**Проектирование корпоративных информационных систем и управление**

Выполнил: Терин В.А. 03-432

Московский авиационный институт (Государственный технический университет)

Москва, 2009 г

**Общее представление о ИС**

1.Специфика информационных программных систем

В зависимости от конкретной области применения информационные системы могут очень сильно различаться по своим функциям, архитектуре, реализации. Однако можно выделить, по крайней мере, два свойства, которые являются общими для всех информационных систем. Во-первых, любая информационная система предназначена для сбора, хранения и обработки информации.

Во-вторых, информационные системы ориентируются на конечного пользователя, например, банковского клерка. Такие пользователи могут быть очень далеки от мира компьютеров. Для них терминал, персональный компьютер или рабочая станция представляют собой всего лишь орудие их собственной профессиональной деятельности. Поэтому информационная система обязана обладать простым, удобным, легко осваиваемым интерфейсом, который должен предоставить конечному пользователю все необходимые для его работы функции, но в то же время не дать ему возможность выполнять какие-либо лишние действия.

Вычислительные программные системы не обязательно обладают развитыми интерфейсами. Конечно, если система предназначена для продажи, то она должна обладать хорошим интерфейсом хотя бы в целях маркетинга. Но как правило, серьезные вычислительные программы почти уникальны. Расчеты выполняются либо разработчиками программ, либо людьми из того же окружения. Для них гораздо важнее быстродействие вычислений, чем удобство запуска программы, а наличие развитого интерфейса предполагает существенный расход компьютерных ресурсов. Как профессионалы компьютерного мира, эти люди могут справиться с некоторыми неудобствами при работе с компьютером.

2.Задачи информационных систем

Конкретные задачи, которые должны решаться информационной системой, зависят от той прикладной области, для которой предназначена система. Области применения информационных приложений разнообразны: банковское дело, страхование, медицина, транспорт, образование и т.д. Трудно найти область деловой активности, в которой сегодня можно было обойтись без использования информационных систем. С другой стороны,, конкретные задачи, решаемые банковскими информационными системами, отличаются от задач, решение которых требуется от медицинских информационных систем.

Но можно выделить некоторое количество задач, не зависящих от специфики прикладной области. Прежде всего, наиболее существенной составляющей является информация, которая долго накапливается и утрата которой невосполнима.

Следующей задачей, которую должно выполнять большинство информационных систем, - это хранение данных, обладающих разными структурами. Трудно представить себе более или менее развитую информационную систему, которая работает с одним однородным файлом данных. Разумным требованием к информационной системе является то, чтобы она могла развиваться. Могут появиться новые функции, для выполнения которых требуются дополнительные данные с новой структурой. При этом вся накопленная ранее информация должна остаться сохранной.

В корпоративных информационных системах по естественным причинам часто возникает потребность в распределенном хранении общей базы данных. Разумно например хранить некоторую часть информации как можно ближе к тем рабочим местам, в которых она чаще всего используется. По этой причине при построении информационной системы приходится решать задачу согласованного управления распределенной базой данных (иногда применяя методы репликации данных). При однородном построении распределенной базы данных (на основе однотипных серверов баз данных) эту задачу обычно удается решить на уровне СУБД (большинство производителей развитых СУБД поддерживает средства управления распределенными базами данных). Если же система разнородна (т.е. для управления отдельными частями распределенной базы данных используются разные серверы), то приходится прибегать к использованию вспомогательных инструментальных средств интеграции разнородных баз данных типа мониторов транзакций.

3.Проблемы построения ИС

Самой первой проблемой является проблема проектирования. Нельзя начинать техническую разработку, не имея тщательно проработанного проекта. Если начинать с решения наиболее очевидных задач, не обращая внимания на потенциально существующие, то такая система будет непрерывно находиться в стадии разработки и переделки.

Первой стадией проектирования должен быть анализ требований корпорации. Для этого на основе экспертных запросов необходимо выявить все актуальные и потенциальные потребности корпорации, которые должны удовлетворяться проектируемой информационной системой, понять какие потоки данных существуют внутри корпорации, оценить объемы информации, которые должны поддерживаться и обрабатываться информационной системой.

Следующая стадия проектирования - выработка концептуальной схемы базы данных, которая будет лежать в основе информационной системы.

Далее, с большой вероятностью в основе информационной системы будет лежать реляционная база данных, поэтому на следующей стадии проектирования понадобится на основе имеющейся концептуальной схемы произвести набор определений схемы реляционной базы данных в терминах языка SQL. На этой же стадии необходимо решить, какие таблицы будут реально хранимыми, а какие - представляемыми (view).

После того, как выработана общая реляционная схема базы данных, необходимо определиться с архитектурой системы. В частности, очень важно решить, какой будет база данных - централизованной или распределенной.

Следующая стадия проектирования состоит в дополнении реляционных схем разделов распределенной базы данных определениями общих ограничений целостности, триггеров и хранимых процедур.

Логическое проектирование базы данных информационной системы закончено и осталось две стадии: физическое проектирование базы данных; проектирование и разработка интерфейсов и обрабатывающей части прикладной системы. Эти две стадии могут выполняться параллельно.

Физическое проектирование включает два основных шага, первый из которых, как правило, не зависит от особенностей выбранного серверного SQL-ориентированного продукта, а второй зависит. На первом шаге этой стадии определяется набор требуемых индексов. Второй шаг состоит в определении областей внешней памяти, в которых будут храниться фрагменты базы данных.

Параллельно с физическим проектированием базы данных информационной системы может проводиться проектирование и разработка интерфейса системы и ее обрабатывающей части. В принципе, к этому моменту уже должно быть ясно, что нужно от интерфейса и какие функции должна выполнять система. Так что основной проблемой этой стадии является выбор инструментальных средств, которые позволят достаточно быстро произвести достаточно эффективную реализацию.

4.Требования к техническим средствам, поддерживающим ИС

Требования к техническим средствам определяются требованиями к информационной системе в целом. Какие бы информационные возможности не требовались сотрудникам корпорации, окончательное решение всегда принимается ее руководством, которое корректирует требования к информационной системе и формирует окончательное представление об аппаратной среде.

Выбор технических средств для построения корпоративной информационной системы - это непростая задача, включающая технические, политические и эмоциональные аспекты.

Общая классификация архитектур информационных приложений

Проектирование и разработка информационной системы может базироваться на разных архитектурных решениях.

Начать можно с традиционных архитектурных решений, основанных на использовании выделенных файл-серверов или серверов баз данных. Затем рассматриваются варианты архитектур корпоративных информационных систем, базирующихся на технологии Internet (Intranet-приложения). Следующая разновидность архитектуры информационной системы основывается на концепции "склада данных" (DataWarehouse) - интегрированной информационной среды, включающей разнородные информационные ресурсы. Наконец, последняя архитектура предназначена для построения глобальных распределенных информационных приложений с интеграцией информационно-вычислительных компонентов на основе объектно-ориентированного подхода.

1.Файл-серверные приложения

Организация информационных систем на основе использования выделенных файл-серверов все еще является наиболее распространенной в связи с наличием большого количества персональных компьютеров разного уровня развитости и сравнительной дешевизны связывания PC в локальные сети. Фактически, компоненты информационной системы, выполняемые на разных PC, взаимодействуют только за счет наличия общего хранилища файлов, которое хранится на файл-сервере. В классическом случае в каждой PC дублируются не только прикладные программы, но и средства управления базами данных. Файл-сервер представляет собой разделяемое всеми PC комплекса расширение дисковой памяти.

Основным достоинством 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 NT. Имеются удобные и развитые средства разработки графического пользовательского интерфейса, простые в использовании средства разработки систем баз данных и/или СУБД. Но во многом эта простота является кажущейся.

Во-первых, информационной системе предстоит работать с базой данных. Следовательно, эта база данных должна быть спроектирована. Часто разработчики файл-серверных приложений считают, что по причине простоты средств управления базами данных проблемой проектирования базы данных можно пренебречь. Конечно, это неправильно. Чем качественнее она спроектирована, тем больше шансов впоследствии эффективно использовать информационную систему. Сложность проектирования базы данных определяется объективной сложностью моделируемой предметной области.

Во-вторых необходимыми требованиями к базе данных информационной системы являются поддержание ее целостного состояния и гарантированная надежность хранения информации. Минимальными условиями, при соблюдении которых можно удовлетворить эти требования, являются:

наличие транзакционного управления,

хранение избыточных данных (например, с применением методов журнализации),

возможность формулировать ограничения целостности и проверять их соблюдение.

В третьих, интерфейс развитых серверов баз данных основан на использовании высокоуровневого языка баз данных SQL, что позволяет использовать сетевой трафик между клиентом и сервером баз данных только в полезных целях (от клиента к серверу в основном пересылаются операторы языка SQL, от сервера к клиенту - результаты выполнения операторов). В файл-серверной организации клиент работает с удаленными файлами, что вызывает существенную перегрузку трафика (поскольку СУБД-ФС работает на стороне клиента, то для выборки полезных данных в общем случае необходимо просмотреть на стороне клиента весь соответствующий файл целиком).

Краткие выводы. Простое, работающее с небольшими объемами информации и рассчитанное на применение в однопользовательском режиме, файл-серверное приложение можно спроектировать, разработать и отладить очень быстро. Очень часто для небольшой компании для ведения, например, кадрового учета достаточно иметь изолированную систему, работающую на отдельно стоящем PC. Конечно, и в этом случае требуется большая аккуратность конечных пользователей (или администраторов, наличие которых в этом случае сомнительно) для надежного хранения и поддержания целостного состояния данных. Однако, в уже ненамного более сложных случаях (например, при организации информационной системы поддержки проекта, выполняемого группой) файл-серверные архитектуры становятся недостаточными.

2.Клиент-серверные приложения

Под клиент-серверным приложением понимается информационная система, основанная на использовании серверов баз данных .

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.

И 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.

Архитектура "клиент-сервер" на первый взгляд кажется гораздо более дорогой, чем архитектура "файл-сервер". Требуется более мощная аппаратура (по крайней мере, для сервера) и существенно более развитые средства управления базами данных. Однако, это верно лишь частично. Громадным преимуществом клиент-серверной архитектуры является ее масштабируемость и вообще способность к развитию.

При проектировании информационной системы, основанной на этой архитектуре, 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. В идеале, которого, конечно же не бывает, информационная система продолжает нормально функционировать после смены аппаратуры.

3. Intranet-приложения

Возникновение и внедрение в широкую практику высокоуровневых служб Всемирной Сети Сетей Internet (e-mail, ftp, telnet, Gopher, WWW и т.д.) естественным образом повлияли на технологию создания корпоративных информационных систем, породив направление, известное теперь под названием Intranet. По сути дела, информационная Intranet-система - это корпоративная система, в которой используются методы и средства Internet. Такая система может быть локальной, изолированной от остального мира Internet, или опираться на виртуальную корпоративную подсеть Internet. В последнем случае особенно важны средства защиты информации от несанкционированного доступа.

Хотя в общем случае в Intranet-системе могут использоваться все возможные службы Internet, наибольшее внимание привлекает гипермедийная служба WWW (World Wide Web - Всемирная Паутина). Для этого имеются две основные причины. Во-первых, с использованием языка гипермедийной разметки документов HTML можно сравнительно просто разработать удобную для использования информационную структуру, которая в дальнейшем будет обслуживаться одним из готовых Web-серверов. Во-вторых, наличие нескольких готовых к использованию клиентских частей - браузеров, или "обходчиков" избавляет от необходимости создавать собственные интерфейсы с пользователями, предоставляя им удобные и развитые механизмы доступа к информации. В ряде случаев такая организация корпоративной информационной системы оказывается достаточной для удовлетворения потребностей компании.

Однако, при всех своих преимуществах (простота организации, удобство использования, стандартность интерфейсов и т.д.) эта схема обладает сильными ограничениями. Все, что может пользователь, это только просмотреть информацию, поддерживаемую Web-сервером. Далее, гипертекстовые структуры трудно модифицируются. Для того, чтобы изменить наполнение Web-сервера, необходимо приостановить работу системы, внести изменения в HTML-описания и только затем продолжить нормальное функционирование. Наконец, далеко не всегда достаточен поиск информации в стиле просмотра гипертекста. Базы данных и соответствующие средства выборки данных по-прежнему часто необходимы.

Все перечисленные трудности могут быть разрешены с использованием более развитых механизмов Web-технологии. Эти механизмы непрерывно совершенствуются, что одновременно и хорошо и плохо. Хорошо, потому что появляются новые возможности. Плохо, потому что отсутствует стандартизация. С другой стороны, если информационная система будет создана с использованием текущего уровня технологии и окажется удовлетворяющей потребностям корпорации, то никто не будет обязан что-либо менять в системе по причине появления более совершенных механизмов.

4 Склады данных (DataWarehousing) и системы оперативной аналитической обработки данных

До сих пор рассматривались способы и возможные архитектуры информационных систем, предназначенных для оперативной обработки данных, т.е. для получения текущей информации, позволяющей решать повседневные проблемы корпорации. Однако у аналитических отделов корпорации и у высшего звена управляющего состава имеются и другие задачи: проанализировав поведение корпорации на рынке с учетом сопутствующих внешних факторов и спрогнозировав хотя бы ближайшее будущее, выработать тактику, а возможно, и стратегию корпорации. Для решения таких задач требуются данные и прикладные программы, отличные от тех, которые используются в оперативных информационных системах. В последние несколько лет все более популярным становится подход, основанный на концепциях склада данных и системы оперативной аналитической обработки данных. Возможно, в российских условиях трудно производить долговременные прогнозы бизнес-деятельности (слишком изменчивы внешние факторы), но анализ прошлого и краткосрочные прогнозы будущего могут оказаться очень полезными.

Основным источником информации, поступающей в оперативную базу данных является деятельность корпорации. Для проведения анализа данных требуется привлечение внешних источников информации (например, статистических отчетов). Тем самым, склад данных должен включать как внутренние корпоративные данные, так и внешние данные, характеризующие рынок в целом.

Если для оперативной обработки, как правило, требуются свежие данные (обычно в оперативных базах данных информация сохраняется не более нескольких месяцев), то в складе данных нужно поддерживать хранение информации о деятельности корпорации и состоянии рынка на протяжении нескольких лет (для проведения достоверных анализа и прогнозирования). Как следствие, аналитические базы данных имеют объем как минимум на порядок больший, чем оперативные.

Во многих достаточно крупных корпорациях одновременно существуют несколько оперативных информационных систем с собственными базами данных. Оперативные базы данных могут содержать семантически эквивалентную информацию, представленную в разных форматах, с разным указанием времени ее поступления, иногда даже противоречивую (например, из-за ошибок ввода данных). Склад данных корпорации должен содержать единообразно представленные данные из всех оперативных баз данных. Эта информация должна максимально полно соответствовать текущему содержанию оперативных баз данных и быть согласованной. Отсюда следует необходимость наличия компонента склада данных, извлекающего информацию из оперативных баз данных и "очищающего" эту информацию.

В 1993 г. основоположник реляционного подхода к организации баз данных Эдвар Кодд, исходя из потребностей систем динамической аналитической обработки данных, сформулировал 12 основных требований к системам, поддерживающим аналитические базы данных.

Многомерное концептуальное представление данных.

Прозрачность.

Доступность

Согласованная эффективность производства отчетов

Архитектура "клиент-сервер".

Родовая многомерность

Управление динамическими разреженными матрицами

Поддержка многопользовательского режима

Неограниченные операции между измерениями

Интуитивное манипулирование данными

Гибкая система отчетов

Неограниченное число измерений и уровней агрегации

Несмотря на то, что подход склад данных еще достаточно молод, чтобы вокруг него сложился круг общепринятых понятий, терминов, технологических приемов, он кажется настолько важным и перспективным, что многие компании (в том числе и ведущие производители СУБД) ведут активную работу, чтобы быть в авангарде этого направления.

5 . Интегрированные распределенные приложения

Нет никаких проблем, если с самого начала информационное приложение проектируется и разрабатывается в духе подхода открытых систем: все компоненты являются мобильными и интероперабельными, общее функционирование системы не зависит от конкретного местоположения компонентов, система обладает хорошими возможностями сопровождаемости и развития. К сожалению, на практике этот идеал является трудно достижимым. По разным причинам возникают потребности в интеграции независимо и по-разному организованных информационно-вычислительных ресурсов. Поэтому теперь существует путь решения проблемы, который сам лежит в русле открытых систем, - подход, предложенный крупнейшим международным консорциумом OMG (Object Management Group).

Решение проблемы интеграции неоднородных информационных ресурсов началось с попыток интеграции неоднородных баз данных. Направление интегрированных или федеративных систем неоднородных БД и мульти-БД появилось в связи с необходимостью комплексирования систем БД, основанных на разных моделях данных и управляемых разными СУБД.

Основной задачей интеграции неоднородных БД является предоставление пользователям интегрированной системы глобальной схемы БД, представленной в некоторой модели данных, и автоматическое преобразование операторов манипулирования БД глобального уровня операторы, понятные соответствующим локальным СУБД.

Как правило, интегрировать приходится неоднородные БД, распределенные в вычислительной сети. Это в значительной степени усложняет реализацию. Дополнительно к собственным проблемам интеграции приходится решать все проблемы, присущие распределенным СУБД: управление глобальными транзакциями, сетевую оптимизацию запросов и т.д. Очень трудно добиться эффективности. Как правило, для внешнего представления интегрированных и мульти-БД используется (иногда расширенная) реляционная модель данных. В последнее время все чаще предлагается использовать объектно-ориентированные модели, но на практике пока основой является реляционная модель. Поэтому, в частности, включение в интегрированную систему локальной реляционной СУБД существенно проще и эффективнее, чем включение СУБД, основанной на другой модели данных. Основным недостатком систем интеграции неоднородных баз данных является то, что при этом не учитываются "поведенческие" аспекты компонентов прикладной системы. Легко заметить, что даже при наличии развитой интеграционной системы, большинство из указанных выше проблем не решается. Естественным развитием взглядов на информационные ресурсы является их представление в виде набора типизированных объектов, сочетающих возможности сохранения информации (своего состояния) и обработки этой информации (за счет наличия хорошо определенного множества методов, применимых к объекту). Наиболее существенный вклад в создание соответствующей технологии внес международный консорциум OMG, выпустивший ряд документов, в которых специфицируются архитектура и инструментальные средства поддержки распределенных информационных систем, интегрированных на основе общего объектно-ориентированного подхода.

Согласованная с архитектурой OMA прикладная информационная система представляется как совокупность классов и экземпляров объектов, которые взаимодействуют при поддержке брокера объектных заявок (ORB - Object Request Broker). ORB, общие средства (Common Facilities) и объектные службы (Object Services) относятся к категории промежуточного программного обеспечения (middleware) и должны поставляться вместе. Объектные службы представляют собой набор услуг (интерфейсов и объектов), которые обеспечивают выполнение базовых функций, требуемых для реализации прикладных объектов и объектов категории "общие средства" (например, специфицированы служба именования объектов, служба долговременного хранения объектов, служба управления транзакциями и т.д.). Общие средства содержат набор классов и экземпляров объектов, поддерживающих функции, полезные в разных прикладных областях (например, средства поддержки пользовательского интерфейса, средства управления информацией и т.д.).

В основе OMA лежит базовая объектная модель COM (Core Object Model), в которой специфицированы такие понятия, как объект, операция, тип, подтипизация, наследование, интерфейс. Определены также способы согласованного расширения COM в разных объектных службах.

Интерфейсы объекта-клиента и объекта-сервера должны быть определены на специальном языке IDL (Interface Definition Language), который очень напоминает компонент спецификации класса (без реализации) языка Си++. Обращения к ORB могут быть сгенерированы статически при компиляции спецификаций IDL или выполнены динамически с использованием специфицированного в документах OMG API брокера объектных заявок. Правила построения и использования ORB определены в документе OMG CORBA (Common Object Request Broker Architecture).

Средства и методологии проектирования,разработки и сопровождения файл-серверных приложений

Файл-серверная информационная система, это система, которая в основном базируется на персональных компьютерах, используя в качестве внешней поддержки один или несколько файловых серверов, обеспечивающих значительные возможности управления внешней памятью, но не обладающих "интеллектом", поддерживая в основном только управление файлами. Практически во всех файл-серверных средствах и методологиях имеется тенденция к переходу к технологии "клиент-сервер". Файл-серверные архитектуры являются в большой степени облегченными вариантами клиент-серверных архитектур , хотя во многих случаях предлагаемые решения являются достаточными для небольшого по объему класса информационных систем.

1. Традиционные средства и методологии разработки файл-серверных приложений

Хотя для разработки файл-серверных приложений имеется целый ряд инструментальных средств, отсутствуют общепринятые методологии. Когда методологии используются, то они те же, что в клиент-серверных приложениях. Обычно же файл-серверные приложения проектируются и разрабатываются "по месту" без использования каких-либо стандартных методов.

Системы программирования третьего поколения 3GL являются предшественниками современных инструментальных средств и могут использоваться для разработки информационных приложений при наличии соответствующих встроенных или библиотечных средств для реализации диалога и доступа к базам данных.

Системы программирования для персональных компьютеров прошли долгий путь развития. Можно выделить три четкие языковые линии, которые оказывали друг на друга большое влияние, взаимно обогащаясь - это Си, Паскаль и Бейсик.

Основные вехи на пути развития систем программирования:

Переход от одиночных утилит систем программирования к интегрированным диалоговым средам программирования (например, семейство Turbo-продуктов фирмы Borland);

Развитие инструментальных наборов, расширяющих возможности систем программирования, в частности, в области диалога (разного рода Tool Box);

Появление объектно-ориентированных диалектов языков Си и Паскаль;

Возникновение операционной среды Windows со встроенной поддержкой диалога и первых Windows-приложений с помощью SDK (Software Development Keet);

Создание объектно-ориентированных библиотек, поддерживающих диалоговый режим работы в среде DOS и Windows (TurboVision, Object Windows и MFC);

Появление систем программирования, облегчающих создание приложений для DOS и Windows;

Развитие механизма встраивания и связывания объектов OLE 2;

Переход к визуальным системам программирования (Visual Си++, Delphi, Visual Basic), которые ориентированы на разработку информационных приложений.

Поддержка диалогового режима развивалась совместно с развитием самих систем программирования и была естественным образом интегрирована с ними. Библиотеки же доступа к базам данных развивались своим путем. Наибольшее число библиотек доступа из языков программирования уровня 3GL к реляционным СУБД на персональных компьютерах поддерживает семейство xBase (Clipper, FoxPro, dBase). Из языков программирования чаще всего используется Си.

Средства и методы разработки приложений на основе СУБД на персональных компьютерах

Приложения, созданные с использованием инструментальных средств программирования приложений, связанных с использованием баз данных на персональных компьютерах, занимают существенную долю файл-серверных приложений. Если рассматривать только "реляционные" (вернее, табличные) СУБД, то семейство xBase-продуктов является явным лидером по использованию для разработки одиночных и групповых информационных приложений. Следующее место занимает СУБД Paradox, а далее идут приложения, базирующиеся на использовании системы управления записями Clarion. Особняком стоят такие пакеты, как MS Access и Lotus Approach, которые позволили взглянуть по-новому на возможности персональных СУБД и до сих пор не оценены по-настоящему как профессиональные средства разработки приложений. Можно отметить следующие вехи на пути развития инструментальных средств и самих СУБД на персональных компьютерах:

Появление компонентов Assistant и Application Generator в dBase III Plus, упрощающих работу пользователя и позволяющих генерировать простейшие приложения или макеты приложений;

Выход в свет dBase-совместимых систем программирования (dBFast и Clipper), создающих исполняемый модуль приложения; разработка быстрого интерпретатора FoxBase для частично откомпилированного кода dBase-совместимых приложений;

Возникновение системы Paradox с оригинальным макроязыком PAL, существенно ориентированной на конечного пользователя;

Развитие многопользовательских версий СУБД для локальных сетей персональных компьютеров, дополненных средствами синхронизации на основе блокировок файлов и записей;

Появление системы dBase IV, включающую диалоговую среду Control Center, индексы, встроенные в файл БД, поддержку языка SQL и средства защиты БД;

Развитие Clipper с объектной ориентацией;

Обеспечение доступа из файл-серверных приложений к серверам БД (Borland SQL Link и Microsoft Connectivity Kit);

Внедрение технологии Rushmore, ускоряющей доступ к данным при помощи использования индексов;

Появление в FoxPro развитой среды разработки, ориентированной на разработку проектов и близкой по возможностям к средствам 4GL;

Дальнейшее расширение средств диалога (Foundation Read) в направление событийной управляемости;

Первые версии инструментальных средств, поддерживающие Windows-приложения, а вместе с ними типы данных Blob (Binary Large Objects);

Появление универсальных интерфейсов к различным СУБД (Borland IDAPI и Microsoft ODBC);

Первый продукт MS Access, направленный сугубо на создание Windows-приложений и содержащий средства объектно-ориентированного диалога, событийно-управляемого программирования, визуального конструирования интерфейса пользователя и многие другие черты, присущие системам программирования 4GL и RAD;

Появление новых визуальных объектно-ориентированных инструментальных средств и СУБД на ПК (MS Access 2.0, Visual FoxPro, CA-VisualObjects и Visual dBase).

2. Новые средства разработки файл-серверных приложений

Чем дальше, тем больше файл-серверные приложения сближаются с более развитыми технологиями клиент-серверных приложений. В последнее время появилась целая серия программных продуктов, позиционируемая одновременно как средства "легкой" разработки приложений для персональных компьютеров, так и более "тяжеловесной" разработки приложений в технологии "клиент-сервер". Это и хорошо, поскольку позволяет плавно изменять технологию.

Общая характеристика современных средств

В индустрии СУБД для персональных компьютеров отразились тенденции нормализации систем (Rightsizing). В последнее время в этой области происходили два встречных процесса: (1) разукрупнение серверов БД - появление новых версий серверов БД Informix, Oracle и т.д. сначала в варианте для рабочих групп, а потом облегченные версии для одиночных персональных компьютеров; (2) укрупнение СУБД для персональных компьютеров - новые "персональные" СУБД и связанные с ними инструментальные средства развивались в сторону "истинно реляционных" СУБД, т.е. серверов БД, приложений клиент-сервер и инструментальных средств программирования 4GL и быстрой разработки RAD.

Новые СУБД для персональных компьютеров и соответствующие инструментальные средства разработки

Визуальный характер программирования приложений особенно в части создания диалогового графического интерфейса пользователя. Это множество поддерживаемых диалоговых объектов, поддержка механизма drag-and-drop и наличие мастеров, помогающих реализовать сложные процедуры.

Управляемость приложений в соответствии с событиями диалога и обеспечение доступа к БД позволяет строить гибкий интерфейс пользователя и поддерживать ссылочную целостность БД.

Встроенная поддержка языка структурированных запросов SQL (Standard Query Language) закладывает возможность масштабирования создаваемых файл-серверных приложений до уровня приложений клиент-сервер.

Имеется возможность построения приложений клиент-сервер за счет реализации доступа к серверам БД напрямую или через интерфейс ODBC для открытого взаимодействия с базами данных.

Использование объектно-ориентированного языка разработки приложений (по крайней мере в части диалога) позволяет широко использовать механизм наследования и тем самым использовать ранее произведенные программные компоненты.

Поддержка компонентно-ориентированного программирования дает возможность расширения приложений за счет использования готовых внешних визуальных объектов типа VBX и OCX (ActiveX).

"Истинно реляционная" база данных представляет собой объединенный набор файлов, содержащий таблицы, индексы и т.п., что облегчает сопровождение БД и приложений и является основой для поддержки целостности данных.

Поддерживается общий для информационной системы словарь данных (data dictionary), который содержит описание структуры БД, типы полей, правила поддержки ограничений целостности и т.п.

Поддержка целостности БД (данных, ссылок и транзакций) позволяет создавать приложения с необходимым уровнем надежности и сохранности данных.

Возможности серверных процедур обработки (триггеров и хранимых процедур) закладывают основу для масштабирования приложений, позволяют гибко распределять прикладную логику между клиентом и сервером при переходе к архитектуре клиент-сервер.

Хранение в БД описания проекта создаваемого приложения является прообразом репозитория инструментальных средств быстрой разработки RAD и CASE-систем.

Средства и методологии проектирования, разработки и сопровождения приложений в архитектуре "клиент-сервер"

1. Базовые средства построения ИС в архитектуре "клиент-сервер"

Применительно к системам баз данных архитектура "клиент-сервер" актуальна главным образом потому, что обеспечивает простое и относительно дешевое решение проблемы коллективного доступа к базам данных в локальной сети. В некотором роде системы баз данных, основанные на архитектуре "клиент-сервер", являются приближением к распределенным системам баз данных, конечно, существенно упрощенным приближением, но зато не требующим решения основного набора проблем действительно распределенных баз данных.

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. Главной побудительной причиной развития концепции открытых систем явились повсеместный переход к использованию локальных компьютерных сетей и необходимость решения проблем комплексирования аппаратно-программных средств, которые вызвал этот переход. В связи с бурным развитием технологий глобальных коммуникаций открытые системы приобретают еще большее значение и масштабность.

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

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.

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

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.

Вызовы удаленных процедур

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.

2. Серверы баз данных как базовая системная поддержка информационной системы в архитектуре "клиент-сервер"

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.

Понятие сервера баз данных

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.

Этот 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. Соблюдая предосторожности при программировании, можно создавать прикладные информационные системы, мобильные в классе SQL-серверов.

Серверы баз данных, интерфейс которых основан исключительно на языке SQL, обладают своими преимуществами и своими недостатками. Очевидное преимущество - стандартность интерфейса. В пределе, который вряд ли полностью достижим, 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.

Типичный сервер баз данных отвечает за выполнение следующих функций:

поддержание логически согласованного набора файлов;

обеспечение языка манипулирования данными;

восстановление информации после разного рода сбоев;

организацию реально параллельной работы нескольких пользователей.

Непосредственное управление данными во внешней памяти

Эта функция включает обеспечение необходимых структур внешней памяти как для хранения непосредственных данных, входящих в БД, так и для служебных целей, например, для убыстрения доступа к данным в некоторых случаях (обычно для этого используются индексы). В некоторых реализациях серверов баз данных активно используются возможности существующих файловых систем, в других работа производится вплоть до уровня устройств внешней памяти. Но в развитых СУБД пользователи в любом случае не обязаны знать, использует ли СУБД файловую систему, и если использует, то как организованы файлы. В частности, СУБД поддерживает собственную систему именования объектов БД.

Управление буферами оперативной памяти

Серверы баз данных обычно работают с БД значительного размера; по крайней мере этот размер обычно существенно больше доступного объема оперативной памяти. Понятно, что если при обращении к любому элементу данных будет производиться обмен с внешней памятью, то вся система будет работать со скоростью устройства внешней памяти. Практически единственным способом реального увеличения этой скорости является буферизация данных в оперативной памяти. При этом, даже если операционная система производит общесистемную буферизацию (как в случае ОС UNIX), этого недостаточно для целей СУБД, которая располагает гораздо большей информацией о полезности буферизации той или иной части БД. Поэтому в развитых СУБД поддерживается собственный набор буферов оперативной памяти с собственной дисциплиной замены буферов.

Управление транзакциями

Транзакция - это последовательность операций над БД, рассматриваемых СУБД как единое целое. Либо транзакция успешно выполняется, и СУБД фиксирует (COMMIT) изменения БД, произведенные этой транзакцией, во внешней памяти, либо ни одно из этих изменений никак не отражается в состоянии БД. Понятие транзакции необходимо для поддержания логической целостности БД. Если воспользоваться примером информационной системы с файлами СОТРУДНИКИ и ОТДЕЛЫ, то единственным способом не нарушить целостность БД при выполнении операции приема на работу нового сотрудника является объединение элементарных операций над файлами СОТРУДНИКИ и ОТДЕЛЫ в одну транзакцию. Таким образом, поддержание механизма транзакций является обязательным условием даже однопользовательских СУБД (если, конечно, такая система заслуживает названия СУБД). Но понятие транзакции гораздо более важно в многопользовательских СУБД.

Журнализация

Одним из основных требований к СУБД является надежное хранение данных во внешней памяти. Под надежностью хранения понимается то, что СУБД должна быть в состоянии восстановить последнее согласованное состояние БД после любого аппаратного или программного сбоя. Обычно рассматриваются два возможных вида аппаратных сбоев: так называемые мягкие сбои, которые можно трактовать как внезапную остановку работы компьютера (например, аварийное выключение питания), и жесткие сбои, характеризуемые потерей информации на носителях внешней памяти. Примерами программных сбоев могут быть: аварийное завершение работы СУБД (по причине ошибки в программе или в результате некоторого аппаратного сбоя) или аварийное завершение пользовательской программы, в результате чего некоторая транзакция остается незавершенной. Первую ситуацию можно рассматривать как особый вид мягкого аппаратного сбоя; при возникновении последней требуется ликвидировать последствия только одной транзакции.

Языки БД

Для работы с базами данных используются специальные языки, в целом называемые языками баз данных. В ранних СУБД поддерживалось несколько специализированных по своим функциям языков. Чаще всего выделялись два языка - язык определения схемы БД (SDL - Sсhema Definition Language) и язык манипулирования данными (DML - Data Manipulation Language). SDL служил главным образом для определения логической структуры БД, т.е. той структуры БД, какой она представляется пользователям. DML содержал набор операторов манипулирования данными, т.е. операторов, позволяющих заносить данные в БД, удалять, модифицировать или выбирать существующие данные.

В современных СУБД обычно поддерживается единый интегрированный язык, содержащий все необходимые средства для работы с БД, начиная от ее создания, и обеспечивающий базовый пользовательский интерфейс с базами данных. Стандартным языком наиболее распространенных в настоящее время реляционных СУБД является язык SQL (Structured или Standard Query Language).

Прежде всего, язык SQL сочетает средства SDL и DML, т.е. позволяет определять схему реляционной БД и манипулировать данными. При этом именование объектов БД (для реляционной БД - именование таблиц и их столбцов) поддерживается на языковом уровне в то смысле, что компилятор языка SQL производит преобразование имен объектов в их внутренние идентификаторы на основании специально поддерживаемых служебных таблиц-каталогов. Внутренняя часть СУБД (ядро) вообще не работает с именами таблиц и их столбцов.

Язык SQL содержит специальные средства определения ограничений целостности БД. Опять же, ограничения целостности хранятся в специальных таблицах-каталогах, и обеспечение контроля целостности БД производится на языковом уровне, т.е. при компиляции операторов модификации БД компилятор SQL на основании имеющихся в БД ограничений целостности генерирует соответствующий программный код.

Специальные операторы языка SQL позволяют определять так называемые представления БД, фактически являющиеся хранимыми в БД запросами (результатом любого запроса к реляционной БД является таблица) с именованными столбцами. Для пользователя представление является такой же таблицей, как любая базовая таблица, хранимая в БД, но с помощью представлений можно ограничить или наоборот расширить видимость БД для конкретного пользователя. Поддержание представлений производится также на языковом уровне.

Средства и методологии проектирования, разработки и сопровождения Intranet-приложений

Intranet-система может основываться на локальной сети компьютеров, собственной корпоративной глобальной сети или виртуальной корпоративной подсети Internet. Различают несколько типов Intranet-систем, для реализации каждого из которых, вообще говоря, применяются разные средства:

Коммуникационные Intranet-системы предназначены главным образом для связывания территориально разнесенных подразделений корпорации, уменьшая потребность в многочисленных выделенных линиях связи.

Интегрирующие Intranet-системы служат для интеграции разнородных существующих коммуникационных и обрабатывающих корпоративных подсистем

Intranet-системы с упрощенной для пользователей процедурой доступа обычно основываются на механизме электронной подписи. Такие системы должны быть особенно надежно защищены от внешнего мира.

В общем случае информационная Intranet-система может включать свойства каждого из перечисленных типов, и в этом случае при ее разработке придется учитывать все требования.

1. Основные понятия Intranet

Поскольку для разработки Intranet-систем используются методы и средства Internet, и главным образом, технология WWW, то и понятия и термины Internet и Intranet совпадают. Некоторые из них:

HTTP (HyperText Transfer Protocol) - протокол обмена гипертекстовой информацией;

URL (Universal Resource Locator) - универсальный локатор ресурсов. Используется в качестве универсальной схемы адресации ресурсов в сети.

HTML (HyperText Markup Language) - язык гипертекстовой разметки документов. Специальная форма подготовки документов для их опубликования в World Wide Web.

CGI (Common Gateway Interface) - спецификация на формат обмена данными между сервером протокола HTTP и прикладной программой.

API (Application Program Interface) - в данном контексте это спецификация, определяющая правила обмена данными между сервером и программным модулем, который должен быть включен в состав сервера.

VRML (Virtual Reality Modeling Language) - язык описания трехмерных сцен и взаимодействия трехмерных объектов.

Javaapplets - мобильные (независимые от архитектуры "железа") программные коды, написанные на языке программирования Java.

Java - объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems и используемый в качестве основного средства мобильного программирования.

MIME (Multipurpose Internet Mail Exchange) - формат почтового сообщения Internet. В данном контексте стандарт MIME используется для установления соответствия между типом информационного файла, именем этого файла и программой просмотра этого файла.

CCI (Common Client Interface) - спецификация обмена данными меду прикладной программой и браузером Mosaic. В случае применения программного обеспечения, выполненного согласно CCI, браузер превращается в сервер-посредник для программного обеспечения пользователя.

2. Языки и протоколы

К 1989 году гипертекст представлял новую, многообещающую технологию, которая имела относительно большое число реализаций с одной стороны, а с другой стороны делались попытки построить формальные модели гипертекстовых систем, которые носили скорее описательный характер и были навеяны успехом реляционного подхода описания данных. Идея Т.Бернерс-Ли заключалась в том, чтобы применить гипертекстовую модель к информационным ресурсам, распределенным в сети, и сделать это максимально простым способом. Он заложил три краеугольных камня системы, разработав: язык гипертекстовой разметки документов HTML (HyperText Markup Language), универсальный способ адресации ресурсов в сети URL (Universal Resource Locator), протокол обмена гипертекстовой информацией HTTP (HyperText Transfer Protocol). Позже команда NCSA (National Center of Supercomputer Applications) добавила к этим трем компонентам четвертый - универсальный интерфейс шлюзов CGI (Common Gateway Interface). Усовершенствования на этом не остановились. В 1995 году Netscape Communication предлагает включить в текст HTML-страниц программы на новом языке Liveware. Позже это название было изменено на JavaScript. Таким образом технология World Wide Web была дополнена еще одним компонентом - языком программирования просмотра гипертекстовых страниц JavaScript.

Исходя из важности технологии WWW для построения корпоративных Intranet-систем, мы для начала коротко рассмотрим два базовых компонента этой технологии - язык гипертекстовой разметки документов HTML и протокол обмена гипертекстовой информацией HTTP.

HTML

Идея гипертекстовой информационной системы состоит в том, что пользователь имеет возможность просматривать документы (страницы текста) в том порядке, в котором ему это больше нравится, а не последовательно, как это принято при чтении книг. Поэтому Т.Нельсон и определил гипертекст как нелинейный текст. Достигается это путем создания специального механизма связи различных страниц текста при помощи гипертекстовых ссылок, т.е. у обычного текста есть ссылки типа "следующий-предыдущий", а у гипертекста можно построить еще сколь угодно много других ссылок. Любимыми примерами специалистов по гипертексту являются энциклопедии, Библия, системы типа "Help".

Простой, на первый взгляд, механизм построения ссылок оказывается довольно сложной задачей, т.к. можно построить статические ссылки, динамические ссылки, ассоциированные с документом в целом или только с отдельными его частями, т.е. контекстные ссылки. Дальнейшее развитие этого подхода приводит к расширению понятия гипертекста за счет других информационных ресурсов, включая графику, аудио- и видео-информацию, до понятия гипермедиа.

При разработке HTML была поставлена задача решить две основных проблемы:

дать дизайнерам гипертекстовых баз данных простое средство создания документов;

сделать это средство достаточно мощным, чтобы отразить имевшиеся на тот момент представления об интерфейсе пользователя гипертекстовых баз данных.

Первая задача была решена за счет выбора таговой модели описания документа. Такая модель широко применяется в системах подготовки документов для печати. Примером такой системы является хорошо известный язык разметки научных документов TeX, предложенный Американским Математическим Обществом, и программы его интерпретации.

К моменту создания HTML существовал стандарт языка разметки печатных документов - Standard Generalised Markup Language, который и был взят в качестве основы HTML. Предполагалось, что такое решение поможет использовать существующее программное обеспечение для интерпретации нового языка. Однако, будучи доступным широкому кругу пользователей Internet, HTML зажил своей собственной жизнью.

Вторым важным моментом, повлиявшим на судьбу HTML, стал выбор в качестве элемента гипертекстовой базы данных обычного текстового файла, который хранится средствами файловой системы операционной среды компьютера. Такой выбор был сделан под влиянием следующих факторов:

такой файл можно создать в любом текстовом редакторе на любой аппаратной платформе в среде любой операционной системы.

к моменту разработки HTML существовал американский стандарт для разработки сетевых информационных систем - Z39.50, в котором в качестве единицы хранения указывался простой текстовый файл в кодировке LATIN1, что соответствует US ASCII.

Таким образом, гипертекстовая база данных в концепции WWW - это набор текстовых файлов, написанных на языке HTML, который определяет форму представления информации (разметка) и структуру связей этих файлов (гипертекстовые ссылки). Но это только концепция, в реальной жизни все оказалось гораздо сложнее. В ряде случаев реальная страница Website собирается из различных файлов, записей баз данных и результатов выполнения скриптов, но в любом случае браузер получает правильно размеченный на HTML текст.

Такой подход предполагает наличие еще одной компоненты технологии - интерпретатора языка. В World Wide Web функции интерпретатора разделены между сервером гипертекстовой базы данных и интерфейсом пользователя.

Сервер, кроме доступа к документам и обработки гипертекстовых ссылок, осуществляет также предпроцессорную обработку документов, в то время как интерфейс пользователя осуществляет интерпретацию конструкций языка, связанных с представлением информации.

К настоящему времени известна уже третья версия языка - HTML 3.2, которая находится в стадии развития. Если первая версия языка (HTML 1.0) была направлена на представление языка как такового, где описание его возможностей носило скорее рекомендательный характер, вторая версия языка (HTML 2.0) фиксировала практику использования конструкций языка, версия ++ (HTML++) представляла новые возможности, расширяя набор элементов HTML в сторону отображения научной информации и таблиц, а также улучшения стиля компоновки изображений и текста, то версия 3.2 призвана упорядочить все нововведения и согласовать их с существующей практикой. Кроме этого, в версии 3.2 снова делается попытка формализации интерфейса пользователя гипертекстовой распределенной системы. Однако, коммерческое использование World Wide Web постепенно вытесняет или сильно замедляет использование научных расширений языка. Так математическую разметку документа поддерживает только один браузер Arena.

Главным следствием влияния программного обеспечения на развитие языка, является включение в HTML механизма форм заполнения (FILL-OUT FORMS).

Формы впервые были подробно описаны в инструкциях по использованию сервера NCSA. До введения форм в WWW был только один механизм передачи параметров - поле ключевых слов. WWW была информационно-справочной системой и только. Формы произвели революцию в технологии WWW и превратили ее в технологию открытой системы. Посредством форм стала возможна передача параметров внешним программам, которые вызываются сервером, что сделало WWW универсальным интерфейсом ко всем ресурсам сети.

Вторым краеугольным камнем WWW стала универсальная форма адресации информационных ресурсов. Universal Resource Identification (URI) представляет собой довольно стройную систему, учитывающую опыт адресации и идентификации e-mail, Gopher, WAIS, telnet, ftp и т. п. Но реально из всего, что описано в URI, для организации баз данных в WWW требуется только Universal Resource Locator (URL). Без наличия этой спецификации вся мощь HTML оказалась бы бесполезной. URL используется в гипертекстовых ссылках и обеспечивает доступ к распределенным ресурсам сети. В URL можно адресовать как другие гипертекстовые документы формата HTML, так и ресурсы e-mail, telnet, ftp, Gopher, WAIS. Различные интерфейсные программы по-разному осуществляют доступ к этим ресурсам. Одни, как, например, Netscape, сами способны поддерживать взаимодействие по протоколам, отличным от протокола HTTP, базового для WWW, другие, как например Chimera, вызывают для этой цели внешние программы. Однако, даже в первом случае, базовой формой представления отображаемой информации является HTML, а ссылки на другие ресурсы имеют форму URL. Следует отметить, что программы обработки электронной почты в формате MIME также имеют возможность отображать документы, представленные в формате HTML. Для этой цели в MIME зарезервирован тип "text/html".

HTTP

HTTP - это протокол прикладного уровня, разработанный для обмена гипертекстовой информацией в сети Internet. Протокол используется одной из популярнейших систем Сети - Word Wide Web - с 1990 года.

Реальная информационная система требует гораздо большего количества функций, чем просто поиск. HTTP позволяет реализовать в рамках обмена данными набор методов доступа, базирующихся на спецификации универсального идентификатора ресурсов (Universal Resource Identifier), применяемого в форме универсального локатора ресурсов (Universe Resource Locator) или универсального имени ресурса (Universal Resource Name). Сообщения по сети при использовании протокола HTTP передаются в формате, схожим с форматом почтового сообщения Internet (RFC-822) или с форматом сообщений MIME (Multiperposal Internet Mail Exchange). HTTP используется для взаимодействия программ-клиентов с программами-шлюзами, разрешающими доступ к ресурсам электронной почты Internet (SMTP), спискам новостей (NNTP), файловым архивам (FTP), системам Gopher и WAIS. Протокол разработан для доступа к этим ресурсам посредством промежуточных программ-серверов (proxy), которые позволяют передавать информацию между различными информационными службами без потерь. Протокол реализует принцип "запрос/ответ". Запрашивающая программа - клиент - инициирует взаимодействие с отвечающей программой - сервером, и посылает запрос, включающий в себя метод доступа, адрес URI, версию протокола, похожее по форме на MIME-сообщение с модификаторами типа передаваемой информации, информацию клиента, и, возможно, тело сообщения клиента. Сервер отвечает строкой состояния, включающей версию протокола и код возврата, за которой следует сообщение в форме, похожей на MIME. Данное сообщение содержит информацию сервера, метаинформацию и тело сообщения. Понятно, что в принципе, одна и та же программа может выступать и в роли сервера и в роли клиента (так собственно и происходит при использовании proxy-серверов).

При работе в Internet для обслуживания HTTP-запросов используется 80 порт TCP/IP. Практика использования протокола такова, что клиент устанавливает соединение и ждет ответа сервера. После отправки ответа сервер инициирует разрыв соединения. Таким образом, при передаче сложных гипертекстовых страниц соединение может устанавливаться несколько раз. Остановимся более подробно на механизме взаимодействия и форме передаваемой информации.

3. Серверы Intranet

В Intranet можно использовать все возможные сервисы Internet (например, электронную почту, возможности удаленного терминала и т.д.).

FTP-серверы

FTP-архивы являются одним из основных информационных ресурсов Internet. Фактически, это распределенный депозитарий текстов, программ, фильмов, фотографий, аудио записей и прочей информации, хранящейся в виде файлов на различных компьютерах во всем мире.

Типы информационных ресурсов

Информация в FTP-архивах разделена на три категории:

Защищенная информация, режим доступа к которой определяется ее владельцами и разрешается по специальному соглашению с потребителем. К этому виду ресурсов относятся коммерческие архивы (например, коммерческие версии программ в архивах ftp.microsoft.com или ftp.bsdi.com), закрытые национальные и международные некоммерческие ресурсы (например, работы по международным проектам CES или IAEA), частная некоммерческая информация со специальными режимами доступа (частные благотворительные фонды, например).

Информационные ресурсы ограниченного использования, к которым относятся, например, программы класса shareware (Trumpet Winsock, Atis Mail, Netscape, и т.п.). В данный класс могут входить ресурсы ограниченного времени использования или ограниченного времени действия, т.е. пользователь может использовать текущую версию на свой страх и риск, но никто не будет оказывать ему поддержку.

Свободно распространяемые информационные ресурсы или freeware, если речь идет о программном обеспечении. К этим ресурсам относится все, что можно свободно получить по сети без специальной регистрации. Это может быть документация, программы или что-либо еще. Наиболее известными свободно распространяемыми программами являются программы проекта GNU Free Software Foundation. Следует отметить, что свободно распространяемое программное обеспечение не имеет сертификата качества, но как правило, его разработчики открыты для обмена опытом.

Технология FTP была разработана в рамках проекта ARPA и предназначена для обмена большими объемами информации между машинами с различной архитектурой. Главным в проекте было обеспечение надежной передачи и поэтому с современной точки зрения FTP кажется перегруженным излишними редко используемыми возможностями. Стержень технологии составляет FTP-протокол.

WWW-серверы

Сервер World Wide Web - это программа, обслуживающая запросы клиентов по протоколу HTTP. Главной задачей сервера "паутины" является обеспечение доступа пользователей к базе данных HTML-документов. Однако, в настоящее время функциональные возможности серверов значительно расширились и вышли за пределы простой отсылки документов на запросы клиентов. Наиболее типичными для современных серверов являются следующие функции:

ведение иерархической базы данных документов,

контроль за доступом к информации со стороны программ-клиентов,

предварительная обработка данных перед ответом на запрос,

взаимодействие с внешними программами через Common Gateway Interface,

реализация взаимодействия с клиентами и другими серверами в режиме посредника,

реализация встроенных или взаимодействие с внешними поисковыми машинами.

Кроме того, такие серверы как NetSite (Netscape Communication) и Apachie реализуют шифрованные протоколы HTTP для обмена информацией с клиентами. Рассмотрение перечисленных свойств и функций WWW-серверов полезно провести на примерах, основанных на практике эксплуатации серверов NCSA httpd, CERN httpd, Winhttpd и WN, Apachie.

4. Язык программирования Java

Весь 1996 год прошел под знаком внедрения в World Wide Web технологии Java. Однако реальные проекты, выполненные в этой технологии появились только к концу года.

Если обратиться к публикациям о новых концепциях разработки программного обеспечения, то Java-язык и Java-технологии ценятся не только и не столько за то, что они дали возможность "оживить" Web, как часто можно прочитать в рекламных проспектах, а за то, что давняя идея мобильного программирования для различных аппаратных платформ наконец-то стала реальностью.

При этом все прекрасно понимают, что сама по себе Java-технология еще чрезвычайно сыра и использовать ее в нынешнем виде для серьезных проектов могут только самые отчаянные смельчаки, тем не менее многие из свойств Java чрезвычайно интересны и полезны.

Мобильность Java

Суть мобильности заключается в том, что написанный на Java код может исполняться на любой компьютерной платформе. Под платформой в данном случае подразумевается программно-аппаратный комплекс, а проще собственно компьютер и операционная система, которая на нем работает. Однако, мобильность Java следует отличать от простого портирования кода с одной платформы на другую. При портировании передается исходный текст, который после этого компилируется и только затем выполняется. В Java передаются так называемые байт-коды. Байт-код - это универсальный формат программы, единый для всех аппаратных платформ. Байт-код един и для рабочих станций, и для больших универсальных ЭВМ, и для персональных компьютеров.

Байт-код Java исполняется интерпретатором, а не является откомпилированной на данной платформе программой. Многие преимущества и недостатки Java проистекают именно из-за этого. Первый и очевидный недостаток - это скорость выполнения кода. Очевидно, что интерпретатор работает гораздо медленнее откомпилированного кода. И здесь не помогут никакие Java-процессоры. Вся элементная база все равно остается двоичной, а это значит, что программы оптимизированные для работы с этим типом вычислений будут работать все равно быстрее, чем какие-либо другие коды. Здесь можно говорить, только о сравнении Java-процессоров с универсальными процессорами при выполнении Java-кода.

Собственно, и другие свойства технологии (объектная ориентируемость, использование многопоточности, отсутствие адресной арифметики и т.п.) в большинстве случаев при стандартной комплектации оборудования, только тормозят выполнение программы.

Второй недостаток - объектный подход к программированию. Главным в Java-технологии являются небольшие программки-апплеты, которые передаются по сети. Размер такого апплета очень мал (до сотни строк текста). При этом разработчик должен весь вывод собрать в одну процедуру, перехват особых ситуаций в другую, порождение потоков в третью, и т. д. При этом некоторые из этих процедур будут вообще состоять из четырех - пяти строк. Это конечно дисциплинирует и ведет к построению стройных кодов, но при этом прокрутка текста на экране превращается в программу с порождением нового потока, что вместо сокращения издержек на ее выполнение приводит только к их увеличению.

Однако, мобильность Java имеет еще одно измерение. Апплеты передаются по сети в тот момент, когда их нужно выполнять. При этом они могут загружаться из разных мест и после загрузки взаимодействовать друг с другом. Если всю совокупность апплетов рассматривать как одну программу, состоящую из множества подгружаемых по мере необходимости модулей, то весь Internet становится одним большим компьютером, а точнее хранилищем данных и программ, т.к. все-таки программа выполняется на локальном процессоре. Java-апплеты также могут обмениваться данными между собой по сети, что заставляет говорить о распределенных вычислениях, и в этом случае программа будет исполняться на многих компьютерах в сети. До Java можно было говорить о распределенных вычислениях только после трансляции кодов для каждой из машин и отладки всего комплекса в целом. Теперь отладиться можно и на локальном компьютере, а реально считать на разных.

Основной протокол обмена апплетами - HTTP. Это значит, что они передаются по сети точно также, как и другие ресурсы Website, и приобретают свое особое значение только в момент получения их браузером. Учитывая неэффективность реализации HTTP поверх TCP, можно сказать, что и обмен апплетами тоже неэффективен, если для каждого обмена устанавливается свое TCP-соединение. Пока сравнение скорости выполнения Java-апплетов и скриптов JavaScript не в пользу первых, т.е. интерпретация скриптов в Netscape Navigator реализована эффективнее. В любом случае загрузка страниц с Java происходит гораздо медленнее, чем без Java.

Часто можно слышать, что использование Java должно снизить затраты на обмен информацией по сети. Аргумент здесь следующий: для отображения информации в виде графиков и гистограмм, и даже трехмерных объектов не надо передавать по сети изображения этих объектов, а можно передать, только числовые характеристики. Это приводит к существенному сокращению трафика, и сокращению числа ошибок при передаче. Это справедливо в том случае, если апплеты Java не перегружаются часто. Пользователь загрузил группу страниц и все апплеты на них и только стартует или останавливает выполнение программ. В случае специализированных корпоративных вычислений может быть это и так, но если брать обычного пользователя, то он практически всегда, переходя на новую страницу, загружает новые апплеты, а вот это уже не сокращает трафик, а только увеличивает.

Мобильность имеет и свою обратную сторону. Некоторые авторы прежде, чем пустить пользователя на свои страницы с Java предупреждают: Are your going to be infected? Суть предупреждения в том, что, фактически, пользователь разрешает выполняться на своем компьютере программе, о свойствах которой он ничего не знает. При этом ссылки на безопасность Java не вполне корректны. Программа может вообще никак себя внешне не проявлять, но при этом выполнять массу нежелательных, с точки зрения пользователей, действий. Например, анализировать кэш браузера, идентифицировать компьютер пользователя и многое другое.

5. Возможные архитектуры Intranet-приложений

Решения, ориентированные на клиентскую часть системы

Наиболее тривиальная архитектура. Аналогично тому, как в файл-серверных решениях, вся прикладная часть системы находится в клиенте, который взаимодействует с разнообразными серверами Internet (электронной почты, ftp и т.д.) и серверами, управляющими файлами и/или базами данных. Клиент должен быть достаточно "толстым", чтобы быть в состоянии уметь работать с разными видами серверов (для каждого из них требуется индивидуальная клиентская часть) и одновременно выполнять прикладную обработку данных. Серверы могут быть разной толщины в зависимости от своей функциональной ориентированности (сервер электронной почты нуждается в существенно меньшем числе ресурсов, чем мощный сервер баз данных).

Трехзвенные архитектуры (Web-ориентированные)

Ориентация на использование Web-технологии позволяет одновременно добиться двух эффектов. Во-первых, за счет использования CGI или API можно перенести на сторону сервера часть логики приложения. Во-вторых, используя технику шлюзования Web-сервера (опять же применяя CGI-шлюзы или API) можно работать через Web-сервер (в стандартном интерфейсе) с другими серверами. Клиента можно сделать очень "тонким", Web-сервер будет достаточно "толстым", а остальные такими, как получится.

Решения, основанные на использовании языка Java

Язык Java можно использовать для программирования Java-апплетов, которые выполняются на стороне клиента, и Java-приложений, выполняемых на стороне сервера. Естественно, клиент, приспособленный к выполнению Java-апплетов, становится несколько толще. Что же касается использования Java-программ на стороне сервера, то большее значение может иметь сравнительная надежность этого языка (в том смысле, что интерпретируемая Java-программа с меньшей вероятностью может нанести вред серверу).

Информационные приложения, основанные на использовании "складов данных" (DataWarehousing)

Здесь рассматриваются вопросы организации специального класса информационных приложений, ориентированных не на оперативную обработку транзакций (On-Line Transaction Processing - OLTP), а на оперативную аналитическую обработку (On-Line Analitical Processing - OLAP). У этих двух разновидностей систем принципиально разные задачи. Корпоративные информационные OLTP-системы создаются для того, чтобы способствовать повседневной деятельности корпорации, и опираются на актуальные для текущего момента данные. OLAP-системы служат для анализа деятельности корпорации или ее компонентов и прогнозирования будущего состояния. Для этого требуется использовать многочисленные накопленные данные о деятельности корпорации в прошлом, а также внешние источники данных, формирующие контекст, в котором работала корпорация.

Система оперативной аналитической обработки данных отличается от статической системы поддержки принятия решений (Decision Support System - DSS) тем, что OLAP-система позволяет аналитику динамически формировать класс вопросов, который требуется для решаемой им текущей аналитической задачи. DSS обеспечивает выдачу отчетов в соответствии с заранее сформулированными правилами. Для удовлетворения нового запроса нужно формально его описать, запрограммировать и только потом выполнить.

1. Проблема интеграции данных

Любая крупная и давно существующая корпорация обладает несколькими базами данных, относящимися к разным видам деятельности. Данные могут иметь разные представления, а иногда могут быть даже несогласованными (например, из-за ошибки ввода в одну из баз данных). Это нехорошо даже для OLTP-систем и в принципе непригодно для OLAP-систем, которые должны обрабатывать общие исторические согласованные корпоративные данные. Для оперативной аналитической обработки требуется привлечение внешних источников данных, которые тем более могут обладать разными форматами и требовать согласования.На подобных рассуждениях и возникла концепция склада данных как предметно-ориентированного, интегрированного, неизменчивого, поддерживающего хронологию набора данных, организованного для целей поддержки управления.

Подход построения склада данных для интеграции неоднородных источников данных принципиально отличается от подхода динамической интеграции разнородных баз данных. В случае склада данных реально строится новое крупномасштабное хранилище, управление данными в котором происходит, вообще говоря, по другим правилам, нежели в исходных оперативных базах данных.

В основе концепции склада данных лежат две основные идеи:

Интеграция разъединенных детализированных данных (детализированных в том смысле, что они описывают некоторые конкретные факты, свойства, события и т.д.) в едином хранилище. В процессе интеграции должно выполняться согласование рассогласованных детализированных данных и, возможно, их агрегация. Данные могут поступать из исторических архивов корпорации, оперативных баз данных, внешних источников.

Разделение наборов данных, используемых для оперативной обработки, и наборов данных, применяемых для решения задач анализа.

Некоторых проблемы реализации складов данных:

неоднородность программной среды;

распределенный характер организации;

повышенные требования к безопасности данных;

необходимость наличия многоуровневых справочников метаданных;

потребность в эффективном хранении и обработке очень больших объемов информации.

Склад данных практически никогда не создается на пустом месте. Почти всегда конечное решение будет разнородным, т.е. в нем будут использоваться автономно разработанные программные средства. Прежде всего это касается формирования интегрированного согласованного набора данных, которые могут поступать из разнородных баз данных, электронных архивов, публичных и коммерческих электронных каталогов, справочников, статистических сборников. При построении склада данных приходится решать задачу построения единой, согласованно функционирующей информационной системы на основе неоднородных программных средств и решений. При выборе средств реализации склада данных приходится учитывать множество факторов, включающих уровень совместимости различных программных компонентов, легкость их освоения и использования, эффективность функционирования и т.д.

В концепции склада данных предопределено то, что операционная аналитическая обработка может выполняться в любом узле сети независимо от места расположения основного хранилища. Хотя при аналитической обработке данные только читаются, и потребность в синхронизации отсутствует, для достижения эффективности необходимо поддерживать репликацию данных в разных узлах сети. (На самом деле, все не так просто. Одним из требований к складам данных является то, чтобы свежая информация поступала на склад как можно быстрее. Т.е. потенциально любая модификация оперативной базы данных может инициировать добавление данных к складу данных, а тогда потребуется обновить и все реплики, для чего синхронизация все-таки нужна.)

Собранная вместе согласованная информация об истории развития корпорации, ее успехах и неудачах, о взаимоотношениях с поставщиками и заказчиками, об истории и состоянии рынка дает возможность анализа прошлой и текущей деятельности корпорации и построения прогнозов для будущего. Эта информация настолько ценна для корпорации, что нельзя допустить возможности ее утечки . В системах, основанных на складах данных, оказывается недостаточной защита данных в стиле языка SQL, которую обеспечивают обычные коммерческие СУБД (этот уровень защиты соответствует классу C2 в соответствии с классификацией Оранжевой Книги Министерства обороны США). Для обеспечения должного уровня защиты доступ к данным должен контролироваться не только на уровне таблиц и их столбцов, но и на уровне отдельных строк (это уже соответствует классу B1 Оранжевой Книги). Приходится также решать вопросы аутентификации пользователей, защиты данных при их перемещении в склад данных из оперативных баз данных и внешних источников, защиты данных при их передаче по сети.

Если роль метаданных (обычно содержащихся в таблицах-каталогах) в оперативных информационных системах достаточно ограничена, то для OLAP-систем наличие развитых метаданных и средств их предоставления конечным пользователям является одним из основных условий успешной реализации. Например, прежде, чем менеджер корпорации задаст системе свой вопрос, он должен понять, какая информация имеется, насколько она актуальна, можно ли ей доверять, сколько времени может занять формирование ответа и т.д. Для пользователя OLAP-системы требуются метаданные, по крайней мере, следующих типов:

Описания структур данных, их взаимосвязей.

Информация о хранимых на складе данных и поддерживаемых им агрегатах данных.

Информация об источниках данных и о степени их достоверности. Одна и та же информация могла попасть в склад данных из разных источников. Пользователь должен иметь возможность узнать, какой источник был выбран основным, и каким образом производились согласование и очистка данных.

Информация о периодичности обновлений данных. Желательно знать не только то, какому моменту времени соответствуют интересующие его данные, но и когда они в следующий раз будут обновлены.

Информация о владельцах данных. Пользователю OLAP-системы может оказаться полезной информация о наличии в системе данных, к которым он не имеет доступа, о владельцах этих данных и о действиях, которые он должен предпринять, чтобы получить доступ к данным.

Статистические оценки времени выполнения запросов. До выполнения запроса полезно иметь хотя бы приблизительную оценку времени, которое потребуется для получения ответа, и объема этого ответа.

Уже сейчас известны примеры складов данных, содержащих терабайты информации. По данным консалтинговой компании Meta Group, около половины корпораций, использующих или планирующих использовать склады данных, предполагает довести их объем до сотен гигабайт. Проблемой таких больших хранилищ является то, что накладные расходы на внешнюю память возрастают нелинейно при возрастании объема хранилища. Исследования, проведенные на основе тестового набора TPC-D, показали, что для баз данных объемом в 100 гигабайт потребуется внешняя память объемом в 4.87 раза большая, чем нужно собственно для полезных данных. При дальнейшем росте баз данных этот коэффициент увеличивается.

В последнее время все более популярной становится идея совместить концепции склада и рынка данных в одной реализации и использовать склад данных в качестве единственного источника интегрированных данных для всех рынков данных. Тогда естественной становится такая трехуровневая организация OLAP-системы:

На первом уровне реализуется корпоративный склад данных на основе одной из развитых современных реляционных СУБД. Это хранилище интегрированных в основном детализированных данных. Реляционные СУБД обеспечивают эффективное хранение и управление данными очень большого объема, но не слишком хорошо соответствуют потребностям OLAP-систем, в частности, в связи с требованием многомерного представления данных.

На втором уровне поддерживаются рынки данных на основе многомерной системы управления базами данных (примером такой системы является Oracle Express Server). Он может содержать ссылки на склад данных и добирать оттуда информацию по мере поступления запросов. Конечно, это несколько увеличивает время отклика, но зато снимает проблему ограниченного объема многомерной базы данных.

Наконец, на третьем уровне находятся клиентские рабочие места конечных пользователей, на которых устанавливаются средства оперативного анализа данных.

2. Примеры реализации технологии складов данных у крупнейших компаний.

Компания IBM

Решение компании IBM называется A Data Warehouse Plus. Целью компании является обеспечение интегрированного набора программных продуктов и сервисов, основанных на единой архитектуре. Основой складов данных является семейство СУБД DB2. Преимуществом IBM является то, что данные, которые нужно извлечь из оперативной базы данных и поместить в склад данных, находятся в системах IBM. Поэтому естественная тесная интеграция программных продуктов.

Предлагаются три решения для складов данных:

Изолированный рынок данных. Предназначен для решения отдельных задач вне связи с общим хранилищем корпорации.

Зависимый рынок данных. Аналогичен изолированному рынку данных, но источники данных находятся под централизованным контролем.

Глобальный склад данных. Корпоративное хранилище данных, которое полностью централизовано контролируется и управляется. Глобальный склад данных может храниться централизовано или состоять из нескольких распределенных в сети рынков данных.

Oracle

Решение компании Oracle в области складов данных основывается на двух факторах: широкий ассортимент продуктов самой компании и деятельность партнеров в рамках программы Warehouse Technology Initiative. Возможности Oracle в области складов данных базируются на следующих составляющих:

наличие реляционной СУБД Oracle 7, которая постоянно совершенствуется для лучшего удовлетворения потребностей складов данных;

существование набора готовых приложений, обеспечивающих возможности разработки склада данных;

высокий технологический потенциал компании в области анализа данных;

доступность ряда продуктов, производимых другими компаниями.

Hewlett Packard

Работы, связанные со складами данных, выполняются в рамках программы OpenWarehouse. Выполнение этой программы должно обеспечить возможность построения складов данных на основе мощных компьютеров HP, аппаратуры других производителей и программных компонентов. Основой подхода HP являются Unix-платформы и программный продукт Intelligent Warehouse, который предназначен для управления складами данных. Основа построения складов данных, предлагаемая HP, оставляет свободу выбора реляционной СУБД, средств реинжиниринга и т.д.

Sybase

Стратегия компании в области складов данных основывается на разработанной ей архитектуре Warehouse WORKS. В основе подхода находится реляционная СУБД Sybase System 11, средство для подключения и доступа к базам данных OmniCONNECT и средство разработки приложений PowerBuilder. Компания продолжает совершенствовать свою СУБД для лучшего удовлетворения потребностей складов данных (например, введена побитная индексация).

Informix Software

Стратегия компании в отношение складов данных направлена на расширение рынка для ее продукта On-Line Dinamic Parallel Server. Предлагаемая архитектура склада данных базируется на четырех технологиях: реляционные базы данных, программном обеспечении для управления складом данных, средствах доступа к данным и платформе открытых систем. Три последние компонента разрабатываются партнерами компании. После выхода Универсального Сервера, основанного на объектно-реляционном подходе, можно ожидать, что и он будет использоваться для построения складов данных.

AT&T GIS

Решение компании направлено на решение проблем корпораций, у которых одинаково сильны потребности и в системах поддержки принятия решений, и в системах оперативной аналитической обработки данных. Предлагаемая архитектура называется Enterprise Information Factory и основывается на опыте использования системы управления базами данных Teradata и связанных с ней методах параллельной обработки.

SAS Institute

Компания считает себя поставщиком полного решения для организации склада данных. Подход основан на следующем:

обеспечение доступа к данным с возможностью их извлечения из самых разнообразных хранилищ данных (и реляционных, и нереляционных);

преобразование данных и манипулирование ими с использованием 4GL;

наличие сервера многомерных баз данных;

большой набор методов и средств для аналитической обработки и статистического анализа.

Software AG

Деятельность компании в области складов данных происходит в рамках программы Open Data Warehouse Initiative. Программа базируется на основных продуктах компании ADABAS и Natural 4GL, собственных и приобретенных средствах извлечения и анализа данных, средстве управления складом данных SourcePoint. SourcePoint позволяет автоматизировать процесс извлечения и пересылки данных, а также их загрузки в склад данных.

Существует еще целый ряд компаний, которые прямо или косвенно связаны с технологией складов данных, но мы ограничимся перечисленными, поскольку их продукты и подходы кажутся наиболее продвинутыми.

Глобально распределенные информационные системы

В мире существует громадное количество готовых к использованию информационно-вычислительных ресурсов. Они создавались в разное время, для их разработки использовались разные подходы. Почти всегда при разработке новой информационной системы можно найти подходящие по своим функциям уже работающие готовые компоненты. Проблема состоит в том, что при их создании не учитывались требования интероперабельности. Эти компоненты не понимают один другого, они не могут работать совместно. Желательно иметь механизм или набор механизмов, которые позволят сделать такие независимо разработанные информационно-вычислительные ресурсы интероперабельными.

Первым шагом на пути решения проблемы интеграции информационных ресурсов была попытка создать средства, позволяющие интегрировать набор разнородных баз данных (иерархических, сетевых, реляционных и т.д.). Такие средства должны были обеспечить возможность работы с неоднородными базами данных в единой концептуальной модели данных. Известные подходы основывались на использовании в качестве единой модели реляционной модели данных.

Несмотря на высокий уровень проработки системы управления интегрированными распределенными неоднородными базами данных так и не вышли за пределы академических экспериментов. Видимо, это связано с целым рядом причин, основной из которых, является то, что реляционная модель данных слишком ограничена, чтобы ее можно было использовать в качестве единой концептуальной модели.

Тем не менее, проблема интеграции остается очень актуальной, и в последние годы все большее число специалистов соглашаются с тем, что ее можно и нужно решать на основе объектно-ориентированного подхода.

Проблема интеграции неоднородных автономно разработанных информационно-вычислительных ресурсов рассматривалась в двух контекстах. Первый контекст - повторное использование (reusability) существующих и доступных по сети ресурсов. Второй контекст - облегчение разработки корпоративных информационных систем, отдельные компоненты которых создаются разными, территориально распределенными группами, каждая из которых в силу исторических причин использует наиболее привычную для нее технологию. Например, канадская компания BNR для разработки новых программных продуктов использует коллективы программистов из разных стран мира. Некоторые группы предпочитают использовать Си++, другие - объектный Лисп, третьи - Smalltalk и т.д. Но в результате должна появиться единая, реально работающая, программная система.

Но имеется и третий контекст, контекст унаследованных систем (legacy systems). В любой крупной, долгое время существующей корпорации накапливаются информационные подсистемы, разработанные в соответствии с морально устаревшими технологиями. Например, трудно найти корпорацию с возрастом больше 25 лет, в которой не использовались бы информационные подсистемы, созданные на основе ранних аппаратно-программных платформ компании IBM. Базы данных таких подсистем содержат громадные объемы ценной информации, и корпорация просто не может обойтись без их использования. С другой стороны, унаследованные системы очень трудно сопровождать и поддерживать. Очень часто программная часть системы написана на языке ассемблера, а люди, которые писали эти программы, больше не работают в корпорации. Возникают проблемы и с аппаратной частью.

Для корпорации было бы желательно перевести унаследованные информационные подсистемы на новые технологии, но работоспособность унаследованной системы может быть настолько важна для корпорации, что эту систему нельзя вывести из использования даже на короткое время.

Одно из наиболее признанных решений проблемы унаследованных систем основывается также на объектно-ориентированном подходе. Идея состоит в том, что "вокруг" системы создается объектная оболочка. Естественно, что при этом порождается и новый объектный интерфейс системы, но до поры сохраняется и ее старый интерфейс. После этого параллельно все другие подсистемы корпорации постепенно переводятся на использование нового интерфейса реконструируемой подсистемы, а сама эта подсистема переделывается в соответствии с современными технологиями. В конце концов, когда сторонние подсистемы полностью готовы работать в новом интерфейсе, и процесс переделки унаследованной подсистемы завершен, она заменяется на вновь разработанный вариант.

2. Аналитический раздел

2.1 Объект Управления

Объект управления: Департамент технического управления и поддержки широкополосного доступа компании ОАО "Вымпелком"( Компания оказывает широкий спектр услуг корпоративным и частным клиентам, число основных услуг, предоставляемых компанией, входят телефонизация объектов и подключение к сети Интернета; мобильная телефония; предоставление услуг междугородней и международной телефонной связи; построение корпоративных сетей (как физических, так и виртуальных); предоставление услуг передачи данных; организация видеоконференций; IP-телефония; веб-хостинг и colocation (размещение серверов заказчиков в технической зоне ОАО «Вымпелком»); продажа универсальных карт «Телефон+Интернет»). Основной деятельности департамента технического управления и поддержки является помощь в настройке соединения с сетью, абоненту; тестирование клиентского оборудования; тестирование качества соединения с сетью и многое другое.(витос можеш сам еще дописать чо вы там делаете) СОТРИ!!!!

2.2 Описание Орг. Структуры объекта управления

Ниже представлена организационная структура управления департамента технического управления и поддержки широкополосного доступа.

Структура представляет собой вертикальную модель иерархической системы управления с элементами горизонтальной модели.

Модель описывает внутриорганизационные отношения и их участников в департаменте.

В компании используется огромное количество информационного, программного и технического обеспечения. Ниже представлены системы, которые используются для обработки обращений клиентов.

Help desk - Intranet система, доступ к которой осуществляется через браузер. HelpDesk решаtn задачи автоматизации процедур регистрации запросов, поступающих в подразделения IT, планирования и контроля работы персонала технической поддержки в крупных компаниях, контроля выполнения работ. Основными функциями систем являются:

Автоматизация регистрации и обработки заявок от пользователей, поступающих в службу технической поддержки, хранение и оперативное представление информации о составе оборудования, программного обеспечения пользователей, поддержка автоматического контроля правил выполнения заявок и т.п.

Автоматизация планирования выполнения работ, оповещения персонала технической поддержки о составе работ и времени их выполнения;

Поддержка процедур получения необходимой информации пользователями без привлечения специалистов службы технической поддержки;

Ведение отчетности о выполнении заявок и работ службой технической поддержки.

Atlant – еще одна helpdesk система, в которой ведется обработка заявок от клиентов, подключенных через канал Golden Telecom.

Customer Relationship Management System, CRM-система)— корпоративная информационная система, предназначенная для автоматизации CRM-стратегии компании, в частности, для повышения уровня продаж, оптимизации маркетинга и улучшения обслуживания клиентов путём сохранения информации о клиентах (контрагентах) и истории взаимоотношений с ними, установления и улучшения бизнес-процедур и последующего анализа результатов. Её основные принципы таковы:

Наличие единого хранилища информации, откуда в любой момент доступны все сведения о предыдущем и планируемом взаимодействии с клиентами.

Использование всех каналов взаимодействия. Постоянный анализ собранной информации о клиентах и подготовка данных для принятия соответствующих организационных решений.

При любом взаимодействии с клиентом по любому каналу, сотруднику компании доступна полная информация обо всех взаимоотношениях с этим клиентом и решение принимается на основе этой информации (информация о решении, в свою очередь, тоже сохраняется).

Классификация по функциональным возможностям:

Управление продажами (SFA - Sales Force Automation)

Управление маркетингом

Управление сервисом и Call-центры (системы по обработке жалоб от абонентов, фиксация и дальнейшая работа с обращениями клиентов)

Классификация по уровням обработки информации:

Оперативный — регистрация и оперативный доступ к первичной информации по событиям, компаниям, проектам, контактам[1], документам и т. д.

Аналитический — отчетность по первичным данным и, самое главное, — более глубокий анализ информации в различных разрезах (воронка продаж, анализ результатов маркетинговых мероприятий, анализ эффективности продаж в разрезе продуктов, сегментов клиентов, регионов и т. п.)

Коллаборационный (англ. collaboration — сотрудничество; совместные, согласованные действия) — уровень организации тесного взаимодействия с конечными потребителями, клиентами, вплоть до влияния клиента на внутренние процессы компании (опросы, для изменения качеств продукта или порядка обслуживания, web-страницы для отслеживания клиентами состояния заказа, уведомление по SMS о проведённых транзакциях по банковскому счету, возможность для клиента самостоятельно скомплектовать и заказать в online, к примеру, автомобиль или компьютер из доступных блоков и опций и др.)

2.4 Обеспечение служб, объекта управления:

Техническое: Рабочие места персонала обеспечены персональными компьютерами последних моделей, локальной сетью и выходом в интернет, также всей необходимой орг. Техникой и канцелярскими принадлежнастями.

Программное: Огромное количество информационного, программного и технического обеспечения, системы, которые используются для обработки обращений клиентов(Helpdesk , CRM, Atlant.), использование лицензионного ПО (т.к. антивирусы, брандмауэры, браузеры)

Обеспечение HR: Многоступенчатый отбор высококлассных специалистов, включающий собой тестирования, практические и теоретические, собеседования и проверки. Подбор специалистов высокого уровня необходим для обеспечения качественного уровня обслуживания.

3. Проектная часть

3.1 Постановка задачи

Постановка задачи: В компании более 700 000 клиентов, и с каждым днем их количество увеличивается. Для этих целей существует несколько подразделений, которые занимаются заведением и обработкой заявок на подключение их эскалацией в последующие отделы для обработки.

Задача : клиент (физическое лицо), с заявкой на подключение. Для обработки обращения используются следующие информационные системы: Helpdesk , CRM, Atlant.

3.2 Экономическая сущность задачи

Цель – Принять заявку клиента, на подключение доступа в интернет ; вход – заявка клиента, выход – готовый договор, доступ в интернет, акт выполненных работ, закрытый наряд ; задача не периодична и выполняется по мере необходимости.

3.3 Архитектура задачи

Для выполнения такого бизнес-процесса, как подключение клиента, необходимо участие абонентского отдела и персонала тех. Служб. Так же необходим клиент желающий подключиться к интернету и его заявка на подключение.

3.4 Описание метода решения задачи

Оператор регистрирует обращение, после чего переводит созданную заявку в соответствующий отдел (в данном случае в Кол центр для проведения первоначальной диагностики на возможность подключения). Далее, если подключение возможно выполнить создается наряд на проведения работ по подключению. Заключается договор, после чего технические службы инсталлируют соединение. Инженер ТП удаленно проверяет работоспособность и конфигурации оборудования через которое подключен клиент, и в случае неполадок или неправильных конфигураций сообщает в соответствующий отдел (техники, системные администраторы, поддержка POS-терминалов, поддержка специального ПО, поддержка специального оборудования (биллинг) и т.д.).

3.5 Информационное обеспечение подсистемы или задачи.

(ВИТОС тут опиши информационные системы: Helpdesk , CRM, Atlant., только не тоже самое что написано сверху…. Какоенить техническую информацию…. Чем больше писанины тем ей больше понравится)

3.6 Описание бизнес-процесса.

Задача решается по алгоритму бизнес-процесса показанному ниже на рисунке.

Модель бизнес-процесса по предоставлению широкополосного доступа к сети Интернет (в нотации IDEF0). Дано обоснование применения метода реинжиниринга бизнес-процессов по отношению к бизнес-процессам по предоставлению широкополосного доступа к сети Интернет и построение на базе модели улучшенных бизнес-процессов системы электронного документооборота.

С использованием методологии IDEF0 была описана модель бизнес-процесса по предоставлению услуги широкополосного доступа к сети Интернет.

Дополнительные схемы по данному объекту управления предоставлены в графе Приложения.

Заключение

Несмотря на разницу в технологиях, в реальности в любой сложной информационной системе на самом деле применяется некоторая смесь технологий.

Скорее всего в будущем эта тенденция не только сохранится, но и будет усиливаться. Можно ожидать появления инструментальных средств и методологий, которые будут явно поддерживать такой смешанный стиль разработки.

Конечно, будут появляться и принципиально новые технологии (хотя обычно в компьютерном мире все новое оказывается хорошо забытым старым). Какими будут эти новые технологии - покажет будущее.

**Приложения**

Бизнес-процесс (детализированная диаграмма по предоставлению широкополосного доступа в интернет).

С целью повышения эффективности производимых работ по подключению абонентов к услуге широкополосного доступа к сети Интернет, были проведены исследования и сделан вывод о неэффективности использования имеющейся технологии и о необходимости модернизации. Основой изменений является внедрение системы электронного документооборота по предоставлению широкополосного доступа к сети Интернет. Изменения были проведены как непосредственно в алгоритмах взаимодействия отделов предприятия, так и автоматизированных систем, использующихся в данных процессах.

Рисунок . Более Детализированная диаграмма бизнес-процесса по предоставлению услуги широкополосного доступа к сети Интернет

Концептуальная модель базы данных.

Концептуальная модель базы данных системы электронного документооборота по предоставлению широкополосного доступа к сети Интернет представлена на рисунке.

Рисунок . Концептуальная модель базы данных

Система электронного документооборота.

Система электронного документооборота на основе модели бизнес процесса по предоставлению услуги широкополосного доступа к сети Интернет. На рисунке представлена диаграмма развертывания системы электронного документооборота.

Рисунок . Диаграмма развертывания системы электронного документооборота

Диаграмма классов.

Диаграмма классов является типом диаграммы статической структуры. Она описывает структуру системы, показывая её классы, их атрибуты и операторы, а также взаимосвязи этих классов.

Диаграмма классов, построения домашней сети с широкополосным доступом в Интерент.

Рис. Диаграмма классов.

Дополнительные документы.

Описание бизнес-процесса

Сотрудник хочет перевестись в другой отдел. После проведения собеседования (если собеседование прошло успешно), сотрудник пишет заявление о переводе, которое передает начальнику подразделения, где в данный момент работает. Схема бизнес процесса ниже.