Содержание:

Введение

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.

Реальное распространение архитектуры клиент-сервер стало возможным благодаря развитию и широкому внедрению в практику концепции открытых систем.

Основным смыслом подхода открытых систем является упрощение комплексирования вычислительных систем за счет международной и национальной стандартизации аппаратных и программных интерфейсов.

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 ресурсам удаленного компьютера, но также получать от этого компьютера некоторый сервис, который специфичен для ресурсов данного рода и программные средства. Так мы приходим к различению клиентов (рабочих станций) и серверов локальной сети.

Технология клиент-сервер применительно к СУБД сводится к разделению системы на две части – приложение-клиент и сервер базы данных. 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.

Цель курсовой работы – изучение основных понятий и моделей архитектуры клиент-сервер.

Для достижения поставленной цели необходимо решить следующие задачи:

- рассмотреть основные понятия архитектуры клиент-сервер;

- раскрыть сущность преимущества архитектуры клиент-сервер;

- рассмотреть варианты архитектуры клиент-сервер.

Объект исследования – варианты архитектуры клиент-сервер.

Глава 1. Теоретическая часть

1.1 Основные определения

Архитектура информационной системы – концепция, определяющая модель, структуру, выполняемые функции и взаимосвязь компонентов информационной системы[1].

Клиент-сервер (*Client-server*) – вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг (сервисов), называемыми серверами, и заказчиками услуг, называемыми клиентами [2].

Сервер – (англ. «server» – обслуживающий процессор, узел обслуживания) понимают подключенную к сети, достаточно мощную вычислительную машину, обладающую определёнными ресурсами общего пользования, а также, как правило, возможностью объединять некоторое количество компьютеров как в локальной, так и в глобальной информационных сетях. Сетевые узлы с серверами, называют хостами (англ. «host» – хозяин)[3]. Также к данному определению можно отнести программы, представляющая какие-то услуги другим программам и обслуживающая запросы клиентов на получение ресурсов определенного вида.

Клиент – это задача, рабочая станция, пользователь. Он может сформировать запрос для сервера: считать файл, осуществить поиск записи и т.п. Клиентский процесс в архитектуре клиент-сервер – процесс, который выполняется на стороне клиента и посылает запрос серверному процессу на выполнение некоторой задачи [4].

Часто люди клиентом или сервером просто называют компьютер, на котором работает какая-либо из этих программ. В сущности, клиент и сервер — это роли, исполняемые программами. Клиенты и сервера физически могут находиться на одном компьютере. Одна и та же программа может быть и клиентом, и сервером одновременно.

Вообще говоря, клиент-серверная система характеризуется наличием двух взаимодействующих самостоятельных процессов – клиента и сервера, которые, в общем случае, могут выполняться на разных компьютерах, обмениваясь данными по сети. По такой схеме могут быть построены системы обработки данных на основе СУБД, почтовые и другие системы.

1.2 Преимущества архитектуры клиент-сервер

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

В большинстве случаев программа обработки (клиентская часть) расположена на одном компьютере, а сама база данных хранится на другом. Помимо хранения централизованной базы данных центральная машина (сервер базы данных) обеспечивает выполнение основного объема обработки данных. Запрос на данные, выдаваемый клиентом (рабочей станцией), порождает поиск и извлечение данных на сервере. Извлеченные данные транспортируются по сети от сервера к клиенту. Причем, по сети передается только полезная информация[5].

Также преимущество архитектуры в том, что постоянно идет работа по совершенствованию самого метода хранения и обработки информации, и если его реализация (т.е. сервер базы данных) сменилась, то не потребуется перекомпилировать с новыми библиотеками все разработанные программы, а достаточно будет инсталлировать новый сервер базы данных взамен старого и перевести базы данных в формат нового сервера, применив для этого прилагаемую к нему утилиту. Так можно сделать, если новый сервер придерживается тех же правил обмена между ним и программой пользователя, что и старый.

Используя множество небольших компьютеров, разработчики систем клиент-сервер могут эмулировать вычислительную мощность больших ЭВМ, распределяя прикладную задачу по различным микрокомпьютерам и серверам. Каждый из них

берет на себя свою часть вычислительной нагрузки, используя информацию совместно с другими процессорами сети. Суть идеи в том, чтобы повысить мощность системы, не увеличивая производительность одного компьютера, а суммируя средства многих.

Быстродействие – основной фактор целесообразности разработки систем для архитектуры клиент-сервер. Применение средств быстрой разработки программ (Rapid Application Development - RAD) позволяет разработчикам создавать прикладные системы для архитектуры клиент-сервер в рекордно короткие сроки. Технология серверов баз данных также становится проще в использовании и сочетается в одних системах со средствами RAD. Таким образом, сокращается время, необходимое для подготовки и передачи прикладной программы пользователю[6].

Спецификой архитектуры клиент-сервер является использование языка запросов SQL.

SQL (Structured Query Language – язык структурированных запросов) – универсальный язык, предназначенный для создания и выполнения запросов, обработки данных как в собственной базе данных приложения, так и с базами данных, созданных другими приложениями, поддерживающими SQL.

Microsoft Access, Microsoft Visual FoxPro, Microsoft Visual Basic обеспечивают средства для создания клиентских частей в приложениях клиент-сервер, которые сочетают в себе средства просмотра, графический интерфейс и средства построения запросов, а Microsoft SQL Server является на сегодняшний день одним из самых мощных серверов баз данных[5].

2.2 Варианты клиент-серверной архитектуры

Клиент-серверная архитектура в вычислительной сети может быть реализована по-разному. Выбор конкретной схемы определяется различными вариантами территориального распределении удаленных подразделений предприятия, требованиями эксплуатационной надежности, быстродействием, простотой обслуживания. Различные схемы клиент - серверной архитектуры представлены на рис. 1.[7]

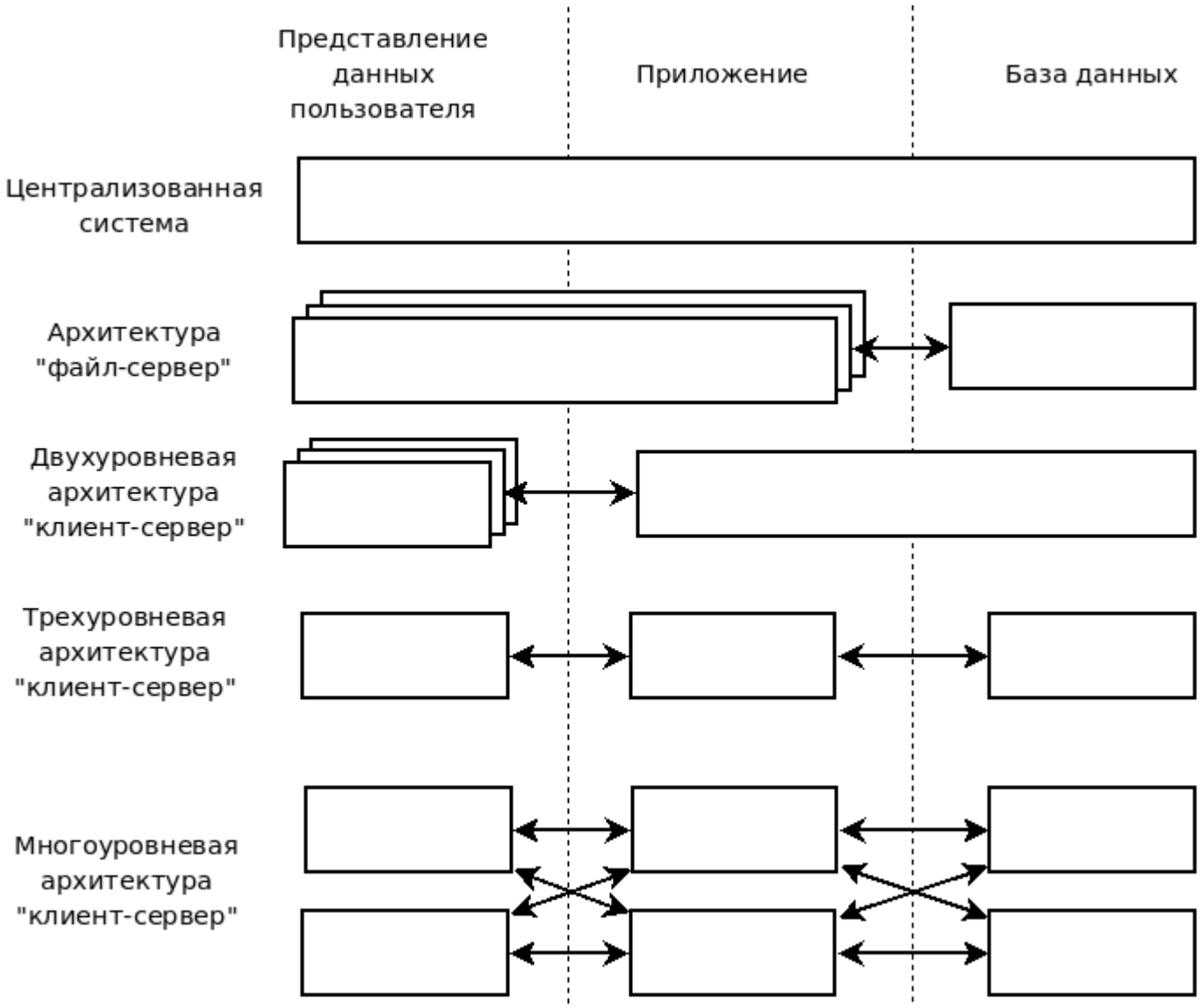

Рис. 1 Варианты клиент-серверной архитектуры

Централизованная архитектура вычислительных систем была распространена в 70-х и 80-х годах и реализовывалась на базе мейнфреймов (например, IBM-360/370 или их отечественных аналогов серии ЕС ЭВМ), либо на базе мини-ЭВМ (например, PDP-11 или их отечественного аналога СМ-4)[8].

Характерная особенность такой архитектуры – полная «неинтеллектуальность» терминалов. Их работой управляет хост-ЭВМ.

Достоинства такой архитектуры[9],[10]:

- пользователи совместно используют дорогие ресурсы ЭВМ и дорогие периферийные устройства;

- централизация ресурсов и оборудования облегчает обслуживание и эксплуатацию вычислительной системы;
- отсутствует необходимость администрирования рабочих мест пользователей;

Главным недостатком для пользователя является то, что он полностью зависит от администратора хост-ЭВМ. Пользователь не может настроить рабочую среду под свои потребности – все используемое программное обеспечение является коллективным.

Использование такой архитектуры является оправданным, если хост-ЭВМ очень дорогая, например, супер-ЭВМ.

Классическое представление централизованной архитектуры показано на рис. 2[11]:

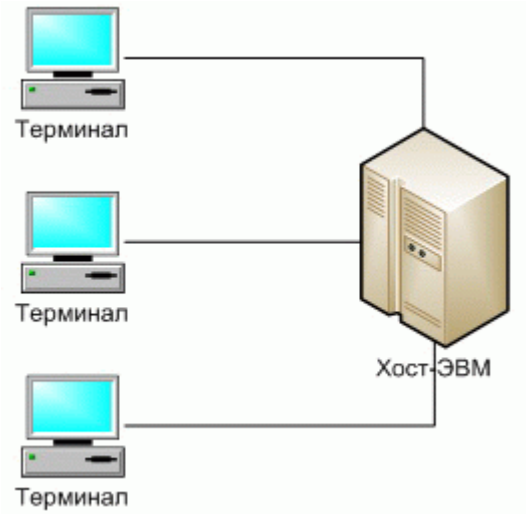

Рис. 2 Классическое представление централизованной архитектуры

Центральная ЭВМ должна иметь большую память и высокую производительность, чтобы обеспечивать комфортную работу большого числа пользователей.

Все приложения, работающие в такой архитектуре, полностью находятся в основной памяти хост-ЭВМ. Терминалы являются лишь устройствами ввода-вывода и таким образом в минимальной степени поддерживают интерфейс пользователя [12].

Файл – серверная архитектура представляет наиболее простой случай распределенной обработки данных, согласно которой на сервере располагаются только файлы данных, а на клиентской части находятся приложения пользователей вместе с СУБД. Файл-сервер в среде сетевой операционной системы

организует доступ к файлам, полностью эквивалентным файлам операционной системы и расположенным во внешней памяти файл – сервера. При данном подходе программы СУБД располагаются в оперативной памяти рабочих станций локальной сети, а файлы базы данных – на магнитных дисках файл-сервера. Специальный интерфейсный модуль распознает, где находятся файлы, к которым осуществляется обращение. В связи с этим данная СУБД может работать как с локальными базами данных, так и с центральной базой данных. Синхронизация совместного использования базы данных файл-сервера возлагается на систему управления базами данных, которая должна обеспечивать блокирование записей на время их корректировки, чтобы сделать их недоступными с других рабочих станций. Использование файл-серверов предполагает, что вся обработка данных выполняется на рабочей станции, а файл-сервер лишь выполняет функции накопителя данных и средств доступа[13].

Файл-серверные приложения – приложения, схожие по своей структуре с локальными приложениями и использующие сетевой ресурс для хранения программы и данных[14].

Функции сервера: хранения данных и кода программы. Функции клиента: обработка данных происходит исключительно на стороне клиента.

Классическое представление информационной системы в архитектуре "файл-сервер" представлено на рис. 3[15]

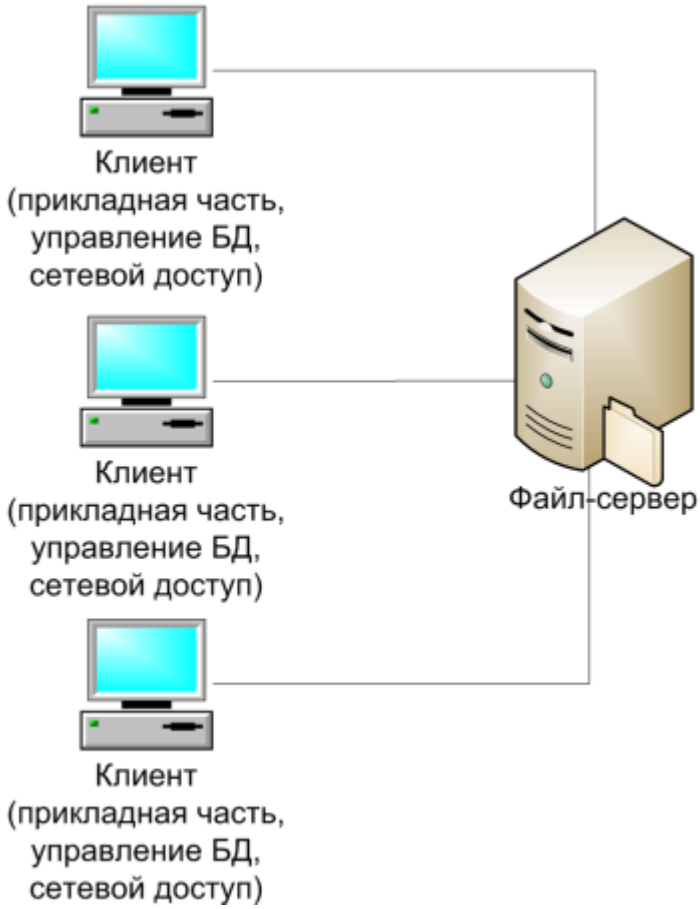

Рис. 3 Классическое представление архитектуры "файл-сервер"

Организация информационных систем на основе использования выделенных файл-серверов все еще является распространенной в связи с наличием большого количества персональных компьютеров разного уровня развитости и сравнительной дешевизны связывания PC в локальные сети[16].

Конечно, основным достоинством данной архитектуры является простота организации. Проектировщики и разработчики информационной системы находятся в привычных и комфортных условиях IBM PC в среде MS-DOS, Windows или какого-либо облегченного варианта Windows Server. Имеются удобные и развитые средства разработки графического пользовательского интерфейса, простые в использовании средства разработки систем баз данных и/или СУБД.

Достоинства такой архитектуры[17]:

- многопользовательский режим работы с данными;
- удобство централизованного управления доступом;
- низкая стоимость разработки;
- высокая скорость разработки;

- невысокая стоимость обновления и изменения ПО.

Недостатки[18]:

- проблемы многопользовательской работы с данными: последовательный доступ, отсутствие гарантии целостности;
- низкая производительность (зависит от производительности сети, сервера, клиента);
- плохая возможность подключения новых клиентов;
- ненадежность системы.

Простое, работающее с небольшими объемами информации и рассчитанное на применение в однопользовательском режиме, файл-серверное приложение можно спроектировать, разработать и отладить очень быстро[19]. Очень часто для небольшой компании для ведения, например, кадрового учета достаточно иметь изолированную систему, работающую на отдельно стоящем PC. Однако, в уже ненамного более сложных случаях (например, при организации информационной системы поддержки проекта, выполняемого группой) файл-серверные архитектуры становятся недостаточными.

Таким образом, непосредственным манипулированием данными занимается несколько независимых и несогласованных между собой процессов. Кроме того, для осуществления любой обработки (поиск, модификация, суммирование и т.п.) все данные необходимо передать по сети с сервера на рабочую станцию[20].

Клиент-сервер (Client-server) – вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг (сервисов), называемых серверами, и заказчиками услуг, называемых клиентами [21]. Нередко клиенты и серверы взаимодействуют через компьютерную сеть и могут быть как различными физическими устройствами, так и программным обеспечением.

Первоначально системы такого уровня базировались на классической двухуровневой клиент-серверной архитектуре (Two-tierarchitecture). Под клиент-серверным приложением в этом случае понимается информационная система, основанная на использовании серверов баз данных.

Схематически такую архитектуру можно представить, как показано на рис. 4[22].

Рис. 4 Классическое представление архитектуры "клиент-сервер"

На стороне клиента выполняется код приложения, в который обязательно входят компоненты, поддерживающие интерфейс с конечным пользователем, производящие отчеты, выполняющие другие специфичные для приложения функции.

Клиентская часть приложения взаимодействует с клиентской частью программного обеспечения управления базами данных, которая, фактически, является индивидуальным представителем СУБД для приложения.

Заметим, что интерфейс между клиентской частью приложения и клиентской частью сервера баз данных, как правило, основан на использовании языка SQL. Поэтому такие функции, как, например, предварительная обработка форм, предназначенных для запросов к базе данных, или формирование результирующих отчетов выполняются в коде приложения.

Наконец, клиентская часть сервера баз данных, используя средства сетевого доступа, обращается к серверу баз данных, передавая ему текст оператора языка SQL[23].

Посмотрим теперь, что же происходит на стороне сервера баз данных. В продуктах практически всех компаний сервер получает от клиента текст оператора на языке SQL. Сервер производит компиляцию полученного оператора. Далее (если компиляция завершилась успешно) происходит выполнение оператора.

Разработчики и пользователи информационных систем, основанных на архитектуре "клиент-сервер", часто бывают неудовлетворены постоянно существующими сетевыми накладными расходами, которые следуют из потребности обращаться от клиента к серверу с каждым очередным запросом. На практике распространена ситуация, когда для эффективной работы отдельной клиентской составляющей информационной системы в действительности требуется только небольшая часть общей базы данных. Это приводит к идее поддержки локального кэша общей базы данных на стороне каждого клиента.

Фактически, концепция локального кэширования базы данных является частным случаем концепции реплицированных баз данных. Как и в общем случае, для поддержки локального кэша базы данных программное обеспечение рабочих станций должно содержать компонент управления базами данных – упрощенный вариант сервера баз данных, который, например, может не обеспечивать многопользовательский режим доступа. Отдельной проблемой является обеспечение согласованности (когерентности) кэшей и общей базы данных. Здесь возможны различные решения – от автоматической поддержки согласованности за счет средств базового программного обеспечения управления базами данных до полного перекладывания этой задачи на прикладной уровень.

Преимуществами данной архитектуры являются[24],[25]:

- возможность, в большинстве случаев, распределить функции вычислительной системы между несколькими независимыми компьютерами в сети;
- все данные хранятся на сервере, который, как правило, защищен гораздо лучше большинства клиентов, а также на сервере проще обеспечить контроль полномочий, чтобы разрешать доступ к данным только клиентам с соответствующими правами доступа;
- поддержка многопользовательской работы;
- гарантия целостности данных.

Недостатки[26],[27]:

- неработоспособность сервера может сделать неработоспособной всю вычислительную сеть;

- администрирование данной системы требует квалифицированного профессионала;
- высокая стоимость оборудования;
- бизнес логика приложений осталась в клиентском ПО.

При проектировании информационной системы, основанной на архитектуре "клиент-сервер", большее внимание следует обращать на грамотность общих решений. Технические средства пилотной версии могут быть минимальными (например, в качестве аппаратной основы сервера баз данных может использоваться одна из рабочих станций). После создания пилотной версии нужно провести дополнительную исследовательскую работу, чтобы выяснить узкие места системы. Только после этого необходимо принимать решение о выборе аппаратуры сервера, которая будет использоваться на практике.

Увеличение масштабов информационной системы не порождает принципиальных проблем. Обычным решением является замена аппаратуры сервера (и, может быть, аппаратуры рабочих станций, если требуется переход к локальному кэшированию баз данных). В любом случае практически не затрагивается прикладная часть информационной системы. Также данный вид архитектуры называют архитектурой с "толстым" клиентом[28].

Здесь логика представления данных и бизнес-логика размещаются на клиенте, который (скажем, в случае, когда сервером является СУБД) общается с логикой хранения и накопления данных на сервере, используя язык структурированных запросов SQL. Однако необходимость установки "толстых клиентов", требующих значительного количества специальных библиотек и специальной настройки окружения, на большое число пользовательских компьютеров с различными операционными средами, как правило, вызывает массу проблем[29].

Как альтернатива возникла также двухзвенная архитектура "с тонким клиентом". При этом в идеале программа-клиент реализует лишь графический интерфейс пользователя (GUI) и передает/принимает запросы, а вся бизнес-логика выполняется сервером. В идеале клиентом является просто интернет-браузер, который имеется в стандартной операционной среде любого пользовательского компьютера и не требует специальной настройки, установки специализированного ПО и т.п. К сожалению, такая схема тоже не свободна от недостатков, хотя бы уже потому, что серверу иногда приходится брать на себя несвойственные для него функции реализации бизнес логики приложения (например, серверу СУБД приходится выполнять расчеты)[30].

Двухуровневая клиент-серверная архитектура основана на использовании только сервера базы - данных (DB-сервера), когда клиентская часть содержит уровень представления данных, а на сервере находится база данных вместе с СУБД и прикладными программами.
DB-сервер отличается от файл-сервера тем, что в его оперативной памяти, помимо сетевой операционной системы, функционирует централизованная СУБД, которая обеспечивает совместное использование рабочими станциями базы данных, размещенной во внешней памяти этого DB-сервера.
DB-сервер дает возможность отказаться от пересылки по сети файлов данных целиком и передавать только ту выборку из базы данных, которая удовлетворяет запросу пользователя. При этом возможно разделение пользовательского приложения на две части: одна часть выполняется на сервере и связана с выборкой и агрегированием данных из базы данных, а вторая часть по представлению данных для анализа и принятия решения выполняется на клиентской машине. Таким образом, увеличивается общая производительность информационной системы в результате объединения вычислительных ресурсов сервера и клиентской рабочей станции[31].

Обращение к базе данных осуществляется на языке SQL, который фактически стал стандартом для реляционных баз данных. Отсюда сервер баз данных часто называют SQL-сервером, который поддерживается всеми реляционными СУБД: Oracle, Informix, MS SQL, ADABAS D, InterBase, SyBase и др. Клиентское приложение может быть реализовано на языке настольных СУБД (MS Access, FoxPro, Paradox, Clipper и др.). При этом взаимодействие клиентского приложения с SQL-сервером осуществляется через ODBC-драйвер (Open Data Base Connectivity), который обеспечивает возможность пересылки и преобразования данных из глобальной базы данных в структуру базы данных клиентского приложения. Применение этой технологии позволило разработчикам не заботиться о специфике работы с той или иной СУБД и делать свои системы переносимыми между базами данных. За время своего существования ODBC стал стандартом де-факто на алгоритм доступа к разнородным базам данных, и на сегодняшний день насчитывается более 160 прикладных систем, которые работают с источниками информации через драйверы ODBC[32].

Трехуровневая клиент-серверная архитектура позволяет помещать прикладные программы на отдельные серверы приложений, с которыми через API-интерфейс (Application Program Interface) устанавливается связь клиентских рабочих станций. Работа клиентской части приложения сводится к вызову

необходимых функций сервера приложения, которые называются «сервисами». Прикладные программы в свою очередь обращаются к серверу базы данных с помощью SQL запросов. Такая организация позволяет еще более повысить производительность и эффективность КИС за счет:

•многократности повторного использования общих функций обработки данных в множестве клиентских приложений при существенной экономии системных ресурсов;

•параллельности в работе сервера приложений и сервера базы данных, причем сервер приложений может быть менее мощным по сравнению с сервером базы данных;

•оптимизации доступа к базе данных через сервер приложений из клиентских мест путем диспетчеризации выполнения запросов в вычислительной сети;

•повышения скорости и надежности обработки данных в результате дублирования программного обеспечения на нескольких серверах приложений, которые могут заменять друг друга в сети в случае перегрузки или выхода из строя одного из них;

•переноса функций администрирования системы по проверке полномочий доступа пользователей с сервера базы данных на сервер приложений[33]’[34].

Многоуровневая архитектура «Клиент-сервер» создается для территориально-распределенных предприятий. Для нее в общем случае характерны отношения «многие ко многим» между клиентскими рабочими станциями и серверами приложений, между серверами приложений и серверами баз данных. Такая организация позволяет более рационально организовать информационные потоки между структурными подразделениями в процессе выполнения общих деловых процессов. Интегрированная база данных находится на отдельном сервере, на котором обеспечиваются централизованное ведение и администрирование общих данных для всех приложений.
Многоуровневая архитектура клиент-сервер (Multitier architecture) – разновидность архитектуры клиент-сервер, в которой функция обработки данных вынесена на один или несколько отдельных серверов[35]. Это позволяет разделить функции хранения, обработки и представления данных для более эффективного использования возможностей серверов и клиентов.

Среди многоуровневой архитектуры клиент-сервер наиболее распространена трехуровневая архитектура (трехзвенная архитектура, three-tier), предполагающая

наличие следующих компонентов приложения: клиентское приложение (обычно говорят "тонкий клиент" или терминал), подключенное к серверу приложений, который в свою очередь подключен к серверу базы данных[36]. Схематически такую архитектуру можно представить, как показано на рис. 5[37]

Рис. 5 Представление многоуровневой архитектуры "клиент-сервер"

Выделение нескольких серверов баз данных особенно актуально для предприятий с филиальной структурой, когда в центральном офисе используется общая база данных, содержащая общую нормативно - справочную, планово-бюджетную информацию и консолидированную отчетность, а в территориально-удаленных филиалах поддерживается оперативная информация о деловых процессах. При обработке данных в филиалах для контроля используется плановая и нормативно-справочная информация из центральной базы данных, а в центральном офисе получение консолидированной отчетности сопряжено с обработкой оперативной информации филиалов.
Для сокращения объема передачи данных по каналам связи в распределенной информационной системе предлагается репликация данных, то есть тиражирование данных на взаимодействующих серверах баз данных с автоматическим поддержанием соответствия копий данных. При этом возможны следующие режимы репликации:

• синхронный режим, когда тиражируемые данные обновляются по мере возникновения необходимости одновременно на серверах баз данных во всех

копиях. Требуемое быстродействие каналов для синхронного режима - единицы Мбит в секунду;

• асинхронный режим, когда тиражирование данных выполняется в строго определенные моменты времени, например каждый час работы информационной системы. Требуемое быстродействие каналов для асинхронного режима - единицы Кбит в секунду. Асинхронный режим может вызывать откладывание выполнения транзакций до момента обновления данных[38].

Направление тиражирования между серверами баз данных может быть:

• равноправным, т.е. в обоих направлениях;

• сверху-вниз типа «ведущий/ведомый», когда на серверах филиалов содержатся только некоторые подмножества данных центральной базы данных;

• снизу-вверх по консолидирующей схеме, когда при обновлении данных в филиалах в определенные моменты времени обновляется центральная база данных.

В простейшей конфигурации физически сервер приложений может быть совмещен с сервером базы данных на одном компьютере, к которому по сети подключается один или несколько терминалов.

В «правильной» (с точки зрения безопасности, надежности, масштабирования) конфигурации сервер базы данных находится на выделенном компьютере (или кластере), к которому по сети подключены один или несколько серверов приложений, к которым, в свою очередь, по сети подключаются терминалы.

Плюсами данной архитектуры являются[39],[40]:

- клиентское ПО не нуждается в администрировании;
- масштабируемость;
- конфигурируемость – изолированность уровней друг от друга позволяет быстро и простыми средствами переконфигурировать систему при возникновении сбоев или при плановом обслуживании на одном из уровней;
- высокая безопасность;
- высокая надежность;
- низкие требования к скорости канала (сети) между терминалами и сервером приложений;

- низкие требования к производительности и техническим характеристикам терминалов, как следствие снижение их стоимости.

Минусы[41] [, 11]:

- растет сложность серверной части и, как следствие, затраты на администрирование и обслуживание;
- более высокая сложность создания приложений;
- сложнее в разворачивании и администрировании;
- высокие требования к производительности серверов приложений и сервера базы данных, а, значит, и высокая стоимость серверного оборудования;
- высокие требования к скорости канала (сети) между сервером базы данных и серверами приложений.

Некоторые авторы (например, Мартин Фаулер[42]) представляют многозвенную архитектуру (трехзвенную) в виде пяти уровней (рис. 6)[43]:

1. Представление;
2. Уровень представления;
3. Уровень логики;
4. Уровень данных;
5. Данные.

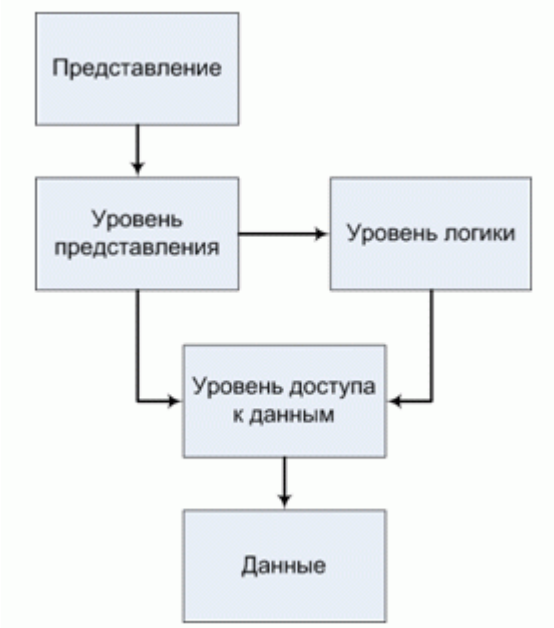

Рис. 6 Пять уровней многозвенной архитектуры "клиент-сервер"

К представлению относится вся информация, непосредственно отображаемая пользователю: сгенерированные html-страницы, таблицы стилей, изображения.

Уровень представления охватывает все, что имеет отношение к общению пользователя с системой. К главным функциям слоя представления относятся отображение информации и интерпретация вводимых пользователем команд с преобразованием их в соответствующие операции в контексте логики и данных.

Уровень логики содержит основные функции системы, предназначенные для достижения поставленной перед ним цели. К таким функциям относятся вычисления на основе вводимых и хранимых данных, проверка всех элементов данных и обработка команд, поступающих от слоя представления, а также передача информации уровню данных.

Уровень доступа к данным – это подмножество функций, обеспечивающих взаимодействие со сторонними системами, которые выполняют задания в интересах приложения.

Данные системы обычно хранятся в базе данных[44].

Таким образом, многоуровневая архитектура клиент-сервер - вариант для территориально-распределённых предприятий. Наиболее удачным примером будет предприятие с легионом филиалов со своими копиями БД, адаптированными под региональные и функциональные обстоятельства, и центральным офисом с интегрированной (полной) базой данных для централизованного ведения и администрирования общих данных для всех филиалов.

Глава 2. Практическая часть

В практической части курсовой работы требуется, используя ППП на ПК, рассчитать оптимальное сочетание цены и количества произведенного товара при максимальном значении получаемой прибыли путем задания переменных издержек на единицу товара. Наибольшую прибыль обеспечивают такой объем выпуска и цена, при которых предельные издержки максимально приближены к предельной выручке или равны ей. По данным таблицы нужно построить гистограмму с заголовком, названием осей координат и легендой.

Решение задачи будет производиться в среде табличного процессора Microsoft Excel.

Описание алгоритма решения задачи можно представить в виде инфологической модели (рис. 7).

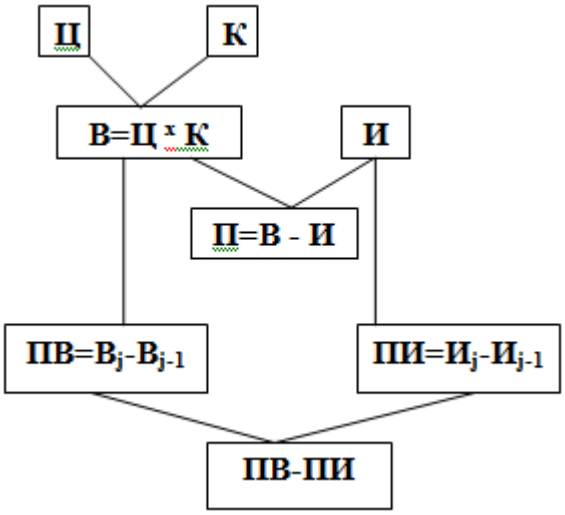

Рис. 7. Инфологическая модель решения задачи

Ц – цена товара

К – количество товара

И – суммарные издержки

В – выручка от реализации

П – прибыль

Bj – текущая выручка от реализации

Bj-1 – предыдущая выручка от реализации

ПВ – предельная выручка

$И_j$ – текущие суммарные издержки

Иj-1 – предыдущие суммарные издержки

ПИ – предельные издержки

ПВ-ПИ – наибольшая прибыль

При выборе программного обеспечения я остановился на использовании Microsoft Excel для решения экономической задачи. Вся задача заключается в вычислении

одной таблицы, следовательно, использовать Microsoft Access нецелесообразно, т.к. для вычисления нужно использовать построитель выражений, который эффективно работает только при построении запросов.

Проектирование форм выходных документов и графическое представление данных по выбранной задаче.

Таблицу с исходными данными на рабочем листе Microsoft Excel переношу в Microsoft Word. Создаю структуру шаблона таблицы:

Колонка электронной таблицы	Наименование (реквизит)	Тип данных	Формат данных	
			длина	точность
A	№ п/п	числовой	2	
B	Цена тыс. руб. (Ц)	числовой	5	3
C	Количество (К)	числовой	4	
D	Суммарные издержки, тыс. руб. (И)	числовой	3	
E	Выручка от реализации В=Ц * К	числовой	4	
F	Прибыль П=В-И	числовой	3	
G	Предельная выручка ПВ=Bj-Bj-1	числовой	2	
H	Предельные издержки ПИ=Иj-Иj-1	числовой	2	

I	ПВ-ПИ	числовой	1

Создаю шаблон таблицы:

Выручка от реализации В=Ц * К	**Прибыль П=В-И**	**Предельная выручка ПВ=Bj-Bj-1**	**Предельные издержки ПИ=Иj-Иj-1**	**ПВ-ПИ**
E4=B4*C4	F4=E4-D4	x	x	x
E5=B5*C5	F5=E5-D5	G5 =E5-E4	H5 =D5-D4	I5=G5-H5
E6=B6*C6	F6=E6-D6	G6 =E6-E5	H6 =D6-D5	I6 =G6-H6
E7=B7*C7	F7=E7-D7	G7 =E7-E6	H7 =D7-D6	I7 =G7-H7
E8=B8*C8	F8=E8-D8	G8 =E8-E7	H8 =D8-D7	I8 =G8-H8
E9=B9*C9	F9=E9-D9	G9 =E9-E8	H9 =D9-D8	I9 =G9-H9
E10=B10*C10	F10=E10-D10	G10 =E10-E9	H10 =D10-D9	I10 =G10-H10

E11=B11*C11	F11=E11-D11	G11 =E11-E10	H11 =D11-D10	I11 =G11-H11	
E12=B12*C12	F12=E12-D12	G12 =E12-E11	H12 =D12-D11	I12 =G12-H12	
E13=B13*C13	F13=E13-D13	G13 =E13-E12	H13 =D13-D12	I13 =G13-H13	
E14=B14*C14	F14=E14-D14	G14 =E14-E13	H14 =D14-D13	I14 =G14-H14	
E15=B15*C15	F15=E15-D15	G15 =E15-E14	H15 =D15-D14	I15 =G15-H15	

После подсчетов итоговую таблицу переношу из Microsoft Excel в Microsoft Word.

Оптимальное сочетание цены и количества произведенного товара при максимальном значении получаемой прибыли

18.03.2019

№ п/п	Цена тыс. руб. (Ц)	Количество (К)	Суммарные издержки, тыс. руб. (И)	Выручка от реализации В=Ц * К	Прибыль П=В-И	Предельная выручка ПВ=Bj-Bj-1	Предельные издержки ПИ=Иj-Иj-1	ПВ-ПИ

1	0,35	3500	750	1225	475			
2	0,345	3600	765	1242	477	17	15	2
3	0,34	3700	776	1258	482	16	11	5
4	0,335	3800	784	1273	489	15	8	7
5	0,33	3900	795	1287	492	14	11	3
6	0,325	4000	806	1300	494	13	11	2
7	0,32	4100	814	1312	498	12	8	4
8	0,315	4200	822	1323	501	11	8	3
9	0,31	4300	830	1333	503	10	8	2
10	0,305	4400	838	1342	504	9	8	1
11	0,3	4500	844	1350	506	8	6	2
12	0,295	4600	852	1357	505	7	8	-1

Представим результаты вычислений в графическом виде (рис. 8).

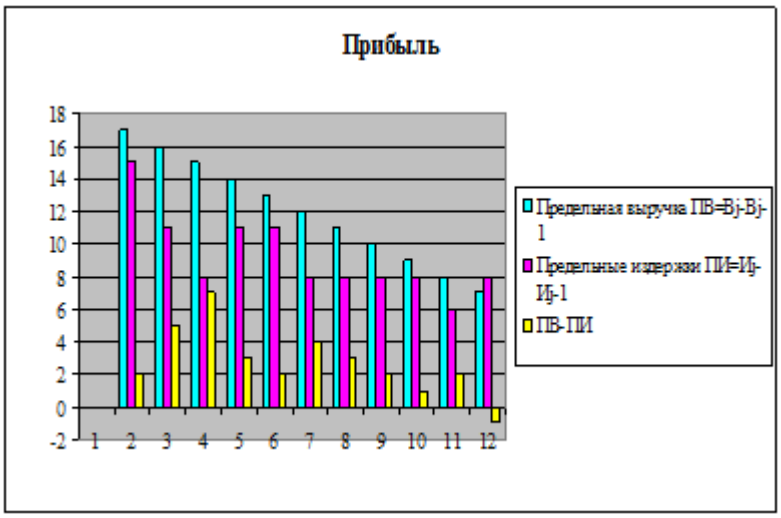

Рис. 8. Гистограмма «Прибыль»

Инструкция пользователя:

1. Открываем табличный процессор Microsoft Excel.
2. Создаем таблицу «Оптимальное сочетание цены и количества произведенного товара при максимальном значении получаемой прибыли». Заголовок пишу в ячейке A1. Т.к. таблица имеет девять столбцов, то нужно объединить ячейки от A1 до I1, используя кнопку на панели инструментов «объединить и поместить в центре».
3. В ячейках с адресами от A3 до I3 пишем названия колонок таблицы. Чтобы задать ячейке размер необходимо поставить курсор мыши на правую границу ячейки и щелкнуть два раза.
4. Заполняем колонки данными. Тип данных в колонках числовой.
5. Необходимо ввести значение даты между таблицей и ее названием. Для этого нужно объединить ячейки от A2 до I2, используя кнопку на панели инструментов «объединить и поместить в центре».
6. Для того чтобы произвести вычисления в таблице, я активизирую ячейку E4, ставлю знак =, активизирую ячейку B4, ставлю знак *, активизирую ячейку C4 и нажимаю клавишу Enter. В строке формул видим: E4=B4*C4.
7. Аналогично проводятся вычисления для других ячеек столбца «Выручка от реализации».
8. Для более быстрого вычисления элементов ячеек используем автозаполнение. Для этого нужно поставить курсор мыши на нижний правый угол активизированной ячейки и довести его до конца данного столбца (т.е. копировать ячейки). Таким способом можно производить расчеты для ячеек

всех столбцов, а также заполнять ячейки уже предложенными данными, где числа изменяются на одинаковую величину.

9. После произведения всех необходимых расчетов требуется построить гистограмму. Для этого на панели инструментов выбираем кнопку «Мастер диаграмм».
10. В открытом окне «Мастер диаграмм» выбираем нужный тип, в данном случае гистограмму. Нажимаем кнопку «Далее».
11. Для создания гистограммы нужно щелкнуть в поле «Диапазон». Затем указать на листе ячейки, содержащие необходимые для построения данные.Нажимаем кнопку «Далее».
12. Для создания легенды нужно в поле «Легенда» указать значок «Добавить легенду» и ее размещение (например, справа). Нажимаем кнопку «Далее».
13. Затем указываем размещение гистограммы. В данном случае надо отметить «размещение на имеющемся листе». Нажимаем кнопку «Готово».

Задачи подобного типа возможно решать вручную и используя вышеописанный проект. Однако при вычислении вручную будет затрачено больше времени, чем по разработанному программному решению. Также при преобразованиях вручную существует большая вероятность совершения ошибок. Данный проект обеспечивает простоту, быстроту и точность вычислений. Он является работоспособным, и по нему можно решать задачи подобного типа.

Заключение

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.

Реальное распространение архитектуры клиент-сервер стало возможным благодаря развитию и широкому внедрению в практику концепции открытых систем.

Основной проблемой систем, основанных на архитектуре клиент-сервер, является то, что в соответствии с этой концепцией от них требуется мобильность в как можно более широком классе аппаратно-программных решений открытых систем.

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

Библиография

Трутнев Д.Р. Архитектуры информационных систем. Основы проектирования. Санкт-Петербург: Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, 2012. URL: http://books.ifmo.ru/file/pdf/919.pdf

Коржов В.В. Многоуровневые системы клиент-сервер. Изд.: Открытые системы, 1997. URL: http://www.osp.ru/nets/1997/06/142618

Никандрова Ю.А. Интернет-курс по дисциплине «Информационные технологии». Московский финансово-промышленный университет «Университет», 2011. URL: http://e-biblio.ru/book/bib/01_informatika/inform_tech/sg.html#_Toc322338433

Власенко В.И. Интернет-курс по дисциплине «Информационные сети». Москва: Московская финансово-промышленная академия, 2010. URL: http://e-biblio.ru/book/bib/01_informatika/informacionnye%20sety/sg.html#_Toc268626612

Култыгин О.П. Интернет-курс по дисциплине «Методы и средства проектирования ин-формационных систем и технологий». Московский финансово-промышленный университет «Университет», 2016. URL: http://e-biblio.ru/book/bib/01_informatika/metod_i_sredstv_proekt_inform_system/sg.html#_Toc356814

Смирнова Г.Н. Проектирование экономических информационных систем. Москва: Московский государственный университет экономики, статистики и информатики, 2004. URL: https://www.studmed.ru/smirnova-gn-telnov-yuf-proektirovanie-ekonomicheskih-informacionnyh-sistem-chast-1_f4d46b0cd50.html

Павлов А.В. Архитектура вычислительных систем. Учебное пособие. Санкт-Петербург: Университет ИТМО, 2016. – 86 с. URL: http://books.ifmo.ru/file/pdf/2074.pdf

Никандрова Ю.А. Интернет-курс по дисциплине «Базы данных (Управление данными)». Москва: Московский финансово-промышленный университет «Университет», 2012. URL: http://e-biblio.ru/book/bib/01_informatika/bazy_dannyx/sg.html#_Toc326925640

Степанов Е.О., Ярцев Б.М. Учебно-методическое пособие по дисциплине «Архитектуры и технологии разработки распределенного программного обеспечения». - СПб: СПбГУ ИТМО, 2008. - 103 с.

Грошев А.С. Базы данных: Учебное пособие. – Архангельск, Издательство Архангельского государственного технического университета, 2005. – 124 с. URL: https://studfiles.net/preview/2983990

Несвижский А.И., Рябов В.А. Современные веб-технологии. Электронная книга. 2 изд., испр. — М.: Интуит, 2016. — 1001 с.

Мартин Фаулер. Архитектура корпоративных программных приложений: [Пер. с англ.] / Мартин Фаулер; При участии Дейвида Райса [и др.]. – М.: Вильямс, 2004. – 539 с.

1. **Д. Р. Трутнев**. Архитектуры информационных систем. Основы проектирования. Санкт-Петербург: Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, 2012. URL: http://books.ifmo.ru/file/pdf/919.pdf ↑

2. **Валерий Коржов**. Многоуровневые системы клиент-сервер. Изд.: Открытые системы, 1997. URL: http://www.osp.ru/nets/1997/06/142618 ↑

3. **Никандрова Ю.А**. Интернет-курс по дисциплине «Информационные технологии». Московский финансово-промышленный университет «Университет», 2011. URL: http://e-biblio.ru/book/bib/01_informatika/inform_tech/sg.html#_Toc322338433 ↑

4. **Никандрова Ю.А**. Интернет-курс по дисциплине «Информационные технологии». Московский финансово-промышленный университет «Университет», 2011. URL: http://e-biblio.ru/book/bib/01_informatika/inform_tech/sg.html#_Toc322338433 ↑

5. **Власенко В.И.** Интернет-курс по дисциплине «Информационные сети». Москва: Московская финансово-промышленная академия, 2010. URL: http://e-biblio.ru/book/bib/01_informatika/informacionnye%20sety/sg.html#_Toc268626612 ↑

6. **Култыгин О.П.** Интернет-курс по дисциплине «Методы и средства проектирования информационных систем и технологий». Московский финансово-промышленный университет «Университет», 2016. URL: http://e-biblio.ru/book/bib/01_informatika/metod_i_sredstv_proekt_inform_system/sg.html#_Toc356 ↑

7. **Смирнова Г.Н.** Проектирование экономических информационных систем. Москва: Московский государственный университет экономики, статистики и информатики, 2004. URL: https://www.studmed.ru/smirnova-gn-telnov-yuf-proektirovanie-ekonomicheskih-informacionnyh-sistem-chast-1_f4d46b0cd50.html ↑

8. **А.В. Павлов**. Архитектура вычислительных систем. Учебное пособие. Санкт-Петербург: Университет ИТМО, 2016. – 86 с. URL: http://books.ifmo.ru/file/pdf/2074.pdf ↑

9. **А.В. Павлов**. Архитектура вычислительных систем. Учебное пособие. Санкт-Петербург: Университет ИТМО, 2016. – 86 с. URL: http://books.ifmo.ru/file/pdf/2074.pdf ↑

10. **Власенко В.И.** Интернет-курс по дисциплине «Информационные сети». Москва: Московская финансово-промышленная академия, 2010. URL: http://e-biblio.ru/book/bib/01_informatika/informacionnye%20sety/sg.html#_Toc268626612 ↑

11. **Смирнова Г.Н.** Проектирование экономических информационных систем. Москва: Московский государственный университет экономики, статистики и информатики, 2004. URL: https://www.studmed.ru/smirnova-gn-telnov-yuf-proektirovanie-ekonomicheskih-informacionnyh-sistem-chast-1_f4d46b0cd50.html ↑

12. **Власенко В.И.** Интернет-курс по дисциплине «Информационные сети». Москва, Московская финансово-промышленная академия, 2010 URL: http://e-biblio.ru/book/bib/01_informatika/informacionnye%20sety/sg.html#_Toc268626612 ↑

13. **Смирнова Г.Н.** Проектирование экономических информационных систем. Москва: Московский государственный университет экономики, статистики и информатики, 2004. URL: https://www.studmed.ru/smirnova-gn-telnov-yuf-proektirovanie-ekonomicheskih-informacionnyh-sistem-chast-1_f4d46b0cd50.html ↑

14. **Власенко В.И.** Интернет-курс по дисциплине «Информационные сети». Москва, Московская финансово-промышленная академия, 2010 URL: http://e-biblio.ru/book/bib/01_informatika/informacionnye%20sety/sg.html#_Toc268626612 ↑

15. **Смирнова Г.Н.** Проектирование экономических информационных систем. Москва: Московский государственный университет экономики, статистики и информатики, 2004. URL: https://www.studmed.ru/smirnova-gn-telnov-yuf-proektirovanie-ekonomicheskih-informacionnyh-sistem-chast-1_f4d46b0cd50.html ↑

16. **Власенко В.И.** Интернет-курс по дисциплине «Информационные сети». Москва, Московская финансово-промышленная академия, 2010 URL: http://e-biblio.ru/book/bib/01_informatika/informacionnye%20sety/sg.html#_Toc268626612 ↑

17. **Никандрова Ю.А**. Интернет-курс по дисциплине «Информационные технологии». Московский финансово-промышленный университет «Университет», 2011. URL: http://e-biblio.ru/book/bib/01_informatika/inform_tech/sg.html#_Toc322338433 ↑

18. **Никандрова Ю.А**. Интернет-курс по дисциплине «Информационные технологии». Московский финансово-промышленный университет «Университет», 2011. URL: http://e-biblio.ru/book/bib/01_informatika/inform_tech/sg.html#_Toc322338433 ↑

19. **Власенко В.И.** Интернет-курс по дисциплине «Информационные сети». Москва, Московская финансово-промышленная академия, 2010 URL: http://e-biblio.ru/book/bib/01_informatika/informacionnye%20sety/sg.html#_Toc268626612 ↑

20. **Смирнова Г.Н.** Проектирование экономических информационных систем. Москва: Московский государственный университет экономики, статистики и информатики, 2004. URL: https://www.studmed.ru/smirnova-gn-telnov-yuf-proektirovanie-ekonomicheskih-informacionnyh-sistem-chast-1_f4d46b0cd50.html ↑

21. **Валерий Коржов**. Многоуровневые системы клиент-сервер. Изд.: Открытые системы, 1997. URL: http://www.osp.ru/nets/1997/06/142618 ↑

22. **Смирнова Г.Н.** Проектирование экономических информационных систем. Москва: Московский государственный университет экономики, статистики и информатики, 2004. URL: https://www.studmed.ru/smirnova-gn-telnov-yuf-proektirovanie-ekonomicheskih-informacionnyh-sistem-chast-1_f4d46b0cd50.html ↑

23. **Власенко В.И.** Интернет-курс по дисциплине «Информационные сети». Москва: Московская финансово-промышленная академия, 2010. URL: http://e-biblio.ru/book/bib/01_informatika/informacionnye%20sety/sg.html#_Toc268626612 ↑

24. **Валерий Коржов**. Многоуровневые системы клиент-сервер. Изд.: Открытые системы, 1997. URL: http://www.osp.ru/nets/1997/06/142618 ↑

25. **Никандрова Ю.А**. Интернет-курс по дисциплине «Информационные технологии». Московский финансово-промышленный университет «Университет», 2011. URL: http://e-biblio.ru/book/bib/01_informatika/inform_tech/sg.html#_Toc322338433 ↑

26. **Валерий Коржов**. Многоуровневые системы клиент-сервер. Изд.: Открытые системы, 1997. URL: http://www.osp.ru/nets/1997/06/142618 ↑

27. **Никандрова Ю.А**. Интернет-курс по дисциплине «Информационные технологии». Московский финансово-промышленный университет «Университет», 2011. URL: http://e-biblio.ru/book/bib/01_informatika/inform_tech/sg.html#_Toc322338433 ↑

28. **Никандрова Ю.А**. Интернет-курс по дисциплине «Информационные технологии». Московский финансово-промышленный университет «Университет», 2011. URL: http://e-biblio.ru/book/bib/01_informatika/inform_tech/sg.html#_Toc322338433 ↑

29. **Власенко В.И.** Интернет-курс по дисциплине «Информационные сети». Москва: Московская финансово-промышленная академия, 2010. URL: http://e-biblio.ru/book/bib/01_informatika/informacionnye%20sety/sg.html#_Toc268626612 ↑

30. **Никандрова Ю.А.** Интернет-курс по дисциплине «Базы данных (Управление данными)». Москва: Московский финансово-промышленный университет «Университет», 2012. URL: http://e-biblio.ru/book/bib/01_informatika/bazy_dannyx/sg.html#_Toc326925640 ↑

31. **Степанов Е.О., Ярцев Б.М**. Учебно-методическое пособие по дисциплине «Архитектуры и технологии разработки распределенного программного обеспечения». - СПб: СПбГУ ИТМО, 2008. - 103 с. ↑

32. **Власенко В.И.** Интернет-курс по дисциплине «Информационные сети». Москва: Московская финансово-промышленная академия, 2010. URL: http://e-biblio.ru/book/bib/01_informatika/informacionnye%20sety/sg.html#_Toc268626612 ↑

33. **Степанов Е.О., Ярцев Б.М**. Учебно-методическое пособие по дисциплине «Архитектуры и технологии разработки распределенного программного обеспечения». - СПб: СПбГУ ИТМО, 2008. - 103 с. ↑

34. **Грошев А.С**. Базы данных: Учебное пособие. – Архангельск, Изд-воАрханг. гос. техн.ун-та,2005. – 124 с. URL: https://studfiles.net/preview/2983990 ↑

35. **А.И. Несвижский, В.А. Рябов**. Современные веб-технологии. Электронная книга. 2 изд., испр. — М.: Интуит, 2016. — 1001 с. ↑

36. **Никандрова Ю.А**. Интернет-курс по дисциплине «Информационные технологии». Московский финансово-промышленный университет «Университет», 2011. URL: http://e-biblio.ru/book/bib/01_informatika/inform_tech/sg.html#_Toc322338433 ↑

37. **А.И. Несвижский, В.А. Рябов**. Современные веб-технологии. Электронная книга. 2 изд., испр. — М.: Интуит, 2016. — 1001 с. ↑

38. **Степанов Е.О., Ярцев Б.М**. Учебно-методическое пособие по дисциплине «Архитектуры и технологии разработки распределенного программного обеспечения». - СПб: СПбГУ ИТМО, 2008. - 103 с. ↑

39. **Никандрова Ю.А**. Интернет-курс по дисциплине «Информационные технологии». Московский финансово-промышленный университет «Университет», 2011. URL: http://e-biblio.ru/book/bib/01_informatika/inform_tech/sg.html#_Toc322338433 ↑

40. **Степанов Е.О., Ярцев Б.М**. Учебно-методическое пособие по дисциплине «Архитектуры и технологии разработки распределенного программного обеспечения». - СПб: СПбГУ ИТМО, 2008. - 103 с. ↑

41. **Никандрова Ю.А**. Интернет-курс по дисциплине «Информационные технологии». Московский финансово-промышленный университет «Университет», 2011. URL: http://e-biblio.ru/book/bib/01_informatika/inform_tech/sg.html#_Toc322338433 ↑

42. **Мартин Фаулер**. Архитектура корпоративных программных приложений : [Пер. с англ.] / Мартин Фаулер; При участии Дейвида Райса [и др.]. - М. [и др.] :

Вильямс, 2004. - 539, [4] с. : ил., табл.; 24 см.; ISBN 5-8459-0579-6 ↑

43. **Грошев А.С**. Базы данных: Учебное пособие. – Архангельск, Изд-воАрханг. гос. техн.ун-та,2005. – 124 с. URL: https://studfiles.net/preview/2983990 ↑

44. **Власенко В.И.** Интернет-курс по дисциплине «Информационные сети». Москва: Московская финансово-промышленная академия, 2010. URL: http://e-biblio.ru/book/bib/01_informatika/informacionnye%20sety/sg.html#_Toc268626612 ↑