**Планирование дискового пространства для баз данных**

Задумываетесь ли вы о чем-то, когда создаете новую базу данных? Я думаю, что большинство скажет, что нет (если будет отвечать честно), потому что все мы используем параметры по умолчанию, а ведь они далеки от оптимальных. Одних только дисковых настроек целое море, а ведь они реально позволяют повысить надежность и производительность системы.

Нет, мы не будем говорить, что для надежности данных в файловой системе нужно использовать NTFS, хотя имена эта файловая система позволяет MS SQL Server использовать диск максимально эффективно. Нас ждет более интересный и конкретный разговор.

Если не хватает ресурсов и что-то начинает работать медленно, то первое, что приходит в голову – наращивание ресурсов и апгрейд. Но ведь это необходимо далеко не всегда. Можно же обойтись и тюнингом, хотя этим нужно заниматься еще до того, как сервер начал тормозить, а на этапе проектирования и установки. Не надо дожидаться, когда грянет гром, чтобы креститься, ведь все можно сделать заранее.

Оптимизация – это процесс комплексный и чаще всего касается не только определенной программы (в нашем случае базы данных), но и ОС и даже железа. Несмотря на то, что основной упор будет делаться именно на базы данных, полностью абстрагироваться от окружающего мира не удастся.

**Архитектура данных**

Сервер SQL хранит, читает и записывает данные блоками по 8кб, эти блоки называются страницами. База данных может хранить 128 страниц на мегабайт (1 мегабайт или 1048576 байт разделить 8 килобайт или 8192 байт). Все страницы хранятся в пространстве. Пространство – это 8 последовательных страниц, или 64кб. Получается, что в одном мегабайте находиться 16 пространств.

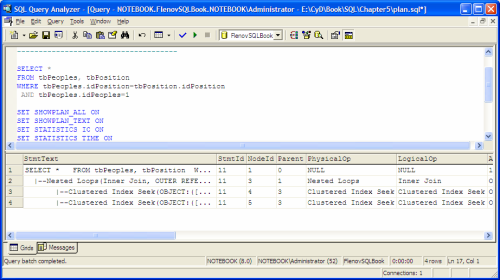
Страницы и пространства это основа структуры физической базы данных SQL Server. Сервер MS SQL использует различные типы страниц, некоторые из которых следят за выделенным пространством, а некоторые содержат пользовательские данные и индексы. Страницы, которые отслеживают выделенное пространство, содержат плотно сжатую информацию. Это позволяет MS SQL Server эффективно помещать их в память для легкого просмотра.

Сервер SQL использует два типа пространств:

1. Пространства, которые хранят страницы от двух и более объектов, называемые смешанными. Каждая таблицы начинается как смешанное пространство. Вы используете смешанное пространство главным образом для страниц, которые хранят пространство и содержат маленькие объекты.

2. Пространства, которые имеют все 8 страниц выделенных одному объекту, называемый однородным пространством. Они используются, когда таблице или индексу надо более 64 кб пространства.

Первое пространство для каждого файла является смешанным и содержит страницы заголовка файла, следующие по три выделенные страницы. Сервер выделяет эти смешанные пространства, когда вы создаете основной файл данных и использует эти страницы для своих внутренних задач. Страница заголовка файла – содержит атрибуты файла, такие как имя базы данных, которая хранится в файле, файловая группа, минимальный размер, размер приращения. Это первая страница в каждом файле (Страница 0).

  
План выполнения запроса в SQL Query Analyzer

Страница свободного пространства (**PFS**) – это выделенная страница, содержащая информацию о свободном пространстве доступном в файле. Эта информация хранится в странице 1. Каждая такая страница может простираться на 8000 смежных страниц, что приблизительно 64мб данных.

Журнал транзакций захватывает всю необходимую информацию о происходящих на сервере изменениях для восстановления базы данных в момент системной ошибки и для обеспечения целостности данных. О журнале мы будем говорить чуть позже, а сейчас необходимо понимать, что это отдельный файл, который требует дискового пространства.

Обратите внимание, что все числа кратны восьми или 16. Это связано с тем, что контроллеру жесткого диска проще читать данные именно такого размера. Данные читаются с диска страницами, т.е. по 8 килобайт, что является достаточно оптимальным значением.

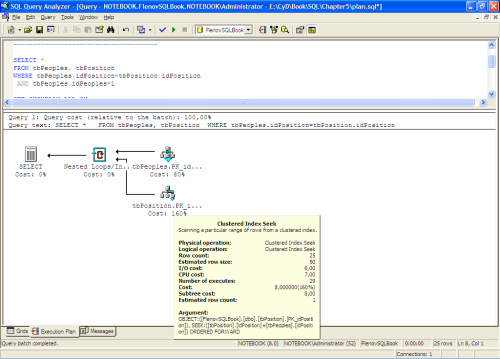
**Защита страниц**

Начиная с MS SQL Server 2005, у сервера баз данных появилась новая функция – контроль данных на уровне страниц. Если включен параметр **PAGE\_VERIFY\_CHECKSUM** (а по умолчанию он включен), то сервер будет контролировать контрольные суммы страниц. Если посмотреть мануал на этот параметр, то вы увидите, что контрольная сумма позволит отлавливать такие ошибки ввода выводы, которые не способна отловить ОС. Интересно, что это за ошибки? Видимо это внутренние проблемы сервера баз данных.

Проверка целостности лишней не бывает, поэтому ее лучше включить, а для этого нужно выполнить команду:

ALTER DATABASE имя базы SET PAGE\_VERIFY

Если в странице произошла ошибка, то сервер сообщит нам о ней. Но как же ее можно быстро исправить? Для этого появилась возможность восстановления данных на уровне страниц.

  
План выполнения запроса в графическом виде

**Расширение файлов**

Когда мы создаем базу данных, то нам предлагают выбрать ее начальный размер и метод приращения. Когда текущего пустого пространства не хватает, сервер расширяет его в соответствии с заданным методом приращения.

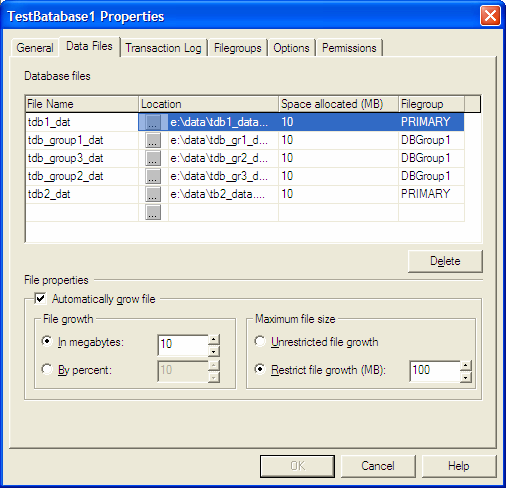
Существует три метода приращения файла:

* Увеличивать на определенное количество мегабайт;
* Увеличивать на определенный процент;
* Увеличивать вручную.

Первые два метода происходят автоматически, но рекомендуются к использованию только в тестовых базах, потому что администратор не имеет контроля над размером файлов. Если вас устраивает отсутствие контроля, то лучше поменять работу.

Если файл увеличивается на определенное количество мегабайт, то в определенный момент скорость наполнения данными может увеличиться и расширение файлов может стать слишком частым, а ведь это лишние расходы. Увеличение файла на процент тоже не выгодно. В качестве приращения рекомендуется использовать 10% и это правильно для маленьких и средних баз. Но когда она достигнет 1000 гигабайт, то при каждом увеличении придется выделять 100 гигабайт данных. А это уже бессмысленные траты дискового пространства.

Всегда контролируйте изменение размера файлов данных и журналов транзакций. Это позволит максимально эффективно использовать дисковые ресурсы.

  
Свойства базы данных MS SQL Server

**Компрессия данных**

Жесткий диск остается самым слабым, а точнее медленным звеном компьютера. Производительность процессоров растет с громадной скоростью, а из механических блинов винчестера выжать что-то новое становиться все сложнее. Дабы сэкономить количество операций ввода вывода и уменьшить размер данных, занимаемых на жестком диске, можно использовать диски с сжатием. Только на такие диски нужно помещать файловые группы, доступные только для чтения. Возможно, это связано с тем, что для записи нужно производить сжатие, а оно потребует излишних затрат процессора.

Сжатие данных и состояние только для чтения выгодно использовать с архивными данными. Например, бухгалтерские данные за прошлые годы работы не нужны для записи и могут занимать очень много места. Поместив данные на архивный раздел диска, вы сэкономите приличное пространство.

**Диски для надежности**

Следующий метод позволяет повысить надежность и производительность одновременно, и он снова связан с жесткими дисками. Ну что поделаешь, если механика не только самая медленная, но и самая не надежная. По поводу надежности я статистические данные не собирал, и не буду утверждать на все 100%, но дома и на работе у меня не работающих именно жестких дисков больше всего. Помниться было время, когда эти бездельники сыпались как листопад осенью.

Итак, повысить производительность и надежность одновременно можно банальным использованием двух и более жестких дисков вместо одного. Будет даже лучше, если они будут подключены к разным контроллерам. На один диск помещаем базу данных, на другом аккуратно располагаем журналы транзакций. Ну а если есть еще и третий диск, то на него помещаем саму систему.

Разбиение данных и журнала по разным дискам позволяет серьезно повысить надежность. Допустим, что у вас все находиться на одном диске, и он выходит из строя. Что делать? Можно поехать в специализированную компанию, которая попытается все восстановить или попробовать выполнить то же самое, но самостоятельно, но гарантия восстановления далека от 100%. К тому же, процесс возвращения сервера к работе затянется во времени. Быстрое восстановление можно сделать только на момент последней резервной копии. Остальное не гарантировано.

А теперь представим, что у нас данные и журнал транзакций разнесены по разным дискам. Если накрывается диск с журналом, то данные никуда не деваются. Единственное – нельзя добавлять новые данные, но стоит только создать новый журнал и можно продолжать работать.

Если ломается диск с данными, мы все еще можем зарезервировать журнал транзакций, чтобы не потерять ни капельки данных. После этого, восстанавливаем данные из полной копии (она обязательно должна быть у вас сделана заранее, а хороший администратор делает это не реже одного раза в день), и добавляем изменения из резервной копии журнала.

**Диски для производительности**

Если данные и журнал расположены на разных дисках, то это не только безопасность, но и повышение производительности. Дело в том, что сервер базы данных может одновременно писать данные в журнал и файл данных.

Можно пойти немного дальше, завести один жесткий диск для журнала транзакций, и несколько дисков для файлов данных. С данными сервер работает намного чаще, поэтому для них и нужно несколько хранилищ, с которыми можно было бы работать параллельно. А если эти хранилища подключены к разным контроллерам, то параллельная работа гарантирована.

Самый быстрый и надежный вариант – использовать **RAID**. Хотя нет, не каждый рейд одинаково надежен и одновременно быстрый. Для файловых групп чаще всего рекомендуется выбирать **RAID10**, потому что он обладает сбалансированными возможностями, но в зависимости от данных, которые находятся в вашей базе можно выбрать и другой вариант.

В качестве **RAID** можно использовать программное или аппаратное решение. Программное решение более дешевое, но отнимает лишние ресурсы центрального процессора. А у процессора лишних ресурсов не бывает. Поэтому, лучше использовать аппаратные решения, где за **RAID** отвечает выделенный чип.

**Индексы**

О том, что **индексы** позволяют повысить скорость поиска данных, об этом знают все. Большинство прекрасно понимает, что индексы отрицательно сказываются на вставке и обновлении данных, поэтому чем больше индексов, тем сложнее серверу их сопровождать. Но при этом, не многие даже задумываются о том, что индексы требуют обслуживания. Ведь страницы базы данных, содержащие индексные данные могут переполняться и со временем окажутся не сбалансированными.

Да, можно не обращать внимания на всякие параметры, а просто пересоздавать индексы раз в месяц, что идентично обслуживанию. Но чтобы индексы не устаревали через пол часа, после их создания, можно, у MS SQL Server есть два параметра: **FILLFACTOR** и **PAD\_INDEX**.

Вы можете использовать опцию **FILLFACTOR** для оптимизации производительности операций вставки и обновления в таблице, которые содержат кластерный или не кластерный индекс. Данные об индексе могут содержать множество страниц с данными. Как мы уже говорили в начале этой главы, каждая страница состоит из 8 кб. Когда индексная страница становится полной, сервер создает новую страницу, а ту страницу, в которую нужно вставить данные делит пополам.

На разделение страниц и создания свободного места для новой страницы серверу необходимо время. Используйте опцию **FILLFACTOR** для определения процента свободного пространства на листве индексной страницы для оптимизации разделения страницы. Чем больше свободного пространства в листовых страницах, тем реже придется в последствии делить страницы индекса. Но при этом дерево индекса будет слишком большим, и его обход будет отнимать лишнее время.

Опция PAD\_INDEX указывает процент, до которого заполняется не листовые индексные страницы. Вы можете использовать **PAD\_INDEX** опцию только тогда, когда указана опция **FILLFACTOR**, потому что процентное значение PAD\_INDEX зависит от процента, указанного в FILLFACTOR.

**Статистика**

Статистика позволяет серверу сделать наиболее правильный выбор между использованием индексов или полным сканированием таблицы. Допустим, что у нас есть список работников литейного цеха. В такой таблице наверно 90% (если не более) будут мужчины, ведь литейное производство достаточно тяжелое занятие для женщин, хотя, в нашей стране может быть что угодно.

Теперь допустим, что нам нужно найти всех женщин. Так как их мало, наиболее эффективным вариантом будет использование индекса. Но если нужно найти мужчин, то эффективность индекса падает. Количество выбираемых записей слишком велико и для каждой из них обходить дерево индекса будет лишними накладными расходами. Намного проще просканировать всю таблицу, что выполниться намного быстрее, потому что серверу достаточно по одному разу прочитать все листья нижнего уровня индекса, без необходимости многоразового чтения всех уровней.

Сервер SQL собирает статистику с помощью чтения всех значений полей или шаблона для создания равномерного распределенного и отсортированного списка значений. Сервер SQL динамически определяет процент строк, которые должны быть опробованы, основываясь на количестве строк в таблице. Оптимизатор запросов выполняет одно из двух – полное сканирование или шаблоны строк при сборке статистики.

Чтобы статистика начала работать, ее нужно создать. В случае массового обновления данных статистика может содержать некорректные данные, и сервер будет принимать неправильное решение. Но это все поправимо, нужно только следить за статистикой, а за более подробной информацией лучше обратиться к книгам по Transact-SQL или MS SQL Server.

**Итого**

Настройки по умолчанию не позволяют использовать оборудование на все 100% и одинаково работать на всем возможном разнообразии серверов. Ответственность за настройку ложиться на администраторов. То, что продукты корпорации Microsoft обладают простыми программами установки, графическими утилитами администрирования и возможностью работы в автономном режиме, не значит, что это оптимальный вариант.

Такие возможности тюнинга базы данных, как разгон компьютерного железа мы не рассматриваем, хотя и он имеет права на жизнь. А вот если все возможности тюнинга уже исчерпаны, то лучше задуматься об апгрейде, ведь разгон железа отрицательно сказывается на стабильности системы.

Ну и самое главное – никакая оптимизация сервера баз данных и никакой апгрейд не поможет, если запросы не оптимизированы.