Министерство образования и науки Российской Федерации

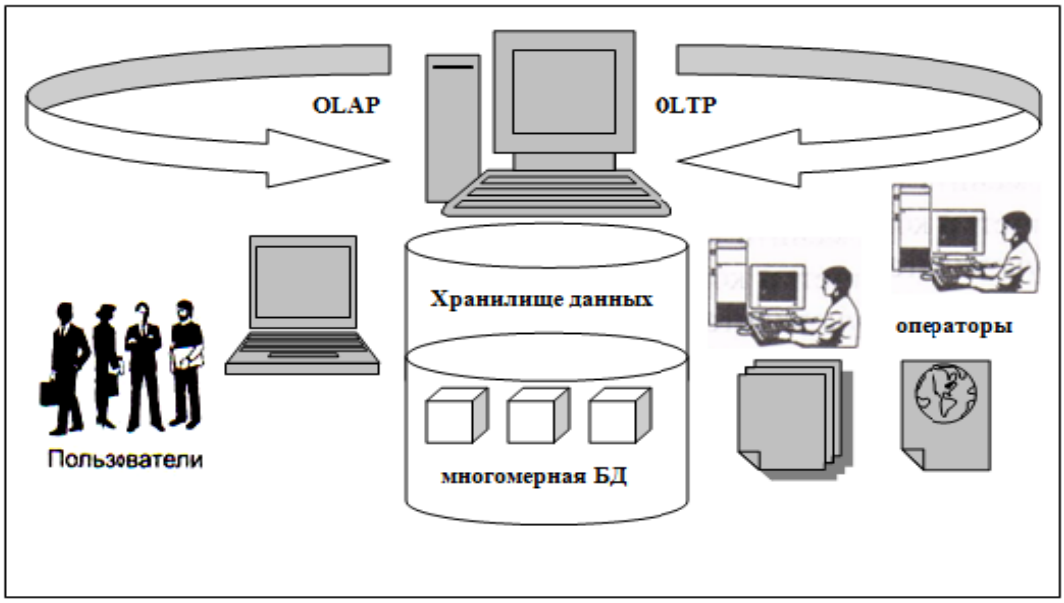
Федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
«Оренбургский государственный университет»

Кафедра программного обеспечения вычислительной
техники и автоматизированных систем

С.А. Щелоков, Е.Н. Чернопрудова

ПРОЕКТИРОВАНИЕ РАСПРЕДЕЛЕННЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

Курс лекций



Рекомендовано Ученым советом федерального государственного бюджетного образовательного учреждения высшего профессионального образования «Оренбургский государственный университет» в качестве учебного пособия для студентов, обучающихся по программам высшего профессионального образования по направлению подготовки 230100.68 Информатика и вычислительная техника.

Оренбург 2012

УДК 004.75 (075.8)
ББК 32.973.202я73
Щ 46

Рецензент – доктор технических наук, профессор В.И. Чепасов

Щелоков, С.А.

Щ46 Проектирование распределенных информационных систем: курс лекций по дисциплине «Проектирование распределенных информационных систем» / С.А. Щелоков, Е.Н. Чернопрудова; Оренбургский гос. ун-т. – Оренбург: ОГУ, 2012. – 195 с.

Курс лекций по дисциплине «Проектирование распределенных информационных систем» предназначен для учебно-методического обеспечения магистерской программы «Информационное и программное обеспечение автоматизированных систем». Дисциплина относится к дисциплинам вариативной части учебного профессионального цикла по направлению подготовки магистров очной формы обучения 230100.68 - «Информатика и вычислительная техника».

В курсе лекций изложены общие теоретические основы баз данных и информационных систем, концептуальные основы распределенной обработки информации, методические основы и примеры практического проектирования распределенных информационных систем с использованием современных программных инструментальных средств.

УДК 004.75 (075.8)
ББК 32.973.202я73

Содержание

Введение……. 4
Обозначение предметной области «распределенные информационные системы» и проблемных вопросов изучения дисциплины……. 5
1 Лекция 1 Назначение и основные компоненты информационной системы……..... 21
2Лекция 2 Концепция баз данных и автоматизированных информационных систем…….. 53
3 Лекция 3 Каноническое и типовое проектирование РИС……. 77
4 Лекция 4 Понятие жизненного цикла ПО РИС…… 98
5 Лекция 5 Распределенная обработка данных……. 113
6 Лекция 6 Общая характеристика распределенных баз данных……. 132
7 Лекция 7 Пример разработки проекта РИС предметной области……. 160
Список использованных источников……... 194

Введение

Целью изучения дисциплины является овладение современными методами и средствами технологии исследования и проектирования, разработки и использования проблемно – ориентированных распределенных информационных систем (РИС). Для достижения поставленной цели предусматривается решение следующих основных задач: изучение распределенной обработки информации в автоматизированных информационных системах, архитектуры РИС, технологической базы РИС, распределенных информационных ресурсов и сетей, распределенных баз данных, принципов и технологий управления обменом информацией в РИС, методов и средств доступа к удаленным информационным ресурсам.

Дисциплина относится к дисциплинам вариативной части учебного профессионального цикла.

Выпускник должен обладать следующими общекультурными компетенциями (ОК):

- способен к самостоятельному обучению новым методам исследования, к изменению научного и научно-производственного профиля своей профессиональной деятельности (ОК- 2);

- осознает сущность и значение информации в развитии современного общества; владеет основными методами, способами и средствами получения, хранения, переработки информации (ОК-11);

Выпускник должен обладать следующими профессиональными компетенциями (ПК):

- применять перспективные методы исследования и решения профессиональных задач на основе знания мировых тенденций развития вычислительной техники и информационных технологий (ПК-1);

- организовывать работу и руководить коллективами разработчиков программных средств информационных и автоматизированных систем (ПК-7).

Обозначение предметной области «распределенные информационные системы» и проблемных вопросов изучения дисциплины

Понятие «персональный компьютер», возникшее в уже далеком 1945 году и обозначающее индивидуальную работу пользователя в отдельно взятой комнате, изолированно от других пользователей, претерпело большие изменения и в реальной обыденной жизни и в виртуальном характере общения пользователя с информационными объектами. Начиная с середины восьмидесятых годов большие и дорогие майнфреймы уступают место компактным компьютерам с более мощными микропроцессорами. Следующий виток технологического развития обозначается появлением локальных сетей (Local-Area Networks, LAN), позволяющих объединить сотни компьютеров, находящихся в здании, таким образом, что машины в состоянии обмениваться небольшими порциями информации за несколько микросекунд. Большие массивы данных передаются с машины на машину со скоростью от 10 Мбит/с до 10 Гбит/с. Затем появляются глобальные сети (Wide-Area Networks, WAN), позволяющие миллионам машин во всем мире обмениваться информацией со скоростями, варьирующимися от 64 кбит/с (килобит в секунду) до гигабит в секунду.

В результате развития этих технологий сегодня не просто возможно, но и достаточно легко можно собрать компьютерную систему, состоящую из множества компьютеров, соединенных высокоскоростной сетью. Ее можно назвать простейшей распределенной информационной системой (РИС), в отличие от предшествовавших ей централизованных, или однопроцессорных систем, состоявших из одного компьютера, его периферии и, возможно, нескольких удаленных терминалов.

Технологический скачок революционного развития вычислительной техники потребовал концептуальных изменений в использовании средств обработки информации.

Появился новый термин – «распределенная информационная система». Возникает научная задача – термину РИС нужно дать лаконичное и научно обоснованное определение. На сегодняшний день по утверждению известного специалиста в

области информатики Э.Таненбаума, не существует общепринятого и в то же время строгого определения распределенной системы. В современной литературе можно выделить следующие научные толкования нашего термина как распределенная автоматизированная система (РАС):

- РАС — это автоматизированная система управления, которая приобрела специфику территориально рассредоточенной автоматизированной системы;

- РАС — это совокупность независимых объектов, которые взаимодействуют с целью решения проблемы, которая не может быть решена одним объектом индивидуально. При таком подходе распределенной является любая вычислительная система, где обработка данных разделена между двумя и более компьютерами;

- РАС — это совокупность независимых компьютеров, представляющаяся пользователям единой объединенной системой.

Такой подход к определению распределенной системы имеет свои недостатки. Например, все используемое в такой распределенной системе программное обеспечение могло бы работать и на одном единственном компьютере, однако с точки зрения приведенного выше определения такая система уже перестанет быть распределенной;

- распределенной является такая вычислительная система, в которой неисправность компьютера, о существовании которого пользователи ранее даже не подозревали, приводит к остановке всей их работы. Значительная часть распределенных вычислительных систем, к сожалению, удовлетворяют такому определению, однако формально оно относится только к системам с уникальной точкой уязвимости.

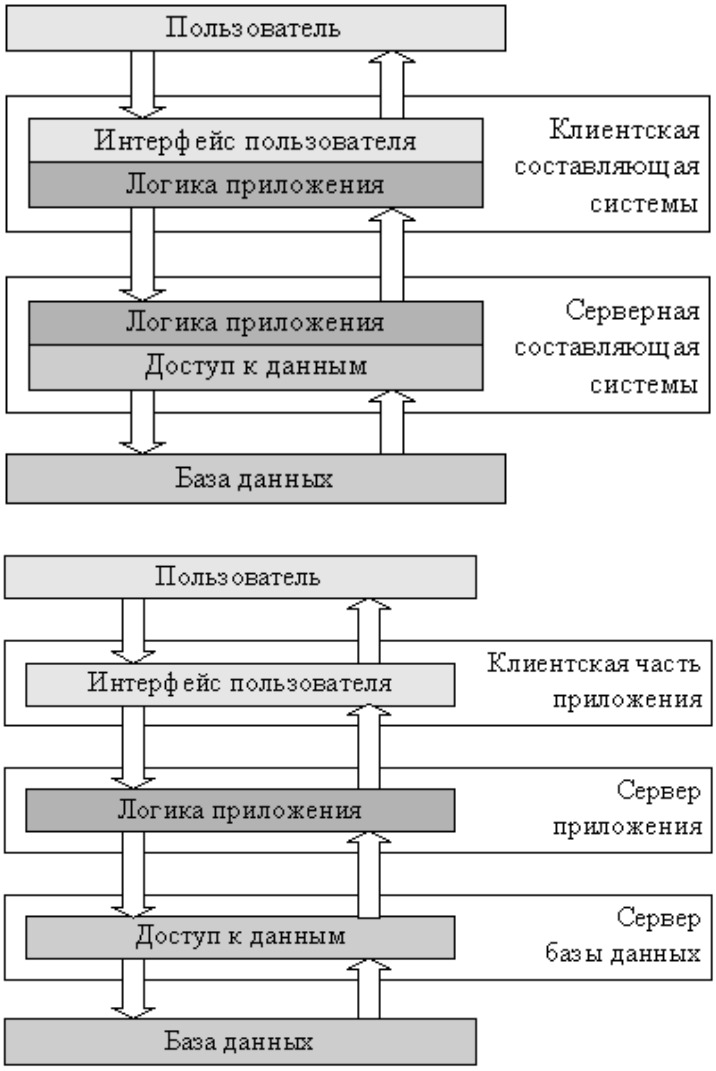
Первый прототип РИС, имеющий структуру **«клиент – сервер»**, следует рассматривать (рисунок 1) как некое типичное приложение, которое в соответствии с современными представлениями может быть разделено на следующие логические уровни:

- пользовательский интерфейс (ИП);

- логика приложения (ЛП);

- доступ к данным (ДД), работающий с базой данных (БД).

Пользователь системы взаимодействует с ней через интерфейс пользователя, база данных хранит данные, описывающие предметную область приложения, а уровень логики приложения реализует все алгоритмы, относящиеся к предметной области.



Двухзвенная (клиент-серверная) архитектура

Трехзвенная Архитектура

Рисунок 1 – Архитектура «клиент - сервер»

Поскольку на практике разных пользователей системы обычно интересует доступ к одним и тем же данным, наиболее простым разнесением функций такой системы между несколькими компьютерами будет разделение логических уровней приложения между одной серверной частью приложения, отвечающим за доступ к

данным, и находящимися на нескольких компьютерах клиентскими частями, реализующими интерфейс пользователя. Логика приложения может быть отнесена к серверу, клиентам, или разделена между ними. Архитектуру построенных по такому принципу приложений называют клиент-серверной или двухзвенной. На практике подобные системы часто не относят к классу распределенных, но формально они могут считаться простейшими представителями распределенных систем.

Развитием архитектуры клиент-сервер является трехзвенная архитектура, в которой интерфейс пользователя, логика приложения и доступ к данным выделены в самостоятельные составляющие системы, которые могут работать на независимых компьютерах. Запрос пользователя в подобных системах последовательно обрабатывается клиентской частью системы, сервером логики приложения и сервером баз данных. Однако обычно под распределенной системой понимают системы с более сложной архитектурой, чем трехзвенная.

Применительно к приложениям автоматизации деятельности предприятия, распределенными обычно называют системы с логикой приложения, распределенной между несколькими компонентами системы, каждая из которых может выполняться на отдельном компьютере. Например, реализация логики приложения системы розничных продаж (рисунок 2) должна использовать запросы к логике приложения третьих фирм, таких как поставщики товаров, системы электронных платежей или банки, предоставляющие потребительские кредиты. Таким образом, в обиходе под распределенной системой часто подразумевают рост многозвенной архитектуры "в ширину", когда запросы пользователя не проходят последовательно от интерфейса пользователя до единственного сервера баз данных.

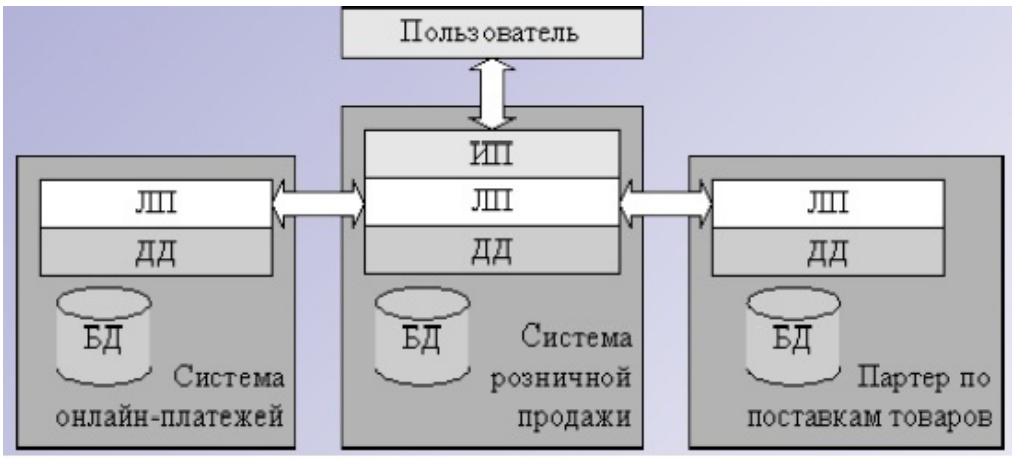


Рисунок 2

В качестве другого примера (рисунок 3) распределенной системы можно привести **сети прямого обмена данными между клиентами** (peer-to-peer networks). Если предыдущий пример имел "древовидную" архитектуру, то сети прямого обмена организованы более сложным образом. Подобные системы являются в настоящий момент, вероятно, одними из крупнейших существующих распределенных систем, объединяющие миллионы компьютеров.

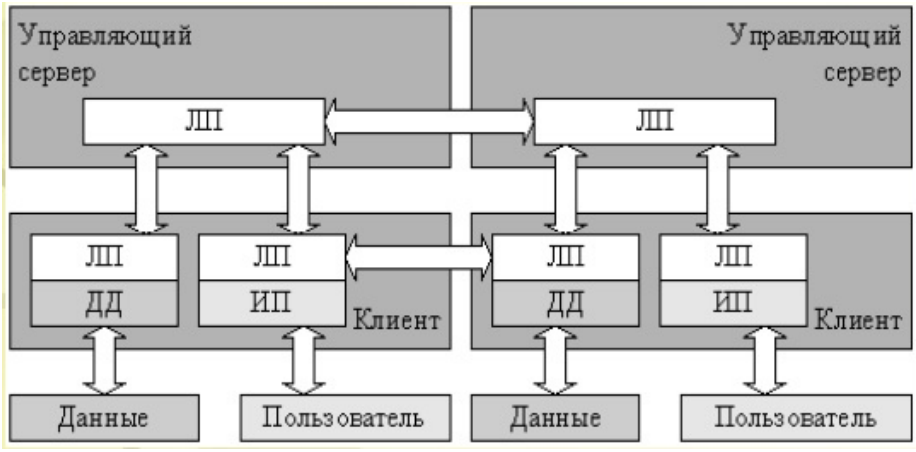


Рисунок 3

Другая популярная архитектура — **Grid.** Грид вычисления — это форма распределённых вычислений, в которой «супер и виртуальный компьютер» представлен в виде кластера соединенных с помощью сети, слабосвязанных компьютеров, работающих вместе для выполнения огромного количества заданий (операций, работ). Эта технология была применена для решения научных, математических задач, требующих для решения значительных вычислительных ресурсов. Грид вычисле-

ния используются также и в коммерческой инфраструктуре для решения таких трудоёмких задач как экономическое прогнозирование, сейсмоанализ, разработка и изучение свойств новых лекарств. Если рассмотреть какие же **характеристики** стали присущи к распределенным информационным системам, то можно отметить следующие:

- прозрачность реализации;
- открытость;
- легкая масштабируемость и расширяемость;
- устойчивость к авариям;
- наличие промежуточного уровня.

Первая из характеристик состоит в том, что от пользователей скрыты различия между компьютерами и способы связи между ними. То же самое относится и к внешней организации распределенных систем.

Другой важной характеристикой распределенных систем является способ, при помощи которого пользователи и приложения единообразно работают в распределенных системах, независимо от того, где и когда происходит их взаимодействие.

Распределенные системы должны также относительно легко поддаваться расширению, или масштабированию. Эта характеристика является прямым следствием наличия независимых компьютеров, но в то же время не указывает, каким образом эти компьютеры на самом деле объединяются в единую систему.

Распределенные системы обычно существуют постоянно, однако некоторые их части могут временно выходить из строя. Пользователи и приложения не должны уведомляться о том, что эти части заменены или починены или что добавлены новые части для поддержки дополнительных пользователей или приложений.

Для того чтобы поддержать представление различных компьютеров и сетей в виде единой системы, организация распределенных систем часто включает в себя дополнительный уровень программного обеспечения, находящийся между верхним

уровнем, на котором находятся пользователи и приложения, и нижним уровнем, состоящим из операционных систем.

Соответственно, такая распределенная система обычно называется **системой промежуточного уровня** (middleware).

Использование протокола TCP/IP посредством сокетов предоставляет стандартный, межплатформенный, но низкоуровневый сервис для обмена данными между компонентами. Для выполнения сформулированных выше требований к распределенным системам функции сеансового и представительского уровня должна взять на себя некоторая **промежуточная среда** (middleware), называемая так же промежуточным программным обеспечением. Такая среда (рисунок 4) должна помогать разработчикам создавать открытые, масштабируемые и устойчивые распределенные системы.

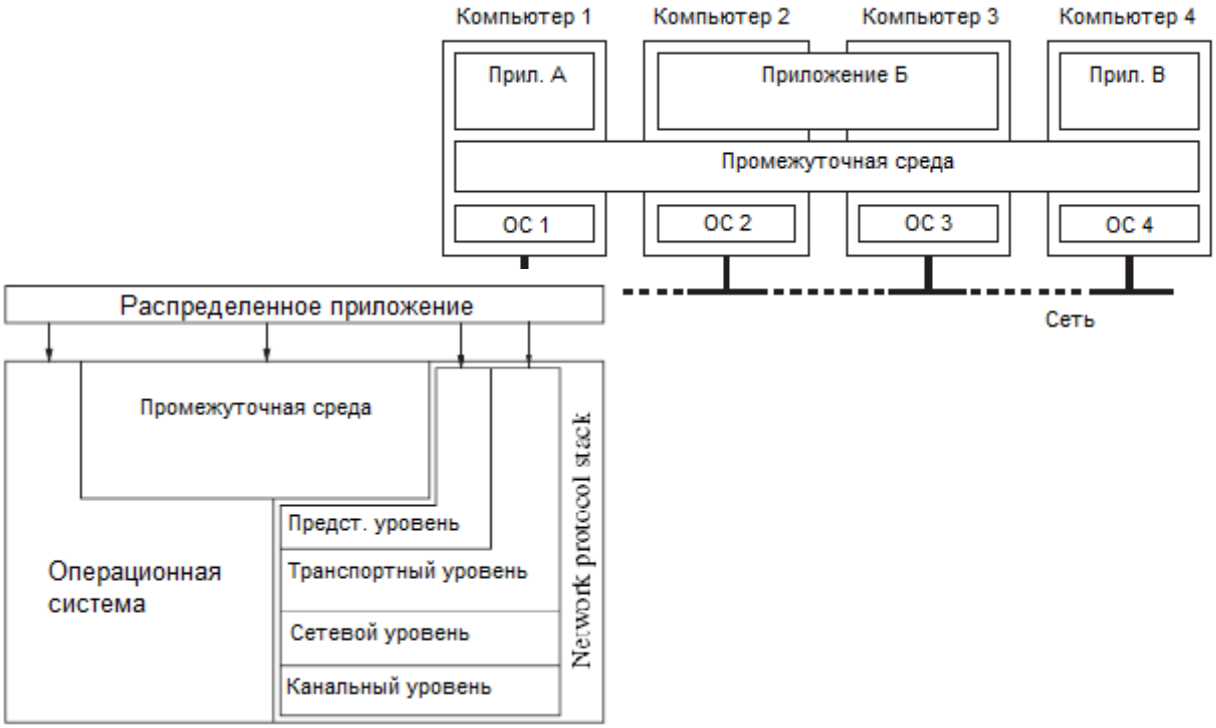


Рисунок 4

Для достижения этой цели промежуточная среда должна обеспечить сервисы для взаимодействия компонент распределенной системы. К таким сервисам относятся:

- обеспечение единого и независимого от операционной системы механизма использования одними программными компонентами сервисов других компонент;

- обеспечение безопасности распределенной системы: аутентификация и авторизация всех пользователей сервисов компоненты и защита передаваемой между компонентами информации от искажения и чтения третьими сторонами;

- обеспечение целостности данных: управление транзакциями, распределенными между удаленными компонентами системами;

- балансировка нагрузки на серверы с программными компонентами;

- обнаружение удаленных компонент.

Чтобы сделать разработку и интеграцию распределенных приложений как можно более простой, основная часть программного обеспечения промежуточного уровня базируется на некоторой модели, или парадигме, определяющей распределение и связь. Относительно простой моделью является **представление всех наблюдаемых объектов в виде файлов**, построенной по принципу распределенной файловой системы (distributed file system). Во многих случаях это программное обеспечение всего на один шаг ушло от сетевых операционных систем в том смысле, что прозрачность распределения поддерживается только для стандартных файлов (то есть файлов, предназначенных только для хранения данных). Процессы, например, часто должны запускаться исключительно на определенных машинах. Программное обеспечение промежуточного уровня, основанное на модели распределенной файловой системы, оказалось достаточно легко масштабируемым, что способствовало его популярности.

Другая важная ранняя модель программного обеспечения промежуточного уровня основана на **удаленных вызовах процедур** (Remote Procedure Calls, RPC). В этой модели акцент делается на сокрытии сетевого обмена за счет того, что процессу разрешается вызывать процедуры, реализация которых находится на удаленной машине. При вызове такой процедуры параметры прозрачно передаются на удаленную машину, где, собственно, и выполняется процедура, после чего результат выполнения возвращается в точку вызова процедуры. За исключением, вероят-

но, некоторой потери производительности, все это выглядит как локальное исполнение вызванной процедуры: вызывающий процесс не уведомляется об имевшем место факте сетевого обмена.

По мере того как все более входит в моду ориентированность на объекты, становится ясно, что если вызов процедуры проходит через границы отдельных машин, он может быть представлен в виде прозрачного обращения к объекту, находящемуся на удаленной машине. Это привело к появлению разнообразных систем промежуточного уровня, реализующих представление о распределенных объектах (distributed objects). Идея распределенных объектов состоит в том, что каждый объект реализует интерфейс, который скрывает все внутренние детали объекта от его пользователя. Интерфейс содержит методы, реализуемые объектом, не больше и не меньше. Все, что видит процесс, — это интерфейс. Когда процесс вызывает метод, реализация интерфейса на машине с процессом просто преобразует вызов метода в сообщение, пересылаемое объекту. Объект выполняет запрашиваемый метод и отправляет назад результаты. Затем реализация интерфейса преобразует ответное сообщение в возвращаемое значение, которое передается вызвавшему процессу. Microsoft DCOM (Distributed COM — распределённая COM) основана на технологии DCE/RPC (разновидности RPC). DCOM позволяет COM-компонентам взаимодействовать друг с другом по сети. Технология DCOM обеспечивает базовые установки безопасности позволяя задавать, кто и из каких машин может создавать экземпляры объекта и вызывать его методы; OMG CORBA (Common Object Request Broker Architecture — общая архитектура брокера объектных запросов) — это технологический стандарт, продвигаемый консорциумом OMG, задачей которого является осуществить интеграцию изолированных систем, дать возможность программам, написанным на разных языках, работающим на разных узлах сети, взаимодействовать друг с другом так же просто, как если бы они находились в адресном пространстве одного процесса; Java RMI (Remote Method Invocation) — программный интерфейс вызова удаленных методов в языке Java.

Как модели могут упростить использование сетевых систем, вероятно, наилучшим образом видно на примере World Wide Web. Успех среды Web в основном определяется тем, что она построена на базе потрясающе простой, но высокоэффективной модели распределенных документов (distributed documents). В модели, принятой в Web, информация организована **в виде документов,** каждый из которых размещен на машине, расположение которой абсолютно прозрачно. Документы содержат ссылки, связывающие текущий документ с другими. Если следовать по ссылке, то документ, с которым связана эта ссылка, будет извлечен из места его хранения и выведен на экран пользователя. Концепция документа не ограничивается исключительно текстовой информацией. Например, в Web поддерживаются аудио- и видеодокументы, а также различные виды документов на основе интерактивной графики.

Итак, еще раз перечислим и кратко охарактеризуем модели ПУ:

1 Распределенная файловая система

обозначение - «Distributed File System»;

достоинство - Относительно простая модель;

цель: обеспечить прозрачный доступ удаленных пользователей к файловой системе;

пример: NFS.

2 Удаленный вызов процедур

обозначение – «Remote Procedure Call (RPC)»;

цель: обеспечение прозрачности удаленного исполнения кода;

особенности функционирования:

- реализация процедуры находится на сервере;
- клиент передает параметры процедуры;
- сервер исполняет процедуру и возвращает результат
- некоторая потеря производительности;
- весь сетевой обмен скрыт от процесса.

3 Распределенные объекты

обозначение – «Distributed Objects:»;

особенности функционирования:

- каждый объект реализует интерфейс;

- интерфейс содержит методы, реализуемые объектом;

- процесс видит только интерфейс;

- наиболее популярные технологии распределенных объектов в настоящее время:

- Microsoft DCOM;

- OMG CORBA;

- Java RMI.

4 Распределенные документы

обозначение «Distributed Documents»;

- реализация: World Wide Web

- цель: Прозрачность размещения документов;

особенности функционирования:

- ссылки связывают документы;

- содержимое не ограничено текстовой информацией.

Кратко сформулируем задачи промежуточного уровня:

- обеспечение интероперабельности;

- обеспечение безопасности;

- обеспечение целостности данных;

- балансировка нагрузки;

- обнаружение удаленных компонент.

Чтобы достигнуть цели своего существования – улучшения выполнения запросов пользователя – распределенная информационная система должна удовлетворять некоторым необходимым требованиям.

Можно сформулировать следующий **набор требований**, которым в наилучшем случае должна удовлетворять РИС.

Открытость. Все протоколы взаимодействия компонент внутри распределенной системы в идеальном случае должны быть основаны на общедоступных стандартах. Это позволяет использовать для создания компонент различные средства разработки и операционные системы. Каждая компонента должна иметь точную и полную спецификацию своих сервисов. В этом случае компоненты распределенной системы могут быть созданы независимыми разработчиками. При нарушении этого требования может исчезнуть возможность создания распределенной системы, охватывающей несколько независимых организаций.

Масштабируемость. Масштабируемость вычислительных систем имеет несколько аспектов. Наиболее важный из них – возможность добавления в распределенную систему новых компьютеров для увеличения производительности системы, что связано с понятием балансировки нагрузки (load balancing) на серверы системы. К масштабированию относятся так же вопросы эффективного распределения ресурсов сервера, обслуживающего запросы клиентов.

Поддержание логической целостности данных. Запрос пользователя в распределенной системе должен либо корректно выполняться целиком, либо не выполняться вообще. Ситуация, когда часть компонент системы корректно обработали поступивший запрос, а часть – нет, является наихудшей.

Устойчивость. Под устойчивостью понимается возможность дублирования несколькими компьютерами одних и тех же функций или же возможность автоматического распределения функций внутри системы в случае выхода из строя одного из компьютеров. В идеальном случае это означает полное отсутствие уникальной точки сбоя, то есть выход из строя одного любого компьютера не приводит к невозможности обслужить запрос пользователя.

Безопасность. Каждый компонент, образующий распределенную систему, должен быть уверен, что его функции используются авторизированными на это компонентами или пользователями. Данные, передаваемые между компонентами, должны быть защищены как от искажения, так и от просмотра третьими сторонами.

Эффективность. В узком смысле применительно к распределенным системам под эффективностью будет пониматься минимизация накладных расходов, связанных с распределенным характером системы. Поскольку эффективность в данном узком смысле может противоречить безопасности, открытости и надежности системы, следует отметить, что требование эффективности в данном контексте является наименее приоритетным. Например, на поддержку логической целостности данных в распределенной системе могут тратиться значительные ресурсы времени и памяти, однако система с недостоверными данными вряд ли нужна пользователям.

Классическим примером системы, в значительной мере отвечающей всем представленным выше требованиям, является система преобразования символьных имен в сетевые IP-адреса (DNS). Система имен – организованная иерархически распределенная система, с дублированием всех функций между двумя и более серверами.

Повышение отношения производительности к затратам. Любая задача может быть разделена между между различными компьютерами в распределенной системе. Такая конфигурация обеспечивает лучшее соотношение производительности к стоимости системы. Это особенно актуально для конфигурации «сеть рабочих станций» (NOW).

Масштабируемость. Компьютеры, как правило, подключены к глобальной компьютерной сети, поэтому установка новых компьютеров непосредственно не создает узких мест в компьютерной сети.

Модульность и дополнительная расширяемость. Гетерогенные единицы могут быть добавлены в систему без снижения производительности, так как используется промежуточный уровень взаимодействия. Аналогично, существующие единицы могут быть легко заменены новыми.

Итак, можно считать, что предметная область обозначена и выделены проблемные вопросы, которые решаются специалистами информационных технологий. Исследования в области распределенных систем достаточно сложные, поэтому за выдающуюся статью по распределенным вычислениям ежегодно вручается

«Премия Дейкстры» (http://www.podc.org/dijkstra/).

На данном этапе изучения дисциплины можно запомнить определение:

РИС – это совокупность автономных компьютеров, взаимодействующих через компьютерную сеть и промежуточную среду, которая позволяет компьютерам координировать свою деятельность и предоставлять доступ к ресурсам системы так, что пользователям система представляется единой и целостной.

Распределенная система обладает следующими свойствами:

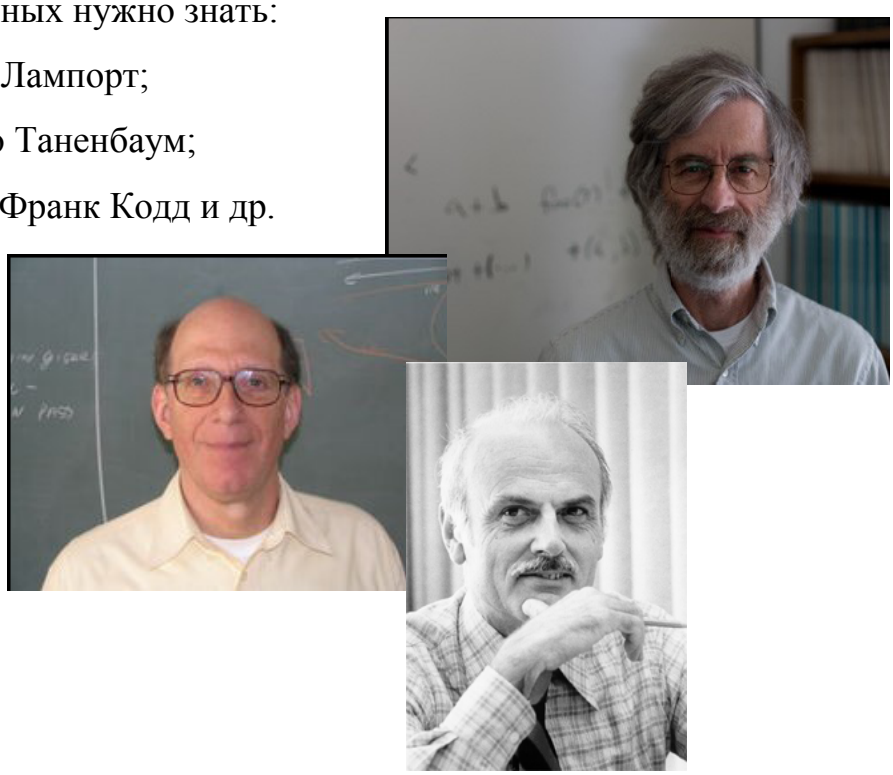
- отсутствие общей физической шины;
- отсутствие общей памяти;
- географическое распределение;
- автономность и гетерогенность.

Известных исследователей в области распределенных информационных систем и баз данных нужно знать:

Лесли Лампорт;

Энрдю Таненбаум;

Эдгар Франк Кодд и др.



Чтобы подробно разобраться со всеми проблемами, обозначенными в теории и практике РИС, ну проштудировать многотомное собрание сочинений. В данном учебном пособии обозначим рамки и направления изучения распределенных информационных систем. Объектом изучения обозначим корпоративную распределенную информационную систему организации, предприятия (рисунок 5).

Типовой пример РИС такой сети можно представить схемой компьютерного интегрированного полиграфического производства (рисунок 6).

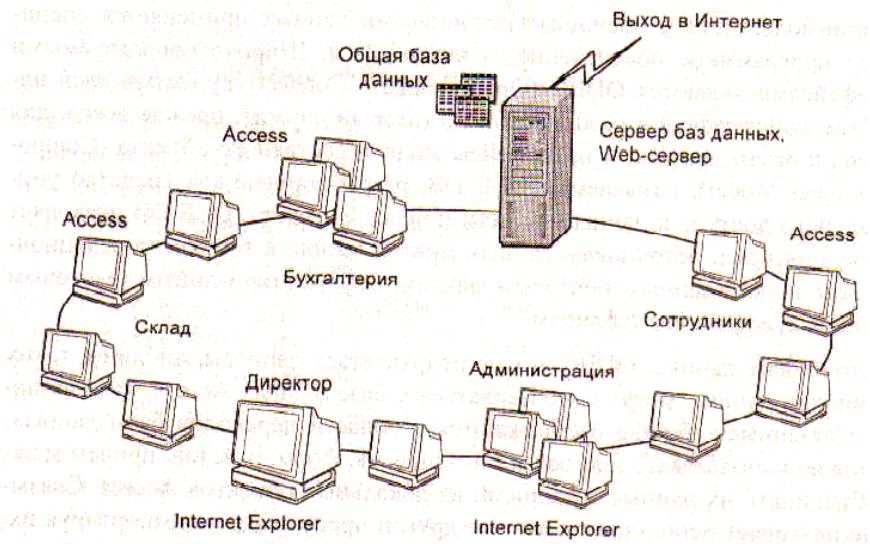


Рисунок 5 - Корпоративная сеть с SQL - и WEB - серверами

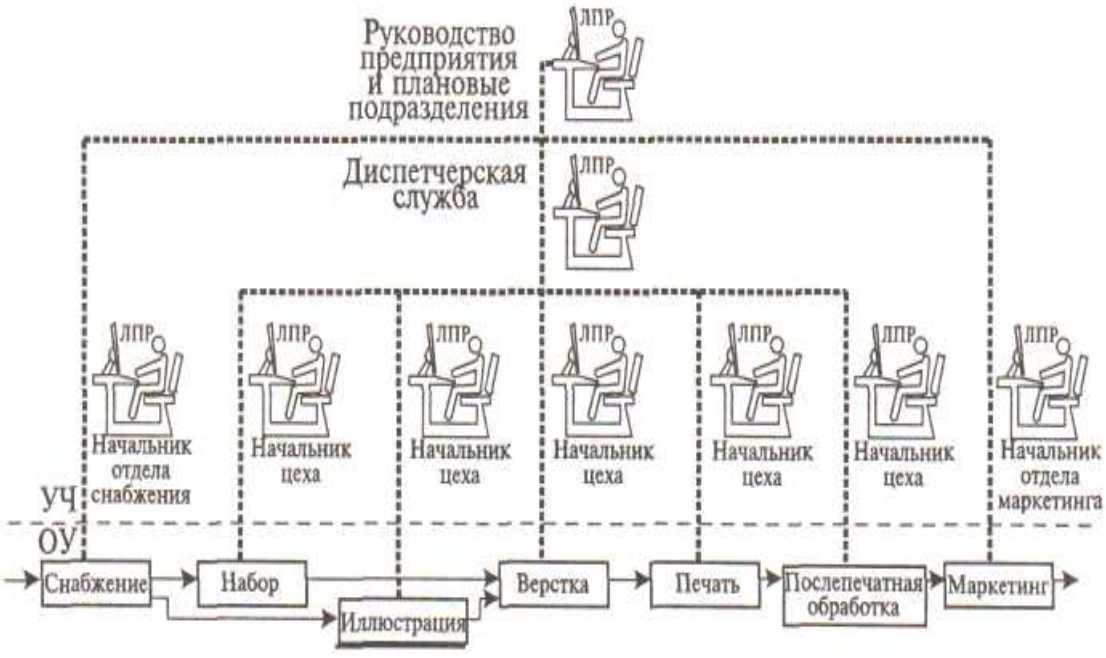


Рисунок 6

УЧ — управляющая часть; ОУ — объект управления

Лекционный материал учебного пособия можно сгруппировать по трем разделам. Первый раздел предназначен для изучения основных составляющих автоматизированной информационной системы (АИС). Проведем маленький экспери-

мент. Ответьте на два простых вопроса: что такое данные? и что такое информация? Большинство отвечающих относятся к одному варианту ответа: данные – это информация; информация – это данные. Круг замыкается! Простые и обыденные понятия, но мы с ними свыклись и не вдаемся в суть определений. Поэтому в первом разделе учебного пособия уделено основное внимание разъяснению терминов, определений объектов, составляющих информационную систему (ИС), концепции построения баз данных как основы ИС.

Второй раздел раскрывает содержание распределенной обработки информации в РИС, определяет концепцию построения распределенных баз данных (хранилищ и витрин данных) в структурах OLTP и OLAP. Важно уяснить как распределяется презентационная логика, логика аналитической обработки, логика баз данных в РИС.

Третий раздел посвящен изучению технологий и методик моделирования ИС. Должен быть понятен алгоритм проектирования РИС с учетом требований CASE - и CALS – технологий, используемых за рубежом и российских аналогов. Специалист по РИС должен знать весь перечень стадий и этапов проектирования РИС. Для проектирования РИС необходимо уметь пользоваться современными инструментальными средствами разработки. Заканчивается курс лекций практическими примерами проектирования и разработки РИС в современной инструментальной среде Visual Studio 2005, 2010, при этом проектировщику достаточно знать основы событийно управляемого программирования и разработки Windows-приложения на языке C#. Разработка базы данных демонстрируется на примере СУБД SQL Server 2005, 2008. Важно и необходимо уяснить, как проектируется и создается удаленный запрос с клиентского приложения на сервер базы данных с помощью функций пользователя или хранимых процедур. Специалист по РИС должен в обязательном порядке выполнять фрагментацию и локализацию данных в распределенной базе данных.

Итак, предметная область изучения обозначена. Содержание предмета изучения изложено в семи лекциях.

1 Лекция 1. Назначение и основные компоненты информационной системы

Первая лекция предназначена для общего введения в теорию и практику построения информационных систем на основе баз данных, рассмотрения и обсуждения основных терминов и определений компонентов информационных систем. Лекция посвящена рассмотрению общетеоретических вопросов, касающихся системы баз данных как информационной системы, использующей информационные технологии по обработке информации. Некоторые термины в системе баз данных не имеют четко определенных государственным стандартом определений, что составляет основную проблему однозначного понимания некоторых определений и понятий баз данных как информационной системы. Для решения данной основной проблемы обсудим различные подходы и общепринятые соглашения (нотации), существующие в современной научной литературе.

Ключевые слова: система, управление, информация, данные, базы данных, система управления базами данных (СУБД), уровни представления баз данных, модели данных.

Введение

В прошлом информация считалась сферой бюрократической работы и ограниченным инструментом для принятия решений. Сегодня информацию рассматривают как один из основных ресурсов развития общества, а информационные системы и технологии как средство повышения производительности и эффективности деятельности людей.

Наиболее широко информационные системы и технологии используются в производственной, управленческой и финансовой деятельности, хотя начались

подвижки в сознании людей, занятых и в других сферах, относительно необходимости их внедрения и активного применения.

Основные идеи современной информационной технологии базируются на концепции *баз данных* (БД), ранее упоминаемых достаточно часто без какого-либо их детального пояснения.

Согласно данной концепции основой информационной технологии являются *данные,* организованные в БД, адекватно отражающие реалии действительности в той или иной предметной области и обеспечивающие пользователя актуальной информацией в соответствующей предметной области.

Развитие современного промышленного производства и бизнеса невозможно без создания автоматизированных информационных систем (АИС), одно из назначений которых — предоставление пользователю достоверной информации, необходимой для принятия оптимального решения. В настоящее время ни одна из задач управления производством и бизнесом не должна выполняться без применения автоматизированных информационных систем. Сегодня мы должны рассматривать любую деятельность любого специалиста как некоторую систему принятия решений, поэтому специалисту и нужна достоверная информация. Таким образом, одной из важнейших функций информационной системы является информационное обеспечение процесса управления.

Итак, что же такое *База данных* и *Система управления базами данных!*

К сожалению, в большинстве книг по этому направлению информационных технологий нет достаточно четких определений. Рассмотрим и обсудим термины и определения, касающиеся баз данных.

Понятие об информационных системах

Под *системой* понимают любой объект, который одновременно рассматривается и как единое целое, и как объединенная в интересах достижения поставлен-

ных целей совокупность разнородных элементов. Системы значительно отличаются между собой как по составу, так и по главным целям.

Приведем примеры нескольких систем, состоящих из разных элементов и направленных на реализацию различных целей (таблица 1.1).

Таблица 1.1

Система	Элементы системы	Главная цель системы
Организация	Люди, оборудование, материалы, здания и др.	Производство
Компьютер	Электронные и электромеханические элементы, линии связи и др.	Обработка данных
Телекоммуникационная система	Компьютеры, модемы, кабели, сетевое программное обеспечение и др.	Передача информации
Информационная система	Компьютеры, компьютерные сети, люди, информационное и программное обеспечение	Производство профессиональной информации

В информатике понятие "*система*" широко распространено и имеет множество смысловых значений. Чаще всего оно используется применительно к набору технических средств и программ. Системой может называться аппаратная часть компьютера. Системой может также считаться множество программ для решения конкретных прикладных задач, дополненных процедурами ведения документации и управления расчетами.

Добавление к понятию "*система*" слова "*информационная*" отражает цель ее создания и функционирования. Информационные системы обеспечивают сбор, хранение, обработку, поиск и выдачу информации, необходимой в процессе принятия решений задач в любой предметной области. Появление электронных вычислительных машин и персональных компьютеров предопределило создание и внедрение автоматизированных информационных систем (АИС), которые значительно повысили производительность и результативность информационных технологий по обработке и выдачи информации.

В качестве основного классификационного признака АИС целесообразно рассматривать особенности автоматизируемой профессиональной деятельности -

процесса переработки входной информации для получения требуемой выходной информации, в котором АИС выступает в качестве инструмента должностного лица или группы должностных лиц, участвующих в управлении организационной системой.

В соответствии с предложенным классификационным признаком можно выделить следующие классы АИС (рисунок 1.1):

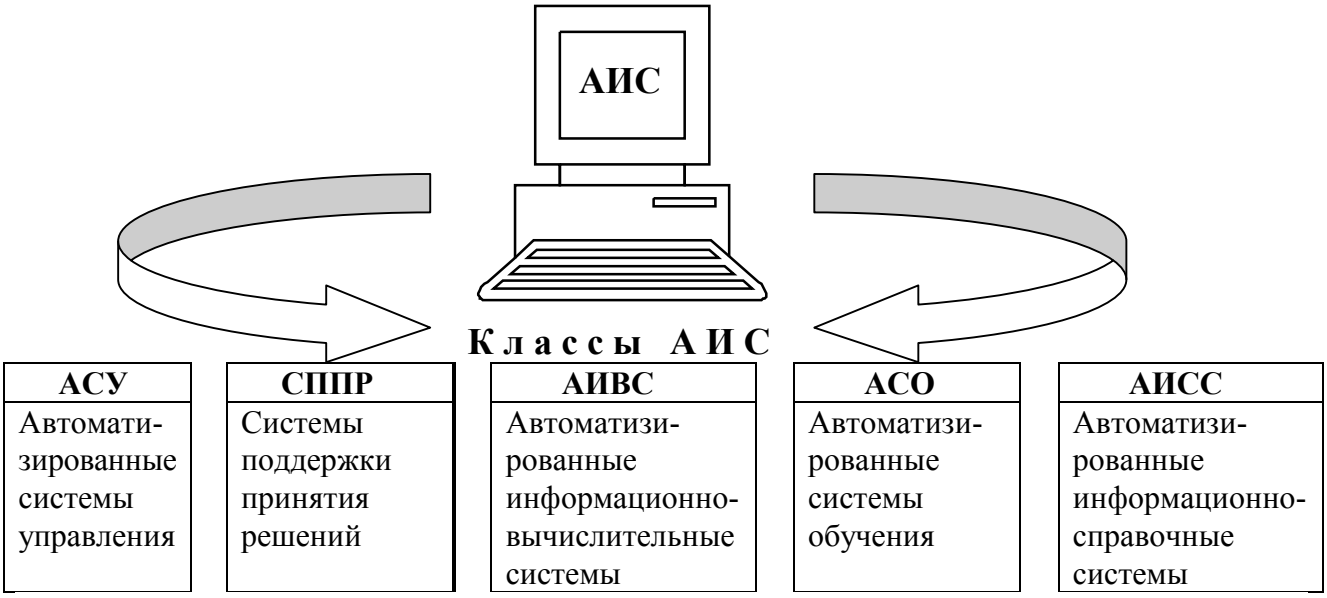


Рисунок 1.1 – Классификация АИС

АИС – взаимосвязанная совокупность средств, методов и персонала, используемых для хранения, обработки и выдачи информации в интересах достижения поставленной цели.

Современное понимание АИС как системы предполагает использование в качестве основного технического средства переработки информации персонального компьютера. Кроме того, техническое воплощение информационной системы само по себе ничего не будет значить, если не будет учтена роль человека, для которого предназначена производимая информация и без которого невозможно ее получение и представление.

Структуру АИС составляет совокупность отдельных ее частей, называемых *подсистемами.*

Подсистема – это часть системы, выделенная по какому-либо признаку.

Общую структуру информационной системы можно рассматривать как совокупность подсистем независимо от сферы применения. В этом случае говорят о *структурном признаке* классификации, а подсистемы называют *обеспечивающими.* Таким образом, структура любой информационной системы может быть представлена совокупностью обеспечивающих подсистем (рисунок 1.2).

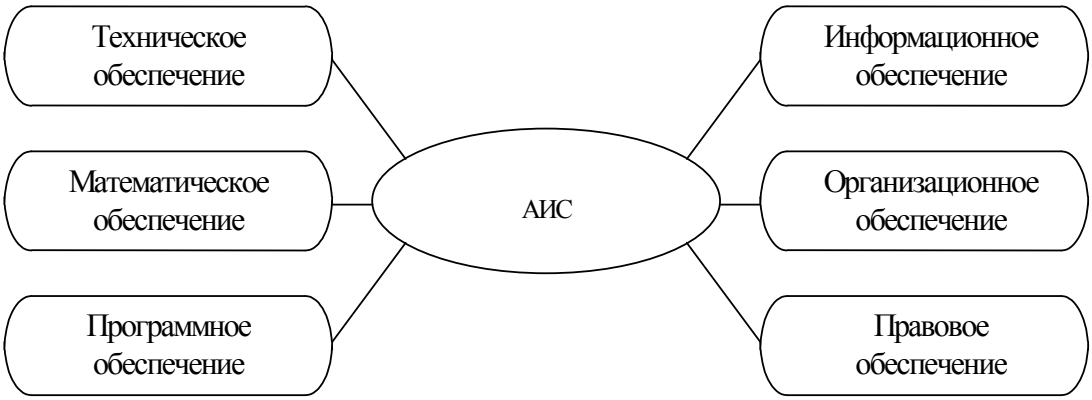


Рисунок 1.2 - Структура АИС как совокупность обеспечивающих подсистем

Среди обеспечивающих подсистем обычно выделяют информационное, техническое, математическое, программное, организационное и правовое обеспечение.

В понятии АИС присутствуют три очень важных, емких, ключевых, философско-методологических и специально-научных понятия:

- система;
- управление;
- информация.

И поэтому, чтобы разобраться в АИС, необходимо прежде всего выяснить:

- что такое система;
- что такое управление;
- что такое информация.

Система (от греч. SYSTEMA –целое, составленное из частей соединение) – это совокупность элементов, взаимосвязанных друг с другом, образующая определенную целостность, единство.

Введем набор понятий, связанных с использованием слова “система”.

Элемент – некоторый объект (материальный, энергетический, информационный), обладающий определенным функциональным назначением, отличающимся от назначения системы.

Введем обозначения:

M – элемент;

{M}- совокупность элементов;

$M \in \{M\}$ - принадлежность элементов совокупности.

Вопрос. Сколько или какое количество элементов необходимо, чтобы их совокупность стала системой? Древние философы спорили – сколько нужно камней, сложенных вместе, чтобы они образовали кучу. Вся доступная разуму человека природа состоит из систем, которые могут быть различны по масштабам: от бесконечно больших (галактика) до бесконечно малых (атом); различны по природе: материальные, энергетические, информационные. Практически любой объект с определенной точки зрения может рассматриваться как система, если совокупность элементов обладать двумя признаками:

а) связями, которые позволяют посредством переходов по ним от элемента к элементу соединить два любых элемента совокупности (рисунок 1.3).

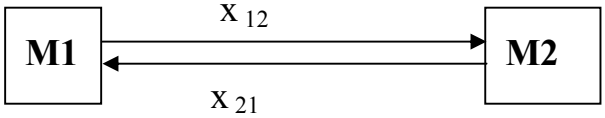


Рисунок 1.3 - Связь двух элементов

X_{12} – воздействие элемента M1 на элемент M2;

X_{21} – воздействие M2 на M1.

б) Свойством (назначением, функцией), отличным от свойств отдельных элементов совокупности.

По степени автоматизации решения своих функциональных задач системы могут быть:

- ручные, т.е. без средств автоматизации;

- автоматизированные, т.е. со средствами автоматизации при участии человека;

- автоматические, т.е. типа «автомат» без участия человека.

Дадим определение АСУ согласно ГОСТ 19675-74:

АСУ - это человеко-машинная система, обеспечивающая автоматизированный сбор и обработку информации, необходимой для оптимизации управления в различных сферах человеческой деятельности.

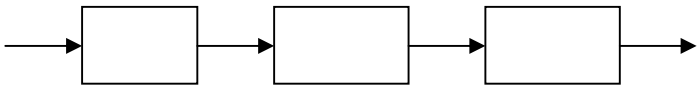
Большой системой называют систему, включающую большое количество однотипных элементов и однотипных связей. Пример: гирлянда, трубопровод.

Сложной системой называют систему, состоящую из элементов разных типов и обладающую разнородными связями между ними. Пример: космический корабль, автомобиль, морское судно и т.п.

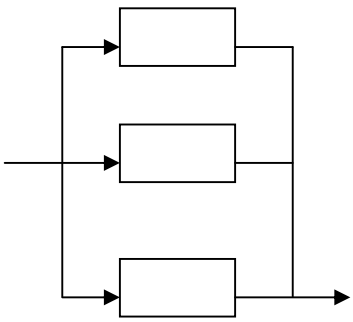
Простая система – это система, состоящая из небольшого числа элементов и не имеющая разветвленной структуры (нельзя выявить иерархические уровни).

Структура системы – совокупность внутренних, устойчивых связей между элементами системы, определяющая ее основные свойства.

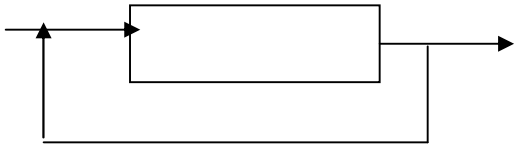
Простейшие типы структур систем показаны на рисунке 1.4.



а) последовательная



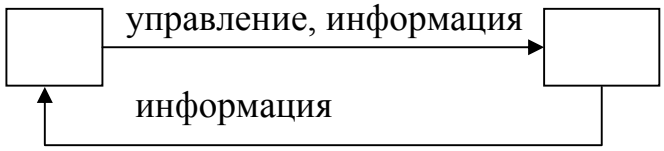
б) параллельная



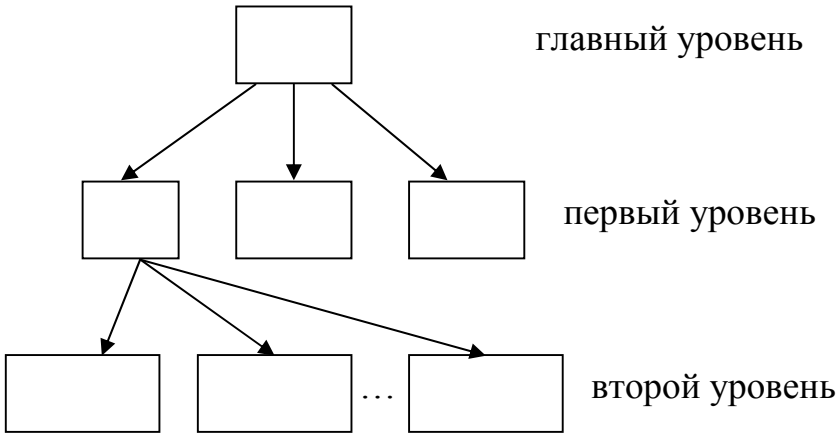
в) с обратной связью

Рисунок 1.4 - Типы структур систем

Иерархические структуры – это структуры с наличием подчиненности (рисунок 1.5).



а) простая



б) сложная

Рисунок 1.5 - Иерархические структуры

С понятием структура тесно связан термин «декомпозиция» - это деление системы на части, удобное для каких-то операций.

Примеры: автомобиль состоит из систем смазки, охлаждения, питания и др. Любая книга имеет содержание (оглавление).

Различают также статистические и динамические системы. Состояние статической системы с течением времени остается постоянным, динамические системы, наоборот, изменяют свое состояние во времени.

Динамические системы разделяют на детерминированные, т.е. полностью определенных в любой момент времени, и вероятностные (стохастические).

По характеру взаимодействия системы и внешней среды различают закрытые и открытые системы.

Системы обладают свойством целостности – это принципиальная несводимость свойств системы к сумме свойств составляющих ее элементов и, в то же время, зависимость свойств каждого элемента от его места и функции внутри системы.

Группа элементов системы, описываемая только своими входами и выходами и обладающая определенной целостностью, называется модулем (рисунок 1.6).

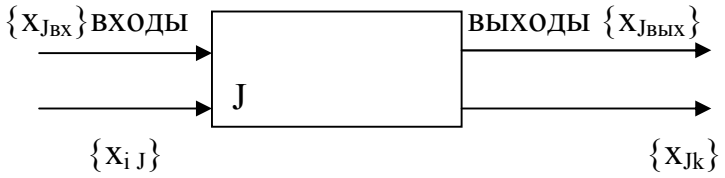


Рисунок 1.6 - Модуль J

$X_{Jвх}$ – внешние (не от системы) воздействия на элементы модуля;

X_{iJ} – связи от других элементов системы;

$X_{J\ вых\ -}$ воздействия на выходе системы;

X_{Jk} – воздействия на элементы системы.

В различных разделах науки и техники понятие модуль имеет различные синонимы:

- в технике: агрегат, блок, узел, механизм;

- в программировании: программа, программный модуль, логический блок;

- в организации и управлении – комиссия, подразделение.

Входы и выходы можно интерпретировать как:

1) сигнал – отклик;

2) воздействие – реакция;

3) запрос – ответ;

4) аргумент – решение и др.

Перейдем к анализу понятия управления в системах. Процессы управления протекают повсеместно, ежедневно, ежечасно, ежесекундно и охватывают буквально все стороны и моменты человеческой деятельности.

В науке управление стало изучаться и исследоваться сравнительно недавно, и связано это было прежде всего с именем всемирного ученого Норберта Винера

(1894-1964). В 1948 году в США и Европе вышла книга «Кибернетика, или Управление и связь в животном и машине», ознаменовавшая своим появлением рождение нового научного направления – кибернетика.

Кибернетические исследования заключаются в изучении общих свойств процессов управления и систем управления в живых и неживых системах.

Направления исследований кибернетики весьма разнообразны:

– теоретические (теория управления, теория информации, информатика);

– механические (исследование и проектирование АИС);

– биологические (нейрокибернетика – обработка информация в нервных тканях человека;

– бионика – искусственные органы;

– гомеостатика – автоматы–роботы;

– экономические;

– социальные.

В современном обществе кибернетика уступила пальму первенства информатике, но значение кибернетики как науки об общих принципах управления в живых и неживых системах, в искусственных системах и в обществе сохраняются и сейчас.

Итак, что такое управление?

Под управлением в самом общем виде понимают совокупность действий, осуществляемых человеком, группой людей или автоматическим устройством.

Эти действия направлены на поддержание или улучшение работы управляемого объекта в соответствии с имеющейся программой (алгоритмом функционирования) или с целью управления.

Управлять – это значит влиять на ход какого-либо процесса или состояния некоторого объекта и его положения в пространстве.

При изучении структуры процессов управления выделяют три основные проблемы. Первая группа задач - это задачи по изучению свойств объектов управления и их характеристик. Вторая группа задач – анализ различных видов органов управ-

ления, форм и способов выработки управляющих воздействий, формирование и согласование множества критериев качества (целей) управления, обеспечение устойчивости функционирования управляющих органов. Третья группа задач связана с передачей информации, ее восприятием, оценкой количества, кодированием, защитой от помех, достоверности. Все это относится к той части кибернетики, которую называют теорией информации.

Что же характерно для любого процесса управления?

1 Есть объект, которым управляют (рисунок 1.7). На него через исполнительный орган подается управляющее воздействие. Информация о состоянии объекта по цепи обратной связи поступает на управляющий орган, который выдает управляющие сигналы на исполнительный орган. Очевидно, что самым сложным с точки зрения функционирования, является управляющий орган, так как на него возложено одновременное и оптимальное управление, прием информации от внешней среды, прием информации от объекта.

2 В управлении всегда решается задача с определенной целью (целевой функцией). Решение на выдачу того или иного управляющего воздействия принимается согласно теории принятия решения. Как правило, целевую функцию в моделях описывают математически, задают параметры и ограничения, временные характеристики. Тогда оптимальность управления достигается в обеспечении экстремума выходных характеристик при определенных входных воздействиях.

3 Без информации нет управления.

Информация нужна и внешняя, и внутренняя (о состоянии объекта). Это информация полезная. Есть и информация в виде возмущающих воздействий (неверная, ложная, ненужная, избыточная и т.д.).

Циркуляция информации происходит в информационной системе (ИС), которая является как бы кровеносными сосудами в структуре элементов управления.

Основу информационной системы составляет база данных, в которой хранится большая по объему информация о какой-либо предметной области.

В неавтоматизированном варианте примером информационной системы является книжный каталог библиотеки. В настоящее время все большее развитие приобретают автоматизированные информационные системы. Ее элементы показаны на рисунке 1.7. Пользователь из органа управления выдает запрос в ИС. Этот запрос расшифровывается, формируется поисковое предписание (или поисковый образ), представляющее задание для процедуры поиска в базе данных. Поиск в базе данных осуществляется блоком поиска. Найденная информация выдается потребителю в удобной для него форме. АИС играют большую роль в инфраструктуре общества и являются одним из обязательных звеньев новых информационных технологий (НИТ).

Проанализируем понятие ИНФОРМАЦИЯ.

Отличительной чертой человеческого общества является то, что в течение длительного времени основным предметом труда оставались материальные объекты. Воздействуя на них, человек добывал себе средства к существованию, и на протяжении многих веков решалась задача усиления мускульных возможностей человека с помощью различных инструментов, агрегатов и машин. На это была направлена механизация производства, которая стала интенсивно внедряться в наличие двадцатого века. Развитие человеческого общества практически на всех этапах проходило на основе технического прогресса. Это и овладение огнем, и использование паровых машин, и проникновение в тайны атомной энергии, и т.п.

Повышению производительности труда способствовала автоматизация. В процессе формирования трудовых коллективов возникла необходимость обмена знаниями. Первоначально знания передавались устно из поколения в поколение.

Появление письменности позволило по-новому показать накопленные знания: представить их в виде информации.

Рисунок 1.7 - Структурная схема АИС

Понятие о данных и информации

В этом разделе рассмотрим и обсудим два основных понятия – это данные и информация, их взаимосвязь. Еще раз подчеркнем, что с философской точки зрения эти понятия однозначно и до конца не определены. Чтобы отличить базу данных от овощной базы необходимо прежде всего дать определение данных.

Данными называют описание в сознании человека предметов, событий и явлений окружающего мира. Существуют три основных формы описания и дальнейшего представления данных:

- символьная;

- текстовая;

- графическая.

Символьная форма, основанная на использовании символов - букв, цифр, знаков, является наиболее простой, но она практически применяется только для передачи несложных сигналов и различных событий. (Например - сигналы светофора).

Более сложной является *текстовая форма*, в которой, как и в предыдущей форме, используются символы - буквы, цифры, математические знаки. Однако информация заложена не только в этих символах, но и в их сочетании, порядке следования. Удобство текстовой информации обусловлено взаимосвязью текста и речи человека.

Самой емкой и сложной является *графическая форма* представления информации. К этой форме относятся виды природы, фотографии, чертежи, схемы, рисунки.

Говоря о формах информации важно еще раз подчеркнуть свойство нематериальной информации - для ее существования обязательно должен быть какой-либо материальный объект: свет, воздух, вода, электрический ток, эфир электромагнитных колебаний и т.д.

Итак, носителем информации может быть как непосредственно наблюдаемый физический объект, так и энергетический субстрат. В последнем случае информация представлена в виде сигналов световых, звуковых, электрических и т.д.

При отображении на носителе информация кодируется, т.е. ей ставится в соответствие форма, цвет, структура и другие параметры элементов носителя.

Примеры:

1 Почему человек различает цвета воспринимаемого изображения?

Потому, что простой белый цвет имеет частотные составляющие спектры электромагнитных колебаний для цветов: красного, оранжевого, желтого, зелено-

го, голубого, синего, фиолетового. Сетчатка глаз способна различать частотные спектры цветности и сообщать сведения в мозг человека.

2 Книга - носитель кодированной последовательности букв, цифр, символов, графики. Читая книгу, мы как раз и воспринимаем информацию, записанную на ее страницах, в виде кодовых комбинаций (слов), состоящих из последовательности символов (букв, цифр) принятого алфавита. То же самое можно сказать и относительно информации, сообщаемой в процессе устной речи.

В теории информации особого внимания заслужила наиболее стандартная и единая форма представления информации - двоичная форма. Она заключается в записи любой информации в виде последовательности только двух символов: 1, или «да», или «истина»; 0, или «нет», или «ложь». В ЭВМ эти символы обозначаются наличием либо отсутствием в рассматриваемой точке электрического или магнитного импульса. В этом случае реквизитом информации, т.е. самой малой порцией информации (меньше не может быть) является ответ на любой вопрос в виде «да» или «нет». Эта порция определяет единицу измерения информации, называемую «битом». Последовательность битов может иметь различную разрядность. Запись нулей и единиц производится по правилам кодирования, используемых в ЭВМ.

Поясняя определение данных, мы непроизвольно стали использовать термин информация. Уясним взаимосвязь этих двух терминов.

Мы с вами уяснили, что информация - это первичное понятие, точного определения которого не существует.

Существует четыре основных направлений толкования термина информация:

1 Информация - это смысл полученного сообщения, его интерпретация.

Пример учителя и ученика. Учитель имеет информацию о предмете. С помощью сообщений (рассказ с показом, демонстрация) передает ученику сведения, данные. Ученик получает сообщения и усваивает их. То, что он понял со своей точки зрения и есть информация. Степень адекватности реальному образу проверяется учителем контрольным опросом. В этом отношении компьютер никогда не

оценивает смысл информации, ему все равно с какими данными работать. Только человек имеет возможность получить информацию на основе данных компьютера.

2 Информация - это как содержание сообщений, так и само сообщение, данные. В этом смысле примером может служить книга с ценными для потребителя сведениями, газета, кодограмма и т.п.

3 Некоторые ученые и, прежде всего, философы считают, что информация - это третья составляющая основ мироздания (материя, энергия и информация).

Н. Винер в одной из работ написал: «Информация и есть информация, а не материя и энергия».

4 В математической теории информации понятие информация определяется только для случайных событий. В этом отношении информация - это то, что уменьшает неопределенность события.

Приведем пример: Компьютер с помощью генератора случайных чисел выдал число от 1 до 16. Наша задача угадать это число. Мы задаем вопросы компьютеру, а он отвечает «да» (истина, 1) или «нет» (ложь, 0). За какое минимальное количество вопросов можно отгадать число? Сколько нужно информации, чтобы угадать загаданное число? Неопределенность равна 16. Первый вопрос: задуманное число меньше 8? Ответ «да» или «нет» уменьшает неопределенность в два раза и мы получаем информацию, равную одному биту. Если число находится в пределах от 1 до 8, то мы задаем вопрос: число меньше 4? Получаем ответ и еще один бит информации и т.д. Итого, количество информации, необходимое для угадывания числа равно 4 битам.

Подведем итоги и решим проблемный вопрос в однозначном понятии информации:

1 Под словом «информация» (в переводе с латинского «Jnformation») понимается разъяснение, изложение, чего-либо, сообщение о чем-либо.

2 Ответьте на вопрос - материальна или нематериальна информация? Будем понимать так, информация не материальна, но информация является свойством

материи и не может существовать без своего материального носителя - средства переноса ее в пространстве и во времени.

Рассмотрим взаимосвязь двух понятий – данных и информации. Если рассматривать процесс передачи данных от источника данных до потребителя (рисунок 1.8), то можно сделать вывод о том, что источник в виде базы данных содержит большое количество различных и неупорядоченных данных, а потребителю информации нужна определенная и необходимая ему информация о конкретной предметной области.

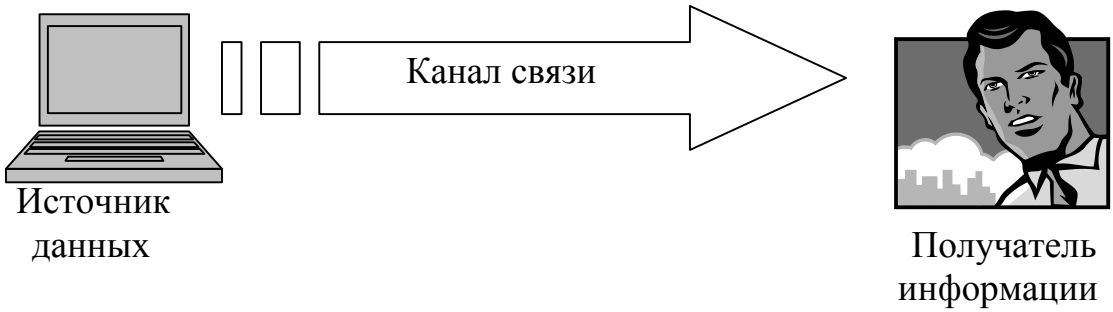


Рисунок 1.8 – Взаимосвязь данных и информации

Исходя из данной взаимосвязи мы можем дать свое определение информации. Информация – это необходимые для получателя данные, переданные по каналу связи от источника данных своевременно и достоверно.

Количество и качество информации – как их анализировать?

В предыдущем разделе мы дали определение информации исходя из взаимосвязи данных и информации, при этом указали два показателя качества информации – достоверность и своевременность. Возникает проблемный вопрос – как анализировать и оценивать качество и количество информации.

В свете идей науки о знаковых системах – семиотики адекватность информации, т.е. соответствие содержания образа отображаемому объекту, может выражаться в трех формах: синтаксический, семантический и прагматический.

Синтаксическая адекватность связана с воспроизведением формально-структурных характеристик отражения независимо от смысловых и потребительских (полезностных) параметров объекта. На синтаксическом уровне учитываются тип носителя и способ представления информации, скорость ее передачи и обработки, размеры кодов представления информации, надежность и точность преобразования этих кодов и т.п. Информацию, рассматриваемую только с синтаксических позиций, обычно называют данными.

Семантическая адекватность выражает аспект соответствия образа, знака и объекта, т.е. отношение информации и ее источника. Проявляется семантическая информация при наличии единства информации (объекта) и пользователя. Семантический аспект имеет в виду учет смыслового содержания информации; на этом уровне анализируются те сведения, которые отражает информация, рассматриваются смысловые связи между кодами представления информации.

Прагматическая адекватность отражает отношение информации и ее потребителя, соответствие информации цели управления, которые на ее основе реализуется. Прагматический аспект связан с ценностью, полезностью использования информации для выработки правильного управленческого решения. С этой точки зрения анализируются потребительские свойства информации.

Три формы адекватности информации соответствуют трем ступеням познания истины, сформулированным философами: «От живого созерцания к абстрактному мышлению и от него к практике – таков диалектический путь познания истины, познания объективной реальности». Первая ступень соответствует восприятию внешних структурных характеристик, т.е. синтаксической стороны информации; вторая – ступень формирования понятий и представлений, выявления смысла, содержания информации и ее обобщения; третья – непосредственно связана с практическим использованием информации в соответствии ее целевой функции деятельности системы. В соответствии с тремя формами адекватности выполняется и измерение информации. Терминологически принято говорить о количестве информации и об объеме данных. Объем данных в сообщении измеряется количеством

символов (разрядов) принятого алфавита в этом сообщении. Часто информация кодируется числовыми кодами в той или иной системе счисления. Естественно, что одно и то же количество разрядов в разных системах счисления может передать разное число состояний отображаемого объекта.

Действительно:

$$N = m^n,$$

где: N – число всевозможных отображаемых состояний;

m – основание системы счисления (разнообразие символов, применяемых в алфавите);

n – число разрядов (символов) в сообщении.

Поэтому в различных системах счисления один разряд имеет различный вес, и соответственно меняется единица измерения данных. Так, в двоичной системе счисления единицей измерения служит «бит» – двоичный разряд, в десятичной системе счисления – «дит», как десятичный разряд.

Например:

а) сообщение в двоичной системе 10111011 имеет объем данных $V_д = 8$ бит;

б) сообщение в десятичной системе 275903 имеет объем данных $V_д = 6$ дит.

В современной ЭВМ наряду с минимальной единицей данных «бит» широко используется укрупненная единица измерения «байт», равная 8 бит.

Определение количества информации на синтаксическом уровне невозможно без рассмотрения понятия неопределенности состояния системы (энтропии системы).

Действительно, получение информации о какой-либо системе всегда связано с изменением степени неосведомленности получателя о состоянии этой системы. До получения информации получатель мог иметь некоторые предварительные (априорные) сведения о системе α. Мера неосведомленности о системе $H(\alpha)$ и является для него мерой неопределенности состояния системы. После получения некоторого сообщения получатель приобретает некоторую дополнительную информацию $I_\beta(\alpha)$, уменьшающую его априорную неосведомленность так, что апо-

стериорная (после получения сообщения β) неопределенность состояния системы становится равной $H_\beta(\alpha)$. Тогда количество информации $I_\beta(\alpha)$ о системе α, полученное в сообщении β, определится как:

$$I_\beta(\alpha) = H(\alpha) - H_\beta(\alpha),$$

т.е. количество информации измеряется изменением (уменьшением) неопределенности состояния системы. Если конечная неопределенность $H_\beta(\alpha)$ обратится в нуль, то первоначальное неполное знание заменится полным знанием и количество информации станет равным:

$$I_\beta(\alpha) = H(\alpha).$$

Иными словами, энтропия системы $H(\alpha)$ может рассматриваться как мера недостающей информации. Энтропия системы $H(\alpha)$, имеющей N возможных состояний согласно формуле ШЕННОНА, равна:

$$H(\alpha) = -\sum_{i=1}^{N} P_i \log P_i,$$

где P_i –вероятность того, что система находится в i-м состоянии.

Для случая, когда все состояния системы равновероятны, ее энтропия определяется по формуле:

$$H(\alpha) = -\sum_{i-1}^{N} \frac{1}{N} \log \frac{1}{N} = \log N.$$

Рассмотрим пример. По каналу связи передается n- разрядное сообщение, использующее m различных символов.

Так как количество всевозможных кодовых комбинаций определяется по формуле $N = m^n$, то при равновероятности появления любой из них количество информации, приобретенной абонентом в результате получения сообщения, будет определяться по формуле ХАРТЛИ:

$$I = \log N = n \log m.$$

Если в качестве основания логарифма принять m, то формула упростится и количество информации станет равным:

$$I = n.$$

В данном случае количество информации (при условии полного априорного незнания абонентном содержания сообщения) будет равно объему данных $I = V_д$, полученных по каналу связи.

Наиболее часто используются двоичные и десятичные логарифмы. Единицами измерения в этих случаях будут соответственно «бит» и «дит».

Степень информативности сообщения определяется отношением количества информации к объему данных, т.е.

$$Y = 1/ V_д \text{ , причем } 0 \le Y \le 1,$$

где: Y – характеризует лаконичность сообщения.

С увеличением Y уменьшаются объемы работы по преобразованию информации (данных) в системе. Поэтому стремятся к повышению информативности, для чего разрабатываются специальные методы оптимального кодирования информации.

Семантическая мера информации

Синтаксические меры количества информации в общем случае не могут быть непосредственно использованы для измерения смыслового содержания, ибо имеют дело с обезличенной информацией, не выражающей смыслового отношения к объекту.

Для измерения смыслового содержания информации, т.е. ее количества на семантическом уровне наибольшее признание получила тезаурусная мера информации, предложенная Ю.И. ШНЕЙДЕРОМ. Он связывает семантические свойства информации прежде всего со способностью пользователя принимать поступившее сообщение. Используется понятие «тезаурус пользователя». Тезаурус можно трак-

товать как совокупность сведений, которыми располагает данная система, пользователь.

В зависимости от соотношений между смысловым содержанием информации $\hat{S}$ и тезаурусом пользователя S_n изменяется количество семантической информации J_c, воспринимаемой пользователем и включаемой им в дальнейшем в свой тезаурус.

При $S_n \approx 0$ пользователь не воспринимает, не понимает поступающую информацию; при $S_n \to \infty$ пользователь все знает, и поступающая информация ему не нужна: и в том, и в другом случае $J_c \approx 0$. Максимальное значение J_c приобретает при согласовании $\hat{S}$ с тезаурусом S_n ($S_n - S_{n\,opt}$), когда поступающая информация понятна пользователю и несет ему ранее не известные (отсутствующие в его тезаурусе) сведения (рисунок 1.9).

Следовательно, количество семантической информации в сообщении, количество новых знаний, получаемых пользователем, является величиной относительной.

Одно и то же сообщение может иметь смысловое содержание для компетентного пользователя и быть бессмысленным (семантический шум) для пользователя некомпетентного.

В то же время понятная, но известная компетентному пользователю информация представляет собой для него тоже семантический шум.

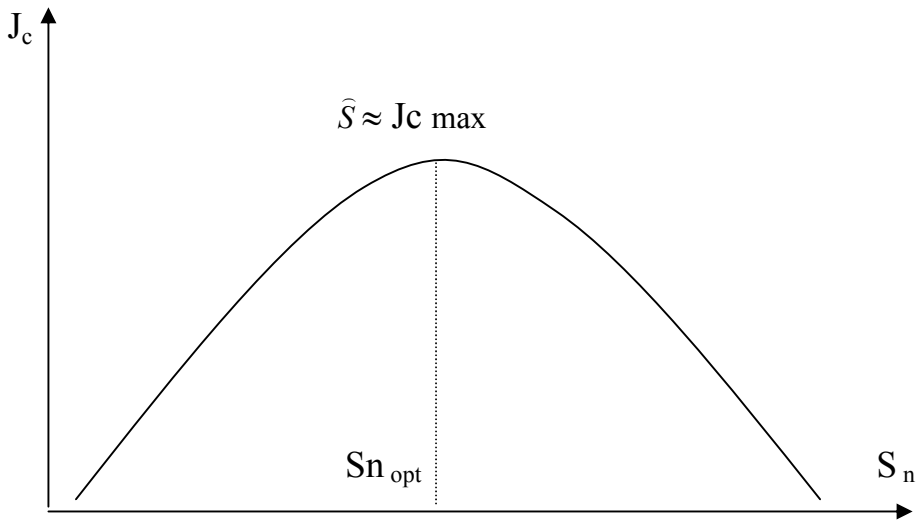


Рисунок 1.9 - Зависимость $J_c = f(S_n)$

При разработке информационного обеспечения АИС следует стремиться к согласованию величины $\hat{S}$ и S_n так, чтобы циркулирующая в системе информация была понятна, доступна для восприятия и обладала наибольшей содержательностью S, т.е.

$$S = J_c/V_д.$$

Прагматическая мера информации – это ее полезность, ценность для управления. Эта мера также величина относительная, обусловленная особенностями использования этой информации в той или иной системе.

Ценность информации целесообразно измерять в тех же самых единицах (или близких к ним), в которых измеряется целевая функция управления системой.

В автоматизированной системе управления производством, например, ценность информации определяется эффективностью осуществляемого на ее основе экономического управления, или иначе, приростом экономического эффекта функционирования системы управления, обусловленным прагматическими свойствами информации:

$$J_{n\beta}(\gamma) = \Pi(\gamma/\beta) - \Pi(\gamma),$$

где: $J_{n\beta}(\gamma)$ – ценность информационного сообщения β для системы управления γ;

$\Pi(\gamma)$ – априорный ожидаемый экономический эффект функционирования системы управления γ;

$\Pi(\gamma/\beta)$ - ожидаемый эффект функционирования системы γ при условии, что для управления будет использована информация, содержащаяся в сообщении β.

Поскольку экономический эффект функционирования АИС складывается из экономического эффекта решения отдельных функциональных задач, то для вычисления J_n следует определить:

Z_β -множество задач, для решения которых используется информация β;

F – частоту решения каждой задачи за период времени, для которого оценивается экономический эффект;

R_β - степень влияния информационного сообщения β на точность решения задачи, $0 \leq R \leq 1$.

Тогда:

$$J_{n\beta}(\gamma) = \Pi(\gamma/\beta) - \Pi(\gamma) = \sum_{j=1}^{Z\beta} F_i R_{\beta j} \Pi_j$$

где Π_j – экономический эффект от решения j-й задачи в системе.

В такой постановке единицей измерения ценности информации АИС является обычно рубль.

Качество информации

Информация в АИС является и предметом труда и продуктом труда, поэтому от ее качества существенно зависят эффективность и качество функционирования системы.

Качество информации можно определить как совокупность свойств, обусловливающих возможность ее использования для удовлетворения определенных в соответствии с ее назначением потребностей.

Возможность и эффективность использования информации для управления обусловливается такими ее потребительскими показателями качества, как репрезентативность, содержательность, полнота, доступность, актуальность, своевременность, устойчивость, точность, достоверность и ценность.

Репрезентативность информации связана с правильностью ее отбора и формирования с целью адекватного отражения заданных свойств объекта. Важнейшее значение здесь имеют: правильность концепции, на базе которой сформулировано исходное понятие; обоснованность отбора существенных признаков и

связей отображаемого явления; правильность методики измерения и алгоритма формирования информации.

Содержательность информации – это ее удельная семантическая емкость, равная отношению количества семантической информации в сообщении к объему данных, его отображающих, т.е. $S = I_c/V_д$. С увеличением содержательности информации растет семантическая пропускная способность информационной системы, так как для получения одних и тех же сведений требуется преобразовать меньший объем данных.

Полнота информации означает, что она содержит минимальный, но достаточный для принятия правильного управленческого решения состав (набор показателей). Как неполная, т.е. недостаточная для принятия правильного решения, так и избыточная информация снижают эффективность управления; наивысшим качеством обладает именно полная информация.

Доступность информации для восприятия при принятии управленческого решения в АИС обеспечивается выполнением соответствующих процедур ее получения и преобразования. Так, назначением автоматизированной системы обработки данных и является увеличение ценности информации путем согласования ее с тезаурусом пользователя, т.е. преобразование ее к доступной и удобной для восприятия управляющими органами форме.

Актуальность определяется степенью сохранения ценности информации для управления в момент ее использования и зависит от статистических характеристик отображаемого объекта (от динамики изменения этих характеристик) и от интервала времени, прошедшего с момента возникновения данной информации.

Своевременность информации: своевременной является такая информация, которая может быть учтена при выработке управленческого решения без нарушения установленной процедуры и регламента, т.е. такая информация, которая поступает на тот или иной уровень управления не позже заранее назначенного момента времени, согласованно со временем решения задачи управления.

Устойчивость есть свойство управляющей информации реагировать на изменения исходных данных, сохраняя необходимую точность. Устойчивость информации, как и ее репрезентативность, обусловлены методической правильностью ее отбора и формирования.

Точность информации определяется степенью близости отображаемого информацией параметра и истинного значения этого параметра.

Для экономических показателей, отображаемых цифровым кодом, известны четыре классификационных понятия точности:

- формальная точность, измеряемая значением единицы младшего разряда числа, которым показатель представлен;

- реальная точность, определяемая значением единицы последнего разряда числа, верность которого гарантируется;

- достижимая точность, - максимальная точность, которую можно получить в данных конкретных условиях функционирования системы;

- необходимая точность, определяемая функциональным назначением показателя.

Достоверность информации – это свойство информации отражать реально существующие объекты с необходимой точностью. Измеряется достоверность информации доверительной вероятностью необходимой точности, т.е. вероятностью того, что отображаемое информацией значение параметра отличается от истинного значения этого параметра в пределах необходимой точности.

Наряду с понятием «достоверность информации» существует понятие «достоверность данных» т.е. информация, рассматриваемой в синтаксическом аспекте. Под достоверностью данных понимается их безошибочность; измеряется она вероятностью появления ошибок в данных.

Недостоверность данных может не повлиять на объем данных, а может и увеличить его в отличии от недостоверности информации, всегда уменьшающей ее количество.

Ценность информации – комплексный показатель ее качества, ее мера на прагматическом уровне.

Исследования показывают, что для целевой функции оптимизация функционирования методически правильно спроектированной информационной системы в качестве ограничений, обусловливаемых параметрами качества информации, достаточно использовать ограничения только по полноте, своевременности и достоверности.

Понятие о базах данных

Термин *база данных* (database) страдает от обилия различных интерпретаций. Он использовался ранее для обозначения чего угодно — от обычной картотеки до многих томов данных, которые правительство собирает о своих гражданах.

Рассмотрим некоторые из них.

Определение 1 База данных – это компьютеризированная система хранения информации, основная цель которой содержать информацию и предоставлять её по требованию.

Определение 2 База данных – это хранение структурированных данных, при этом данные должны быть не противоречивыми, минимально избыточными и целостными.

В настоящее время действует Закон «О правовой охране программ для ЭВМ и баз данных» от 23.09.92 г. В нем дается определение БД:

Определение 3 База данных - это объектная форма представления и организации совокупности данных (например статей, расчетов), систематизированных таким образом, чтобы эти данные могли быть найдены и обработаны с помощью ЭВМ.

Определение 4 База данных – это набор интегрированных записей с самоописанием.

Определение 5 База данных – это организованная в соответствии с определенными правилами и поддерживаемая в памяти компьютера совокупность данных, характеризующая актуальное состояние некоторой предметной области и используемая для удовлетворения информационных потребностей пользователей.

Определение 6 База данных — именованная совокупность данных, отражающая состояние объектов и их отношений в рассматриваемой предметной области.

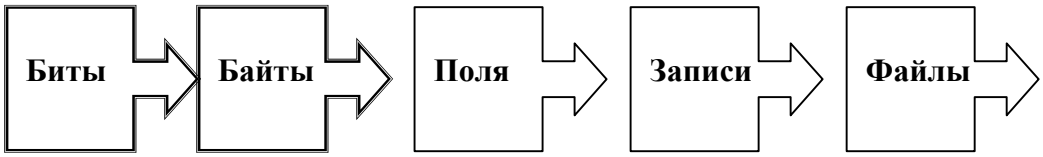
В Толковом словаре по вычислительным системам дается следующее определение: Определение 7 База данных, в обычном, строгом смысле слова — файл данных, для определения и обращения к которому используются средства управления базой данных.

Рассмотрим определение базы данных Дэвида Кренке, который предложил использовать термин база данных в конкретном значении:

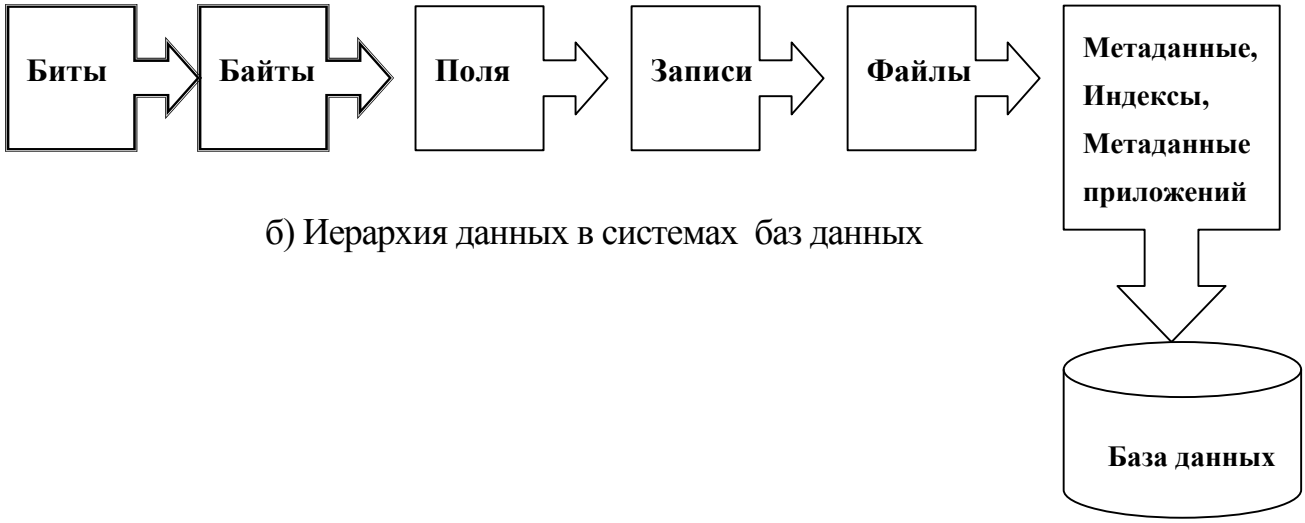
Определение 8 База данных — это самодокументированное собрание интегрированных записей.

Возьмем данное определение за основное, как самое лаконичное и объемное. Обсудим данное определение. База данных является самодокументированной (self-describing) если она содержит, в дополнение к исходным данным пользователя, описание собственной структуры. Это описание называется словарем данных (data dictionary), каталогом данных (data directory) или метаданными (metadata).

Интегрированность записей проявляется в стандартной иерархии данных, которая выглядит следующим образом: биты объединяются в байты, или символы; символы группируются в поля; из полей формируются записи; записи организуются в файлы. Информационная технология баз данных была разработана для того, чтобы преодолеть ограничения, свойственные системам обработки файлов. Чтобы понять, каким образом это было сделано, сравните систему обработки файлов (рисунок 1.10) с системой обработки базы данных (database processing system).



а) Иерархия данных в системах обработки файлов



б) Иерархия данных в системах баз данных

Рисунок 1.10 – Иерархия данных в системах баз данных

Программы обработки файлов обращаются непосредственно к файлам данных. В отличие от них, программы обработки баз данных для доступа к данным вызывают системы управления БД. Это отличие важно тем, что оно упрощает прикладное программирование: программистам больше не нужно задумываться о том, как физически организовано хранение данных, и они могут смело сконцентрироваться на вопросах, представляющих важность для пользователя, а не для компьютерной системы.

В зависимости от вида организации данных различают следующие основные модели представления данных в базе:

- иерархическую;

- сетевую;

- реляционную.

Иерархические модели данных имеют *древовидную* структуру, когда каждому узлу структуры соответствует один *сегмент,* представляющий собой поимено-

ванный линейный кортеж *полей данных.* Каждому сегменту соответствует один *входной* и несколько *выходных* сегментов (рисунок 1.11,*а*). Каждый *сегмент* структуры лежит на единственном иерархическом пути, начинающемся от *корневого* сегмента.

Для описания такой логической организации данных достаточно предусматривать для каждого сегмента данных только идентификацию *входного* для него сегмента. В иерархической модели каждому *входному* сегменту данных соответствует *N выходных.*

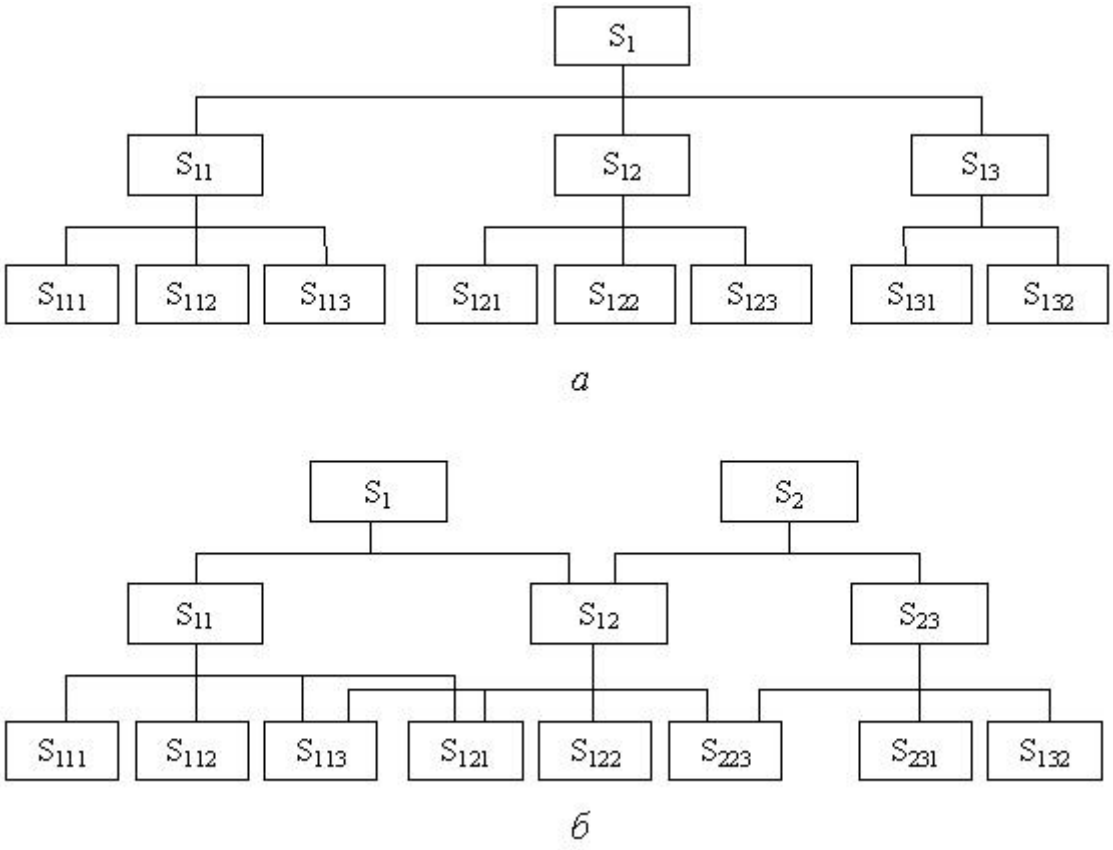


Рисунок 1.11 - Структура иерархической (*а*) и сетевой (*б*) БД

Сетевая модель БД во многом подобна *иерархической*.. Отличие заключается в том, что в *сетевой* модели для сегментов допускается *несколько входных* сегментов наряду с возможностью наличия сегментов без входов с точки зрения иерархической структуры. На рисунке 1.11,*б* представлен простой пример *сетевой* структуры, полученной на основе *модификации* иерархической структуры. Графическое изображение структуры связей сегментов в такого типа моделях представ-

ляет собой *сеть*. Сегменты данных в *сетевых* БД могут иметь множественные связи с сегментами старшего уровня.

Реляционной моделью БД называется модель, в которой средства управления БД поддерживают *реляционную* (табличную) структуру данных. Концепция *реляционной* модели была предложена в 1970 г. Эдгаром Коддом и имеет в настоящее время большое значение в деле организации работы с БД. Данная модель позволяет представить все данные в виде двумерного массива в виде таблиц, соответствующих какому-то определенному объекту или действию, называемым сущностью. Для работы с такой моделью используется математический инструмент – реляционная алгебра, позволяющая манипулировать данными путем склеивания и разрезания таблиц с использованием специального языка манипулирования данными (ЯМД). Итак, основой реляционных баз данных является таблица. Важно запомнить терминологию, которую используют разработчики баз данных.

В таблице 1.2 представлены термины, которые используют различные специалисты по базам данных

Таблица 1.2

Заказчик, пользователь	Системный аналитик, проектировщик	Программист
Таблица	Сущность, отношение	Файл
Столбец	Поле	Атрибут
Строка	Запись	Кортеж
Количество столбцов	Арность	Степень
Количество строк	Мощность	Кардинальное число

Системы управления базами данных (СУБД)

СУБД представляет собой совокупность *лингвистических* и *программных* средств, предназначенных для создания, ведения и совместного использования БД многими пользователями.

В Толковом словаре по вычислительным системам дано следующее понятие СУБД:

Система управления базой данных — система программного обеспечения, имеющая средства обработки на языке базы данных, позволяющая обрабатывать обращения к базе данных, которые поступают от прикладных программ и (или) конечных пользователей, и поддерживать целостность базы данных.

Заключение

Итак, если мы уяснили основные понятия и определения теории информационных систем, можно считать, что цель вводного обучения достигнута.

В следующих лекциях будет раскрываться учебный материал на основе понятий и определений первой лекции.

Вопросы для самоконтроля

1 Что называется данными?
2 Дайте определение и толкования термина «информация».
3 Что называется базами данных?
4 Как определить качество информации?
5 Какие показатели качества информации вы знаете?
6 Что означает тезаурус пользователя информации?
7 Какие модели данных вы знаете?
8 Дайте характеристику реляционных баз данных?
9 Какими терминами пользуются специалисты реляционных баз данных при разработке таблиц, строк, столбцов?
10 Что называется системой управления БД?

2 Лекция 2. Концепция баз данных и автоматизированных информационных систем

Концепция в общем смысле представляет некоторую систему взглядов на процесс или явление. Составными частями концепции являются совокупность принципов и методология. Под методологией понимается совокупность методов решения проблемы.

Принцип — правила, которыми следует руководствоваться в деятельности. Часто принципы формулируются в виде ограничений и требований, в частности, требований к базам данных.

Ключевые слова: базы данных, хранилище данных, информационно-поисковая система, информационно-советующая система, проектирование баз данных, создание и использование баз данных, функционирование баз данных.

Требования, предъявляемы к базам данных

С современных позиций следует порознь рассматривать требования, предъявляемые к транзакционным (операционным) базам данных и к хранилищам данных.

Первоначально перечислим основные требования, которые предъявляются к операционным базам данных, а следовательно, и к СУБД, на которых они строятся.

1 Простота обновления данных. Под операцией обновления понимают добавления, удаления и изменения данных.

2 Высокое быстродействие (малое время отклика на запрос). Время отклика — промежуток времени от момента запроса к БД и фактическим получением данных. Похожим является термин время доступа — промежуток времени между выдачей команды записи (считывания) и

фактическим получением данных. Под доступом пони мается операция поиска, чтения данных или записи их.

3 Независимость данных.

4 Совместное использование данных многими пользователями.

5 Безопасность данных — защита данных от преднамеренного или непреднамеренного нарушения секретности, искажения или разрушения.

6 Стандартизация построения и эксплуатации БД (фактически СУБД).

7 Адекватность отображения данных соответствующей предмет ной области.

8 Дружелюбный интерфейс пользователя.

Важнейшими являются первые два противоречивых требования: повышение быстродействия требует упрощения структуры БД, что, в свою очередь, затрудняет процедуру обновления данных, увеличивает их избыточность.

Независимость данных — возможность изменения логической и физической структуры БД без изменения представлений пользователей. Независимость данных предполагает инвариантность к характеру хранения данных, программному обеспечению и техническим средствам. Она обеспечивает минимальные изменения структуры БД при изменениях стратегии доступа к данным и структуры самих исходных данных. Это достигается, как будет показано далее, «смещением» всех изменений на этапы концептуального и логического проектирования с минимальными изменениями на этапе физического проектирования.

Безопасность данных включает их целостность и защиту. Целостность данных — устойчивость хранимых данных к разрушению и уничтожению, связанных с неисправностями технических средств, системными ошибками и ошибочными действиями пользователей.

Она предполагает:

– отсутствие неточно введенных данных или двух одинаковых

записей об одном и том же факте;

– защиту от ошибок при обновлении БД;

– невозможность удаления порознь (каскадное удаление) связанных данных разных таблиц;

– неискажение данных при работе в многопользовательском ре жиме и в распределенных базах данных;

– сохранность данных при сбоях техники (восстановление данных).

Целостность обеспечивается триггерами целостности — специальными приложениями-программами, работающими при определенных условиях. Для некоторых СУБД (например, Access, Paradox) триггеры являются встроенными.

Защита данных от несанкционированного доступа предполагает ограничение доступа к конфиденциальным данным и может достигаться:

– введением системы паролей;

– получением разрешений от администратора базы данных (АБД);

– запретом от АБД на доступ к данным;

– формированием видов — таблиц, производных от исходных и предназначенных конкретным пользователям.

Три последние процедуры легко выполняются в рамках языка структурированных запросов Structured Query Language — SQL, часто называемом SQL2.

Стандартизация обеспечивает преемственность поколений СУБД, упрощает взаимодействие БД одного поколения СУБД с одинаковыми и различными моделями данных. Стандартизация (ANSI/SPARC) осуществлена в значительной степени в части интерфейса пользователя СУБД и языка SQL. Перейдем к требованиям, предъявляемым к хранилищам данных, которые структурно являются продолжением операционных баз данных.

Поскольку данные в хранилище практически не изменяются, а лишь добавляются, требование простоты обновления становится неактуальным. На первое место — в силу значительного объема данных в хранилище — выходит требование высокого быстродействия.

К хранилищам данных предъявляются следующие дополнительные требования:

– высокая производительность загрузки данных из операционных БД;

– возможность фильтрования, переформатирования, проверки целостности исходных данных, индексирования данных, обновления метаданных;

– повышенные требования к качеству исходных данных в части обеспечения их непротиворечивости, поскольку они могут быть получены из разных источников;

– высокая производительность запросов;

– обеспечение высокой размерности;

– одновременность доступа к ХД;

– наличие средств администрирования.

Э.Ф. Кодд на основе своего опыта предъявил следующие требования к системе OLAP:

– Многомерное концептуальное представление данных.

– Прозрачность технологии и источников данных.

– Доступность к источникам данных при использовании различных моделей данных.

– Неизменная производительность подготовки отчетов при росте объема, количества измерений, процедур обобщения данных.

– Использование гибкой, адаптивной, масштабируемой архитектуры клиент—сервер.

– Универсальность измерений (формулы и средства создания отчетов не должны быть привязаны к конкретным видам размерностей).

– Динамическое управление разреженностью матриц (пустые значения NULL должны храниться эффективным образом).

– Многопользовательская поддержка.

– Неограниченные операционные связи между размерностями.

– Поддержка интуитивно понятных манипуляций с данными.

– Гибкость средств формирования отчетов.

– Неограниченное число измерений и уровней обобщения.

Перечисленные требования отличны от требований к операционным БД, что вызвало появление специализированных БД — хранилищ данных.

Концепция построения баз данных

Концепция предполагает изложение основных положений чего-либо. Описание концепции БД невозможно выполнить без учета хронологии.

1 Инженерные и экономические задачи. Первоначально (начало 60-х г. XXв.) использовалась файловая система хранения. Для решения преимущественно инженерных задач, характеризующихся небольшим количеством данных и значительным объемом вычислений, данные хранились непосредственно в программе. Применялся последовательный способ организации данных, имелась их высокая избыточность, идентичность логической и физической структур и полная зависимость данных.

С появлением экономико-управленческих задач (информационная система руководства — M1S), отличающихся большими объемами данных и малой долей вычислений, указанная организация данных оказалась неэффективной. Требовалось упорядочение данных, которое, как выяснилось, возможно было проводить по двум критериям: использование (информационные массивы); хранение (базы данных).

2 Информационно-поисковые и информационно-советующие системы управления. Следует отметить, что экономические задачи часто связаны с управлением организационными системами. По характеру применения компьютеров такие системы возможно разделить на информационно-поисковые (рисунок 2.1), получившие также название «традиционные», и информационно-советующие или современные (рисунок 2.2) системы. Сначала шло построение и изучение традиционных систем.

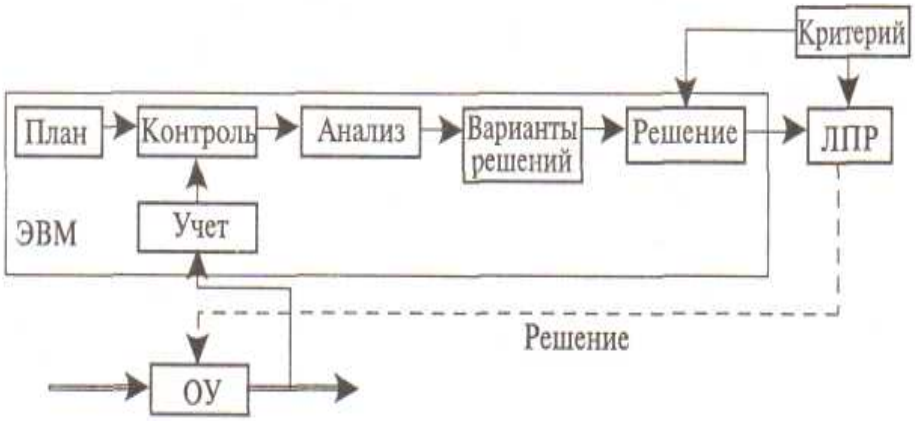


Рисунок 2.1 - Информационно-поисковая система

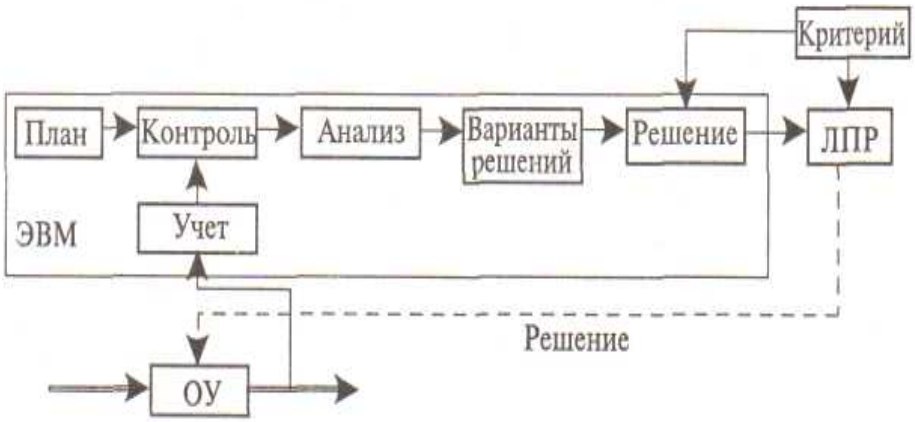


Рисунок 2.2 - Информационно-советующая система

3 Информационные массивы и базы данных. Первоначально в информационно-поисковых системах применяли информационные массивы. При этом возникала необходимость хранения избыточной информации при дефиците компьютерной памяти. Выяснилось так же, что алгоритмы задач более подвижны, чем данные для них. При довольно частом изменении алгоритмов в процессе совершенствования систем управления каждый раз требовалось проводить трудоемкую процедуру создания новых массивов. В этих условиях стало ясно превосходство баз данных, несмотря на их более сложную структуру по сравнению с системой массивов. В дальнейшем базы данных стали снабжаться программной составляющей, позволяющей легко реализовать и оперативно изменять алгоритмы приложения.

4 Модели данных. Использование файлов для хранения только данных (рисунок 2.3,а) предложено МакГри в 1959 г. В 1963 г. С. Бахман построил первую промышленную базу данных IDS с сетевой моделью данных, которая все еще характеризовалась избыточностью данных и ее использованием только для одного приложения. В 1969 г. сформировалась группа, создавшая набор стандартов CODASYL (КОДАСИЛ) для сетевой модели данных. Фактически начала тогда использоваться (рисунок 2.3,б) современная архитектура базы данных. Существенный скачок в развитии технологии баз данных произошел в 1970 г., когда М. Кодд предложил парадигму реляционной модели данных.

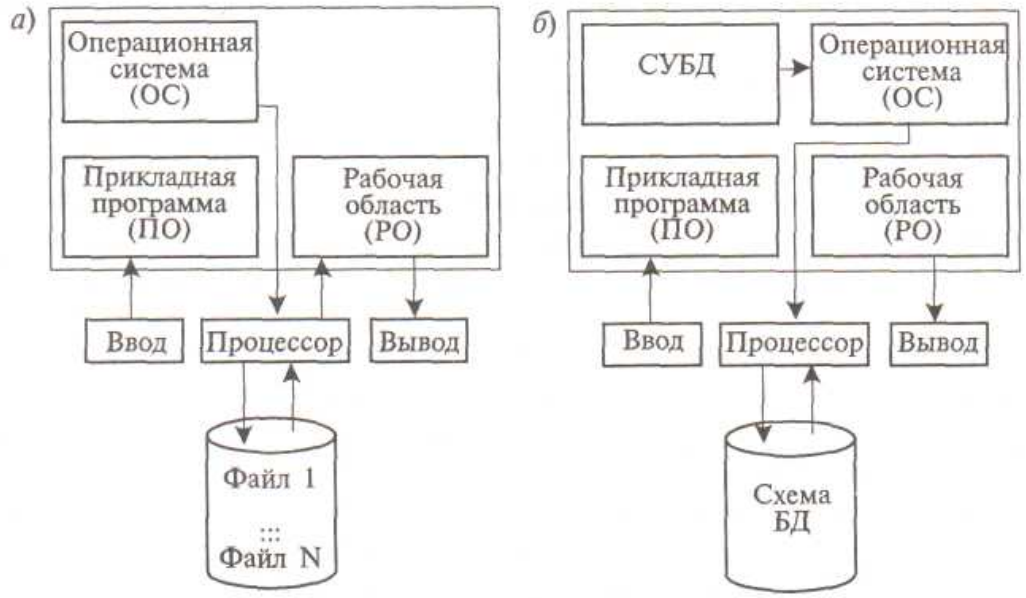


Рисунок 2.3. Файловая система (а) и СУБД (б) для хранения данных

Под парадигмой понимается научная теория, воплощенная в систему понятий, отражающих существенные черты действительности. Теперь логические структуры могли быть получены из одних и тех же физических данных, т. е. доступ к одним и тем же физическим данным мог осуществляться различными приложениями по разным путям. Стало возможным обеспечение целостности и независимости данных. В конце 70-х годов XX в. появились современные СУБД, обеспечивающие физиче-

скую и логическую независимость, безопасность данных, обладающие развитыми языками БД.

В начале 90-х годов реляционные БД получили наиболее широкое распространение, особенно при использовании персональных компьютеров. Появились разнообразные СУБД, рассчитанные как на пользователя-профессионала (в программировании), так и на пользователя-непрофессионала, предназначенные для построения и небольших (по объему памяти), и сверхбольших БД, работающие как в локальном, так и в сетевом режимах. При этом базы данных строились как статические (в зарубежной терминологии — операционные, транзакционные, Online Transactional Processing — OLTP).

К середине 90-х годов в базах данных накопилось такое количество информации, что ее стало возможным использовать для аналитических процедур выработки решений-советов. Появились динамические (аналитические) базы данных, называемые за рубежом Online Analytical Processing — OLAP. Их основными составляющими стали электронный архив и хранилище данных (Data Warehouse). Одновременно выявились недостатки реляционных БД, у которых появились конкуренты в виде объектно-ориентированных баз данных.

Последнее десятилетие характеризуется появлением распределенных и объектно-ориентированных баз данных, характеристики которых определяются приложениями средств автоматизации проектирования и интеллектуализации БД.

5 Подходы к построению БД. Они базируются на двух подходах к созданию автоматизированной системы управления (АСУ). Первый из них связан с автоматизацией документооборота (совокупность документов, движущихся в процессе работы предприятия).

К 90-м годам XX в. сформировался второй, современный подход, связанный с автоматизацией управления. Он предполагает первоначальное выявление стандартных алгоритмов приложений (алгоритмов бизнеса в зарубежной терминологии), под которые определяются данные, а стало быть, и база данных. Объектно-

ориентированное программирование только усилило значимость этого подхода. Состав БД для различных подходов представлен на рисунке 2.4.

Рисунок 2.4 - Схема классического (а) и современного (б) подхода при построении БД

В работе БД возможны одно- и многопользовательский режимы. В последнем случае несколько пользователей подключаются к одному компьютеру через разные порты.

6 Восходящее и нисходящее проектирование БД. Первое применяют в распределенных БД при интеграции спроектированных локальных баз данных, которые могут быть выполнены с использованием различных моделей данных. Более характерным для централизованных БД является нисходящее проектирование.

В последующих лекциях первоначально будет рассмотрен классический подход для централизованных БД, а затем — современный. Распределенным БД посвящены заключительные лекции.

Работа с базами данных может быть представлена в виде схемы, приведенной на рисунке 2.5. Из нее видно, что следует выделять методологию создания, методологию использования и методологию функционирования БД.

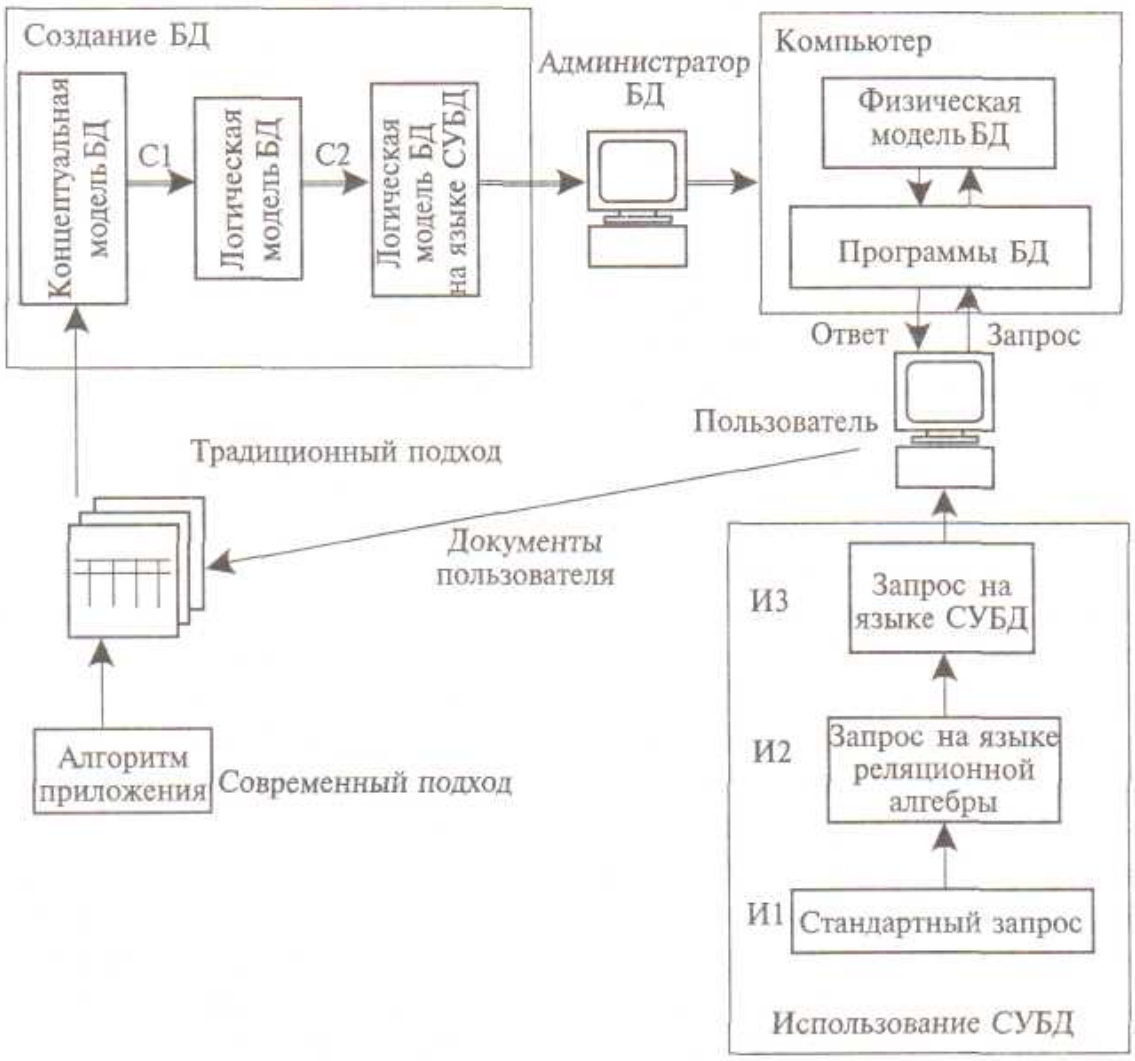


Рисунок 2.5 - Этапы создания (Cl, C2) и использования (И1— ИЗ) БД

Методология БД определяется в процедуре проектирования, но проявляется и в процедуре использования.

7 Хранилище данных — предметно-ориентированный, интегрированный, привязанный ко времени и неизменный набор данных, предназначенный для поддержки принятия решений. В соответствии с определением хранилище данных ориентировано не на алгоритм приложения, как (операционная) БД, а на предметную область.

Интегрированность определяется тем фактом, что источниками данных могут быть несколько БД, которые могут иметь разные форматы данных и степень заполнения БД. Эти данные должны быть приведены к «стандарту», используемому в

ХД. Привязка ко времени означает, что исходные данные характеризуют какой-то интервал времени, при этом время присутствует в БД явно. В силу этого вновь поступающие данные не изменяют прежние данные в ХД, а дополняют их.

Методология проектирования баз данных

Существует много разновидностей методологии рассмотрения баз данных в классическом подходе, однако чаще всего придерживаются методологии ANSI/SPARC, схема которой представлена на рисунке 2.6. Совокупность процедур проектирования централизованной БД можно разделить на четыре этапа.

На этапе формулирования и анализа требований устанавливаются цели организации, определяются требования к БД. Для формирования специфических требований обычно используется методика интервьюирования персонала различных уровней управления. Все требования документируются в форме, доступной конечному пользователю и проектировщику БД.

Этап концептуального проектирования заключается в описании и синтезе информационных требований пользователей в первоначальный проект БД. Исходными данными могут быть совокупность документов пользователя при классическом подходе или алгоритмы приложений (алгоритмы бизнеса) при современном подходе. Результатом этого этапа является высокоуровневое представление (в виде системы таблиц БД) информационных требований пользователей на основе различных подходов. Сначала выбирается модель БД. Затем с помощью ЯОД создается структура БД, которая затем заполняется данными с помощью команд ЯМД, систем меню, экранных форм или в режиме просмотра таблиц БД. Здесь же обеспечивается защита и целостность (в том числе — ссылочная) данных с помощью СУБД или путем построения триггеров. В процессе логического проектирования высокоуровневое представление данных преобразуется в структуру используемой СУБД. Основной целью этапа является устранение избыточности данных с использованием специальных правил — нормализации.

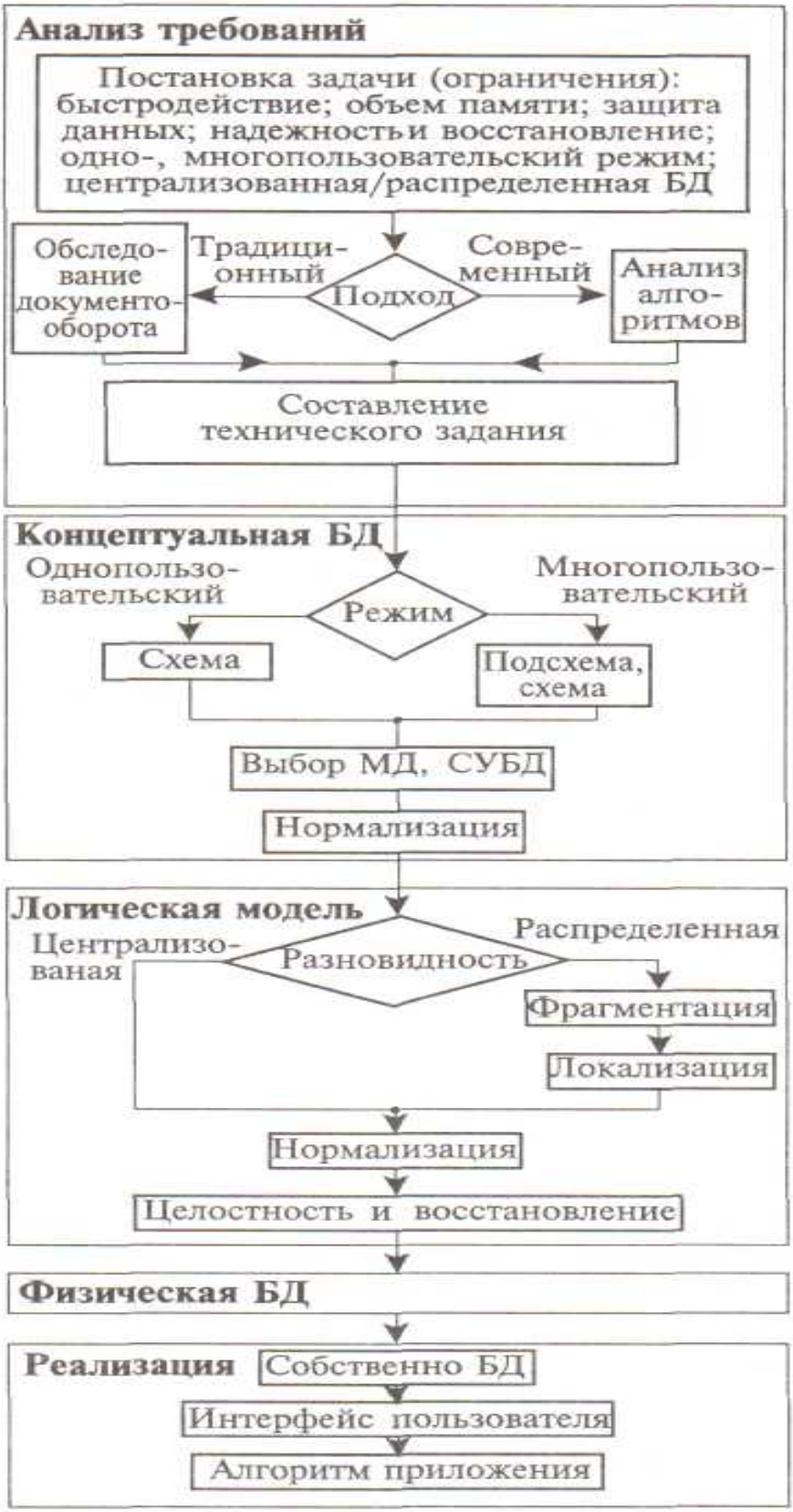


Рисунок 2.6 - Этапы проектирования операционных БД

На этапе физического проектирования решаются вопросы, связанные с производительностью системы, определяются структуры хранения данных и методы

доступа. Взаимодействие между этапами проектирования и словарной системой необходимо рассматривать отдельно. Процедуры проектирования могут использоваться независимо в случае отсутствия словарной системы. Сама словарная система может рассматриваться как элемент автоматизации проектирования. Средства проектирования и оценочные критерии используются на всех стадиях разработки. В настоящее время неопределенность при выборе критериев является наиболее слабым местом в проектировании БД. Это связано с трудностью описания и идентификации большого числа альтернативных решений. В то же время существует много критериев оптимальности, являющихся неизмеримыми свойствами, трудно выразимыми в количественном представлении или в виде целевой функции.

К качественным критериям могут относиться гибкость, адаптивность, доступность для новых пользователей, совместимость с другими системами, возможность конвертирования в другую вычислительную среду, возможность восстановления данных, возможность распределения и расширения.

Проще обстоит дело при работе с количественными критериями, к которым относятся время ответа на запрос, стоимость модификации, стоимость памяти, время на создание, стоимость на реорганизацию. Затруднение может вызывать противоречие критериев друг другу.

Методология использования баз данных

БД используются обычно не самостоятельно, а являются компонентой различных информационных систем: банков данных, информационно-поисковых и экспертных систем, систем автоматизированного проектирования, автоматизированных рабочих мест, автоматизированных систем управления.

В БД имеется три уровня представления данных: концептуальная, логическая и физическая базы данных. В процедуре использования чаще всего имеют дело с логической и значительно реже с концептуальной и физической моделями.

Словарь данных представляет собой как бы внутреннюю БД, содержащую централизованные сведения о всех типах данных, их имена, структуру, а также информацию об их использовании. Преимущество словаря данных — в эффективном накоплении и управлении информационными ресурсами предметной области. Его применение позволяет уменьшить избыточность и противоречивость данных при их вводе, осуществить простое и эффективное управление при их модификации, упростить процедуру проектирования БД за счет централизации управления данными, установить связи с другими пользователями. Таким образом, словарь данных содержит обобщенное представление всех трех уровней: концептуального, логического и физического.

В логическом представлении применяются следующие виды моделей данных: иерархические, сетевые, реляционные, объектно-ориентированные (объектно-реляционные). Иерархическая модель служит разновидностью сетевой, являющейся совокупностью деревьев (лесом). Сетевая модель допускает только бинарные связи «многие к одному» и использует для описания модель ориентированных графов. Реляционная модель использует представление данных в виде таблиц (реляций, связей). В ее основе лежит математическое понятие теоретико-множественного отношения: она базируется на реляционной алгебре и теории отношений. В объектно-ориентированной модели используются понятия класса, объекта, метода.

В процессе использования БД имеются операции обновления (запись, удаление, модификация данных) и запрос-ответ (чтение).

В общем случае процесс запроса состоит из ряда этапов (И1—ИЗ на рисунке 2.5). Пользователь должен знать структуру БД или обратиться к АБД.

На этапе И1 пользователь должен выяснить, какие формы документов ему нужны. Это могут быть не только логические модели пользователя, но и различные их модификации при разных сочетаниях полей. Поскольку логические (а тем более модифицированные логические) модели могут отличаться от логической модели

БД, следует определить, какие сочетания полей необходимы для выводимых машинных документов.

Эти сочетания образуются с помощью элементарных правил (этап И2), изучаемых реляционной алгеброй и реляционным исчислением. Далее правила следует трансформировать в соответствующие варианты обращения к СУБД через ее интерфейс. Это могут быть меню, экранные формы, язык программирования (например, SQL), запрос по примеру, режим просмотра таблиц БД. Результат может быть представлен в виде таблиц или отчетов.

При эксплуатации БД используют и две специфические операции: навигацию и спецификацию.

Для работы с БД используется специальный обобщенный инструментарий в виде СУБД, предназначенный для управления БД и обеспечения интерфейса пользователя. Существует два основных направления реализации СУБД: программное и аппаратное.

Программная реализация (в дальнейшем СУБД) представляет собой набор программных модулей, работает под управлением конкретной ОС и выполняет следующие функции: описание данных на концептуальном и логическом уровнях; загрузку данных; хранение данных; поиск и ответ на запрос (транзакцию); внесение изменений; обеспечение безопасности и целостности; предоставление пользователю языковых средств: языка описания данных (ЯОД), языка манипулирования данными (ЯМД), языка запросов.

Методология функционирования баз данных

Речь пойдет прежде всего о функционировании операционных БД в рамках СУБД. Независимо от класса БД здесь приходится решать проблемы, к которым относятся обеспечение одно- и многопользовательского функционирования, защита данных, обеспечение целостности, восстановление данных после сбоя в БД. В централизованных однопользовательских БД функционирование обеспечивается

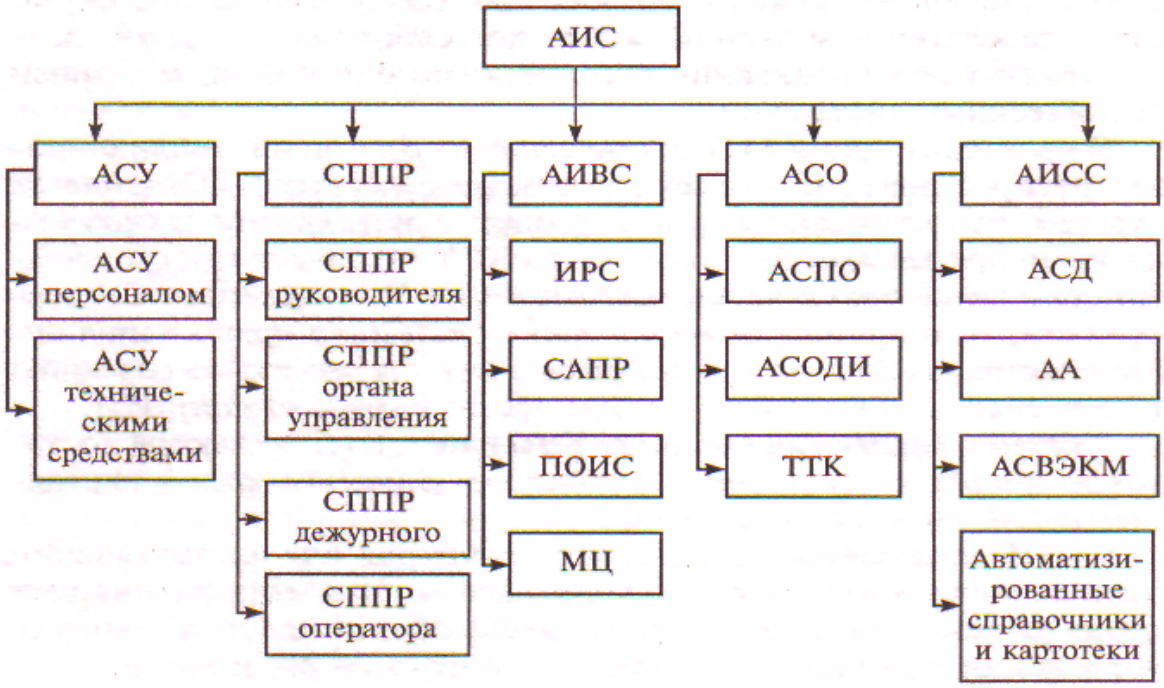
так называемыми транзакциями, в результате выполнения которых данные в БД либо обновляются (фиксация), либо остаются прежними (откат). Для операционных БД (СУБД) характерны так называемые короткие транзакции с длительностью в микро- и миллисекунды. Хранилища данных должны работать при длительных (часы) транзакциях. В случае многопользовательского режима дополнительно возникает необходимость одновременного доступа нескольких пользователей к одним и тем же данным, что чаще всего достигается блокировкой данных.

Защита данных от несанкционированного доступа осуществляется либо запретом доступа (пароль), либо разрешением на доступ, что особенно легко обеспечить с помощью языка программирования SQL. Обеспечение целостности определяется специальными программами, получившими название триггеры. Они реализуют различного рода ограничения. Например, для поля Пол программа ограничивает задание только значений «муж» и «жен». Другие значения базой данных не воспринимаются. Решение названных проблем несколько упрощается при использовании в распределенных БД режима клиент—сервер.

Концепция автоматизированных информационных систем

В качестве основного классификационного признака АИС целесообразно рассматривать особенности автоматизируемой профессиональной деятельности — процесса переработки входной информации для получения требуемой выходной информации, в котором АИС выступает в качестве инструмента должностного лица или группы должностных лиц, участвующих в управлении организационной системой.

В соответствии с предложенным классификационным признаком можно выделить следующие классы АИС (рисунок 2.6):



– автоматизированные системы управления (АСУ);
– системы поддержки принятия решения (СППР);
– автоматизированные информационно-вычислительные системы (АИ ВС);
– автоматизированные системы обучения (АСО);
– автоматизированные информационно-справочные системы (АИСС).

Рисунок 2.6

Рассмотрим особенности каждого класса АИС и характеристики возможных видов АИС в составе каждого класса.

Автоматизированные системы управления. АСУ представляет собой АИС, предназначенную для автоматизации всех или большинства задач управления, решаемых коллективным органом управления (министерством, дирекцией, правлением, службой, группой управления и т.д.). В зависимости от объекта управления различают АСУ персоналом и АСУ техническими средствами. АСУ является организационной и технической основой реализации рациональной технологии коллективного решения задач управления в различных условиях обстановки. В этой связи

разработка рациональной технологии организационного управления является определяющим этапом создания любой АСУ.

АСУ персоналом обеспечивает автоматизированную переработку информации, необходимой для управления организацией в повседневной деятельности, а также при подготовке и реализации программ развития.

АСУ техническими средствами предназначены для реализации соответствующих технологических процессов. Они являются, по сути, передаточным звеном между должностными лицами, осуществляющими управление техническими системами, и самими техническими системами.

В настоящее время АСУ техническими средствами нашли широкое распространение во всех развитых государствах. Объясняется это тем, что управление существующими новейшими технологическими процессами без применения АСУ техническими средствами становится практически невозможным. Что касается АСУ персоналом, то в настоящее время такие системы широко используются в странах Запада, и непрерывно ведутся работы по созданию новых систем, в том числе на базе достижений в области ИИ.

Системы поддержки принятия решений. СППР являются достаточно новым классом АИС, теория создания которых в настоящее время интенсивно развивается.

СППР называется АИС, предназначенная для автоматизации деятельности конкретных должностных лиц при выполнении ими своих должностных (функциональных) обязанностей в процессе управления персоналом и (или) техническими средствами.

Выделяются четыре категории должностных лиц, деятельность которых отличается различной спецификой переработки информации: руководитель, должностное лицо аппарата управления, оперативный дежурный, оператор. В соответствии с четырьмя категориями должностных лиц различают и четыре вида СППР: СППР руководителя, СППР органа управления, СППР дежурного и СППР оператора.

Автоматизированные информационно-вычислительные системы. АИВС предназначены для решения сложных в математическом отношении задач, требующих больших объемов самой разнообразной информации. Таким образом, видом деятельности, автоматизируемым АИВС, является проведение различных (сложных и «объемных») расчетов. Эти системы используются для обеспечения научных исследований и разработок, а также как подсистемы АСУ и СППР в тех случаях, когда выработка управленческих решений должна опираться на сложные вычисления.

В зависимости от специфики области деятельности, в которой используются АИБС, различают следующие виды этих систем.

Информационно-расчетные системы (ИРС). Эти системы предназначены для обеспечения оперативных расчетов и автоматизации обмена информацией между рабочими местами в пределах некоторой организации или системы организаций. ИРС обычно сопрягаются с АСУ и в рамках последней могут рассматриваться как подсистема.

Системы автоматизации проектирования (САПР). Эти системы предназначены для автоматизации деятельности подразделений проектной организации или коллектива специалистов в процессе разработки проектов изделий на основе применения единой информационной базы, математических и графических моделей, автоматизированных проектных и конструкторских процедур. САПР является одной из систем интегральной автоматизации производства, обеспечивающих реализацию автоматизированного цикла создания нового изделия от предпроектных научных исследований до выпуска серийного образца.

В области экономики САПР могут использоваться при проектировании экономических информационных систем и их элементов. Кроме того, технология САПР может обеспечить создание автоматизированной системы отображения обстановки на экране в процессе ведения экономических операций или в ходе деловых игр различных типов.

Проблемно-ориентированные имитационные системы (ПОИС). Эти системы предназначены для автоматизации разработки имитационных моделей в некоторой предметной области. Например, если в качестве предметной области взять развитие автомобилестроения, то любая модель, создаваемая в этой предметной области, может включать стандартные блоки, моделирующие деятельность предприятий, поставляющих комплектующие; собственно сборочные производства; сбыт, обслуживание и ремонт автомобилей; рекламу и др. Эти стандартные блоки могут строиться с различной детализацией моделируемых процессов и различной оперативностью расчетов. Пользователь, работая с ПОИС, сообщает ей, какая модель ему нужна (т.е. что необходимо учесть при моделировании и с какой степенью точности), а ПОИС автоматически формирует имитационную модель, необходимую пользователю.

В состав программного обеспечения ПОИС входят блоки типовых моделей предметных областей, планировщик моделей, БД предметных областей, а также средства диалогового общения пользователя с ПОИС.

ПОИС является достаточно сложной АИС, реализуемой, как правило, с использованием технологии ИИ на высокопроизводительных ЭВМ.

Моделирующие центры (МЦ) - это АИС, представляющая собой комплекс готовых к использованию моделей, объединенных единой предметной областью, информационной базой и языком общения с пользователями. МЦ, как и ПОИС, предназначены для обеспечения проведения исследований на различных моделях.

Автоматизированные системы обучения. Традиционные методы обучения специалистов в различных областях профессиональной деятельности складывались многими десятилетиями, в течение которых накоплен большой опыт. Однако, как свидетельствуют многочисленные исследования, традиционные методы обучения обладают рядом недостатков. К таким недостаткам следует отнести пассивный характер устного изложения, трудность организации активной работы студентов, невозможность учета в полной мере индивидуальных особенностей отдельных обучаемых и т.д.

Одним из возможных путей преодоления этих трудностей является создание АСО — АИС, предназначенных для автоматизации подготовки специалистов с участием или без участия преподавателя и обеспечивающих обучение, подготовку учебных курсов, управление процессом обучения и оценку его результатов. Основными видами АСО являются автоматизированные системы программированного обучения (АСПО), системы обучения деловым играм (АСОДИ), тренажеры и тренажерные классы (ТТК).

АСПО ориентированы на обучение в основном по теоретическим разделам курсов и дисциплин. В рамках АСПО реализуются заранее подготовленные квалифицированными преподавателями компьютерные курсы. При этом учебный материал разделяется на порции (дозы) и для каждой порции материала указывается возможная реакция обучаемого. В зависимости от действий обучаемого и его ответов на поставленные вопросы АСПО формирует очередную дозу представляемой информации.

Наибольшую сложность при создании АСПО составляет разработка компьютерного курса для конкретной дисциплины. Именно поэтому в настоящее время наибольшее распространение получили «компьютерные курсы» по традиционным, отработанным в методическом плане дисциплинам (физике, элементарной математике, программированию и т.д.).

АСОДИ предназначена для подготовки и проведения деловых игр, сущность которых заключается в имитации принятия должностными липами индивидуальных и групповых решений в различных проблемных ситуациях путем игры по заданным правилам.

ТТК предназначены для обучения практическим навыкам работы на конкретных рабочих местах (постах). Они являются средствами индивидуального (тренажеры) и группового (тренажерные комплексы) обучения.

Автоматизированные информационно-справочные системы (АИСС). Это АИС, предназначенная для сбора, хранения, поиска и выдачи в требуемом виде потребителям информации справочного характера.

В зависимости от характера работы с информацией различают следующие виды АИСС:

– автоматизированные архивы (АА);

– автоматизированные системы делопроизводства (АСД);

– автоматизированные справочники и картотеки;

– автоматизированные системы ведения электронных карт местности (АСВЭКМ) и др.

В настоящее время разработано большое количество разновидностей АИСС, и их количество продолжает увеличиваться. АИСС создаются с использованием технологии БД, достаточно хорошо разработанной и получившей широкое распространение. Для создания АИСС, как правило, не требуется высокопроизводительной вычислительной техники.

Простота создания АИСС и высокий положительный эффект от их использования определили их активное использование во всех сферах профессиональной (в том числе и управленческой) деятельности.

Функции, выполняемые АИС

Анализ содержания и систематизация основных понятий АИС позволяют выделить и определить ее обобщенные функции:

1) вычислительную, своевременно и качественно выполняющую обработку информации во всех видах интересующих систему управления аспектах;

2) следящую, отслеживающую и формирующую всю необходимую для управления внешнюю и внутреннюю информацию;

3) запоминающую, обеспечивающую непрерывное накопление, систематизацию, хранение и обновление необходимой информации;

4) коммуникационную, обеспечивающую передачу нужной информации между необходимыми пунктами;

5) информирующую, реализующую быстрый доступ, поиск и выдачу необходимой информации всех видов: научной, технической, экономической и пр.;

6) регулирующую осуществляющую информационно-управляющее воздействие на объект управления и его звенья при отклонении параметров их функционирования от заданных (запланированных)значений;

7) оптимизирующую, обеспечивающую оптимальные плановые расчеты и перерасчеты по мере измерения целей, критериев и условия функционирования системы;

8) самоорганизующуюся, гибко изменяющую свою структуру и параметры для достижения вновь поставленных целей (в том числе для реализации цикла «исследование – разработка – внедрение – производство» с минимальными затратами ресурсов);

9) самосовершенствующуюся, накапливающую и анализирующую опыт с целью обоснованного отбора лучших методов проектирования, производства и управления, организационных форм осуществления мероприятий;

10) исследовательскую, обеспечивающую выполнение научных исследований производственных проблем, процессы создания новых видов техники и технологии, а также формирование тематики целевых программ комплексных научных исследований;

11) прогнозирующую, определяющую основные тенденции, закономерности и показатели развития производственных комплексов и окружающей среды;

12) анализирующую, определяющую основные показатели технико-экономического уровня производства;

13) синтезирующую, обеспечивающую автоматизированную разработку, проектирование и совершенствование конструкторско-технологических решений и сводных нормативов;

14) контролирующую, выполняющую автоматизированный контроль качества средств производства и выпускаемой продукции;

15) диагностирующую, обеспечивающую автоматизированный процесс диагностики состояния объектов управления (в первую очередь технологического оборудования) и выявления ослабленных и ненадежных звеньев в объектах;

16) документирующую, обеспечивающую формирование всех необходимых учетно-отчетных, планово-распорядительных, конструкторско-технологических и других форм документов.

Анализ функций, выполняемых АИС, показателей качества и содержания всех видов преобразуемой в производственном комплексе информации позволяет определить основные факторы, совершенствование которых существенно зависит от качества информации.

Вопросы для самоконтроля

1 Назовите требования, предъявляемые к операционным БД; к хранилищам данных.

2 Что такое независимость, безопасность, целостность, защита данных?

3 Как обеспечиваются целостность и независимость данных?

4 Что такое «модель данных» (МД)? Назовите виды МД.

5 Что такое концепция? методология?

6 Расскажите историю развития технологии баз данных.

7 Назовите варианты СУБД.

8 Дайте схематическое представление классического и современного подходов к построению БД.

9 Опишите этапы проектирования централизованной, транзакционной базы данных, хранилища данных.

10 Что такое «хранилище данных»?

11 Классификация АИС.

12 Функции, выполняемые АИС.

3 Лекция 3. Каноническое и типовое проектирование РИС

Рассмотрим назначение и содержание канонического и типового проектирования по следующим вопросам.

Стадии и этапы процесса канонического проектирования ИС. Цели и задачи предпроектной стадии создания ИС. Модели деятельности организации ("как есть" и "как должно быть"). Состав работ на стадии технического и рабочего проектирования. Состав проектной документации. Типовое проектирование ИС. Понятие типового проекта, предпосылки типизации. Объекты типизации. Методы типового проектирования. Оценка эффективности использования типовых решений. Типовое проектное решение (ТПР). Классы и структура ТПР. Состав и содержание операций типового элементного проектирования ИС. Функциональные пакеты прикладных программ (ППП) как основа ТПР. Адаптация типовой ИС. Методы и средства прототипного проектирования ИС.

Каноническое проектирование ИС

Организация канонического проектирования ИС ориентирована на использование главным образом каскадной модели жизненного цикла ИС. Стадии и этапы работы описаны в стандарте ГОСТ 34.601-90.

В зависимости от сложности объекта автоматизации и набора задач, требующих решения при создании конкретной ИС, стадии и этапы работ могут иметь различную трудоемкость. Допускается объединять последовательные этапы и даже исключать некоторые из них на любой стадии проекта. Допускается также начинать выполнение работ следующей стадии до окончания предыдущей.

Стадии и этапы создания ИС, выполняемые организациями-участниками, прописываются в договорах и технических заданиях на выполнение работ.

Стадия 1. Формирование требований к ИС.

На начальной стадии проектирования выделяют следующие этапы работ:

– обследование объекта и обоснование необходимости создания ИС;

– формирование требований пользователей к ИС;

– оформление отчета о выполненной работе и тактико-технического задания на разработку.

Стадия 2. Разработка концепции ИС.

– изучение объекта автоматизации;

– проведение необходимых научно-исследовательских работ;

– разработка вариантов концепции ИС, удовлетворяющих требованиям пользователей;

– оформление отчета и утверждение концепции.

Стадия 3. Техническое задание.

– разработка и утверждение технического задания на создание ИС.

Стадия 4. Эскизный проект.

– разработка предварительных проектных решений по системе и ее частям;

– разработка эскизной документации на ИС и ее части.

Стадия 5. Технический проект.

– разработка проектных решений по системе и ее частям;

– разработка документации на ИС и ее части;

– разработка и оформление документации на поставку комплектующих изделий;

– разработка заданий на проектирование в смежных частях проекта.

Стадия 6. Рабочая документация.

– разработка рабочей документации на ИС и ее части;

– разработка и адаптация программ.

Стадия 7. Ввод в действие.

– подготовка объекта автоматизации;

– подготовка персонала;

– комплектация ИС поставляемыми изделиями (программными и техническими средствами, программно-техническими комплексами, информационными изделиями);

– строительно-монтажные работы;

– пусконаладочные работы;

– проведение предварительных испытаний;

– проведение опытной эксплуатации;

– проведение приемочных испытаний.

Стадия 8. Сопровождение ИС.

– выполнение работ в соответствии с гарантийными обязательствами;

– послегарантийное обслуживание.

Обследование - это изучение и диагностический анализ организационной структуры предприятия, его деятельности и существующей системы обработки информации. Материалы, полученные в результате обследования, используются для:

– обоснования разработки и поэтапного внедрения систем;

– составления технического задания на разработку систем;

– разработки технического и рабочего проектов систем.

На этапе обследования целесообразно выделить две составляющие: определение стратегии внедрения ИС и детальный анализ деятельности организации.

Основная задача первого этапа обследования - оценка реального объема проекта, его целей и задач на основе выявленных функций и информационных элементов автоматизируемого объекта высокого уровня. Эти задачи могут быть реализованы или заказчиком ИС самостоятельно, или с привлечением консалтинговых организаций. Этап предполагает тесное взаимодействие с основными потенциальными пользователями системы и бизнес-экспертами. Основная задача взаимодействия - получить полное и однозначное понимание требований заказчика. Как правило, нужная информация может быть получена в результате интервью, бесед или семинаров с руководством, экспертами и пользователями.

По завершении этой стадии обследования появляется возможность определить вероятные технические подходы к созданию системы и оценить затраты на ее реализацию (затраты на аппаратное обеспечение, закупаемое программное обеспечение и разработку нового программного обеспечения).

Результатом этапа определения стратегии является документ (технико-экономическое обоснование проекта), где четко сформулировано, что получит заказчик, если согласится финансировать проект, когда он получит готовый продукт (график выполнения работ) и сколько это будет стоить (для крупных проектов должен быть составлен график финансирования на разных этапах работ). В документе желательно отразить не только затраты, но и выгоду проекта, например время окупаемости проекта, ожидаемый экономический эффект (если его удается оценить).

Ориентировочное содержание этого документа:

– ограничения, риски, критические факторы, которые могут повлиять на успешность проекта;

– совокупность условий, при которых предполагается эксплуатировать будущую систему: архитектура системы, аппаратные и программные ресурсы, условия функционирования, обслуживающий персонал и пользователи системы;

– сроки завершения отдельных этапов, форма приемки/сдачи работ, привлекаемые ресурсы, меры по защите информации;

– описание выполняемых системой функций;

– возможности развития системы;

– информационные объекты системы;

– интерфейсы и распределение функций между человеком и системой;

– требования к программным и информационным компонентам ПО, требования к СУБД;

– что не будет реализовано в рамках проекта.

На этапе детального анализа деятельности организации изучаются задачи, обеспечивающие реализацию функций управления, организационная структура,

штаты и содержание работ по управлению предприятием, а также характер подчиненности вышестоящим органам управления. На этом этапе должны быть выявлены:

– инструктивно-методические и директивные материалы, на основании которых определяются состав подсистем и перечень задач;

– возможности применения новых методов решения задач.

Аналитики собирают и фиксируют информацию в двух взаимосвязанных формах:

– функции - информация о событиях и процессах, которые происходят в бизнесе;

– сущности - информация о вещах, имеющих значение для организации и о которых что-то известно.

При изучении каждой функциональной задачи управления определяются:

– наименование задачи; сроки и периодичность ее решения;

– степень формализуемости задачи;

– источники информации, необходимые для решения задачи;

– показатели и их количественные характеристики;

– порядок корректировки информации;

– действующие алгоритмы расчета показателей и возможные методы контроля;

– действующие средства сбора, передачи и обработки информации;

– действующие средства связи;

– принятая точность решения задачи;

– трудоемкость решения задачи;

– действующие формы представления исходных данных и результатов их обработки в виде документов;

– потребители результатной информации по задаче.

Одной из наиболее трудоемких, хотя и хорошо формализуемых задач этого этапа является описание документооборота организации. При обследовании доку-

ментооборота составляется схема маршрута движения документов, которая должна отразить:

- количество документов;
- место формирования показателей документа;
- взаимосвязь документов при их формировании;
- маршрут и длительность движения документа;
- место использования и хранения данного документа;
- внутренние и внешние информационные связи;
- объем документа в знаках.

По результатам обследования устанавливается перечень задач управления, решение которых целесообразно автоматизировать, и очередность их разработки.

На этапе обследования следует классифицировать планируемые функции системы по степени важности. Один из возможных форматов представления такой классификации - MuSCoW.

Эта аббревиатура расшифровывается так: Must have - необходимые функции; Should have - желательные функции; Could have - возможные функции; Won't have - отсутствующие функции.

Функции первой категории обеспечивают критичные для успешной работы системы возможности.

Реализация функций второй и третьей категорий ограничивается временными и финансовыми рамками: разрабатывается то, что необходимо, а также максимально возможное в порядке приоритета число функций второй и третьей категорий.

Последняя категория функций особенно важна, поскольку необходимо четко представлять границы проекта и набор функций, которые будут отсутствовать в системе.

Модели деятельности организации создаются в двух видах:

- модель "как есть"("as-is")- отражает существующие в организации бизнес-процессы;

– модель "как должно быть"("to-be") - отражает необходимые изменения бизнес-процессов с учетом внедрения ИС.

На этапе анализа необходимо привлекать к работе группы тестирования для решения следующих задач:

– получения сравнительных характеристик предполагаемых к использованию аппаратных платформ, операционных систем, СУБД, иного окружения;

– разработки плана работ по обеспечению надежности информационной системы и ее тестирования.

Привлечение тестировщиков на ранних этапах разработки является целесообразным для любых проектов. Если проектное решение оказалось неудачным и это обнаружено слишком поздно (на этапе разработки или, что еще хуже, на этапе внедрения в эксплуатацию), то исправление ошибки проектирования обходится очень дорого. Чем раньше группы тестирования выявляют ошибки в информационной системе, тем ниже стоимость сопровождения системы. Время на тестирование системы и на исправление обнаруженных ошибок следует предусматривать не только на этапе разработки, но и на этапе проектирования.

Для автоматизации тестирования следует использовать системы отслеживания ошибок (bug tracking). Это позволяет иметь единое хранилище ошибок, отслеживать их повторное появление, контролировать скорость и эффективность исправления ошибок, видеть наиболее нестабильные компоненты системы, а также поддерживать связь между группой разработчиков и группой тестирования (уведомления об изменениях по e-mail и т.п.). Чем больше проект, тем сильнее потребность в bug tracking.

Результаты обследования представляют объективную основу для формирования технического задания на информационную систему.

Техническое задание- это документ, определяющий цели, требования и основные исходные данные, необходимые для разработки автоматизированной системы управления.

При разработке технического задания необходимо решить следующие задачи:

- установить общую цель создания ИС, определить состав подсистем и функциональных задач;
- разработать и обосновать требования, предъявляемые к подсистемам;
- разработать и обосновать требования, предъявляемые к информационной базе, математическому и программному обеспечению, комплексу технических средств (включая средства связи и передачи данных);
- установить общие требования к проектируемой системе;
- определить перечень задач создания системы и исполнителей;
- определить этапы создания системы и сроки их выполнения;
- провести предварительный расчет затрат на создание системы и определить уровень экономической эффективности ее внедрения.

Типовые требования к составу и содержанию технического задания приведены в таблице 3.1

Таблица 3.1. Состав и содержание технического задания (ГОСТ 34.602- 89)

№ раздела	Раздел	Содержание
1	2	3
1	Общие сведения	- полное наименование системы и ее условное обозначение - шифр темы или шифр (номер) договора; - наименование предприятий разработчика и заказчика системы, их реквизиты - перечень документов, на основании которых создается ИС - плановые сроки начала и окончания работ - сведения об источниках и порядке финансирования работ - порядок оформления и предъявления заказчику результатов работ по созданию системы, ее частей и отдельных средств

Продолжение таблицы 3.1

1	2	3
2	Назначение и цели создания (развития) системы	– вид автоматизируемой деятельности – перечень объектов, на которых предполагается использование системы – наименования и требуемые значения технических, технологических, производственно-экономических и др. показателей объекта, которые должны быть достигнуты при внедрении ИС
3	Характеристика объектов автоматизации	– краткие сведения об объекте автоматизации – сведения об условиях эксплуатации и характеристиках окружающей среды
4	Требования к системе	Требования к системе в целом: – требования к структуре и функционированию системы (перечень подсистем, уровни иерархии, степень централизации, способы информационного обмена, режимы функционирования, взаимодействие со смежными системами, перспективы развития системы) – требования к персоналу (численность пользователей, квалификация, режим работы, порядок подготовки) – показатели назначения (степень приспособляемости системы к изменениям процессов управления и значений параметров)

Продолжение таблицы 3.1

1	2	3
		– требования к надежности, безопасности, эргономике, транспортабельности, эксплуатации, техническому обслуживанию и ремонту, защите и сохранности информации, защите от внешних воздействий, к патентной чистоте, по стандартизации и унификации Требования к функциям (по подсистемам): – перечень подлежащих автоматизации задач – временной регламент реализации каждой функции – требования к качеству реализации каждой функции, к форме представления выходной информации, характеристики точности, достоверности выдачи результатов – перечень и критерии отказов Требования к видам обеспечения: – математическому (состав и область применения мат. моделей и методов, типовых и разрабатываемых алгоритмов) – информационному (состав, структура и организация данных, обмен данными между компонентами системы, информационная совместимость со смежными системами, используемые классификаторы, СУБД, контроль данных и ведение информационных массивов, процедуры придания юридической силы выходным документам)

Продолжение таблицы 3.1

1	2	3
		– лингвистическому (языки программирования, языки взаимодействия пользователей с системой, системы кодирования, языки ввода- вывода) – программному (независимость программных средств от платформы, качество программных средств и способы его контроля, использование фондов алгоритмов и программ) – техническому – метрологическому – организационному (структура и функции эксплуатирующих подразделений, защита от ошибочных действий персонала) методическому (состав нормативно- технической документации
5	Состав и содержание работ по созданию системы	– перечень стадий и этапов работ – сроки исполнения – состав организаций — исполнителей работ – вид и порядок экспертизы технической документации – программа обеспечения надежности – программа метрологического обеспечения
6	Порядок контроля и приемки системы	– виды, состав, объем и методы испытаний системы – общие требования к приемке работ по стадиям – статус приемной комиссии

Продолжение таблицы 3.1

1	2	3
7	Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие	– преобразование входной информации к машиночитаемому виду – изменения в объекте автоматизации – сроки и порядок комплектования и обучения персонала
8	Требования к документированию	– перечень подлежащих разработке документов – перечень документов на машинных носителях
9	Источники разработки	документы и информационные материалы, на основании которых разрабатывается ТЗ и система

Эскизный проект предусматривает разработку предварительных проектных решений по системе и ее частям.

Выполнение стадии эскизного проектирования не является строго обязательной. Если основные проектные решения определены ранее или достаточно очевидны для конкретной ИС и объекта автоматизации, то эта стадия может быть исключена из общей последовательности работ.

Содержание эскизного проекта задается в ТЗ на систему. Как правило, на этапе эскизного проектирования определяются:

- функции ИС;
- функции подсистем, их цели и ожидаемый эффект от внедрения;
- состав комплексов задач и отдельных задач;
- концепция информационной базы и ее укрупненная структура;
- функции системы управления базой данных;
- состав вычислительной системы и других технических средств;
- функции и параметры основных программных средств.

По результатам проделанной работы оформляется, согласовывается и утверждается документация в объеме, необходимом для описания полной совокупности

принятых проектных решений и достаточном для дальнейшего выполнения работ по созданию системы.

На основе технического задания (и эскизного проекта) разрабатывается технический проект ИС. Технический проект системы - это техническая документация, содержащая общесистемные проектные решения, алгоритмы решения задач, а также оценку экономической эффективности автоматизированной системы управления и перечень мероприятий по подготовке объекта к внедрению.

На этом этапе осуществляется комплекс научно-исследовательских и экспериментальных работ для выбора основных проектных решений и расчет экономической эффективности системы.

Состав и содержание технического проекта приведены в таблице 3.2.

Таблица 3.2 - Содержание технического проекта

№ раздела	Раздел	Содержание
1	2	3
1	Пояснительная записка	– основания для разработки системы – перечень организаций разработчиков – краткая характеристика объекта с указанием основных технико-экономических показателей его функционирования и связей с другими объектами – краткие сведения об основных проектных решениях по функциональной и обеспечивающим частям системы
2	Функциональная и организационная структура системы	– обоснование выделяемых подсистем, их перечень и назначение – перечень задач, решаемых в каждой подсистеме, с краткой характеристикой их содержания – схема информационных связей между подсистемами и между задачами в рамках каждой подсистемы

Продолжение таблицы 3.2

1	2	3
3	Постановка задач и алгоритмы решения	– организационно-экономическая сущность задачи (наименование, цель решения, краткое содержание, метод, периодичность и время решения задачи, способы сбора и передачи данных, связь задачи с другими задачами, характер использования результатов решения, в которых они используются) – экономико-математическая модель задачи (структурная и развернутая форма представления)
		– входная оперативная информация (характеристика показателей, диапазон изменения, формы представления) – нормативно-справочная информация (НСИ) (содержание и формы представления) – информация, хранимая для связи с другими задачами – информация, накапливаемая для последующих решений данной задачи – информация по внесению изменений (система внесения изменений и перечень информации, подвергающейся изменениям) – алгоритм решения задачи (последовательность этапов расчета, схема, расчетные формулы)

Продолжение таблицы 3.2

1	2	3
		- контрольный пример (набор заполненных данными форм входных документов, условные документы с накапливаемой и хранимой информацией, формы выходных документов, заполненные по результатам решения экономико-технической задачи и в соответствии с разработанным алгоритмом расчета)
4	Организация информационной базы	– источники поступления информации и способы ее передачи – совокупность показателей, используемых в системе – состав документов, сроки и периодичность их поступления
5	Альбом форм документов	
6	Система математического обеспечения	– обоснование структуры математического обеспечения – обоснование выбора системы программирования – перечень стандартных программ
7	Принцип построения комплекса технических средств	– описание и обоснование схемы технологического процесса обработки данных – обоснование и выбор структуры комплекса технических средств и его функциональных групп – обоснование требований к разработке нестандартного оборудования – комплекс мероприятий по обеспечению надежности функционирования технических средств

Продолжение таблицы 3.2

1	2	3
8	Расчет экономической эффективности системы	– сводная смета затрат, связанных с эксплуатацией систем – расчет годовой экономической эффективности, источниками которой являются оптимизация производственной структуры хозяйства (объединения), снижение себестоимости продукции за счет рационального использования производственных ресурсов и уменьшения потерь, улучшения принимаемых управленческих решений
9	Мероприятия по подготовке объекта к внедрению системы	– перечень организационных мероприятий по совершенствованию бизнес-процессов – перечень работ по внедрению системы, которые необходимо выполнить на стадии рабочего проектирования, с указанием сроков и ответственных лиц
10	Ведомость документов	

В завершение стадии технического проектирования производится разработка документации на поставку серийно выпускаемых изделий для комплектования ИС, а также определяются технические требования и составляются ТЗ на разработку изделий, не изготовляемых серийно.

На стадии "рабочая документация" осуществляется создание программного продукта и разработка всей сопровождающей документации. Документация должна содержать все необходимые и достаточные сведения для обеспечения выполнения работ по вводу ИС в действие и ее эксплуатации, а также для поддержания уровня эксплуатационных характеристик (качества) системы. Разработанная документация должна быть соответствующим образом оформлена, согласована и утверждена.

Для ИС, которые являются разновидностью автоматизированных систем, устанавливают следующие основные виды испытаний: предварительные, опытная эксплуатация и приемочные. При необходимости допускается дополнительно проведение других видов испытаний системы и ее частей.

В зависимости от взаимосвязей частей ИС и объекта автоматизации испытания могут быть автономные или комплексные. Автономные испытания охватывают части системы. Их проводят по мере готовности частей системы к сдаче в опытную эксплуатацию. Комплексные испытания проводят для групп взаимосвязанных частей или для системы в целом.

Для планирования проведения всех видов испытаний разрабатывается документ "Программа и методика испытаний". Разработчик документа устанавливается в договоре или ТЗ. В качестве приложения в документ могут включаться тесты или контрольные примеры.

Предварительные испытания проводят для определения работоспособности системы и решения вопроса о возможности ее приемки в опытную эксплуатацию. Предварительные испытания следует выполнять после проведения разработчиком отладки и тестирования поставляемых программных и технических средств системы и представления им соответствующих документов об их готовности к испытаниям, а также после ознакомления персонала ИС с эксплуатационной документацией.

Опытную эксплуатацию системы проводят с целью определения фактических значений количественных и качественных характеристик системы и готовности персонала к работе в условиях ее функционирования, а также определения фактической эффективности и корректировки, при необходимости, документации.

Приемочные испытания проводят для определения соответствия системы техническому заданию, оценки качества опытной эксплуатации и решения вопроса о возможности приемки системы в постоянную эксплуатацию.

Типовое проектирование ИС

Типовое проектирование ИС предполагает создание системы из готовых типовых элементов. Основополагающим требованием для применения методов типового проектирования является возможность декомпозиции проектируемой ИС на множество составляющих компонентов (подсистем, комплексов задач, программных модулей и т.д.). Для реализации выделенных компонентов выбираются имеющиеся на рынке типовые проектные решения, которые настраиваются на особенности конкретного предприятия.

Типовое проектное решение (ТПР)- это тиражируемое (пригодное к многократному использованию) проектное решение.

Принятая классификация ТПР основана на уровне декомпозиции системы. Выделяются следующие классы ТПР:

- элементные ТПР - типовые решения по задаче или по отдельному виду обеспечения задачи (информационному, программному, техническому, математическому, организационному);
- подсистемные ТПР - в качестве элементов типизации выступают отдельные подсистемы, разработанные с учетом функциональной полноты и минимизации внешних информационных связей;
- объектные ТПР - типовые отраслевые проекты, которые включают полный набор функциональных и обеспечивающих подсистем ИС.

Каждое типовое решение предполагает наличие, кроме собственно функциональных элементов (программных или аппаратных), документации с детальным описанием ТПР и процедур настройки в соответствии с требованиями разрабатываемой системы.

Основные особенности различных классов ТПР приведены в таблице 3.3.

Таблица 3.3 - Достоинства и недостатки ТПР

Класс ТПР **Реализация ТПР**	**Достоинства**	**Недостатки**
1	2	3
Элементные ТПР Библиотеки методо-ориентированных программ	• обеспечивается применение модульного подхода к проектированию и документированию ИС	• большие затраты времени на сопряжение разнородных элементов вследствие информационной, программной и технической несовместимости • большие затраты времени на доработкуТПР отдельных элементов
Подсистемные ТПР Пакеты прикладных программ	• достигается высокая степень интеграции элементов ИС • позволяют осуществлять: модульное проектирование; параметрическую настройку программных компонентов на различные объекты управления	• адаптивность ТПР недостаточна с позиции непрерывного инжиниринга деловых процессов • возникают проблемы в комплексировании разных функциональных подсистем, особенно в случае использования решений нескольких производителей программного обеспечения

Продолжение таблицы 3.3

1	2	3
	обеспечивают: сокращение затрат на проектирование и программирование взаимосвязанных компонентов; хорошее документирование отображаемых процессов обработки информации	
Объектные ТПР Отраслевые проекты ИС	• комплексирование всех компонентов ИС за счет методологического единства и информационной, программной и технической совместимости • открытость архитектуры — позволяет устанавливатьТПР на разных программно-технических платформах • масштабируемость — допускает конфигурацию ИС для переменного числа рабочих мест • конфигурируемость — позволяет выбирать необходимое подмножество компонентов	• проблемы привязки типового проекта к конкретному объекту управления, что вызывает в некоторых случаях даже необходимость изменения организационно-экономической структуры объекта автоматизации

Для реализации типового проектирования используются два подхода: параметрически-ориентированное и модельно-ориентированное проектирование.

Параметрически-ориентированное проектирование включает следующие этапы: определение критериев оценки пригодности пакетов прикладных программ (ППП) для решения поставленных задач, анализ и оценка доступных ППП по

сформулированным критериям, выбор и закупка наиболее подходящего пакета, настройка параметров (доработка) закупленного ППП.

Модельно-ориентированное проектирование заключается в адаптации состава и характеристик типовой ИС в соответствии с моделью объекта автоматизации. Технология проектирования в этом случае должна обеспечивать единые средства для работы как с моделью типовой ИС, так и с моделью конкретного предприятия.

Типовая ИС в специальной базе содержит модель объекта автоматизации, на основе которой осуществляется конфигурирование программного обеспечения. Типовые модели описывают конфигурации информационной системы для определенных отраслей или типов производства. Реализация типового проекта предусматривает выполнение следующих операций:

- установку глобальных параметров системы;
- задание структуры объекта автоматизации;
- определение структуры основных данных;
- задание перечня реализуемых функций и процессов;
- описание интерфейсов;
- описание отчетов;
- настройку авторизации доступа;
- настройку системы архивирования.

Вопросы для самоконтроля

1 Назовите стадии и этапы проектирования РИС согласно стандарта ГОСТ 34.601-90.

2 В чем заключается различие модели деятельности организации "как есть"("as-is") от модели "как должно быть"("to-be")?

3 В чем заключается отличие типового проектирования ИС от канонического?

4 Содержание технического проекта.

5 Достоинства и недостатки ТПР.

4 Лекция 4. Понятие жизненного цикла ПО РИС

Процессы жизненного цикла: основные, вспомогательные, организационные. Содержание и взаимосвязь процессов жизненного цикла программного обеспечения (ПО) ИС. Модели жизненного цикла: каскадная, модель с промежуточным контролем, спиральная. Стадии жизненного цикла ПО ИС. Регламентация процессов проектирования в отечественных и международных стандартах.

Стандарт жизненного цикла ПО РИС

Понятие жизненного цикла ПО (ЖЦ ПО) является одним из базовых понятий программной инженерии. ЖЦ ПО определяется как период времени, который начинается с момента принятия решения о необходимости создания ПО и заканчивается в момент его полного изъятия из эксплуатации.

Основным нормативным документом, регламентирующим состав процессов ЖЦ ПО, является международный стандарт ISO/IEC 12207:1995 «Information Technology - Software Life Cycle Processes». Он определяет структуру ЖЦ, содержащую процессы, действия и задачи, которые должны быть выполнены во время создания ПО (его российский аналог ГОСТ Р ИСО/МЭК 12207—99 введен в действие в июле 2000г.). В данном стандарте процесс определяется как совокупность взаимосвязанных действий, преобразующих некоторые входные данные в выходные. Каждый процесс характеризуется определенными задачами и методами их решения, исходными данными, полученными от других процессов, и результатами.

Каждый процесс разделен на набор действий, каждое действие — на набор задач. Каждый процесс, действие или задача инициируется и выполняется другим процессом по мере необходимости, причем не существует заранее определенных последовательностей выполнения (естественно, при сохранении связей по входным данным).

В соответствии со стандартом ГОСТ Р ИСО/МЭК 12207-99 все процессы ЖЦ ПО разделены на три группы (рисунок 4.1).

• пять основных процессов (приобретение, поставка, разработка, эксплуатация, сопровождение);

• восемь вспомогательных процессов (документирование, управление конфигурацией, обеспечение качества, верификация, аттестация, совместная оценка, аудит, разрешение проблем);

• четыре организационных процесса (управление, инфраструктура, усовершенствование, обучение).

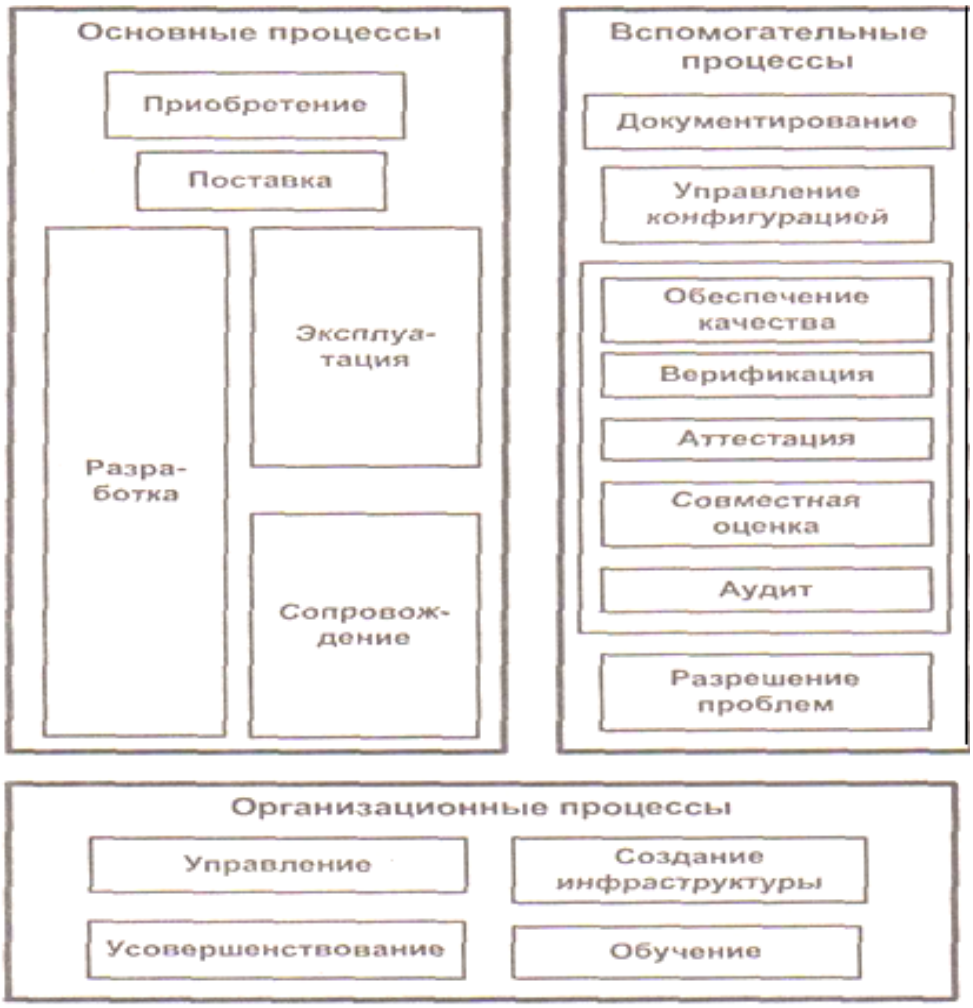


Рисунок 4.1 - Стандарт ГОСТ Р ИСО/МЭК 12207-99

Современным направлением развития информационного обеспечения систем организационного и технического управления являются CALS-технологии, направленные на создание единого информационного пространства, основу которого составляют интегрированные базы данных.

В России принят адекватный аналог CALS — Информационная поддержка жизненного цикла изделий (ИПИ). Схема ИПИ представлена на рисунке 4.2. Основу ИПИ составляет интегрированная информационная среда (ИИС).

Госстандарт России определяет ИИС как совокупность распределенных баз данных, содержащих сведения об изделиях, производственной среде, ресурсах и процессах предприятия, обеспечивающая корректность, актуальность, сохранность и доступность данных тем субъектам производственно-хозяйственной деятельности, участвующим в осуществлении жизненного цикла изделия, которым это необходимо и разрешено.

Следовательно, основой для создания ИИС являются базы данных. Компоненты формирования баз данных представлены на рисунке 4.3.

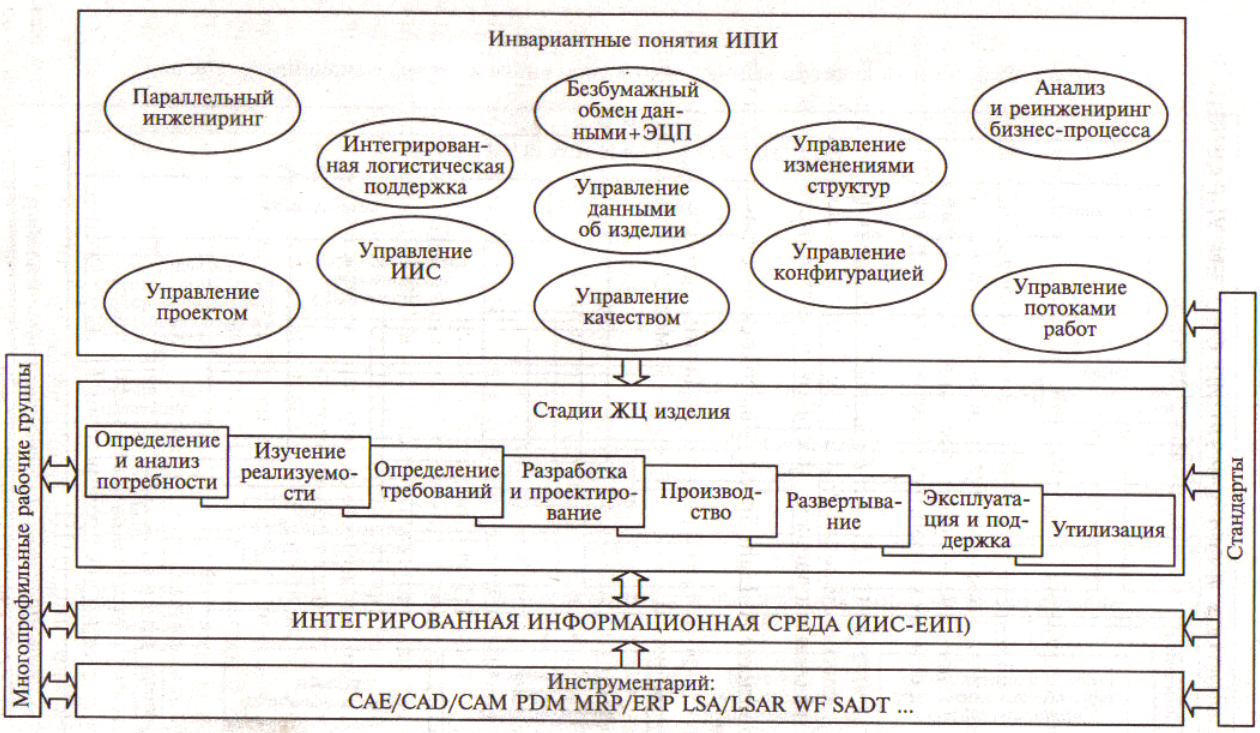


Рисунок 4.2 – Схема ИПИ

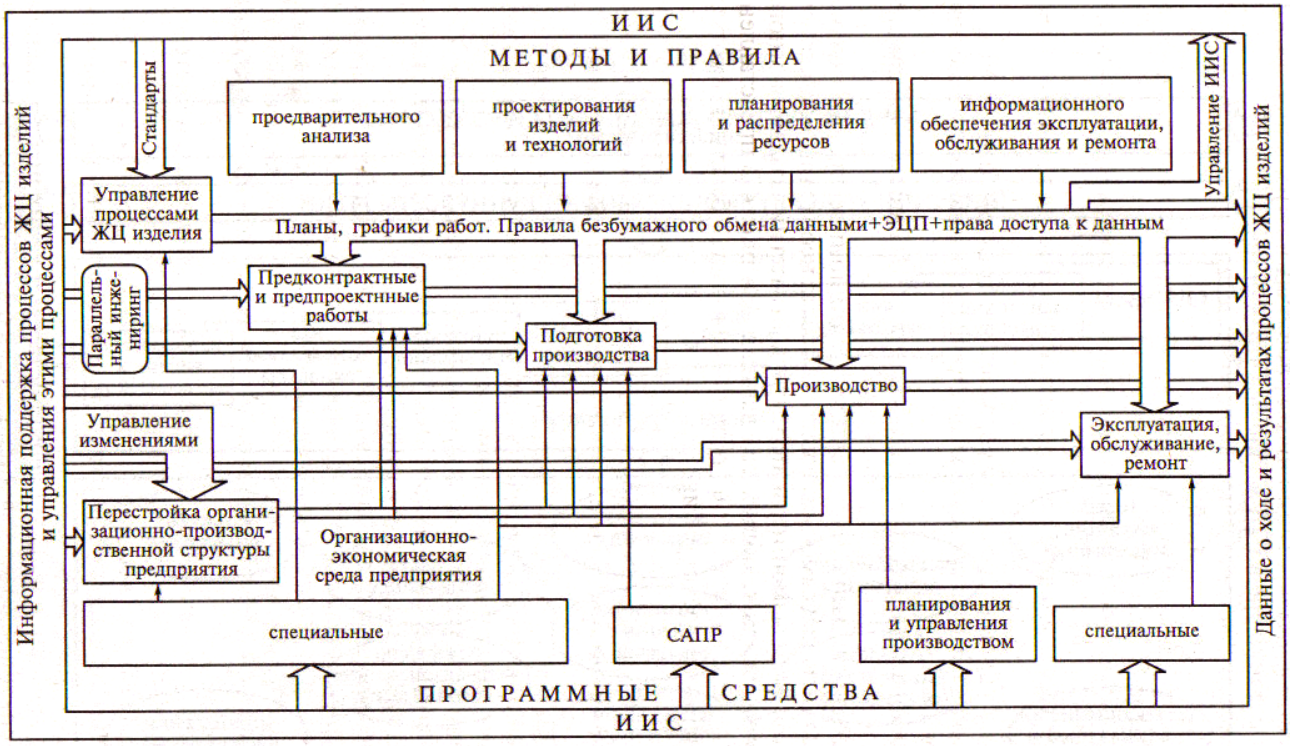


Рисунок 4.3 - Компоненты формирования баз данных

Методология проектирования РИС

Методология проектирования информационных систем описывает процесс создания и сопровождения систем в виде жизненного цикла (ЖЦ) ИС, представляя его как некоторую последовательность стадий и выполняемых на них процессов. Для каждого этапа определяются состав и последовательность выполняемых работ, получаемые результаты, методы и средства, необходимые для выполнения работ, роли и ответственность участников и т.д. Такое формальное описание ЖЦ ИС позволяет спланировать и организовать процесс коллективной разработки и обеспечить управление этим процессом.

Жизненный цикл ИС можно представить как ряд событий, происходящих с системой в процессе ее создания и использования. Модель жизненного цикла отражает различные состояния системы, начиная с момента возникновения необходимости в данной ИС и заканчивая моментом ее полного выхода из употребления. Модель жизненного цикла - структура, содержащая процессы, действия и задачи,

которые осуществляются в ходе разработки, функционирования и сопровождения программного продукта в течение всей жизни системы, от определения требований до завершения ее использования.

В настоящее время известны и используются следующие модели жизненного цикла:

— Каскадная модель (рисунок 4.4) предусматривает последовательное выполнение всех этапов проекта в строго фиксированном порядке. Переход на следующий этап означает полное завершение работ на предыдущем этапе.

— Поэтапная модель с промежуточным контролем (рисунок 4.5). Разработка ИС ведется итерациями с циклами обратной связи между этапами. Межэтапные корректировки позволяют учитывать реально существующее взаимовлияние результатов разработки на различных этапах; время жизни каждого из этапов растягивается на весь период разработки.

— Спиральная модель (рисунок 4.6). На каждом витке спирали выполняется создание очередной версии продукта, уточняются требования проекта, определяется его качество и планируются работы следующего витка. Особое внимание уделяется начальным этапам разработки - анализу и проектированию, где реализуемость тех или иных технических решений проверяется и обосновывается посредством создания прототипов (макетирования).

Рисунок 4.4 - Каскадная модель ЖЦ ИС

Рисунок 4.5 - Поэтапная модель с промежуточным контролем

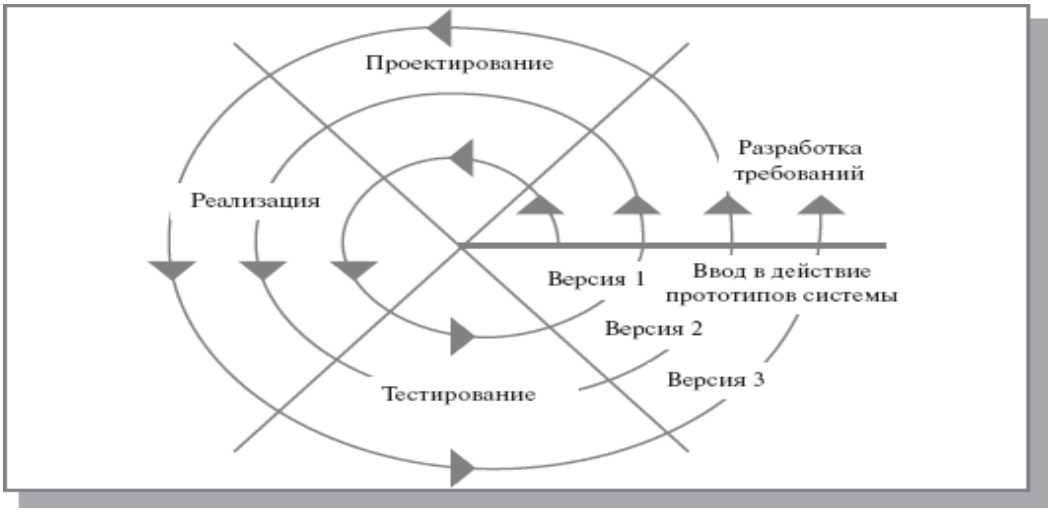


Рисунок 4.6 - Спиральная модель ЖЦ ИС

На практике наибольшее распространение получили две основные модели жизненного цикла:

- каскадная модель (характерна для периода 1970-1985 гг.);
- спиральная модель (характерна для периода после 1986.г.).

В ранних проектах достаточно простых ИС каждое приложение представляло собой единый, функционально и информационно независимый блок. Для разработки такого типа приложений эффективным оказался каскадный способ. Каждый этап завершался после полного выполнения и документального оформления всех предусмотренных работ.

Можно выделить следующие положительные стороны применения каскадного подхода:

- на каждом этапе формируется законченный набор проектной документации, отвечающий критериям полноты и согласованности;
- выполняемые в логической последовательности этапы работ позволяют планировать сроки завершения всех работ и соответствующие затраты.

Каскадный подход хорошо зарекомендовал себя при построении относительно простых ИС, когда в самом начале разработки можно достаточно точно и полно сформулировать все требования к системе. Основным недостатком этого подхода является то, что реальный процесс создания системы никогда полностью не укладывается в такую жесткую схему, постоянно возникает потребность в возврате к предыдущим этапам и уточнении или пересмотре ранее принятых решений. В результате реальный процесс создания ИС оказывается соответствующим поэтапной модели с промежуточным контролем.

Однако и эта схема не позволяет оперативно учитывать возникающие изменения и уточнения требований к системе. Согласование результатов разработки с пользователями производится только в точках, планируемых после завершения каждого этапа работ, а общие требования к ИС зафиксированы в виде технического задания на все время ее создания. Таким образом, пользователи зачастую получают систему, не удовлетворяющую их реальным потребностям.

Спиральная модель ЖЦ была предложена для преодоления перечисленных проблем. На этапах анализа и проектирования реализуемость технических решений и степень удовлетворения потребностей заказчика проверяется путем создания прототипов. Каждый виток спирали соответствует созданию работоспособного фрагмента или версии системы. Это позволяет уточнить требования, цели и характеристики проекта, определить качество разработки, спланировать работы следующего витка спирали. Таким образом углубляются и последовательно конкретизируются детали проекта и в результате выбирается обоснованный вариант, который удовлетворяет действительным требованиям заказчика и доводится до реализации.

Итеративная разработка отражает объективно существующий спиральный цикл создания сложных систем. Она позволяет переходить на следующий этап, не дожидаясь полного завершения работы на текущем и решить главную задачу - как можно быстрее показать пользователям системы работоспособный продукт, тем самым активизируя процесс уточнения и дополнения требований.

Основная проблема спирального цикла - определение момента перехода на следующий этап. Для ее решения вводятся временные ограничения на каждый из этапов жизненного цикла, и переход осуществляется в соответствии с планом, даже если не вся запланированная работа закончена. Планирование производится на основе статистических данных, полученных в предыдущих проектах, и личного опыта разработчиков.

Несмотря на настойчивые рекомендации компаний и экспертов в области проектирования и разработки ИС, многие компании продолжают использовать каскадную модель вместо какого-либо варианта итерационной модели. Основные причины, по которым каскадная модель сохраняет свою популярность, следующие:

1 **Привычка** - многие ИТ - специалисты получали образование в то время, когда изучалась только каскадная модель, поэтому она используется ими и в наши дни.

2 **Иллюзия снижения рисков** участников проекта (заказчика и исполнителя).Каскадная модель предполагает разработку законченных продуктов на каждом этапе: технического задания, технического проекта, программного продукта и пользовательской документации. Разработанная документация позволяет не только определить требования к продукту следующего этапа, но и определить обязанности сторон, объем работ и сроки, при этом окончательная оценка сроков и стоимости проекта производится на начальных этапах, после завершения обследования. Очевидно, что если требования к информационной системе меняются в ходе реализации проекта, а качество документов оказывается невысоким (требования неполны и/или противоречивы), то в действительности использование каскадной модели создает лишь иллюзию определенности и на деле увеличивает риски, уменьшая

лишь ответственность участников проекта. При формальном подходе менеджер проекта реализует только те требования, которые содержатся в спецификации, опирается на документ, а не на реальные потребности бизнеса. Есть два основных типа контрактов на разработку ПО. Первый тип предполагает выполнение определенного объема работ за определенную сумму в определенные сроки (fixed price). Второй тип предполагает повременную оплату работы (time work). Выбор того или иного типа контракта зависит от степени определенности задачи. Каскадная модель с определенными этапами и их результатами лучше приспособлена для заключения контракта с оплатой по результатам работы, а именно этот тип контрактов позволяет получить полную оценку стоимости проекта до его завершения. Более вероятно заключение контракта с повременной оплатой на небольшую систему, с относительно небольшим весом в структуре затрат предприятия. Разработка и внедрение интегрированной информационной системы требует существенных финансовых затрат, поэтому используются контракты с фиксированной ценой, и, следовательно, каскадная модель разработки и внедрения. Спиральная модель чаще применяется при разработке информационной системы силами собственного отдела ИТ предприятия.

3 **Проблемы внедрения** при использовании итерационной модели. В некоторых областях спиральная модель не может применяться, поскольку невозможно использование/тестирование продукта, обладающего неполной функциональностью (например, военные разработки, атомная энергетика и т.д.). Поэтапное итерационное внедрение информационной системы для бизнеса возможно, но сопряжено с организационными сложностями (перенос данных, интеграция систем, изменение бизнес-процессов, учетной политики, обучение пользователей). Трудозатраты при поэтапном итерационном внедрении оказываются значительно выше, а управление проектом требует настоящего искусства. Предвидя указанные сложности, заказчики выбирают каскадную модель, чтобы "внедрять систему один раз".

Каждая из стадий создания системы предусматривает выполнение определенного объема работ, которые представляются в виде процессов ЖЦ. Процесс оп-

ределяется как совокупность взаимосвязанных действий, преобразующих входные данные в выходные. Описание каждого процесса включает в себя перечень решаемых задач, исходных данных и результатов.

Существует целый ряд стандартов, регламентирующих ЖЦ ПО, а в некоторых случаях и процессы разработки.

Значительный вклад в теорию проектирования и разработки информационных систем внесла компания IBM, предложив еще в середине 1970-х годов методологию BSP (Business System Planning - методология организационного планирования). Метод структурирования информации с использованием матриц пересечения бизнес-процессов, функциональных подразделений, функций систем обработки данных (информационных систем), информационных объектов, документов и баз данных, предложенный в BSP, используется сегодня не только в ИТ-проектах, но и проектах по реинжинирингу бизнес-процессов, изменению организационной структуры. Важнейшие шаги процесса BSP, их последовательность (получить поддержку высшего руководства, определить процессы предприятия, определить классы данных, провести интервью, обработать и организовать данные интервью) можно встретить практически во всех формальных методиках, а также в проектах, реализуемых на практике.

Среди наиболее известных стандартов можно выделить следующие:

- ГОСТ 34.601-90 - распространяется на автоматизированные системы и устанавливает стадии и этапы их создания. Кроме того, в стандарте содержится описание содержания работ на каждом этапе. Стадии и этапы работы, закрепленные в стандарте, в большей степени соответствуют каскадной модели жизненного цикла.
- ISO/IEC 12207:1995 - стандарт на процессы и организацию жизненного цикла. Распространяется на все виды заказного ПО. Стандарт не содержит описания фаз, стадий и этапов.
- Custom Development Method (методика Oracle) по разработке прикладных информационных систем - технологический материал, детализированный до уровня заготовок проектных документов, рассчитанных на использование в проектах с

применением Oracle. Применяется CDM для классической модели ЖЦ (предусмотрены все работы/задачи и этапы), а также для технологий "быстрой разработки" (Fast Track) или "облегченного подхода", рекомендуемых в случае малых проектов.

- Rational Unified Process (RUP) предлагает итеративную модель разработки, включающую четыре фазы: начало, исследование, построение и внедрение. Каждая фаза может быть разбита на этапы (итерации), в результате которых выпускается версия для внутреннего или внешнего использования. Прохождение через четыре основные фазы называется циклом разработки, каждый цикл завершается генерацией версии системы. Если после этого работа над проектом не прекращается, то полученный продукт продолжает развиваться и снова минует те же фазы. Суть работы в рамках RUP - это создание и сопровождение моделей на базе UML
- Microsoft Solution Framework (MSF) сходна с RUP, так же включает четыре фазы: анализ, проектирование, разработка, стабилизация, является итерационной, предполагает использование объектно-ориентированного моделирования. MSF в сравнении с RUP в большей степени ориентирована на разработку бизнес-приложений.
- Extreme Programming (XP). Экстремальное программирование (самая новая среди рассматриваемых методологий) сформировалось в 1996 году. В основе методологии командная работа, эффективная коммуникация между заказчиком и исполнителем в течение всего проекта по разработке ИС, а разработка ведется с использованием последовательно дорабатываемых прототипов.

В соответствии с базовым международным стандартом ISO/IEC 12207 все процессы ЖЦ ПО делятся на три группы:

1 Основные процессы: приобретение; поставка; разработка; эксплуатация; сопровождение.

2 Вспомогательные процессы: документирование; управление конфигурацией; обеспечение качества; разрешение проблем; аудит; аттестация; совместная оценка; верификация.

3 Организационные процессы: создание инфраструктуры; управление; обучение; усовершенствование.

В таблице 4.1 приведены ориентировочные описания основных процессов ЖЦ. Вспомогательные процессы предназначены для поддержки выполнения основных процессов, обеспечения качества проекта, организации верификации, проверки и тестирования ПО. Организационные процессы определяют действия и задачи, выполняемые как заказчиком, так и разработчиком проекта для управления своими процессами.

Для поддержки практического применения стандарта ISO/IEC 12207 разработан ряд технологических документов: Руководство для ISO/IEC 12207 (ISO/IEC TR 15271:1998 Information technology - Guide for ISO/IEC 12207) и Руководство по применению ISO/IEC 12207 к управлению проектами (ISO/IEC TR 16326:1999 Software engineering - Guide for the application of ISO/IEC 12207 to project management).

Таблица 4.1 - Содержание основных процессов ЖЦ ПО ИС (ISO/IEC 12207)

Процесс (исполнитель процесса)	**Действия**	**Вход**	**Результат**
1	2	3	4
Приобретение (заказчик)	Инициирование Подготовка заявочных предложений Подготовка договора Контроль деятельности поставщика Приемка ИС	Решение о начале работ по внедрению ИС Результаты обследования деятельности заказчика Результаты анализа рынка ИС/ тендера План поставки/ разработки Комплексный тест ИС	Технико-экономическое обоснование внедрения ИС Техническое задание на ИС Договор на поставку/ разработку Акты приемки этапов работы Акт приемно-сдаточных испытаний

Продолжение таблицы 4.1

1	2	3	4
Поставка (разработчик ИС)	инициирование Ответ на заявочные предложения Подготовка договора Планирование исполнения Поставка ИС	Техническое задание на ИС Решение руководства об участии в разработке Результаты тендера Техническое задание на ИС План управления проектом	Решение об участии в разработке Коммерческие предложения/ конкурсная заявка Договор на поставку/ разработку План управления проектом
Разработка (разработчик ИС)	Подготовка Анализ требований к ИС Проектирование архитектуры ИС Разработка требований к ПО Проектирование архитектуры ПО Детальное проектирование ПО Кодирование и тестирование ПО Интеграция ПО и квалификационное тестирование ПО Интеграция ИС и квалификационное тестирование ИС	Техническое задание на ИС Техническое задание на ИС, модель ЖЦ Техническое задание на ИС Подсистемы ИС Спецификации требования к компонентам ПО Архитектура ПО Материалы детального проектирования ПО План интеграции ПО, тесты Архитектура ИС, ПО, документация на ИС, тесты	Используемая модель ЖЦ, стандарты разработки План работ Состав подсистем, компоненты оборудования Спецификации требования к компонентам ПО Состав компонентов ПО, интерфейсы с БД, план интеграции ПО Проект БД, спецификации интерфейсов между компонентами ПО, требования к тестам Тексты модулей ПО, акты автономного тестирования

Продолжение таблицы 4.1

1	2	3	4
			Оценка соответствия комплекса ПО требованиям ТЗ Оценка соответствия ПО, БД, технического комплекса и комплекта документации требованиям ТЗ

Позднее был разработан и в 2002 г. опубликован стандарт на процессы жизненного цикла систем (ISO/IEC 15288 System life cycle processes). К разработке стандарта были привлечены специалисты различных областей: системной инженерии, программирования, управления качеством, человеческими ресурсами, безопасностью и пр. Был учтен практический опыт создания систем в правительственных, коммерческих, военных и академических организациях. Стандарт применим для широкого класса систем, но его основное предназначение - поддержка создания компьютеризированных систем.

Согласно стандарту ISO/IEC серии 15288 в структуру ЖЦ следует включать следующие группы процессов:

1 Договорные процессы: приобретение (внутренние решения или решения внешнего поставщика); поставка (внутренние решения или решения внешнего поставщика).

2 Процессы предприятия: управление окружающей средой предприятия; инвестиционное управление; управление ЖЦ ИС; управление ресурсами; управление качеством.

3 Проектные процессы: планирование проекта; оценка проекта; контроль проекта; управление рисками; управление конфигурацией; управление информационными потоками; принятие решений.

4 Технические процессы: определение требований; анализ требований; разработка архитектуры; внедрение; интеграция; верификация; переход; аттестация; эксплуатация; сопровождение; утилизация.

5 Специальные процессы: определение и установка взаимосвязей исходя из задач и целей.

Стадии создания системы, предусмотренные в стандарте ISO/IEC 15288, несколько отличаются от рассмотренных выше. Перечень стадий и основные результаты, которые должны быть достигнуты к моменту их завершения, приведены в таблице 4.2.

Таблица 4.2 - Стадии создания систем (ISO/IEC 15288)

№ стадии	Стадия	Описание
1	2	3
1	Формирование концепции	Анализ потребностей, выбор концепции и проектных решений
2	Разработка	Проектирование системы
3	Реализация	Изготовление системы
4	Эксплуатация	Ввод в эксплуатацию и использование системы
5	Поддержка	Обеспечение функционирования системы
6	Снятие с эксплуатации	Прекращение использования, демонтаж, архивирование системы

Вопросы для самоконтроля

1 Дайте определение жизненного цикла изделия.

2 Какие модели ЖЦ вы знаете, их содержание.

3 На какие группы в соответствии с базовым международным стандартом ISO/IEC 12207 делятся все процессы ЖЦ ПО.

4 Какие группы процессов включены в структуру ЖЦ согласно стандарту ISO/IEC серии 15288.

5 Стадии создания систем (ISO/IEC 15288).

5 Лекция 5. Распределенная обработка данных

Основные понятия

При размещении СУБД на персональном компьютере, который не находится в сети, БД всегда используется в монопольном режиме. Даже если с ней работают несколько пользователей, они могут работать только последовательно.

Однако, как показала практика применения локальных баз данных, в большинстве случаев информация, которая в них содержится, носит многопользовательский характер, поэтому возникает необходимость разработки таких СУБД, которые обеспечили бы возможность одновременной работы пользователей с базами данных. Тем более, что все современные предприятия строят свою политику в области информационного обеспечения на основе принципов CALS-технологий.

Системы управления базами данных, обеспечивающие возможность одновременного доступа к информации различным пользователям называют *системами управления распределенными базами данных.* В общем случае режимы использования БД имеют вид, представленный на рисунке 5.1. Рассмотрим основные понятия, применяемые в системах управления распределенными базами данных.

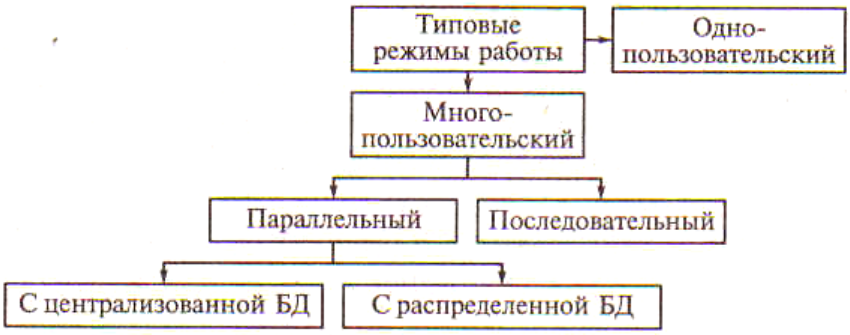


Рисунок 5.1 - Режимы работы с базами данных

Пользователь БД — программа или человек, обращающийся к базе данных.

Запрос — процесс обращения пользователя к БД с целью ввода, получения или изменения информации в БД.

Транзакция — последовательность операций модификации данных в БД, переводящая БД из одного непротиворечивого состояния в другое непротиворечивое состояние.

Логическая структура БД — определение БД на физически независимом уровне; ближе всего соответствует концептуальной модели БД.

Топология БД, или *структура распределенной БД,* — схема распределения физической организации базы данных в сети.

Локальная автономность означает, что информация локальной БД и связанные с ней определения данных принадлежат локальному владельцу и им управляются.

Удаленный запрос — запрос, который выполняется с использованием модемной связи.

Возможность реализации удаленной транзакции — обработка одной транзакции, состоящей из множества SQL-запросов, на одном удаленном узле.

Поддержка распределенной транзакции допускает обработку транзакции, состоящей из нескольких запросов SQL, которые выполняются на нескольких узлах сети (удаленных или локальных), но каждый запрос в этом случае обрабатывается только на одном узле.

Распределенный запрос — запрос, при обработке которого используются данные из БД, расположенные в разных узлах сети.

Системы распределенной обработки данных в основном связаны с первым поколением БД, которые строились на мультипрограммных операционных системах и использовали централизованное хранение БД на устройствах внешней памяти центральной ЭВМ и терминальный многопользовательский режим доступа. При этом пользовательские терминалы не имели собственных ресурсов, т. е. процессоров и памяти, которые могли бы использоваться для хранения и обработки данных. Первой полностью реляционной системой, работающей в многопользовательском режиме, была СУБД SYSTEM R фирмы IBM. Именно в ней были реализованы как язык манипулирования данными SQL, так и основные принципы синхронизации, применяемые при

распределенной обработке данных, которые до сих пор являются базисными практически во всех коммерческих СУБД.

Модели клиент - сервер в технологии распределенных баз данных

Вычислительная модель клиент - сервер связана с появлением в 1990-х гг. открытых систем. Термин «клиент - сервер» применялся к архитектуре программного обеспечения, которое состояло из двух процессов обработки информации: клиентской и серверной. Клиентский процесс запрашивал некоторые услуги, а серверный процесс обеспечивал их выполнение. При этом предполагалось, что один серверный процесс может обслужить множество клиентских процессов. Учитывая что аппаратная реализация этой модели управления базами данных связана с созданием локальных вычислительных сетей предприятия, такую организацию процесса обработки информации называют архитектурой клиент - сервер.

Основной принцип технологии клиент - сервер применительно к технологии управления базами данных заключается в разделении функций стандартного интерактивного приложения на пять групп, имеющих различную природу:

- функции ввода и отображения данных (Presentation Logic);
- прикладные функции, определяющие основные алгоритмы решения задач приложения (Business Logic);
- функции обработки данных внутри приложения (Database Logic);
- функции управления информационными ресурсами (Database Manager System);
- служебные функции, играющие роль связок между функциями первых четырех групп.

Структура типового приложения, работающего с базой данных в архитектуре клиент— сервер, приведена на рисунке 5.2.

Презентационная логика как часть приложения определяется тем, что пользователь видит на своем экране, когда работает приложение. Сюда относятся все интерфейсные экранные формы, которые пользователь видит или заполняет в ходе работы

приложения. К этой же части относится все то, что выводится пользователю на экран как результаты решения некоторых промежуточных задач либо как справочная информация. Поэтому основными задачами презентационной логики являются:

- формирование экранных изображений;
- чтение и запись в информации экранные формы;
- управление экраном;
- обработка движений мыши и нажатие клавиш клавиатуры.

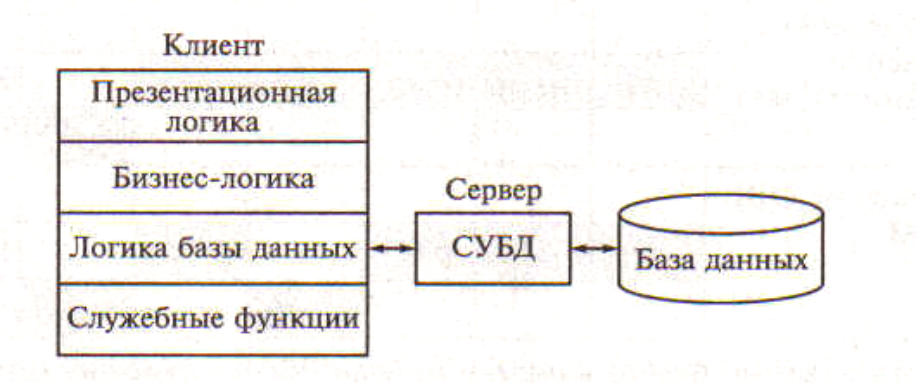


Рисунок 5.2 - Структура типового приложения, работающего с базой данных

Бизнес-логика, или *логика собственно приложений* — это часть кода приложения, которая определяет собственно алгоритмы решения конкретных задач приложения. Обычно этот код пишется с использованием различных языков программирования, таких как C, C++, Visual Basic и др.

Логика обработки данных — это часть кода приложения, которая непосредственно связана с обработкой данных внутри приложения. Данными управляет собственно СУБД. Для обеспечения доступа к данным используется язык SQL.

Процессор управления данными — это собственно СУБД. В идеале функции СУБД должны быть скрыты от бизнес-логики приложения, однако для рассмотрения архитектуры приложения их надо выделить в отдельную часть приложения.

В централизованной архитектуре эти части приложения располагаются в единой среде и комбинируются внутри одной исполняемой программы.

В децентрализованной архитектуре эти задачи могут быть по-разному распределены между серверным и клиентским процессами. В зависимости от характера распределения можно выделить следующие модели распределений:

• распределенная презентация (DR – Distribution Presentation);

• удаленная презентация (RP - Remote Presentation);

• распределенная бизнес-логика (RBL – Remote business logic);

• распределенное управление данными (DDM – Distributed data manegement);

• удаленное управление данными (RDM – Remote data manegement).

Эта условная классификации показывает, как могут быть распределены отдельные задачи между серверным и клиентскими процессами. В этой классификации отсутствует реализация удаленной бизнес-логики. Считается, что она не может быть удалена сама по себе полностью, а может быть лишь распределена между разными процессами, которые могут взаимодействовать друг с другом.

Двухуровневые модели

Двухуровневая модель фактически является результатом распределения пяти указанных выше функций между двумя процессами, которые выполняются на двух платформах: на клиенте и на сервере. В чистом виде почти никакая модель не существует, однако рассмотрим наиболее характерные особенности каждой двухуровневой модели: модели удаленного управления данными и модели удаленного доступа к данным.

Модель удаленного управления данными. Она также называется моделью файлового сервера (FS – File Server). В этой модели презентационная логика и бизнес-логика располагаются на клиентской части. На сервере располагаются файлы с данными, и поддерживается доступ к файлам. Функции управления информационными ресурсами в этой модели находятся на клиентской части. Распределение функций в этой модели представлено на рисунке 5.3.

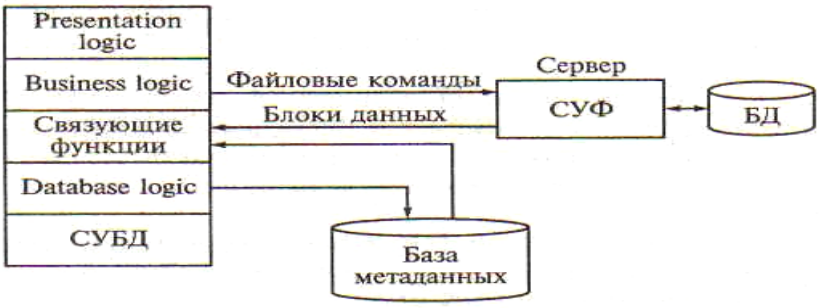


Рисунок 5.3 – модель файлового сервера

В этой модели файлы базы данных хранятся на сервере, клиент обращается к серверу с файловыми командами, а механизм управления всеми информационными ресурсами, собственно база метаданных, находится на клиенте.

Достоинство этой модели заключается в том, что приложение разделено на два взаимодействующих процесса. При этом сервер (серверный процесс) может обслуживать множество клиентов, которые обращаются к нему с запросами.

Собственно СУБД должна находиться в этой модели на клиентском компьютере.

Алгоритм выполнения клиентского запроса сводится к следующему.

1 Запрос формулируется в командах ЯМД.

2 СУБД переводит этот запрос в последовательность файловых команд.

3 Каждая файловая команда вызывает перекачку блока информации на компьютер клиента, а СУБД анализирует полученную информацию; если в полученном блоке не содержится ответ на запрос, то принимается решение о перекачке следующего блока информации, и т.д.

4 Перекачка информации с сервера на клиентский компьютер производится до тех пор, пока не будет получен ответ на запрос клиента.

Данная модель имеет следующие недостатки:

• высокий сетевой трафик, который связан с передачей по сети множества блоков и файлов, необходимых приложению;

• узкий спектр операций манипулирования с данными, который определяется только файловыми командами;

• отсутствие адекватных средств безопасности доступа к данным (защита только на уровне файловой системы).

Модель удаленного доступа к данным. В модели удаленного доступа (RDA – Remote Data Access) база данных хранится на сервере. На сервере же находится и ядро СУБД. На компьютере клиента располагается презентационная логика и бизнес-логика приложения. Клиент обращается к серверу с запросами на языке SQL. Структура модели удаленного доступа приведена на рисунке 5.4.

Рисунок 5.4 - Структура модели удаленного доступа к данным

Преимущества данной модели заключаются в следующем:

• перенос компонента представления и прикладного компонента на клиентский компьютер существенно разгружает сервер БД, сводя к минимуму общее число выполняемых процессов в операционной системе;

« сервер БД освобождается от несвойственных ему функций; процессор или процессоры сервера целиком загружаются операциями обработки данных запросов и транзакций;

• резко уменьшается загрузка сети, так как по ней от клиентов к серверу передаются не запросы на ввод-вывод в файловой терминологии, а запросы на SQL, а их объем существенно меньше. В ответ на запросы клиент получает только данные, соответствующие запросу, а не блоки файлов.

Основное достоинство RDA-модели — унификация интерфейса клиент—сервер (стандартом при общении приложения-клиента и сервера становится язык SQL).

Данная модель имеет следующие недостатки:

• запросы на языке SQL при интенсивной работе клиентской части приложения могут существенно загрузить сеть;

• так как в этой модели на клиенте располагается и презентационная логика, и бизнес-логика приложения, то при повторении аналогичных функций в разных приложениях код соответствующей бизнес-логики должен быть повторен для каждого клиентского приложения. Это вызывает излишнее дублирование приложения;

• сервер в этой модели играет пассивную роль, поэтому функции управления информационными ресурсами должны выполняться на клиенте.

Модель сервера баз данных

Для того чтобы избавиться от недостатков модели удаленного доступа, должны быть соблюдены следующие условия.

1 Необходимо, чтобы БД в каждый момент отражала текущее состояние предметной области, которое определяется не только собственно данными, но и связями между объектами данных, т.е. данные, которые хранятся в БД, в каждый момент времени должны быть непротиворечивыми.

2 БД должна отражать некоторые правила предметной области, законы, по которым она функционирует (business rules). Например, завод может нормально работать только в том случае, если на складе имеется некоторый достаточный запас (страховой запас) деталей определенной номенклатуры; деталь может быть запущена в производство только в том случае, если на складе имеется в наличии достаточно материала для ее изготовления, и т.д.

3 Необходим постоянный контроль за состоянием БД, отслеживание всех изменений и адекватная реакция на них. Например, при достижении некоторым измеряемым параметром критического значения должно произойти отключение определенной аппаратуры; при уменьшении товарного запаса ниже допустимой нормы должна быть сформирована заявка конкретному поставщику на поставку соответствующего товара и т. п.

4 Необходимо, чтобы возникновение некоторой ситуации в БД четко и оперативно влияло на ход выполнения прикладной задачи.

5 Одной из важнейших проблем СУБД является контроль типов данных. В настоящий момент СУБД контролирует синтаксически только стандартно-допустимые типы данных, т.е. такие, которые определены в DDL (data definition language) — языке описания данных, который является частью SQL. Однако в реальных предметных областях действуют данные, которые несут в себе еще и семантическую составляющую, например координаты объектов или единицы измерений.

Такую модель поддерживают большинство современных СУБД: Informix, Ingres, Sybase, Oracle, MS SQL Server. Основу данной модели составляет механизм хранимых процедур как средство программирования SQL-сервера, механизм триггеров как механизм отслеживания текущего состояния информационного хранилища и механизм ограничений на пользовательские типы данных, который иногда называется механизмом поддержки доменной структуры. Модель активного сервера базы данных представлена на рисунке 5.5.

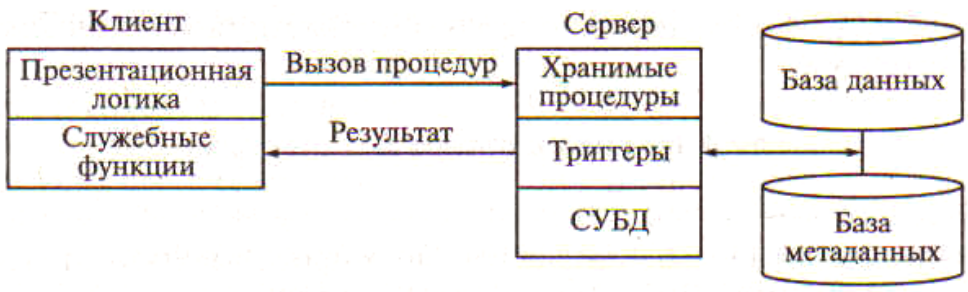


Рисунок 5.5 - Модель активного сервера базы данных

В этой модели бизнес-логика разделена между клиентом и сервером. На сервере бизнес-логика реализована в виде хранимых процедур — специальных программных модулей, которые хранятся в БД и управляются непосредственно СУБД. Клиентское приложение обращается к серверу с командой запуска хранимой процедуры, а сервер выполняет эту процедуру и регистрирует все изменения в БД, которые в ней предусмотрены. Сервер возвращает клиенту данные выполненного запроса, которые требуются клиенту либо для вывода на экран, либо для выполнения

части бизнес-логики. При этом трафик обмена информацией между клиентом и сервером резко уменьшается.

Централизованный контроль в модели сервера баз данных выполняется с использованием механизма триггеров. Триггеры также являются частью БД.

Термин «триггер» взят из электроники и семантически очень точно характеризует механизм отслеживания специальных событий, которые связаны с состоянием БД. Триггер в БД является неким тумблером, который срабатывает при возникновении определенного события в БД. Ядро СУБД проводит мониторинг всех событий, которые вызывают созданные и описанные триггеры в БД, и при возникновении соответствующего события сервер запускает соответствующий триггер. Каждый триггер представляет собой также некоторую программу, которая выполняется над базой данных. Триггеры могут вызывать хранимые процедуры.

Механизм использования триггеров предполагает, что при срабатывании одного триггера могут возникнуть события, которые вызовут срабатывание других триггеров.

В данной модели сервер является активным, потому что не только клиент, но и сам сервер, используя механизм триггеров, может быть инициатором обработки данных в БД.

И хранимые процедуры, и триггеры хранятся в словаре БД. Они могут быть использованы несколькими клиентами, что существенно уменьшает дублирование алгоритмов обработки данных в разных клиентских приложениях.

Недостатком данной модели является очень большая загрузка сервера, так как он обслуживает множество клиентов и выполняет следующие функции:

• осуществляет мониторинг событий, связанных с описанными триггерами;

• обеспечивает автоматическое срабатывание триггеров при возникновении связанных с ними событий;

• обеспечивает исполнение внутренней программы каждого триггера;

• запускает хранимые процедуры по запросам пользователей;

• запускает хранимые процедуры из триггеров;

• возвращает требуемые данные клиенту;

• обеспечивает все функции СУБД (доступ к данным, контроль и поддержку целостности данных в БД, контроль доступа, обеспечение корректной параллельной работы всех пользователей с единой БД).

Если мы перенесем на сервер большую часть бизнес-логики приложений, то требования к клиентам в этой модели резко уменьшатся. Иногда такую модель называют моделью с тонким клиентом. Ранее рассмотренные модели называют моделями с толстым клиентом.

Для разгрузки сервера была предложена трехуровневая модель — модель сервера приложений.

Модель сервера приложений

Эта модель является расширением двухуровневой модели, в ней вводится дополнительный промежуточный уровень между клиентом и сервером. Архитектура трехуровневой модели приведена на рисунке 5.6. Этот промежуточный уровень содержит один или несколько серверов приложений.

В этой модели компоненты приложения делятся между тремя исполнителями: клиентом, сервером, сервером базы данных.

Клиент обеспечивает логику представления, включая графический пользовательский интерфейс, локальные редакторы; клиент может запускать локальный код приложения клиента, который может содержать обращения к локальной БД, расположенной на компьютере-клиенте. Клиент исполняет коммуникационные функции front-end части приложения, которые обеспечивают доступ клиенту в локальную или глобальную сеть. Дополнительно реализация взаимодействия между клиентом и сервером может включать в себя управление распределенными транзакциями, что соответствует тем случаям, когда клиент также является клиентом менеджера распределенных транзакций.

Серверы приложений составляют новый промежуточный уровень архитектуры. Серверы приложений поддерживают функции клиентов как частей взаимодействующих рабочих групп, поддерживают сетевую доменную операционную среду, хранят и исполняют наиболее общие правила бизнес-логики, поддерживают каталоги с данными, обеспечивают обмен сообщениями и поддержку запросов, особенно в распределенных транзакциях..

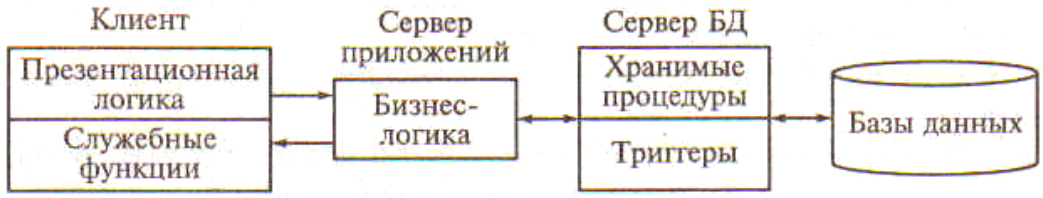


Рисунок 5.6 - Модель сервера приложений

Серверы баз данных в этой модели занимаются исключительно функциями СУБД: обеспечивают функции создания и ведения БД, поддерживают целостность реляционной БД, обеспечивают функции хранилищ данных (warehouse services). Кроме того, на них возлагаются функции создания резервных копий БД и восстановления БД после сбоев, управления выполнением транзакций и поддержки устаревших (унаследованных) приложений (legacy application).

Эта модель обладает большей гибкостью, чем двухуровневые модели. Наиболее заметны преимущества модели сервера приложений в тех случаях, когда клиенты выполняют сложные аналитические расчеты над базой данных, которые относятся к области OLAP-приложений (On-line analytical processing).

В этой модели большая часть бизнес-логики клиента изолирована от возможностей встроенного SQL, реализованного в конкретной СУБД, и может быть выполнена на языках программирования, таких как C, C++, CoЬo1. Это повышает переносимость системы, ее масштабируемость.

Модели серверов баз данных

В период создания первых СУБД технология клиент - сервер только зарождалась. Поэтому изначально в архитектуре систем не было адекватного механизма организации взаимодействия процессов типа «клиент» и процессов типа «сервер». В современных же СУБД он является фактически основополагающим и от эффективности его реализации зависит эффективность работы системы в целом.

Рассмотрим эволюцию типов организации подобных механизмов. В основном этот механизм определяется структурой реализации серверных процессов, и часто он называется *архитектурой сервера баз данных.*

Первоначально, как мы уже отмечали, существовала модель, у которой управление данными (функция сервера) и взаимодействие с пользователем были совмещены в одной программе. Это можно назвать нулевым этапом развития серверов БД.

Затем функции управления данными были выделены в самостоятельную группу — сервер, однако модель взаимодействия пользователя с сервером соответствовала структуре связей между таблицами баз данных «один к одному» (рисунок 5.7), т.е. сервер обслуживал запросы только одного пользователя (клиента), а для обслуживания нескольких клиентов нужно было запустить эквивалентное число серверов.

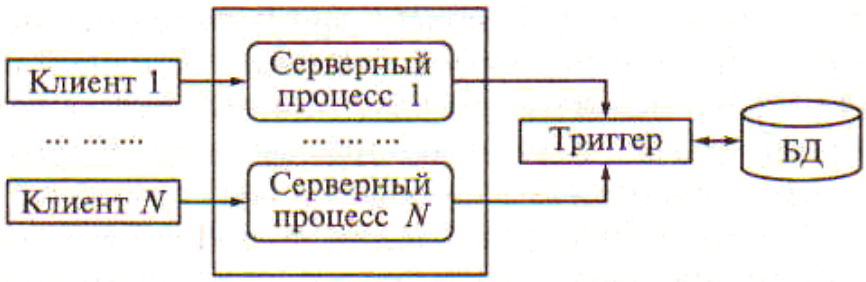


Рисунок 5.7 - Взаимодействие клиентских и серверных процессов в модели «один к одному»

Выделение сервера в отдельную программу было революционным шагом, который позволил, в частности, поместить сервер на одну машину, а программный интерфейс с пользователем — на другую, осуществляя взаимодействие между ними по сети.

Однако необходимость запуска большого числа серверов для обслуживания множества пользователей сильно ограничивала возможности такой системы.

Для обслуживания большого числа клиентов на сервере должно было быть запущено большое число одновременно работающих серверных процессов, а это резко повышало требования к ресурсам ЭВМ.

Кроме того, каждый серверный процесс в этой модели запускался как независимый, поэтому если один клиент сформировал запрос, который был только что выполнен другим серверным процессом для другого клиента, то запрос выполнялся повторно. В такой модели весьма сложно обеспечить взаимодействие серверных процессов. Эта модель самая простая, и она появилась первой.

Проблемы, возникающие в информационной модели «один к одному», решаются в архитектуре систем с выделенным сервером, который способен обрабатывать запросы от многих клиентов. Сервер единственный обладает монополией на управление данными и взаимодействует одновременно со многими клиентами (рисунок 5.8). Логически каждый клиент связан с сервером отдельной нитью или потоком, по которому пересылаются запросы. Такая архитектура получила название *многопотоковой односерверной*.

Она позволяет значительно уменьшить нагрузку на операционную систему, возникающую при работе большого числа пользователей.

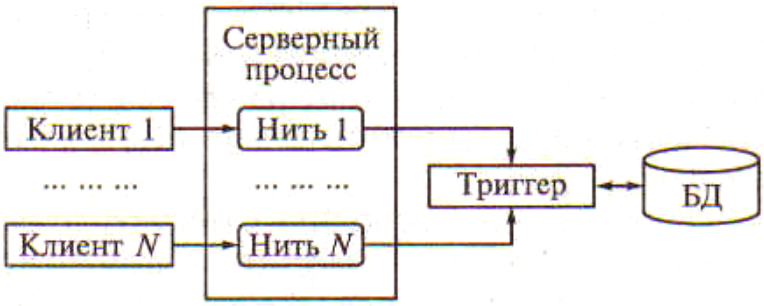


Рисунок 5.8 - Многопотоковая односерверная архитектура

Кроме того, возможность взаимодействия многих клиентов с одним сервером позволяет в полной мере использовать разделяемые объекты (начиная с открытых

файлов и кончая данными из системных каталогов), что значительно уменьшает потребности в памяти и общее число процессов операционной системы.

Например, системой с моделью «один к одному» будет создано 100 копий процессов СУБД для 100 пользователей, тогда как системе с многопотоковой архитектурой для этого понадобится только один серверный процесс.

Однако такое решение имеет свои недостатки. Так как серверный процесс может выполняться только на одном процессоре, возникает естественное ограничение на применение СУБД для мультипроцессорных платформ. Если компьютер имеет, например, четыре процессора, то СУБД с одним сервером используют только один из них, не загружая оставшиеся три.

В некоторых системах эта проблема решается вводом промежуточного диспетчера. Подобная архитектура называется *архитектурой виртуального сервера* (рисунок 5. 9).

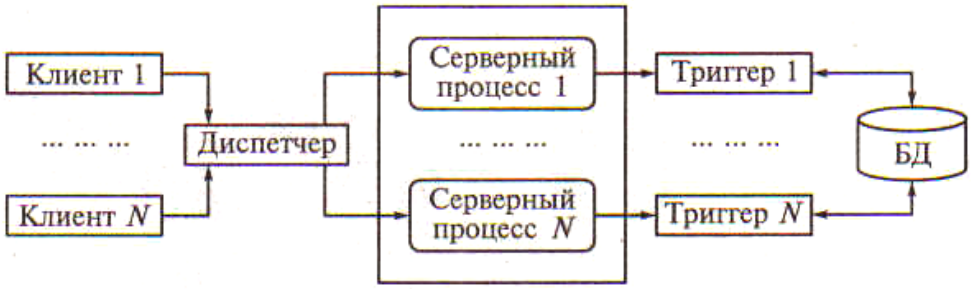


Рисунок 5.9 - Архитектура виртуального сервера

В этой архитектуре клиенты подключаются не к реальному серверу, а к промежуточному звену, называемому диспетчером, который выполняет только функции диспетчеризации запросов к серверам. В этом случае нет ограничений на использование многопроцессорных платформ. Число серверов может быть согласовано с числом процессоров в системе.

Однако и эта архитектура не лишена недостатков, потому что здесь в систему добавляется новый слой, который размещается между клиентом и сервером, что увеличивает потребность в ресурсах на поддержку баланса загрузки серверов и ограничивает возможности управления взаимодействием клиент— сервер. Во-первых, становится невозможным направить запрос от конкретного клиента кон-

кретному серверу; во-вторых, серверы становятся равноправными, так как нет возможности устанавливать приоритеты для обслуживания запросов.

Подобная организация взаимодействия между клиентом и сервером может рассматриваться как аналог банка, где имеется несколько окон кассиров и специальный банковский служащий, администратор зала, который направляет каждого пришедшего посетителя (клиента) к свободному кассиру (актуальному серверу).

Система работает нормально, пока все посетители равноправны (имеют равные приоритеты), однако стоит появиться посетителям с высшим приоритетом, которые должны обслуживаться в специальном окне, как возникают проблемы. Учет приоритета клиентов особенно важен в системах оперативной обработки транзакций, однако именно эту возможность не может предоставить архитектура систем с диспетчеризацией.

Современное решение проблемы СУБД для мультипроцессорных платформ заключается в возможности запуска нескольких серверов базы данных, в том числе и на различных процессорах. При этом каждый из серверов должен быть многопотоковым. Если эти два условия выполнены, то есть основания говорить о многопотоковой архитектуре с несколькими серверами, представленной на рисунке 5.10.

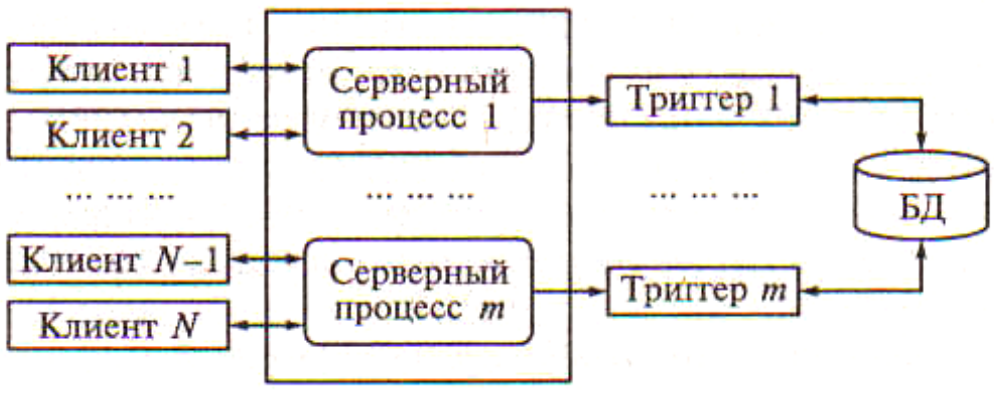


Рисунок 5.10 - Многопотоковая мультисерверная архитектура

Такую архитектуру называют также *многонитиевой мулътисерверной архитектурой.* Эта архитектура обеспечивает распараллеливание выполнения одного пользовательского запроса несколькими серверными процессами.

В этом случае пользовательский запрос разбивается на ряд подзапросов, которые могут выполняться параллельно, а результаты их выполнения потом объединяются в общий результат выполнения запроса. Тогда для обеспечения оперативности выполнения запросов их подзапросы могут быть направлены отдельным серверным процессам, а затем полученные результаты объединены в общий результат (рисунок 5.11).

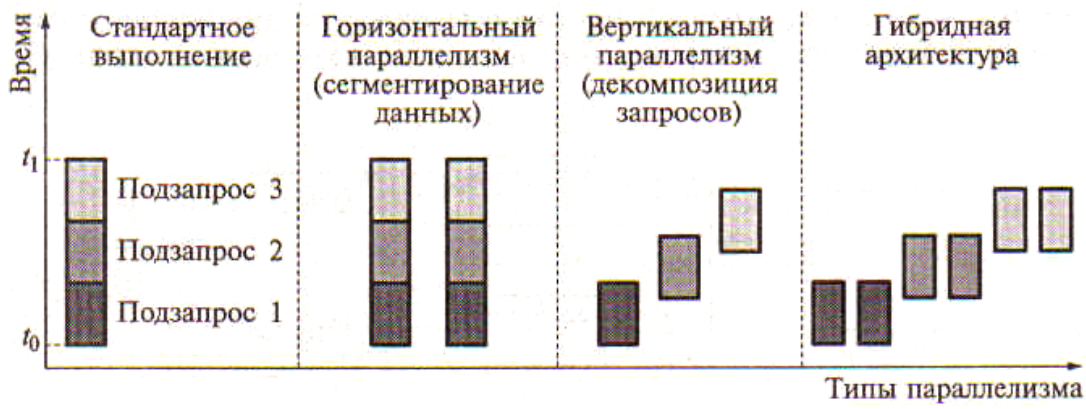


Рисунок 5.11 - Многонитевая мультисерверная архитектура

В данном случае серверные процессы не являются независимыми процессами — такими, как рассматривались ранее. Эти серверные процессы принято называть *нитями*. Управление нитями множества запросов пользователей требует дополнительных расходов от СУБД, однако при оперативной обработке информации в хранилищах данных такой подход наиболее перспективен.

Типы параллелизма

Рассматривают следующие программно-аппаратные способы распараллеливания запросов: горизонтальный, вертикальный и смешанный параллелизм.

Горизонтальный параллелизм. Этот параллелизм возникает тогда, когда хранимая в БД информация распределяется по нескольким физическим устройствам хранения — нескольким дискам. При этом информация из одного отношения разбивается на части по горизонтали. Этот вид параллелизма иногда называют распараллеливанием, или сегментацией, данных. Параллельность достигается путем

выполнения одинаковых операций, например фильтрации, над разными физическими хранимыми данными. Эти операции могут выполняться параллельно разными процессами — они независимы. Результат выполнения целого запроса складывается из результатов выполнения отдельных операций. Время выполнения такого запроса при соответствующем сегментировании данных существенно меньше, чем время выполнения этого же запроса традиционными способами одним процессом.

Вертикальный параллелизм. Этот параллелизм достигается конвейерным выполнением операций, составляющих запрос пользователя. Этот подход требует серьезного усложнения модели выполнения реляционных операций ядром СУБД. Он предполагает, что ядро СУБД может произвести декомпозицию запроса, базируясь на его функциональных компонентах; при этом ряд подзапросов может выполняться параллельно, с минимальной связью между отдельными шагами выполнения запроса.

Общее время выполнения подобного запроса, конечно, будет существенно меньше, чем при традиционном способе выполнения последовательности из четырех операций (рисунок 5.12).

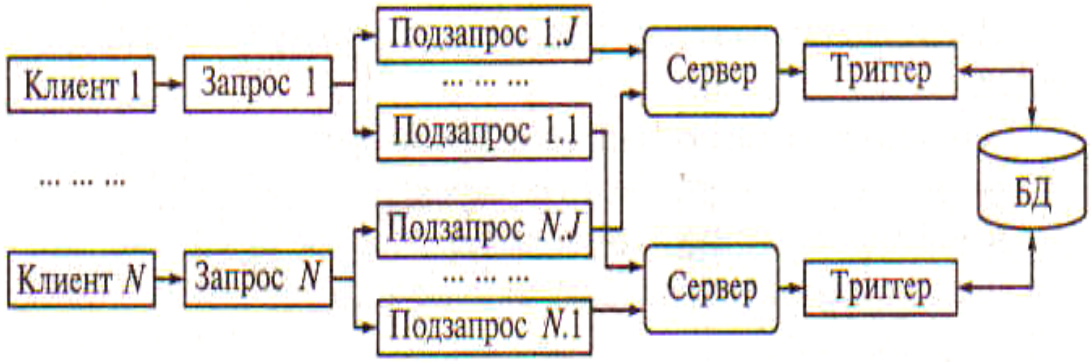


Рисунок 5.12 - Схема выполнения запроса при вертикальном параллелизме

Смешанный параллелизм. Этот параллелизм является гибридом горизонтального и вертикального параллелизма (рисунок 5.13).

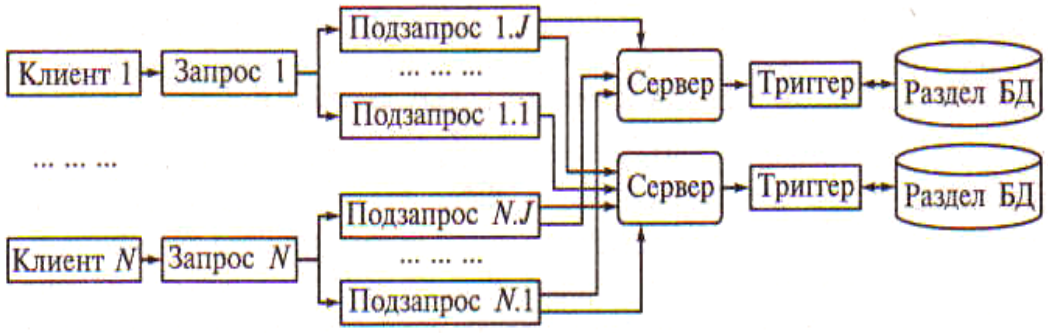


Рисунок 5.13 - Схема выполнения запроса при смешанном параллелизме

Все виды параллелизма применяются в приложениях, где они позволяют существенно сократить время выполнения сложных запросов над очень большими объемами данных.

Вопросы для самоконтроля

1 Дайте определения следующих понятий:

• топология БД, или структура распределенной БД;

• локальная автономность;

• удаленный запрос;

• поддержка распределенной транзакции; « презентационная логика;

• бизнес-логика.

2 Какие двухуровневые модели вы знаете? Назовите их достоинства и недостатки.

3 Назовите характеристики следующих архитектур организации баз данных:

« многопотоковая односерверная архитектура;

• архитектура с виртуальным сервером;

• многонитиевая мультисерверная архитектура.

4 Для чего применяют распараллеливание запросов и какие типы параллелизма вы знаете?

6 Лекция 6. Общая характеристика распределенных баз данных

До сих пор изучались централизованные, локальные базы данных. В то же время распределенные базы данных (РБД) находят все более широкое применение в связи с массовым распространением «сетевых» технологий.

Теория создания, использования и функционирование РБД имеет свои особенности по сравнению с централизованными БД.

В силу сказанного в данной части акцент сделан преимущественно на отличия теории РБД от теории централизованных БД. Отмечено все более широкое распространение режима клиент - сервер, для которого представлена специфика одно- и многоуровневой структуры.

Новые требования, предъявляемые к базам данных

Базы данных явились в значительной мере следствием развития автоматизированных систем управления (АСУ). Первоначально АСУ строились по централизованному принципу: данные из источников передавались в центральный вычислительный центр с суперЭВМ и там обрабатывались. В силу этого базы данных первоначально назывались банками данных.

Сразу же выяснилось, что хотя бы часть данных необходимо обрабатывать непосредственно в источниках данных. Доля децентрализованной обработки данных неуклонно росла.

К концу 80-х годов XX в. возникли новые условия работы для БД: большие объемы информации возникли во многих местах (например, розничная торговля, полиграфическое и другие производства); источником большого количества данных мог быть и центр, но к этим данным требовался быстрый доступ с периферии (географически распределенное производство, работающее по одному графику). К тому же данные могли запрашиваться и центром, и удаленными потребителями. Появилось большое количество данных, которые должны использоваться в срочн-

ных запросах, чаще всего местного характера (продажа авиа- и железнодорожных билетов).

Пример 6.1. В компьютерном интегрированном полиграфическом производстве необходимой является РБД (рисунок 6.1), связывающая в единое целое процесс управления комплексом различных технологических процессов. Здесь осуществляется работа не с одним, а с системой приложений.

Централизованные БД, особенно построенные на классическом подходе, не могли удовлетворить новым требованиям. Возникла потребность в сетях передачи данных и формировании сети удаленных компьютеров, в том числе сети локальных БД, т. е. в распределенных базах данных (РБД).

Быстрое распространение сетей передачи данных, резкое увеличение объема внешней памяти ПК при ее удешевлении в 80-е годы XX в., развитие возможностей персональных компьютеров, которые по своим характеристикам в 90-е годы уже превосходили суперЭВМ 80-х годов, создали необходимую базу для реализации и широкого внедрения РБД.

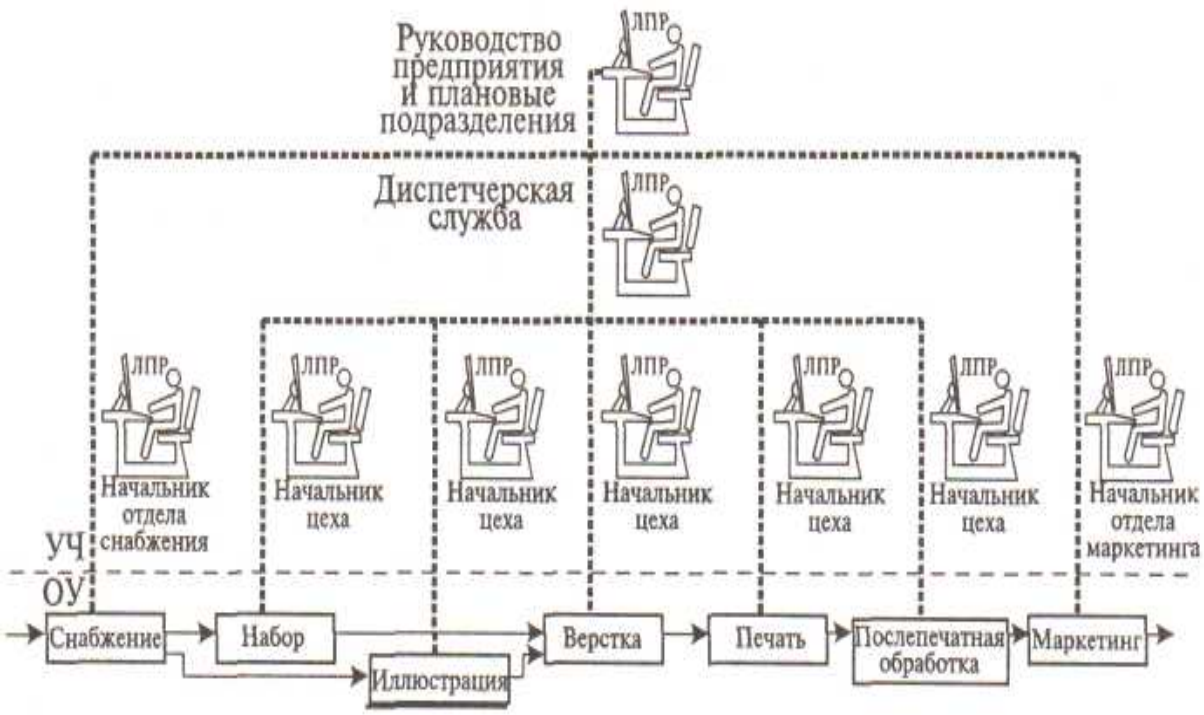


Рисунок 6.1 - Схема компьютерного интегрированного полиграфического производства:

УЧ — управляющая часть; ОУ — объект управления

К достоинствам РБД относятся:

• соответствие структуры РБД структуре организаций;

• гибкое взаимодействие локальных БД;

• широкие возможности централизации узлов;

• непосредственный доступ к информации, снижение стоимости передач (за счет уплотнения и концентрации данных);

• высокие системные характеристики (малое время отклика за счет распараллеливания процессов, высокая надежность);

• модульная реализация взаимодействия, расширения аппаратных средств, возможность использования объектно-ориентированного подхода в программировании;

• возможность распределения файлов в соответствии с их активностью;

• независимые разработки локальных БД через стандартный интерфейс.

Вместе с тем РБД обладают более сложной структурой, что вызывает появление дополнительных проблем (избыточность, несогласованность данных по времени, согласование процессов обновления и запросов, использования телекоммуникационных ресурсов, учет работы дополнительно подсоединенных локальных БД, стандартизация общего интерфейса, усложнение защиты данных) согласования работы элементов.

Серьезные проблемы возникают при интеграции в рамках РБД однородных (гомогенных) локальных БД с одинаковыми, чаще всего реляционными, моделями данных.

Проблемы значительно усложняются, если локальные БД построены с использованием различных моделей данных (неоднородные, гетерогенные РБД).

Возникла необходимость в теоретической проработке процессов в РБД.

Распределенная база данных (РБД) — система логически интегрированных и территориально распределенных БД, языковых, программных, технических и организационных средств, предназначенных для создания, ведения и обработки информации.

Это означает, что информация физически хранится на разных ЭВМ, связанных сетью передачи данных. Любой узел (участок) может выполнять приложение и участвовать в работе по крайней мере одного приложения.

В распределенной базе данных используется следующая специфическая терминология.

Архитектура клиент—сервер — структура локальной сети, в которой применено распределенное управление сервером и рабочими станциями (клиентами) для максимально эффективного использования вычислительной мощности.

Атрибут — характеристика элемента данных в объекте.

Вызов процедуры — передача управления подпрограмме или процедуре с последующим возвратом к основной программе по окончании выполнения подпрограммы или процедуры.

Вызов удаленной процедуры (Remote Procedure Call) — вызов процедуры с другого компьютера.

Инкапсуляция — объединение данных и программы (кода) в «капсуле», модуле.

Класс — объединяющая концепция набора объектов, имеющих общие характеристики (атрибуты).

Компонент — аналог класса в приложении Delphi.

Клиент — компьютер, обращающийся к совместно используемым ресурсам, которые предоставляются другим компьютером (сервером).

Локализация (размещение) — распределение данных по узлам (участкам) сети с учетом дублирования (наличия копий).

Локальная вычислительная сеть (ЛВС) — коммуникационная система, поддерживающая в пределах здания или некоторой другой территории один или несколько скоростных каналов передачи цифровой информации, предоставленных подключенным устройствам для кратковременного монопольного использования.

Метод — набор подпрограмм, оперирующих с данными.

Мост — устройство для соединения двух полностью идентичных подсетей в общую сеть.

Наследование — передача определенных свойств от класса к его производному.

Объект — комбинация элементов данных, характеризующихся атрибутами, и методов их обработки, упакованных вместе в одном модуле.

Полиморфизм — возможность переопределения процедуры в производном классе.

Прозрачность — устройство или часть программы, которая работает настолько четко и просто, что ее действия незаметны пользователю.

Свойство — аналог атрибута в приложении Delphi.

Сервер — узловая станция компьютерной сети, предназначенная в основном для хранения данных коллективного пользования и для обработки в ней запросов, поступающих от пользователей других узлов.

Событие — сигнал запуска метода.

Среда (Computer Aided Software Engineering — CASE) — среда создания программного обеспечения, ориентированная на разработку программы от планирования и моделирования до кодирования и документирования.

Фрагмент логический — блок данных, однородных для транзакций с точки зрения доступа.

Фрагментация (расчленение) — процесс разбиения целостного объекта глобального типа на несколько частей (фрагментов).

Фрагмент хранимый — физическая реализация логического фрагмента.

Шлюз — устройство для соединения разнотипных сетей, работающих по разным протоколам.

Большинство требований, предъявляемых к РБД, аналогично требованиям к централизованным БД, но их реализация имеет свою, рассматриваемую ниже специфику. В частности, в РБД иногда полезна избыточность.

Дополнительными специфическими требованиями являются:

• ЯОД в рамках схемы должен быть один для всех локальных БД;

- доступ должен быть коллективным к любой области РБД с соответствующей защитой информации;
- подсхемы должны быть определены в месте сосредоточения алгоритмов (приложений, процессов) пользователя;
- степень централизации должна быть разумной;
- необходим сбор и обработка информации об эффективности функционирования РБД.

В дальнейшем Кристофер Дейт сформулировал 12 правил для РБД.

1 Локальная автономность.

2 Отсутствие опоры на центральный узел.

3 Непрерывное функционирование (развитие) РБД.

4 Независимость РБД от расположения локальных БД.

5 Независимость от фрагментации данных.

6 Независимость от репликации (дублирования) данных.

7 Обработка распределенных запросов.

8 Обработка распределенных транзакций.

9 Независимость от типа оборудования.

10 Независимость от операционной системы.

11 Независимость от сетевой архитектуры.

12 Независимость от типа СУБД.

В дальнейшем рассмотрении РБД выделим:

- общие вопросы (состав, работа РБД);
- теоретические вопросы РБД, в которых по-прежнему выделим три составляющие (создание, использование и функционирование РБД).

Состав и работа РБД

Схема РБД может быть представлена в виде, показанном на рисунке 6.2. В ней выделяют пользовательский, глобальный (концептуальный), фрагментарный

(логический) и распределенный (локальный) уровни представления данных (рисунок 6.3), определяющие сетевую СУБД.

Общий набор (система таблиц) данных, хранимых в РБД, приведен в таблице 6.1. Это глобальный уровень, который определяется при проектировании теми же методами, что и концептуальная модель централизованной БД.

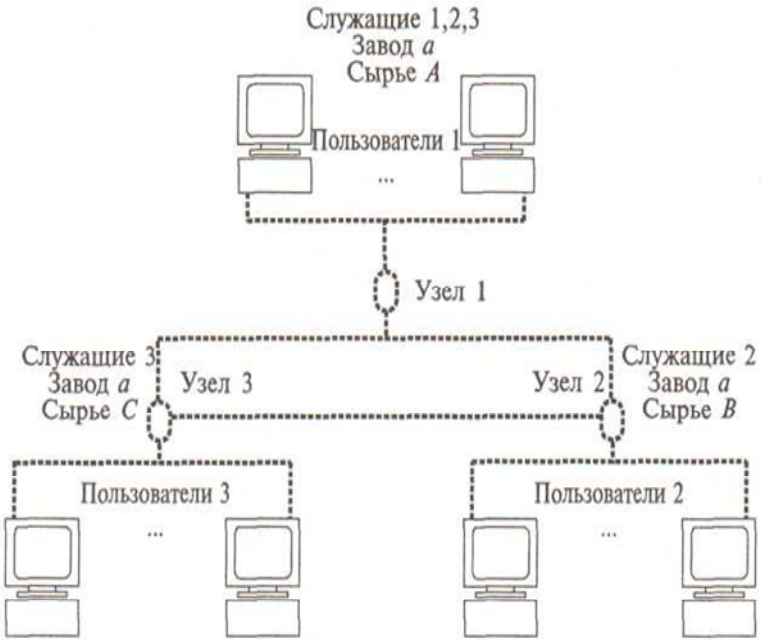


Рисунок 6.2 - Схема РБД

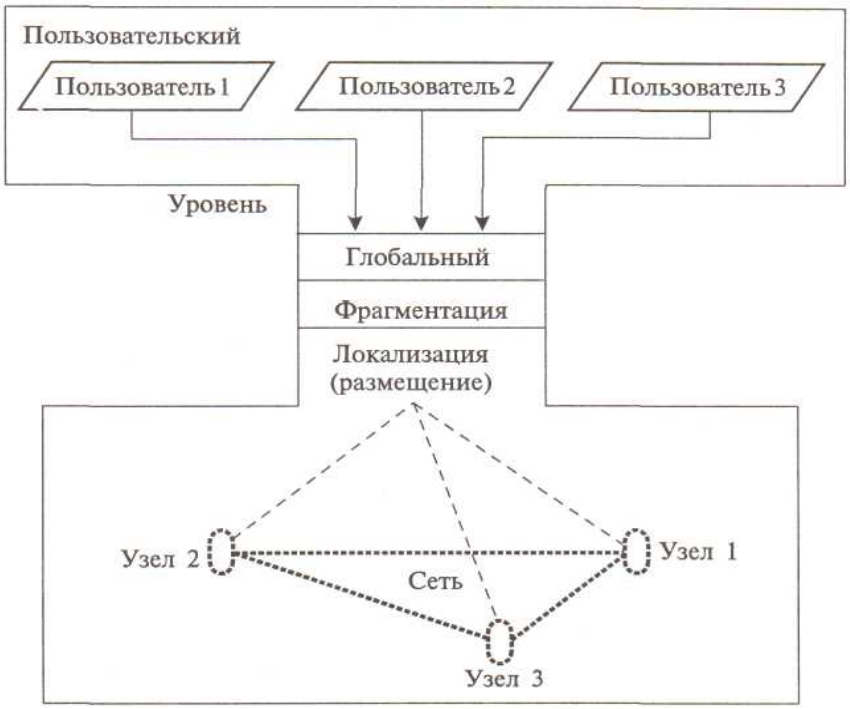


Рисунок 6.3 - Уровни представления данных в РБД

Таблица 6.1 - Данные в РБД (глобальный уровень) «Служащий»

N	Имя	Завод	Тариф
100	Иванов	1	6
101	Петров	1	6
102	*Сидоров*	*2*	*10*
103	*Артемов*	*2*	*12*
104	**Печкин**	**3**	**5**
105	**Крамов**	**3**	**11**

В таблице 6.1 разными шрифтами выделены фрагменты БД.

Не все данные глобального уровня доступны конкретному пользователю. Наиболее полные данные (пользовательский, внешний уровень) имеются в узле 1 головного предприятия. В узлах (участках) 2, 3 данные менее полные. Так, в узле 3 они имеют вид, показанный в таблице 6.2.

Таблица 6.2 – Пользовательский уровень в РБД

а) Служащий

N	Имя	Завод	Тариф
104	**Печкин**	**3**	**5**
105	**Крамов**	**3**	**11**

б) Завод

N	Расположение
1	С.-Петербург
2	Вологда
3	Сыктывкар

в) Сырье

N	Название	Количество
3	**Брошюры**	**75**

Пользовательский уровень состоит из фрагментов (например, строки 1, 2, 3 таблица «Завод» таблица 6.1) глобального уровня, которые составляют фрагментарный, логический уровень.

Выделяют горизонтальную и вертикальную фрагментации (расчленение). Горизонтальная фрагментация связана с делением данных по узлам. Горизонтальные фрагменты не перекрываются. Вертикальная фрагментация связана с группированием данных по задачам.

Фрагментация чаще всего не предполагает дублирования информации в узлах. В то же время при размещении фрагментов по узлам {локализации) распределенного уровня в узлах разрешается иметь копии той или иной части РБД. Так, например, локализация для примера в таблице 6.1 может иметь вид, показанный в таблице 6.3.

Таблица 6.3 Локализация данных

Имя таблицы	Распределение фрагментов по узлам
Служащие	1
	1, 2
	1, 3
Завод	1, 2, 3
	1, 2, 3
	1, 2, 3
Сырье	1
	2
	3

Очевидно, что для таблицы «Завод» осуществляется дублирование, а для таблицы «Сырье» — расчленение.

После размещения данных каждый узел имеет локальное, узловое представление (локальная логическая модель). Физическую реализацию (логического) фрагмента называют хранимым фрагментом.

Иначе говоря, РБД можно представить в виде, показанном на рисунке 6.3.

Сеть в РБД образуют сетевые операционные системы (например, Windows NT, Novell NetWare). В качестве СУБД, изначально предназначавшихся для использования в сети, следует назвать BTrieve, Oracle, InterBase, Sybase, Informix.

В силу распределенности данных особую значимость приобретает словарь данных (справочник) РБД, который в отличие от словаря централизованной БД имеет распределенную, многоуровневую структуру.

В общем случае могут быть выделены сетевой, общий внешний, общий концептуальный, локальные внешние, локальные концептуальные и внутренние составляющие словаря РБД.

Естественно, что для работы в РБД необходимы администраторы РБД и локальных БД, рабочими инструментами которых являются перечисленные словари.

Схема работы РБД показана на рисунке 6.4.

Пользовательский запрос, определяемый приложением, поступает в систему управления распределенной базы данных (СУРБД), через сетевую и локальную операционные системы попадает в локальную СУБД.

Рисунок 6.4 - Схема работы РБД

Если запрос связан с локальными данными, СУБД осуществляет вызов данных из локальной БД, которые поступают пользователю. Если часть данных для выполнения приложения находится в другой локальной БД, локальная СУБД дополнительно через локальные и сетевую операционные системы осуществляет удаленный вызов процедуры (Remote Procedure Call — PRC), после выполнения которой данные передаются пользователю.

Возможны четыре стратегии хранения данных: централизованная (часто обеспечиваемый архитектурой клиент—сервер), расчленение (фрагментации), дублирование, смешанная.

Простейший алгоритм выбора стратегии приведен на рисунке 6.5.

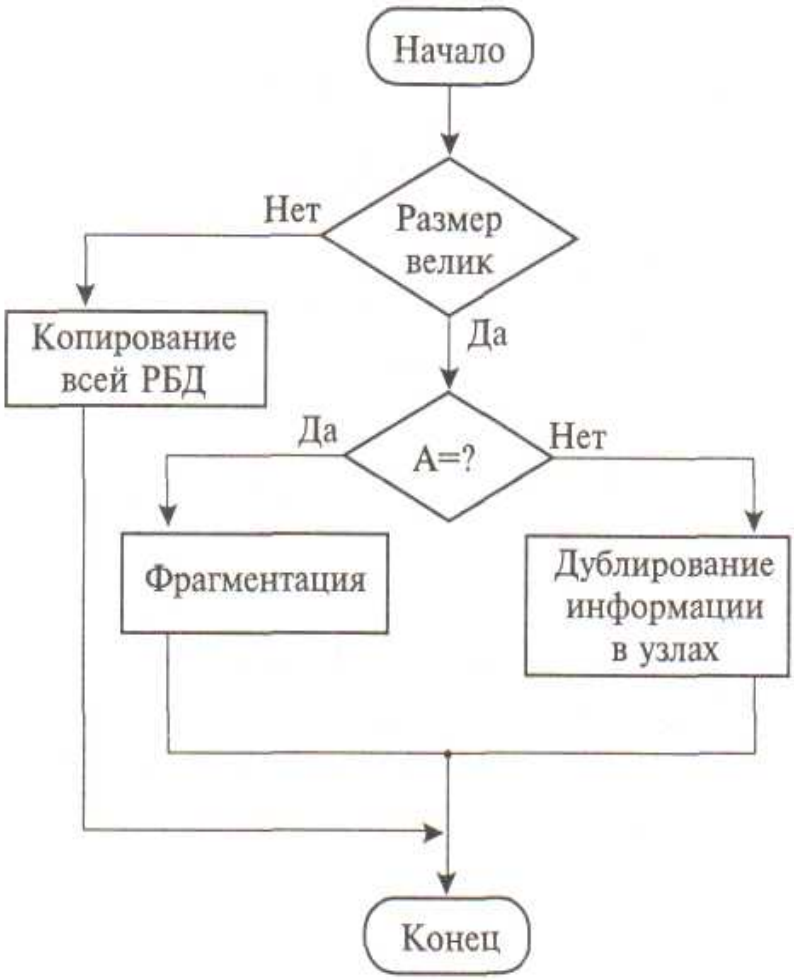


Рисунок 6.5 - Алгоритм выбора стратегии хранения (А — запрос локален)

Отметим, что в обычной сети имеет место равноправие компьютеров, что может вызвать дополнительные осложнения в части доступа к данным в процедурах обновления и запросов.

В связи с этим часто используют архитектуру клиент—сервер (рисунок 6.6) — структуру локальной сети, в которой применено распределенное управление сервером и рабочими станциями (клиентами) для максимально эффективного использования вычислительной мощности.

В этой структуре один из компьютеров, имеющий самый большой объем памяти и наиболее высокое быстродействие, становится приоритетным, называемым

сервером. На сервере чаще всего хранятся только данные, запрашиваемые клиентами.

К клиентам не предъявляются столь жесткие требования по памяти и быстродействию. На них располагаются словари и приложения, служащие своеобразными фильтрами для данных сервера. В связи с этим обмен информацией в архитектуре (рисунок 6.6) фактически минимизируется.

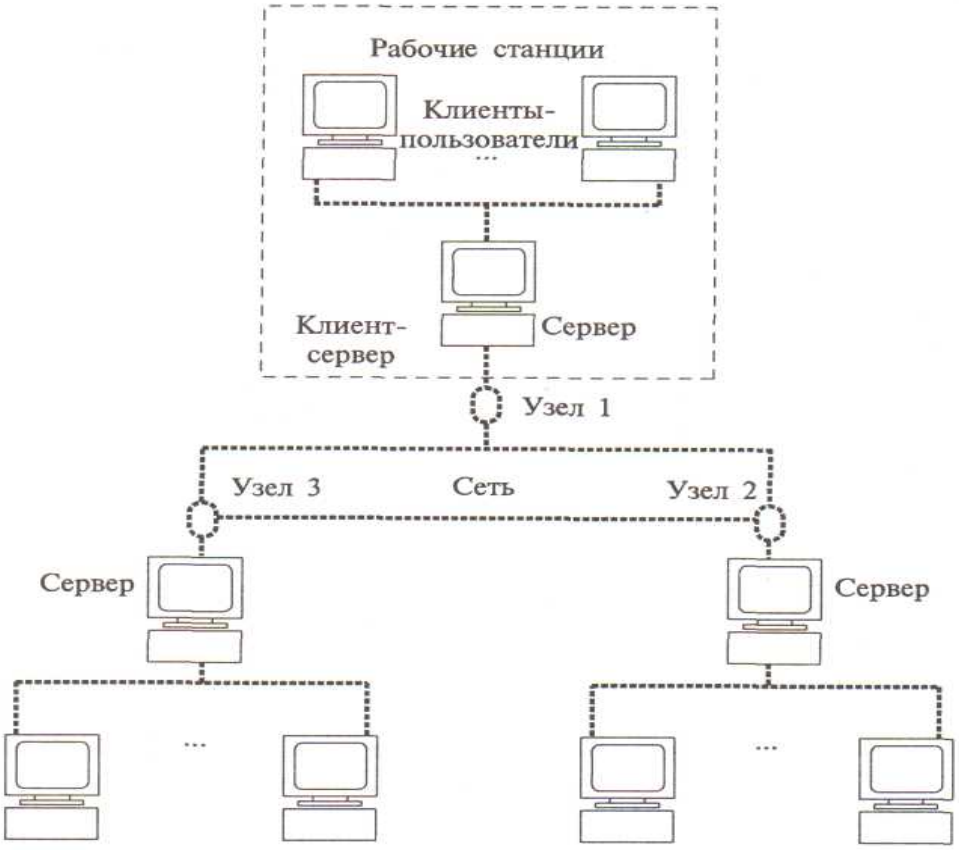


Рисунок 6.6 - Архитектура клиент—сервер

Работа в архитектуре клиент—сервер может поддерживаться и с помощью схемы Open DataBase Connectivity (ODBC), как показано на рисунке 6.7. В этом случае сеть образуется путем соединения серверов. Такое соединение обеспечивается или средствами СУБД (SQL Server) или мониторами транзакций (TUXEDO).

Обсудим более подробно вариант реализации РБД — архитектуру клиент—сервер.

Рисунок 6.7 - ODBC в архитектуре клиент—сервер

Система клиент - сервер

Совместно с термином «клиент—сервер» используются три понятия.

1 Архитектура: речь идет о концепции построения варианта РБД.

2 Технология: говорят о последовательности действий в РБД.

3 Система: рассматриваются совокупность элементов и их взаимодействие.

Об архитектуре клиент—сервер говорилось ранее. Технология клиент—сервер позволяет повысить производительность труда:

• сокращается общее время выполнения запросов за счет мощного сервера;

• уменьшается доля и увеличивается эффективность использования клиентом (для вычислений) центрального процессора;

• уменьшается объем использования клиентом памяти «своего» компьютера;

• сокращается сетевой трафик.

К таким крупномасштабным системам предъявляются следующие требования:

1) гибкость структуры;

2) надежность;

3) доступность данных;

4) легкость обслуживания системы;

5) масштабируемость приложений;

6) переносимость приложений (на разные платформы);

7) многозадачность (возможность выполнения многих приложений).

Отметим, что архитектуре клиент—сервер предшествовала архитектура файл—сервер, в которой возможны следующие варианты.

1. На компьютерах-клиентах имеется копия БД. Работа по такому варианту имеет следующие сложности: синхронизация данных различных копий в конце работы БД; высокий трафик (потоки данных между сервером и клиентами, поскольку передается в любом случае содержимое всей БД).

2. В СУБД Access, которая изначально создана как локальная, предусмотрен режим деления базы данных. Таблицы остаются на сервере (back-end), а остальные объекты (запросы, отчеты) передаются клиентам (front-end). В этом случае попрежнему большой трафик, в силу чего при использовании файл-серверов количество подключаемых клиентов — при их надежной работе — до четырех.

В то же время требовалось подключение десятков и даже сотен клиентов. Этого удалось достичь в архитектуре (режиме, технологии) клиент - сервер. В этом режиме трафик резко уменьшается, поскольку по сети передаются только те данные, которые соответствуют запросам клиентов.

Для этого пришлось построить СУБД, изначально предназначенные для работы в сети. Фирма Microsoft вынуждена была — в дополнение к СУБД Access, которая использовалась с помощью приложения ODBC только для клиентских целей — предложить в качестве сервера Microsoft SQL Server. Такая структура оказалась тяжеловесной и неудобной, так как разработчику требовалось знать уже две СУБД.

Из других предложений очень удачным оказался программный продукт Delphi, в рамках которого могут использоваться СУБД dBase, Paradox, и, при отдельной инсталляции, InterBase (режим клиент - сервер). При этом СУБД InterBase поддерживается, наряду с языком программирования SQL, мощным, понятным, про-

стым и широко распространенным языком программирования (Object) Pascal, построенным с применением объектно-ориентированного подхода.

Последнее обстоятельство позволяет легко строить объектно-реляционные базы данных. Высокая степень автоматизации программирования дает возможность резко упростить и снизить трудоемкость процедур создания интерфейса пользователя, и особенно алгоритма приложения.

В системе клиент - сервер возможно выделить следующие составляющие: сервер, клиент, интерфейс между клиентом и сервером, администратор.

Сервер осуществляет управление общим для множества клиентов ресурсом. Он выполняет следующие задачи:

- управляет общей БД;
- осуществляет доступ и защиту данных, их восстановление;
- обеспечивает целостность данных.

К БД на сервере предъявляются те же требования, как и к централизованной многопользовательской БД.

Следует отметить, что результаты запросов клиента помещаются в рабочую область памяти сервера, которая в ряде СУБД (например, Oracle) называют «табличная область». Поскольку она не занимает много места, для каждого клиента-пользователя целесообразно создавать свою табличную область. В этом случае исходные таблицы становятся для пользователя недоступными, а архивация (копирование) БД приложения клиента упрощается.

Клиент хранит в компьютере свои приложения, с помощью которых осуществляется запрос данных на сервере. Клиент решает следующие задачи:

- предоставляет интерфейс пользователю;
- управляет логикой работы приложений;
- проверяет допустимость данных;
- осуществляет запрос и получение данных с сервера.

Средством передачи данных между клиентом и сервером является сеть (коаксиальный кабель, витая пара) с сетевым (сетевая операционная система — СОС) и коммуникационным программным обеспечением.

В качестве СОС могут использоваться Windows NT, Novell NetWare. Коммуникационное программное обеспечение позволяет компьютерам взаимодействовать на языке специальных программ — коммуникационных протоколов.

В общем случае такое взаимодействие осуществляется с помощью семиуровневой схемы ISO с соответствующими протоколами. Для локальных сетей схема упрощается. Протоколоми для Windows NT служит Transmission Control Program/Internet Program (TCP/IP), для NetWare — Sequenced Packed eXchange/Internet Packed eXchaned (SPX/IPX).

Разнообразие сетевых средств делает необходимым создание стандартного промежуточного программного обеспечения клиент—сервер, находящегося на сервере и клиентах. Говорят о прикладном программном интерфейсе (Application Programming Interface — API). Сюда относятся Open DataBase Connectivity (ODBC) и Integrated Database Application Programming Interface (IDAPI), используемый в приложении Delphi и СУБД InterBase.

Взаимодействие клиентов и сервера можно представить себе следующим образом.

При обращении пользователя к приложению компьютер-клиент запрашивает у пользователя имя и пароль. После этого — при правильном ответе — приложение может быть запущено клиентом. Приложение дает возможность подключиться к серверу, которому сообщается имя и пароль пользователя.

Если подключение осуществлено, начинает работать сервер, выполняющий два вида процессов: переднего раздела и фоновые.

Процессы переднего раздела непосредственно обрабатывают запросы, фоновая составляющая связана с управлением процессом обработки.

Работа сервера может иметь такой порядок.

1 После поступления запроса диспетчер ставит его в очередь по

схеме «первым пришел — первым обслужен».

2 Процесс переднего раздела выбирает «самый старый» запрос и начинает его обработку. После завершения результаты помещаются в очередь для передачи клиенту.

3 Диспетчер посылает результаты из очереди соответствующему клиенту.

При обработке запроса фоновые процессы выполняют другие важные операции, основными из которых являются следующие:

- запись данных из БД в промежуточную (буферную) память рабочей области (при чтении) и обратно (при обновлении);
- запись в журнал транзакций;
- архивация (копирование) групп транзакций;
- аварийное завершение транзакций;
- периодическая запись на диск контрольных точек для обеспечения восстановления данных в РБД после аппаратного сбоя.

Администратор РБД (АРБД) должен решать следующие задачи.

1 Планирование РБД и распределение памяти.

2 Настройка конфигурации сети.

3 Создание РБД.

4 Работа с разработчиками приложений.

5 Создание новых пользователей и управление полномочиями.

6 Регулярная архивация БД и выполнение операций по ее восстановлению.

7 Управление доступом к БД с помощью ОС и СОС, средств защиты и доступа.

В больших системах АРБД может состоять из ряда лиц, отвечающих, например, за ОС, сеть, архивацию, защиту.

Таким образом, система клиент - сервер своеобразна: с одной стороны, ее можно считать разновидностью централизованной многопользовательской БД, с другой стороны, она является частным случаем РБД.

В связи с этим имеется специфика и в процессе проектирования. Оно по-прежнему начинается с создания приложения, затем — интерфейса и БД. Однако в силу специфики системы этапы фрагментации и размещения отсутствуют и есть свои особенности.

Основное ограничение для работы такой системы — минимальный трафик. Поэтому при разработке приложения, помимо обычных задач (уяснения цели приложения, логики обработки, вида интерфейса) особое внимание следует обратить на разработку DLL-сценария и распределение функций между клиентами и сервером.

Использование для составления сценария CASE-средств значительно сокращает трудоемкость работ по проектированию. Иначе эта процедура выполняется вручную с помощью команд языка SQL.

Важнейшей является задача распределения функций. По самой сути технологии на сервере расположена БД, а на компьютерах-клиентах — приложения. Однако при прямолинейных процедурах обеспечения целостности и запросах в сети может возникнуть объемный сетевой трафик.

Чтобы его снизить, возможно использовать следующие рекомендации.

1 Обеспечение целостности для всех приложений лучше централизовать и осуществлять на сервере. Это позволит не только сократить трафик, но и рационально использовать СУРБД, улучшив управление целостностью (ссылочной, ограничений, триггеров) данных.

2 Целесообразно использовать на сервере хранимые процедуры, совокупность которых можно инкапсулировать в виде пакета (модуля). В результате трафик уменьшится: клиент будет передавать только вызов процедуры и ее параметры, а сервер — результаты выполнения процедуры.

3 В ряде случаев клиентам следует получать уведомления базы

данных (например, заведующему складом — о нижнем уровне запасов, при котором следует выполнять новый заказ). Если уведомление производится по запросу клиента, трафик увеличивается. Проще эту (хранимую) процедуру разместить на сервере, который будет автоматически уведомлять клиента о возникновении события. В то же время клиент при необходимости может получать информацию с помощью простых вызовов процедур.

В режиме клиент - сервер выделяют две разновидности структуры: одноуровневую, о которой шла речь до сих пор, и многоуровневую (рисунок 6.8).

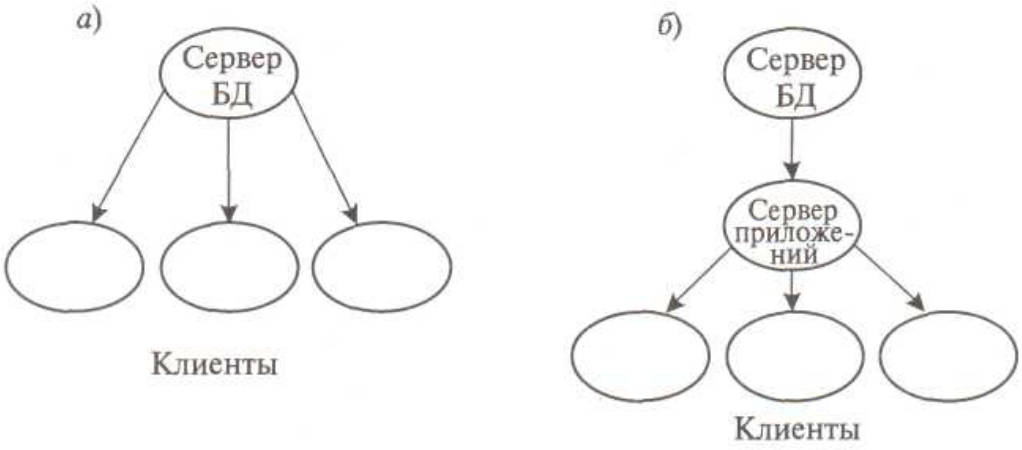


Рисунок 6.8 - Одноуровневая — «толстый» клиент (а) и многоуровневая — «тонкий» клиент (б) структуры клиент—сервер

При использовании режима «толстого» клиента (одноуровневая структура) клиентские приложения находятся непосредственно на машинах пользователей, либо частично на сервере в виде системы хранимых процедур. Сложность клиентской части порой требует ее администрирования. Использование системы хранимых процедур в значительной мере снижает нагрузку на сеть. При использовании «тонкого» клиента (многоуровневая структура) приложения пользователей лежат и выполняются на отдельном мощном сервере (сервере приложений, который может быть и на одном компьютере с сервером БД), что в значительной мере снижает требования к пользовательским машинам.

Специфические этапы создания РБД — это **фрагментация и локализация.** Проектные решения на этих этапах неоднозначны, и в ряде случаев (при выборе структуры РБД) полезно использовать математическое моделирование.

РБД состоит из связанных локальных баз данных (ЛБД).

Возможны два варианта создания РБД: проектирование РБД «с нуля»; объединение (интеграция) уже готовых ЛБД.

В первом варианте модели данных ЛБД, как правило, одинаковы (однородные ЛБД), во втором модели данных могут быть различны (неоднородные ЛБД).

Рассмотрена теория интеграции для однородных и неоднородных ЛБД как совокупности процессов описания и манипулирования данными. Представлены правила преобразования моделей данных друг в друга.

Обеспечение целостности

Будем придерживаться порядка изложения в соответствии с этапами проектирования. В качестве принципов проектирования можно использовать:

• максимум локализации данных и сокращение количества пересылаемых по кратчайшему пути данных: рекомендуется иметь до 90% ее в локальной БД (ЛБД) узла и около 10% — в ЛБД других узлов;

• локальность расположения данных следует определять по отношению к наибольшему числу приложений.

В качестве критериев проектирования РБД могут быть:

1) минимум объема пересылаемых данных и сообщений;

2) минимум стоимости трафика;

3) минимум общего времени, необходимого для обслуживания запросов к БД.

В рассмотрении РБД возможно выделить два случая работы: с одним приложением и с системой приложений. Возможно нисходящее и восходящее проектирование.

Восходящее проектирование используется обычно в случае, когда РБД создается из уже работающих локальных БД. Эти особенности освещены в проблеме интеграции однородных и неоднородных БД.

Восходящее проектирование может быть этапом нисходящего проектирования, которое, в силу того что оно применяется гораздо чаще, обсудим подробнее.

В общем случае целостность данных может нарушаться по следующим основным причинам:

• ошибки в создании структуры локальных БД и их заполнении;

• просчеты в построении структуры РБД (процедуры фрагментации и локализации);

• системные ошибки в программном обеспечении взаимодействия локальных БД (одновременный доступ);

• аварийная ситуация (неисправность технических средств) и восстановление РБД.

Первая позиция подробно освещена ранее и здесь рассматриваться не будет. Специфика остальных трех позиций для РБД может быть зафиксирована в виде совокупности проблем (рисунок 6.9).

Рисунок 6.9 – проектирование РБД

Иногда к нарушению целостности относят умышленное искажение информации, т. е. несанкционированный доступ.

Фрагментация и локализация

Общая этапность проектирования РБД напоминает этапность при создании централизованной БД и отличие имеет место лишь в этапах фрагментации (расчленения) и локализации (размещения).

Основными факторами, определяющими методику расчленения, являются допустимый размер каждого раздела; модели и частоты использования приложений; структурная совместимость; факторы производительности БД. Связь между разделом БД и приложениями характеризуется идентификатором типа приложения, идентификатором узла сети, частотой использования приложения и его моделью.

Сложность реализации этапа размещения БД определяется многовариантностью. Поэтому на практике рекомендуется в первую очередь рассмот-

реть возможность использования определенных допущений, упрощающих функции СУРБД (например, допустимость временного рассогласования БД, осуществление процедуры обновления БД из одного узла).

Фрагментация, как отмечалось ранее, может быть горизонтальной и вертикальной. Фрагмент может быть определен последовательностью операции селекции и проекции реляционной алгебры. При декомпозиции следует выполнить ряд условий.

1 **Полнота** — все данные глобального отношения R должны быть отображены в его фрагменты.

2 **Восстанавливаемость** — всегда возможно восстановить глобальное отношение из фрагментов.

3 **Непересечение** — целесообразно, чтобы фрагменты не пересекались (дублирование производится на этапе локализации).

При горизонтальной фрагментации с помощью селекции любое подмножество кортежей объединено общностью свойств, определяемых описанием предметной области.

Вертикальная фрагментация с помощью проекции делит глобальное отношение (схему R) по приложениям (или по географическому признаку).

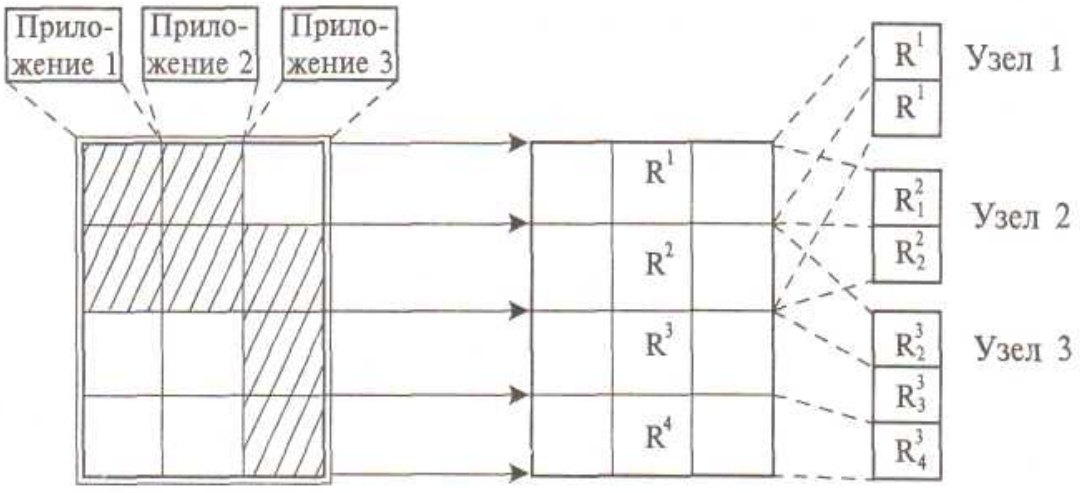


Рисунок 6.10 - Схема фрагментации и локализации данных

Фрагментация корректна, если любой атрибут глобального отношения (схемы R) присутствует в каком-либо подмножестве атрибутов и глобальное отношение восстанавливается естественным соединением.

Фрагментация совместно с локализацией (рисунок 6.10) определяет в конечном итоге быстроту реакции РБД на запрос.

Обозначим узлы через j(j = l, J), а приложения — через к (к = 1, К). Если имеется отношение г со схемой R, то в узле j имеется отношение-фрагмент R^j. Если в узле j имеется к тому же копия фрагмента Ri, i(i = 1,I), то обозначим ее через Ri^j.

При фрагментации декомпозиция глобального отношения должна обладать свойством соединения без потерь.

Возможна смешанная фрагментация, которой соответствует совокупность операций селекции и проекции.

$$R_i^{jk} = \sigma_{A = F}(\pi_{A, B, C}(R)),$$

где $F = \{a_{ij}., ..., a_{mj}.\}$;

m — число записей в горизонтальном фрагменте.

После фрагментации осуществляют локализацию. В качестве критерия локализации удобно использовать влияние локализации (размещения) на задачу оптимизации запроса при известной его структуре. Для этого необходимы моделирование и оптимизация всех приложений для любого варианта размещения.

Воспользуемся обозначениями, введенными в данной главе.

Пусть f_{kj} — частота активизации приложения к в узле j; $г_к.$ — число ссылок поиска приложения к во фрагменте i; $u_k.$ — число ссылок обновления данных приложения к во фрагменте i; $п_к. = г_к. + и_к.$.

Задача размещения имеет две основные разновидности: без использования и с использованием копий.

Рассмотрим первую разновидность. Здесь возможны эвристические и строго математические алгоритмы.

Обсудим сначала один из эвристических алгоритмов размещения, состоящий из нескольких шагов.

Шаг 1 Используем метод «наиболее подходящего» размещения: фрагмент R. размещаем в узле j, где число ссылок на него максимально.

Число локальных ссылок

$$B_{ij} = \sum_{k=1}^{K} f_{kj} n_{ki} \rightarrow \max . \qquad (6.1)$$

Из выражения (6.1) определяем узел j*, где следует разместить фрагмент.

Шаг 2 Применим метод выделения «всех выгодных узлов» для избыточного размещения: помещаем R.во всех узлах, где стоимость ссылок приложений, осуществляющих поиск, больше стоимости ссылок приложений, обновляющих данные во фрагменте R. в любом узле: B.. > 0

или

$$\sum_{k=1}^{K} f_{kj} r_{ki} > C \sum_{k=1}^{K} \sum_{\substack{j'=1, \\ i \neq j}}^{J} f_{kj'} u_{ki}, \qquad (6.2)$$

где C — отношение стоимости обновления и поиска.

Шаг 3. Используем метод «добавочного копирования».

Пусть dj — степень избыточности $R_{;}$; $F_{;}$ — выгодность размещения копии в любом узле РБД и $\beta(d_i) = (1 — 2^{1-d})F_i$. Модифицируя выражение (6.2), получим

$$B_{ij} = \sum_{k=1}^{\cdot\cdot} f_{kj} r_{ki} - C \sum_{k=1}^{\cdot\cdot} \sum_{\substack{j'=1, \\ i \neq j}}^{\cdot} f_{kj'} u_{ki} + \beta(d_i) > 0. \qquad (6.3)$$

Учтем вертикальную фрагментацию.

Пусть схема R. декомпозирована на R_s и R_t, где s и t — узлы.

Используем следующие рассуждения.

1 Если существует два приложения $П_s$ и $П_t$, которые используют только атрибуты R_s и R_t, т.е. обращаются только к узлам s и t, то результатом фрагментации и локализации будет отсутствие удаленных ссылок.

2 Если имеется приложение (множество приложений) $П_{q1}$,, локальное для узла w, которое ссылается на R_s или R_t, то появится одна удаленная ссылка.

3 Если имеется приложение Пq2, локальное по отношению к w и ссылающееся на атрибуты Rs и Rt, то получатся две удаленные ссылки.

4 Если имеется приложение Пq3 в узлах, отличных от w, s или t, и ссылающееся на Rs и Rt, то появится еще одна удаленная ссылка.

В общем случае выгодность фрагментации и локализации (при C=1)

$$B_{ist} = \sum_{k \in П_s} f_{ks} n_{ki} + \sum_{k \in П_t} f_{kt} n_{ki} - \sum_{k \in П_{q1}} f_{kw} n_{ki} - \sum_{k \in П_{q2}} f_{kw} n_{ki} - \sum_{\substack{k \in П_{q3}, \\ i \neq w,s,t}} f_{ki} n_{ki} > 0.$$

Описанные эвристические алгоритмы могут не дать рационального решения, поэтому рассмотрим некоторые строгие алгоритмы, базирующиеся на целочисленном программировании.

Введем обозначения: i (i = 1,I) — независимые файлы-данные; j(j = l,I) — узлы; L_i. — объем файла; b_j. — объем памяти узла (для файлов); d_{sj}. — коэффициенты, учитывающие расстояние между узлами s(s = l, I) и j (d_{ss}= 0); r_{sj} — стоимость передачи; λ_{ij} — интенсивность запросов к файлу i из узла j; α_{ij} — интенсивность корректировки сообщений; a — объем запросов к файлу i из узла j; β_{ij} — объем запрашиваемых данных при выполнении запроса i из узла j;

$$x_{ij} = \begin{cases} 1, \text{ если файл i находится в узле j,} \\ 0, \text{ в остальных случаях.} \end{cases}$$

Тогда объем данных, поступающих в узел j, содержащий файл i, при выполнении запроса к этому файлу с учетом интенсивности равен

$\lambda_{ij} (a_{ij} + \beta_{ij})(1 — x_{ij})$, а объем данных, составляющих запросы и ответы,

$\lambda_{ij} (a_{ij} + \beta_{ij})(1 — x_{ij}) x_{ij}$.

Возможны следующие критерии: объем передаваемых данных; общая стоимость трафика. При использовании первого критерия получаем следующую задачу целочисленного программирования:

$$F = \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{s=1}^{J} \lambda_{ij}\left(\alpha_{ij} + \beta_{ij}\right) d_{sj}\left(1 - x_{ij}\right) x_{ij} \rightarrow \min;$$

$$\sum_{j=1}^{J} x_{ij} = 1;$$

$$\sum_{i=1}^{I} L_i x_{ij} \le b_j.$$

Аналогичные выражения получаются при использовании второго критерия.

«Суммируя» представленные задачи, можно решить задачу размещения и определить количество копий.

Более детальные результаты получены с помощью теории массового обслуживания и целочисленного программирования.

Вопросы для самоконтроля

1 Сформулируйте 12 правил Кристофера Дейта для РБД;

2 Перечислите достоинства РБД;

3 Поясните состав и работу РБД;

4 В чем отличие «толстого» клиента от «тонкого» клиента?

5 Дайте определение фрагментации и локализации.

7 Лекция 7. Пример разработки проекта РИС предметной области

Поставим целью лекции: демонстрация приемов и правил создания и разработки распределенной базы данных в SQL Server 2005 и программной системы приложения на базе инструментальной среды Microsoft Visual Studio 2008 и языка программирования C#.

Для достижения цели сформулируем конкретные задачи:

– создать таблицы, определить связи между таблицами и обеспечить ссылочную целостность, создать представления;

– разработать функции, триггеры и роли;

– разработать клиентское приложение пользователя базы данных;

– разработать руководство по эксплуатации системы баз данных.

Описание предметной области

Организация (Код, Название, Краткое название, Адрес, Контактные телефоны, Электронный адрес) сдает в аренду помещения. Каждое помещение характеризуется следующими показателями:

Адрес, площадь (кв.м.), площадь подвала (кв.м.) при наличии, коэффициент подвала (значение от 0 до 1), коэффициент технического обустройства помещения (КТ) – значение от 1 до 2.

Арендная плата зависит от базовой ставки за 1 кв.м. (в рублях), которая утверждается документом (Номер, Дата) агентства Госкомимущества России.

Формула расчета месячной арендной платы (МАП):

МАП = (базовая ставка/12 * площадь помещения + базовая ставка/12 * площадь подвала * коэффициент подвала) * КТ.

При изменении базовой ставки МАП изменяется со следующего месяца после даты изменения ставки. Оплата производится ежемесячно.

Договор об аренде может заключаться как с организациями (юридическими лицами), так и с физическими лицами. В договоре об аренде помещения, имеющего номер, дату, фиксируется дата начала аренды, дата завершения аренды. Для юридического лица в БД заносится название, адрес, ИНН, номер и дата лицензии о деятельности. Для физического лица — ФИО, паспортные данные (Серия, Номер, Дата выдачи, Кем выдан), ИНН и адрес.

Необходимо осуществить следующую обработку данных:

– итоговая сумма оплат за текущий месяц (на заданную дату);

– список арендаторов (тип, название, адрес и другие характеристики арендуемого помещения) на текущую дату;

– список помещений, не сданных в аренду на текущую дату.

Проектирование базы данных

Построение инфологической модели

В результате анализа предметной области была построена информационно-логическая модель (ИЛМ), представляющая собой описание классов объектов и связей между ними. В данном случае ИЛМ предметной области была представлена в терминах семантической модели данных, в виде ER-диаграммы.

На рисунке 7.1 показана ER-модель предметной области на основе методологии Ричарда Баркера.

В методологии Ричарда Баркера используются следующие соглашения:

– класс объектов отображается в виде четырехугольника с закругленными углами, а имя класса объектов указывается внутри четырехугольника, это имя существительное в единственном числе, отображенное заглавными буквами;

– свойства записываются внутри четырехугольника, отображающего класс объектов строчными буквами, это имя существительное в единственном числе;

– четырехугольник, отображающий класс объектов, можно увеличивать до любых размеров, четырехугольники могут быть разных размеров;

– опциональность свойств помечается: обязательное свойство – звездочкой (*), необязательное – кружочком (о);

– уникальный идентификатор помечается #, если уникальных идентификаторов несколько, тогда каждый помечается номером, указанным в скобках, например, # (1), #(2);

– обязательная связь помечается сплошной линией, необязательная связь пунктирной линией;

– тип (мощность) связи "один" помечается линией, "много" – "вороньей лапой".

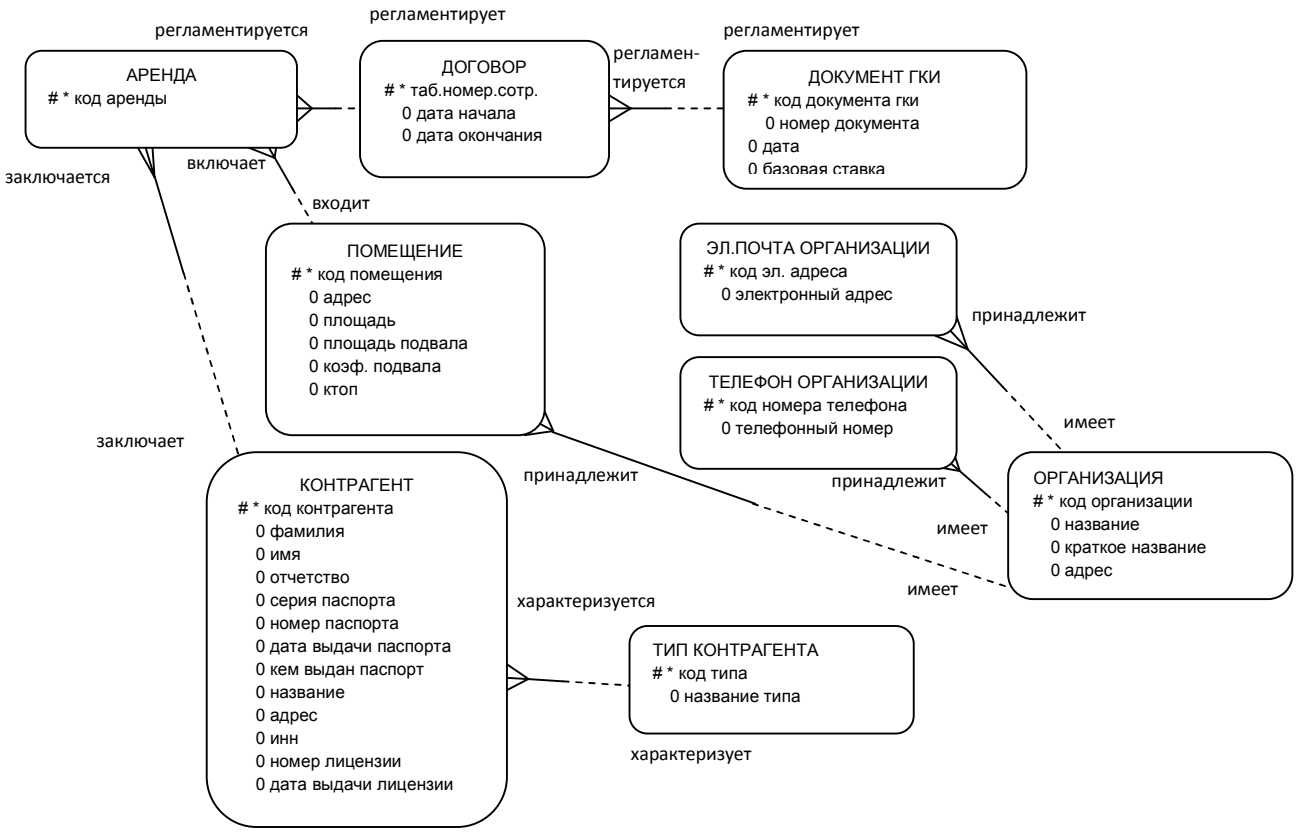


Рисунок 7.1 – Инфологическая модель предметной области

Построение даталогической модели

После построения и проверки инфологической модели предметной области была создана даталогическая модель базы данных. Полученная в рамках лабораторной работы база данных является реляционной, поэтому даталогическая модель соответствует реляционной модели.

На рисунке 7.2 представлена даталогичекая модель реляционной базы данных.

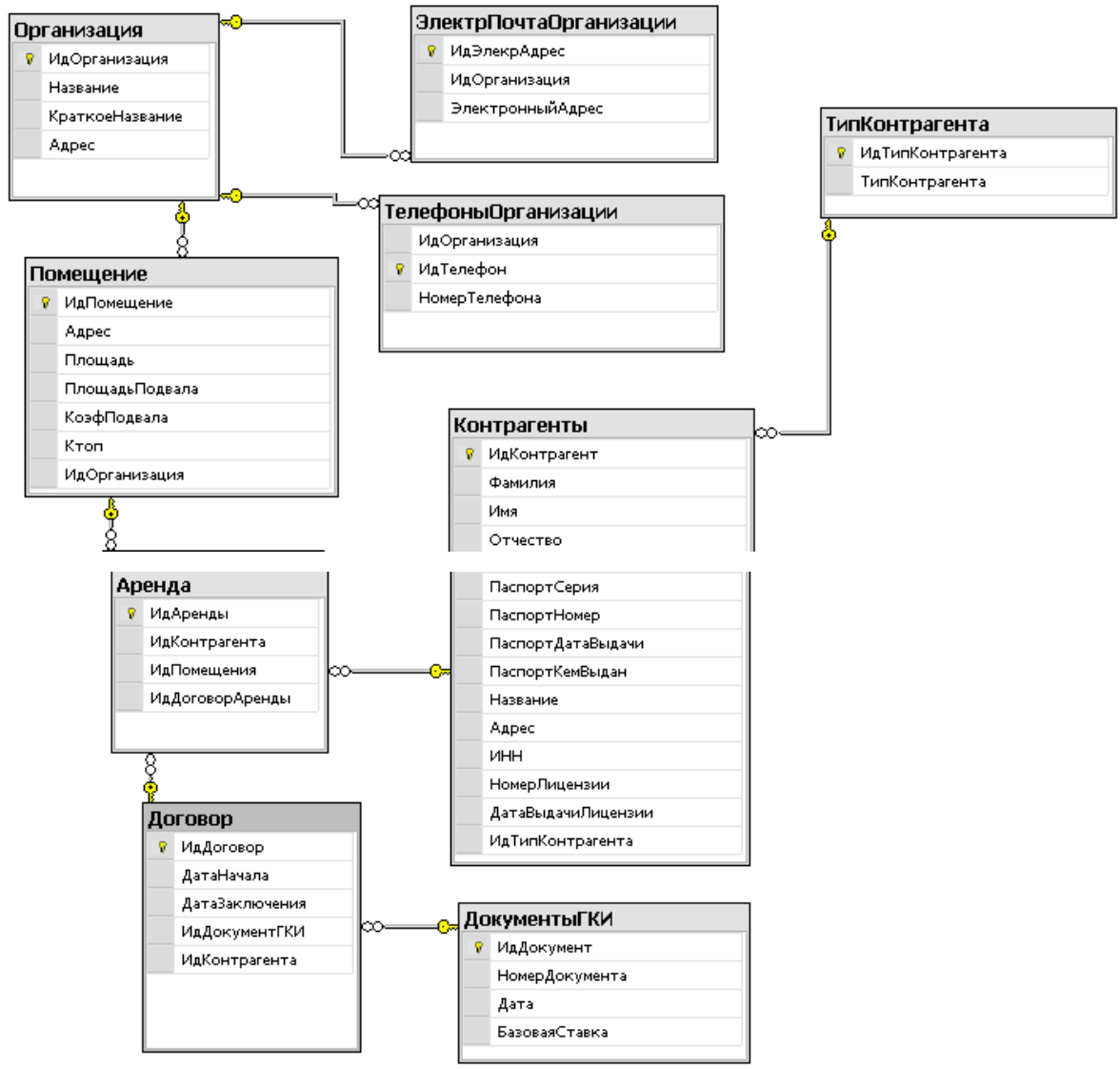


Рисунок 7.2 – Даталогическая модель

Работа с сервером баз данных

Обоснование выбора СУБД

Для разработки приложения была использована СУБД Microsoft SQL Server 2005. Современный сервер баз данных, сочетающий в себе простоту установки, актуальный набор объектов БД, средств регламентации доступа и защиты данных, возможность генерации отчетов и наиболее полное взаимодействие с продуктами компании-разработчика, как офисными (Office System), так и инструментальными (Visual Studio).

Разработку клиент-серверного приложения необходимо начать с подготовки к выполнению своих функций серверной части, в качестве которой выступает СУБД. База данных должна быть окончательно физически реализована и наполнена тестовым набором данных, прежде чем будет осуществлен этап реализации клиентской части — программы языке высокого уровня.

Таким образом, на данном этапе необходимо создать базу данных, добавить таблицы, с помощью диаграммы связей установить связи между первичными и внешними ключами таблиц, а также реализовать представления и триггеры.

Работа с СУБД SQL Server 2005 может происходить как в графическом, так и в консольном режиме, в первом случае используется среда SQL Server Management Studio, во втором – программа SQLCMD, работающая из командной строки. Каждый способ имеет свои преимущества и недостатки, однако, на этапе знакомства с технологией создания клиент-серверных приложений на основе БД стоит ограничиться использованием графического режима обслуживания сервера баз данных. Кроме того, интерфейс Management Studio поддерживает исполнение команд на языке SQL, помимо обеспечения манипуляций с графическими представлениями объектов БД.

Создание базы данных

Для создания базы данных необходимо авторизоваться в качестве пользователя СУБД посредством среды Management Studio. В панели Обозреватель объектов появится дерево объектов соответствующего сервера. Для создания базы данных необходимо вызвать контекстное меню узла Базы данных и выбрать пункт «Создать базу данных» как показано на рисунке 7.3.

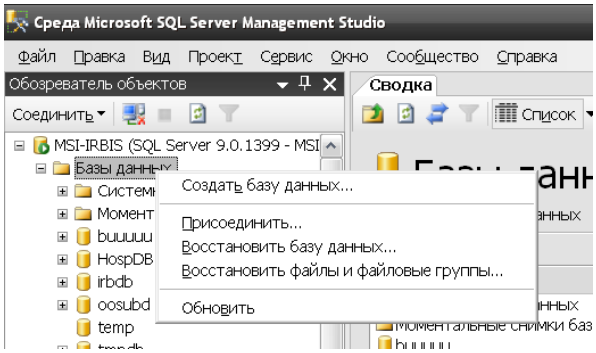


Рисунок 7.3 – Создание базы данных

В появившемся окне следует указать имя новой базы данных, а также можно указать собственный путь к файлам БД в таблице «Файлы базы данных», оперируя значением параметров столбца «Путь». Для остальных настроек рекомендуется сохранить значения по умолчанию. Вид диалогового окна добавления базы данных показан на рисунке 7.4.

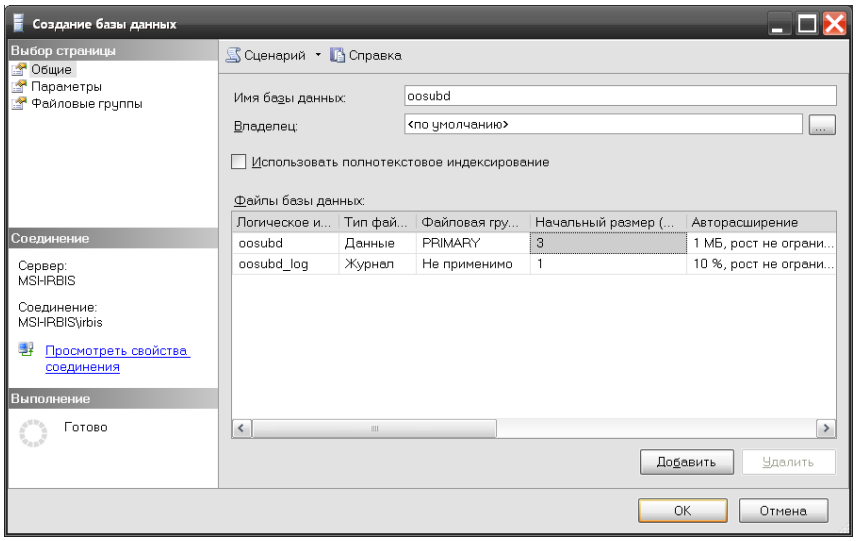


Рисунок 7.4 – Окно параметров создаваемой БД

После создания база данных появится в качестве потомка узла базы данных, которая содержит в качестве дочерних узлов объекты СУБД различных классов.

Добавление таблиц

Добавление таблиц происходит аналогично добавлению базы данных на сервер: в контекстном меню категории объектов «Таблицы» выбирается пункт «Создать таблицу». Интерфейс добавления новой таблицы состоит из 3 колонок «Имя столбца», «Тип данных», «Разрешить значения Null». Выбранная версия СУБД поддерживает русскоязычные наименования полей таблиц, однако, сложные названия с использованием пробелов и других допустимых символов могут быть автоматически заключены в квадратные скобки. Таким образом, в качестве имени столбцов следует использовать короткие, но емкие названия, адекватно отражающие семантику предметной области. Вид интерфейса создания новой таблицы показан на рисунке 7.5.

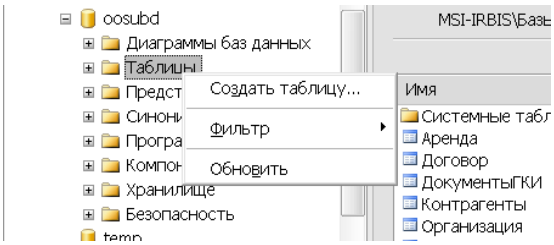


Рисунок 7.5 – Создание новой таблицы

Для первичных ключей, которые были выделены на этапе логического проектирования, следует установить признак первичного ключа — выбрать в контекстном меню строки таблицы пункт «Задать первичный ключ». Для атрибутов, которые не могут содержать первичные значения должен быть сброшен флаг «Разрешить значения Null», размещенный в 3 колонке интерфейса добавления таблиц.

Ключевые значения справочников, посредством которых обеспечивается целостность данных, часто требуют автоматического присвоения значений при добавлении новой записи в таблицу. Подобный функционал может быть реализован

посредством триггеров, однако в SQL Server 2005 включена возможность задать столбец таблицы автоматически идентифицируемым. Для этого следует на вкладке Свойства столбцов найти группу параметров «Спецификация идентифицирующего столбца» и установить значение «Да» для параметра «Является идентифицирующим столбцом» в соответствии с рисунком 7.6. Также доступно для установки начальное значение автоинкрементного поля и величина его приращения.

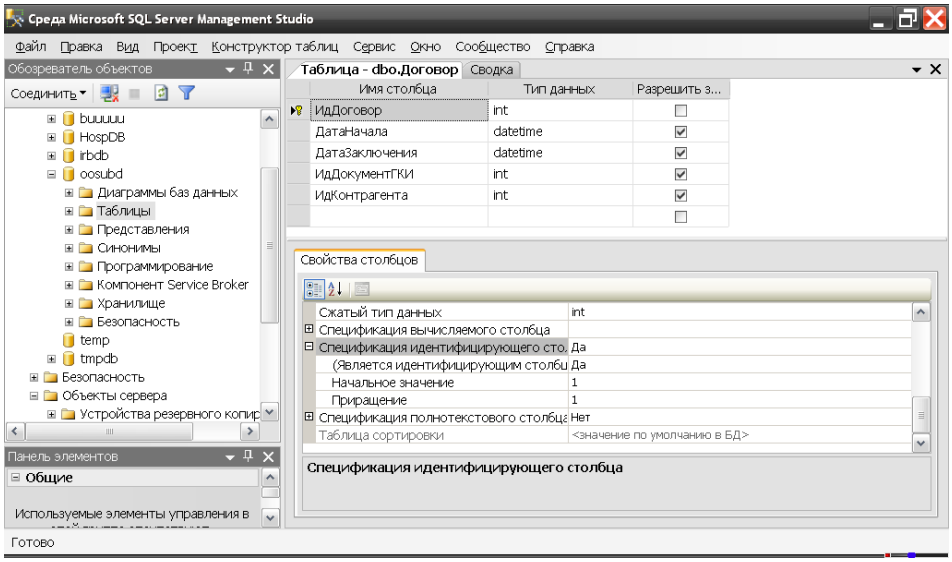


Рисунок 7.6 – Изменение параметров таблицы

Сохранение таблицы можно произвести как из строки меню с помощью команды Файл—Сохранить (имя таблицы), так и через контекстное меню заголовка вкладки соответствующего интерфейса добавления.

Добавление диаграммы баз данных

Структура связей, полученная на этапе физического проектирования базы данных, переносится на таблицы СУБД SQL Server 2005 посредством объекта «Диаграмма баз данных».

После добавления в иерархию объектов БД новой диаграммы необходимо разместить на ней созданные на предыдущем этапе таблицы и организовать между ними требуемые связи.

Добавление новых таблиц на рабочее поле редактирования элементов диаграммы происходит с помощью диалога Добавление таблицы, как показано на рисунке 7.7.

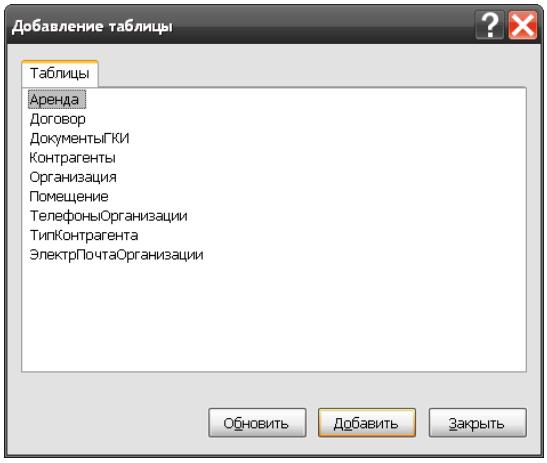


Рисунок 7.7 – Добавление таблиц в область диаграммы данных

Организация связей между таблицами осуществляется посредством протягивания мышью строки ключевого поля до поля внешнего ключа другой таблицы. Дальнейшие настройки связи происходят в окне «Отношение внешнего ключа», вид которого представлен на рисунке 7.8.

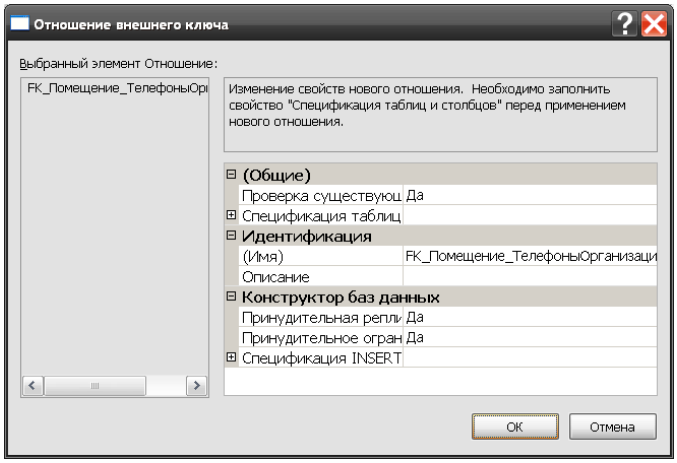


Рисунок 7.8 – Редактирование параметров связи

При грамотно спроектированной логической модели БД физическая реализация связей между таблицами в используемой СУБД обеспечит целостность и непротиворечивость данных в процессе использования.

Создание представлений

При выполнении лабораторной работы был использован метод выборки информации из базы данных посредством такого объекта СУБД SQL Server 2005 как представления. Данный подход доступа к данным имеет явные преимущества, например, код SQL-сценариев хранится в базе данных, что не требует перекомпиляции приложения при незначительных изменениях в структуре запроса. Также использование представлений наравне с таблицами в качестве источника данных очень удобно для клиентов, построенных на технологии ADO.NET. Представления позволяют сосредоточить в себе запросы с привилегиями, отличными от уровня доступа пользователя, которому разрешен только просмотр результатов запроса, что делает процесс разграничения прав пользователей более гибким.

Для добавления нового представления в иерархию объектов БД следует в контекстном меню категории Представления выбрать пункт «Создать представление», как показано на рисунке 7.9.

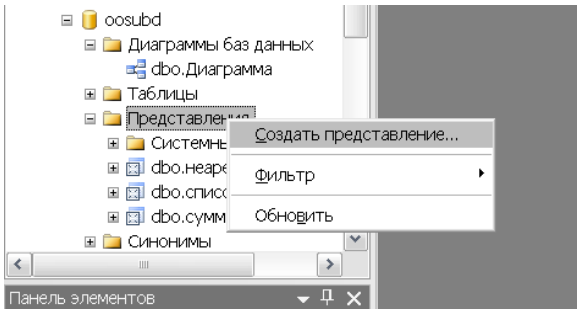


Рисунок 7.9 – Создание представления

Простые представления, в которых происходит обращение к малому числу таблиц, могут быть построены с помощью конструктора запросов. В нашем случае запрос к базе данных содержит вложенные запросы и ряд условий, что привело к необходимости вручную сформировать код SQL-запроса в консоли, предусмотренной интерфейсом создания представления в соответствии с рисунком 7.10

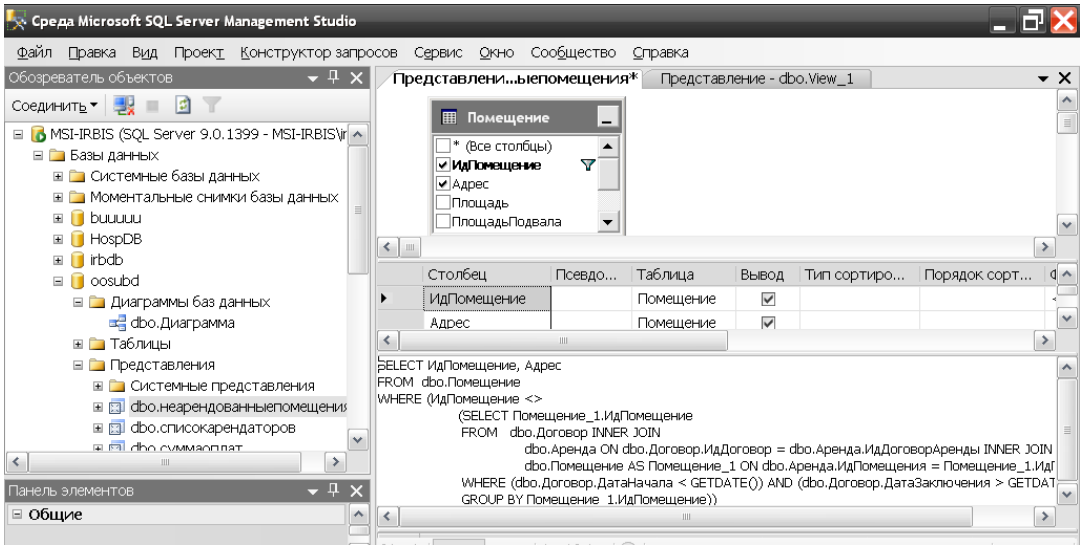


Рисунок 7.10 – Интерфейс формирования представления

Код представления, возвращающего информацию о количестве помещений, не арендованных на текущую дату представлен в следующем листинге:

```
SELECT ИдПомещение, Адрес
FROM  dbo.Помещение
WHERE (ИдПомещение <>
            (SELECT Помещение_1.ИдПомещение
             FROM   dbo.Договор INNER JOIN
                          dbo.Аренда ON dbo.Договор.ИдДоговор =
dbo.Аренда.ИдДоговорАренды INNER JOIN
                          dbo.Помещение AS Помещение_1 ON
dbo.Аренда.ИдПомещения = Помещение_1.ИдПомещение
```

WHERE (dbo.Договор.ДатаНачала < GETDATE()) AND (dbo.Договор.ДатаЗаключения > GETDATE())
GROUP BY Помещение_1.ИдПомещение))

SQL-код представления, возвращающего информацию о списке арендаторов, представлен в следующем листинге:

```
SELECT dbo.Помещение.ИдПомещение, dbo.Помещение.Адрес,
dbo.Контрагенты.Фамилия, dbo.Контрагенты.Имя, dbo.Контрагенты.Отчество,
        dbo.ТипКонтрагента.ТипКонтрагента,
dbo.Контрагенты.Название, dbo.Контрагенты.Адрес AS АдресАрендатора,
        dbo.Контрагенты.НомерЛицензии,
dbo.Контрагенты.ПаспортСерия, dbo.Контрагенты.ПаспортНомер
FROM  dbo.Аренда INNER JOIN
        dbo.Договор ON dbo.Аренда.ИдДоговорАренды =
dbo.Договор.ИдДоговор INNER JOIN
        dbo.Контрагенты ON dbo.Аренда.ИдКонтрагента =
dbo.Контрагенты.ИдКонтрагент INNER JOIN
        dbo.Помещение ON dbo.Аренда.ИдПомещения =
dbo.Помещение.ИдПомещение INNER JOIN
        dbo.ТипКонтрагента ON dbo.Контрагенты.ИдТипКонтрагента =
dbo.ТипКонтрагента.ИдТипКонтрагента
WHERE (dbo.Помещение.ИдПомещение = 1) AND
(dbo.Договор.ДатаЗаключения > GETDATE()) AND (dbo.Договор.ДатаНачала <
GETDATE())
```

Код представления отчета о сумме оплат за текущий месяц

```
SELECT SUM(МАП) AS СуммаМАП
```

```
FROM  (SELECT dbo.Помещение.ИдПомещение,
dbo.ДокументыГКИ.ИдДокумент, dbo.Договор.ИдДоговор,
dbo.ДокументыГКИ.БазоваяСтавка, dbo.Помещение.Площадь,
dbo.Помещение.ПлощадьПодвала, dbo.Помещение.КоэфПодвала,
dbo.Помещение.Ктоп, (dbo.ДокументыГКИ.БазоваяСтавка / 12 *
dbo.Помещение.Площадь + dbo.ДокументыГКИ.БазоваяСтавка / 12 *
dbo.Помещение.ПлощадьПодвала * dbo.Помещение.КоэфПодвала) *
dbo.Помещение.Ктоп AS МАП
                FROM   dbo.Помещение INNER JOIN
                             dbo.Аренда ON dbo.Помещение.ИдПомещение =
dbo.Аренда.ИдПомещения INNER JOIN dbo.Договор ON
dbo.Аренда.ИдДоговорАренды = dbo.Договор.ИдДоговор INNER JOIN
dbo.ДокументыГКИ ON dbo.Договор.ИдДокументГКИ =
dbo.ДокументыГКИ.ИдДокумент WHERE (MONTH(dbo.ДокументыГКИ.Дата)
= 10)) AS derivedtbl_1
```

Создание триггеров

С помощью ограничений целостности, правил и значений по умолчанию не всегда можно добиться нужного уровня функциональности. Часто требуется реализовать сложные алгоритмы проверки данных, чтобы гарантировать их достоверность и реальность. Кроме того, бывает необходимо отслеживать изменения значений таблицы, чтобы нужным образом изменить связанные данные. Для решения этих и многих других проблем используются триггеры.

Триггер – это специальный тип хранимых процедур, запускаемых сервером автоматически при выполнении тех или иных действий с данными таблицы или при глобальном событии.

Обычно триггер привязывается к конкретной таблице. Для его создания следует обратиться к иерархическому представления структуры БД: в дочерних узлах

объекта таблица имеется контейнер «Триггеры», в контекстном меню которого имеется пункт «Создать триггер», как показано на рисунке 7.11.

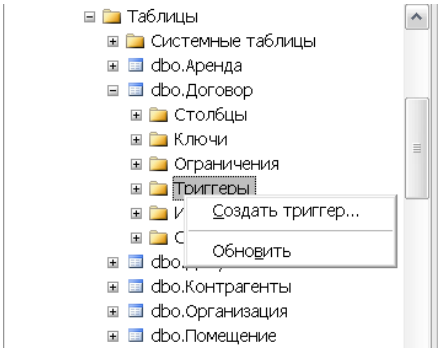


Рисунок 7.11 – Создание триггера

При создании нового триггера предлагается шаблон, который может быть изменен в соответствии с требованиями и нуждами пользователя, что продемонстрировано на рисунке 7.12.

Необходимо отметить, что исполнение инструкции триггера при его инициализации осуществляется посредством вызова процедуры GO. Использование данного ключевого слова TSQL аналогично переносу строки в консольном режиме исполнения сценариев.

В рамках предметной области требуется учитывать дату начала и дату завершения договора, таким образом, недопустимо, чтобы дата завершения была раньше даты начала действия договора. Для того чтобы избежать нарушения логической целостности данных описанного данном примере, был создан триггер следующего вида:

```
set ANSI_NULLS ON
set QUOTED_IDENTIFIER ON
go
CREATE TRIGGER [CorrectDate2]
ON [dbo].[Договор] FOR INSERT, UPDATE
AS
IF @@ROWCOUNT=1
```

```
BEGIN
  IF EXISTS(SELECT *
     FROM inserted
     WHERE inserted.ДатаНачала > inserted.ДатаЗаключения)
    BEGIN
     ROLLBACK TRAN
    PRINT
        'Дата начала договора не может быть позже даты окончания дейст-
вия договора!'
     END
END
```

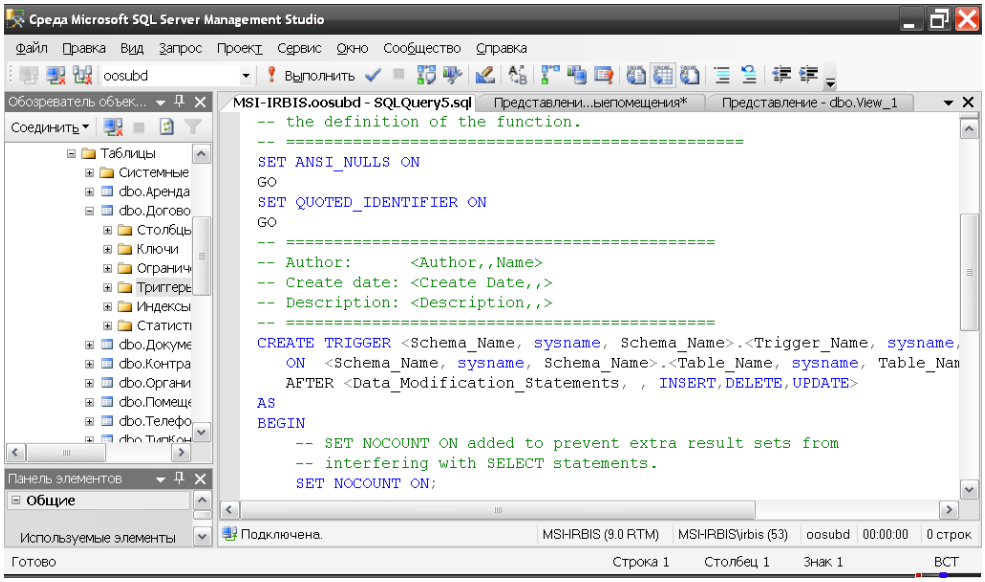


Рисунок 7.12 – Шаблон триггера SQL Server

Вид сообщения, которое выводится попытке добавления противоречивых данных с БД, показан на рисунке 7.13.

Из описания триггера видно, что он привязан к таблице *ЗаписьТрудовой-Книги* и выполняется при осуществлении событий вставки или обновления.

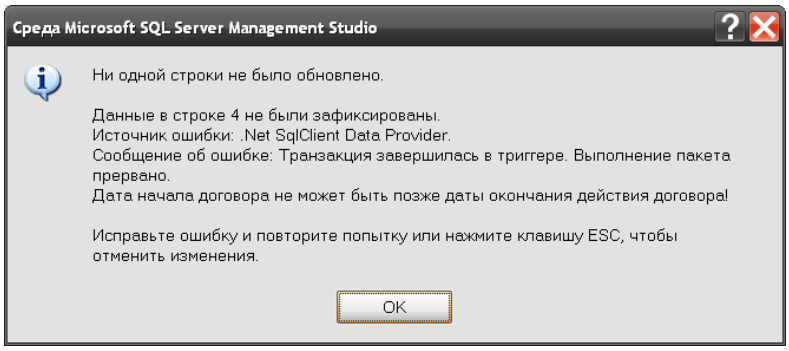


Рисунок 7.13 – Сообщение, инициируемое при срабатывании триггера

Необходимо отметить, что прежде чем данные будут внесены в БД – будет завершена транзакция – они находятся во временной таблице inserted. В этой таблице происходит поиск некорректных записей посредством запроса Select, если количество ошибочных строк не равно нулю, значит вставка или обновление данных не может быть допущено и происходит откат транзакции с выводом информационного сообщения.

Создание ролей

Чтобы продемонстрировать порядок работы с SQL-консолью в среде Management Studio были установлены уровни доступа посредством SQL-сценариев. Для добавления пользователей Оператор1, Оператор2, которые должны иметь одинаковые права доступа, были созданы SQL-сценарии, вид которых представлен ниже.

```
CREATE LOGIN Оператор1 WITH PASSWORD = '123';
CREATE USER Оператор1 FOR LOGIN ИнспекторОК1;
GO
CREATE LOGIN Оператор2 WITH PASSWORD = '123';
CREATE USER Оператор2 FOR LOGIN ИнспекторОК2;
GO
```

В ходе работы были созданы SQL-скрипты для реализации уровней доступа для пользователей ИнспекторОК1 и ИнспекторОК2 с использованием ролей.

```
Create Role R_Оператор;
Grant Select, Insert, Update (Адрес, Площадь, ПлощадьПодвала, КоэфПодвала, Ктоп, ИдОрганизация) On Помещение To R_Оператор;
Grant Select (ИдОрганизация, Название, КраткоеНазвание, Адрес) On Организация To R_Оператор;
Grant Select, Insert (ИдАренды, ИдКонтрагента, ИдПомещения, ИдДоговорАренды
) On Аренда To R_ Оператор;
Grant Select On [неарендованныепомещения] To R_Оператор;
```

Предоставление роли пользователям осуществляет следующий код:

```
Grant R_Оператор To Оператор1
Grant R_Оператор To Оператор2
```

Чтобы осуществить запрос к СУБД через среду Management Studio, необходимо в контекстном меню базы данных, представленной в обозревателе объектов, выбрать пункт «Создать запрос». В результате в рабочей области среды появится поле ввода сценария. По завершении написания команд следует нажать кнопку «Выполнить», это инициирует запрос к БД, результат которого будет выведен в информационном поле. Порядок создания ролей посредством SQL-сценариев в среде Management Studio показан на рисунке 7.14.

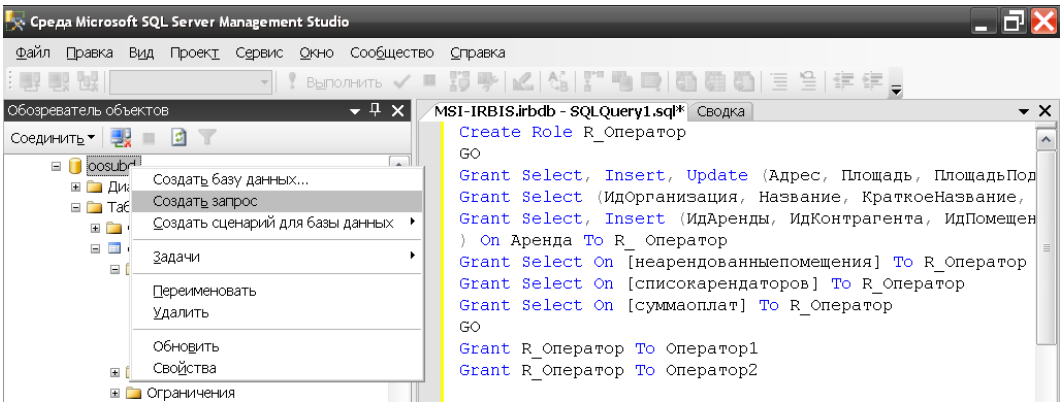


Рисунок 7.14 – Создание ролей пользователей

Разработка клиентского приложения

Обоснование выбора системы программирования

Система программирования Microsoft Visual Studio 2008, выбранная для разработки поддерживает возможность создания приложений на базе технологии ADO .NET, (ActiveX Data Objects .NET), которая является частью Framework Class Library.

Данная технология является набором классов, реализующих программные интерфейсы для облегчения подключения к базам данных из приложения независимо от особенностей реализации конкретной системы управления базами данных и от структуры самой базы данных, а также независимо от места расположения этой самой базы – в частности, в распределенной среде (клиент-серверное приложение) на стороне сервера.

Работа с базой данных на уровне приложения .NET представляет собой работу:

– с множествами объявлений классов, которые содержат объявления унаследованных методов и свойств, предназначенных для решения задачи извлечения информации из базы данных;

– с множеством объектов-представителей классов, которые обеспечивают работу с базами данных;

– с множеством значений и свойств конкретных объектов, отражающих специфику структуры конкретной базы данных.

Таким образом, приложение на основе ADO .NET и СУБД SQL Server представляют собой объектно-реляционную СУБД, являющуюся переходным этапом на пути к объектно-ориентированным СУБД.

Организация клиент-серверного взаимодействия

Приложения, использующие технологию ADO .NET используется модель доступа к отсоединенным данным. В основе концепция доступа к данным лежат два компонента:

– набор данных (представляется объектом класса DataSet) со стороны клиента, локальное временное хранилище данных;

– провайдер данных (представляется объектом класса DataProvider), посредник, обеспечивающий взаимодействие приложения и базы данных со стороны базы данных (в распределенных приложениях – со стороны сервера).

Чтобы использовать возможности этих компонентов, необходимо добавить новый источник данных в проект Visual Studio, вызвав мастер, показанный на рисунке 7.15.

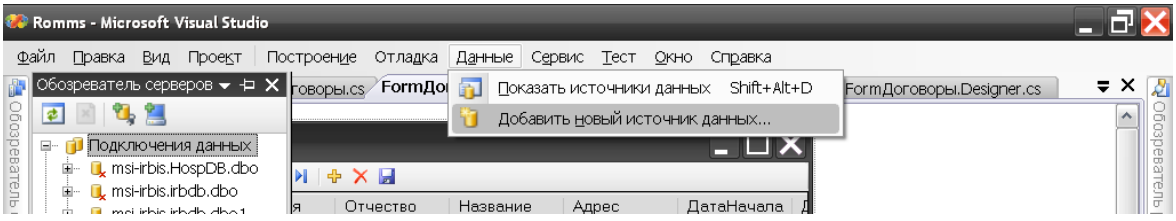


Рисунок 7.15 – Вызов мастера добавления нового источника данных

В нашем случае были выбраны таблицы и представления. Новые источники данных появятся в соответствующей панели системы программирования, что продемонстрировано на рисунке 7.16.

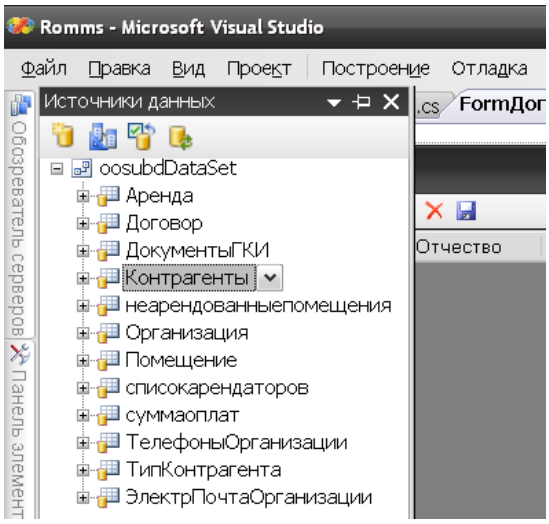


Рисунок 7.16 – Список активных источников данных Visual Studio

При перетаскивании объекта данных на форму автоматически будут созданы провайдер данных, набор данных, средства отображения данных на экране – компонент DataGridView, а также добавлены вспомогательные элементы интерфейса: кнопки сохранения, удаления, навигации по данным, как показано на рисунке 7.17.

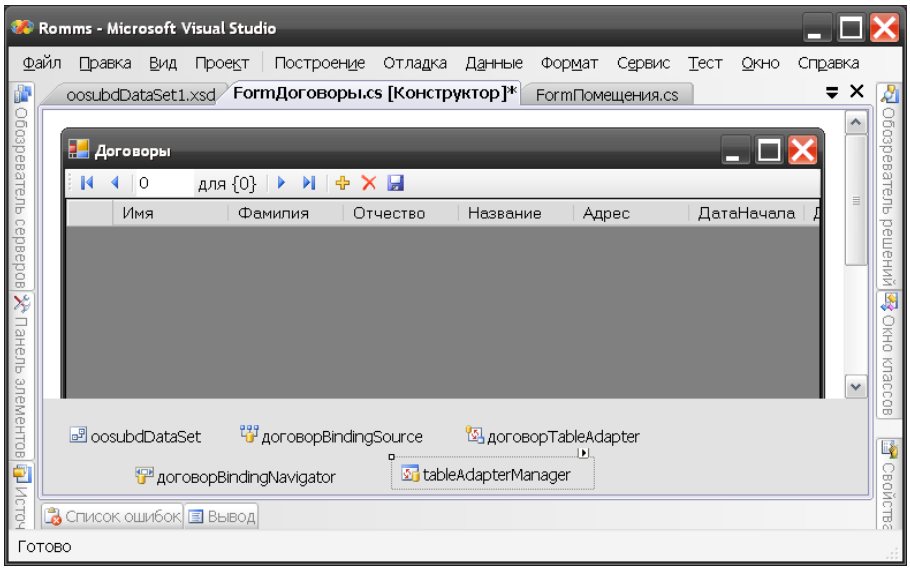


Рисунок 7.17 – Группа компонентов, обеспечивающих взаимодействие с БД

На основе полученных таблиц и представлений строится объектная структура данных, выраженная в виде классов. Редактирование данных классов происходит автоматически при работе с конструктором данных — внутренней интерпретации схемы данных БД.

Модификация конструктора данных может потребоваться при формировании запросов, охватывающих несколько таблиц, а также создании параметрических запросов. Интерфейс конструктора данных показан на рисунке 7.18.

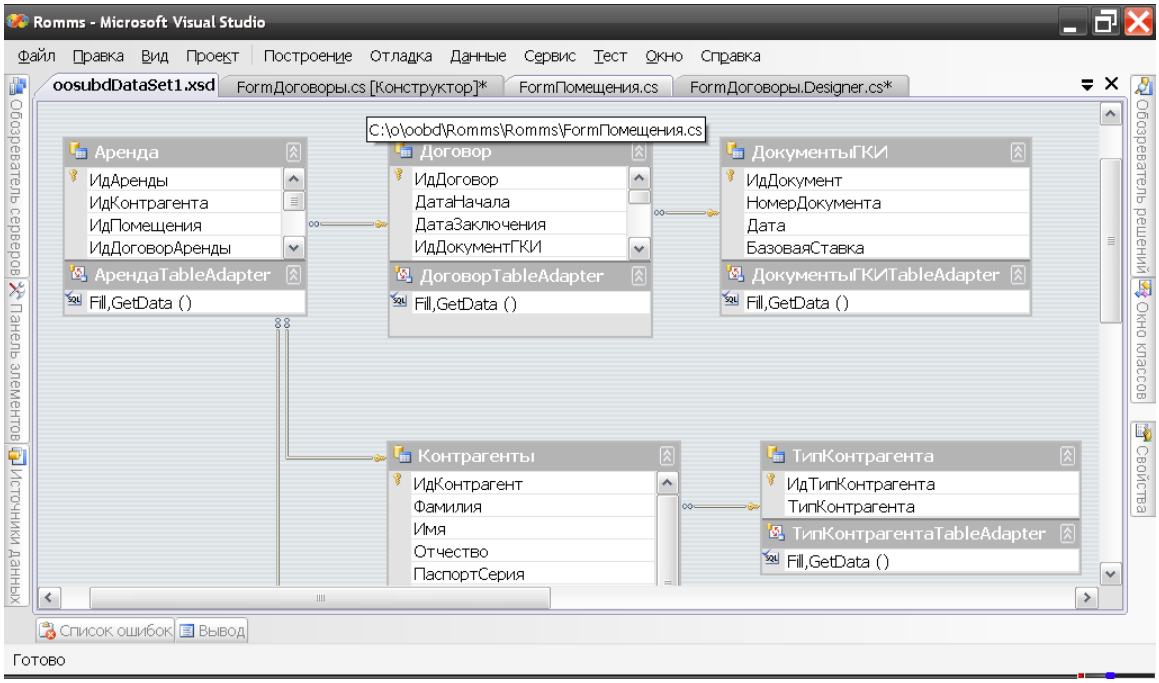


Рисунок 7.18 – Конструктор данных Visual Studio

Руководство системного программиста

Общие сведения о программном средстве

Программное средство предназначено для автоматизации процесса учета арендованных физическими и юридическими лицами помещений, а также позволяет рассчитать итоговую сумму оплат за текущий месяц, список арендаторов (тип, название, адрес и другие характеристики арендуемого помещения) на текущую дату, список помещений, не сданных в аренду на текущую дату.

Структура программного средства

Программное средство состоит из файлов с расширениями *.exe, *.mdf и *.ldf. Программное средство включает в себя следующие файлы:

– Romms.exe – исполняемый файл программы;

– oosubd_log.ldf – журнал транзакций базы данных;

– oosubd.mdf – файл базы данных.

Установка программного средства

Для работы клиентской части требуется компьютер со следующей минимальной аппаратно-программной конфигурацией:

– Процессор с частотой 800 МГц (32- или 64-разрядный).

– Оперативная память объемом 256 Мбайт.

– Жесткий диск со свободным пространством не менее чем 100 Мбайт.

– Операционная система Microsoft Windows XP, 2003, Vista, 7.

– Набор библиотек Microsoft .NET Framework 3.5 и выше.

– Монитор с разрешением VGA или выше, по крайней мере, 1024x768 точек.

– Указывающее устройство мышь или совместимое указывающее устройство.

Для размещения базы данных требуется компьютер со следующей минимальной аппаратно-программной конфигурацией:

– Процессор 500 МГц или выше (рекомендуется 1 ГГц или выше).

– Оперативная память объемом 1 Гбайт и более.

– 350 МБ свободного дискового пространства для полной установки.

– Монитор VGA или большего разрешения.

– СУБД SQL Server 2005 Enterprise Edition и Standard Edition и выше.

– Операционная система Windows Windows 2000 Server, Windows XP Professional, Server 2003 Standard Edition, Windows Server 2008, Windows 7.

Для установки программного средства необходимо скопировать все файлы и папки с носителя на свободное дисковое пространство.

Для подготовки к работе, необходимо зарегистрировать в базу данных в СУБД, создать имена входа и добавить пользователей с требуемыми правами. Данная группа операций осуществляется через средство управления SQL-сервером «Management Studio», которая входит в его рекомендуемую поставку.

Первоначально необходимо подключить базу данных к серверу. Для этого требуется совершить следующие действия:

После запуска утилиты необходимо нажать правой кнопкой на разделе «Базы данных» и выбрать пункт «Присоединить…» согласно рисунку 7.19.

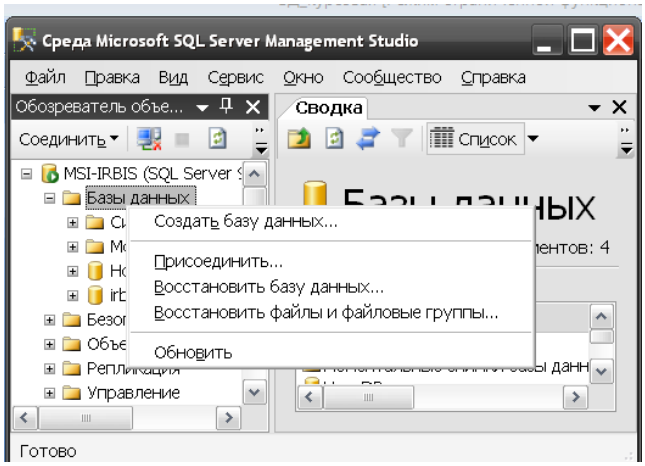


Рисунок 7.19 – Регистрация базы данных в MS SQL Server

После чего в появившемся окне нажать кнопку «Добавить» и в дереве каталогов найти и выбрать oosubd.mdf, как показано на рисунке 7.20. После чего, нажав кнопку «ОК» на этом и на следующем окне, база данных будет добавлена.

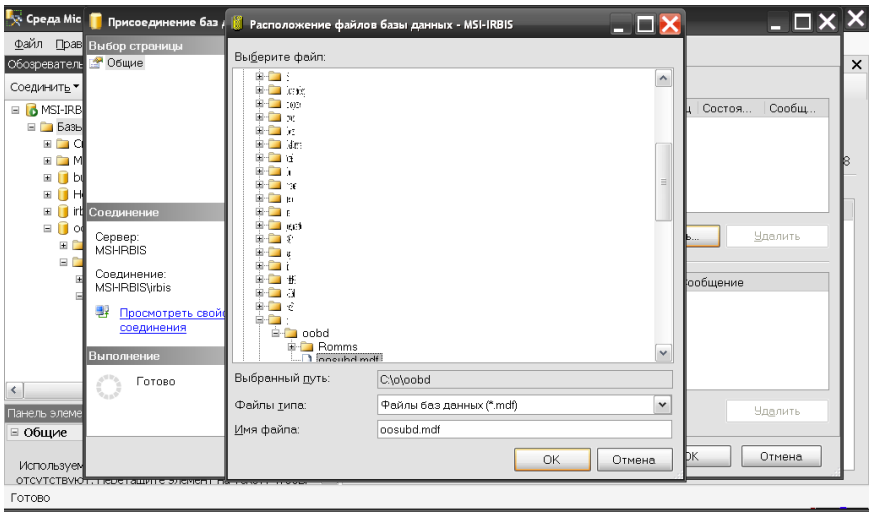


Рисунок 7.20 – Выбор базы данных для подключения к MS SQL Server

Так как для работы с базой данных используется система авторизации Windows, необходимо проследить, чтобы роль владельца базы данных была назначена либо администратору компьютера, либо пользователю NT Authority\System. Так же можно создать имя входа, руководствуясь следующим ниже алгоритмом.

После запуска утилиты необходимо для каждого будущего пользователя создать логин, чтобы впоследствии выделить ему необходимые права в базе данных. Для этого необходимо нажать правой кнопкой на разделе «Имена входа» и в появившемся меню выбрать пункт «Создать имя входа» в соответствии с рисунком 7.21.

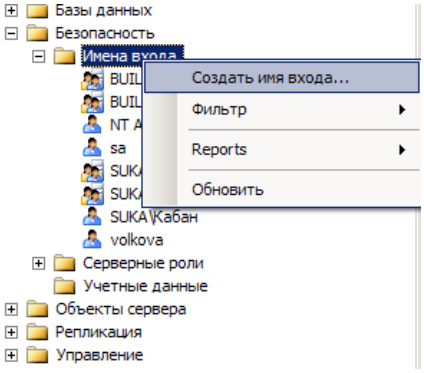


Рисунок 7.21 – Добавление имени входа в MS SQL Server

В появившемся окне «Создание имени входа» необходимо выбрать одно из имеющихся имен входа в системе, как показано на рисунке рисунок 7.22.

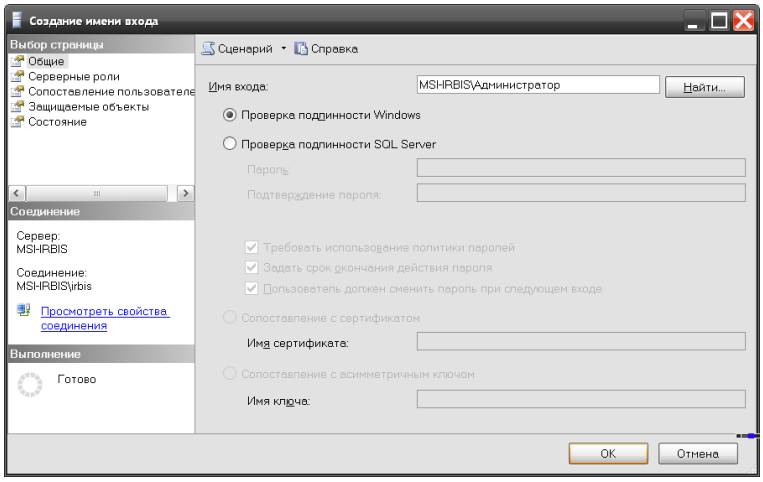


Рисунок 7.22 – Создание имени входа

После создания имени входа необходимо создать пользователя базы данных. Для этого необходимо выбрать в дереве параметром базы данных «Безопасность» и в контекстном меню объекта «Пользователи» выбрать пункт «Создать пользователя» согласно рисунку 7.23.

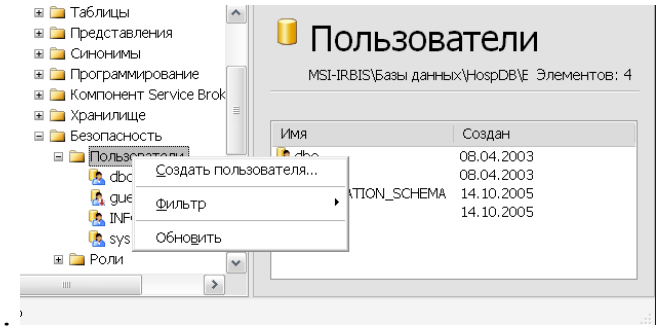


Рисунок 7.23 – Создания пользователя базы данных

В появившемся окне «Пользователь базы данных – Создать» необходимо ввести имя пользователя базы данных и логин SQL Server, который будет соответ-

ствовать этому пользователю базы данных, указать роль этого пользователя и нажать кнопку «OK» в соответствии с рисунком 7.24.

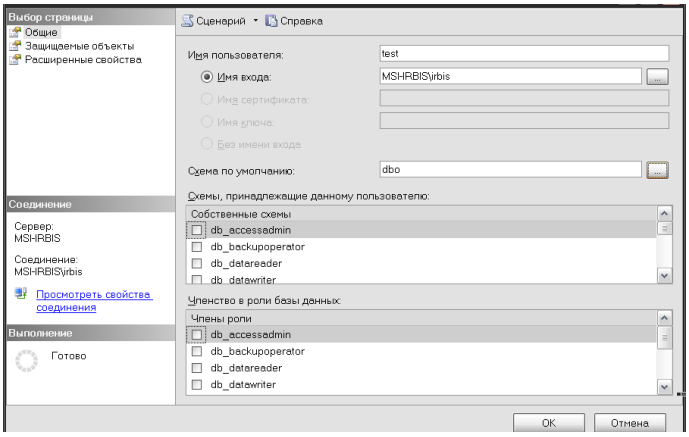


Рисунок 7.24 – Создание пользователя базы данных

Полномочия пользователя по работе с базой данных устанавливаются в окне настроек пользователя базы данных, вызываемом двойным щелчком по имени пользователя в списке, как показано на рисунке 13.25. Для администратора можно установить полномочия владельца базы данных, они же применяются при создании базы данных с нуля.

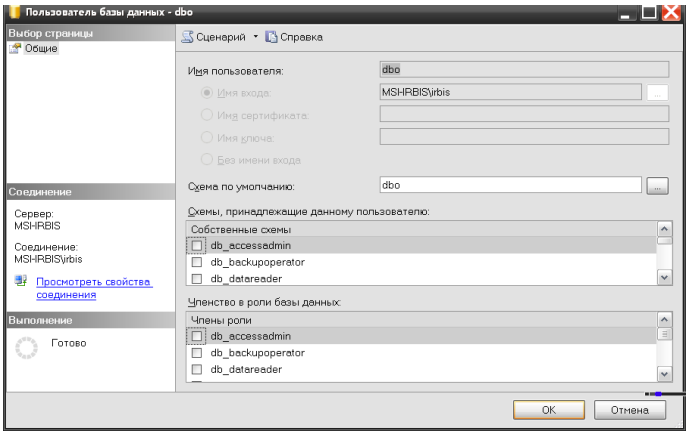


Рисунок 7.25 – Менеджмент полномочий пользователей БД

Проверка программного средства

После того как ПС установлено, проверить правильность установки можно путём запуска программы. Если приложение при начальном запуске не выдаёт системных ошибок, то можно сделать вывод, что приложение установлено правильно, находится в рабочем состоянии и может выполнять своё функциональное назначение.

Сообщение системному программисту

При установке ПС на ОС Windows 7 возможно придется подтвердить копирование файлов.

Необходимо, чтобы пользователи Windows были внесены в список пользователей, имеющих доступ к БД программного средства.

Руководство программиста

Назначение и условия применения программного средства

Программное средство предназначено для автоматизации процесса учета арендованных физическими и юридическими лицами помещений, а также позволяет рассчитать итоговую сумму оплат за текущий месяц, список арендаторов (тип, название, адрес и другие характеристики арендуемого помещения) на текущую дату, список помещений, не сданных в аренду на текущую дату.

Минимально необходимая конфигурация компьютера, при котором программа будет работать корректно:

– Процессор с частотой 800 МГц (32- или 64-разрядный).

– Оперативная память объемом 256 Мбайт.

– Жесткий диск со свободным пространством не менее чем 100 Мбайт.

– Операционная система Microsoft Windows XP, 2003, Vista, 7.

– Набор библиотек Microsoft .NET Framework 3.5 и выше.

– Монитор с разрешением VGA или выше, по крайней мере, 1024x768 точек.

– Указывающее устройство мышь или совместимое указывающее устройство.

Характеристика программного средства

При запуске программы активизируется оконный режим работы для главного окна. Данный режим наиболее удобен для работы. При желании пользователь может изменить режим работы на полноэкранный для главного окна программы или полноэкранный для дочерних окон.

Для корректной работы программы необходимо 256 Мбайт оперативной памяти и 100 Мбайт на жестком магнитном диске.

Для контроля программы программист может установить среду разработки программы Microsoft Visual Studio 2008.

Работа с программным средством

После установки программы, необходимо запустить файл Romms.exe, данный файл является исполняемым. После запуска управление перейдёт главному окну программы. Вызов дочерних окон происходит из строки меню, навигацию по которому осуществляет пользователь.

Входные и выходные данные

Входными данными являются данные вводимые пользователем. Выходными данными является отчеты и документы, которые генерирует ПС, представленные в виде таблиц.

Сообщение программисту

Данное программное средство не поддерживает модернизацию.

Руководство пользователя

Назначение программного средства

Программное средство предназначено для автоматизации процесса учета арендованных физическими и юридическими лицами помещений, а также позволяет рассчитать итоговую сумму оплат за текущий месяц, список арендаторов (тип, название, адрес и другие характеристики арендуемого помещения) на текущую дату, список помещений, не сданных в аренду на текущую дату.

Данное программное средство применимо непосредственно по назначению, является специализированным.

Условия выполнения программного средства.

Нормальное функционирование программного средства будет при соответствующей минимальной конфигурации компьютера:

- Процессор с частотой 800 МГц (32- или 64-разрядный).
- Оперативная память объемом 256 Мбайт.
- Жесткий диск со свободным пространством не менее чем 100 Мбайт.
- Операционная система Microsoft Windows XP, 2003, Vista, 7.
- Набор библиотек Microsoft .NET Framework 3.5 и выше.
- Монитор с разрешением VGA или выше, по крайней мере, 1024x768 точек.

Эксплуатация программного средства

Запуск программы «Аренда помещений» осуществляется щелчком по пиктограмме в файловом менеджере. При запуске открывается главное окно, вид которого показан на рисунке 7.26. Главное окно является родительских для всех окон, которые будут открыты в программе, для переключения между окнами использует-

ся пункт меню «Окна». Элементы меню сгруппированы в соответствии с соображениями удобства и частоты использования функций программного средства.

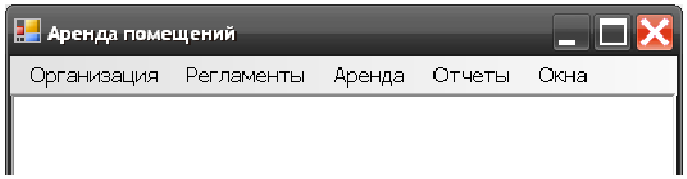


Рисунок 7.26 – Главное окно программы

Для организации учета необходимо задать в программе вспомогательные данные, такие как:

– организации,

– помещения,

– телефоны,

– адреса электронной почты,

– список арендаторов.

Когда вспомогательные данные будут введены в работу, можно приступать к созданию договоров аренды, для этого необходимо выбрать в строке меню пункт Аренда—Договоры. В результате, будет показано окно программы для работы с договорами, представленное на рисунке 7.27.

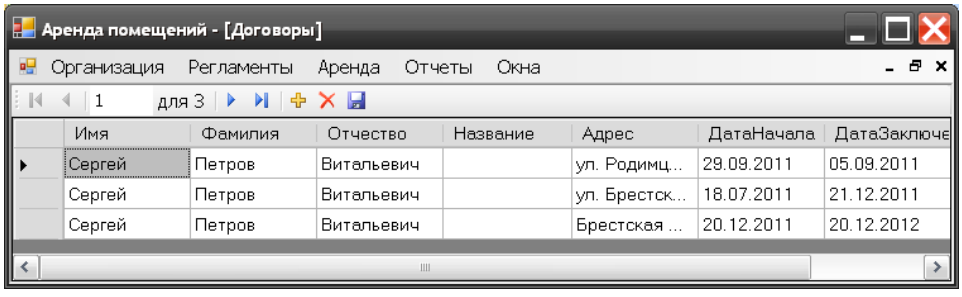


Рисунок 7.27 – Создание договоров аренды

Чтобы добавить новый договор аренды, следует нажать на кнопку с пиктограммой «+». В результате в окне появится новая строка, куда можно будет внести данные. Пользователю необходимо выбрать фамилию клиента для физического лица, или название юридического лица, выбрать адрес арендуемого помещения, указать дату начала и номер договора, которые можно открыть двойным щелчком по целевым ячейкам. Работа с цепочкой окон показана на рисунке 7.28 на примере выбора документа ГКИ.

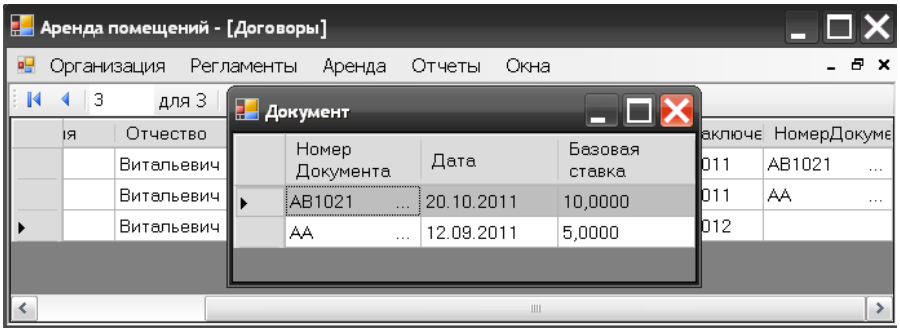


Рисунок 7.28 – Работа с окнами справочников

Необходимо обязательно указать дату заключения договора, окно выбора даты может быть вызвано двойным щелчком по ячейке по аналогии с выбором типа документ ГКИ. Процесс выбора даты покупки представлен на рисунке 7.29.

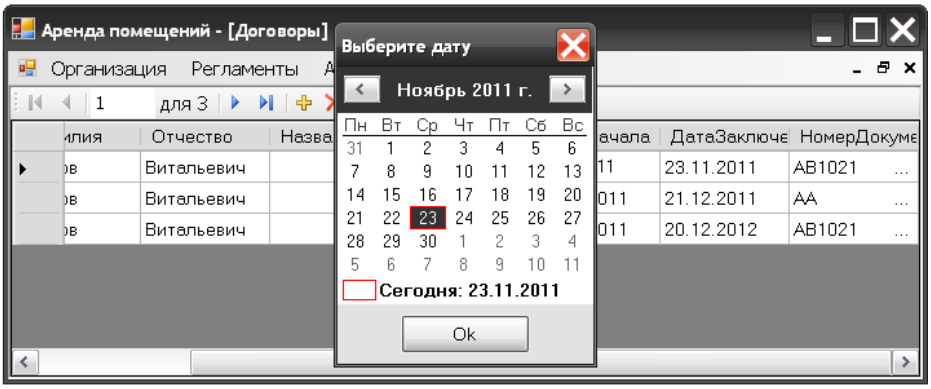


Рисунок 7.29 – Выбор даты заключения договора

Если после внесения всех данных не появится сообщение об ошибке корректности данных, можно приступить к сохранению изменений. Для этого следует

нажать на кнопку с изображением дискеты в меню текущего окна, как показано на рисунке 7.30.

Рисунок 7.30 – Элементы управления содержимым справочника

Для удаления данных следует выделить необходимую строку и нажать в меню управления содержимым кнопку с изображением красного перечеркивания «X». Затем требуется сохранить требуемое состояние базы данных, нажав на кнопку с изображением дискеты. В случае успешного удаления сообщения выводиться не будут, однако, если затронута какая-либо зависимость, будет выведено сообщение о невозможности удаления, показанное на рисунке 7.31.

В данном случае, требуется удалить все ссылающиеся на удаляемую позицию элементы базы данных и повторить попытку удаления.

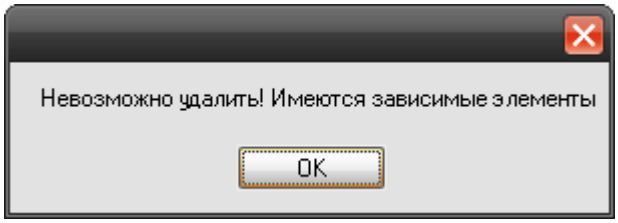


Рисунок 7.31 – Сообщение о нарушении зависимости данных

Аналогичным образом происходит работа с данными других таблиц.

Также в программе предусмотрен отчет об итоговой сумме оплат за текущий месяц (на заданную дату), который может быть вызван через пункт меню Отчеты – Сумма оплат. Результат выполнения запроса по отчету показан на рисунке 7.32.

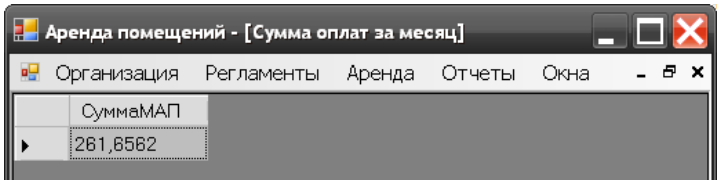


Рисунок 7.32 – Отчет о сумме оплат за месяц

Сведения о списке арендаторов могут быть получены при помощи отчета «Список арендаторов». Результатом выполнения отчета является таблица, в которой присутствуют имя, фамилия, отчество, паспортные данные для физических лиц и название, номер лицензии, дата ее выдачи для юридических лиц. Вид данного отчета показан на рисунке 7.33.

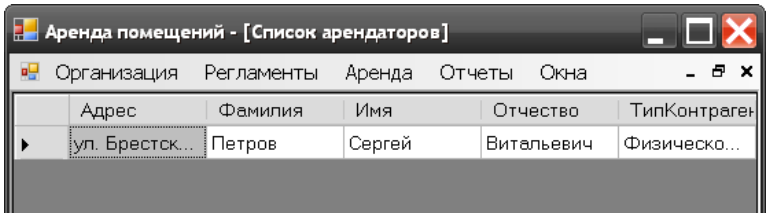


Рисунок 7.33 – Отчет о лицах, арендующих помещения

Информацию о количестве свободных помещениях и их адресе на заданную дату можно получить, выполнив запрос по отчету «Свободные помещения», вызов которого возможен через строку меню пункт Отчеты – Свободные помещения. Результат выполнения отчета показан на рисунке 7.34.

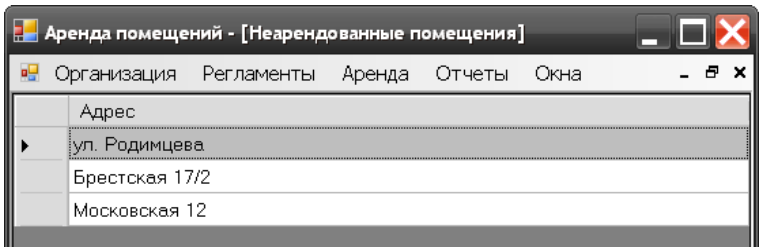


Рисунок 7.34 – Отчет о количестве каждой модели на заданную дату

В результате выполнения работы были спроектированы и реализованы реляционная база данных в СУБД MS SQL Server 2005 и клиентское приложение на языке C#, использующее технологию ADO .NET для взаимодействия с БД.

Применение этих двух компонентов в рамках клиент-серверного приложения позволило создать объектно-реляционную СУБД, в которой работа с реляци-

онной структурой данных в БД происходит при посредничестве объектных моделей таблиц в рамках приложения.

Помимо непосредственных плюсов объектно-реляционной модели в качестве достоинства можно отметить, что использование технологии ADO .NET позволило уменьшить число обменов между клиентским приложением и сервером баз данных.

Таким образом, рассмотренная в рамках лабораторной работы концепция реализации объектно-реляционной СУБД, реализованная с помощью актуальных инструментальных средств, может эффективно применяться на этапе перехода к объектно-ориентированной модели СУБД.

Список использованных источников

1. **Пирогов В.Ю.** SQL Server 2005: Программирование клиент-серверных приложений. – СПб.: БХВ-Петербург, 2006. – 336 с.: ил.

2. **Д. Кренке.** Теория и практика построения баз данных. 8-е изд./ Д. Кренке. – СПб.: Питер, 2003. – 800с.: ил.

3. **Советов Б.Я.** Базы данных: теория и практика: Учеб. Для втузов/ Б.Я. Советов, В.В. Цехановский, В.Д. Чертовской – 2-е изд., стер. – М.: Высш. шк., 2007. - 463 с.: ил.

4. **Марков А.С.**, Лисовский К.Ю. Базы данных. Введение в теорию и методологию: Учебник. – М.: Финансы и статистика, 2004. – 512 с.: ил.

5. **Кузин А.В.** Базы данных: Учеб. Пособие для студ. высш. учеб. заведений / А.В.Кузин, С.В. Левонисова. – М.: Издательский центр «Академия», 2005. – 320с.

6. **Глушаков С.В., Ломотько Д.В.** Базы данных: Учебный курс/Худож.-оформитель А.С.Юхтман. – Харьков: Фолио; ООО «Издательство АСТ», 2001. – 504 с. – (Домашняя библиотека).

7. **Пирогов В.Ю.** MS SQL Server 2000: управление и программирование. – СПб.: БХВ-Петербург, 2005. – 608 с.: ил.

8. **Таненбаум Э**., Стеен М. Распределенные системы. Принципы и парадигмы. - СПб.: Питер, 2003. - 877 с.: ил.

9. **Уткин В.Б.** Информационные системы в экономике: Учебник для студ. высш. учеб. заведений / В.Б. Уткин, К.В. Балдин. – М.: Издательский центр «Академия», 2009, 2010. – 288 с.

10. **Вендров А.М.** Проектирование программного обеспечения экономических информационных систем: Учебник. – М.: Финансы и статистика, 2006.

11**. Вендров А.М.** Практикум по проектированию программного обеспечения экономических информационных систем: Учеб. пособие – М.: Финансы и статистика, 2006.

12. **Бройдо В.Л., Крылова В.С.** Научные основы организации управления и построения АСУ. – М.: Высш.шк., 1990. – 192с.: ил.

13. **Советов Б.Я.** Информационная технология. Учебник для ВУЗов по спец. «Автоматизированные системы обработки информации и управления». – М.: Высш.шк. 1994. – 368с.: ил.

14. **Строгний А.А.** Построение концептуальной модели систем управления. УС и М. 1988. №2. с.60.

15. Учебный комплекс INTUIT.RU (версия 1.0) Интернет- университета Информационных технологий (www.intuit.ru): Второе высшее образование дома: «Проектирование информационных систем»; «Базы данных».