. Начальный разделитель выполняет функции преамбулы пакета.

В Аrcnet определены 5 типов пакетов:

1. *Пакет IТТ* (Information To Transmit) - приглашение к передаче. Эта посылка передает управление от одного узла сети другому. Станция, принявшая этот пакет, получает право на передачу данных.
2. *Пакет FBE* (Free Buffeг Еnquiries) - запрос о готовности к приему данных. Этим пакетом проверяется готовность узла к приему данных.
3. *Пакет данных.* С помощью этой посылки производиться передача данных.
4. *Пакет АСК (*ACKnowledgments) - подтверждение приема. Подтверждение готовности к приему данных или подтверждение приема пакета данных без ошибок, т.е. в ответ на FBE и пакет данных.
5. *Пакет NAK* ( Negative AcKnowledgments) - неготовность к приему. Неготовность узла к приему данных ( ответ на FBE ) или принят пакет с ошибкой.

В сети Arknet можно использовать две топологии: звезда и шина.

### Локальная сеть Ethernet

Это метод доступа, разработанный фирмой Xerox в 1975 году, пользуется наибольшей популярностью. Он обеспечивает высокую скорость передачи данных и надежность. Позднее к этому проекту присоединились компании Digital Equipment Corporation (DEC) и Intel Corporation. В 1982 году была опубликована спецификация на Ethernet версии 2.0. На базе Ethernet институтом IEEE был разработан стандарт IEEE 802.3. Различия между ними незначительные.

Для данного метода доступа используется топология "общая шина". Поэтому сообщение, отправляемое одной рабочей станцией, принимается одновременно всеми остальными, подключенными к общей шине. Но сообщение, предназначенное только для одной станции (оно включает в себя адрес станции назначения и адрес станции отправителя). Та станция, которой предназначено сообщение, принимает его, остальные игнорируют.

Метод доступа Ethernet является методом множественного доступа с прослушиванием несущей и разрешением коллизий (конфликтов) (CSMA/CD - Carier Sense Multiple Access with Collision Detection).

Перед началом передачи рабочая станция определяет, свободен канал или занят. Если канал свободен, станция начинает передачу.

Ethernet не исключает возможности одновременной передачи сообщений двумя или несколькими станциями. Аппаратура автоматически распознает такие конфликты, называемые коллизиями. После обнаружения конфликта станции задерживают передачу на некоторое время. Это время небольшое и для каждой станции свое. После задержки передача возобновляется.

Реально конфликты приводят к уменьшению быстродействия сети только в том случае, если работает порядка 80-100 станций.

**Основные принципы работы.**

На логическом уровне в Ethernet применяется топология шина :

1. все устройства, подключенные к сети, равноправны, т.е. любая станция может начать передачу в любой момент времени( если передающая среда свободна);
2. данные, передаваемые одной станцией, доступны всем станциям сети.

## Сетевые операционные системы для локальных сетей.

Основное направление развития современных Сетевых Операционных Систем (Network Operation System - NOS ) - перенос вычислительных операций на рабочие станции, создание систем с распределенной обра­боткой данных. Это в первую очередь связано с ростом вычислительных возможностей персональных компьютеров и все более активным внедрением мощных многозадачных операционных систем: OS/2, Windows NТ, Windows 95. Кроме этого внедрение объектно-ориентированных технологий (ОLЕ, DСЕ, IDAPI) позволяет упростить организацию распределенной обработки данных. В такой ситуации основной задачей NOS становится объединение неравноценных операционных систем рабочих станций и обеспечение транспортного уровня для широкого круга задач: обработка баз данных, передача сообщений, управление распределенными ресурсами сети (directoгу/namе service).

В современных NOS применяют три основных подхода к организации управления ресурсами сети.

Первый - это Таблицы Объектов (Bindery). Используется в сетевых операционных системах NetWare v3.1х. Такая таблица находится на каждом файловом сервере сети. Она содержит информацию о пользователях, группах, их правах доступа к ресурсам сети (данным, сервисным услугам и т.п.). Такая организация работы удобна, если в сети только один сервер. В этом случае требуется определить и контролировать только одну информационную базу. При расширении сети, добавлении новых серверов объем задач по управлению ресурсами сети резко возрастает. Администратор системы вынужден на каждом сервере сети определять и контролировать работу пользователей. Абоненты сети, в свою очередь, должны точно знать, где расположены те или иные ресурсы сети, а для получения доступа к этим ресурсам - регистрироваться на выбранном сервере. Конечно, для информационных систем, состоящих из большого количества серверов, такая организация работы не подходит.

Второй подход используется в LANServer и LANMahager - Структура Доменов (Domain). Все ресурсы сети и пользователи объединены в группы. Домен можно рассматривать как аналог таблиц объектов (bindery), только здесь такая таблица является общей для нескольких серверов, при этом ресурсы серверов являются общими для всего домена. Поэтому пользователю для того чтобы получить доступ к сети, достаточно подключиться к домену (зарегистрироваться), после этого ему ста­новятся доступны все ресурсы домена, ресурсы всех серверов и устройств, входящих в состав домена. Однако и с использованием этого под­хода также возникают проблемы при построении информационной системы с большим количеством пользователей, серверов и, соответственно, доменов. Например, сети для предприятия или большой разветвленной организации. Здесь эти проблемы уже связаны с организацией взаимодействия и управления несколькими доменами, хотя по содержанию они такие же, как и в первом случае.

Третий подход - Служба Наименований Директорий или Каталогов (Directory Name Services - DNS) лишен этих недостатков. Все ресурсы сети: сетевая печать, хранение данных, пользователи, серверы и т.п. рассматриваются как отдельные ветви или директории информационной системы. Таблицы, определяющие DNS, находятся на каждом сервере. Это, во-первых, повышает надежность и живучесть системы, а во-вторых, упрощает обращение пользователя к ресурсам сети. Зарегистрировавшись на одном сервере, пользователю становятся доступны все ресурсы сети. Управление такой системой также проще, чем при использовании доменов, так как здесь существует одна таблица, определяющая все ресурсы сети, в то время как при доменной организации необходимо определять ресурсы, пользователей, их права доступа для каждого домена отдельно.

В настоящее время по оценке компании IDC наиболее распростра­ненными являются следующие сетевые операционные системы:

1. NetWare v2.х и vЗ.х, Nowell Inc. 65%
2. LAN Server, IВМ Согр. 14%
3. LAN Manager, Microsoft Corp. 3%
4. VINES, Ваnуаn Systems Inc. 2%

Рассмотрим более подробно возможности этих и некоторых других сетевых операционных систем и требования, которые они предъявляют к программному и аппаратному обеспечению устройств сети.

### NetWare 3.11, Nowell Inc.

*Отличительные черты:*

1. самая эффективная файловая система среди современных NOS;
2. самый широкий выбор аппаратного обеспечения

***Основные характеристики и требования к аппаратному обеспечению.***

1. Центральный процессор: 38б и выше.
2. Минимальный объем жесткого диска: 9 МБайт.
3. Объем ОП (Оперативной Памяти) на сервере: 4 МБайт - 4ГБайт.
4. Минимальный объем ОП РС (Рабочей Станции) клиента: б40 Кбайт.
5. Операционная система: собственная разработка Nowell
6. Протоколы: IРХ/SРХ.
7. Мультипроцессорность: нет.
8. Количество пользователей: 250.
9. Максимальный размер файла: 4ГБайт.
10. Шифрование данных: нет.
11. Монитор UPS: есть.
12. ТТS: есть.
13. Управление распределенными ресурсами сети: таблицы bindeгу на сервере.
14. Система отказоустойчивости: дублирование дисков, зеркальное отражение дисков, SFT II,SFT III, поддержка накопителя на магнитной ленте, резервное копирование таблиц bindery и данных.
15. Компрессирование данных: нет.
16. Фрагментация блоков ( Block suballocations): нет.
17. Файловая система клиентов: DOS, Windows, Мас(доп.), ОS/2(доп.), UNIX(доп.), Windows NT.

### LAN Server, IВМ Согр.

*Отличительные черты:*

1. использование доменной организации сети упрощает управление и доступ к ресурсам сети;
2. обеспечивает полное взаимодействие с иерархическими системами (архитектурой SNА).

Целостная операционная система с широким набором услуг. Рабо­тает на базе ОS/2, поэтому сервер может быть невыделенным (nondedicated). Обеспечивает взаимодействие с иерархическими системами, поддерживает межсетевое взаимодействие.

Выпускаются две версии LAN Server: Entry и Advanced. Advanced в отличие от Entry поддерживает высокопроизводительную файловую систему (High Perfomance File System - HPFS). Она включает системы отказоустойчивости (Fail Tolerances) и секретности (Local Security).

Серверы и пользователи объединяются в домены. Серверы в до­мене работают как единая логическая система. Все ресурсы домена дос­тупны пользователю после регистрации в домене. В одной кабельной системе могут работать несколько доменов. При использовании на рабочей станции OS/2 ресурсы этих станций доступны пользователям других рабочих станций, но только одному в данное время. Администратор может управлять работой сети только с рабочей станции, на которой установлена операционная система OS/2. LAN Server поддерживает удален­ную загрузку рабочих станций DOS, OS/2 и Windows (Remote Interface Procedure Load - RIPL).

К недостаткам можно отнести:

1. сложная процедура установки NOS;
2. ограниченное количество поддерживаемых драйверов сетевых адаптеров.

***Основные характеристики и требования к аппаратному обеспечению.***

1. Центральный процессор: 38б и выше.
2. Минимальный объем жесткого диска: 4.6 МБайт для клиента (requestor)/7.2 МБайт для сервера.
3. Минимальный объем ОП на сервере: 1.3 МБайт - 16 МБайт.
4. Минимальный объем ОП РС клиента: 4.2 Мбайт для OS/2, 640 КБайт для DOS.
5. Операционная система: OS/2 2.х.
6. Протоколы: NetBIOS, ТСР/IР.
7. Мультипроцессорность: поддерживается.
8. Количество пользователей: 1016.
9. Максимальный размер файла: 2 Гбайт.
10. Шифрование данных: нет.
11. Монитор UPS: есть.
12. ТТS: есть.
13. Управление распределенными ресурсами сети: домены.
14. Система отказоустойчивости: дублирование дисков, зеркальное отражение дисков, поддержка накопителя на магнитной ленте, резервное копирование таблиц домена.
15. Компрессирование данных: нет.
16. Фрагментация блоков (Block suballocation): нет.
17. Файловая система клиентов: DOS, Windows, Мас (доп.), OS/2, UNIX, Windows NT (доп.).

### Windows NT Advanced Server 3.1, Microsoft Corp.

*Отличительные черты:*

1. простота интерфейса пользователя
2. доступность средств разработки прикладных программ и поддержка прогрессивных объектно-ориентированных технологий

Всё это привело к тому, что эта операционная система может стать одной из самых популярных сетевых операционных систем.

Интерфейс напоминает оконный интерфейс Windows 3.1, инсталяция занимает около 20 минут. Модульное построение системы упрощает внесение изменений и перенос на другие платформы. Обеспечивается защищенность подсистем от несанкционированного доступа и от их взаимного влияния (если зависает один процесс, это не влияет на работу остальных). Есть поддержка удаленных станций - Remote Access Service (RAS), но не поддерживается удаленная обработка заданий.

Windows NT предъявляет более высокие требования к производительности компьютера по сравнению с NetWare.

***Основные характеристики и требования к аппаратному обеспечению.***

1. Центральный процессор: 386 и выше, MIPS, R4000, DEC Alpha АХР.
2. Минимальный объем жесткого диска: 90 Мбайт.
3. Минимальный объем ОП на сервере: 16 Мбайт.
4. Минимальный объем ОП РС клиента; 12 Мбайт для NТ/512 КБайт для DOS.
5. Операционная система: Windows NT.
6. Протоколы: NetBEUI, ТСР/IР, IРХ/SРХ, АррlеТаlk, АsyncBEUI.
7. Мультипроцессорность: поддерживается.
8. Количество пользователей: неограниченно.
9. Максимальный размер файла: неограничен.
10. Шифрование данных: уровень С-2.
11. Монитор UPS: есть.
12. ТТS: есть.
13. Управление распределенными ресурсами сети: домены.
14. Система отказоустойчивости: дублирование дисков, зеркальное отражение дисков, RAID 5, поддержка накопителя на магнитной ленте, резервное копирование таблиц домена и данных.
15. Компрессирование данных: нет.
16. Фрагментация блоков (Block suballocation): нет.
17. Файловая система клиентов: DOS, Windows, Мас, ОS/2, UNIX, Windows NT.

### NetWare 4, Nowell Inc.

*Отличительная черта:*

1. применение специализированной системы управления ресурсами сети (NetWare Directory Services - NDS) позволяет строить эффективные информационные системы с количеством пользователей до 1000. В NDS определены все ресурсы, услуги и пользователи сети. Эта информация распределена по всем серверам сети.

Для управления памятью используется только одна область (рооl), поэтому оперативная память, освободившаяся после выполнения каких-либо процессов, становится сразу доступной операционной системе (в отличие от NetWare 3).

Новая система управления хранением данных (Data Storage Managment) состоит из трех компонент, позволяющих повысить эффективность файловой системы:

1. Фрагментация Блоков или Разбиение Блоков Данных на Подблоки (Block Suballocation). Если размер блока данных на томе 64 КБайта, а требуется записать файл размером 65 КБайт, то ранее потребовалось бы выделить 2 блока по б4 Кбайта. При этом 6З Кбайта во втором блоке не могут использоваться для хранения других данных. В NetWare 4 система выделит в такой ситуации один блок размером 64 КБайта и два блока по 512 Байт. Каждый частично используемый блок делится на подблоки по 512 Байт, свободные подблоки доступны системе при записи других файлов.

2. Упаковка Файлов (File Compression). Долго не используемые данные система автоматически компрессирует, упаковывает, экономя таким образом место на жестких дисках. При обращении к этим данным автоматически выполняется декомпрессия данных.

3. Перемещение Данных (Data Migration). Долго не используемые данные система автоматически копирует на магнитную ленту либо другие носители, экономя таким образом место на жестких дисках.

Встроенная поддержка Протокола Передачи Серии Пакетов (Packet-Burst Migration). Этот протокол позволяет передавать несколько пакетов без ожидания подтверждения о получении каждого пакета. Подтверждение передается после получения последнего пакета из серии.

При передаче через шлюзы и маршрутизаторы обычно выполня­ется разбиение передаваемых данных на сегменты по 512 Байт, что уменьшает: скорость передачи данных примерно на 20%. Применение в NetWare 4 протокола LIP (Large Internet Packet) позволяет повысить эффективность обмена данными между сетями, так как в этом случае разбиение на сегменты по 512 Байт не требуется.

Все системные сообщения и интерфейс используют специальный модуль. Для перехода к другому языку достаточно поменять этот модуль или добавить новый. Возможно одновременное использование не­скольких языков: один пользователь при работе с утилитами использует английский язык, а другой в это же время немецкий.

## Среда Delphi как средство разработки ПО баз даных.

Реализация дипломной работы проводится в системе программирования Delphi 5.0, располагающей широкими возможностями по созданию приложений баз данных. Уже с более ранних версии система Delphi снабжена необходимым набором драйверов для доступа к самым известным форматам баз данных, удобными и развитыми средствами для доступа к информации, расположенной как на локальном диске, так и на удаленном сервере. В поставку продукта входит большое количество коллекций визуальных компонент для построения отображаемых на экране окон, что необходимо для создания удобного интерфейса между пользователем и исполняемым кодом.

Поскольку использование баз данных является одним из краеугольных камней, на которых построено существование различных организаций, пристальное внимание разработчиков приложений баз данных вызывают инструменты, при помощи которых такие приложения можно было бы создавать. Выдвигаемые к ним требования в общем виде можно сформулировать как: "быстрота, простота, эффективность, надежность".

Среди большого разнообразия продуктов для разработки приложений Delphi занимает одно из ведущих мест. Delphi отдают предпочтение разработчики с разным стажем, привычками, профессиональными интересами. С помощью Delphi написано колоссальное количество приложений, десятки фирм и тысячи программистов-одиночек разрабатывают для Delphi дополнительные компоненты.[4].

В основе такой общепризнанной популярности лежит тот факт, что Delphi, как никакая другая система программирования, удовлетворяет изложенным выше требованиям. Действительно, приложения с помощью Delphi разрабатываются быстро, причем взаимодействие разработчика с интерактивной средой Delphi не вызывает внутреннего отторжения, а наоборот, оставляет ощущение комфорта. Delphi-приложения эффективны, если разработчик соблюдает определенные правила (и часто - если не соблюдает). Эти приложения надежны и при эксплуатации обладают предсказуемым поведением.[4, 22].

Пакет Delphi - продолжение линии компиляторов языка Pascal корпорации Borland. Pascal как язык очень прост, а строгий контроль типов данных способствует раннему обнаружению ошибок и позволяет быстро создавать надежные и эффективные программы. Корпорация Borland постоянно обогащала язык. Когда-то в версию 4.0 были включены средства раздельной трансляции, позже, начиная с версии 5.5, появились объекты, а в состав шестой версии пакета вошла полноценная библиотека классов Turbo Vision, реализующая оконную систему в текстовом режиме работы видеоадаптера. Это был один из первых продуктов, содержавших интегрированную среду разработки программ.

В классе инструментальных средств для начинающих программистов продуктам компании Borland пришлось конкурировать со средой Visual Basic корпорации Microsoft, где вопросы интеграции и удобства работы были решены лучше. Когда в начале 70-х годов Н. Вирт опубликовал сообщение о Pascal, это был компактный, с небольшим количеством основных понятий и зарезервированных слов язык программирования, нацеленный на обучение студентов. Язык, на котором предстоит работать пользователю Delphi, отличается от исходного не только наличием множества новых понятий и конструкций, но и идейно: в нем вместо минимизации числа понятий и использования самых простых конструкций (что, безусловно, хорошо для обучения, но не всегда оправдано в практической работе), предпочтение отдается удобству работы профессионального пользователя. Как язык Turbo Pascal естественно сравнивать с его ближайшими конкурентами - многочисленными вариациями на тему языка Basic (в первую очередь с Visual Basic корпорации Microsoft) и с C++.[4, 6]. Turbo Pascal существенно превосходит Basic за счет полноценного объектного подхода, включающего в себя развитые механизмы *инкапсуляции, наследование и полиморфизм*. Последняя версия языка, применяемая в Delphi, по своим возможностям приближается к C++. Из основных механизмов, присущих C++, отсутствует только множественное наследование. (Впрочем, этим красивым и мощным механизмом порождения новых классов пользуется лишь небольшая часть программистов, пишущих на С++.)

Плюсы применения языка Pascal очевидны: с одной стороны, в отличие от Visual Basic, основанного на интерпретации промежуточного кода, для него имеется компилятор, генерирующий машинный код, что позволяет получать значительно более быстрые программы. С другой - в отличие от C++ синтаксис языка Pascal способствует построению очень быстрых компиляторов. [6].

Среда программирования представляет собой несколько отдельных окон: меню и инструментальные панели, Object Inspector (в котором можно видеть свойства объекта и связанные с ним события), окна визуального построителя интерфейсов (Visual User Interface Builder), Object Browser (позволяющее изучать иерархию классов и просматривать списки их полей, методов и свойств), окна управления проектом (Project Manager) и редактора.

Delphi содержит полноценный текстовый редактор типа Brief, назначения клавиш в котором соответствуют принятым в Windows стандартам, а глубина иерархии операций Undo неограниченна. Как это стало уже обязательным, реализовано цветовое выделение различных лексических элементов программы. Процесс построения приложения достаточно прост. Нужно выбрать форму (в понятие формы входят обычные, диалоговые, родительские и дочерние окна MDI), задать ее свойства и включить в нее необходимые компоненты (видимые и, если понадобится, неотображаемые): меню, инструментальные панели, строку состояния и т. п., задать их свойства и далее написать (с помощью редактора исходного кода) обработчики событий. Object Browser Окна типа Object Browser стали неотъемлемой частью систем программирования на объектно-ориентированных языках. Работа с ними становится возможной сразу после того, как вы скомпилировали приложение.

Projeсt Manager - это отдельное окно, где перечисляются модули и формы, составляющие проект. При каждом модуле указывается маршрут к каталогу, в котором находится исходный текст. Жирным шрифтом выделяются измененные, но еще не сохраненные части проекта. В верхней части окна имеется набор кнопок: добавить, удалить, показать исходный текст, показать форму, задать опции и синхронизировать содержимое окна с текстом файла проекта, т. е. с головной программой на языке Pascal.

Опции, включая режимы компиляции, задаются для всего проекта в целом. В этом отношении традиционные make-файлы, используемые в компиляторах языка C, значительно более гибки.

Visual Component Library (VCL) Богатство палитры объектов для построения пользовательского интерфейса - один из ключевых факторов при выборе инструмента визуального программирования. При этом для пользователя имеет значение как число элементов, включенных непосредственно в среду, так и доступность элементов соответствующего формата на рынке. [4, 22].

### Высокопроизводительный компилятор в машинный код

Компиляторы языка Pascal компании Borland никогда не заставляли пользователя подолгу ждать результатов компиляции. Производители утверждают, что на сегодня данный компилятор - самый быстрый в мире. Компилятор, встроенный в Delphi позволяет обрабатывать до 390 тыс. строк исходного текста в минуту на машине Pentium-100. Он предлагает легкость разработки и быстрое время проверки готового программного блока, характерного для языков четвертого поколения (4GL) и в то же время обеспечивает качество кода, характерного для компилятора 3GL.

В смысле проектирования Delphi мало чем отличается от проектирования в интерпретирующей среде, однако после выполнения компиляции мы получаем код, который исполняется в 10-20 раз быстрее, чем тоже самое, сделанное при помощи интерпретатора. Кроме того, компилятор компилятору рознь, в Delphi компиляция производится непосредственно в родной машинный код, в то время как существуют компиляторы, превращающие программу в так называемый p-код, который затем интерпретируется виртуальной p-машиной. Это не может не сказаться на фактическом быстродействии готового приложения.

Следует отметить также, что благодаря опции оптимизации сегментов удается существенно сократить размер выполняемого файла. Можно запустить компилятор в режиме проверки синтаксиса. При этом наиболее длительная операция компоновки и изготовления исполняемого файла выполняться не будет.

Вероятно, то обстоятельство, что Delphi позиционируется как средство создания приложений, взаимодействующих с базами данных, и ориентировано преимущественно на рынок инструментальных средств клиент/сервер, где до настоящего момента доминируют интерпретируемые языки, позволило его авторам не задумываться над созданием оптимизирующего компилятора, способного использовать все достоинства архитектур современных процессоров. [22].

### Мощный объектно-ориентированный язык

Совместимость с программами, созданными ранее средствами Borland Pascal, сохраняется, несмотря на то, что в язык внесены существенные изменения. Необходимость в некоторых усовершенствованиях давно ощущалась. Самое заметное из них - аппарат исключительных ситуаций, подобный тому, что имеется в C++, был первым реализован в компиляторах корпорации Borland. Не секрет, что при написании объектно-ориентированных программ, активно работающих с динамической памятью и другими ресурсами, немалую трудность представляет аккуратное освобождение этих ресурсов в случае возникновения нештатных ситуаций. Особенно это актуально для среды Windows, где число видов ресурсов довольно велико, а неправильная работа с ними может быстро привести к зависанию всей системы. Предусмотренный в Delphi аппарат исключений максимально упрощает кодирование обработки нештатных ситуаций и освобождения ресурсов.

Объектно-ориентированный подход в новой версии языка получил значительное развитие. Перечислим основные новшества.

введено понятие класса.

реализованы методы классов, аналогичные статическим методам C++. Они оперируют не экземпляром класса, а самим классом.

механизм *инкапсуляции* во многом усовершенствован. Введены защищенные поля и методы, которые, подобно приватным, не видны извне, но отличаются от них тем, что доступны из методов класса- наследника.

введена обработка исключительных ситуаций. В Delphi это устроено в стиле С++. Исключения представлены в виде объектов, содержащих специфическую информацию о соответствующей ошибке (тип и место- нахождение ошибки). Разработчик может оставить обработку ошибки, существовавшую по умолчанию, или написать свой собственный обработчик. Обработка исключений реализована в виде *exception-handling blocks* (также еще называется *protected blocks*), которые устанавливаются ключевыми словами try и end. Существуют два типа таких блоков: *try...except* и *try...finally*.

появилось несколько удобных синтаксических конструкций, в числе которых преобразование типа объекта с контролем корректности (в случае неудачи инициируется исключение) и проверка объекта на принадлежность классу.

Ссылки на классы придают дополнительный уровень гибкости, так, когда вы хотите динамически создавать объекты, чьи типы могут быть известны только во время выполнения кода. К примеру, ссылки на классы используются при формировании пользователем документа из разного типа объектов, где пользователь набирает нужные объекты из меню или палитры. Собственно, эта технология использовалась и при построении Delphi.

введено средство, известное как механизм делегирования. Под *делегированием* понимается то, что некий объект может предоставить другому объекту отвечать на некоторые события. Он используется в Delphi для упрощения программирования событийно-ориентированных частей программ, т. е. пользовательского интерфейса и всевозможных процедур, запускаемых в ответ на манипуляции с базой данных.

После того как Borland внесла перечисленные изменения, получился мощный объектно-ориентированный язык, сопоставимый по своим возможностям с C++. Платой за новые функции стало значительное повышение требований к профессиональной подготовке программиста.

Язык программирования Delphi базируется на Borland Object Pascal.

Кроме того, Delphi поддерживает такие низкоуровневые особенности, как подклассы элементов управления Windows, перекрытие цикла обработки сообщений Windows, использование встроенного ассемблера.[22].

### Объектно-ориентированная модель программных компонент

Основной упор этой модели в Delphi делается на максимальном повторном использовании кода. Это позволяет разработчикам строить приложения весьма быстро из заранее подготовленных объектов, а также дает им возможность создавать свои собственные объекты для среды Delphi. Никаких ограничений по типам объектов, которые могут создавать разработчики, не существует. Действительно, все в Delphi написано на нем же, поэтому разработчики имеют доступ к тем же объектам и инструментам, которые использовались для создания среды разработки. В результате нет никакой разницы между объектами, поставляемыми Borland или третьими фирмами, и объектами, которые можно создать самостоятельно.

В стандартную поставку Delphi входят основные объекты, которые образуют удачно подобранную иерархию из 270 базовых классов. На Delphi можно одинаково хорошо писать как приложения к корпоративным базам данных, так и, к примеру, игровые программы. Во многом это объясняется тем, что традиционно в среде Windows было достаточно сложно реализовывать пользовательский интерфейс. Событийная модель в Windows всегда была сложна для понимания и отладки. Но именно разработка интерфейса в Delphi является самой простой задачей для программиста.

Благодаря такой возможности приложения, изготовленные при помощи Delphi, работают надежно и устойчиво. Delphi поддерживает использование уже существующих объектов, включая DLL, написанные на С и С++, OLE сервера, VBX, объекты, созданные при помощи Delphi. Из готовых компонент работающие приложения собираются очень быстро. Кроме того, поскольку Delphi имеет полностью объектную ориентацию, разработчики могут создавать свои повторно используемые объекты для того, чтобы уменьшить затараты на разработку.

Delphi предлагает разработчикам - как в составе команды, так и индивидуальным - открытую архитектуру, позволяющую добавлять компоненты, где бы они ни были изготовлены, и оперировать этими вновь введенными компонентами в визуальном построителе. Разработчики могут добавлять CASE-инструменты, кодовые генераторы, а также авторские help’ы, доступные через меню Delphi. [22].

### Библиотека визуальных компонент

Компоненты, используемые при разработке в Delphi, встроены в среду разработки приложений и представляют из себя набор типов объектов, используемых в качестве фундамента при строительстве приложения.

Этот костяк называется Visual Component Library (VCL). В VCL есть такие стандартные элементы управления, как строки редактирования, статические элементы управления, строки редактирования со списками, списки объектов. Еще имеются такие компоненты, которые ранее были доступны только в библиотеках третьих фирм: табличные элементы управления, закладки, многостраничные записные книжки. Все объекты разбиты на страницы по своей функциональности и представленны в *палитре компонент*.

VCL содержит специальный объект, предоставлющий интерфейс графических устройств Windows, и позволяющий разработчикам рисовать, не заботясь об обычных для программирования в среде Windows деталях.

Ключевой особенностью Delphi является возможность не только использовать визуальные компоненты для строительства приложений, но и создание новых компонент. Такая возможность позволяет разработчикам не переходить в другую среду разработки, а наоборот, встраивать новые инструменты в существующую среду. Кроме того, можно улучшить или полностью заменить существующие по умолчанию в Delphi компоненты.

Здесь следует отметить, что обычных ограничений, присущих средам визуальной разработки, в Delphi нет. Сам Delphi написан при помощи Delphi, что говорит об отсутствии таких ограничений.

Классы объектов построены в виде иерархии, состоящей из абстрактных, промежуточных, и готовых компонент. Разработчик может пользоваться готовыми компонентами, создавать собственные на основе абстрактных или промежуточных, а также создавать собственные объекты. Рассмотрим некоторые из них.

TMainMenu позволяет поместить главное меню в программу. При помещении TMainMenu на форму это выглядит, как просто иконка. Иконки данного типа называют *невизуальным компонентом*, поскольку они невидимы во время выполнения программы.



TPopupMenu позволяет создавать всплывающие меню. Этот тип меню появляется по щелчку правой кнопки мыши на объекте, к которому привязано данное меню. У всех видимых объектов имеется свойство PopupMenu, где и указывается нужное меню. Создается PopupMenu аналогично главному меню.



TLabel служит для отображения текста на экране. Можно изменить шрифт и цвет метки, если дважды щелкнуть на свойство Font в Инспекторе Объектов. Это легко сделать и во время выполнения программы, написав всего одну строчку кода.



TEdit - стандартный управляющий элемент Windows для ввода. Он может быть использован для отображения короткого фрагмента текста и позволяет пользователю вводить текст во время выполнения программы.



TMemo - иная форма TEdit. Подразумевает работу с большими текстами. TMemo может переносить слова, сохранять в ClipBoard фрагменты текста и восстанавливать их, и другие основные функции редактора. TMemo имеет ограничения на объем текста в 32Кб, это составляет 10-20 страниц (есть подобные компоненты, где этот предел снят).



TButton позволяет выполнить какие-либо действия при нажатии кнопки во время выполнения программы. В Delphi все делается очень просто. Поместив TButton на форму, по двойному щелчку можно создать заготовку обработчика события нажатия кнопки.



TCheckBox отображает строку текста с маленьким окошком рядом. В окошке можно поставить отметку, которая означает, что что-то выбрано.



TRadioButton позволяет выбрать только одну опцию из нескольких.



TListBox нужен для показа прокручиваемого списка. Классический пример ListBox’а в среде Windows - выбор файла из списка в пункте меню File | Open многих приложений. Названия файлов или директорий и находятся в ListBox’е.



TComboBox во многом напоминает ListBox, за исключением того, что позволяет водить информацию в маленьком поле ввода сверху ListBox. Есть несколько типов ComboBox, но наиболее популярен спадающий вниз (drop-down combo box), который можно видеть внизу окна диалога выбора файла.



TScrollbar - полоса прокрутки, появляется автоматически в объектах редактирования, ListBox’ах при необходимости прокрутки текста для просмотра.

TGroupBox используется для визуальных целей и для указания Windows, каков порядок перемещения по компонентам на форме (при нажатии клавиши TAB).



TRadioGroup используется аналогично TGroupBox, для группировки объектов TRadioButton.



TPanel - управляющий элемент, похожий на TGroupBox, используется в декоративных целях. Чтобы использовать TPanel, можно просто поместить его на форму и затем положите другие компоненты на него. Теперь при перемещении TPanel будут передвигаться и эти компоненты. TPanel используется также для создания линейки инструментов и окна статуса.



TBitBtn - кнопка вроде TButton, однако на ней можно разместить картинку (glyph). TBitBtn имеет несколько предопределенных типов (bkClose, bkOK и др), при выборе которых кнопка принимает соответствующий вид. Кроме того, нажатие кнопки на модальном окне приводит к закрытию окна с соответствующим модальным результатом.



TSpeedButton - кнопка для создания панели быстрого доступа к командам (SpeedBar). Пример - SpeedBar слева от Палитры Компонент в среде Delphi. Обычно на данную кнопку помещается только картинка (glyph).



TTabSet - горизонтальные закладки. Обычно используется вместе с TNoteBook для создания многостраничных окон. Название страниц можно задать в свойстве Tabs.



TNoteBook - используется для создания многостраничного диалога, на каждой странице располагается свой набор объектов. Используется совместно с TTabSet.



TTabbedNotebook - многостраничный диалог со встроенными закладками, в данном случае - закладки сверху.



TOutline - используется для представления иерархических отношений связанных данных. Например - дерево директорий.



TStringGrid - служит для представления текстовых данных в виде таблицы. Доступ к каждому элементу таблицы происходит через свойство Cell.



TDrawGrid - служит для представления данных любого типа в виде таблицы. Доступ к каждому элементу таблицы происходит через свойство CellRect.



TImage - отображает графическое изображение на форме. Воспринимает форматы BMP, ICO, WMF. Если картинку подключить во время дизайна программы, то она прикомпилируется к EXE файлу.



TShape - служит для отображения простейших графических объектов на форме: окружность, квадрат и т.п.



TBevel - элемент для рельефного оформления интерфейса.



THeader - элемент оформления для создания заголовков с изменяемыми размерами для таблиц.

TScrollBox - позволяет создать на форме прокручиваемую область с размерами большими, нежели экран. На этой области можно разместить свои объекты.



TTimer - таймер, событие OnTimer периодически вызывается через промежуток времени, указанный в свойстве Interval. Период времени может составлять от 1 до 65535 мс.



TPaintBox - место для рисования. В обработчики событий, связанных с мышкой передаются относительные координаты мышки в TPaintBox, а не абсолютные в форме.



TFileListBox - специализированный ListBox, в котором отображаются файлы из указанной директории (св-во Directory). На названия файлов можно наложить маску, для этого служит св-во Mask. Кроме того, в св-ве FileEdit можно указать объект TEdit для редактирования маски.



TDirectoryListBox - специализированный ListBox, в котором отображается структура директорий текущего диска. В св-ве FileList можно указать TFileListBox, который будет автоматически отслеживать переход в другую директорию.



TDriveComboBox - специализированный ComboBox для выбора текущего диска. Имеет свойство DirList, в котором можно указать TDirectoryListBox, который будет отслеживать переход на другой диск.

TFilterComboBox - специализированный ComboBox для выбора маски имени файлов. Список масок определяется в свойстве Filter. В свойстве FileList указывается TFileListBox, на который устанавливается маска.



С помощью последних четырех компонент (TFileListBox, TDirectoryListBox, TDriveComboBox, TFilterComboBox) можно построить свой собственный диалог выбора файла, причем для этого не потребуется написать ни одной строчки кода.



TOLEContainer - контейнер, содержащий OLE объекты. Поддерживается OLE 2.02

TDDEClientConv,TDDEClientItem, TDDEServerConv, TDDEServerItem - 4 объекта для организации DDE. С помощью этих объектов можно построить приложение как DDE-сервер, так и DDE-клиент.



TChartFX - деловая графика. Компонент позволяет строить всевозможные графики и гистограммы.



### Формы, модули и метод разработки “Two-Way Tools”

Формы - это объекты, в которые помещаются другие объекты для создания пользовательского интерфейса любого приложения. Модули состоят из кода, который реализует функционирование приложения, обработчики событий для форм и их компонент.

Информация о формах хранится в двух типах файлов - *.dfm* и *.pas*, причем первый тип файла - двоичный - хранит образ формы и ее свойства, второй тип описывает функционирование обработчиков событий и поведение компонент. Оба файла автоматически синхронизируются Delphi, так что если добавить новую форму проект, связанный с ним файл *.pas* автоматически будет создан, и его имя будет добавлено в проект.

Такая синхронизация и делает Delphi *two-way-инструментом*, обеспечивая полное соответствие между кодом и визуальным представлением. Как только добавляется новый объект или код, Delphi устанавливает т.н. *“кодовую синхронизацию”* между визуальными элементами и соответствующими им кодовыми представлениями.

*Two-way tools* - *однозначное соответствие между визуальным проектированием и классическим написанием текста программы* Это означает, что разработчик всегда может видеть код, соответствующий тому, что он построил при помощи визуальных инструментов и наоборот.

Визуальный построитель интерфейсов (Visual User-interface builder) дает возможность быстро создавать клиент-серверные приложения визуально, просто выбирая компоненты из соответствующей палитры. В процессе построения приложения разработчик выбирает из палитры компонент готовые компоненты как художник, делающий крупные мазки кистью. Еще до компиляции он видит результаты своей работы - после подключения к источнику данных их можно видеть отображенными на форме, можно перемещаться по данным, представлять их в том или ином виде.[4, 22].

### Масштабируемые средства для построения баз данных

Мощность и гибкость Delphi при работе с базами данных основана на низкоуровневом *ядре* - процессоре баз данных *Borland Database Engine* (BDE). Его интерфейс с прикладными программами называется *Integrated Database Application Programming Interface* (IDAPI). В принципе, сейчас не различают эти два названия (BDE и IDAPI) и считают их синонимами. BDE позволяет осуществлять доступ к данным как с использованием традиционного record-ориентированного (навигационного) подхода, так и с использованием set-ориентированного подхода, используемого в SQL-серверах баз данных. Кроме BDE, Delphi позволяет осуществлять доступ к базам данных, используя технологию (и, соответственно, драйверы) Open DataBase Connectivity (ODBC) фирмы Microsoft. Но, как показывает практика, производительность систем с использованием BDE гораздо выше, чем оных при использовании ODBC. ODBC драйвера работают через специальный “ODBC socket”, который позволяет встраивать их в BDE.

Все инструментальные средства баз данных Borland - Paradox, dBase, Database Desktop - используют BDE. Все особенности, имеющиеся в Paradox или dBase, “наследуются” BDE, и поэтому этими же особенностями обладает и Delphi.

Библиотека объектов содержит набор визуальных компонент, значительно упрощающих разработку приложений для СУБД с архитектурой клиент-сервер. Объекты инкапсулируют в себя нижний уровень - Borland Database Engine.

Предусмотрены специальные наборы компонент, отвечающих за доступ к данным, и компонент, отображающих данные. Компоненты доступа к данным позволяют осуществлять соединения с БД, производить выборку, копирование данных, и т.п.

Компоненты визуализации данных позволяют отображать данные виде таблиц, полей, списков. Отображаемые данные могут быть текстового, графического или произвольного формата.

Таблицы сохраняются в базе данных. Некоторые СУБД сохраняют базу данных в виде нескольких отдельных файлов, представляющих собой таблицы (в основном, все локальные СУБД), в то время как другие состоят из одного файла, который содержит в себе все таблицы и индексы (InterBase). Например, таблицы dBase и Paradox всегда сохраняются в отдельных файлах на диске. Директорий, содержащий dBase *.DBF* файлы или Paradox *.DB* файлы, рассматривается как база данных. Другими словами, любой директорий, содержащий файлы в формате Paradox или dBase, рассматривается Delphi как единая база данных. Для переключения на другую базу данных нужно просто переключиться на другой директорий. InterBase сохраняет все таблицы в одном файле, имеющем расширение *.GDB*, поэтому этот файл и есть база данных InterBase.

Объекты БД в Delphi основаны на SQL и включают в себя полную мощь Borland Database Engine. В состав Delphi также включен Borland SQL Link, поэтому доступ к СУБД Oracle, Sybase, Informix и InterBase происходит с высокой эффективностью. Кроме того, Delphi включает в себя локальный сервер Interbase для того, чтобы можно было разработать расширяемые на любые внешние SQL-сервера приложения в офлайновом режиме. Разработчик в среде Delphi, проектирующий информационную систему для локальной машины (к примеру, небольшую систему учета медицинских карточек для одного компьютера), может использовать для хранения информации файлы формата .dbf (как в dBase или Clipper) или .db (Paradox). Если же он будет использовать локальный InterBase for Windows 4.0 (это локальный SQL-сервер, входящий в поставку), то его приложение безо всяких изменений будет работать и в составе большой системы с архитектурой клиент-сервер.

*Масштабируемость* на практике - одно и то же приложение можно использовать как для локального, так и для более серьезного клиент-серверного вариантов.[4, 22].

### Настраиваемая среда разработчика

После запуска Delphi в верхнем окне горизонтально располагаются иконки палитры компонент. Если курсор задерживается на одной из иконок, под ней в желтом прямоугольнике появляется подсказка

Из этой палитры компонент можно выбирать компоненты, из которых можно строить приложения. Компоненты включают в себя как визуальные, так и логические компоненты. Такие вещи, как кнопки, поля редактирования - это визуальные компоненты; а таблицы, отчеты - это логические.

Поскольку в Delphi программа строится визуальным образом, все эти компоненты имеют свое графическое представление в поле форм для того, чтобы можно было бы ими соответствующим образом оперировать. Но для работающей программы видимыми остаются только визуальные компоненты. Компоненты сгруппированы на страницах палитры по своим функциям. К примеру, компоненты, представляющие Windows “common dialogs” все размещены на странице палитры с названием “Dialogs”.

Delphi позволяет разработчикам настроить среду для максимального удобства. Можно легко изменить палитру компонент, инструментальную линейку, а также настраивать выделение синтаксиса цветом.

В Delphi можно определить свою группу компонент и разместить ее на странице палитры, а если возникнет необходимость, перегруппировать компоненты или удалить неиспользуемые.

Интеллектуальный редактор. Редактирование программ можно осуществлять, используя запись и исполнение макросов, работу с текстовыми блоками, настраиваемые комбинации клавиш и цветовое выделение строк.

*Графический отладчик*. Delphi обладает мощнейшим, встроенным в редактор графическим отладчиком, позволяющим находить и устранять ошибки в коде. Можно установить точки останова, проверить и изменить переменные, при помощи пошагового выполнения в точности понять поведение программы. Если же требуются возможности более тонкой отладки, можно использовать отдельно доступный Turbo Debugger, проверив ассемблерные инструкции и регистры процессора.

*Инспектор объектов*. Этот инструмент представляет из себя отдельное окно, где вы можете в период проектирования программы устанавливать значения свойств и событий объектов (Properties & Events).

*Менеджер проектов.* Дает возможность разработчику просмотреть все модули в соответствующем проекте и снабжает удобным механизмом для управления проектами. Менеджер проектов показывает имена файлов, время/дату выбранных форм и пр. Можно немедленно попась в текст или форму, просто щелкнув мышкой на соответствующее имя.

*Навигатор объектов.* Показывает библиотеку доступных объектов и осуществляет навигацию по приложению. Можно посмотреть иерархию объектов, прекомпилированные модули в библиотеке, список глобальных имен вашего кода.

*Дизайнер меню.* Можно создавать меню, сохранить созданные в виде шаблонов и затем использовать в их в любом приложении.

*Эксперты.* Это набор инструментальных программ, облегчающих проектирование и настройку Ваших приложений. Есть возможность подключать самостоятельно разработанные эксперты. Потенциально это та возможность, при помощи которой третьи фирмы могут расширять Delphi CASE-инструментами, разработанными специально для Delphi. Включает в себя:

Эксперт форм, работающих с базами данных

Эксперт стилей и шаблонов приложений

Эксперт шаблонов форм

В состав RAD Pack входит эксперт для преобразования ресурсов, изготовленных в Borland Pascal 7.0, в формы Delphi. Уже появились эксперты, облегчающие построение DLL и даже написание собственных экспертов

*Интерактивная обучающая система.* Позволяет более полно освоить Delphi. Она являются не просто системой подсказок, а показывает возможности Delphi на самой среде разработчика.

### SQL сервер InterBase 5.1.1

*InterBase* - это система управления реляционными базами данных, поставляемая корпорацией BORLAND для построения приложений с архитектурой клиент-сервер произвольного масштаба: от сетевой среды небольшой рабочей группы с сервером под управлением Novell NetWare или Windows NT на базе IBM PC до информационных систем крупного предприятия на базе серверов IBM, Hewlett-Packard, SUN и т.п.

Для первичной разработки в пакет Delphi раньше входила однопользовательская версия InterBase для Windows - Local InterBase. Используя Local InterBase можно создавать и отлаживать приложения, работающие с данными по схеме клиент-сервер, без подключения к настоящему серверу. В дальнейшем потребуется только перенастроить используемый псевдоним базы данных и программа будет работать с реальной базой без перекомпиляции. Кроме того, Local InterBase можно использовать в приложениях для работы с данными вместо таблиц Paradox.

Данная дипломная работа базируется на многопользовательской версии сервера InterBase 5.1.1.

В состав пакета Delphi также входит множество утилит для работы и управления базами данных. Описание наиболее часто используемой программы для работы с базами данных в терминальном режиме приведено ниже.

*WISQL* (Windows Interactive SQL) - интерактивное средство посылки SQL-запросов к InterBase (в том числе и локальному InterBase), входящее в поставку Delphi, позволяет создавать таблицы - через посылку SQL-запросов. Database Desktop не обладает всеми возможностями по управлению SQL-серверными базами данных. Поэтому с помощью Database Desktop удобно создавать или локальные базы данных или только простейшие SQL-серверные базы данных, состоящие из небольшого числа таблиц, не очень сильно связанных друг с другом. Если же необходимо создать базу данных, состоящую из большого числа таблиц, имеющих сложные взаимосвязи, можно воспользоваться языком SQL. Можно записать всю последовательность SQL-предложений в один так называемый скрипт и послать его на выполнение. Конкретные реализации языка SQL незначительно отличаются в различных SQL-серверах, однако базовые предложения остаются одинаковыми для всех реализаций. Практика показывает, что если нет необходимости создавать таблицы во время выполнения программы, то лучше воспользоваться WISQL.



## Заключение

В начале работы над дипломным проектом возникла необходимость изучить ряд теоретических вопросов, связанных с выбором средств программной и аппаратной реализации поставленной задачи. К ним относится изучение принципов построения автоматизированных рабочих мест, локальных сетей и технологии клиент-сервер. Также были углублены знания относительно применения языка структурированных запросов SQL в среде программирования Delphi.

# Методический раздел

## Введение

После изучения вопросов, описанных выше в исследовательском разделе, были разработаны структура базы данных и интерфейс программы после чего она была создана. При этом пришлось решить ряд сложностей, выявленных уже на этапе программирования.

К созданному программному продукту была составлена документация, включая инструкцию по вводу в эксплуатацию и работе с программой.

Данные темы подробно раскрыты в этом разделе.

### Назначение АРМ «Отдел Кадров»

Приложение “Отдел кадров” предназначено для автоматизации деятельности отдела кадров организации. Система позволяет обеспечить многопользовательский доступ к единой базе данных организации, при этом возможна одновременная работа нескольких пользователей с информацией, что позволит гибко распределить обязанности между сотрудниками (например, регистрация новых сотрудников, модификация информации, поиск, формирование отчетов и прочее). При работе в многопользовательском режиме пользователям назначаются права доступа к информации (например, только начальник отдела кадров может редактировать справочники цехов завода). Однако система не исключает возможность работы и в локальном варианте - на одном рабочем месте.

Программа "Отдел Кадров" предназначена для облегчения условий труда сотрудников отдела кадров достаточно абстрактного предприятия. Наиболее рутинными и в то же время наиболее ответственными процессами являются:

* ввод метрики работника в его личную учетную карточку.
* ведение архива отдела кадров.
* подготовка различных отчетов по личным данным работников (стаж, зарплата, отгулы и так далее)

При этом сначала заводится так называемая личная карточка работника. В нее заносится вся необходимая метрика работника. Далее происходит работа с внесенной информацией (корректировка, поиск, удаление). Архив состоит из личных карточек уволенных сотрудников, т.к. после увольнения по соответствующему закону личные данные должны храниться не менее 3 лет до момента их уничтожения.

Дополнительно происходит наполнение информацией словарей базы данных. Словарь в данном контексте является справочником какой-либо однотипной информации (например справочник институтов, справочник районов и улиц). При вводе новой информации у пользователя есть возможность выбрать допустимые значения из справочников, либо пополнить его новыми данными.

На архив отдела кадров прямо или косвенно опирается ряд дополнительных операций (бухгалтерия, материальная часть, учет выработки продукции и другое).

Облегчения условий труда достигается благодаря возможности автоматизировать основные процессы ведения архива предприятия с помощью данной программы. Фактически, работа с личными карточками и архивом превращается в четкий и удобный процесс работы с базой данной. Это сильно упрощает работу и исключает ошибки, часто встречающиеся при обычной организации работы отдела кадров.

### Составные части программы

С точки зрения пользователя-оператора (т.е. сотрудника отдела кадров) существует лишь запускаемая программа АРМ Отдел Кадров. Вся работа с карточками и архивом происходит в пределах запущенной программы без необходимости вызова каких-либо других программ.

Однако программа состоит из двух логически раздельных блоков - базы данных и программы - оболочки.

База данных хранит всю необходимую информацию. К ней относятся данные непосредственно архива и служебная информация, необходимая для работы программы-оболочки. База данных абсолютно не имеет никакой привязки к оболочке, и к ее данным может обращатся какая-либо другая программа. Таким образом изначально заложена возможность развития всей программы ОК. Например, бухгалтерская программа может получать сведения о сотрудниках завода, обращаясь к указанной базе данных. При этом сама бухгалтерская программа может быть разработана другой группой программистов, без использования знаний о создании программы ОК.

Программа жестко привязана к базе данных. Она выполняет две наиболее выделяющихся функции. Во-первых она предоставляет данные из базы данных в удобном для пользователя виде а во-вторых производит различные манимуляции с хранящейся информацией (расчет,поиск,печать и т.д.).

В программе реализован принцип разделения прав доступа на изменение информации.

### Права доступа. Вход в программу

При работе с программой АРМ Отдел кадров первым шагом является авторизация пользователя. Авторизация определяет уровень доступа пользователя к хранящейся информации и возможность ее редактирования (как ввод новой , так и удаление существующей). Максимальный уровень доступа по установившейся тенденции принадлежит системному администратору. Он наделен правами регистрации новых пользователей в системе, изменения их паролей, полным доступом к хранящейся информации.

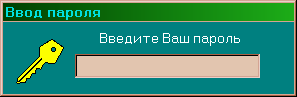
Данный подход диктовался максимальным приближением электронного АРМ к реальным условиям работы. Только начальник отдела кадров может удалить личную карточку сотрудника или изменить введенную информацию после того как она занесена в архив и считается проверенной на точность. При неэлектронной организации работ например, факт изменения возраста сотрудника или его стажа, не может расцениваться как нормальный.

Конфиденциальность информации обеспечивается фактически на двух уровнях – защита со стороны SQL сервера и описанное выше разграничение доступа. Защита от несанкционированного доступа со стороны сервера означает что клиентская программа

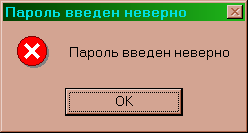
АРМ при соединении с базой данных является пользователем с точки зрения сервера. Поэтому чтобы подключение произошло он передает серверу авторизующую информацию (password и login). Системный администратор должен завести соответствующую учетную запись в настройках SQL сервера. Как это сделать – описано в приложении.

При старте программа выводит окно ввода пароля. У пользователя есть только 2 варианта продолжения работы – ввести корректный пароль и войти в систему или выйти из программы. Ввод некорректного пароля вызывает соответствующее информационное сообщение и естественно не приводит к входу в систему.

Внешний вид окна ввода пароля :



Внешний вид сообщения о некорректности пароля



### Справочники

Справочники в данном АРМ напоминают простые, неэлектронные справочники в традиционном толковании данного термина. Например справочник зарегистрированных профессий, справочник по цехам, справочник институтов и так далее. Назначение справочников – систематизировать хранящуюся информацию, увеличить скорость ввода новой информации и уменьшить кол-во возможных ошибок ввода. Это достигается тем , что все новые данные определенного типа (например название цеха) вносятся в справочник. Повторное использование одного и того же названия приводит фактически к многократному использованию одной и той же записи из таблицы справочника.

В программе существуют следующие виды словарей.

1. Образование
   1. Учебные заведения
   2. Виды образования
   3. Специальности по диплому
   4. Квалификации по диплому
2. Завод
   1. Цеха, отделы, склады
   2. Специальности
3. Военкоматы
   1. Военкоматы
   2. Составы
   3. Воинские звания
   4. Группы учета
4. Общие
   1. Национальности
   2. Семейные положения
   3. Страны
   4. Причины увольнения
5. Поощрения и почести
   1. Награды и поощрения
   2. Почетные звания
6. Муниципальные районы, улицы

В каждом справочнике хранится информация, соответствующая его названию. Доступ пользователя к справочникам определяется выданными ему правами доступа к программе. Возможен доступ только на чтение , с возможностью добавление новых данных, с возможностью добавления и удаления данных. Как и в других случаях, привилегии устанавливаются системным администратором.

### Личные карточки

На каждого сотрудника предприятия заводится его личная карточка, в которую заноситься его метрика. Личная карточка представлена несколькими разделами (закладками).

*Главная.* В нем заносятся и отображаются метрические данные сотрудника – Ф.И.О., дата рождения, семейное положение, номер паспорта, группа инвалидности и т.д. Также заносится его стаж и табельный номер внутри завода.

Кроме основной закладки «Главная» созданы еще следующие разделы

* Образование,
* Заграница
* Аттестация
* Награды и поощрения
* Почетные звания
* Поощрения по заводу
* Взыскания
* Работа по совместительству
* Отпуска
* Назначения и перемещения
* Последнее место работы
* Увольнение
* Воинский учет.

Кроме того, по каждому сотруднику можно ввести заметки вольного характера. Для этого надо выбрать раздел Заметки. При вводе информации происходит активное использование словарей, описанных ранее.

По уже введенным личным карточкам возможно провести поиск, сформировать отчет, осуществить печать и произвести их подсчет.

### Архив

Данные об уволившихся, либо ушедших в бессрочный отпуск, сотрудниках переводятся в архив отдела кадров предприятия. В программе для этого организован раздел Архив. Он хранит переведенные в него личные карточки. По архиву также возможен поиск и отбор информации. Кроме того любая карточка может быть как помещена в архив, так и восстановлена из него. Это соответствует повторному взятию на работу ранее работавшего на этом же предприятии сотрудника, либо его возврат после бессрочного отпуска.

При обычной организации работы отдела кадров поиск личного дела занимал определенное время. Затем повторно заводилась новая личная карточка и в нее рутинно переписывались данные из старой карточки. При использовании программы АРМ Отдел Кадров экономия времени очевидна. Достаточно найти личную карточку в архиве и вернуть ее в активное состояние. В целом, архив очень похож на раздел Личные карточки. Это сделано для максимального удобства работу с программой и реализации интуитивно-понятного интерфейса.

### Отчеты

В программе возможно формирование отчетов и вывод их на печать. Отчетом можно считать выведенную на печать отобранную личную карточку сотрудника. Функция печати также доступна в некоторых разделах формы личной карточки.

Кроме указанных отчетов существует специальный отчет по так называемой «13-зарплате». На некоторых предприятиях (в частности, на МЗР Орбита) если сотрудник проработал непрерывно более года, ему кроме штатной зарплаты выплачивается премия, как бы 13 по счету зарплата. В каждом последующем году 13-ая зарплата может увеличиваться на установленный на предприятии процент надбавки. Как правило это 5%.

Соответственно, иногда возникает необходимость получить список сотрудников в связи с назначением, изменением или лишением 13-ой зарплаты. Для этого в программе реализовано формирование отчета сотрудников, которые проработали непрерывно более года.

Следующим отчетом является отчет по всем военнообязанным работникам. Здесь важно заметить, что форма выводимая на печать должна максимально точно соответствовать стандартному бланку «форма №6».

Оба указанных отчета вынесены в отдельный пункт главного меню программы.

### Дополнительные функции

Кроме реализации основных требований к программе, были реализованы две дополнительные возможности - возможность вести «он-лайн» переговоры по сети между несколькими запущенными программами (на разных машинах) и ведение журнала отзывов и предложений по улучшению программы.

Возможность вести переговоры , отсылая и принимая сообщения с одного терминала на другой полезна при расположении отдела кадров в различных помещениях. Если помещения находятся на большом расстоянии друг от друга, то выяснения каких-либо производственных вопросов можно вести с помощью этого средства сообщения. При этом экономится рабочее время.

Так как данная программа достаточно большая разработка, то неминуемо возникновение отзывов и пожеланий относительно ее работы. У пользователей могут возникать нарекание или советы по ее дальнейшему развитию. Чтобы эта ценная информация не пропадала и в дальнейшем использовалась другими программистами-разработчиками в программе реализовано ведение журнала подобных отзывов. Также в журнал можно занести какой-либо вопрос по работе, требующий уточнения или разъяснения. Поскольку журнал доступен практически всем, то в нем можно разместить ответ на заданный вопрос. Например начальник отдела кадров может разместить в журнале официальный ответ на вопрос сотрудника отдела кадров.

Те сотрудники, которым права доступа позволяют работать с журналом могут просмотреть журнал, добавить в него свой вопрос либо ответить на заданный.

Журнал доступен по пункту меню «Пожелания и предложения» (если на него есть доступ по привилегии).

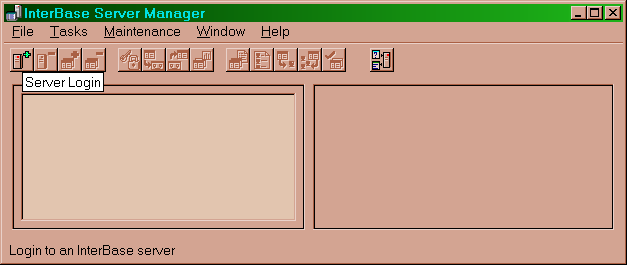
### Установка и настройка программы

Сразу стоит отметить, что для установки и ввода в эксплуатацию программы требуются хотя бы начальные знания системного администрирования. Как правило, на крупном предприятии есть специалист – системный администратор, поэтому непосредственных пользователей программы ее первичная настройка тревожить не должна. Однако, при установке на сетевой сервер, либо целиком на локальную машину все-таки надо выполнить ряд действий. Они выполняются только один раз, при установке, и при нормальной работе программно-аппаратного комплекса не требуют повторного выполнения.

Первым этапом следует установить SQL-сервер InterBase. Данная программа разработана и протестирована на сервере версии 5.1.1. Более младшие версии использовать нельзя, так как они содержали ошибки и неточности. На более старших версиях программа не проверялась, по причине их не существования на момент разработки. По неписаным законам совместимости, как правило, более старшая версия поддерживает все возможности предыдущей.

Указанный InterBase SQL сервер входит в комплект поставки Delphi и его установка достаточно тривиальна.

При установленном сервере необходимо создать в BDE-Administrator так называемые «алиасы» - псевдонимы баз данных. Также следует завести пользователя баз данных, которым с точки зрения сервера и является написанная программа АРМ Отдел Кадров. Для этого в менеджере SQL сервера надо завести пользователя с именем «MATHER» и установить его пароль «1». Приведем внешний вид менеджера.



Подключение к серверу

Управление безопасностью

Рис. 2.1 Менеджер SQL InterBase сервера

Сначала надо подключится к серверу. Для этого надо выбрать кнопку «Подключение к серверу», по нажатию на которую вызывается диалог выбора сервера. Выбрав установленный локальный или установленный на сетевом сервере SQL-сервер, произвести к нему подключение. Стоит помнить, что сразу после установки SQL сервера по умолчанию имя сисадмина SYSDBA, логин MASTERKEY. Затем выбрать пункт Управление безопасностью и добавить пользователя с именем «MATHER» и установить его пароль «1».

Внешний вид диалогов :

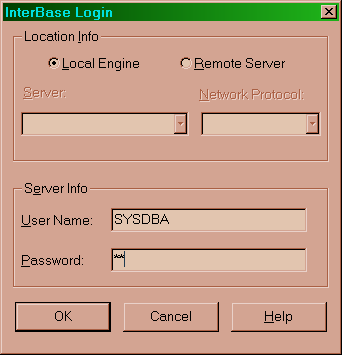


Рис 2.2 Диалог подключения к серверу

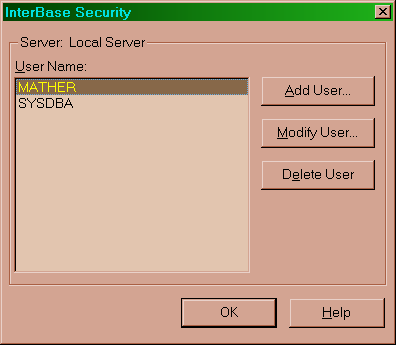


Рис 2.3 Диалог добавления нового пользователя

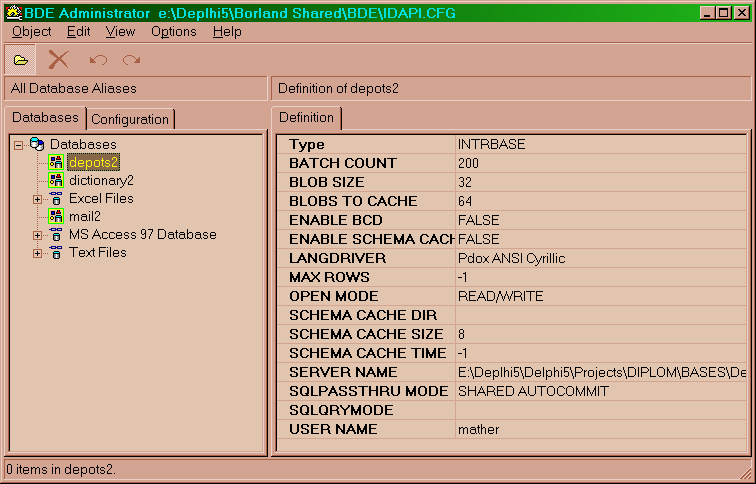
Далее надо завести "алиасы" в BDE администраторе на используемые базы данных. Сами файлы баз данных "Depots.gdb", "Dict.gdb", "Mail.gdb" надо разместить на сетевом сервере (или в любой удобной директории при локальной установке). Соответствия следующие :

"Depots.gdb" - aliase = depots2.

"Dict.gdb" - aliase = dictionary2.

"Mail.gdb"- aliase = mail2.

Рис 2.4. Внешний вид BDE-Administrator при создании алиасов.



Для созданных алиасов надо выбрать тип драйвера INTRBASE и язык LANGDRIVER = "Pdox ANSI Cyrillic". Данные действия иллюстрирует рисунок 2.4.

### Особенности реализации поставленной задачи

Первая задача, с которой пришлось столкнуться, состояла в разработке удобного интерфейса. Поскольку программа рассчитана на длительную работу оператора с ней, а также есть опыт неудачного интерфейса программы-прототипа, то этому вопросу было уделено немало внимания. Ключевым аспектом стал выбор цветовой палитры всей программы, поскольку как показала практика, слишком контрастная цветовая палитра ведет к сильному утомлению оператора и недовольству программой в целом.

В результате была выбрана наиболее оптимальная для считывания с экрана цветовая палитра, а именно белый или светло-серый шрифт на темно-зеленом фоне. Пользователи программы подтвердили, что именно при таких цветах глаза меньше всего устают и работа с программой не вызывает раздражения.

Следующей задачей стало хорошее структурирование обрабатываемой информации с целью ее разнесения по различным формам и диалогам. В программе-прототипе многие логически разные данные вводились в одной и той же форме, что приводило к путанице и не делало ее интерфейс интуитивно понятным. В АРМ-Отдел Кадров представление информации более наглядно и структурировано в целом.

Для повышения удобства работы были реализованы словари, назначение и принцип работы с которыми описаны выше.

С точки зрения программной реализации был решен ряд вопросов. В Delphi, как это не печально, набор стандартных функций для работы со строками и датами не покрывает все встречающиеся задачи. Поэтому при работе над дипломом были написаны различные функции и процедуры для проверки и обработки даты и строк. Они вынесены в отдельный модуль и могут быть применены при решении новых задач.

Также были закодированы алгоритмы для специфичной работы с периодами времени при работе с месяцами , кварталами и т.п.

Многие операторы привыкли работать с программой, минимально используя манипулятор "мышь" используя для повышения скорости набора текста и перемещения по полям ввода. Для этого были написаны специальные визуальные компоненты но основе простых полей ввода, но которые по нажатию клавиш "вверх", "вниз" передают фокус ввода на другое поле или компонент. Эти поля ввода также можно применить при решении других задач.

При реализации самой базы данных основной трудностью являлось продумывание структур таблиц. По сколько требовалось создание программы, создающей минимальную нагрузку на сеть, то это требовало достаточно ответственного подхода. Выяснилась необходимость создать ряд хранимых процедур, чтобы программа минимально нагружала сеть и соответствовала технологии клиент-сервер.

### Экранные формы

После запуска программы и ввода правильного пароля пользователь оказывается в основном окне программы :

Рис 2.5. Основное окно программы

Форма отображения списка личных карточек (активного и архива):

Рис 2.6 Список личных карточек (активных либо из архива).

Рис 2.7 Внешний вид учетной карточки-главный раздел

Рис 2.8 Внешний вид учетной карточки-Воинский учет.

Рис 2.9 Внешний вид учетной карточки-Назначения, перемещения

Рис 2.10 Комлексный запрос на поиск учетной карточки в архиве

Рис 2.11 Словарь - Виды образования

Рис 2.12 Словарь Специальностей в режиме добавления новой записи.

Рис 2.13 Системный журнал для просмотра совершенных действий.

## Заключение

В данном разделе приводиться общее описание созданной программы. Дана инструкция по ее вводу в эксплуатацию и работе с ней. Информация подкреплена графическим материалом - внешним видом диалоговых окон программы.

# Безопасность жизнедеятельности

## Введение

Пользователь ПЭВМ испытывает вредное действие работы ПЭВМ, поэтому рабочие места пользователей должны отвечать безопасным и безвредным условием труда.

В связи с этим предполагается разработать комплекс мер, обеспечивающих безопасные и безвредные условия труда и рассмотреть экологические вопросы

## Анализ опасных и вредных факторов, возникающих при работе с компьютером

При разработке программного продукта на разработчика работающего на ПЭВМ постоянно или периодически действуют следующие опасные и вредные факторы [3.1., 3.2.]:

1. Загрязнение воздуха вредными веществами, пылью, микроорганизмами и положительными аэронами.

2.Несоответствие нормам параметров микроклимата.

3.Возникновение на экране монитора статистических зарядов, заставляющих частички пыли двигаться к ближайшему заземлённому предмету, часто им оказывается лицо разработчика.

4.Повышенный уровень шума на рабочем месте.

5.Повышенный уровень статистического электричества при неправильно спроектированной рабочей зоне.

6.Опасный уровень напряжения в электрической цепи, замыкание которой может произойти через тело человека.

7.Широкий спектр излучения от дисплея, который включает рентгеновскую, ультрафиолетовую и инфракрасную области, а так же широкий диапазон электромагнитных излучений других частот.

8.Повышенный уровень электромагнитных излучений.

9.Повышенный уровень ионизирующих излучений ( мягкое рентгеновское, гамма - излучение).

10.Отсутствие или недостаток естественного света.

11.Недостаточная освещенность рабочей зоны.

12.Повышенная яркость света.

13.Пониженная контрастность.

14.Прямая и обратная блесткость.

15.Повышенная пульсация светового потока (мерцание изображения).

16.Длительное пребывание в одном и том же положении, и повторение одних и тех же движений приводит к синдрому длительных статических нагрузок (СДСН).

17.Нерациональная организация рабочего места.

18.Несоответствие эргономических характеристик оборудования нормируемым величинам.

19.Умственное перенапряжение, которое обусловлено характером решаемых задач приводит к синдрому длительных психологическим нагрузкам (сдпн).

20.Большой объем перерабатываемой информации приводит к значительным нагрузкам на органы зрения.

21.Монотонность труда.

22.Нервно-психические нагрузки.

23.Нервно-эмоциональные стрессовые нагрузки.

24.Опасность возникновения пожара.

Остановимся подробнее на недостаточной освещенности рабочей зоны помещения, где установлены ПЭВМ, а также на влиянии повышенной яркости света, пониженной контрастности, прямой и обратной блёсткости и повышенной пульсации светового потока.

При работена ПЭВМ органы зрения пользователя выдерживают боль­шую нагрузку с одновременным постоянным напряженным характером труда что приводит к нарушению функционального состояния зри­тельного анализатора и центральной нервной системы.

Нарушение функционального состояния зрительного анализатора проявляется в снижении остроты зрения, устойчивости ясного видения, аккомодации, электрической чувствительности и лабильности.

Причинами нарушения функционального состояния зрительного анализатора являются постоянная переадаптация органов зрения в условиях наличия в поле зрения объекта различения и фона различной яркости; недостаточной четкостью и контрастностью изображения на экране; строчностью воспринимаемой информации; постоянными яркостными мельканиями; наличием ярких пятен на клавиатуре и экране за счет отражения светового потока, большой разницей между яркостью рабочей поверхности я яркостью окружающих предметов, нали­чием равноудаленных предметов, невысоким качеством исходной инфор­мации на бумаге, неравномерной и недостаточной освещенностью на рабочем месте.

Наряду с перечисленными общепринятыми особенностями работы пользователя на рабочем месте ПЭВМ существуют особенности восприятия информации с экрана монитора.

Особенностью восприятия информации с экрана монитора органами зрения пользователя ПЭВМ являются:

* экран монитора является источником света, на который в процессе работы непосредственно обращены органы зрения пользователя, что вводит оператора в другое психофизиологическое состояние;
* привязанность внимания пользователя к экрану монитора является причиной длительности неподвижностиглазных и внутриглазных мышц, что приводит к их ослаблению;
* длительная и повышенная сосредоточенность органов зрения приводит к большим нагрузкам а, следовательно, к утомлению органов зрения, способствует возникновению близорукости, головной боли и раздраженности, нервного напряжения и стресса;
* длительная привязанность внимания пользователя к экрану монитора создает дискомфортное восприятие информации, в отличие от чтения обычной печатной информации;
* экран монитора является источником падающего светового потока на органы зрения пользователя, в отличие от обычной печатной информации, которая считывается за счет отраженного светового потока;
* информацияна экране монитора периодически обновляется в процессе сканирования электронного луча по поверхности экрана и при низкой частоте происходит мерцание изображения, в отличиеот неизменной информации на бумаге.

## Мероприятия по предотвращению и уменьшению влияния вредных факторов

### Нормирование искусственного и естественного освещения.

Для снижения нагрузки на органы зрения пользователя при работе на ПЭВМ необходимо соблюдать следующие условия зрительной работы. При работе на ПЭВМ пользователь выполняет работу высокой точности, при минимальном размере объекта различения 0.3-0.5 мм (толщина символа на экране), разряда работы III, подразряда работы Г (экран - фон светлый символ - объект различения темным или наоборот).

Естественное боковое освещение должно составлять 2%, комбинированное искусственное освещение 400 лк при общем освещении 200 лк [3.3.]

### Основные требования к искусственному освещению в производственном помещении.

К системам производственного освещения предъявляются следующие основные требования: [2.4., 2.5]

* соответствие уровня освещённости рабочих мест характеру выполняемой работы
* достаточно равномерное распределение яркости на рабочих поверхностях и в окружающем пространстве
* отсутствие резких теней, прямой и отражённой блёскости (блёскость – повышенная яркость светящихся поверхностей, вызывающая ослеплённость);
* оптимальная направленность излучаемого осветительными приборами светового потока.
* Искусственное освещение в помещении и на рабочем месте создаёт хорошую видимость информации, машинописного и рукописного текста, при этом должна быть исключена отражённая блёскость.

В связи с этим предусматриваются мероприятия по ограничению слепящего воздействия оконных проёмов и прямое попадание солнечных лучей, а так же исключение на рабочих поверхностях ярких и тёмных пятен. Это достигается за счёт соответствующей ориентации оконных проёмов и рационального размещения рабочих мест.

Площадь оконных проёмов должна составлять не менее 25% площади пола. В помещении рекомендуется комбинированная система освещения с использованием люминесцентных ламп. Для проектирования местного освещения рекомендуются люминесцентные лампы, светильники которых установлены на столе или его вертикальной панели.

Светильники местного освещения должны иметь приспособления для ориентации в разных направлениях, устройствах для регулирования яркости и защитные решётки от ослепления и отражённого света.

### Расчёт искусственного освещения

Имеется помещение инженера-разработчика размером:

длина 5 м;

ширина 4 м;

высота 3 м.

Потолок, пол и стены окрашены краской. Метод светового потока сводится к определению количества светильников по следующей формуле [3.5] :

N = (\*Sп\*К\*Z) / (F\* \*n)

где Енорм - нормируемая минимальная освещённость на рабочем месте, лк;

Енорм= 400лк;

Sn - площадь производственного помещения, м2; S=20 м2;

К - коэффициент запаса светового потока, зависящий от степени загрязнения ламп, К=1.4,

Z – коэффициент минимальной освещенности, для люминесцентных ламп = z = 1.1

F – световой поток лампы, лм;

коэффициент использования светового потока ламп;

n – число ламп в светильнике, n = 2.4;

коэффициент затенения, = 0.9

Индекс помещения определяется по формуле:

А и В - длина и ширина помещения, м;

Нр - высота подвеса светильника над рабочей поверхностью, м.

После подстановки данных, находим индекс помещения:

i = (5\*4) / (2\*(5+4)) = 1.11

Коэффициенты отражения потолка и пола принимаем 0.75 и 0.50 соответственно. В зависимости от индекса помещения и коэффициентов отражения потолка и пола находим коэффициент использования светового потока по таблице [2.5]

Выбираем тип люминесцентных ламп низкого давления:

Лампа ЛТБ-20, световой поток 975 лм;

Лампа ЛТБ-30, световой поток 1720 лм;

Лампа ЛТБ-40, световой поток 3000 лм.

Подставив все значения, найдем количество светильников:

N = (400\*20\* 1.4\*1.1)/(975\*0.54\*2.4\*0.9)=10.8 = 11 шт;

N = (400\*20\* 1.4\*1.1)/(1720\*0.54\*2,4\*0.9)=6.1 = 6 шт;

N = (400\*20\*1.4\*1.1)/(3000\*0.54\*2.4\*0.9)=3.52 = 4 шт.

Из трех вариантов выбираем наиболее экономичный.

Для определения оптимального варианта надо рассчитать:

Руд =N\*F/Sn

1. Руд = 11\*975 / 20 = 536.25

2. Руд = 6\*1720 / 20 = 516

3. Руд = 4\*3000 / 20 = 600

Следовательно, наиболее экономичным будет вариант 2:

ЛТБ-30, и поэтому конструктивно выбираем его.

### Рациональная планировка рабочих мест.

Для создания равномерной освещённости рабочих мест при общем освещении светильники с люминесцентными лампами встраиваются непосредственно в потолок помещения и располагается в равномерно-прямоугольном порядке. Наиболее желательное расположение светильников в непрерывный сплошной ряд вдоль длинной стороны помещения. Коэффициент наивыгоднейшего расположения светильников определяется по формуле [2.5]:

Lm = Lc / Hp ,

где Lm - коэффициент наивыгоднейшего расположения светильников, Lm =1.3;

Lс - расстояние между центрами светильников, м. Отсюда, Lс = l.3\*2 = 2.6м.

Число рядов светильников определяем по формуле:

m=B/Lс, m=4/2.6=1.53=2.

Число светильников в ряду определяем по формуле:

M=N/m, М=6/2=3шт.

Суммарная длина светильников в ряду -1св М, учитывая, что

1св=[1л+(0.05-0.1)],

где 1св - длина светильника, м ;

1л - длина лампы, м.

1св = 0.909+0.9=1 м

Отсюда расстояние между светильниками в ряду определим из следующего соотношения:

A-1св\*M

К = , K = (5-1\*3) / (3+1) = 0.5 м

M+1

Схема расположения светильников приведена на рис. 3.1.



рис. 3.1. Схема расположения светильников в помещении.

1 – оконный проем; 2- светильник; 3 – рабочий стол;

## Утилизация и переработка ртути в люминесцентных лампах

Определив количество ламп в помещении и приняв срок службы одной лампы в среднем полгода, рассмотрим вопросы утилизации и переработки ртути в люминесцентных лампах.

Только в приборостроительной области количество используемых люминесцентных ламписчисляется миллионами и через 1.5-2 года выбрасывается на свалки. [3.6].

В связи с этим большое практическое значение приобретает разработка и внедрение технологии извлечений дорогостоящих материалов из люминесцентных ламп после окончания срока их эксплуатации, в частности технология извлечения ртути.

Разработка технологии извлечения ртути является составной частью создания ресурсосберегающей технологии и природоохранительной системы.

Ртуть (Hg) имеет атомный вес 200,59. Она мало распространена в природе: ее содержание в земной коре составляет всего 0,000005 вес.%. Изредка ртуть встречается в самородном виде, вкрапленная в горные породы, но главным образом она находится в природе в виде сульфида ртути HgS , или киновари. Ртуть - единственный металл, жидкий при обыкновенной температуре, ее плотность составляет 13,546г/см3.

Ртуть является весьма дорогостоящим элементом. Добыча ее отличается трудоемкой технологией, которая приводит к нарушению земель по форме рельефа, т.е. к нарушению экологического равновесия.

Кроме того, не утилизированные люминесцентные лампы могут приводить к попаданию паров ртути в атмосферный воздух, через почву и воду,

Ртуть относится к веществам первого класса опасности, а ее величина ПДК- 0,0003 мг/м3 согласно СН 245-71 т.е. ртуть является чрезвычайно опасным веществом, оказывающее пагубное влияние на окружающую среду и живой мир,

Каждая лампа содержит 60,.. 120мг ртути. Примерно 100г ртути можно получить из 1000 ламп. Испарение такого количества ртути из разбитых ламп приводит к загрязнению 10 млн.м3 воздуха по ПДК.

Переработка использованных люминесцентных ламп исключает это воздействие.

Отделение по извлечению ртути из люминесцентных ламп может располагаться на территории предприятия по изготовлению ламп или на предприятии любой отрасли, где эксплуатируется большое количество люминесцентных ламп

В основу технологии извлечения ртути из люминесцентных ламп лежит способ демеркуризации.(рис. 3.2)

### Дробление ламп

Погрузка боя в контейнер

Демеркуризация боя ламп в ванне с раствором в течении 1.5 часа

Установка контейнера на лотке для стока раствора

### Сбор раствора в

приемный бак

Транспортировка боя и арматуры в контейнерах к линии сортировки

Перекачка раствора через фильтр с сульфоуглем типа КУ-2

Сжигание фильтра с сульфоуглем и образование металлической ртути

Сбор раствора в хозяйственно-фекальную канализацию

Сбор раствора для повторного приготовления

Рис 3.2 Схема демеркуризации люминесцентных ламп

Операция дробления ламп осуществляется в барабане, при вращении лопастей которого происходит измельчение стекла ламп.

Операция погрузки в контейнер осуществляется перемещением боя стекла ламп и арматуры по желобу.

Операция демеркуризации боя стекла ламп производят помещением контейнера в ванну с демеркуризационным раствором, где его выдерживают в течение 1,5 часов.

В табл 3.1 приведены типы, химический состав и краткая характеристика демеркуризационных растворов.

Таблица 3.1.

### Химический состав и удельный расход демеркуризационных растворов

|  |  |  |
| --- | --- | --- |
| Тип раствора | Состав и удельный расход на одну лампу демеркуризационного раствора | Состав и удельный расход на одну лампу демеркуризационного раствора |
| Раствор №1, Температура Раствора 280° | Перманганат калия Ктп04-0.00025г/л Соляная кислота НС1 -0,000125г/л Техническая вода-0,0375г/л | Ионы в перечете на металлическую ртуть:  KMn04-0.5\* 10г/л  НС1-0.25\* 10г/л |
| Раствор №2, Температура Раствора 28° | Хлорное железо Fed \* 6Н20 - 0.0025г/л Карбонат кальция СаСОз-0,0015г/л Техническая вода -0,0375 г/л | Ионы в пересчете на  металлическую ртуть:  Fed \* 6Н2О - 0.25 \* 10г/л,  СаСОз -3.75\* 10г/л |

Операция установки контейнера на лотке преследует цель стока демеркуризационного раствора.

Операция сбора демеркуризационного раствора производится в приемный бак емкостью 1,6м3.

Операция перекачки отработанного раствора производится насосом в ионообменный фильтр с сульфоуглем типа ККУ-2, предварительно прошедшем регенерацию раствором СаСОз.

Операция выделения металлической ртути происходит за счет сжигания фильтра с сульфоуглем, которое производится один раз в два года.

Наряду с основными операциями имеются дополнительные. Отработанный демеркуризационный раствор может быть направлен в бак емкостью 1,6м3 для повторного приготовления демеркуризационного раствора или в системы хозяйственно-фекальной канализации предприятия.

Массу подвергают обработке (отделению металлической арматуры от боя стекла).

Бой стекла ламп направляют для переработки на предприятие по производству ламп или на предприятие стеклянных изделий.

Металлическую арматуру направляют для переплава на машиностроительные и металлургические предприятия.

Общее количество ртути, которое может быть извлечено при демеркуризации люминесцентных ламп определяют по формуле:

М = m \* N,

где М - общее количество ртути, которое может быть извлечено из люминесцентных ламп.

m - количество ртути, которое может быть извлечено из одной лампы, г;

Количество ртути в одной люминесцентной лампе - 0,05-0,12г.

После подстановки известных значений получаем :

M = 0.12 \* 12 = 1.44 г

С учетом всех производственных помещений это уже значительная цифра и путь к созданию природоохранной системы.

## 

## Заключение

В разделе «Безопасность жизнедеятельности» проведен анализ вредных факторов, оказывающих влияние на органы зрения пользователя ПЭВМ.

Сформированы общие требования к помещению и произведен расчет искусственного освещения.

Проведена экологическая оценка люминесцентных ламп, которые используются в производственном помещении.

# Приложения

### Листинг основного файла-проекта

program deports;

uses

Forms,

sysutils,

mainform in 'mainform.pas' {form1},

PasswordsDB in '..\library\PasswordsDB.pas' {Passwords\_: TDataModule},

PasswordChange in '..\library\PasswordChange.pas' {PasswordChange\_},

Login in '..\library\Login.pas' {Login\_},

NEUser in '..\library\NEUser.pas' {NEUser\_},

Variables in '..\library\Variables.pas',

utils in '..\library\utils.pas',

keygenDB in '..\library\keygenDB.pas' {keygen\_: TDataModule},

JornalDB in '..\library\JornalDB.pas' {Jornal\_: TDataModule},

ViewJornal in '..\library\ViewJornal.pas' {ViewJornal\_},

BureausDB in '..\library\BureausDB.pas' {Bureaus\_: TDataModule},

viewBureaus in '..\library\viewBureaus.pas' {ViewBureaus\_},

NewBureau in '..\library\NewBureau.pas' {NewBureau\_},

MainBD in '..\library\MainBD.pas' {MainDB: TDataModule},

mailer in '..\library\mailer.pas' {Mailer\_},

UsersMonitor in '..\library\UsersMonitor.pas' {UsersMonitor\_},

educationDB in '..\library\educationDB.pas' {edu\_: TDataModule},

ViewEdu in '..\library\viewedu.pas' {ViewEdu\_},

NewEdu in '..\library\Newedu.pas' {Newedu\_},

InstitutesDB in '..\library\InstitutesDB.pas' {Institutes\_: TDataModule},

ViewInstitutes in '..\library\viewInstitutes.pas' {ViewInstitutes\_},

NewInstitute in '..\library\NewInstitute.pas' {NewInstitute\_},

DipSpcDB in '..\library\DipSpcDB.pas' {DipSpc\_: TDataModule},

ViewDipSpc in '..\library\viewDipSpc.pas' {ViewDipSpc\_},

NewDipSpc in '..\library\NewDipSpc.pas' {NewDipSpc\_},

qualifDB in '..\library\QualifDB.pas' {qualif\_: TDataModule},

Viewqualif in '..\library\viewQualif.pas' {Viewqualif\_},

Newqualif in '..\library\NewQualif.pas' {Newqualif\_},

WkSpcDB in '..\library\wkSpcDB.pas' {WkSpc\_: TDataModule},

ViewWkSpc in '..\library\viewwkSpc.pas' {ViewWkSpc\_},

NewWkSpc in '..\library\NewwkSpc.pas' {NewWkSpc\_},

ITRSpcDB in '..\library\ITRSpcDB.pas' {ITRSpc\_: TDataModule},

ViewITRSpc in '..\library\viewITRSpc.pas' {ViewITRSpc\_},

NewITRSpc in '..\library\NewITRSpc.pas' {NewITRSpc\_},

AroundsDB in '..\library\AroundsDB.pas' {Arounds\_: TDataModule},

ViewArounds in '..\library\viewArounds.pas' {ViewArounds\_},

NewAround in '..\library\NewAround.pas' {NewAround\_},

NewStreet in '..\library\Newstreet.pas' {NewStreet\_},

WarBursDB in '..\library\WarBursDB.pas' {WarBurs\_: TDataModule},

ViewWarBurs in '..\library\viewWarBurs.pas' {ViewWarBurs\_},

NewWarBur in '..\library\NewWarBur.pas' {NewWarBur\_},

WarGrpsDB in '..\library\WarGrpsDB.pas' {WarGrps\_: TDataModule},

ViewWarGrps in '..\library\viewWarGrps.pas' {ViewWarGrps\_},

NewWarGrp in '..\library\newWarGrp.pas' {NewWarGrp\_},

ContsDB in '..\library\ContsDB.pas' {Conts\_: TDataModule},

ViewConts in '..\library\viewConts.pas' {ViewConts\_},

NewCont in '..\library\NewCont.pas' {NewCont\_},

NewWarName in '..\library\NewWarName.pas' {NewWarName\_},

NeedDB in '..\library\NeedDB.pas' {Needs\_: TDataModule},

ViewQuestions in '..\library\viewQuestions.pas' {ViewQuestions\_},

NewQuestion in '..\library\NewQuestion.pas' {NewQuestion\_},

Answer in '..\library\Answer.pas' {Answer\_},

NationsDB in '..\library\NationsDB.pas' {Nations\_: TDataModule},

ViewNations in '..\library\viewNations.pas' {ViewNations\_},

NewNation in '..\library\NewNation.pas' {NewNation\_},

CountrysDB in '..\library\CountrysDB.pas' {Countrys\_: TDataModule},

viewCountrys in '..\library\viewCountrys.pas' {ViewCountrys\_},

NewCountry in '..\library\NewCountry.pas' {NewCountry\_},

FamalysDB in '..\library\FamalysDB.pas' {Famalys\_: TDataModule},

ViewFamalys in '..\library\viewFamalys.pas' {ViewFamalys\_},

NewFamaly in '..\library\NewFamaly.pas' {NewFamaly\_},

ViewExcesses in '..\library\viewExcesses.pas' {ViewExcesses\_},

ExcessesDB in '..\library\ExcessesDB.pas' {Excesses\_: TDataModule},

NewExcess in '..\library\NewExcess.pas' {NewExcess\_},

MeedsDB in '..\library\MeedsDB.pas' {Meeds\_: TDataModule},

ViewMeeds in '..\library\viewMeeds.pas' {ViewMeeds\_},

NewMeed in '..\library\NewMeed.pas' {NewMeed\_},

emeritusDB in '..\library\emeritusDB.pas' {emeritus\_: TDataModule},

viewemeritus in '..\library\viewemeritus.pas' {Viewemeritus\_},

Newemeritus in '..\library\Newemeritus.pas' {Newemeritus\_},

CardsDB in 'CardsDB.pas' {Cards\_: TDataModule},

LoadLed in '..\library\LoadLed.pas' {LoadLed\_},

ViewCards in 'viewcards.pas' {ViewCards\_},

ViewCard in 'viewcard.pas' {ViewCard\_},

NewChild in 'Newchild.pas' {NewChild\_},

NewCardsEdu in 'newcardsedu.pas' {NewCardsEdu\_},

NewWorkMove in 'Newworkmove.pas' {NewWorkMove\_},

NewcommonWork in 'Newcommonwork.pas' {NewcommonWork\_},

lockScreen\_ in 'lockScreen\_.pas' {lockScreen\_\_},

Newleaf in 'Newleaf.pas' {Newleaf\_},

NewPunishment in 'Newpunishment.pas' {NewPunishment\_},

DisPunishment in 'Dispunishment.pas' {DisPunishment\_},

NewOutCountry in 'NewOutCountry.pas' {NewOutCountry\_},

Newplantmeed in 'Newplantmeed.pas' {Newplantmeed\_},

Newcardmeed in 'Newcardmeed.pas' {Newcardmeed\_},

search in 'search.pas' {Search\_},

Newcardemeritus in 'Newcardemeritus.pas' {Newcardemeritus\_},

MainTable in 'MainTable.pas' {MainTable\_: TQuickRep},

PRNTitle in 'PRNTitle.pas' {PRNTitle\_},

cards\_Main in 'cards\_Main.pas' {cards\_main\_: TQuickRep},

MainPFTable in 'MainPFTable.pas' {MainPFTable\_: TQuickRep},

viewcdpf in 'viewcdpf.pas' {viewcdpf\_},

viewcdedu in 'viewcdedu.pas' {viewcdedu\_},

Statedu in 'Statedu.pas' {StatEdu\_},

MaineduTable in 'MaineduTable.pas' {MaineduTable\_: TQuickRep},

Statqual in 'Statqual.pas' {Statqual\_},

viewcdqual in 'viewcdqual.pas' {viewcdqual\_},

MainqualTable in 'MainqualTable.pas' {MainqualTable\_: TQuickRep},

StatAround in 'Stataround.pas' {StatAround\_},

Statbur in 'Statbur.pas' {Statbur\_},

s13\_thpayment in 's13\_thpayment.pas' {s13\_thpayment\_: TQuickRep},

PRNSTG13 in 'PRNSTG13.pas' {PRNSTG13\_},

viewWarNames in '..\library\viewWarNames.pas' {ViewWarNames\_},

megastat in 'megastat.pas' {megaStat\_},

MaininvalTable in 'MaininvalTable.pas' {MaininvalTable\_: TQuickRep},

viewcdinval in 'viewcdinval.pas' {viewcdinval\_},

viewarchive in 'viewarchive.pas' {viewarchive\_},

wait in 'wait.pas' {Wait\_},

setingsdb in '..\library\setingsdb.pas' {Setings\_: TDataModule},

s14\_thpayment in 's14\_thpayment.pas' {s14\_thpayment\_: TQuickRep},

Warreport in 'Warreport.pas' {WarReport\_: TQuickRep},

MedTable in 'MedTable.pas' {MedTable\_: TQuickRep},

helper in '..\library\helper.pas' {Form2};

{$R \*.RES}

begin

Application.Initialize;

Application.CreateForm(Tform1, form1);

Application.Run;

end.

### Листинг созданного VCL-компонента.

unit DBSearchEdit;

interface

uses

Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, DBTables,

DB,stdctrls;

type

TDBSearchEdit = class(TEdit)

procedure onchange\_ (sender : TObject);

procedure onkeypress\_ (Sender: TObject; var Key: Char);

private

needsearch\_ : boolean;

table1\_ : Ttable;

keyFieldName\_,textFieldName\_ : TField;

indexname\_ : string;

onchange2 : tnotifyevent;

srch,ch : integer;

onkeypress2 : TKeyPressEvent;

{ Private declarations }

protected

{ Protected declarations }

public

constructor Loaded;

constructor Create(aOwner: TComponent); override;

destructor destroy;

{ Public declarations }

published

property needsearch : boolean read needsearch\_ write needsearch\_;

property Dataset : Ttable read table1\_ write table1\_;

property KeyFieldName : TField read keyfieldname\_ write keyfieldname\_;

property TextFieldName : TField read textfieldname\_ write textfieldname\_;

property indexname : string read indexname\_ write indexname\_;

property newonkeypress : TKeyPressEvent read onkeypress2 write onkeypress2;

property newonchange : TNotifyEvent read onchange2 write onchange2;

{ Published declarations }

end;

procedure Register;

implementation

function compress(s : shortstring):shortstring;

var

s2 : shortstring;

x : integer;

begin

s2 :='';

for x:=1 to length(s) do if s[x]<>' ' then s2:=s2+s[x];

s2:=s2+#0;

s2:=ansistrupper(@s2[1]);

compress:=s2;

end;

constructor TDBSearchEdit.Loaded;

begin

inherited loaded;

end;

procedure TDBSearchEdit.onchange\_(sender : tobject);

var

s1,s : shortstring;

poz : integer;

begin

if srch=0 then exit;

srch:=0;

if ch=1 then exit;

ch:=1;

if assigned(onchange2) then onchange2(sender);

if not TDBSearchEdit(sender).needsearch\_ then exit;

try

s1:=table1\_.Indexname;

table1\_.indexname:=indexname\_;

s:=compress(TDBSearchEdit(sender).text);

table1\_.findnearest([s]);

if pos(s,table1\_.fieldbyname(keyfieldname\_.fieldname).asstring)=1 then

begin

poz:=length(TDBSearchEdit(sender).text);

TDBSearchEdit(sender).text:=Table1\_.fieldbyname(textfieldname\_.fieldname).asstring;

TDBSearchEdit(sender).perform(em\_setsel,poz,length(TDBSearchEdit(sender).text));

end

else begin beep; beep; beep; beep; beep; beep; end;

finally

try

ch:=0;

table1\_.indexname:=s1

except

end;

end;

end;

procedure TDBSearchEdit.onkeypress\_(Sender: TObject; var Key: Char);

begin

if assigned(onkeypress2) then onkeypress2(sender,key);

if key<>#8 then srch:=1;

end;

constructor TDBSearchEdit.Create(aOwner: TComponent);

begin

inherited create(aOwner);

needsearch:=true;

onchange:=onchange\_;

onkeypress:=onkeypress\_;

end;

destructor TDBSearchEdit.destroy;

begin

inherited destroy;

end;

procedure Register;

begin

RegisterComponents('Diplom',[TDBSearchEdit]);

end;

end.

### Модуль блокировки программы от несанкционированного доступа

unit lockScreen\_;

interface

uses

Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,

StdCtrls, ExtCtrls,variables;

type

TlockScreen\_\_ = class(TForm)

Panel1: TPanel;

Label1: TLabel;

Edit1: TEdit;

procedure Edit1KeyPress(Sender: TObject; var Key: Char);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure FormCreate(Sender: TObject);

procedure FormResize(Sender: TObject);

procedure WNDPROC(var m : tmessage); override;

private

canclose : integer;

{ Private declarations }

public

{ Public declarations }

end;

var

lockScreen\_\_: TlockScreen\_\_;

implementation

{$R \*.DFM}

procedure TlockScreen\_\_.WNDPROC(var m : tmessage);

begin

inherited wndproc(m);

end;

procedure TlockScreen\_\_.Edit1KeyPress(Sender: TObject; var Key: Char);

begin

if key=#13 then

begin

if edit1.text=password\_\_ then

begin

canclose:=1;

close;

end else edit1.text:='';

end;

end;

procedure TlockScreen\_\_.FormClose(Sender: TObject;

var Action: TCloseAction);

begin

if canclose=1 then action:=cafree else action:=canone;

end;

procedure TlockScreen\_\_.FormCreate(Sender: TObject);

begin

canclose:=0;

end;

procedure TlockScreen\_\_.FormResize(Sender: TObject);

begin

panel1.left:=(clientwidth-panel1.width) div 2;

panel1.top:=(clientheight-panel1.height) div 2;

end;

end.

# Список использованных источников

* 1. Дж.Ульман, "Основы систем баз данных", М.:Финансы и статистика,1983г.
  2. Дейт К., "Введение в системы баз данных", М.: Hаука, 1980 г.
  3. Корячко В.П., Курейчик В.М., Hоренков И.П. "Теоретические основы САПР", М.: Энергоатомиздат, 1987г.
  4. Когаловский М.Р., "Технология баз данных на персональных ЭВМ",

М.:Финансы и статистика, 1992 г.

* 1. А.H.Hаумов, А.М.Вендров и др., "Системы управления базами данных и знаний", М.:Финансы и статистика, 1991г.
  2. Брябрин В.М., "Программное обеспечение персональных ЭВМ", М.:Hаука, 1989 г.
  3. Аппак М.А., "Автоматизированные рабочие места на основе персональных ЭВМ", М.:'Радио и связь', 1989 г.
  4. Крайзмер Л.П., Кулик Б.А., "Персональный компьютер на вашем рабочем месте", 'Лениздат', 1991 г.
  5. Шумаков П. В. “Delphi 3.0 и создание баз данных”. Москва 1997г.
  6. Дж. Мартин., "Организация баз данных в вычислительных системах" М: Мир 1978г.
  7. С.М.Диго "Проектирование и использования баз данных". Москва: Финансы и статистика 1995.
  8. A.M.Епанешников., "Программирование в среде Delphi 2.0"
  9. Д.Веттинг Nowell NetWare для пользователя М.:Радио и связь, 1997
  10. С.И.Казаков Основы сетевых технологий М.:Радио и связь, 1999
  11. Nowell NetWare 4.02 for Lan Managers” Nowell Corp. Документация к продукту.
  12. Кириллов В.В. Структуризованный язык запросов (SQL). – СПб.: ИТМО, 1994. – 80 с.
  13. Б.Г. Голованов “ Введение в программирование в сетях Nowell NetWare

С-П.: Питер, 2000.

3.1 ГОСТ 12.0.003-74. ССБТ Опасные и вредные факторы.Классификация.

3.2 Шумилин В.К., Гетия И.Г. Охрана труда при работе на ПЭВМ

и ЭВМ. Учебное пособие (часть 1).- М.: МИП, 1994г.

3.3 СНиП23-05-95. Естественное и искусственное освещение.

Нормы проектирования.

3.4 СанПиН 2.2.2 542-96 Санитарные правила и нормы. Гигиенические

требования к видеодисплейным терминалам, персональным

электронно-вычислительным машинам и организации работ.

3.5.И.Г.Гетия, И.Н.Леонтьева, Е.Н.Кулемина. Учебное пособие. Проектирование вентиляции и кондиционирование воздуха, искусственного и естественного освещения в помещении ВЦ (спец.01.02; 21.01; 21.03;21.06; 22.01; 22.03; 23.03)-М:МГАПИ,1995.

3.6 И.Г. Гетия, В.К. Шумилин, И.Н. Леонтьева и др. Экология компьютерной техники: - М.:МГАПИ,1996.