Содержание:

ВВЕДЕНИЕ

В последние годы развития информационных и аппаратных технологий можно говорить об увеличенном внимании и частых попытках совершенствования приложений клиент-сервер. На сегодняшний день уже реализованы такие приложения, которые обеспечивают поддержку совместной работы нескольких пользователей, при этом источник данных в сети остается один.

Распространение технологии «клиент-сервер» стало повсеместным при начале общения пользователя с компьютером или с системой, также основанной на работе компьютера. Примерами ежедневного применения технологии «клиент-сервер» являются такие повсеместные действия, о которых мы даже не задумываемся, как телефонная связь или совершение покупок. Применяя автоматический кассовый аппарат, осуществляя считывание штрих-кода с товара, расплачиваясь за совершенную покупку,- мы взаимодействуем с системой компьютера «клиент-сервер».

Технология «клиент-сервер» представляет собой архитектуру взаимодействия, при которой одна программа отправляет запрос на выполнение определенного перечня действий, а другая программа реализует непосредственно выполнение этого перечня. Участниками подобного взаимодействия являются клиент и сервер. Зачастую в качестве клиента или сервера выступает компьютер, который обеспечивает функционирование того или иного программного обеспечения.

Для того чтобы подробнее разобраться в приведенной архитектуре, была определена цель выполнения работы – изучение технологии «клиент-сервер».

Объектом исследования будет выступать сама технология «клиент-сервер».

Для достижения поставленной цели необходимо выполнить следующие задачи:

1. охарактеризовать архитектуру информационной системы;
2. рассмотреть архитектуру «файл-сервер»;
3. кратко охарактеризовать архитектуру «клиент-сервер»;
4. изучить классическую двухуровневую архитектуру «клиент-сервер»;

5. проанализировать трехуровневую архитектуру «клиент-сервер»;
6. произвести анализ программного обеспечения технологии «клиент-сервер».

В структуру работы включены такие элементы, как Введение, Заключение и Список использованных источников. Кроме того, работа содержит две главы, каждая из которых состоит из трех параграфов.

Основной теоретической базой исследования выступают работы современных исследователей проблемы, в их числе – Конноли Т., Мельников В., Морозевич А. Н., Зеневич А. М. и Хоморенко А. Д.

Архитектура информационной системы

Общие характеристики архитектуры информационной системы

Приложения и информационные системы, актуальные для современного уровня развития технологий, могут похвастаться достижением настолько сложного и высокого уровня развития, что применение к ним термина «архитектура» уже давно никого не удивляет.

К сегодняшнему дню известно огромное количество понятий определения «архитектура информационной системы». Рассмотрим определение данного понятия из разных источников:

Архитектура представляет собой организационную структуру системы[1].

Архитектура информационной системы является концепцией, определяющей модель, структуру, выполняемые функции и взаимосвязь составных частей информационной системы[2].

Архитектура представляет собой базовую организация системы, воплощенную в ее компонентах, их отношениях между собой и с окружением, а также принципы, определяющие проектирование и развитие системы[3].

Архитектура является набором значимых решений по поводу организации системы программного обеспечения; набором структурных элементов и их интерфейсов, при помощи которых компонуется система, вместе с их поведением, которое

определяется во взаимодействии между этими элементами; компоновкой элементов в постепенно укрупняющиеся подсистемы, а также стилем архитектуры, направляющим эту организацию – элементы и их интерфейсы, взаимодействия и компоновку[4].

Архитектура программы или компьютерной системы – это структура или структуры системы, которые включают элементы программы, видимые извне свойства этих элементов и связи между ними[5].

Архитектуру информационной системы можно рекурсивно разобрать на части, которые взаимодействуют при помощи интерфейсов, связи, которые соединяют части, и условия сборки частей. Части, которые взаимодействуют через интерфейсы, включают классы, компоненты и подсистемы.

Архитектура программного обеспечения системы или набора систем состоит из всех важных проектных решений по поводу структур программы и взаимодействий между этими структурами, которые составляют системы. Проектные решения обеспечивают желаемый набор свойств, которые должна поддерживать система, чтобы быть успешной. Проектные решения предоставляют концептуальную основу для разработки системы, ее поддержки и обслуживания[6].

Хотя определения несколько отличаются, можно заметить немалую степень сходства. Например, большинство определений указывают на то, что архитектура связана со структурой и поведением, а также только со значимыми решениями, может соответствовать некоторому архитектурному стилю, на нее влияют заинтересованные в ней лица и ее окружение, она воплощает решения на основе логического обоснования.

Под архитектурой программных систем будем понимать совокупность решений относительно:

- организации программной системы;
- выбора структурных элементов, составляющих систему и их интерфейсов;
- поведения этих элементов во взаимодействии с другими элементами;
- объединение этих элементов в подсистемы;
- архитектурного стиля, определяющего логическую и физическую организацию системы: статические и динамические элементы, их интерфейсы и способы их объединения.

Архитектура программной системы охватывает не только ее структурные и поведенческие аспекты, но и правила ее использования и интеграции с другими системами, функциональность, производительность, гибкость, надежность, возможность повторного применения, полноту, экономические и технологические ограничения, а также вопрос пользовательского интерфейса[7].

По мере развития программных систем все большее значение приобретает их интеграция друг с другом с целью построения единого информационного пространства предприятия. Как можно видеть из вышеприведенных определений интеграция является важнейшим элементом архитектуры.

Для того чтобы построить правильную и надежную архитектуру и грамотно спроектировать интеграцию программных систем необходимо четко следовать современным стандартам в этих областях. Без этого велика вероятность создать архитектуру, которая неспособна развиваться и удовлетворять растущим потребностям пользователей ИТ. В качестве законодателей стандартов в этой области выступают такие международные организации как SEI (Software Engineering Institute), WWW (консорциум World Wide Web), OMG (Object Management Group), организация разработчиков Java – JCP (Java Community Process), IEEE (Institute of Electrical and Electronics Engineers) и другие[8].

Рассмотрим классификацию программных систем по их архитектуре:

- Централизованная архитектура;
- Архитектура «файл-сервер»;
- Двухзвенная архитектура «клиент-сервер»;
- Многозвенная архитектура «клиент-сервер»;
- Архитектура распределенных систем;
- Архитектура Веб-приложений;
- Сервис-ориентированная архитектура.

Необходимо иметь в виду тот факт, что, равно как и любую другую, приведенную классификацию нельзя назвать абсолютной. Так, в архитектуре некоторой ИС зачастую очевидны элементы влияния нескольких разных архитектурных взглядов [9].

Архитектура файл-сервер

Принято считать, что первой структурой организации работы информационной системы была архитектура файл-сервер. Такая архитектура осуществляет только извлечение информации из файлов базы данных и передачу ее клиенту для последующей обработки.

Усложнение решения задач, повсеместное распространение персональных компьютеров и вычислительных сетей способствовало становлению архитектуры файл-сервер. Технология файл-сервер обеспечивает назначение одного из компьютеров, состоящих в сети, в качестве выделенного сервера. На нем впоследствии и хранятся файлы базы данных.

Файлы, отобранные в соответствии с запросами пользователей, передаются напрямую с сервера на рабочие станции пользователей. После этого, на рабочих станциях пользователей реализуется основная часть обработки информации. В такой технологии центральный сервер является в большинстве случаев только хранилищем файлов, который при этом не участвует в обработке самой информации (Рисунок 1)[10]

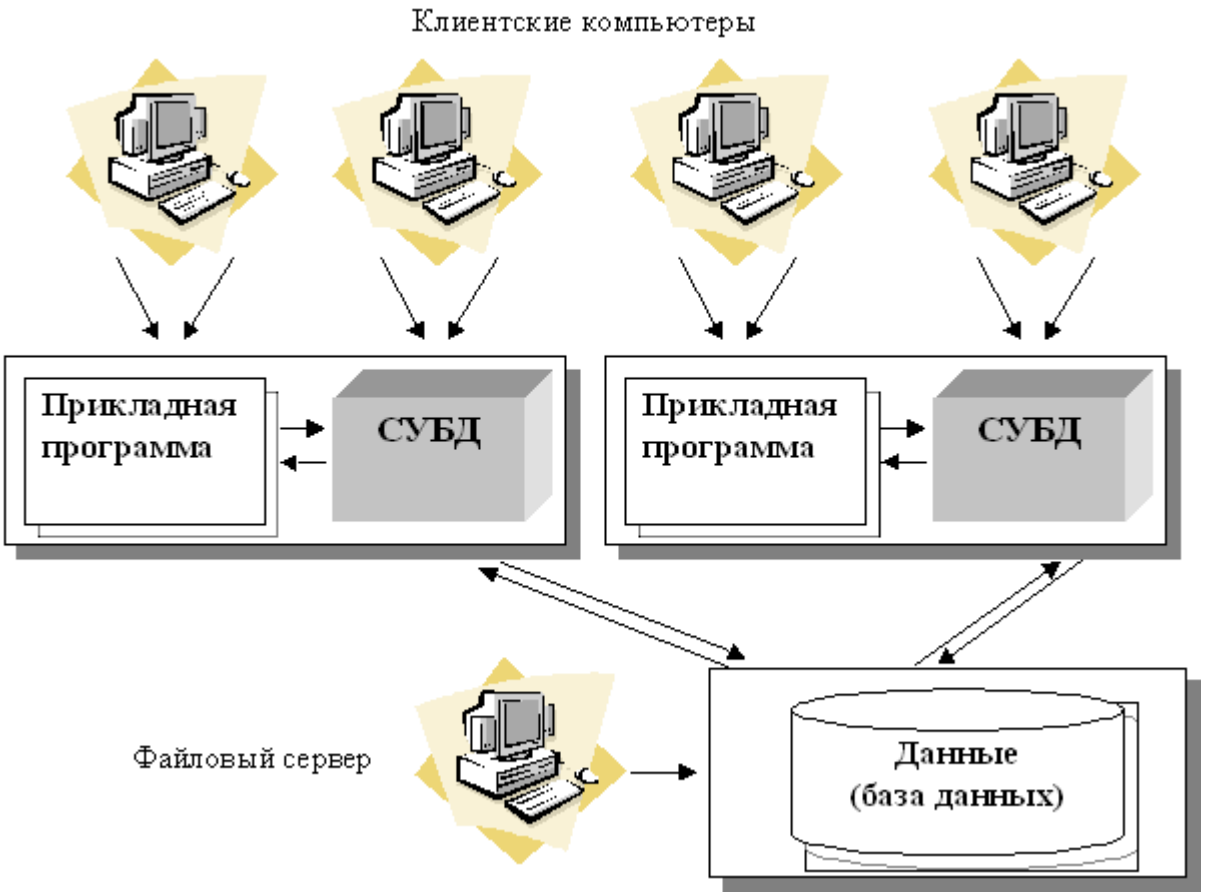

Рисунок 1 Архитектура «файл-сервер»

Работа построена следующим образом:

База данных в виде набора файлов находится на жестком диске специально выделенного компьютера (файлового сервера).

Существует локальная сеть, состоящая из клиентских компьютеров, на каждом из которых установлены СУБД и приложение для работы с БД.

На каждом из клиентских компьютеров пользователи имеют возможность запустить приложение. Используя предоставляемый приложением пользовательский интерфейс, он инициирует обращение к БД на выборку/обновление информации.

Все обращения к БД идут через СУБД, которая инкапсулирует внутри себя все сведения о физической структуре БД, расположенной на файловом сервере.

СУБД инициирует обращения к данным, находящимся на файловом сервере, в результате которых часть файлов БД копируется на клиентский компьютер и обрабатывается, что обеспечивает выполнение запросов пользователя (осуществляются необходимые операции над данными).

При необходимости (в случае изменения данных) данные отправляются назад на файловый сервер с целью обновления БД.

Результат СУБД возвращает в приложение.

Приложение, используя пользовательский интерфейс, отображает результат выполнения запросов[11].

В рамках архитектуры « файл-сервер « были выполнены первые версии популярных так называемых настольных СУБД, таких, как dBase и Microsoft Access.

В литературе указываются следующие основные недостатки данной архитектуры:

1. При одновременном обращении множества пользователей к одним и тем же данным производительность работы резко падает, т.к. необходимо дождаться пока пользователь, работающий с данными, завершит свою работу. В противном случае возможно затирание исправлений, сделанных одними пользователями, изменениями других пользователей.
2. Вся тяжесть вычислительной нагрузки при доступе к БД ложится на приложение клиента, так как при выдаче запроса на выборку информации из

таблицы вся таблица БД копируется на клиентскую машину и выборка осуществляется на клиенте. Таким образом, неоптимально расходуются ресурсы клиентского компьютера и сети. В результате возрастает сетевой трафик и увеличиваются требования к аппаратным мощностям пользовательского компьютера.

3. Как правило, используется навигационный подход, ориентированный на работу с отдельными записями.
4. В БД на файл-сервере гораздо проще вносить изменения в отдельные таблицы, минуя приложения, непосредственно из инструментальных средств (например, из утилиты Database Desktop фирмы Borland для файлов Paradox и dBase); подобная возможность облегчается тем обстоятельством, что фактически у таких СУБД база данных – понятие более логическое, чем физическое, поскольку под БД понимается набор отдельных таблиц, сосуществующих в отдельном каталоге на диске. Все это позволяет говорить о низком уровне безопасности – как с точки зрения хищения и нанесения вреда, так и с точки зрения внесения ошибочных изменений.
5. Недостаточно развитый аппарат транзакций служит потенциальным источником ошибок в плане нарушения смысловой и ссылочной целостности информации при одновременном внесении изменений в одну и ту же запись [12].

Архитектура «клиент-сервер»

Использование технологии «клиент – сервер» предполагает наличие некоторого количества компьютеров, объединенных в сеть, один из которых выполняет особые управляющие функции (является сервером сети).

Так, архитектура « клиент – сервер « разделяет функции приложения пользователя (называемого клиентом) и сервера. Приложение-клиент формирует запрос к серверу, на котором расположена БД, на структурном языке запросов SQL (Structured Query Language), являющемся промышленным стандартом в мире реляционных БД. Удаленный сервер принимает запрос и переадресует его SQL-серверу БД. SQL-сервер – специальная программа, управляющая удаленной базой данных. SQL-сервер обеспечивает интерпретацию запроса, его выполнение в базе данных, формирование результата выполнения запроса и выдачу его приложению-клиенту. При этом ресурсы клиентского компьютера не участвуют в физическом выполнении запроса; клиентский компьютер лишь отсылает запрос к серверной БД

и получает результат, после чего интерпретирует его необходимым образом и представляет пользователю. Так как клиентскому приложению посылается результат выполнения запроса, по сети «путешествуют» только те данные, которые необходимы клиенту. В итоге снижается нагрузка на сеть. Поскольку выполнение запроса происходит там же, где хранятся данные (на сервере), нет необходимости в пересылке больших пакетов данных. Кроме того, SQL-сервер, если это возможно, оптимизирует полученный запростаким образом, чтобы он был выполнен в минимальное время с наименьшими накладными расходами. Архитектура системы представлена на Рисунке 2[13].

Все это повышает быстродействие системы и снижает время ожидания результата запроса. При выполнении запросов сервером существенно повышается степень безопасности данных, поскольку правила целостности данных определяются в базе данных на сервере и являются едиными для всех приложений, использующих эту БД. Таким образом, исключается возможность определения противоречивых правил поддержания целостности. Мощный аппарат транзакций, поддерживаемый SQL-серверами, позволяет исключить одновременное изменение одних и тех же данных различными пользователями и предоставляет возможность откатов к первоначальным значениям при внесении в БД изменений, закончившихся аварийно[14].

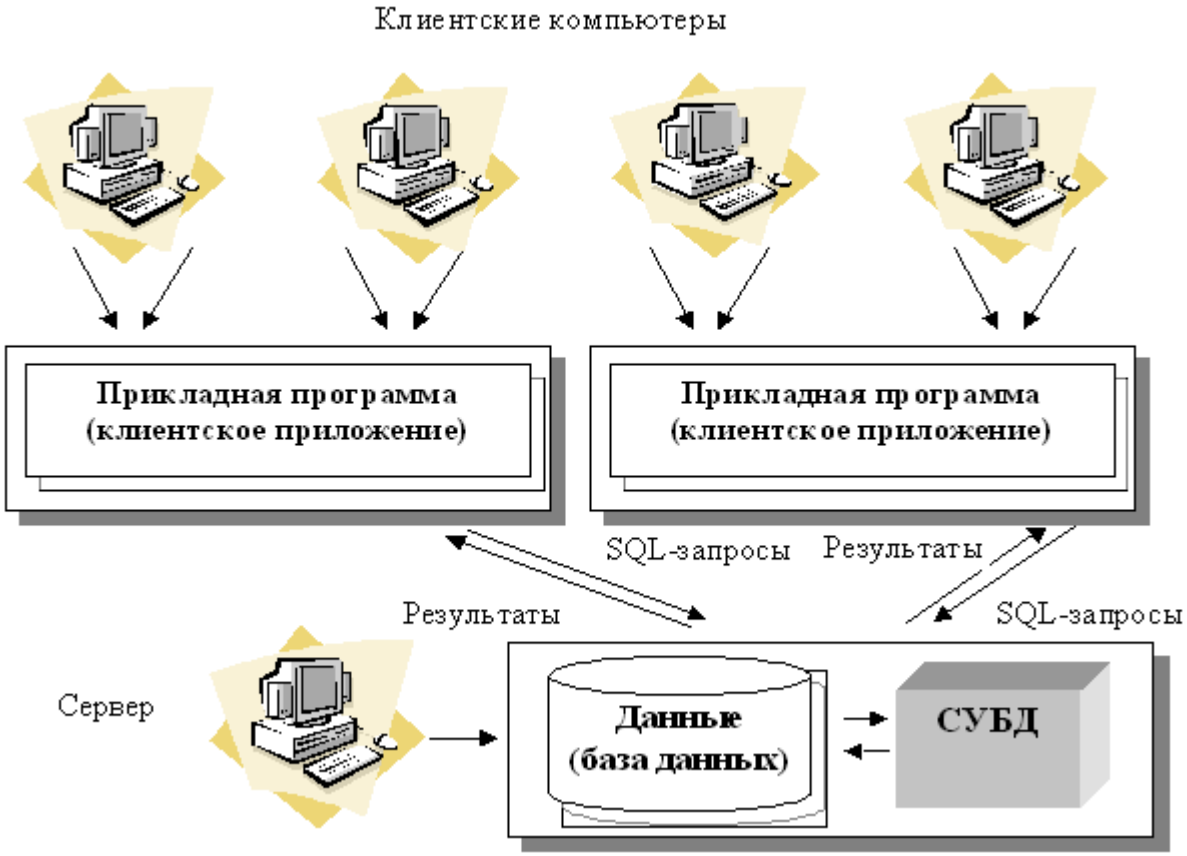

Рисунок 2 Архитектура «клиент – сервер»

Итак, в результате работа построена следующим образом:

База данных в виде набора файлов находится на жестком диске специально выделенного компьютера (сервера сети).

СУБД располагается также на сервере сети.

Существует локальная сеть, состоящая из клиентских компьютеров, на каждом из которых установлено клиентское приложение для работы с БД.

На каждом из клиентских компьютеров пользователи имеют возможность запустить приложение. Используя предоставляемый приложением пользовательский интерфейс, он инициирует обращение к СУБД, расположенной на сервере, на выборку/обновление информации. Для общения используется специальный язык запросов SQL, т.е. по сети от клиента к серверу передается лишь текст запроса.

СУБД инкапсулирует внутри себя все сведения о физической структуре БД, расположенной на сервере.

СУБД инициирует обращения к данным, находящимся на сервере, в результате которых на сервере осуществляется вся обработка данных и лишь результат выполнения запроса копируется на клиентский компьютер. Таким образом СУБД возвращает результат в приложение.

Приложение, используя пользовательский интерфейс, отображает результат выполнения запросов[15].

Рассмотрим, как выглядит разграничение функций между сервером и клиентом.

Функции приложения-клиента:

- Посылка запросов серверу.
- Интерпретация результатов запросов, полученных от сервера.
- Представление результатов пользователю в некоторой форме (интерфейс пользователя).

Функции серверной части:

- Прием запросов от приложений-клиентов.

- Интерпретация запросов.
- Оптимизация и выполнение запросов к БД.
- Отправка результатов приложению-клиенту.
- Обеспечение системы безопасности и разграничение доступа.
- Управление целостностью БД.
- Реализация стабильности многопользовательского режима работы[16].

В архитектуре «клиент – сервер» работают так называемые «промышленные» СУБД. Промышленными они называются из-за того, что именно СУБД этого класса могут обеспечить работу информационных систем масштаба среднего и крупного предприятия, организации, банка. К разряду промышленных СУБД принадлежат MS SQL Server, Oracle, Gupta, Informix, Sybase, DB2, InterBase и ряд других.

Как правило, SQL-сервер обслуживается отдельным сотрудником или группой сотрудников (администраторы SQL-сервера). Они управляют физическими характеристиками баз данных, производят оптимизацию, настройку и переопределение различных компонентов БД, создают новые БД, изменяют существующие и т.д., а также выдают привилегии (разрешения на доступ определенного уровня к конкретным БД, SQL-серверу) различным пользователям.

Рассмотрим основные достоинства данной архитектуры по сравнению с архитектурой «файл-сервер»:

1. Существенно уменьшается сетевой трафик.
2. Уменьшается сложность клиентских приложений (большая часть нагрузки ложится на серверную часть), а, следовательно, снижаются требования к аппаратным мощностям клиентских компьютеров.
3. Наличие специального программного средства – SQL-сервера – приводит к тому, что существенная часть проектных и программистских задач становится уже решенной.
4. Существенно повышается целостность и безопасность БД[17].

К числу недостатков можно отнести более высокие финансовые затраты на аппаратное и программное обеспечение, а также то, что большое количество клиентских компьютеров, расположенных в разных местах, вызывает определенные трудности со своевременным обновлением клиентских приложений на всех компьютерах-клиентах. Тем не менее, архитектура « клиент – сервер « хорошо зарекомендовала себя на практике, в настоящий момент существует и функционирует большое количество БД, построенных в соответствии с данной

архитектурой[18].

Технология «клиент-сервер»

2.1 Классическая двухуровневая архитектура «клиент-сервер»

Как правило, части сети не являются равноправными полностью: некоторые обладают доступом к различным ресурсам (например – сканер, матрица, СУБД и др.), другие же обладают возможностью обращения к таким ресурсам.

Технология «клиент-сервер» представляет собой архитектуру программного комплекса, внутри которой осуществляется распределение прикладной программы по двум логически различным компонентам (клиент и сервер), которые взаимодействуют согласно схеме «запрос-ответ» и решают некоторые задачи (рисунок 3)[19].

Рисунок 3 Архитектура «Клиент – сервер»

Компьютер или программный продукт, который управляет и владеет некоторым ресурсом, называется сервером этого некоторого ресурса.

Клиентом называют такой компьютер или программу, который запрашивает и использует некоторый ресурс.

Сервер и клиент могут располагаться как на одном компьютере, так и на разных компьютерах, соединенных сетью. Кроме того, возможно возникновение такой ситуации, когда некая программа или их блок будут одновременно реализовывать функции сервера по отношению по отношению к одному блоку и функции клиента по отношению к другому.

Основной принцип технологии «Клиент-сервер» заключается в разделении функций приложения как минимум на три группы:

- модули интерфейса с пользователем;

Группа модулей интерфейса с пользователем также может быть названа логикой представления. Посредством взаимодействия с логикой представления пользователи могут успешно работать с приложением. Вне зависимости от определенных характеристик модулей интерфейса с пользователем, их задача состоит в обеспечении средств для максимально эффективного и оперативного обмена данными между пользователем и системой.

- модули хранения данных;

Группу модулей хранения данных также можно назвать бизнес-логикой. Модули хранения данных определяют конкретное предназначение приложения. Разделение приложения по границам между программами обеспечивает естественную основу для распределения приложения на нескольких компьютерах.

- модули обработки данных (функции управления ресурсами);

Данная группа также именуется логикой доступа к данным или алгоритмами доступа к данным. Алгоритмы доступа к данным исторически рассматривались как специфический для конкретного приложения интерфейс к механизму постоянного хранения данных наподобие файловой системы или СУБД. При помощи модулей обработки данных организуется специфический для приложения интерфейс к СУБД. При помощи интерфейса приложение управляет соединениями с базой данных и запросами к ней (перевод специфических для конкретного приложения запросов на язык SQL, получение результатов и перевод этих результатов обратно в специфические для конкретного приложения структуры данных)[20].

Каждая из перечисленных групп может быть претворена в жизнь вне зависимости от остальных. Так, не осуществляя изменения программ, применяемых для хранения и обработки данных, можно реализовать изменения интерфейса с пользователем таким образом, что одна и та же информация будет отображаться в виде таблиц, графиков или гистограмм. Очень простые приложения часто способны собрать все три части в единственную программу, и подобное разделение соответствует функциональным границам.

В соответствии с разделением функций в любом приложении можно выделить следующие компоненты:

- компонент представления данных;
- прикладной компонент;
- компонент управления ресурсом[21].

В классической архитектуре клиент-сервер приходится распределять три основные части приложения по двум физическим модулям. Обычно прикладной компонент располагается на сервере (например, сервере базы данных), компонент представления данных - на стороне клиента, а компонент управления ресурсом распределяется между клиентской и серверной частями. В этом заключается основной недостаток классической двухуровневой архитектуры.

В двухзвенной архитектуре при разбиении алгоритмов обработки данных разработчики должны иметь полную информацию о последних изменениях, внесенных в систему, и понимать эти изменения, что создает большие сложности при разработке клиент-серверных систем, их установке и сопровождении, поскольку необходимо тратить значительные усилия на координацию действий разных групп специалистов. В действиях разработчиков часто возникают противоречия, а это тормозит развитие системы и вынуждает изменять уже готовые и проверенные элементы.

Чтобы избежать несогласованности различных элементов архитектуры были созданы две модификации двухзвенной архитектуры «Клиент – сервер»: «Толстый клиент» («Тонкий сервер») и «Тонкий клиент» («Толстый сервер»).

В данных архитектурах разработчики попытались выполнять обработку данных на одной из двух физических частей - либо на стороне клиента («Толстый клиент»), либо на сервере («Тонкий клиент).

Каждый подход имеет свои недостатки. В первом случае неоправданно перегружается сеть, потому что по ней передаются необработанные, а значит, избыточные данные. Кроме того, усложняется поддержка системы и ее изменение, так как замена алгоритма вычислений или исправление ошибки требует одновременной полной замены всех интерфейсных программ, а иначе могут возникнуть ошибки или несогласованность данных. Если же вся обработка информации выполняется на сервере, то возникает проблема описания встроенных процедур и их отладки. Систему с обработкой информации на сервере абсолютно невозможно перенести на другую платформу (ОС), что является серьезным

недостатком[22].

Есди все-таки разрабатывается двухуровневая классическая архитектура «Клиент – сервер», то необходимо помнить, что архитектура «Толстый сервер» аналогична архитектуре «Тонкий клиент» (Рисунок 4).

Рисунок 4 Архитектура «Тонкий клиент»

Передача запроса от клиента на сервер, обработка запроса сервером и передача результата клиенту. При этом архитектуры имеют следующие недостатки:

- усложняется реализация, так как языки типа SQL не приспособлены для разработки подобного ПО и нет хороших средств отладки;
- производительность программ, написанных на языках типа SQL, значительно ниже, чем созданных на других языках, что имеет важное значение для сложных систем;
- программы, написанные на СУБД-языках, обычно работают недостаточно надежно; ошибка в них может привести к выходу из строя всего сервера баз данных;
- получившиеся таким образом программы полностью непереносимы на другие системы и платформы.
- архитектура «Тонкий сервер» аналогична архитектуре «Толстый клиент» (Рисунок 5)[23].

Обработка запроса происходит на стороне клиента, то есть происходит передача клиенту всех необработанных данных с сервера. При этом архитектуры имеют следующие недостатки:

- усложняется обновление ПО, поскольку его замену нужно производить одновременно по всей системе;
- усложняется распределение полномочий, так как разграничение доступа происходит не по действиям, а по таблицам;
- перегружается сеть вследствие передачи по ней необработанных данных;
- слабая защита данных, поскольку сложно правильно распределить полномочия[24].

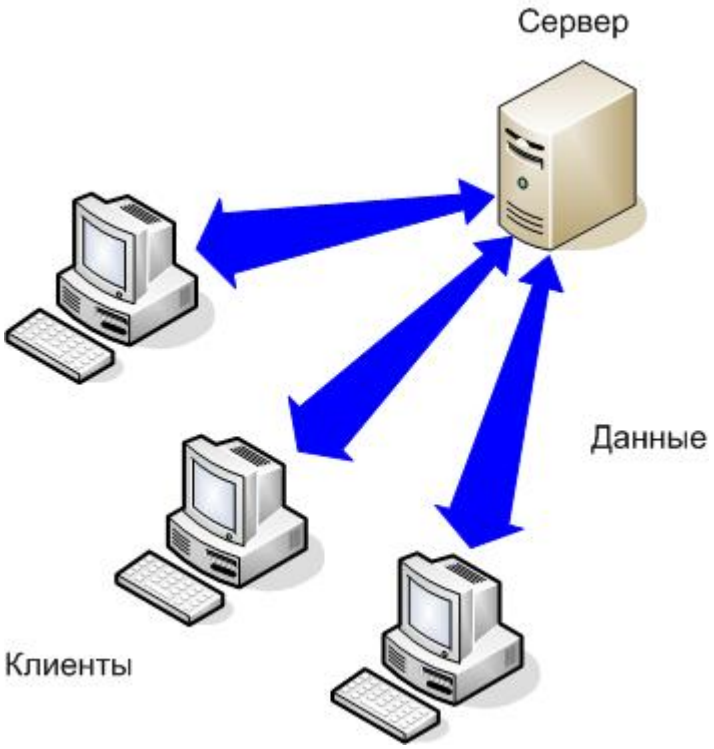

Рисунок 5 Архитектура «Толстый клиент»

Трехуровневая архитектура «клиент-сервер»

С середины 90-х годов прошлого века признание специалистов получила трехзвенная архитектура «Клиент – сервер», которая разделила информационную систему по функциональным возможностям на три отдельных компонента: логика представления, бизнес-логика и логика доступа к данным. В отличие от двухзвенной архитектуры в трехзвенной появляется дополнительное звено – сервер приложений, который предназначен для осуществления бизнес-логики, при этом полностью разгружается клиент, который направляет запросы

промежуточному программному обеспечению, и максимально используются все возможности серверов[25].

В технологии трехуровневой архитектуры, как правило, клиент не перегружается функциями обработки информации, а только реализует свою основную функцию – является системой представления данных, поступающих с сервера приложений. Подобный интерфейс может быть реализован при помощи стандартных методов и средств Web-технологии, таких как браузер, CGI и Java. Такой подход позволяет уменьшить объем данных, которые передаются от сервера к клиенту и обратно, а это, в свою очередь, предоставляет возможность подключения клиентских компьютеров даже по медленным линиям (например, по телефонным каналам). Помимо всего прочего, клиентская часть может быть так просто организована, что в большинстве случаев возможна ее организация при помощи самого простого браузера. Это также предоставляет возможность быстрого и безболезненного внесения всех необходимых изменений в клиентскую часть, если это станет необходимо.

Сервер приложений – это программное обеспечение, которое является промежуточным слоем между клиентом и сервером (Рисунок 6)[26].

Рисунок 6 Сервер приложений

Существует несколько категорий продуктов промежуточного слоя:

- Message orientated – яркие представители MQseries и JMS;
- Object Broker – яркие представители CORBA и DCOM;
- Component based – яркие представители.NET и EJB.

Использование сервера приложений дает больше возможностей, например, уменьшается нагрузка на клиентские компьютеры, потому что сервер приложений распределяет нагрузку и обеспечивает защиту от сбоев. Так как бизнес-логика

хранится на сервере приложений, то при каких-либо изменениях в отчетности или расчетах клиентские программы никоим образом не затрагиваются[27].

Существует несколько серверов приложений от таких знаменитых компаний как Sun Microsystem, Borland, IBM, Oracle и каждый из них отличается набором предоставляемых сервисов (производительность в данном случае учитывать не будем). Эти сервисы облегчают программирование и развертывание приложений масштаба предприятия. Обычно сервер приложений предоставляет следующие сервисы:

- WEB Server – чаще всего включают в поставку самый популярный и мощный Apache;
- WEB Container – позволяет выполнять JSP и сервлеты. Для Apache таким сервисом является Tomcat;
- CORBA Agent – может предоставлять распределенную директорию для хранения CORBA объектов;
- Messaging Service – брокер сообщений;
- Transaction Service – уже из названия понятно, что это сервис транзакций;
- JDBC – драйвера для подключения к базам данных, ведь именно серверу приложений придется общаться с базами данных и ему нужно уметь подключаться к используемой в вашей компании базе;
- Java Mail – данный сервис может предоставлять сервис к SMTP;
- JMS (Java Messaging Service) – обработка синхронных и асинхронных сообщений;
- RMI (Remote Method Invocation) - вызов удаленных процедур[28].

Многоуровневые клиент-серверные системы достаточно легко можно перевести на Web-технологию - для этого достаточно заменить клиентскую часть универсальным или специализированным браузером, а сервер приложений дополнить Web-сервером и небольшими программами вызова процедур сервера. Для разработки этих программ можно использовать как Common Gateway Interface (CGI), так и более современную технологию Java.

В трехуровневой системе в качестве каналов связи между сервером приложений и СУБД можно использовать более скоростные линии, которые потребуют минимальных затрат, так как сервера обычно находятся в одном помещении (серверной) и не будет перегружать сеть из-за передачи большого количества информации.

Из всего вышесказанного можно сделать вывод, что двухуровневая архитектура сильно уступает многоуровневой архитектуре, поэтому в настоящее время используется только многоуровневая архитектура «Клиент – сервер», в которой различают три модификации - RDA, DBS и AS[29].

2.3 Программное обеспечение технологии «клиент-сервер»

Типичная архитектура «клиент-сервер» требует того, чтобы компьютеры-серверы были мощнее и производительнее компьютеров-клиентов. Это связано с большой нагрузкой на серверы, и с тем, что на них лежит обязанность решения более серьезных задач, например, администрирования, выполнения большого количества одновременных запросов, обеспечения защиты данных и т.п.

Для эффективного применения технологии «клиент-сервер» требуется использование соответствующего программного обеспечения, которое, в свою очередь, должно включать в себя серверную и клиентскую части.

Программное обеспечение, которое устанавливается на сервере с целью управления базой данных, при реагировании на запросы клиентов, осуществляет поиск данных. Будучи частью архитектуры «клиент-сервер», соответствующее программное обеспечение возвращает результаты поиска по запросу.

Обработка информации на сервере представляет собой совокупность таких операций, как сортировка, извлечение и отправка пользователю необходимой информации.

В первую очередь программное обеспечение сервера обеспечивает такие действия над информацией, как обновление, удаление, добавление, защита и так далее[30].

Кроме того, на сервере размещаются и хранятся процедуры, которые подлежат использованию посредством любого клиента. Процедуры, хранимые на сервере, обеспечивают содействие в обработке информации, посредством уменьшения длины кода и применяемого пространства на клиентском аппаратном обеспечении. Так, любая одна хранимая процедура может подлежать вызову со стороны множества клиентов. При этом есть очень удобная особенность – каждый раз включать код процедуры в программу нет необходимости.

Помимо некоторой обработки информации, хранимые процедуры также могут уменьшать сетевой трафик, что объясняется тем фактом, что единственное обращение клиента приводит к выполнению серии команд хранимой процедуры, каждая из которых потребовала бы отдельного запроса, и могут проводить контроль безопасности, чтобы предотвратить несанкционированный запуск пользователями некоторых процедур.

Инструментальные средства, приложения и утилиты для интерфейсной части дополняют возможности модели «Клиент-сервер». К ним относятся средства запросов, которые упрощают доступ к данным сервера, используя предопределенные запросы и встроенные возможности для построения отчетов, пользовательские приложения, которые могут работать в качестве интерфейсной части, предоставляя доступ к серверу базы данных. Другие приложения (такие, как Microsoft Access) имеют свой собственный SQL, который обеспечивает доступ к системам управления базами данных от разных производителей. Для реализации систем «Клиент-сервер» необходимы специально разработанные интерфейсные части. Средства разработки программ (например, Microsoft Visual Basic) значительно облегчают программистам и администраторам информационных систем создание приложений, которые отвечают за доступ к серверам базы данных [31].

Выбор подходящего программного обеспечения зависит от выбора операционной системы и задач, которые необходимо выполнить. Например, в случае, когда применяется операционная система Windows, то на компьютере – клиенте в подавляющем большинстве случаев будет применяться пакет продуктов Microsoft Office, наделенный большим количеством разнообразных прикладных продуктов.

В связи с успехом распространения пакета Microsoft Office корпорация Microsoft решила собрать комплекс программ для сервера –пакет MS BackOffice. В состав названного пакета входят Windows Server – сетевая операционная система, System Management Server – система администрирования сети, SQL Server – сервер управления базами данных, SNA Server – сервер для соединения с хост-компьютерами, Exchange Server – сервер системы электронной почты и Internet Information Server – сервер для работы с Internet.

Windows Server 2000/2003/2008 способна обеспечить совместное использование файлов, печатающих устройств, предоставить услуги по соединению с рабочими станциями (клиентскими компьютерами) и другой сервис.

В качестве сетевой операционной системы используют Windows 2000/2003/2008 Server, которую можно использоваться и на рабочей станции для реализации дополнительных возможностей.

Windows Server 2000/2003/2008 обеспечивает совместное использование не только множества процессов, но и ресурсов многими пользователями. Возможность соединения с удаленными сетями реализуется через сервис удаленного доступа – RAS (Remote Access Service), а также через средства связи с сетями других фирм (Novell, Digital Pathworks и Apple)[32].

System Management Server (SMS) позволяет сетевому администратору централизованно управлять всей сетью. При этом обеспечивается возможность администрирования каждого компьютера, подключенного к сети, включая установленное на нем программное обеспечение. SMS предоставляет различные виды сервиса, например управление сетевыми приложениями и инвентаризацией программного и аппаратного обеспечения. SMS включает в себя автоматизацию установки и распространения программного обеспечения, включая его обновление, удаленное устранение неисправностей и предоставление полного контроля администратору за устройствами ввода и экранами всех компьютеров в сети.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы была достигнута поставленная цель, а именно – изучена технология «клиент-сервер».

Для достижения данной цели были выполнены следующие задачи:

1. охарактеризована архитектура информационной системы;
2. рассмотрена архитектура «файл-сервер»;
3. кратко охарактеризована архитектура «клиент-сервер»;
4. изучена классическая двухуровневая архитектура «клиент-сервер»;
5. проанализирована трехуровневая архитектура «клиент-сервер»;
6. произведен анализ программного обеспечения технологии «клиент-сервер».

Кроме того, при работе над исследованием было обнаружено, что, действительно, в настоящее время каждый пользователь компьютера или сети Интернет вступает во взаимодействие с технологией «клиент-сервер». Просматривая веб-страницы, работая в офисной программе или редактируя фотографии.

На настоящий момент данную тему нельзя назвать полностью изученной и изжившей себя. Это легко подтверждается непрерывным процессом совершенствования информационных и аппаратных технологий, огромным количеством открытий новых средств и методов обработки информации и работы с компьютером.

Помимо всего прочего, при выполнении работы были значительно расширены теоретические знания по проблеме, что может оказаться полезным, как в повседневной жизни, так и при дальнейшем изучении дисциплины и дисциплин, смежных с ней.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Брюхов Д., Задорожный В., Калиниченко Л. и др. Интероперабельные информационные системы: архитектуры и технологии. - СУБД, 4, 1995.
2. Вескес Л.Дж. Access и SQL Server. Руководство разработчика /Дж.Л. Вескес - Москва: Лори, 1997.
3. Волков В.Б. Понятный самоучитель работы в Windows XP, СПб, Питер, 2004
4. Вудворд Дж. «Технология совместной работы», электронная версия
5. Дейт К.Дж. Введение в системы баз данных /К.Дж. Дейт - Москва: ДМК, 2000.
6. Дрога А. А., Жукова П. Н., Копонев Д. Н., Лукьянов Д. Б., Прокопенко А. Н. Информатика и математика. - Минск, 2008.
7. Калиниченко Л.. Стандарт систем управления объектными базами данных ODMG 93: краткий обзор и оценка состояния. - СУБД, 1, 1996.
8. Конноли Т. Базы данных. Проектирование, реализация и сопровождение /Т. Конноли, К. Бегг. - Москва: Вильяме, 2003.
9. Кузнецов С. Д. Основы баз данных. - 1-е изд. - М.: «Интернет- университет информационных технологий - ИНТУИТ.ру», 2005.
10. Мельников В. Защита информации в компьютерных системах. - М.: Финансы и статистика, Электронинформ, 1997
11. Морозевич А.Н., Зеневич А.М. Информатика. Минск, 2008.
12. Скотт В. Эмблер, Прамодкумар Дж. Садаладж. Рефакторинг баз данных: эволюционное проектирование - М.: «Вильямс», 2007
13. Тейлор А.Дж. SQL для «чайников» /А.Дж. Тейлор.- Москва: Вильяме, 2005.
14. Титоренко Г.А. Информационные технологии управления. М., Юнити: 2002.
15. Хомоненко А.Д. Базы данных /А.Д. Хомоненко, В.М. Цыганков - Санкт-Петербург: БХВ-Петербург, 2004.

1. Конноли Т. Базы данных. Проектирование, реализация и сопровождение /Т. Конноли, К. Бегг. - Москва: Вильяме, 2003, 123 с. ↑

2. Кузнецов С. Д. Основы баз данных. - 1-е изд. - М.: «Интернет- университет информационных технологий - ИНТУИТ.ру», 2005 ↑

3. Морозевич А.Н., Зеневич А.М. Информатика. Минск, 2008, 304 с. ↑

4. Титоренко Г.А. Информационные технологии управления. М., Юнити: 2002, 287 с. ↑

5. Хомоненко А.Д. Базы данных /А.Д. Хомоненко, В.М. Цыганков - Санкт-Петербург: БХВ-Петербург, 2004, 115 с. ↑

6. Хомоненко А.Д. Базы данных /А.Д. Хомоненко, В.М. Цыганков - Санкт-Петербург: БХВ-Петербург, 2004, 121 с. ↑

7. Морозевич А.Н., Зеневич А.М. Информатика. Минск, 2008, 165 с. ↑

8. Кузнецов С. Д. Основы баз данных. - 1-е изд. - М.: «Интернет- университет информационных технологий - ИНТУИТ.ру», 2005 ↑

9. Кузнецов С. Д. Основы баз данных. - 1-е изд. - М.: «Интернет- университет информационных технологий - ИНТУИТ.ру», 2005 ↑

10. Дрога А. А., Жукова П. Н., Копонев Д. Н., Лукьянов Д. Б., Прокопенко А. Н. Информатика и математика. - Минск, 2008, 176 с. ↑

11. Вудворд Дж. «Технология совместной работы», электронная версия ↑

12. Морозевич А.Н., Зеневич А.М. Информатика. Минск, 2008, 316 с. ↑

13. Скотт В. Эмблер, Прамодкумар Дж. Садаладж. Рефакторинг баз данных: эволюционное проектирование - М.: «Вильямс», 2007, 231 с. ↑

14. Скотт В. Эмблер, Прамодкумар Дж. Садаладж. Рефакторинг баз данных: эволюционное проектирование - М.: «Вильямс», 2007, 242 с. ↑

15. Брюхов Д., Задорожный В., Калиниченко Л. и др. Интероперабельные информационные системы: архитектуры и технологии. 1995, 198 с. ↑

16. Брюхов Д., Задорожный В., Калиниченко Л. и др. Интероперабельные информационные системы: архитектуры и технологии. 1995, 214 с. ↑

17. Вескес Л.Дж. Access и SQL Server. Руководство разработчика /Дж.Л. Вескес - Москва: Лори, 1997, 154 с. ↑

18. Вескес Л.Дж. Access и SQL Server. Руководство разработчика /Дж.Л. Вескес - Москва: Лори, 1997, 167 с. ↑

19. Дрога А. А., Жукова П. Н., Копонев Д. Н., Лукьянов Д. Б., Прокопенко А. Н. Информатика и математика. - Минск, 2008, 401 с. ↑

20. Дрога А. А., Жукова П. Н., Копонев Д. Н., Лукьянов Д. Б., Прокопенко А. Н. Информатика и математика. - Минск, 2008, 384 с ↑

21. Дрога А. А., Жукова П. Н., Копонев Д. Н., Лукьянов Д. Б., Прокопенко А. Н. Информатика и математика. - Минск, 2008, 391 с ↑

22. Скотт В. Эмблер, Прамодкумар Дж. Садаладж. Рефакторинг баз данных: эволюционное проектирование - М.: «Вильямс», 2007, 231 с. ↑

23. Хомоненко А.Д. Базы данных /А.Д. Хомоненко, В.М. Цыганков - Санкт-Петербург: БХВ-Петербург, 2004, 281 с. ↑

24. Хомоненко А.Д. Базы данных /А.Д. Хомоненко, В.М. Цыганков - Санкт-Петербург: БХВ-Петербург, 2004, 293 с. ↑

25. Скотт В. Эмблер, Прамодкумар Дж. Садаладж. Рефакторинг баз данных: эволюционное проектирование - М.: «Вильямс», 2007, 324 с. ↑

26. Титоренко Г.А. Информационные технологии управления. М., Юнити: 2002, 409 с. ↑

27. Титоренко Г.А. Информационные технологии управления. М., Юнити: 2002, 404 с ↑

28. Титоренко Г.А. Информационные технологии управления. М., Юнити: 2002, 411 с. ↑

29. Кузнецов С. Д. Основы баз данных. - 1-е изд. - М.: «Интернет- университет информационных технологий - ИНТУИТ.ру», 2005, 261 с. ↑

30. Конноли Т. Базы данных. Проектирование, реализация и сопровождение /Т. Конноли, К. Бегг. - Москва: Вильяме, 2003, 167 с. ↑

31. Вескес Л.Дж. Access и SQL Server. Руководство разработчика /Дж.Л. Вескес - Москва: Лори, 1997, 328 с. ↑

32. Волков В.Б. Понятный самоучитель работы в Windows XP, СПб, Питер, 2004, 395 с. ↑