. Теоретическое описание баз данных

1.1 Определение и характеристики распределенных систем баз данных

База данных - это организованная структура, предназначенная для хранения информации. В современных базах данных хранятся не только данные, но и информация.

Под распределенной базой данных (Distributed DataBase - DDB) подразумевается база данных, которая включает в себя фрагменты из нескольких баз данных, располагающихся на различных узлах сети компьютеров и могут управляться различными СУБД. Распределенная база данных выглядит с точки зрения пользователей и прикладных программ как обычная локальная база данных. В этом смысле слово «распределенная» отражает способ организации базы данных, но не внешнюю ее характеристику («распределенность» базы данных невидима извне).

К. Дейтом были сформулированы 12 свойств типичной распределенной базы данных:

1. Локальная автономия (local autonomy) - управление данными на каждом из узлов распределенной системы выполняется локально. База данных, расположенная на одном из узлов, является неотъемлемым компонентом распределенной системы. Будучи фрагментом общего пространства данных, она, в то же время функционирует как полноценная локальная база данных; управление ею выполняется локально и независимо от других узлов системы.

2. Независимость узлов (no reliance on central site) - все узлы равноправны и независимы, а расположенные на них базы являются равноправными поставщиками данных в общее пространство данных. База данных на каждом из узлов самодостаточна - она включает полный собственный словарь данных и полностью защищена от несанкционированного доступа.

3. Непрерывные операции (continuous operation) - возможность непрерывного доступа к данным (известное «24 часа в сутки, семь дней в неделю») в рамках базы данных вне зависимости от их расположения и вне зависимости от операций, выполняемых на локальных узлах. Это качество можно выразить лозунгом «данные доступны всегда, а операции над ними выполняются непрерывно».

4. Прозрачность расположения (location independence) - полную прозрачность расположения данных. Пользователь, обращающийся к базе данных, ничего не должен знать о реальном, физическом размещении данных в узлах информационной системы. Все операции над данными выполняются без учета их местонахождения. Транспортировка запросов к базам данных осуществляется встроенными системными средствами.

5. Прозрачная фрагментация (fragmentation independence) - озможность распределенного (то есть на различных узлах) размещения данных, логически представляющих собой единое целое. Существует фрагментация двух типов: горизонтальная и вертикальная. Первая означает хранение строк одной таблицы на различных узлах (фактически, хранение строк одной логической таблицы в нескольких идентичных физических таблицах на различных узлах). Вторая означает распределение столбцов логической таблицы по нескольким узлам.

6. Прозрачное тиражирование (replication independence) - возможность переноса изменений между базами данных средствами, невидимыми пользователю распределенной системы. Данное свойство означает, что тиражирование возможно и достигается внутрисистемными средствами.

7. Обработка распределенных запросов (distributed query processing) - возможны несколько способов пересылки данных, позволяющих выполнить рассматриваемый запрос.

8. Обработка распределенных транзакций (distributed transaction processing) - возможность выполнения операций обновления распределенной базы данных (INSERT, UPDATE, DELETE), которые не разрушают целостность и согласованность данных. Эта цель достигается применением двухфазного протокола фиксации транзакций (two-phase commit protocol), ставшего фактическим стандартом обработки распределенных транзакций. Его применение гарантирует согласованное изменение данных на нескольких узлах в рамках распределенной (или, как ее еще называют, глобальной) транзакции.

9. Независимость от оборудования (hardware independence) - в качестве узлов распределенной системы могут выступать компьютеры любых моделей и производителей.

10. Независимость от операционных систем (operationg system independence) - многообразие операционных систем, управляющих узлами распределенной системы.

11. Прозрачность сети (network independence) - доступ к любым базам данных может осуществляться по сети. Спектр поддерживаемых конкретной СУБД сетевых протоколов не должен быть ограничением системы с распределенными базами данных. Данное качество формулируется максимально широко - в распределенной системе возможны любые сетевые протоколы.

12. Независимость от баз данных (database independence) - могут сосуществовать СУБД различных производителей, и возможны операции поиска и обновления в базах данных различных моделей и форматов.

Основой этих правил является то, что распределенная база данных должна восприниматься пользователем точно так же, как и привычная централизованная база данных.

Работу с распределенными базами данных обеспечивают распределенные системы управления баз данных. Распределенная система управления баз данных (РаСУБД) - комплекс программ, предназначенный для управления распределенной базой данных и позволяющий сделать распределенность информации «прозрачной» для конечного пользователя. Из определения РаСУБД следует, что распределенная база данных состоит из нескольких фрагментов, которые могут размещаться на нескольких компьютерах, расположенных в сети и к ней возможен параллельный доступ нескольких пользователей. Назначение обеспечения «прозрачности» состоит в том, чтобы распределенная система внешне вела себя точно так же, как и централизованная. Такое распределение данных позволяет, например, хранить в узле сети те данные, которые наиболее часто используются в этом узле. Такой подход облегчает и ускоряет работу с этими данными и оставляет возможность работать с остальными данными базы данных, хотя для доступа к ним требуется потратить некоторое время на передачу данных по сети.

Основной целью системы распределенных баз данных является обеспечение управляемого доступа и независимого обращения к данным, распределенным в сети ЭВМ. При управляемым доступом понимается степень безопасности, необходимая для защиты данных от неавторизованного доступа. Независимость обращения, или разделимость, позволяет пользователям получать доступ к данным через различные вычислительные средства.

Система управления распределенными базами данных обеспечивает средства интеграции локальных баз данных, располагающихся в некоторых узлах компьютерной сети, с тем, чтобы пользователь, работающий в любом узле сети, имел доступ ко всем этим базам данных как к единой базе данных [1]. При этом должны обеспечиваться простота использования системы, возможности автономного функционирования при нарушениях связности сети или при административных потребностях, высокая степень эффективности.

Для клиентских приложений распределенная база данных представляется не набором баз, а единым целым. Каждый фрагмент базы данных сохраняется на одном или нескольких компьютерах, которые соединены между собой линиями связи и каждый из них работает под управлением отдельной системой управления базой данных. Пользователь взаимодействует с распределенной базой данной через приложения. Приложения могут быть классифицированы как те, которые требуют доступа к данным на других узлах (локальные приложения), и те, которые требуют подобного доступа (глобальные приложения). В РаСУБД должно существовать хотя бы одно глобальное приложение, поэтому любая РаСУБД должна иметь следующие особенности:

набор логически связанных разделяемых данных;

сохраняемые данные разбиты на некоторое количество фрагментов;

между фрагментами может быть организована репликация данных;

фрагменты и их реплики распределены по различным узлам;

узлы связаны между собой сетевыми соединениями;

работа с данными на каждом узле управляется локальной СУБД.

СУБД на каждом узле способна поддерживать автономную работу локальных приложений.

1.2 Однородные и неоднородные базы данных

Распределенные системы часто строятся путем «интеграции» разнородных аппаратных и программных средств. Следовательно, должен быть сделан выбор между однородной и неоднородной вычислительными системами. Во-первых, необходимо определить связь сетевой и локальных СУБД. В случае однородных СУБД нет проблем ни с моделью данных, ни с языком запросов, ни с другими средствами, которые должны быть предоставлены пользователям. Все это совпадает с тем, что поддерживается локальной СУБД. Однако все же существует ряд вопросов, которые необходимо решить, например, должны ли абсолютно все пользователи взаимодействовать с сетевой СУБД или же те пользователи, которым нужны данные, хранящиеся в локальном узле, должны взаимодействовать непосредственно с локальной СУБД. Так как желательно, чтобы возможности, предоставляемые пользователям сетевой СУБД, были эквивалентны или по меньшей мере весьма сходны, вопрос касается главным образом архитектуры программного обеспечения и преимуществ общего программного интерфейса по сравнению с затратами на его поддержание. Если же распределенная база данных поддерживается неоднородными СУБД, то вопросы усложняются.

Использование неоднородных СУБД обычно является следствием формирования распределенной базы данных из ряда существовавших ранее автономных баз данных. Стоящая перед разработчиками цель -- достичь прозрачности доступа, что представляет собой нечто большее, чем простое обеспечение доступа к удаленным СУБД и их базам данных. Прозрачность означает, что пользователю либо неизвестно расположение данных, либо такие сведения для работы с базой данных ему не требуются. В системах с неоднородными СУБД такое возможно лишь в том случае, если локальная СУБД, управляющая данными, также «прозрачна», т. е. пользователь не обязан знать, какая локальная СУБД обслуживает его запрос. Это можно реализовать двумя принципиально отличными путями. Один путь состоит в том, чтобы дать пользователю возможность использовать в каждом узле пользовательский интерфейс, предоставляемый локальной СУБД. При этом имеющаяся схема должна быть расширена с целью включения данных, имевшихся в других узлах, но пользователи должны работать так, как если бы удаленные данные были добавлены в данную локальную базу данных.

Проблема заключается в том, чтобы программное обеспечение сетевой СУБД в каждом узле позволяло обращаться к данным любого другого узла независимо от модели данных и других факторов. При увеличении количества разнородных локальных СУБД количество типов преобразования схем быстро растет.

Использование единого для всей сети стандартного пользовательского интерфейса и стандартных внутренних форм представления запроса облегчает решение проблемы преобразования схем. При таком подходе все пользователи или по меньшей мере те из них, кому нужны данные из удаленных узлов, используют общий интерфейс, который может быть отличен от использующегося в любом локальной СУБД. Должна существовать одна схема сетевой базы данных, а не различные схемы в каждом узле, которые зависят от конкретной локальной СУБД.

Однородные распределенные системы баз данных имеют в своей основе один продукт СУБД, обычно с единственным языком баз данных (например, SQL с расширениями для управления распределенными данными). СУБД с поддержкой однородного распределения являются сильно связанными системами, их встроенные средства поиска данных и средства обработки запросов оптимизированы и настроены для достижения максимальной производительности и пропускной способности.

Существует множество вариантов однородной системы РБД. Так, на некотором узле может существовать одна глобально доступная "главная машина СУБД", с которой связаны компоненты для доступа к данным локальных баз данных, размещенные совместно с самими этими базами данных в пределах всей компании (или отдельного ее подразделения в зависимости от масштаба распределения). Более сложные модели могут допускать распределенность самой СУБД, когда каждый ее компонент на равных правах имеет доступ к данным любого другого узла. Однако относительно собственно управления данными мы имеем здесь идентичные модели хранения, структуры индексирования и форматы данных в рамках всей распределенной среды.

Противоположностью однородных систем распределенных БД являются неоднородные распределенные системы БД. Неоднородные системы включают два или более существенно различающихся продукта управления данными (например, реляционные СУБД от разных поставщиков, таких, как Oracle и Digital Equipment Corp., или СУБД одного поставщика, но функционирующие на разных платформах и использующие различные структуры баз данных, такие, как DB2 и SQL/DS компании IBM).

Неоднородные системы баз данных можно, в свою очередь, также подразделить на классы в широком диапазоне - от федеративных систем до различных типов систем мультибаз данных; существует и формальная таксономия неоднородных моделей.

Однородные распределенные системы баз данных обычно проектируются методом «сверху вниз»; неоднородные же, напротив, чаще всего строятся «снизу вверх» с целью создать общую среду управления над существовавшими ранее разрозненными информационными ресурсами.

Проектирование распределенных БД «сверху вниз» осуществляется в целом аналогично проектированию централизованных баз данных. В идеале оно проводится с помощью одной из формальных методологий, которые включают создание концептуальной модели базы данных, отображение ее в логическую модель данных и, наконец, создание и настройку специфических для конкретной СУБД структур (например, таблиц базы данных СУБД Oracle).

Однако при проектировании распределенной базы данных методом «сверху вниз» предполагается, что ее объекты не будут сосредоточены в одном месте, а распределятся по нескольким вычислительным системам.

В однородных системах все узлы используют один и тот же тип СУБД. В неоднородных системах на узлах могут функционировать различные типы СУБД, использующие разные модели данных. Однородные системы значительно проще проектировать и сопровождать, добавляя новые узлы к уже существующей распределенной системе и повышая производительность системы за счет параллельной обработки информации.

Неоднородные системы обычно возникают в тех случаях, когда узлы, уже эксплуатирующие свои собственные системы с базами данных, со временем интегрируются в распределенную систему. В неоднородных системах для организации взаимодействия между различными типами СУБД требуется обеспечить преобразование передаваемых сообщений, для чего каждый из узлов должен иметь возможность формулировать запросы на языке той СУБД, которая используется на их локальном узле или система должна взять на себя выполнение всех необходимых преобразований.

1.3 Дифференциальные файлы

Дифференциальный файл в базе данных аналогичен списку опечаток в книге, Вместо того чтобы печатать новое издание книги, всякий раз, когда требуется внести изменения в текст, издатель описывает исправление и указывает номер страницы и строки, куда следует поместить исправление, а затем вносит это описание в список исправлений, который помещает в каждую книгу. Издание, имеющее список исправлений, продается по более низкой цене. При использовании отредактированного варианта книги читатель, прежде чем начать чтение основного текста, должен просмотреть список опечаток. Увеличение времени доступа компенсируется таким образом снижением стоимости поддержания. Если количество изменений в тексте растет, то также растет список опечаток. Он может достичь такого размера, при котором становятся оправданными затраты на реорганизацию. При этом все изменения должны быть непосредственно включены в книгу, тем самым образуя новое издание.

Обновление больших баз данных ставит аналогичную задачу. Как и в случае с книгой, обычно наиболее простой и наименее дорогой путь состоит в накоплении изменений за период времени и в последующем перенесении их всех вместе во вновь созданную редакцию (поколение) базы данных. Гораздо более дорогой, основываясь на стоимости хранения, времени поддержания и суммарной сложности системы, является модификация базы данных всякий раз, когда выполняется транзакция обновления. В качестве компромиссного решения может быть использован дифференциальный файл, аналогичный списку опечаток, который осуществляет сбор и идентификацию будущих изменений записей. Предварительное обращение к дифференциальному файлу, являющееся первым шагом при выполнении операции выборки, -- эффективное средство доступа к самому последнему состоянию базы данных. Таким образом, путем увеличения времени доступа затраты на обновление базы дынных могут быть уменьшены. Когда дифференциальный файл достигает достаточно больших размеров, проводится реорганизация, в процессе которой все изменения, хранящиеся в дифференциальном файле, вносятся в базу данных, образуя тем самым ее новое поколение, А опустевший дифференциальный файл может снова начать накапливать изменении.

Концепция дифференциального файла для различных приложений и различных вариантов появлялась много раз. Применительно к распределенным системам баз данных описаны три способа дифференциального файла. Дифференциальная структура для ленточных систем разработана для того, чтобы исключить запись неизменяемых данных при последовательной пакетной обработке обновлений. Файл данных разбивается на одинаково упорядоченные подфайлы: большая совокупность записей, для которых разрешено только чтение, хранится на одной ленте, в то время как небольшая совокупность модифицируемых записей содержится на отдельной «ленте изменений». Для обновления файла данных обе ленты сливаются, при этом получается новая измененная лента. Неизменяемые записи с ленты, доступной только для чтения, никогда не записываются. Поэтому рекомендуется проводить реорганизацию файла данных после модификации половины всех записей.

В файловой системе с прямым доступом необходимо использовать принцип дифференциального файла для обработки изменений. Система обращается к записям через уникальный идентификатор, и любая ссылка на данные проходит через индекс базы данных, которая адресует все записи. Созданный однажды, главный файл данных никогда не модифицируется. Новые записи базы данных обращаются к индексу, но запоминаются в отдельной области переполнения. Все модификации записей данных трактуются как записи добавления. При этом создается новая копия записи и обновляется индекс для указания на область переполнения. Старая запись не уничтожается, а, наоборот, поддерживается как предшествующий образ, на который указывает новая запись, и свидетельствует о том, что новая запись в результате обновления увеличивается в размерах без нарушения размещения соседних записей.

Система с подобной структурой была разработана с целью обеспечения восстановления базы данных при внезапном отключении электрического питания, И в этом случае все обращения осуществляются через системный индекс и все модификации выделяются в файл изменений, называемый MODFILE. Каждая измененная запись указывает на свой предыдущий образ. При отключении электрического питании информация из журнала транзакций совместно с информацией из файла MODFILE используется для удаления незавершенных обновлений.

Всякий раз, когда запись обновляется одним из описанных в способов, механизм поиска записи (связанный ранее лишь с главным файлом) модифицируется таким образом, чтобы указывать на новую копию записи, которая запоминается в дифференциальном файле. Доступ к текущему значению идентифицированной записи независимо от того, находится ли она в главном или дифференциальном файле, осуществляется при помощи общего механизма поиска - системного индекса.

При наличии у каждой записи базы данных своего идентификатора поиск вначале всегда проводится в дифференциальном файле. Если запись там не найдена, выборка осуществляется из главного файла. Подразумевается, что каждый файл может иметь свой собственный механизм поиска. При этом индекс главного файла является неизменяемым и может быть быстро восстановлен в случае сбоя по копии. Изменяемая часть индекса перенесена в меньший и более легко восстанавливаемый индекс дифференциального файла.

Чтобы защитить главный файл и его механизм поиска от изменений, приходится идти на накладные расходы в виде поиска в дифференциальном файле при каждом обращении к записи. В том случае, когда оба файла могут быть размещены на отдельных устройствах и доступ к ним осуществляется через независимые каналы, поиск проводится параллельно и пользователи системы не чувствуют дополнительной задержки. Если же параллельный поиск невозможен, приходится ожидать увеличения среднего времени выборки на время произвольного доступа к вторичной памяти. При этом дополнительное время доступа может оказаться сравнительно большим, что значительно снижает производительность системы.

В тех случаях, когда значительное увеличение времени доступа недопустимо, предлагается применять модифицированную стратегию поиска, использующую предпоисковый фильтрующий алгоритм для уменьшения количества ненужных обращений к дифференциальному файлу .

Дифференциальная организация базы данных имеет ряд преимуществ. Пять из них связаны с целостностью базы данных. Они устанавливают тот факт, что при использовании дифференциального файла можно не только снизить стоимость копирования и скорость восстановления, но даже минимизировать вероятность серьезных потерь данных. Остальные три преимущества связаны с эксплуатацией. Дифференциальный файл может привести к повышению доступности данных и одновременно снизить затраты памяти и стоимость выборки.

Снижение стоимости создания дампа базы данных. Путем перезагрузки восстанавливается состояние базы данных, существовавшее в некоторый предыдущей момент времени. Затем в базу данных добавляется накопленный результат обработки всех транзакций обновления, выполняемых с момента последнего копирования базы данных. Частое копирование обеспечивает быстрое восстановление базы данных, но связано с большими системными издержками. Время, требующееся на копирование, пропорционально объему копируемых данных, поэтому применение дифференциального файла может значительно снизить стоимость восстановления большой базы данных, особенно, если доля записей, измененных с момента последнего копирования, невелика.

Возможность создания дампа приращениями. Для того, чтобы в любое время обеспечить полное восстановление базы данных, необходимо добавить записи дифференциального файла, созданные после последнего копирования базы данных, к текущему состоянию текущей копии файла. При каждом дампировании приращениями можно сохранять текущее состояние двоичного вектора дифференциального файла и индекс поиска.

Возможность дампирования и реорганизации параллельно с обновлением. Путем создания дампа дифференциального файла время, в течение которого запрещены запросы на обновление, может быть существенно снижено. После завершения создания дампа содержащиеся в нем записи должны быть включены в основной дифференциальный файл. Эта же идея дает возможность организовать реорганизацию параллельно с эксплуатацией. После завершения реорганизации происходит замена старого дифференциального файла.

Ускорение восстановления после потери данных. Имеется возможность проводить быстрое восстановление системы посредством отката неправильно обработанной или частично не завершенной транзакции.

Снижение риска безвозвратной потери данных. Дифференциальный файл концентрирует изменения на малом физическом пространстве и дает потенциальные преимущества: минимизируется критическая незащищенная область; критическая область может быть размещена на более надежном устройстве; небольшая критическая область может быть дублирована для достижения максимальной надежности.

Эффективная поддержка «файла памяти». Для того, чтобы исключить существенные издержки, связанные с применением программных средств, обеспечивающих управление многопользовательским доступом, многие системы накапливают изменения для их пакетной обработки после окончания.

Упрощение разработки программного обеспечения. В системе с дифференциальным файлом основной файл данных и связанный с ним индекс не подвергаются действию обновлений - это дает возможность использовать имеющиеся средства для разработки нового программного обеспечения.

Снижение стоимости хранения базы данных в будущем. Снижение стоимости, обеспечиваемое использование дифференциального файла, значительно увеличит возможную сферу применения автоматизированных информационных систем.

2. Распределение баз данных

2.1 Архитектура распределенных СУБД

В сегодняшнем быстро меняющемся компьютерном мире сосуществуют по крайней мере три основные идеологии: клиент-сервер, Web и распределенные объекты (DCOM, CORBA). Внутри каждого направления также существует большое количество решений и стандартов от разных производителей. Сегодняшняя ситуация вызывает очень большую озабоченность независимых разработчиков и потребителей: Какую технологию выбрать и что будет со мной и моим бизнесом, если я приму неправильное решение? При этом очевидно, что цена ошибки будет весьма высока, кроме того большие средства уже вложены в разработку и эксплуатацию уже существующих систем.

«Клиент-сервер» - это вид распределенной системы, в которой есть сервер, выполняющий запросы клиента, причем сервер и клиент общаются между собой с использованием того или иного протокола. Под клиентом понимается программа, использующая ресурсы, а под сервером программа, обслуживающая запросы клиентов на получение ресурсов определенного вида. Столь широкое определение включает в себя практически любую программную технологию, в которой участвуют больше одной программы, функции между которыми распределены асимметрично [18].

Существует два подхода к распределению обязанностей между сервером и клиентом при выполнении запросов к базам данных. При использовании технологии файлового сервера клиент делает запрос, сервер передает ему необходимые для выполнения запроса файлы, после чего клиент выбирает из этих файлов нужную информацию. В этом случае львиная доля работы ложится на клиента, по каналам связи передается большое количество данных, большая часть из которых на самом деле не появляется в ответе на запрос.

Технология «клиент-сервер» предусматривает, что отбор данных для ответа на запрос делается сервером, а клиенту передается только результат - те данные, которые были запрошены. На рисунке 5 показан процесс обмена информацией между сервером и клиентом при выполнении одного из запросов. При работе с файл-серверной версией вся ответственность за сохранность и целостность базы данных лежит на программе и сетевой операционной системе. Обработка всех данных происходит на рабочих местах, а сервер используется только как разделяемый накопитель. Каждый пользователь непосредственно использует информацию и вносит изменения в файлы данных и в индексные файлы [20].

Рисунок 5 - Сравнение технологий файлового сервера и «клиент-сервера».

При больших объемах данных и работе в многопользовательском режиме существенно снижается быстродействие - ведь чем больше пользователей, тем выше требования к разделению данных. Кроме того, может возникнуть повреждение баз данных. Например, в момент записи в файл может возникнуть сбой сети или авария питания. В этом случае, компьютер прерывает работу и база данных может оказаться поврежденной, а индексный файл - разрушенным. Переиндексация, которую необходимо провести после подобных сбоев, может длиться несколько часов [12].

Клиент-серверная версия позволяет обойти эти проблемы, так как вся работа с базой данных происходит на сервере, не проходит по проводам и не зависит от сбоев на рабочих станциях. Все запросы на запись в файл перехватываются сервером. В файл изменения вносятся только после того, как сервер получит сообщение о том, что корректировка файла завершена. Это исключает повреждение индексных файлов и существенно повышает быстродействие системы. распределенный база данные

Кроме высокого быстродействия и надежности, архитектура «клиент-сервер» дает много преимуществ и в части технического обеспечения. Во-первых, сервер оптимизирует выполнение функций обработки данных, что избавляет от необходимости оптимизации рабочих станций. Рабочая станция может быть укомплектована не очень быстрым процессором, и, тем не менее, сервер позволит быстро получить результаты обработки запроса. Во-вторых, поскольку рабочие станции не обрабатывают все промежуточные данные, существенно снижается нагрузка на сеть. Появляется возможность ведения журнала операций, в котором автоматически регистрируются все прошедшие транзакции что, в свою очередь, поможет быстрому восстановлению системы при аппаратных сбоях [21].

Различают двухуровневую и трехуровневую модель «клиент-сервер». На рисунке 6 представлена двухуровневая модель «клиент-сервер». Эта модель, возможно, наиболее общая, поскольку она подобна схеме разработки локальных баз данных. Многие системы «клиент-сервер», используемые сегодня, развились из существующих локальных приложений базы данных, которые хранят свои данные в файле на сервере. Перенос систем осуществляется для повышения эффективности работы, защищенности и надежности базы данных [3].

Рисунок 6 - Двухуровневая модель «клиент-сервер».

Преимущества модели:

делает возможным, в большинстве случаев, распределение функций вычислительной системы между несколькими независимыми компьютерами. Это позволяет упростить обслуживание вычислительной системы. В частности, замена, ремонт, модернизация или перемещение сервера не затрагивают клиентов. Все данные хранятся на сервере, который, как правило, защищён гораздо лучше большинства клиентов. На сервере проще обеспечить контроль полномочий, чтобы разрешать доступ к данным только клиентам с соответствующими правами доступа;

позволяет объединить различные клиенты. Использовать ресурсы одного сервера часто могут клиенты с разными аппаратными платформами, операционными системами и т. п. [24].

Недостатки модели:

неработоспособность сервера может сделать неработоспособной всю вычислительную сеть;

поддержка работы данной системы требует отдельного специалиста - системного администратора;

высокая стоимость оборудования [5].

В такой модели данные постоянно находятся на сервере, а клиентные приложения на своем компьютере. Бизнес-правила при этом могут располагаться на любом из компьютеров (или даже на обоих одновременно).

Трехуровневая архитектура клиент-сервер -- разновидность модели «клиент-сервер», в которой функция обработки данных вынесена на один или несколько отдельных серверов. Это позволяет разделить функции хранения, обработки и представления данных для более эффективного использования возможностей серверов и клиентов [24].

Достоинства:

масштабируемость;

конфигурируемость -- изолированность уровней друг от друга позволяет (при правильном развертывании архитектуры) быстро и простыми средствами переконфигурировать систему при возникновении сбоев или при плановом обслуживании на одном из уровней;

высокая безопасность;

высокая надёжность;

низкие требования к скорости канала (сети) между терминалами и сервером приложений;

низкие требования к производительности и техническим характеристикам терминалов, как следствие снижение их стоимости. Терминалом может выступать не только компьютер, но и, например, мобильный телефон [20].

Недостатки вытекают из достоинств. По сравнению c клиент-серверной или файл-серверной архитектурой можно выделить следующие недостатки трёхуровневой архитектуры:

более высокая сложность создания приложений;

сложнее в разворачивании и администрировании;

высокие требования к производительности серверов приложений и сервера базы данных, а, значит, и высокая стоимость серверного оборудования;

высокие требования к скорости канала (сети) между сервером базы данных и серверами приложений.

На рисунке 7 показана трехуровневая модель «клиент-сервер». Здесь клиент это пользовательский интерфейс к данным, а данные находятся на удаленном сервере. Клиентное приложение делает запросы для получения доступа или изменения данных через сервер. Если клиент, сервер и бизнес-правила распределены по отдельным компьютерам, разработчик может оптимизировать доступ к данным и поддерживать их целостность во всей системе.

Рисунок 7 - Трехуровневая модель «клиент-сервер».

Технология клиент-сервер -- это особый способ взаимодействия компьютеров в локальной сети, при котором один из компьютеров (сервер) предоставляет свои ресурсы другому компьютеру (клиенту). В соответствии с этим различают одноранговые сети и серверные сети [12].

При одноранговой архитектуре в сети отсутствуют выделенные серверы, каждая рабочая станция может выполнять функции клиента и сервера. В этом случае рабочая станция выделяет часть своих ресурсов в общее пользование всем рабочим станциям сети. Как правило, одноранговые сети создаются на базе одинаковых по мощности компьютеров. Одноранговые сети являются достаточно простыми в наладке и эксплуатации. В том случае, когда сеть состоит из небольшого числа компьютеров и ее основной функцией является обмен информацией между рабочими станциями, одноранговая архитектура является наиболее приемлемым решением [3].

Наличие распределенных данных и возможность изменения своих серверных ресурсов каждой рабочей станцией усложняет защиту информации от несанкционированного доступа, что является одним из недостатков одноранговых сетей. Понимая это, разработчики начинают уделять особое внимание вопросам защиты информации в одноранговых сетях. Другим недостатком одноранговых сетей является их более низкая производительность. Это объясняется тем, что сетевые ресурсы сосредоточены на рабочих станциях, которым приходится одновременно выполнять функции клиентов и серверов.

В серверных сетях осуществляется четкое разделение функций между компьютерами: одни их них постоянно являются клиентами, а другие -- серверами. Учитывая многообразие услуг, предоставляемых компьютерными сетями, существует несколько типов серверов, а именно: сетевой сервер, файловый сервер, сервер печати, почтовый сервер и др. Сетевой сервер представляет собой специализированный компьютер, ориентированный на выполнение основного объема вычислительных работ и функций по управлению компьютерной сетью. Этот сервер содержит ядро сетевой операционной системы, под управлением которой осуществляется работа всей локальной сети. Сетевой сервер обладает достаточно высоким быстродействием и большим объемом памяти. При подобной сетевой организации функции рабочих станций сводятся к вводу-выводу информации и обмену ею с сетевым сервером [18].

Термин файловый сервер относится к компьютеру, основной функцией которого является хранение, управление и передача файлов данных. Он не обрабатывает и не изменяет сохраняемые и передаваемые им файлы. Сервер может «не знать», является ли файл текстовым документом, графическим изображением или электронной таблицей. В общем случае на файловом сервере может даже отсутствовать клавиатура и монитор. Все изменения в файлах данных осуществляются с клиентских рабочих станций. Для этого клиенты считывают файлы данных с файлового сервера, осуществляют необходимые изменения данных и возвращают их обратно на файловый сервер. Подобная организация наиболее эффективна при работе большого количества пользователей с общей базой данных. В рамках больших сетей может одновременно использоваться несколько файловых серверов.

Сервер печати (принт-сервер) представляет собой печатающее устройство, которое с помощью сетевого адаптера подключается к передающей среде. Подобное сетевое печатающее устройство является самостоятельным и работает независимо от других сетевых устройств. Сервер печати обслуживает заявки на печать от всех серверов и рабочих станций. В качестве серверов печати используются специальные высокопроизводительные принтеры. При высокой интенсивности обмена данными с глобальными сетями в рамках локальных сетей выделяются почтовые серверы, с помощью которых обрабатываются сообщения электронной почты. Для эффективного взаимодействия с сетью Internet могут использоваться Web-серверы [12].

2.2 Стратегии распределения данных

Стратегия распределения данных по узлам сети ЭВМ могут классифицироваться в зависимости от количества узлов, содержащих данные, и наличия дублирования информации. Допустимые стратегии определяются архитектурой системы и программным обеспечением системы управления базой данных. Особенности реализации стратегий распределения данных определяются обычно в процессе проектирования базы данных.

Существует четыре альтернативные стратегии распределения данных:

Централизация (единственная копия базы данных, расположена в одном узле),

Расчленение (единственная копня базы данных, непересекающиеся подмножеств распределены по различным узлам),

Дублирование (несколько копий базы данных, в каждом узле располагается полная копня всех данных),

Смешанная (несколько копий подмножеств базы данных, в каждом узле может содержаться произвольный фрагмент базы данных).

Система управления распределенными базами данных, допускающая лишь централизованное распределение, является простейшей, а система, допускающая смешанное распределение данных, --наиболее сложной. Стратегии расчленения и дублирования являются в различной степени более сложными, чем централизованная. Стратегии расчленении предполагает наличие лишь одной копии базы данных, но при этом необходимо знать, какая часть базы данных расположена в каждом узле. Стратегия дублирования предполагает наличие в каждом узле полной копии базы данных, причем все копии должны обслуживаться согласованно для обеспечения их полноты и целостности.

Смешанная стратегия сочетает сложности двух других распределенных стратегии, приобретая при этом гибкость и достоинства обеих стратегий. Для системы управления распределенными базами данных может потребоваться следить за изменением состояний копий каждого подмножества базы данных, а также за размещением каждой копии.

Основным преимуществом централизованной базы данных (рис. 1) является простота. Все операции осуществляются под контролем единственного узла, все проблемы и действия полностью ясны. Так как в централизованных базах данных все данные располагаются в единственном узле, то наличие вторичной памяти в этом узле ограничивает возможный размер базы данных. Все запросы на выборку и обновление данных направляются в центральный узел со всеми сопутствующими затратами на стоимость связи и временную задержку. База может быть недоступной для удаленных пользователей при появлении ошибок связи и полностью выходит из строя при отказе центрального сервера.

Рис. 1 - Стратегия централизации распределения данных

При распределении данных на основе стратегии расчленения база данных распределяется по многим узлам сети, но существование копий отдельных частей базы данных не допускается. База данных разделяется на непересекающееся подмножества (логические фрагменты) и каждый логический фрагмент размещается в отдельном узле. Объем базы данных ограничивается объемом вторичной памяти, имеющейся во всей сети, анне в единственном узле. Эффективность стратегии расчленения тем выше, чем выше степень локализации ссылок, то есть чем больше число запросов пользователей реализуется в базах данных соответствующих локальных информационных систем (рис.2).

Рис. 2 - Стратегия расчленения распределения данных

Достоинствами данного метода являются увеличение объема базы данных; большинство запросов удовлетворяется локальными базами, что сокращает время ответа; увеличение доступности и надежности; снижение стоимости запросов на выборку и обновление по сравнению с централизованным распределением; система останется частично работоспособной, если выйдет из строя один сервер.

К недостаткам метода относится то, что часть удаленных запросов или транзакций могут потребовать доступ ко всем серверам, что увеличивает время ожидания и цену; необходимо иметь сведения о размещении данных в БД. Однако доступность и надежность увеличиваются. Расчлененные базы данных наиболее подходят к случаю совместного использования локальных и глобальных сетей ЭВМ.

При распределении данных с использованием стратегии дублирования (рис. 3) в каждом узле сети размещается полная копия базы данных, т.е. в каждом узле, в котором имеются данные, имеется вся база данных. Основное преимущество данной стратегии заключаются в высокой надежности, доступности и эффективности выборки. Недостатками стратегии дублирования являются повышенные требования к объему внешней памяти, усложнение корректировки баз, т.к. требуется синхронизация с целью согласования копий. К достоинствам стратегии относится то, что все запросы выполняются локально, благодаря чему обеспечивается быстрый доступ.

Рис. 3 - Стратегия дублирования распределения данных

Смешанная стратегия распределения данных (рис. 4) объединяет подходы, связанные с расчленением и дублированием данных с целью приобретения преимуществ, которыми они обладают. Эта стратегия подразделяет базы данных на логические фрагменты, как это делается в стратегии расчленения, но в дополнение к этому дает возможность иметь произвольное количество физических копий каждого фрагмента, называемых хранимыми фрагментами.

Ключевым преимуществом этой системы является гибкость, т.к. можно установить компромисс между объемом памяти, используемой в целом и в каждом отдельном узле, для обеспечения надежности и эффективности работы. В этой стратегии легко реализуется параллельная обработка, т.е. обслуживание распределенного запроса или транзакции. Недостатком стратегии остается проблема взаимозависимости факторов, влияющих на производительность системы, ее надежность, повышаются требования к памяти.

Рис. 4 - Стратегия смешанного распределения данных

2.3 Распределение сетевого справочника данных

Справочник сетевой базы данных может быть распределен по узлам сети согласно любой из четырех стратегий, применяющихся для самой базы данных: централизованной, расчлененной, дублированной или смешанной. Какая стратегия является наиболее подходящей, определяется обычно сочетанием стратегий распределения данных, используемым сетевой СУБД, и допущениями относительно требований (пользователей и приложений) к системам баз данных. Это означает, что обычно основная стратегия будет предоставляться сетевой СУБД и, следовательно, ее воздействия должны быть оценены до выбора именно сетевой СУБД. Многое из сказанного о распределении данных может быть перенесено и на распределения справочника. Например, наличие нескольких копий справочника приводит к необходимости согласования при его модификации.

Хотя теоретически можно пользоваться любой стратегией распределения справочника независимо от стратегии распределения данных, на практике используется лишь несколько сочетаний. Использование стратегии централизации при распределении справочника совместно со смешанной стратегией распределении данных обычно приводит к потеря многих преимуществ, присущих смешанной стратегией распределения данных. Необходимо отметить, что характеристики запросов к сетевому справочнику, такие, как отношение количества обновлений к количеству выборок (изменчивость), и ряд других важных при выборе стратегии распределения характеристик могут сильна различаться для сетевого справочника и собственно для базы данных, управляемой этим справочником. Запросы на выборку из справочника возникают при обработке любого запроса пользователя, что служит доводом в пользу расчлененного и дублированного справочника. При использовании смешанной стратегии требуется иметь справочник справочника, чтобы обеспечить возможность сетевой СУБД найти необходимую часть справочника. В системе SDD-1 такой справочник называется указателем справочника и дублируется в каждом узле. В общем сетевой справочник рассматривается как часть сетевой СУБД, а не как отдельный объект. Применяемая стратегия распределения справочника должна рассматриваться в свете стратегии распределения данных и целей функционирования базы данных.

Проектирование сетевого справочника может потребоваться уже при реализации базы данник, хотя при этом возможно возникновение большого количества ограничений по сравнению с количеством ограничений на этапе проектирования базы данных Некоторые СУБД могут потребовать размещения сетевого справочника аналогично размещению данных. То есть, сетевой справочник может быть разделен на логические фрагменты, а затем логические фрагменты могут быть размещены по узлам сети, возможно с использованием избыточности. Если справочник обрабатывается аналогично базе данных, то такие механизмы поддержания базы данных, как управление параллелизмом и безопасность, также применяются и для поддержания справочника. При этом необходимо еще раз оценить такие факторы, как надежность, и характеристики производительности.

Размещение справочников в распределенной системе исследовалось в Калифорнийском университете Лос-Анджелеса. Это исследование было направлено на анализ схем размещения справочника в различных ситуациях.

Авторы работы |304] рассматривали функцию справочника данных как связь между именами объектов и их размещением в системе распределенных данных. К решению этой проблемы существуют два подхода: использование полного имени доступа или функции каталогизации. При использовании полного имени доступа связь осуществляется путем включения поля, указывающего размещение распределенных данных непосредственно в имени каталога файла, что дает явное указание размещения. Функция каталогизации обеспечивает для распределенных разделяемых систем данных поддержание внутренних связей имя-расположение таким образом, что пользователю нет надобности знать расположение файлов.

2.4 Основы проектирования распределенной базы данных

При проектировании баз данных можно воспользоваться широко известными методами проектирования программного обеспечения. В частности, полностью применимо к структурам баз данных нисходящее проектирование с последовательными итерациями. На начальной стадии концептуальная модель, представляющая элементы данных и взаимосвязи предметной области, последовательно преобразуется в СУБД-ориентированную структуру базы данных. Процесс проектирования хорошо структурирован, так как каждый его этап завершается четко определенным результатом, а также потому, что допускает итеративное повторение предыдущих этапов в случае, если полученный результат не соответствует требованиям пользователей или системным ограничениям, либо если накладываются дополнительные требования. В общем случае это позволяет проектировщику пересматривать проектные решения с любого предыдущего этапа. На практике стоимость проектирования при этом значительно возрастает, особенно если проектные решения пересматриваются после того, как некоторые из них уже реализованы, полому итерации наиболее эффективны на этапах, предшествующих этапу реализации. Хотя итерация может значительно уменьшить общую стоимость разработки базы данных, она затрудняет достижение одной из целей методологии проектирования а именно воспроизводимости. Тем не менее на сегодняшний день итерация является наиболее необходимым и полезным средством, которое может применяться на любом этапе процесса проектирования базы данных. Тесно связана с процессом проектирования многошаговая методология экспертной оценки проекта или проектной экспертизы, включающая такие методы, как сквозной структурный контроль структуры базы данных, а также прикладного программного обеспечения. Проектная экспертиза может особенно широко применяться в системах баз данных с использованием стратегии, предложенной для более общих информационных систем. Цель экспертизы - обнаружить ошибки системного проектирования и исправить их на более ранних этапах жизненного цикла, чтобы уменьшить стоимость разработки системы.

Обычно проектная экспертиза проводится не менее четырех раз в течении жизненного цикла, а именно:

После анализа требований и проектирования информационной структуры, т.е. после концептуального проектирования.

После детального проектирования системы.

После реализации, но для начала эксплуатации системы.

После начала эксплуатации, когда уже собрана содержательная информация об эксплуатационных характеристик.

Средства проектирования и оценочные критерии

Каждый этап процесса проектирования характеризуется набором методов проектирования и критериями оценки альтернативных решений. Несмотря на то что методы проектирования могут носить аналитический, эвристический или процедурный характер, любой метод, который может быть реализован в виде программы, становится инструментальным средством проектирования и, как таковой, он практически не подвержен влиянию стиля проектирования.

Оценочные критерии, как средство проектирования, необходимы для выбора рациональном структуры базы данных среди нескольких альтернативных возможностей. Можно сказать, что большинство проблем и неудач при проектировании баз данных возникают из-за нечеткого представления того, что понимается под оптимальным проектированием баз данных, В настоящее время, а также в ближайшее будущее неопределенность при выборе критериев будет оставаться наиболее слабым звеном в проектировании баз данных.

Трудности в определении критериев выбора альтернативных решений вызваны в основном двумя факторами. Первая проблема заключается в том, что может быть построено практически бесконечное число различных структур баз данных, удовлетворяющих одному и тому же множеству системных требований. Критерии выбора должны позволять дифференцировать все имеющиеся в данный момент альтернативы. Вторая проблема состоит в том, что альтернативы трудно поддаются оценке, так как критерии обладают свойством чувствительности и время действия различных критериев различно.

Средства описания.

И спецификации требований, и структуры баз данных на всех этапах проектирования должны быть представлены в виде простой модели или комплекса связанных моделей, понятных как конечному пользователю данных, так и прикладному программисту, а также системным аналитикам и программистам

Существует три основных класса описательных средств, использующихся в методологии проектирования. Первым из них является язык описания данных ЯОД, входящий в состав СУБД и использующийся для описания конечного результата процесса проектирования реализации.

Ко второму классу относится описание исходной информации. Все средства сбора и анализа информации, используемые в настоящее время, подобны в том. что они обеспечивают форматы для спецификации как информации типа ISP, так и типа UP, а также выполняют основные проверки согласованности данных.

Тритий класс описательных средств используется для представления результатов промежуточных этапов и является промежуточным между ЯОД и описанием исходной информации.

Средства проектирования и оценочные критерии используются на всех стадиях разработки. При работе с количественными критериями, к которым относятся время отклика на запрос, стоимость обновления, стоимость памяти, время на создание, стоимость реорганизации, затруднение может вызывать противоречие критериев друг другу. В то же время существует много критериев оптимальности, являющихся неизмеримыми свойствами, трудно выразимыми в количественном представлении или в виде целевой функции.

К качественным критериям оценки базы данных относятся гибкость, адаптивность, доступность для новых пользователей, совместимость с другими системами, возможность конвертирования для использования в другой вычислительной среде, возможность восстановления, возможность расчленения и расширения структуры.

В распределенных системах баз данных логически целостная база данных может быть фрагментирована и широко распределена по сети с целью улучшения производительности системы. Фрагментация и распределение базы данных без внимательного централизованного планирования часто приводят в беспорядку и несогласованности при использовании баз данных.

Основные этапы последовательности проектирования распределенной базы данных заключаются в:

формулировании и анализе требований;

концептуальном проектировании;

проектировании реализации;

расчленении базы данных;

размещении базы данных;

физическом проектирование

На этапе формулирования и анализе требований устанавливаются цели организации, определяются специфичные требования к базе данных, вытекающие из этих целей или сформулированные непосредственно управляющим персоналом организации. Эти требования документируются в форме доступной как конечному пользователю, так и проектировщику базы данных. Специфичные цели и требования к базе данных необходимо определить на самом высоком уровне организации. Собранные и документированные требования должны включать ограничения, обуславливающие безопасность, надежность, уровень достигнутой технологии, а также политические и бюрократические ограничения.

Этап концептуального проектирования связан с описанием и синтезом разнообразных информационных требований пользователей в первоначальный проект баз данных. Результатом этого этапа является высокоуровневое представление информационных требований, например, такое как диаграмма «сущность-связь». Основу этой диаграммы составляет набор сущностей, который представляет или моделирует определенную совокупность сведений, специфицированную в требованиях. Связи между сущностями отображают функциональные аспекты информации, представленной сущностями.

Существует несколько подходов к построению диаграмм типа «сущность-связь». Общим для всех подходов является набор из четырех основных проектных решений или шагов:

1. Определение сущностей.

2. Определение атрибутов сущностей.

3. Идентификации ключевых атрибута сущностей,

4. Определение связей между сущностями.

После построения первоначальной информационной структуры следует ее уточнение и совершенствование. Главной целью этапа проектирования реализации является создание СУБД-ориентированной схемы с использованием в качестве исходных данных результатов концептуального проектирования и требований обработки (UР-информации).

Вначале анализируется содержание требований обработки данных. Формат локальных информационных структур, подлежащих обработке, соответствует формату первоначальной структуры, полученной на этапе концептуального проектирования. После того как представлены все виды обработки, первоначальная структура может быть объединена со всеми локальными структурами в новую информационную структуру. Затем, используя знания, полученные в процессе пересмотра и объединения информационных структур, учитывая связи обрабатываемых данных и характеристики типов записей, допускаемых СУБД, можно сформировать предварительные типы записей.

Построенная таким образом логическая структура базы данных (схема) может быть оценена количественно с помощью таких характеристик, как число обращении к логическим записям, объем обрабатываемых в каждом приложении данных, общий объем хранимых данных.

Этап расчленения базы данных связан с расчленением глобальной базы данных и синтезом различных приложений на основе модели. Существуют три класса выходных данных этапа расчленения: 1) совокупность расчлененных частей базы данных (разделов), 2) размер каждого раздела. 3) модели и частоты использования приложений. На этом этапе проектирования исходная глобальная база данных расчленяется на множество подфайлов, которые содержат в точности все сведения, имевшиеся в глобальной базе данных. Каждый подфайл в расчлененной базе данных выбирается как неделимая единица размещения данных. Далее проводится анализ того, как приложения базы данных используют возможные разделы базы данных. Связь между разделом базы данных и приложениями характеризуется идентификатором типа приложения, идентификатором узла сети, создающего приложение, частотой использования приложения и моделью приложения.

Размещение распределенной базы данных является многовариантной задачей. Количество возможных вариантов реализации расчлененной или смешанной базы данных велико. Для выбора наиболее подходящей стратегии распределения данных, необходимо еще до выбора СУБД провести оценку пользовательских и системных требований.

Физическое проектирование базы данных заключается в расширении ее логической модели такими характеристиками, которые необходимы, во-первых, для определения способов физического хранения и использования базы данных и, во-вторых, для определения объемов памяти, требуемой для всей системы и для оценки эффективности обработки. Подобные характеристики касаются того, как и где хранить данные, как их можно найти и использовать.

Следует отметить, что проектирование распределенных баз данных является сложным процессом, в реализации которого можно выделить четыре основные проблемы:

1) проблему дезагрегации, состоящую в необходимости рационального, в соответствии с системой расчетов (решаемых задач), распределения учетной информации по уровням обработки и участкам учета с обеспечением их взаимосвязи;

2) проблему, связанную с созданием инфологической структуры информационного фонда распределенной базы данных, ориентированного на решение всего комплекса задач избранной системы расчетов;

3) технологическую проблему, состоящую в удовлетворении требований рационализации вычислительного процесса на основе распределенной базы данных и распределенного комплекса технических средств;

4) организационно-правовую проблему, состоящую в обеспечении защиты данных н соблюдении юридических норм доступа к базам данных, их заполнения, изменения и уничтожения.

Заключение

Ввиду того, что информатика является одной из фундаментальных отраслей научного знания, стремительно развивающейся и постоянно расширяющейся областью практической деятельности человека, преподавание информатики в школе приобретает в настоящее время огромное значение, при этом методика преподавания информатики постоянно совершенствуется.

Так как информатика представляет собой самую динамическую область знаний, то она требует непрерывного обновления подходов, форм, методов и методических технологий к ее преподаванию.

Под методами обучения следует понимать способы обучающей работы учителя и организации учебно-познавательной деятельности учащихся по решению различных дидактических задач, направленных на овладение изучаемым материалом. А методика преподавания линий информации и информационных процессов в базовом курсе информатики является неотрывной частью общей методики преподавания информатики.

В приведенных планах-конспектах наиболее достаточно раскрывается тема «Информация. Информационные процессы и системы» и в процессе изучения информатики эта тема играет значительную роль. Поэтому следует уделять данной теме огромное внимание. Незнание данной темы приводит к потере смысла изучения дисциплины «Информатика и ИКТ» в дальнейшем.