

Оптимизируем работу MS SQL Server

Алексей Бережной

В статье рассматриваются вопросы первоначальной оптимизации работы MS SQL Server 2000. Несмотря на кажущуюся простоту описанных методов, следование изложенным рекомендациям позволит улучшить быстродействие, повысить надежность и управляемость SQL-сервером. Несмотря на то что статья предназначена в первую очередь для начинающих администраторов баз данных и системных администраторов, некоторые рекомендации пригодятся и опытным DBA.

Оптимизация работы SQL-сервера – вопрос весьма непростой. И MS SQL Server не является исключением. Существует огромное количество рецептов улучшения работы данного приложения, начиная от перестройки индексов таблиц и заканчивая переписыванием кода хранимых процедур, как говорится, с нуля.

Углубляясь в дебри индексов и фрагментов кода, как часто мы забываем о простых и понятных любому администратору вещах, таких как выделение нужного количества памяти или правильного размещения составных частей базы данных. Об этом и пойдет речь в этой статье. Предполагаю, что вы хотя бы вкратце знакомы с осно-

вами работы MS SQL Server 2000 и инструментарием для его управления, и знаете, что такое Enterprise Manager, Query Analyzer, SQL Server Books Online и как обращаться с этими инструментами. Также предполагаю, что вы обладаете необходимыми правами для выполнения всех манипуляций с настройками SQL-сервера и базами данных, описанными в этой статье. Вам будут встречаться некоторые конструкции языка Transact SQL (T-SQL) для использования их посредством Query Analyzer. Для разяснения непонятных моментов следует обращаться в SQL Server Books Online или к соответствующей литературе, список которой приведен в конце статьи. Несмотря на то что все

вопросы касаются MS SQL Server 2000, в большинстве своем они также справедливы и для SQL Server 2005.

Общая настройка MS SQL Server 2000

В этом разделе мы коснемся вопросов по изменению параметров конфигурации MS SQL Server 2000.

Распределение памяти

MS SQL Server 2000 поддерживает два режима распределения памяти: динамический и статический (см. **рис. 1**).

Просмотреть или изменить текущие установки памяти можно при помощи Enterprise Manager. Необходимо выбрать подключенный SQL-сервер

и, щелкнув на нем правой клавишей мыши, в появившемся меню выбрать пункт «Properties». В появившемся окне настроек выбрать пункт «Memory».

Динамический режим

В этом режиме MS SQL Server занимает всю доступную память под свои нужды. Если операционной системе или другому приложению, работающему на этом же сервере, требуется больше памяти, операционная система обрабатывает соответствующий запрос, и MS SQL Server освобождает необходимое количество памяти.

Динамический режим назначается по умолчанию при установке сервера как наиболее универсальный. Вы можете установить верхний и нижний предел объема выделяемой памяти. Установка для этих пределов одинакового значения фактически соответствует статическому режиму использования памяти.

Статический режим

Если на сервере не работают другие приложения, кроме MS SQL Server, или требования этих приложений к оперативной памяти невелики, то имеет смысл использовать статический режим распределения памяти. В этом случае мы выделяем определенный объем памяти для использования. Другие приложения и операционная система не имеют возможности использовать память, отведенную для нужд MS SQL Server. В свою очередь и MS SQL Server не «покушается» на участки памяти, которые ему не отведены.

Несмотря на то что динамический режим является наиболее универсальным и безопасным с точки зрения начинающего администратора, в большинстве случаев предпочтительней использовать статический режим распределения оперативной памяти. Это позволяет нашему SQL-серверу более выгодно использовать выделенную память, не заботясь о необходимости высвободить или перераспределить те или иные ее фрагменты. Даже если на одном сервере используется несколько приложений, имеет смысл при помощи статического режима распределить имеющуюся память во избежание траты ресурсов (включая процессорное время, обращение к файлу подкачки и т. д.) на процессы высвобож-

Предупреждение

Данный материал не является инструкцией по оптимизации, которую надлежит исполнять беспрекословно. Советы и рекомендации, представленные здесь, не являются окончательными и безапелляционными. Например, на протяжении всего материала настоятельно не рекомендуется использовать SQL-сервер для каких-либо других за-

дач и последующего ассигнования оперативной памяти.

Reverse physical memory for SQL server

Параметр, который нуждается в более пристальном рассмотрении, дословно можно перевести как «обратимое использование памяти SQL Server». Установка этого параметра запрещает использование файла подкачки для работы SQL-сервера. Если на аппаратном (или виртуальном) сервере достаточно оперативной памяти, чтобы обеспечить нужды MS SQL Server для решения поставленных задач, можно включить данный параметр, чтобы избежать процесса свопинга на диск. В некоторых случаях (в основном при использовании статического режима) это позволяет ощутимо повысить быстродействие работы MS SQL Server и использующих его приложений.

Замечание: несмотря на то что SQL Server чаще всего рассматривается как привилегированное приложение, которому выделяют максимальный объем имеющихся ресурсов, бывают случаи, когда следует пренеб-

дач. Но в то же время мы все прекрасно понимаем, что жизнь накладывает свои коррективы. С другой стороны, есть мудрая китайская пословица: «Кто предупрежден – тот вооружен». Поэтому важно не только изначально следовать той или иной методике, но и знать, где остались скрытые резервы, и вовремя использовать их, если этого потребуют обстоятельства.

речь этим правилом. Речь идет о ситуациях, когда при большой загрузке SQL Server (например, при выполнении «тяжелых» запросов) становится невозможным «достучаться» до сервера. В этом случае специально уменьшают объем памяти, используемый MS SQL Server. Для динамического режима это уменьшение верхнего предела используемой памяти (параметр «Maximum»), для статического режима это достигается снижением объема выделяемой памяти. В этом случае приходится жертвовать производительностью SQL-сервера в обмен на его управляемость.

Эффективное использование ресурсов процессоров

Описанные здесь настройки помогут использовать вычислительные ресурсы с максимальным выигрышем в производительности работы SQL-сервера.

Выбор количества используемых процессоров

Если на сервере установлен один процессор, то задача облегчается. Однако если мы имеем дело с мультипро-

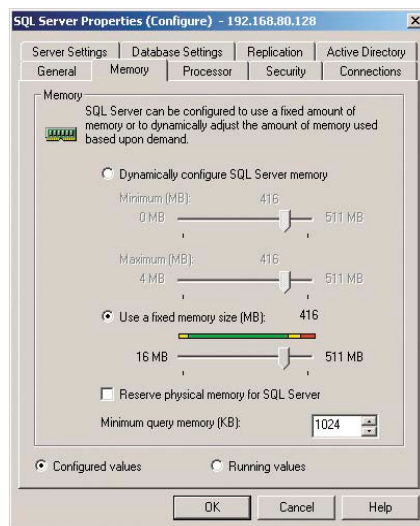


Рисунок 1. Установка памяти для MS SQL Server 2000

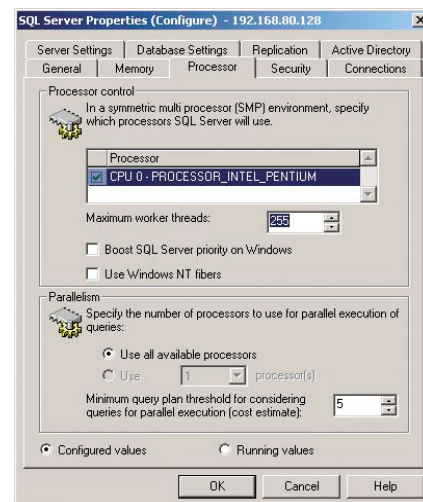


Рисунок 2. Изменение настройки параметров использования процессора (процессоров)

цессорной системой, у нас появляется возможность выбора и более тонкой настройки. В уже открытом нами в Enterprise Manager окне настроек SQL-сервера переходим на вкладку «Processor» (см. **рис. 2**).

Настройка используемых процессоров

При наличии в системе установлено одно процессора (или в настройках BIOS включена функция Hyper-Threading) появляется возможность использовать два и более процессоров. При этом можно явно указать, какие процессоры могут быть задействованы для нужд SQL-сервера.

Замечание: если имеются проблемы с доступом к серверу в моменты его работы под нагрузкой (см. выше соответствующий пункт о распределении памяти), следует помнить, что Windows для своих системных нужд всегда использует самый первый процессор (то есть CPU 0). Запретив его использовать SQL-серверу, мы теряем в производительности, но зато получаем возможность обращаться к серверу в момент его высокой загрузки. Принцип все тот же: жертвуем производительностью сервера в обмен на его управляемость. Следует также заметить, что в системах, в которых установлено четыре процессора и более, потери производительности будут не очень значительными.

«Boost SQL Server priority on Windows» (повышение приоритета SQL Server priority относительно Windows)

С этим параметром следует быть очень осторожным. При установке этого флажка резко повышается приоритет процесса MSSQLSERVER (основного процесса MS SQL Server 2000). Соответственно возможны ситуации снижения доступности сервера в периоды работы под нагрузкой. Но если есть необходимость в повышении производительности SQL-сервера и позволяет аппаратная и программная конфигурация, можно включить этот параметр.

Использование технологии Hyper-Threading

Использование технологии (создание виртуальных ядер) вызывает целый ряд вопросов.

Для начала рассмотрим, как работает технология. Понятие «виртуализации» говорит о том, что на самом деле внутри процессора нет двух вычислительных ядер. Удвоены только массивы регистров – общих и служебных. В данном случае имеет место конкуренция за кэш и другие ресурсы физического ядра, и в некоторых случаях используют его более эффективно, нежели одно ядро, как в процессорах, не поддерживающих Hyper-Threading.

Процессор по очереди отправляет на исполнение команды первого, второго или сразу обоих потоков, если есть свободные вычислительные ресурсы. Ни один из потоков не является приоритетным – процессор старается обработать оба. Когда первый поток команд останавливается в ожидании события или закидывается, то процессор переключается на второй поток команд и т. д. Это создает эффект ускорения исполнения низкоуровневых операций (например, доступ к подсистемам ввода-вывода).

Часто имеет место вредная конкуренция, когда виртуальные ядра в борьбе за обладание физическими ресурсами мешают работе друг друга. В любом случае один процессор не может работать с такой же скоростью, как два «нормальных» процессора. Он может лишь более эффективно использовать свои внутренние ресурсы.

Напомню, что в первую очередь процессоры с Hyper-Threading были

придуманы для того, чтобы ускорить время реакции пользовательских приложений. Но в ситуации, когда из приложений запущен только SQL-сервер, Hyper-Threading будет практически бесполезен.

Поэтому применять Hyper-Threading на многопроцессорных системах (т.е. «делать» из двух процессоров четыре и т. д.) для ускорения работы MS SQL Server 2000 в большинстве случаев не имеет смысла. Да и в системах с одним процессором лучше провести ряд тестов по оценке производительности, прежде чем принимать решение об установке MS SQL Server на компьютер, использующий Hyper-Threading. Ситуация осложняется также тем фактом, что для полноценной работы в этом режиме необходимо, чтобы операционная система была установлена с поддержкой многопроцессорного режима (в ряде случаев может потребоваться полная переустановка системы).

Увеличение интервала восстановления («recovery interval»)

В окне настроек SQL-сервера (Enterprise Manager) переходим на вкладку «Database Setting» (см. **рис. 3**).

По умолчанию MS SQL Server старается размещать всю необходимую для своей работы информацию в оперативной памяти, время от времени делая записи в журнал транзакций, необходимые для восстановления после сбоя. Процесс ведения таких записей в журнале транзакций называется созданием контрольных точек, то есть отправных моментов, основываясь на которые MS SQL Server будет восстанавливать состояние базы данных по журналу транзакций после аварии.

Параметр recovery interval определяет максимальное количество минут, которое затратит SQL на восстановление после некорректного завершения работы.

Примечание: на практике это не всегда так. Дело в том, что время восстановления после сбоя напрямую зависит от количества и объема незавершенных транзакций. Имели место случаи, когда recovery interval был установлен в «0», а восстановление базы шло продолжительное время. Но этот параметр создан, чтобы управлять временем восстановления.

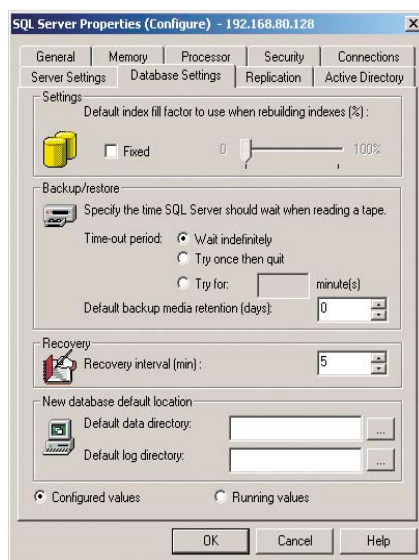


Рисунок 3. Изменение параметра recovery interval

Это время зависит от момента, когда была создана контрольная точка. Таким образом, параметр *recovery interval* определяет, с какой частотой будут создаваться контрольные точки. По умолчанию установлено нулевое значение, что указывает на автоматическое конфигурирование интервала (около 1 минуты). Если MS SQL Server работает на надежном оборудовании, защищенном от сбоев по питанию, рекомендуется увеличить этот интервал от 5 до 15 минут. Большой интервал выставлять не стоит, во-первых, из-за длительного временного периода, необходимого для восстановления базы данных, во-вторых, чем реже происходит создание контрольной точки, тем больший объем информации сбрасывается из оперативной памяти на диск. Поэтому не исключен вариант, когда очередное длительное создание контрольной точки совпадет по времени с выполнением «тяжелого» запроса и SQL-сервер будет сильно загружен в этот период.

Настройка сетевого подключения

При установке MS SQL Server 2000 оптимальные настройки сетевого подключения настраиваются автоматически. Однако после установки других приложений настройки могут быть изменены. Кроме того, в некоторых случаях (например, когда сервер с установленным MS SQL Server вынужден использоваться еще и как файл-сервер) данные настройки могут быть подвергнуты редактированию.

Просмотр или изменение установок оптимизации сервера Windows Server 2003 производится следующим образом (см. **рис. 4**).

Вызовите «Control Panel», запустите апплет «Network Connection», затем щелкните на значке нужного сетевого подключения (в нашем случае это Local Area Connection) правой кнопкой мыши. Из появившегося меню выберите «Properties».

На вкладке «General» диалогового окна выберите «File and Printer sharing for Microsoft Networks» и щелкните «Properties».

В диалоговом окне «File and Printer sharing for Microsoft Networks Properties» выберите одну из следующих возможностей:

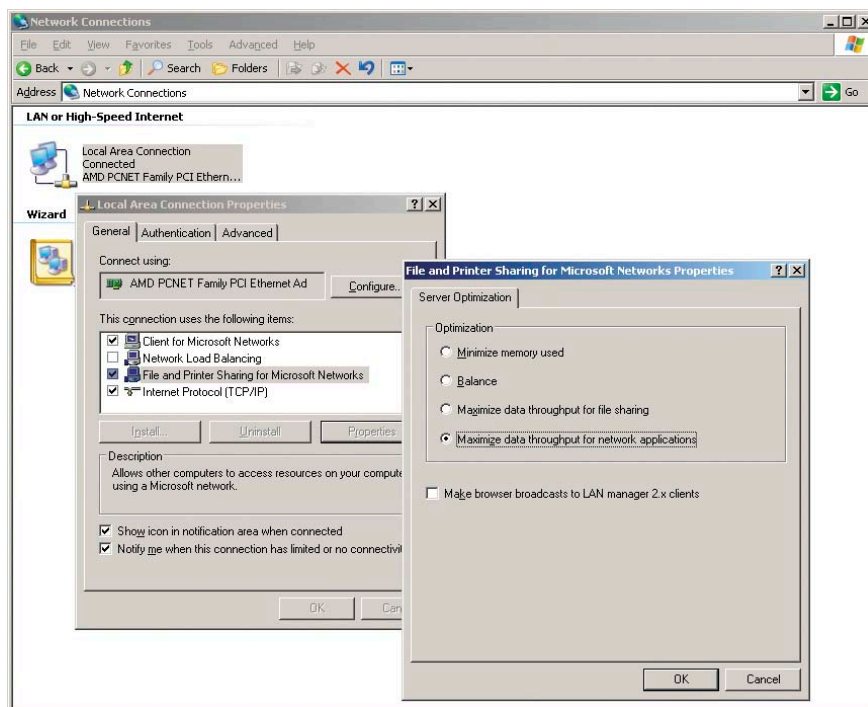


Рисунок 4. Установка оптимальных параметров сетевого подключения

- «**Maximize data throughput for network applications**» – вариант, который автоматически устанавливает MS SQL Server 2000;
- «**Maximize data throughput for file sharing**» – установите этот параметр, если компьютер, на котором установлен MS SQL Server, вынужден использоваться как файл-сервер.

Нажмите «OK» для сохранения изменений. Хочется еще раз напомнить, что самый лучший вариант, когда MS SQL Server работает на выделенном сервере, на котором не запущено никаких других приложений, за исключением вспомогательных утилит типа Performance Monitor и т. д. Также не рекомендуется использовать этот компьютер в качестве файл-сервера.

Оптимизация работы файлов баз данных

В приведенном здесь материале описываются настройки, позволяющие не только увеличить производительность, но и значительно облегчить себе дальнейшую работу путем оптимизации параметров прироста файлов баз данных.

Управление ростом файла базы данных

Все таблицы, индексы, хранимые процедуры и другие объекты, необходи-

мые для работы приложений, MS SQL Server хранит в файлах баз данных. Если во время работы растут таблицы, индексы, соответственно должен увеличиваться и файл базы данных.

Ростом этих файлов можно управлять вручную, можно все операции предоставить MS SQL Server (автоматический режим). Зачастую имеет смысл отредактировать параметры, используемые по умолчанию. Дело в том, что на автоматическое увеличение файлов баз данных тратятся дополнительные ресурсы, что заметно влияет на производительность SQL-сервера.

Чтобы отредактировать параметры файла базы данных, откройте Enterprise Manager, выберите необходимую базу, параметры которой необходимо отредактировать, щелкните правой кнопкой мыши и в появившемся меню выберите пункт «Properties». Откроется окно свойств базы данных (см. **рис. 5**). Перейдите на вкладку «Data Files».

Рекомендуется применять следующую стратегию:

- Лучше сразу увеличить файл базы данных на значительную величину. В примере указана величина 1 Гб. На практике данный параметр ограничен только имеющимся доступным дисковым пространством и здравым смыслом.

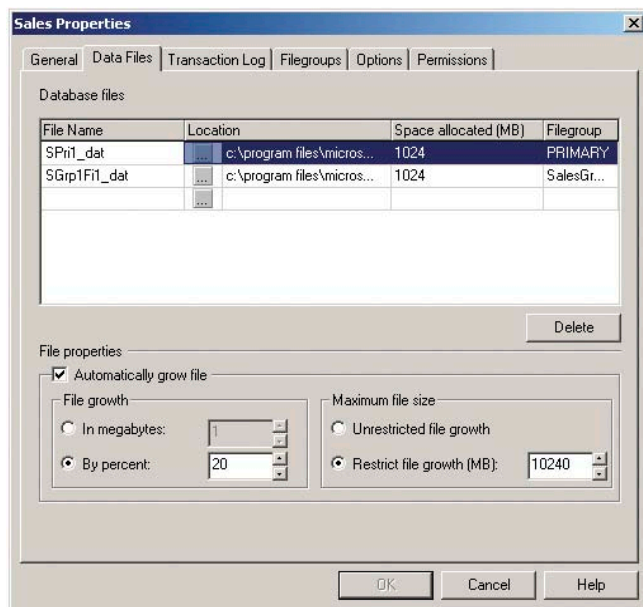


Рисунок 5. Изменение свойств файлов базы данных

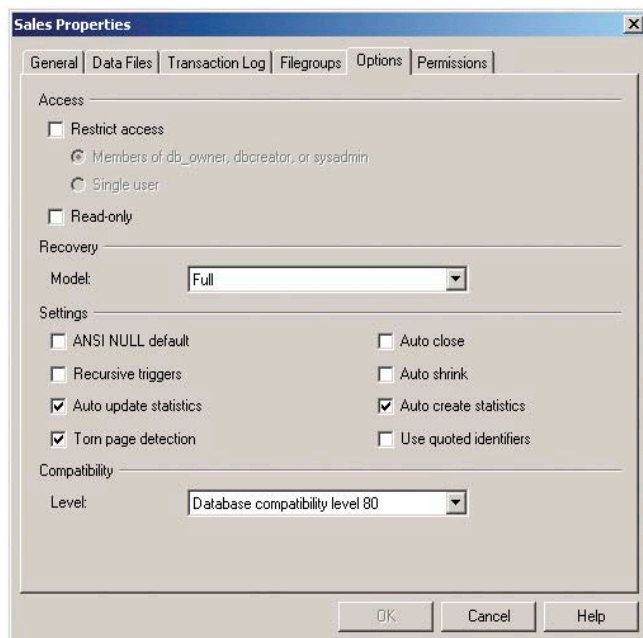


Рисунок 6. Отключение «Auto Shrink»

- Включить параметр «Automatically grow file».
- Установить высокий процент прироста базы данных (в приведенном примере – 20%).
- Выбрать параметр «Restrict file grow» и ввести достаточно большой предельный объем роста базы данных. Это необходимо для предотвращения случаев, если файл базы данных вырастает настолько, что займет весь дисковый объем без остатка, и администратор баз данных не успевает выполнить профилактические операции в виде сжатия баз данных (в приведенном примере установлен конечный размер, десятикратно превышающий первоначальный объем базы данных).

Отключение параметра «Auto Shrink»

MS SQL Server хранит информацию в файлах базы данных не в непрерывном последовательном виде, а разбив

на страницы по 8 Кб. Восемь страниц объединяются в один экстенст.

При росте объема хранимой информации (например, при увеличении числа записей в таблицах) растет размер файла базы данных. При удалении информации MS SQL Server не удаляет освободившиеся экстенсты, а использует их повторно. В случае если необходимо уменьшить файл данных за счет высвободившихся экстенстов, освободив при этом место на диске, используют процедуру shrink, выполняющую уплотнение («сжатие») файла базы данных и высвобождающую дисковое пространство.

Как и большинство операций, MS SQL Server способен выполнять данную процедуру автоматически (параметр «Auto Shrink»), не привлекая внимания администратора. Проблема заключается в том, что, во-первых, на процедуру постоянного автоматического сжатия тратятся драгоценные системные ресурсы, во-вторых, экстенсты при этом перетасовываются не самым оптимальным образом, файлы баз данных становятся сильно фрагментированными, что увеличивает время на поиск необходимой информации в базе данных. Соответственно растет время формирования запросов и т. д. Поэтому если дисковое пространство позволяет использовать файлы большего размера, данный параметр лучше отключить. В этом случае файлы баз данных вначале быстро возрастут до определенного объема, после чего их рост сильно замедлится за счет повторного использования высвобождающихся экстенстов. Если же все-таки необходимо периодически высвобождать часть дискового пространства, можно выполнить процедуру shrink над базой данных вручную непосредственно из Enterprise Manager или использовать конструкции T-SQL типа:

```
DBCC SHRINKDATABASE
( { имя базы данных } [ , {необходимый процент} ]
[ , { NOTRUNCATE | TRUNCATEONLY } ]
)
```

и/или:

```
DBCC SHRINKFILE
( { имя файла/ID файла }
{ [ , {необходимый размер} ]
| [ , { EMPTYFILE | NOTRUNCATE | TRUNCATEONLY } ]
}
)
```

В этом случае можно создать соответствующую хранимую процедуру и запускать ее как запланированное задание в часы минимальной нагрузки, например, по выходным дням.

Отключение «Auto Shrink»

Откройте Enterprise Manager, выберите необходимую базу, щелкните правой кнопкой мыши и в появившемся меню выберите пункт «Properties». В окне свойств базы данных перейдите на вкладку «Options» (см. рис. 6). Снимите галочку с параметра «Auto Shrink».

Эффективное распределение дискового пространства

В этой главе упоминаются понятия «отдельный раздел», «отдельный том». Отмечу, что имеется в виду физический

раздел или физический том, то есть либо самостоятельно используемый жесткий диск, либо RAID-массив жестких дисков (о том, что такое дисковые массивы и по каким принципам они организовываются, речь пойдет в следующем разделе).

Что же касается логических дисков, или, как их еще называют, дисковых подразделов, то следует понимать, что использование разных дисковых подразделов на одном физическом диске или массиве не даст никакого преимущества в быстродействии. Физическое устройство будет записывать или считывать данные с конструктивно заданной скоростью независимо от логического разбиения дискового пространства.

Перемещение файла журнала транзакций на отдельный раздел

Удивительно, как мало администраторов следует этой простой, но эффективной рекомендации.

Приведу всего лишь один из примеров, когда эта простая мера способна помочь в решении вопросов быстродействия. Представим себе ситуацию, когда один из пользователей формирует «тяжелый» отчет, а остальные в этот момент проводят пополнение базы данных. Если и файлы данных, и файлы журнала транзакций находятся на одном дисковом массиве, то жесткий диск вынужден постоянно переключаться между массовыми операциями чтения данных при построении отчета и операциями записи в журнал транзакций.

В итоге работа всего дискового массива будет крайне неэффективной. Перенос файлов журналов транзакций решает подобные проблемы.

Замечание: если база данных уже создана, невозможно перенести в другое место файл журнала транзакций, используя исключительно Enterprise Manager. Чтобы перенести файл журнала транзакций в другое расположение, необходимо выполнить detach базы данных (для этого можно воспользоваться Enterprise Manager), переписать файл журнала в нужное место и из Query Analyzer выполнить процедуру `sp_attach_db`, например:

```
EXEC sp_attach_db @dbname = N'{имя базы данных}',
    @filename1 = N'{полный путь к файлу.mdf}',
    @filename2 = N'{полный путь к файлу.ldf}'
```

Использование вторичных файлов и пользовательских файловых групп

По умолчанию при создании новой базы данных создается первичная файловая группа с первичным файлом данных. Этот файл базы данных имеет расширение *.mdf (рекомендуется придерживаться этого правила). Каждая база данных может иметь не более одного первичного файла.

К этой группе будут отнесены системные таблицы, в которых описываются пользователи, их права, объекты и другая информация, необходимая для нормального функционирования базы данных. Эта информация обновляется нечасто: при создании новых таблиц и индексов и т. д.

Вторичные файлы данных не являются обязательными. Они могут хранить данные и объекты, создаваемые в процессе работы. База данных может не иметь вторичных фай-

лов или иметь их несколько. Для этих файлов рекомендуется использовать расширение *.ndf

Можно также создавать и использовать пользовательские файловые группы (User-defined). Использование файловых групп позволяет более гибко использовать имеющиеся ресурсы, более эффективно контролировать ресурсы проводить модернизацию дисковой системы.

Существует понятие файловой группы по умолчанию (ее также называют стандартной файловой группой). Все таблицы и индексы, у которых при создании не была указана файловая группа, помещаются в файловую группу по умолчанию. Можно назначить любую файловую группу, принадлежащую к данной базе данных, файловой группой по умолчанию.

Чтобы сменить файловую группу по умолчанию используется команда `Transact SQL ALTER DATABASE`, например:

```
ALTER DATABASE {имя базы данных} 1
MODIFY FILEGROUP {имя файловой группы} DEFAULT
```

Суть метода оптимизации заключается в использовании пользовательских файловых групп и вторичных файлов данных, размещенных на других разделах, отдельно от первичных файлов и файлов журналов транзакций.

Пример команды создания базы данных с разными файловыми группами:

```
CREATE DATABASE Sales

ON PRIMARY
( NAME = SPri1_dat,
  FILENAME = '{полный путь к файлу.mdf}',
  SIZE = 10,
  MAXSIZE = 50,
  FILEGROWTH = 15% ),
FILEGROUP SalesGroup1
( NAME = SGrp1Fil_dat,
  FILENAME = '{полный путь к файлу.ndf}',
  SIZE = 10,
  MAXSIZE = 50,
  FILEGROWTH = 5 )
LOG ON
( NAME = 'Sales_log',
  FILENAME = '{полный путь к файлу.ldf}',
  SIZE = 5MB,
  MAXSIZE = 25MB,
  FILEGROWTH = 5MB )
GO
```

Также может быть полезно создать несколько разных вторичных файлов и пользовательских файловых групп и по возможности разместить их на разных дисковых массивах. В один файл поместить некластерные индексы, в другой – крупные пользовательские таблицы, в третий – вспомогательные таблицы и хранимые процедуры и т. д.

Размещение операционной системы

И наконец было бы неплохо операционную систему также разместить на отдельном физическом томе. Если оперативной памяти не очень много, то имеет смысл переместить файл подкачки (swap file) на отдельный физический том. Особенно это актуально, если имеется небольшое количество оперативной памяти, а также когда сервер используется не только для SQL, но и для других ролей. В этом случае исключается конкуренция за ресурсы дисковой подсистемы между операционной системой и SQL-сервером,

Таблица 1. Расположение файлов баз данных на типичном сервере баз данных

Дисковый массив (RAID1)	Размещенные компоненты	Примечание
1.	Операционная система, файл подкачки, MS SQL Server со служебными базами данных (включая tempdb), первичные файлы пользовательских баз данных	–
2.	Пользовательские файловые группы с вторичными файлами баз данных	Соответствующие пользовательские файловые группы назначены как файловые группы по умолчанию
3.	Файлы журналов транзакций	–

Таблица 2. Расположение файлов баз данных на сервере, ориентированном на формирование «тяжелых» запросов

Дисковый массив (RAID1)	Размещенные компоненты	Примечание
1.	Операционная система, файл подкачки, MS SQL Server со служебными базами данных (включая tempdb), первичные файлы пользовательских баз данных, файлы журналов транзакций	–
2.	Пользовательские файловые группы с вторичными файлами баз данных, содержащие таблицы для построения запросов	Некоторые пользовательские файловые группы назначены как файловые группы по умолчанию
3.	Пользовательские файловые группы с вторичными файлами баз данных, содержащие некластерные индексы таблицы для построения запросов	–

например, в момент загрузки сторонних приложений в память.

Примеры эффективного размещения файлов баз данных

Подводя итоги вышеописанных рекомендаций по размещению файлов баз данных, операционной системы и программного обеспечения MS SQL Server, получаем некую абстрактную картину:

- Отдельный дисковый массив для операционной системы.
- Отдельный дисковый массив для файла подкачки.
- Отдельный дисковый массив для первичных файловых групп и первичных файлов.
- Отдельный дисковый массив для пользовательских файловых групп и вторичных файлов, содержащих основные пользовательские таблицы.
- Отдельный дисковый массив для пользовательских файловых групп и вторичных файлов, содержащих индексы пользовательских таблиц.
- Ну и для полноты картины создать отдельный дисковый массив для хранения пользовательских процедур и вспомогательных таблиц, чтобы ничего не мешало выполнению «тяжелых» запросов с использованием основных таблиц и индексов.

В результате получилась некая избыточная модель, далекая от реальности. Даже предположив, что объем данных не слишком велик и самый большой файл базы данных легко поместится на один жесткий диск (скажем, на SAS размером 300 Гб). При этом если использовать RAID1 для создания отказоустойчивости, получаем минимальное количество дисков: $6 \times 2 = 12$. То есть для реализации вышеописанной схемы понадобится 12 винчестеров. Как правило, в большинстве своем серверы имеют возможность быть укомплектованными максимум 8 жесткими дисками. Обычно серверы имеют место на установку не более 6 винчестеров, если, конечно, речь не идет о схемах, использующих хранилища SAN. Следовательно, в обычной ситуации придется использовать более простые схемы размещения файлов баз данных и т. д., чтобы максимально приблизить идеальную модель к реальности.

Пример 1. Сервер с 6 жесткими дисками для текущей работы с базами данных

Допустим, имеется сервер с контроллером, поддерживающим RAID0, 1, 10 и возможностью установки 6 жестких дисков. Предполагается, что на нем будет работать MS SQL Server 2000 с поддержкой базы данных в обычном режиме: ввод новых данных, редакти-

рование имеющихся и построение отчетов. Предположим также, что по истечению определенного периода старые записи (созданные и не подвергшиеся редактированию в течение определенного периода) переносятся в архив и удаляются из основной базы данных. При этом объем данных не очень велик и может поместиться на одном жестком диске. Для решения подобных задач на выше описанном оборудовании можно предложить следующую схему (см. таблицу 1):

- Шесть дисков объединяются в три дисковых массива RAID1.
- На первый дисковый массив устанавливается операционная система, файл подкачки, MS SQL Server 2000 со всеми служебными базами и располагаются первичные файлы пользовательских баз данных.
- На втором дисковом массиве располагаются пользовательские группы и вторичные файлы баз данных. Соответствующие пользовательские файловые группы назначены как файловые группы по умолчанию.
- На третьем дисковом массиве располагаются файлы журналов транзакций.

Достоинства данной схемы размещения:

- Размещение файла журнала транзакций на другом дисковом разделе, отдельно от файлов данных дает выигрыш как при повседневных операциях по пополнению базы данных (ввод и редактирование), так и при массовых обработках данных (например, при массовом изменении значений полей и т. п.).
- Размещение основной информации во вторичных файлах данных на отдельном дисковом массиве дает выигрыш при построении запросов. Создание пользовательской группы с вторичными файлами данных на массиве RAID1 также позволяет легче увеличивать дисковое пространство (достаточно создать резервную копию соответствующих файлов, заменить диски на более емкие и перестроить RAID-массив, после чего восстановить файлы данных из резервной копии).

Недостатки данной системы размещения данных:

- При формировании «тяжелых» запросов описанных мер может быть недостаточно. В этом случае может потребоваться схема, при которой индексы и таблицы размещены в разных файловых группах.
- Также в данной схеме операционная система, файл подкачки, база данных tempdb находятся на одном разделе, что может несколько снижать производительность работы SQL-сервера.

Пример 2. Сервер с 6 жесткими дисками используется в основном для построения отчетов и небольшой корректировки данных

Имеется сервер, аналогичный приведенному в Примере 1. Предполагается, что на нем будет работать MS SQL Server 2000, использующийся в основном для построения отчетов, отображения необходимой информации и выполнения других действий, связанных с построением запросов, в том числе и «тяжелых». При этом объем данных может поместиться на одном жестком диске. Операции ввода данных незначительны, чтобы их принимать во внимание.

Итак, схема для решения этой и подобных задач может быть таковой (см. **таблицу 2**):

- Как и в первом примере, шесть дисков объединяются в три массива RAID1.
- На первый дисковый массив устанавливаются: операционная система с файлом подкачки, MS SQL Server 2000 со всеми служебными базами и располагаются первичные файлы пользовательских баз данных. Здесь же располагаются файлы журнала транзакций.
- На втором дисковом массиве располагаются пользовательские группы и вторичные файлы баз данных. Некоторые пользовательские файловые группы назначены как файловые группы по умолчанию. В файлах баз данных, размещенных на этом носителе, находятся пользовательские таблицы, по которым и будут формироваться необходимые запросы.

- На третьем дисковом массиве также располагаются пользовательские группы и вторичные файлы баз данных. Но в отличие от второго дискового массива в них располагаются преимущественно не кластерные индексы таблиц, участвующих в построении запросов.

Достоинства данной схемы размещения:

- Распределение таблиц и некластерных индексов позволяет ускорить время выполнения запросов, в том числе и тяжелых.
- Также имеется возможность без особого труда увеличить объем дискового пространства путем замены жестких дисков на более емкие.

Недостатки данной системы размещения данных:

- Размещение файла журнала транзакций на одном томе с операционной системой является не самым лучшим решением для увеличения быстродействия. Но в данном случае интенсивность обращений к файлу транзакций невелика, при этом существует необходимость высвободить отдельный дисковый массив для размещения некластерных индексов. Поэтому такое размещение оправданно.
- Точно так же операционная система, файл подкачки, база данных tempdb находятся на одном разделе, что может несколько снижать производительность работы SQL-сервера.

Приведенные схемы размещения файлов баз данных являются всего лишь примерами из практики. В каждом конкретном случае специфика бизнеса, используемые приложения, оборудование и другие факторы могут накладывать свой отпечаток на схему размещения файлов баз данных, файлов журнала транзакций и других необходимых компонентов системы.

Рекомендации по применению RAID-массивов для размещения файлов баз данных

Системы RAID состоят из двух или более дисков, которые обеспечивают бо-

лее высокую производительность и отказоустойчивость по сравнению с отдельным диском большого объема за меньшие деньги. Аббревиатура «RAID» расшифровывается как Redundant Array of Independent Disks (дословный перевод: «избыточный массив независимых дисков»). Основная задача RAID-массива – обеспечение высокой надежности дисковой системы за счет избыточности. Ведь жесткие диски – самая «скоропортящаяся» часть сервера. В то же время использование RAID-массивов в ряде случаев позволяет значительно увеличить производительность. В современных серверах обычно устанавливаются контроллеры, поддерживающие RAID0, 1, 5, 10.

Рекомендации по применению RAID0

В случае RAID0 два или несколько жестких диска обеспечиваются в один массив, равный суммарному объему дисков. За счет того, что разные фрагменты данных в режиме чередования записываются на разные дисковые накопители, повышается скорость чтения-записи.

Это самый ненадежный тип RAID-массива. Он всего лишь обеспечивает более высокий объем дискового пространства и/или более высокую скорость за счет чередования.

Очень редко возникает необходимость использовать данный тип RAID-массива. Чаще всего он используется для каких-то временных тестовых серверов, когда надежностью данных жертвуют в угоду повышения объема дискового пространства и производительности.

Рекомендации по применению RAID1

Данный тип дискового массива является наиболее распространенным за счет своей простоты организации и хорошей отказоустойчивости. Он поддерживается практически всеми контроллерами RAID-массивов. Дисковый массив на базе RAID1 – это два жестких диска, объединенных в зеркало. При применении данного типа дисковых массивов скорость выполнения операций чтения удваивается за счет чередования дисков. Из-за того что информация дублируется на оба диска, скорость записи определяется са-

мым медленным из дисков. Дисковый массив на основе RAID1 легко подлечит модернизации за счет установки более производительных и/или более емких дисков (в последнем случае придется сделать полную копию данных, расположенных на данном массиве на другой носитель и после замены дисков восстановится из резервной копии). RAID1 хорошо подходит для размещения журнала транзакций, так как файлы журнала транзакций обычно не отличаются большим размером, а запись в файл происходит последовательно.

Таким образом, RAID1 хорошо подходит для ситуаций:

- Когда используется недорогой RAID-контроллер с небольшим количеством функций и поддержка RAID1 – одно из немногих его достоинств.
- Когда планируется использовать распределение файлов баз данных по разным дисковым массивам.
- Для размещения файла подкачки.
- Когда в перспективе планируется произвести поэтапную модернизацию дисковой системы.
- Когда имеются конструктивные ограничения на количество устанавливаемых дисков.

Ограничения использования массива RAID1:

- Если размещаемые данные (например, файл базы данных) невозможно разместить на одном жестком диске, входящем в RAID1. Допустим, для данного сервера может быть использован винчестер с максимальной емкостью 300 Гб (280 Гб на отформатированном разделе). Соответственно, если необходимо разместить файл больше указанного объема, придется использовать другой тип RAID-массива, например RAID10 или RAID5.
- Другое ограничение заключается в скорости обмена данными. Если нужно получить скорость чтения записи больше чем скорость одного винчестера или скорость чтения больше двух скоростей винчестера, необходимо использовать другие типы RAID (например, RAID10 для ускорения операций чтения-записи или RAID5 для ускорения операций чтения).

Рекомендации по применению RAID10

RAID10 по сути является синтезом RAID0 и RAID1. Для каждого диска создается «зеркало», но каждый диск содержит только часть данных.

RAID10 обеспечивает отказоустойчивость подобно RAID1 и производительность за счет чередования, как у RAID0.

RAID10 хорошо подходит для ситуаций:

- Когда операции записи составляют более 10%.
- Когда производительность является критически важной.
- Когда размер файлов достаточно велик и данные не могут поместиться на соответствующий массив уровня RAID1.

Ограничение использования массива RAID10:

- RAID10 содержит минимум 4 жестких диска. Поэтому его использование может быть ограничено конструктивными особенностями сервера или хранилища SAN (некуда поместить необходимые жесткие диски).

Рекомендации по применению RAID5

RAID5 использует дополнительную запись контрольных сумм, которая позволяет восстановить информацию в случае выхода из строя одного из дисков. В связи с этим дисковые массивы на основе RAID5 самые медленные при работе с операциями записи.

В то же время за счет большого количества жестких дисков (3 и более накопителей) дисковые массивы на основе RAID5 показывают высокие скорости при операциях чтения.

Еще один плюс RAID5 – самые низкие потери при резервировании путем создания контрольных сумм. Например, при создании массива из 4 одинаковых жестких дисков реальный объем будет равен объему 3 накопителей. Для сравнения – RAID10 при тех же условиях позволит создать массив, равный 2 жестким дискам.

RAID5 хорошо подходит для ситуаций:

- Когда требуется высокая производительность при операциях чтения.

- Когда операции записи составляют не более 10% от всего объема.
- Когда требуется надежное и очень экономичное решение для создания дискового массива большого объема, при этом требования к производительности находятся на втором плане.


Ограничение использования массива RAID5:

- Когда требуется высокая производительность при операциях записи данных.
- Когда конструктивные особенности не позволяют использовать большое число дисков.

Заключение

В данной статье были рассмотрены самые простые вопросы оптимизации работы MS SQL Server 2000. Указанные методы не затрагивают такие сложные аспекты, как программирование хранимых процедур, перестройка индексов, обновление статистики и т. д.

В то же время эти простые приемы, доступные начинающим DBA и системным администраторам, способны улучшить работу SQL-сервера, сделать его работу более продуктивной и надежной.

Остается пожелать вам успехов в этом нелегком деле – оптимизации работы SQL-сервера. 

1. М.Ф. Гарсия, Дж. Реддинг, Э. Уолен, С.А. Делюк. «Microsoft SQL Server 2000. Справочник администратора». Издательство: «ЭКОМ», Москва, 2002.
2. «Администрирование Microsoft SQL Server 2000. Учебный курс MCSA/MCSE, MCDBA. Сертификационный экзамен 70-228». Издательство «Русская редакция», Издание 2-е, исправленное. Москва, 2003.
3. «Проектирование и реализация баз данных Microsoft SQL Server 2000. Учебный курс MCAD/MCSE/MCDBA». Экзамен 70-229. Издательство «Русская редакция», Питер, Microsoft Corporation, 2006 г.
4. Mike Aubert. «Изучи сервер Windows SQL 2000 за 15 минут в неделю» (серия статей).
5. Учебное пособие к «Microsoft Official Course 2072a Administering a Microsoft SQL Server 2000 Database».