МОСКОВСКИЙ ИНСТИТУТ РАДИОТЕХНИКИ, ЭЛЕКТРОНИКИ И АВТОМАТИКИ (ТУ)

**Курсовая работа по предмету системное программное обеспечение**

Тема: Архитектура аппаратно-программных средств распределенной обработки информации для интранет-технологии.

Студента группы ВВ-22-95

Головченко В.

Преподаватель Малыгина О.П.

Москва 1998

# Содержание

# 1. Архитектура “клиент-сервер”

1.1. Открытые системы

# 1.2. Клиенты и серверы локальных сетей

# 1.3. Системная архитектура “клиент-сервер”

# 1.4. Серверы баз данных

1.5. Принципы взаимодействия между клиентскими

 и серверными частями

1.6. Преимущества протоколов удаленного вызова

 процедур

1.7. Типичное разделение функций между клиентами

 и серверами

1.8. Архитектуры процессора базы данных

2. Трехуровневая архитектура “клиент-сервер”

3. Программные средства разработки

3.1. Универсальные средства

3.2. Персональные СУБД

4. Intranet и архитектура “клиент-сервер”.

4.1. Двухуровневая архитектура “клиент-сервер”

4.2. Трехуровневая архитектура “клиент-сервер”

4.2.1. Программы расширения серверной части

5. Пример базы данных

**1. Архитектура "клиент-сервер"**

Применительно к системам баз данных архитектура "клиент-сервер" интересна и актуальна главным образом потому, что обеспечивает простое и относительно дешевое решение проблемы коллективного доступа к базам данных в локальной сети.

**1.1. Открытые системы**

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

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

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

Практической опорой системных и прикладных программных средств открытых систем является стандартизованная операционная система. В настоящее время такой системой является UNIX. Фирмам-поставщикам различных вариантов ОС UNIX в результате длительной работы удалось придти к соглашению об основных стандартах этой операционной системы. Сейчас все распространенные версии UNIX в основном совместимы по части интерфейсов, предоставляемых прикладным (а в большинстве случаев и системным) программистам. Как кажется, несмотря на появление претендующей на стандарт системы Windows NT, именно UNIX останется основой открытых систем в ближайшие годы.

Технологии и стандарты открытых систем обеспечивают реальную и проверенную практикой возможность производства системных и прикладных программных средств со свойствами мобильности (portability) и интероперабельности (interoperability). Свойство *мобильности* означает сравнительную простоту переноса программной системы в широком спектре аппаратно-программных средств, соответствующих стандартам. *Интероперабельность* означает упрощения комплексирования новых программных систем на основе использования готовых компонентов со стандартными интерфейсами.

Преимуществом для пользователей является то, что они могут постепенно заменять компоненты системы на более совершенные, не утрачивая работоспособности системы. В частности, в этом кроется решение проблемы постепенного наращивания вычислительных, информационных и других мощностей компьютерной системы.

**1.2. Клиенты и серверы локальных сетей**

В основе широкого распространения локальных сетей компьютеров лежит известная идея разделения ресурсов. Высокая пропускная способность локальных сетей обеспечивает эффективный доступ из одного узла локальной сети к ресурсам, находящимся в других узлах.

Развитие этой идеи приводит к функциональному выделению компонентов сети: разумно иметь не только доступ к ресурсами удаленного компьютера, но также получать от этого компьютера некоторый сервис, который специфичен для ресурсов данного рода и программные средства. Так мы приходим к различению рабочих станций и серверов локальной сети.

*Рабочая станция* предназначена для непосредственной работы пользователя или категории пользователей и обладает ресурсами, соответствующими локальным потребностям данного пользователя.

*Сервер* локальной сети должен обладать ресурсами, соответствующими его функциональному назначению и потребностям сети. Заметим, что в связи с ориентацией на подход открытых систем, правильнее говорить о логических серверах (имея в виду набор ресурсов и программных средств, обеспечивающих услуги над этими ресурсами), которые располагаются не обязательно на разных компьютерах. Особенностью логического сервера в открытой системе является то, что если по соображениям эффективности сервер целесообразно переместить на отдельный компьютер, то это можно проделать без потребности в какой-либо переделке как его самого, так и использующих его прикладных программ.

Примерами сервером могут служить:

•сервер телекоммуникаций, обеспечивающий услуги по связи данной локальной сети с внешним миром;

•вычислительный сервер, дающий возможность производить вычисления, которые невозможно выполнить на рабочих станциях;

•дисковый сервер, обладающий расширенными ресурсами внешней памяти и предоставляющий их в использование рабочим станциями и, возможно, другим серверам;

•файловый сервер, поддерживающий общее хранилище файлов для всех рабочих станций;

•сервер баз данных фактически обычная СУБД, принимающая запросы по локальной сети и возвращающая результаты.

Сервер локальной сети предоставляет ресурсы (услуги) рабочим станциям и/или другим серверам.

Принято называть клиентом локальной сети, запрашивающий услуги у некоторого сервера и сервером - компонент локальной сети, оказывающий услуги некоторым клиентам.

**1.3. Системная архитектура "клиент-сервер"**

Понятно, что в общем случае, чтобы прикладная программа, выполняющаяся на рабочей станции, могла запросить услугу у некоторого сервера, как минимум требуется некоторый интерфейсный программный слой, поддерживающий такого рода взаимодействие (было бы по меньшей мере неестественно требовать, чтобы прикладная программа напрямую пользовалась примитивами транспортного уровня локальной сети). Из этого, собственно, и вытекают основные принципы системной архитектуры "клиент-сервер".

Система разбивается на две части, которые могут выполняться в разных узлах сети, - клиентскую и серверную части. Прикладная программа или конечный пользователь взаимодействуют с клиентской частью системы, которая в простейшем случае обеспечивает просто надсетевой интерфейс. Клиентская часть системы при потребности обращается по сети к серверной части. Заметим, что в развитых системах сетевое обращение к серверной части может и не понадобиться, если система может предугадывать потребности пользователя, и в клиентской части содержатся данные, способные удовлетворить его следующий запрос.

Интерфейс серверной части определен и фиксирован. Поэтому возможно создание новых клиентских частей существующей системы (пример интероперабельности на системном уровне).

Основной проблемой систем, основанных на архитектуре "клиент-сервер", является то, что в соответствии с концепцией открытых систем от них требуется мобильность в как можно более широком классе аппаратно-программных решений открытых систем. Даже если ограничиться UNIX-ориентированными локальными сетями, в разных сетях применяется разная аппаратура и протоколы связи. Попытки создания систем, поддерживающих все возможные протоколы, приводит к их перегрузке сетевыми деталями в ущерб функциональности.

Еще более сложный аспект этой проблемы связан с возможностью использования разных представлений данных в разных узлах неоднородной локальной сети. В разных компьютерах может существовать различная адресация, представление чисел, кодировка символов и т.д. Это особенно существенно для серверов высокого уровня: телекоммуникационных, вычислительных, баз данных.

Общим решением проблемы мобильности систем, основанных на архитектуре "клиент-сервер" является опора на программные пакеты, реализующие протоколы *удаленного вызова процедур* (RPC - Remote Procedure Call). При использовании таких средств обращение к сервису в удаленном узле выглядит как обычный вызов процедуры. Средства RPC, в которых, естественно, содержится вся информация о специфике аппаратуры локальной сети и сетевых протоколов, переводит вызов в последовательность сетевых взаимодействий. Тем самым, специфика сетевой среды и протоколов скрыта от прикладного программиста.

При вызове удаленной процедуры программы RPC производят преобразование форматов данных клиента в промежуточные машинно-независимые форматы и затем преобразование в форматы данных сервера. При передаче ответных параметров производятся аналогичные преобразования.

*Если система реализована на основе стандартного пакета RPC, она может быть легко перенесена в любую открытую среду.*

Технология “клиент-сервер” применительно к СУБД сводится к разделению системы на две части – приложение-клиент (front-end) и сервер базы данных (back-end). Эта архитектура совмещает лучшие черты обработки данных на мэйнфреймах и технологии “файл-сервер”. От мэйнфреймов технология “клиент-сервер” позаимствовала такие черты, как централизованное администрирование, безопасность, надежность. От технологии “файл-сервер” унаследованы низкая стоимость и возможность распределенной обработки данных, используя ресурсы компьютеров-клиентов. Сейчас графический интерфейс пользователя стал стандартом для систем “клиент-сервер”. Кроме того, архитектура “клиент-сервер” значительно упрощает и ускоряет разработку приложений за счет того, что правила проверки целостности данных находятся на сервере. Неправильно работающее кли­ентское приложение не может привести к потере или искаже­нию данных. Все эти возможности, ранее свойственные только сложным и дорогостоящим системам, сейчас доступны даже небольшим организациям. Стоимость оборудования, про­граммного обеспечения и обслуживания для персональных компьютеров в десятки раз ниже, чем для мэйнфреймов.

Особенности обработки данных в различных архитектурах по­казаны на рис.1.

Рис.1. Обработка данных в различных архитектурах

Локальный компьютер

 Локальное приложение

 СУБД

 Данные

Архитектура “файл-сервер”

 Клиент

 Файл-сервер

 Сетевое приложение

 Данные

 СУБД

 Клиент

 пересылка Сетевое приложение

 данных

 СУБД

Архитектура “клиент-сервер”

 Сервер БД

 Клиентское

 СУБД приложение

 Данные

 Клиентское приложение

пересылка запросов

 и результатов

**1.4. Серверы баз данных**

Термин "сервер баз данных" обычно используют для обозначения всей СУБД, основанной на архитектуре "клиент-сервер", включая и серверную, и клиентскую части. Такие системы предназначены для хранения и обеспечения доступа к базам данных.

Хотя обычно одна база данных целиком хранится в одном узле сети и поддерживается одним сервером, серверы баз данных представляют собой простое и дешевое приближение к распределенным базам данных, поскольку общая база данных доступна для всех пользователей локальной сети.

**1.5. Принципы взаимодействия между клиентскими и серверными частями**

Доступ к базе данных от прикладной программы или пользователя производится путем обращения к клиентской части системы. В качестве основного интерфейса между клиентской и серверной частями выступает язык баз данных SQL.

Это язык по сути дела представляет собой текущий стандарт интерфейса СУБД в открытых системах. Собирательное название SQL-сервер относится ко всем серверам баз данных, основанных на SQL.

Серверы баз данных, интерфейс которых основан исключительно на языке SQL, обладают своими преимуществами и своими недостатками. Очевидное преимущество – стандартность интерфейса. В пределе, хотя пока это не совсем так, клиентские части любой SQL-ориентированной СУБД могли бы работать с любым SQL-сервером вне зависимости от того, кто его произвел.

Недостаток тоже довольно очевиден. При таком высоком уровне интерфейса между клиентской и серверной частями системы на стороне клиента работает слишком мало программ СУБД. Это нормально, если на стороне клиента используется маломощная рабочая станция. Но если клиентский компьютер обладает достаточной мощностью, то часто возникает желание возложить на него больше функций управления базами данных, разгрузив сервер, который является узким местом всей системы.

Одним из перспективных направлений СУБД является гибкое конфигурирование системы, при котором распределение функций между клиентской и пользовательской частями СУБД определяется при установке системы.

**1.6. Преимущества протоколов удаленного вызова процедур**

Упоминавшиеся выше протоколы удаленного вызова процедур особенно важны в системах управления базами данных, основанных на архитектуре "клиент-сервер".

Во-первых, использование механизма удаленных процедур позволяет действительно перераспределять функции между клиентской и серверной частями системы, поскольку в тексте программы удаленный вызов процедуры ничем не отличается от удаленного вызова, и следовательно, теоретически любой компонент системы может располагаться и на стороне сервера, и на стороне клиента.

Во-вторых, механизм удаленного вызова скрывает различия между взаимодействующими компьютерами. Физически неоднородная локальная сеть компьютеров приводится к логически однородной сети взаимодействующих программных компонентов. В результате пользователи не обязаны серьезно заботиться о разовой закупке совместимых серверов и рабочих станций.

**1.7. Типичное разделение функций между клиентами и серверами**

В типичном на сегодняшний день случае на стороне клиента СУБД работает только такое программное обеспечение, которое не имеет непосредственного доступа к базам данных, а обращается для этого к серверу с использованием языка SQL.

В некоторых случаях хотелось бы включить в состав клиентской части системы некоторые функции для работы с "локальным кэшем" базы данных, т.е. с той ее частью, которая интенсивно используется клиентской прикладной программой. В современной технологии это можно сделать только путем формального создания на стороне клиента локальной копии сервера базы данных и рассмотрения всей системы как набора взаимодействующих серверов.

С другой стороны, иногда хотелось бы перенести большую часть прикладной системы на сторону сервера, если разница в мощности клиентских рабочих станций и сервера чересчур велика. В общем-то при использовании RPC это сделать нетрудно. Но требуется, чтобы базовое программное обеспечение сервера действительно позволяло это. В частности, при использовании ОС UNIX проблемы практически не возникают.

**1.8. Архитектуры процессора базы данных.**

Основная часть любой системы “клиент-сервер” – это сервер БД. Со времени возникновения архитектуры “клиент-сервер” появилось много вариантов архитектуры процессора БД, поскольку он во многом определяет успех всей системы. Основное требование к серверу БД – обеспечение минимального времени выполнения запросов при максимально возможном числе пользователей. Существуют две основные архитектуры для построения процессора БД: архитектура с несколькими процессами и многопоточная архитектура.

1. *Архитектура с несколькими процессами*

Характеризуется тем, что несколько экземпляров исполняемого файла работают одновременно. Эти системы отличаются хорошей масштабируемостью, но требуют значительных расходов памяти, так как память каждому экземпляру приложения выделяется отдельно. Эта архитектура подразумевает наличие эффективного механизма взаимодействия процессов и полагается на операционную систему при разделении процессорного времени между отдельными экземплярами приложения. Самый известный пример сервера, построенного по этой архитектуре, - Oracle Server. Когда пользователь подключается к БД Oracle, он в действительности запускает отдельный экземпляр исполняемого файла процессора базы данных.

2. *Многопоточная архитектура*

Эта архитектура использует только один исполняемый файл, с несколькими потоками исполнения. Главное преимущество – более скромные требования к оборудованию, чем для архитектуры с несколькими процессами. Здесь сервер берет на себя разделение времени между отдельными потоками, иногда давая преимущество некоторым задачам над другими. Кроме того, отпадает необходимость в сложном механизме взаимодействия процессов. По этой архитектуре построены MS SQL Server и Sybase SQL Server.

**2. Трехуровневая архитектура “клиент-сервер”**

На верхнем уровне абстрагирования взаимодействия клиента и сервера достаточно четко можно выделить следующие компоненты:

•презентационная логика (Presentation Layer - **PL**), предназначенная для работы с данными пользователя;

•бизнес-логика (Business Layer - **BL**), предназначенная для проверки правильности данных, поддержки ссылочной целостности ..;

•логика доступа к ресурсам (Access Layer - **AL**), предназначенная для хранения данных;

Таким образом можно, можно придти к нескольким моделям клиент-серверного взаимодействия :

1. *"Толстый" клиент.* (fat client)

 Сервер БД Пользовательский интерфейс

 Данные Бизнес-логика

 Пользовательский интерфейс

 Бизнес-логика

Наиболее часто встречающийся вариант реализации архитектуры клиент-сервер в уже внедренных и активно используемых системах. Такая модель подразумевает объединение в клиентском приложении как PL, так и BL, таким образом обеспечивается полная децентрализация управления бизнес-логикой. Однако в случае необходимости выполнения каких-либо изменений в клиентском приложении придется менять исходный код. Серверная часть, при описанном подходе, представляет собой сервер баз данных, реализующий AL. К описанной модели часто применяют аббревиатуру RDA - Remote Data Access.

2*. "Тонкий" клиент.* (thin client)

 Бизнес-

 Логика Пользовательский интерфейс

 Данные

 Пользовательский интерфейс

Модель, начинающая активно использоваться в корпоративной среде в связи с распространением Internet-технологий и, в первую очередь, Web-браузеров. В этом случае клиентское приложение обеспечивает реализацию PL, поэтому клиент может довольствоваться довольно скромной аппаратной платформой, а сервер объединяет BL и AL. Максимальная загрузка сервера предусматривает выполнение бизнес-логики только с помощью хранимых процедур сервера (*Хранимые процедуры* – откомпилированные SQL-инструкции, хранящиеся на сервере). Это позволяет максимально централизовать контроль над данными и легко изменять правила работы сразу для целого предприятия. С другой стороны, незначительная корректировка правил, касающаяся только части пользователей, потребует длительной процедуры согласования. В этом случае невозможно реализовать какие-то исключения из общих правил для некоторых пользователей или приложений. В принципе, это хорошо и является залогом безопасности и целостности данных.

3. *Сервер бизнес-логики. (трехуровневая архитектура)*

 Промежуточный сервер

 Пользовательский

 Бизнес-логика интерфейс

 второго уровня

 Сервер БД

 Пользовательский

 Бизнес-логика интерфейс

 сервера

 Данные

Модель с физически выделенным в отдельное приложение блоком BL, таким образом получаем трехуровневую архитектуру “клиент-сервер”. На сервере БД может функционировать “универсальная” часть бизнес-логики (правила на уровне предприятия или группы связанных приложений). Такая схема позволяет поддерживать тонких клиентов на пользовательских компьютерах и в то же время разгрузить сервер БД от чрезмерной загрузки при сохранении гибкой системы работы с бизнес-правилами. В качестве промежуточного сервера может использоваться второй SQL-сервер, но чаще рациональней задействовать персональную СУБД, которая менее требовательна к аппаратным ресурсам и может обеспечить удобные средства построения и поддержки бизнес-логики.

## 3. Программные средства разработки

## 3.1. Универсальные средства

Для разработки клиентских приложений существует громадное число универсальных пакетов программ, которые позволяют выполнить соединение с сервером и разработать для пользователя удобный графический интерфейс, позволяющий эффективно работать с данными. Некоторые из этих средств для разработки приложений в архитектуре “клиент-сервер” перечислены в таблице.

|  |  |
| --- | --- |
| Наименование | Краткая характеристика |
| **CA-OpenROAD** | Полнофункциональная объектно-ориентированная среда для разработки приложений на основе языка четвертого поколения 4GL.  |
| **Delphi** **Client/Server** | Универсальный пакет для разработки клиентских приложений. Обеспечивает объектно-ориентированную разработку с использованием визуальных средств. Поддерживает групповую работу над приложением.  |
| **Magic 6.0** | Таблично-управляемый инструментарий для разработки трехуровневых приложений “клиент-сервер”.  |
| **MS Visual Basic 5.0** | Универсальный пакет разработки пользовательских приложений. Обеспечивает визуальное построение форм и компиляцию приложения. В полном объеме поддерживаются OLE 2.0 и OLE Automation. Для работы с данными предназначен визуальный инструментарий Visual Database Tools. |
| **PowerBuilder 4.0** | Объектно-ориентированное средство разработки приложений “клиент-сервер”. Имеет мощные визуальные средства; поддерживает стандарты OLE и ODBC. |
| **Progress 8** | Пакет поддерживает компонентную объектно-ориентированную разработку приложений. Используется новая технология SmartObject и среда компонентов приложения (ACE). |
| **SAS System** | Обеспечивает инструментарий для доступа, управления, анализа и представления данных в приложении для громадного числа систем и компьютерных платформ, включая мэйнфреймы. Имеет 35 видов интерфейса для различных систем и язык программирования четвертого поколения. Поддерживает ODBC. |
| **Uniface Six** | Независимая среда разработки. Поддерживает управление на уровне модели и компонентное программирование. Имеет мощные визуальные средства. Допускает групповую разработку. Имеет интерфейс к более чем 30 серверам БД на различных платформах. |

**3.2. Персональные СУБД.**

Для разработки клиентских приложений в большинстве случаев вместо универсальных средств разработки удобнее использовать персональные СУБД. Использование персональных СУБД позволяет не только эффективно организовывать работу с бизнес-правилами, но и поддержать независимую работу клиентского приложения за счет наличия собственных форматов хранения данных. Краткая характеристика некоторых персональных СУБД приведена в таблице.

|  |  |
| --- | --- |
| Наименование | Краткая характеристика |
| **Lotus Approach 97** | Позволяет выполнять все виды обработки данных. Имеет очень простой интерфейс. СУБД тесно интегрирована с базами данных Notes и электронными таблицами Lotus 1-2-3. Поддерживает технологию электронного обмена сообщениями MAPI. |
| **MS Access 97** | Полнофункциональная СУБД, обладающая богатым набором визуальных средств, многочисленными мастерами и мощным языком программирования Visual Basic for Applications. Имеет гибкую систему подготовки отчетов. Поддерживаются технологии ODBC и OLE 2.0. СУБД тесно интегрирована со всеми приложениями MS Office.  |
| **MS Visual FoxPro 5** | Одна из наиболее быстрых персональных СУБД, сочетающая технологию xBase и объектно-ориентированный язык программирования. Имеет богатый набор визуальных средств разработки и мастеров для быстрого построения приложений и отчетов. Поддерживаются технологии ActiveX, ODBC и OLE 2.0. Позволяет создавать OLE-сервера и имеет очень развитые средства разработки и поддержки приложений “клиент-сервер”. |
| **Paradox 7** | Поддерживает все виды работы с данными. Для визуального выполнения стандартных задач имеется специальное средство Experts. Наделен собственным достаточно сложным языком ObjectPAL. Поддерживает технологии OLE 2.0, ActiveX, MAPI и ODBC. |

**4. Intranet и архитектура “клиент-сервер”.**

**4.1. Двухуровневая архитектура “клиент-сервер”**

 Web-броузер Источник данных

 Web-сервер

 NOS (Network Operation System)

Разграничение функций между Web-броузером и Web-сервером является очень четким. Web-сервер предоставляет HTML-страницы, а броузер отображает эти страницы путем интерпретации тегов HTML.

**4.2. Трехуровневая архитектура “клиент-сервер”**

 Web-броузер Источник данных

 Третий уровень

 Программа

 расширения

 сервера

 HTML

 Web-сервер

 NOS

Клиентский уровень занимает броузер, на уровне сервера находится сервер БД, а на промежуточном уровне располагаются Web-сервер и программа расширения сервера. Такое архитектурное решение позволяет уменьшить сетевой трафик, делает компоненты взаимозаменяемыми и повышает уровень безопасности. Однако такая архитектура также затрудняет обработку транзакций БД ввиду природы протокола HTTP, не запоминающего состояния (этот протокол использует для передачи данных между броузером и сервером БД).

Броузер посылает Web-серверу запросы на доставку Web-страниц или данных. Web-сервер обслуживает заявки на Web-страницы, а запросы отправляет программе-расширению серверной части. Последняя принимает передаваемые ей запросы, преобразует их в форму, понятную серверу БД, и передает их серверу БД.

Затем сервер БД выполняет работу по обслуживанию запроса и возвращает результат программе-расширению серверной части. Наконец та преобразует результаты в формат, приемлемый для броузера, и передает их Web-серверу, а тот в свою очередь – броузеру.

**4.2.1. Программы расширения серверной части**

Одной из главных причин использования программ-расширений серверной части на промежуточном уровне является возможность использовать стандарты, существующих для двух крайних уровней, путем осуществления трансляции между ними. Другие применения расширений серверной части состоят в поддержании соединений между БД с целью уменьшить трафик в сети и в поддержании резерва соединений между БД для уменьшения затрат ресурсов на открытие/закрытие БД. Расширения серверной части также поддерживают взаимозаменяемость в своих стандартных интерфейсах. Поэтому Web-серверы и серверы БД можно сравнительно легко заменять или наращивать.

Существует три категории расширений серверной части: с обычным CGI, с гибридным CGI и с API.

**5. Пример базы данных**

Пример базы данных см. в прилагаемом к курсовой работе техническом задании.

Источники:

1. А.Горев, С.Макашарипов, Ю.Владимиров

“SQL Server 6.5 для профессионалов”

Изд. “Питер” Санкт-Петербург 1998

1. К.Ланг, Д.Чоу

“Публикация баз данных в Интернете”

Изд. “Символ-Плюс” Санкт-Петербург 1998

1. Д.Боуман, C.Эмерсон, М.Дарновски

“Практическое руководство по SQL”

Изд. “Диалектика” Киев 1997

1. Microsoft Press

“Секреты создания интрасетей”

Изд. “Питер” Санкт-Петербург 1998

5. http:\\www.citforum.ru