Уравнение правильной базы

Фленов Михаил http://www.vr-online.ru

Какие бывают базы данных? Нет, я не имею ввиду фирмы производители, я говорю о них в общем смысле. В большинстве случаев, знания программистов останавливаются на двух типах: локальная и клиент серверная. В первом случае получается шампунь все в одном. Во втором мы разделяем данные и клиентское приложение и получаем два уровня. Но уже достаточно давно существует выделение третьего уровня, и именно трехуровневую модель все обходят стороной, боясь ее сложности. Давайте рассмотрим каждую модель в отдельности и увидим их преимущества и недостатки.

Локальная база

Самая простая база данных – локальная. В этом случае база и программа расположены на одном компьютере. Соединение с файлом базы данных происходит через спец драйвер или напрямую. Даже если подключение происходит через драйвер, то он умеет только обрабатывать простые запросы SQL стандарта 92-го года и предоставлять данные программе или сохранять изменения в таблице. Все остальные манипуляции могут выполняться только программой. Таким образом, логика, данные и приложение работают как единое целое, и не могут быть разделены.

Работа локальной базы данных

Яркими и наиболее распространенными представителями такого рода баз являются Dbase (файлы с расширением dbf), Paradox (расширение db) и Access (расширение mdb). Форматы Dbase и Paradox это даже не базы данных, а таблицы, потому что в одном файле может храниться только одна таблица данных. Индексы, ускоряющие поиск и осуществляющие сортировку находятся в отдельных файлах. Таким образом, одна база данных может состоять из множества файлов и это иногда приводит к определенным проблемам при поставке приложения конечному юзеру.

Файлы Access являются гибридом таблиц и баз данных. Здесь уже все таблицы и индексы хранятся в одном файле, что намного удобнее в управлении. К тому же среда управления базами Access наиболее удобна и доступна в любом офисном пакете от MS. В остальном, MS Access обладает теми же недостатками, что и остальные представители этого сословия.

Самый главный недостаток локальных баз данных, как говорит юморист М. Задорнов – «они тупые». Да, да. Качество и скорость доступа напрямую зависит от драйвера. Большинство из них не имели оптимизаторов SQL запросов, и отсутствовало какое-либо кэшировании. Возможности железа использовались минимально, поэтому на больших базах запросы выполняются сверх медленно.

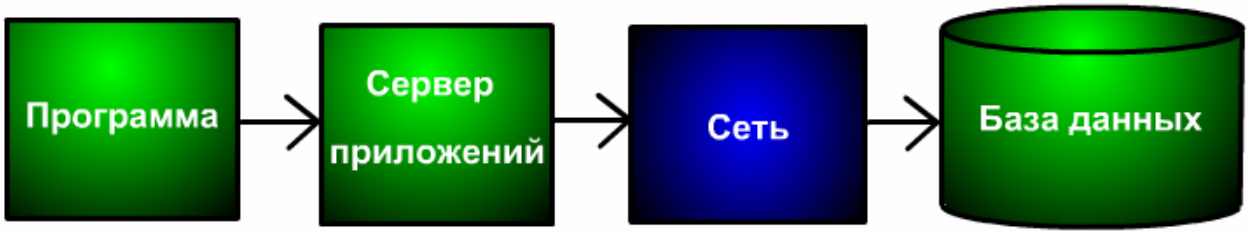
Таблицы Dbase и Paradox были разработаны слишком давно и самое слабое звено это индексы. В этих таблицах нет транзакций и соответствующего журнала. После добавления новой записи, если драйвер не успел обработать изменения в индексах, и произошла ошибка, пропал свет или зависон, то индекс рушится и для восстановления приходится использовать спец утилиты или переформировывать индексы. В базах Access у меня таких проблем не было, потому что тут индексы более защищены.

Что такое разрушенный индекс? Индекс – это колонка, в которой все значения строк обязательно уникальны. Чаще всего для этих целей используется простой счетчик. Допустим, что пользователь добавил запись и счетчик присвоил ей значение 195, но само значение счетчика не изменилось. При добавлении следующей записи счетчик снова пытается втулить нам число 195, но так как такая запись уже есть, происходит ошибка. Это и есть нарушение индекса, и лечить его достаточно просто – переформировать индекс, но нудно.

Сетевая база данных

Почему локальные базы называют локальными? Да потому что с данными работает только один пользователь, а также база данных и программа находятся на одном компьютере. В случае с небольшими проектами, это нормальная ситуация, но для больших объемов данных один оператор не справится с задачей и нужно, чтобы несколько человек могли работать с общими данными.

Решением этой проблемы стали сетевые базы данных. В принципе, это те же локальные базы, только выложены они на сетевой диск сервера (это может быть простой файловый сервер или компьютер с шарами), и несколько клиентов обращаются к одной базе по сети.



Сетевая модель доступа к данным

Давайте посмотрим, как происходит обращение к базе данных. Программа и драйвер находятся на клиенте, а данные находятся на сервере или просто на удаленном компьютере. Теперь подумаем, как программа получает данные. Клиент передает драйверу SQL запрос, который должен быть выполнен, но данные ведь находятся удаленно. Чтобы отработать запрос, вся нужная таблица (в случае с Access вся база данных, потому что все в одном файле) выкачивается на компьютер клиента, где драйвер обрабатывает данные.

Я бы убил того, кто придумал такую технологию, потому что это самое настоящее издевательство над системой. Представляешь, что будет, если надо выполнить запрос на базе данных в 1 гига с телефонным соединением в 34кб/с? Это то же самое, что заставить ЮКОС добывать нефть через трубочку для молочных коктейлей.

А ведь некоторые российские компании (не будет тыкать пальцем) тулили нам сетевые решения на основе dbf файлов в области бухгалтерии, делопроизводства и экономики. Это же издевательство. Меня несколько раз просили восстановить умершие базы складской программы, после того, как встроенные в программу средства не справлялись с задачей.

Но страшнее всего начали вести себя индексы. У таблиц Paradox, если они находились на расшаренном диске Win95, мне приходилось ремонтировать индексы как минимум раз в неделю. Но когда я убрал файлы базы данных на сетевой диск сервера NetWare 3.11 (это был где-то 1998-й год), как сразу проблемы с нарушением индексации исчезли. Просто это действительно сервер, а не корявый Windows 9x.

При сетевом соединении многопользование получилось не полное. Изменения одного пользователя небыли видны другим, пока они не перезапустят программу или не переконнектятся. Почему? Да потому что именно в момент коннекта прога сосет все

данные с сетевого диска. Потом мы работаем как бы с локальной версией и чужие изменения не видны.

Клиент сервер

Лохонувшись с сетевыми базами, монотонную модель наконец-то решили разделить на два уровня – приложение и база данных. Теперь база данных – это не просто таблица с данными, а целый движок, в задачи которого входит не только хранение данных, но и обработка запросов.

Доступ к данным по технологии Клиент-Сервер

В технологии клиент сервер, драйвер уже изменил свое назначение, и теперь он уже должен только знать, как подключится к серверу и передать ему запрос. Остальное уже ложится на плечи сервера. Такая технология намного сокращает трафик, особенно, при хорошем программировании. Допустим, что юзеру нужно увидеть, все данные, в которых имя определенной колонки содержит слова на букву «А». Клиенту достаточно направить серверу всего лишь такой текст:

```
SELECT *
FROM Имя таблицы
WHERE Колонка LIKE 'A%'
```

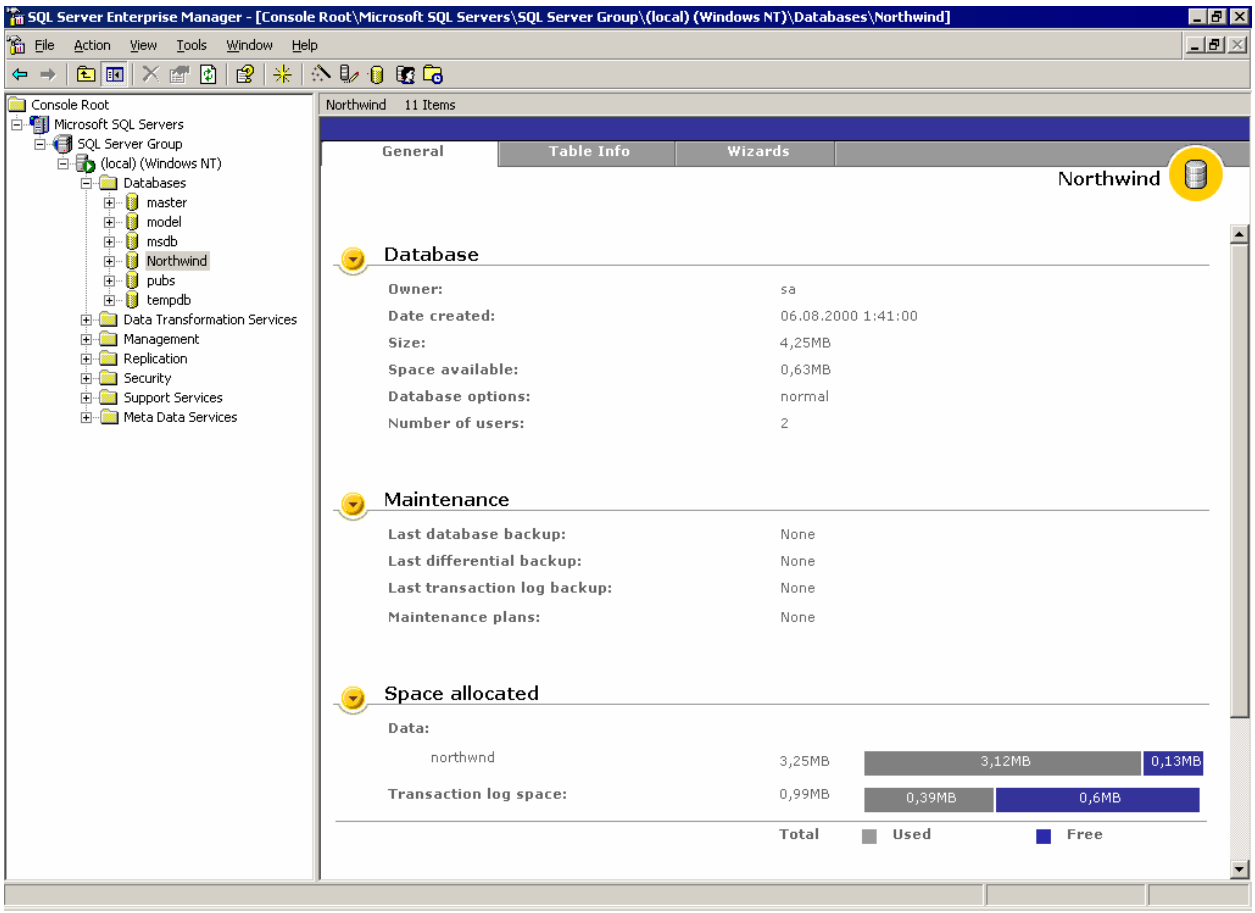
Я думаю, что не надо даже считать, сколько кило занимает этот текст, и как долго он будет отправляться по сети. Даже через медный провод с железом на 2400бод все произойдет практически мгновенно.

Сервер базы данных, получив запрос, разбирает его и придумывает для себя оптимальный план выполнения, в данном случае поиска нужных строк.

Получив нужные данные, сервер возвращает только их и ничего больше. Таким образом, клиент в любой момент может запросить у сервера нужные данные и нет необходимости гонять по сети всю базу данных. При хорошо построенном приложении и оптимальных запросах, клиент сможет работать с базой данных любого размера даже через модем в 56 кбит/с. Неплохо? Я тоже сказал: «неужели мозги у особо умных заработали в нужном направлении». Главное, запрашивать только то, что нужно и маленькими кусками и все будет в ажуре.

Особенности клиент-сервера

Возможности клиент серверных баз данных зависят от производителя. Самые простые возможности предоставляют такие базы как My SQL. В них сервер имеет встроенный движок обработки запросов и основные возможности по обеспечению безопасности, распределении прав.



Цент управления базами данных MS SQL Server

В более солидных клиент-серверных базах (MS SQL Server, Oracle и т.д.) есть следующие дополнительные возможности:

1. Вьюшки – более подробное о них мы поговорим в статье по безопасности;

2. Триггеры – это как бы функции, которые могут вызываться на определенные события (вставка, изменение и удаление данных). В этих функциях может производиться какая-то логика по обеспечению целостности данных

3. Репликация – объединение баз данных. Допустим, что у фирмы есть два офиса и в каждом из них своя база. Настроив репликацию, обе базы могут автоматически сливаться в одну в главном офисе или обмениваться изменениями по расписанию.

4. Хранимые процедуры и функции, которые выполняются на сервере по мизерному запросу клиента. Они могут содержать целые подпрограммы с логикой, которые будут выполнять какие-либо действия. Для написания таких программ используется уже не просто язык SQL, а его расширение – Transact-SQL (для MS баз) и PL/SQL (для Oracle и др.).

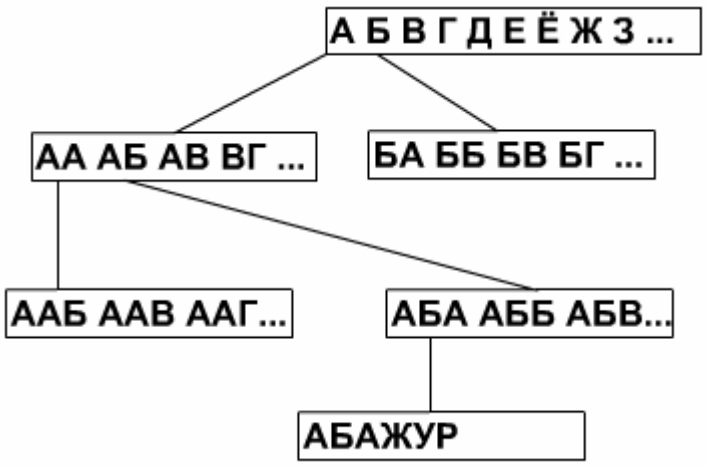
Список возможностей зависит от конкретной базы данных, ее навороченности и может быть больше или меньше.

Индексы на сервере

За счет наличия в серверных базах данных управления транзакциями, про проблемы с индексами можно забыть. Допустим, что пользователь добавил запись. В этот момент начинается транзакция (не явная), в течение которой происходят все необходимые действия по сохранению данных. Если что-то пошло неправильно и сохранение не прошло до конца, то все изменения откатываются, и ничего в работе сервера не нарушается.

Транзакции могут быть и явными, когда программист сам указывает, где начало и конец и в них может выполняться несколько операций изменения или добавления данных. В этом случае сервер в случае ошибки в указанном блоке откатит любые изменения всех операций, сделанные во время выполнения явной транзакции.

В локальных базах данных индексы хранятся линейно. Это как колонка из упорядоченных данных, и для строк, это то же самое, что выстроить все слова по алфавиту. Конечно же, такой индекс упрощает поиск. Когда происходит сканирование по индексу и программа видит, что уже пошло слово больше, чем задано в условии поиска, сканирование может прекращаться и не придется просматривать всю базу данных. Например, вам надо найти слово «Абажур». Оно будет где-то в начале и чтобы его найти, нужно просканировать всего лишь начало таблицы, не дальше, чем все слова на букву А. За счет того, что данные упорядочены, мы можем быть уверенными, что все остальные слова будут на буквы Б, В и т.д.



Древообразные индексы

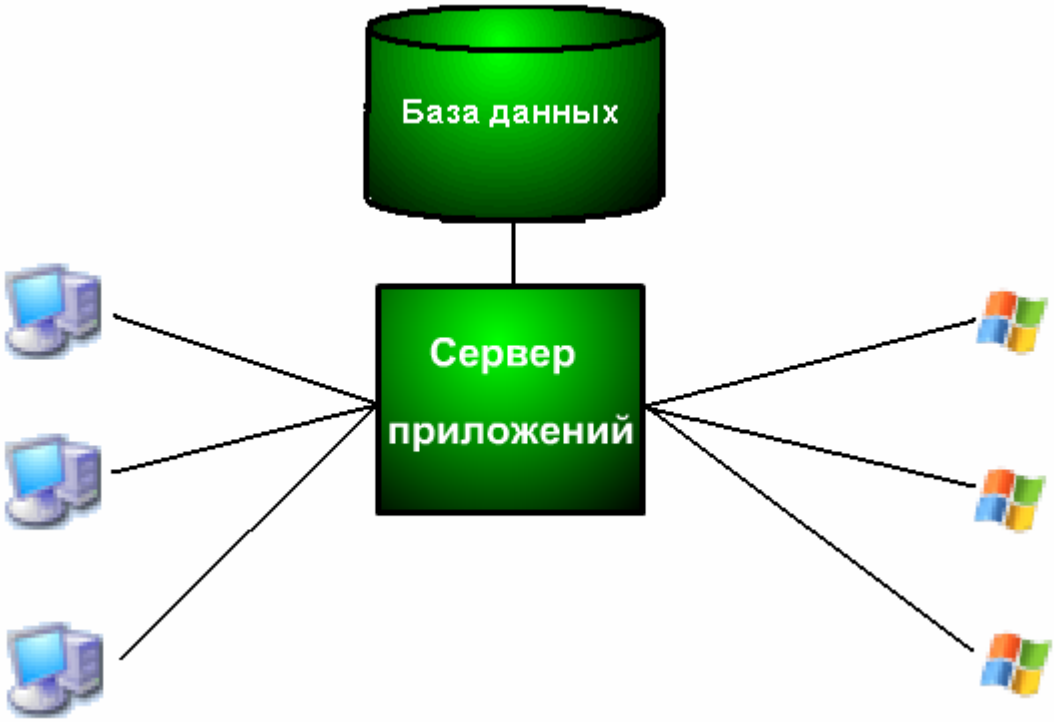
В случае с серверной базой, индексы чаще всего (зависит от базы и типа индекса) хранятся немного по-другому – в виде дерева. Сколько слов надо проверить для поиска слова Якорь в базе данных при линейном индексе? Да практически все :(. При древовидном хранении индекса не более чем для слова «Абажур». Для пояснения древообразного индекса рассмотрим классическую задачу (в реальности все немного сложнее, но идея такая же). В самом верху дерева хранятся алфавит. Программа находит букву А, и спускается на уровень ниже. Здесь она находит все слова на буквы АБ и двигается еще ниже. И так пока не найдется нужное слово.

Таким образом, даже если нужное слово находится в самом конце, его поиск будет не намного дольше, чем поиск слова из начала таблицы.

Третий уровень

Многие программисты, которых я знаю, способны работать только с двух уровневой моделью, т.е. клиент-серверными приложениями. Нет, это не потому, что они больше ничего не знают, просто не видят преимуществ 3-х уровневой модели. К тому же не хотят мучиться с лишними проблемами, а ведь будущем, во время сопровождения программ, три уровня спасают от лишних болезней анального отверстия.

Я работал на одной фирме (не будем в нее тыкать вилами), у которой было несколько офисов по России и в каждом из них парк компьютеров из 20-30 штук. В московском офисе эта цифра превышала сотню. Корпоративные программы обновлялись каждые две недели (вносились изменения, добавления и т.д.). Бедные админы в момент обновлений работали по субботам, чтобы пропатчить софт на каждой машине и убедится в функциональности. Как решить эту проблему?



Трехуровневая система

Самое простое – использовать трех уровневую систему: клиент, сервер логики (умники любят выражаться «бизнес логика») и сервер приложения. В такой системе вся логика собрана в сервере приложений. Если что-то изменилось в базе данных или в логике обработки данных, достаточно обновить его и все клиенты будут работать по-новому без каких-либо патчей.

Преимущество такой системы состоит еще и в том, что на клиентских машинах не нужно держать драйвера доступа к каким либо базам. Клиенты должны только знать, где находится сервер приложений, уметь к нему подключится и правильно отобразить данные.

Представим себе классическую задачу – появление новой версии базы данных или переход на базу качественно более нового уровня. Ну не хватает уже возможностей MySQL и захотелось получить всю мощь от Oracle!!! Для этого переустанавливается сервер баз данных, изменяется сервер приложений на подключение к новой базе и клиенты готовы к работе. Их обновлять не надо!!!

Но самое интересное, клиентская программа может быть какой угодно. Можно написать сценарии, которые позволят работать с сервером приложении прямо из браузера. В этом случае с базой смогут работать пользователи на любой платформе (Windows, Linux и т.д.).

Логика

Не смотря на наличие сервера приложений, нет смысла засовывать в него всю логику обработки данных. Если вы используете мощную базу данных, которая поддерживает хранимые процедуры и функции, то лучше переложить часть логики на сервер базы. В этом случае, внесенные изменения в хранимый код вступают в силу моментально и не надо даже обновлять сервер приложений.

Если в сети не так уж и много компьютеров (не более 20), и достаточно мощный сервер, то можно сервер приложений и базу данных расположить на одном физическом сервере. В этом случае обмен данными между сервером приложений и базой будет

происходит внутри одного компьютера, а не по сети, что может существенно снизить нагрузку на сетевое оборудование.

Сервер приложений и база данных находятся на раздельных серверах

Допустим, что сервер приложений и база данных находятся на разных серверах. Результаты запросов будут сначала идти через коммутатор от базы данных к серверу приложений, а затем через тот же коммутатор к компьютерам клиентов. Таким образом, по сети дважды пролетают одни и те же данные. Чтобы от этого избавится, я чаще всего объединяю в одном физическом сервере логику и данные.

Итого

Что же выбрать для своего проекта? Все очень просто. Если ты пишешь базу, с которой будет работать одновременно только один человек, то однозначный выбор – локальная база. Я больше всего люблю MS Access за его надежность и за то, что драйвера доступа к этой базе есть на всех компах (особенно, если там установлен MS Office) и их не надо тянуть с инсталлятором.

Если с базой будет работать хотя бы два человека, то не надо выдумывать сетевые коннекты, а лучше воспользоваться клиент-серверной технологией. Она избавит сеть от лишнего трафика, более надежна при многопользовательской работе и дает максимальное количество возможностей.

Если количество пользователей растет слишком сильно и появились проблемы с обновлением системы, то лучшим выходом будет переход на трех-уровневую систему. Это немного сложнее в разработке, зато намного лучше во время сопровождения.

Делай правильный выбор технологии, иначе впоследствии придется долго мучаться с переделками.

Не все мощные базы данных являются платными. Например, Interbase от дяди Бормана не только бесплатен, но и с открытым кодом.

При работе с 3-х уровневыми базами кэширование обновлений обязательно, поэтому метод Post запоминает данные локально, а ApplyUpdates загружает изменения на сервер.

В качестве хранилища для 3-х уровневой системы можно использовать и локальные таблицы (dbf, paradox), но я рекомендую юзать серверные базы. Они дадут максимальную мощь.

Дополнительную инфу по базам можно найти на сайтах www.sql.ru, http://delphi.mastak.ru/ или www.vr-online.ru.