Практическая работа 23
Обслуживание Microsoft SQL Server

Цель занятия: Получить практический опыт обслуживания Microsoft SQL Server

Перечень оборудования и программного обеспечения

Персональный компьютер
Microsoft Office (Word, Visio, Access)
Microsoft SQL Server Management Studio

Краткие теоретические сведения

Контрольные точки и их создание

Возможность создания *контрольных точек* была включена в SQLServer с целью фиксации изменений в базе данных или параметрах настройки в заранее заданный удобный момент времени. Если производилась настройка сервера или в его функционирование вносились изменения, потребовавшие перезагрузки управляющей программы, это значит, наступил момент для ручного запуска процедуры создания контрольной точки.

После запуска процедуры создания контрольной точки (независимо от способа ее вызова – вручную или посредством автоматического процесса) все модифицированные страницы памяти сбрасываются на диск. *Модифицированная страница* – это такая страница памяти, которая содержит изменения в данных, еще не зафиксированные в образе базы данных на жестком диске. По умолчанию процедура создания контрольной точки автоматически запускается приблизительно каждые 60 секунд. Реальный промежуток времени между двумя контрольными точками определяется текущей загрузкой сервера, заданными параметрами защиты от сбоев и текущими установками общей настройки производительности SQLServer. Однако этот реальный промежуток времени будет достаточно близок к 60 секундам.

Вероятно, вы уже заметили, что если SQLServer неожиданно прекратил работу в результате случайного сбоя, то на его повторный запуск может потребоваться достаточно много времени. Это происходит по той причине, что SQLServer осуществляет откат и повторное выполнение всех транзакций, которые происходили после создания последней контрольной точки. При этом база данных предварительно приводится в последнее, известное системе, корректное состояние. Оно соответствует именно тому моменту, когда был запущен и успешно завершен процесс создания последней контрольной точки.

Работа SQL Server может быть в любой момент остановлена с помощью выдачи команды shutdown. Если перед выдачей этой команды ввести команду checkpoint, то все внесенные в результате последних транзакций изменения будут корректно зафиксированы в базе данных до остановки ее работы.

Если по какой-либо причине предстоит остановить работу SQL Server (особенно если требуется указать параметр WITH NOWAIT), то можно избежать продолжительной процедуры последующего запуска системы, предварительно вручную выдав команду CHECKPOINT. Выполнение этой команды ни чем не отличается от процедуры создания контрольной точки, вызванной автоматически. Вся информация будет сохранена на диске и запуск системы будет сведен к простой загрузке программ ядра базы данных и предоставления доступа к данным всем клиентским приложениям. Это замечание особенно полезно в тех случаях, когда необходима экстренная остановка сервера – например, при отказе системы электропитания и наличии источника бесперебойного питания, способного обеспечить функционирование сервера на время, достаточное для выполнения цикла создания контрольной точки.

Команда checkpoint выполняется на уровне базы данных и всегда применяется по отношению к текущей базе данных. Если в системе имеется несколько баз данных, эту команду следует выдать для каждой из баз данных в отдельности. Чтобы иметь право выдать команду checkpoint для некоторой базы данных, необходимо иметь статус ее владельца.

Кроме этого, следует упомянуть параметр Truncatelogoncheckpoint, который также может оказаться довольно полезным. Установка этого параметра вызывает автоматическую очистку журнала транзакций после завершения создания очередной контрольной точки. Таким образом, журнал транзакций будет вестись только в промежутках между созданием контрольных точек. Для установки этого параметра в окне SQLServerEnterpriseManager щелкните правой кнопкой мыши на пиктограмме базы данных и выберите в контекстном меню команду Properties.

При установленном параметре Truncate log on checkpoint невозможно сделать резервную копию журнала транзакций в процессе выполнения стандартной процедуры резервного копирования базы данных. Обычно это не является проблемой, поскольку стандартное копирование базы данных по-прежнему будет выполняться. Однако данное замечание следует учитывать при планировании процедур копирования. Процесс резервного копирования и восстановления системы подробно обсуждается в последующих разделах этой главы.

Если для некоторой базы данных установлен параметр Truncate log on checkpoint, то ее журнал транзакций будет каждый раз очищаться в момент успешной фиксации последней транзакции (при отсутствии в системе репликации данных). Если в системе используется репликация данных, информация в журнале очищается в момент успешного завершения

репликации и фиксации в базе данных последней транзакции. Поскольку процедуры репликации данных используют данные журнала транзакций, информация о проведенной транзакции не удаляется из журнала до тех пор, пока сведения о ней не будут доставлены всем подписчикам данного источника.

Проверка целостности базы данных

USE master;

GO

EXEC sp_dboption 'Demo_DataBase', 'read only', 'TRUE';

Режимы выполнения команды dbcc

Команда DBCC поддерживает несколько режимов выполнения. В последующих разделах будут рассмотрены те из них, которые используются чаще всего. Кроме того, мы проанализируем, чем они могут помочь в управлении системой SQLServer.

DBCCCHECKALLOC При выполнении команды с указанием режима CHECKALLOC в отчет помещается детальная информация о системе и существующих в ней объектах базы данных.

DBCC CHECKALLOC [('*имя_базы_данных'[,* NOINDEX])] [WITH NO_INFOMSGS]

Если при вызове команды опустить имя базы данных, SQLServer осуществит проверку текущей базы данных. Эту информацию можно использовать для поиска возможных проблем в системе. Обнаруженные проблемы следует изучать и устранять по отдельности. Помещаемая в отчет информация – это сведения о существующей структуре таблиц, распределении страниц памяти и т.п. Каждое сообщение об обнаруженной ошибке обязательно сопровождается конкретными инструкциями по ее устранению.

Пример:

USE Demo_DataBase

GO

DBCC CHECKALLOC

Результат (фрагмент):

DBCC results for 'Demo_DataBase'.

****

Table sys.sysrowsetcolumns Object ID 4.

Index ID 1, partition ID 262144, alloc unit ID 262144 (type In-row data). FirstIAM (1:139). Root (1:66). Dpages 5.

Index ID 1, partition ID 262144, alloc unit ID 262144 (type In-row data). 7 pages used in 0 dedicated extents.

Total number of extents is 0.

****

...

CHECKALLOC found 0 allocation errors and 0 consistency errors in database 'Demo_DataBase'.

DBCC execution completed. If DBCC printed error messages, contact your system administrator.

DBCCcheckdbПри вызове команды DBCC в режиме CHECKDB, на наличие ошибок проверяется каждая таблица и связанные с ней страницы данных, индексы и указатели. Индексы и страницы данных контролируются на предмет корректности установленных связей. Кроме того, индексы дополнительно анализируются на соответствие заданному порядку сортировки. Для каждой страницы проверяется обоснованность данных, целостность указателей и корректность ее формата. Дополнительно в режиме CHECKDB контролируется правильность связей между текстом, графикой и страницами типа NTEXT, а также корректность их размера. Если выполняется вызов команды DBCC в режиме CHECKDB, то повторно запускать ее в режимах CHECKALLOC или CHECKTABLE не потребуется. Команда DBCC в режиме CHECKDB имеет следующий синтаксис:

DBCCCHECKDB [('*имя_базы_данных'*[, **NOINDEX])] [WITHNO_INFOMSGS]**

Если опустить параметр имени базы данных, будет выполнена проверка текущей базы данных.

DBCCCHECKTABLE

Если проверка всей базы данных в целом занимает слишком много времени, можно воспользоваться режимом checktable. В этом случае в команде dbcc задается режим checktable и указывается перечень имен таблиц базы данных, которые следует проверить. Будет выполнен анализ состояния только указанных таблиц, что позволит сократить общее время проверки.

DBCCCHECKFILEGROUP

При выполнении команды DBCC в режиме CHECKFILEGROUP, проверяется каждая таблица и связанные с ней страницы данных, индексы и указатели. Индексы и страницы данных контролируются на корректность установленных связей. Кроме того, индексы дополнительно анализируются на соответствие заданному порядку сортировки. Для каждой страницы проверяется обоснованность данных, целостность указателей и корректность ее формата. Дополнительно в режиме CHECKFILEGROUP контролируется правильность связей между текстом, графикой и страницами типа NTEXT, а также корректность их размера.

Команда DBCC в режиме CHECKFILEGROUP имеет следующий синтаксис:

DBCC CHECKFILEGROUP [([{'*filegroup_name*' | *filegroup_id* | 0 }] [, NOINDEX])]

[WITH {[ALL_ERRORMSGS][NO_INFOMSGS]] [, [TABLOCK]]

[, [ESTIMATEONLY]] }]

Команды UPDATE STATISTICS и RECOMPILE

Высокая производительность системы SQLServer в значительной степени обеспечивается за счет интеллектуальных процессов обработки сохраняемых в таблицах данных. Необходимый анализ информации осуществляется различными методами, однако самым рациональным из них является изучение сохраняемых в системе реальных данных с целью определения оптимальных путей доступа к ним.

Чтобы лучше уяснить, о чем идет речь, давайте в качестве примера рассмотрим маршрут следования к вашему собственному дому. Весьма вероятно, что вы уже нашли самый оптимальный и быстрый путь между местом работы и местом жительства. И раз этот маршрут уже найден, то чаще всего вы будете пользоваться именно им, даже если кто-то предложит вам воспользоваться иным путем. Ведь вы уже потратили достаточно времени на анализ различных возможных маршрутов и выбрали из них самый оптимальный.

Для того чтобы в сложившийся маршрут были внесены изменения, необходимы достаточно серьезные причины. Это может быть новая дорога, реконструкция уже существующих путей или что-либо в этом роде, способное оказать влияние на время, за которое вы добираетесь домой.

В системе SQLServer подобная аналогия вполне справедлива. При создании хранимой процедуры SQLServer анализирует логику ее оператора SELECT совместно с любыми прочими условиями, зависящими от обрабатываемых данных. На основе этого анализа выбирается оптимальный маршрут обращения к данным при выполнении хранимой процедуры. После того как маршрут определен, информация о нем запоминается, поэтому при каждом следующем выполнении хранимой процедуры используемый в ней алгоритм доступа к данным будет оптимальным. "Маршруты" к данным в SQLServer прокладываются на основе существующих в системе индексов таблиц. Именно эти элементы анализирует система, определяя наиболее эффективный способ доступа к требуемой информации.

Если в процессе работы в таблицу будет добавлено достаточно большое количество строк данных (превышающее 20% ее исходного размера), следует выполнить обновление статистических данных об этой таблице. Используемая для этой цели команда имеет следующий синтаксис:

UPDATESTATISTICS*table*| *view*
[
{
{ *index* | *statistics_name* }
| ({*index* |*statistics_name* } [,...*n*])
}
]
[WITH
[
[FULLSCAN]
| SAMPLE *number* { PERCENT | ROWS }]
| RESAMPLE

| <update_stats_stream_option>[,...*n*]
]
[[,] [ALL | COLUMNS | INDEX]
[[,] NORECOMPUTE]
] ;
<update_stats_stream_option> ::=
[STATS_STREAM = *stats_stream*]
[ROWCOUNT = *numeric_constant*]
[PAGECOUNT = *numeri△contant*]

Ниже приведен пример команды обновления статистики для таблицы Authors:

updatestatisticsAuthors

В команде можно потребовать выполнить пересчет статистики как для одного определенного индекса или столбца данных, так и для всей таблицы в целом. Если указывается отдельный индекс или столбец данных, то это делается так, как показано выше, с одновременным указанием имени таблицы, в которой этот индекс или столбец располагается.

Периодически SQL Server по собственной инициативе использует промежутки времени, когда процессор простаивает, для обработки таблиц с целью поддержания актуальности статистических данных, что способствует повышению эффективности обработки информации в системе.

Режим ручного обновления статистики также сохранен, причем соответствующая команда дополнена новыми параметрами. Так, при указании в команде обновления статистики параметра FULLSCAN SQL Server выполнит полное сканирование указанного индекса или таблицы. Наличие в команде параметра SAMPLE потребует от SQL Server использовать технологию выборок данных на основе указанного количества или процента строк. В этом случае SQL Server должен будет убедиться, что в качестве образца выбрано статистически значимое число значений. Если указанное число или процент строк недостаточно велики для выполнения этого требования, система автоматически увеличит заданное значение. Последний из параметров INDEX, COLUMN или ALL указывает, для каких объектов должна собираться статистика – индексов, столбцов или и тех, и других. При опускании этого параметра статистика собирается только для индексов.

Для прекращения автоматического сбора статистики (хотя я и не знаю, зачем это может кому-либо потребоваться) следует ввести команду UPDATE STATISTICS с параметром NORECOMPUTE. Указание этого параметра блокирует выполнение процедур автоматического сбора статистики в системе SQL Server.

Статистика по определенной таблице, индексу или столбцу может быть удалена, для чего предназначена команда DROP STATISTICS, имеющая приведенный ниже синтаксис:

DROPSTATISTICS таблица.столбец [,...таблица.столбец]

Последним шагом на пути обновления индексов с целью использования новых статистических данных в хранимых процедурах является уведомление SQL Server о необходимости пересчитать маршруты, используемые для извлечения информации из базы данных. Это достигается посредством перекомпиляции хранимых процедур, работающих с данными, статистические сведения о которых были обновлены только что выполненными командами UPDATE STATISTICS.

Обычно хранимые процедуры компилируются в момент первого их вызова после перезапуска системы SQL Server. Существуют и другие ситуации, в которых выполняется автоматическая перекомпиляция хранимых процедур, однако простая выдача команды UPDATE STATISTICS к ним не относится. В качестве примера можно указать ситуацию, когда ликвидируется индекс, ранее использовавшийся в некоторой хранимой процедуре. Иногда можно заметить, что при первом вызове некоторой хранимой процедуры время реакции системы оказывается существенно большим, чем обычно. Причем все последующие вызовы этой процедуры выполняются уже с нормальной скоростью. Это происходит по той причине, что оптимизатор выполнил перекомпиляцию хранимой процедуры с целью учета новых статистических данных, полученных для используемых в запросе таблиц или индексов.

Чтобы перекомпилировать хранимые процедуры, необходимо установить для таблицы соответствующий флажок, предназначенный для извещения системы о том, что любые работающие с этой таблицей и находящиеся в процедурном кэше копии хранимых процедур следует считать недействительными. В результате SQL Server перезагрузит и перекомпилирует все соответствующие процедуры при первом же их вызове. Для установки данного флажка используется следующая команда:

sp_recompile<*имя_таблицы*>

Здесь параметр *имя_таблицы* определяет имя той таблицы, для которой следует перекомпилировать все обращающиеся к ней хранимые процедуры. В случае успешного выполнения команда выводит простое сообщение, уведомляющее, что все связанные с указанной таблицей хранимые процедуры будут перезагружены и перекомпилированы.

Пример:

EXEC sp_recompile 'authors'

GO

Результат:

Object 'authors' was successfully marked for recompilation.

Если не перекомпилировать те хранимые процедуры, которые связаны с результатами выполнения команды update statistics, to ожидаемое улучшение производительности достигнуто не будет вплоть до очередной остановки и перезагрузки сервера. Только в этом случае все хранимые процедуры будут автоматически заново перекомпилированы.

Выполнение команд UPDATE STATISTICS и RECOMPILE не может нанести системе никакого вреда, однако делать это рекомендуется в то время,

когда активность пользователей в системе относительно невысока. Особенно важно придерживаться этого правила при работе с крупными базами данных. Время, которое будет затрачено на получение новых статистических данных, может существенно отразиться на производительности системы по обработке запросов пользователей. Лучше всего выполнять процедуры обновления статистики как часть регулярных манипуляций, связанных с плановым обслуживанием системы. Например, имеет смысл выполнять эту процедуру не реже, чем раз в месяц для каждой из интенсивно используемых в системе таблиц с невысоким уровнем изменения данных. В случае, когда система только запускается в работу и производится интенсивное заполнение данными ее таблиц, может потребоваться выполнять процедуры обновления статистики гораздо чаще – возможно, даже ежедневно, – если поток новых данных достаточно велик.

Задания

1 Изучить теоретические сведения.

2 В соответствии с вариантом задания выполнить операции по обслуживанию сервера:

- установить для рабочей базы данных режим «только для чтения»;
- осуществить проверку целостности базы данных;
- осуществить полную проверку базы данных;
- осуществить проверку каждой таблицы в отдельности;
- убрать режим «только для чтения»;
- открыть монитор активности, запустить разработанные ранее приложения базы данных и отследить изменения в окнах;
- экспортировать журналы за 3 последних дня.

Содержание отчета

1 Название работы
2 Цель работы
3 Перечень технических средств обучения
4 Порядок выполнения работы
5 Вывод

Варианты заданий

Варианты заданий представлены в практической работе № 13.

Используемая литература

– Г.Н.Федорова Основы проектирования баз данных. М.: Академия, 2020
– Г.Н.Федорова Разработка, администрирование и защита баз данных. М.: Академия, 2018