Резервное копирование и восстановление базы данных

Цель: сформировать знание об индексах и их типах, сформировать представление о резервном копировании и восстановлении базы данных, их способах

Индексы

Системы баз данных обычно используют индексы для обеспечения быстрого доступа к реляционным данным. *Индекс* представляет собой отдельную физическую структуру данных, которая позволяет получать быстрый доступ к одной или нескольким строкам данных. Таким образом, правильная настройка индексов является ключевым аспектом улучшения производительности запросов.

Индекс базы данных во многом сходен с индексом (алфавитным указателем) книги. Когда нужно быстро найти какою-либо тему в книге, тогда сначала просматривается индекс, на каких страницах книги эта тема рассматривается, а потом сразу же открывается нужная страница. Подобным образом, при поиске определенной строки таблицы SQL Server обращается к индексу, чтобы узнать ее физическое местонахождение.

Но между индексом книги и индексом базы данных есть две существенные разницы:

- ✓ читатель книги имеет возможность самому решать, использовать ли индекс в каждом конкретном случае или нет. Пользователь базы данных такой возможности не имеет, и за него это решение принимает компонент системы, называемый *оптимизатором запросов*;
- ✓ индекс для определенной книги создается вместе с книгой, после чего он больше не изменяется. Это означает, что индекс для определенной темы всегда будет указывать на один и тот же номер страницы. В противоположность, индекс базы данных может меняться при каждом изменении соответствующих данных.

Если для таблицы отсутствует подходящий индекс, для выборки строк система использует метод сканирования таблицы. Выражение *сканирование таблицы* означает, что система последовательно извлекает и исследует каждую строку таблицы (от первой до последней), и помещает строку в результирующий набор, если для нее удовлетворяется условие поиска в предложении WHERE. Таким образом, все строки извлекаются в соответствии с их физическим расположением в памяти. Этот метод менее эффективен, чем доступ с использованием индексов. Индексы сохраняются в дополнительных структурах базы данных, называющихся *страницами индексов*.

Для каждой индексируемой строки имеется *элемент индекса (index entry)*, который сохраняется на странице индексов. Каждый элемент индекса состоит из *ключа индекса* и *указателя*. Вот поэтому элемент индекса значительно короче, чем строка таблицы, на которую он указывает. По этой причине количество элементов индекса на каждой странице индексов намного больше, чем количество строк в странице данных. Это свойство индексов играет очень важную роль, поскольку количество операций ввода/вывода, требуемых для прохода по страницам индексов, значительно меньше, чем количество операций ввода/вывода, требуемых для прохода по соответствующим страницам данных. Иными словами, для сканирования таблицы, скорей всего, потребовалось бы намного больше операций ввода/вывода, чем для сканирования индекса этой таблицы.

Индексы в SQL Server создаются, используя структуру данных сбалансированного дерева B+. B+-дерево имеет древовидную структуру, в которой все самые нижние узлы (листья) находятся на расстоянии одинакового количества уровней от вершины (корневого узла) дерева. Это свойство поддерживается даже тогда, когда в индексированный столбец добавляются или удаляются данные.

Индексированный поиск обычно является предпочтительным методом поиска в таблицах с большим количеством строк по причине его очевидного преимущества. Используя индексированный поиск, возможно найти любую строку в таблице за очень короткое время, применив лишь несколько операций ввода/вывода. А последовательный поиск (т. е. сканирование таблицы от первой строки до последней) требует тем больше времени, чем дальше находится требуемая строка.

Кластеризованные индексы

Кластеризованный индекс определяет физический порядок данных в таблице. SQL Server позволяет создавать для таблицы лишь один кластеризованный индекс, т. к. строки таблицы нельзя упорядочить физически более чем одним способом. Поиск с использованием кластеризованного индекса выполняется от корневого узла B+-дерева по направлению к листьям дерева, которые связаны между собой в двунаправленный связанный список (doubly linked list), называющийся *цепочкой страниц (page chain)*. Важным свойством кластеризованного индекса является та особенность, что его листья дерева (узлы-листья) содержат страницы данных. Таблица, для которой определен кластеризованный индекс (явно или неявно), называется *кластеризованной таблицей*.

Кластеризованный индекс создается по умолчанию для каждой таблицы, для которой с помощью ограничения первичного ключа определен первичный ключ. Кроме этого, каждый кластеризованный индекс однозначен по умолчанию, т. е. в столбце, для которого определен кластеризованный индекс, каждое значение данных может встречаться только один раз. Если кластеризованный индекс создается для столбца, содержащего повторяющиеся значения, система баз данных принудительно обеспечивает однозначность, добавляя четырехбайтовый идентификатор к строкам, содержащим дубликаты значений.

Некластеризованные индексы

Структура некластеризованного индекса точно такая же, как и кластеризованного, но с двумя важными отличиями:

- ✓ некластеризованный индекс не изменяет физическое упорядочивание строк таблицы;
- ✓ сраницы листьев некластеризованного индекса состоят из ключей индекса и закладок.

Поиск данных с использованием некластеризованного индекса можно осуществлять двумя разными способами, в зависимости от типа таблицы:

- ✓ *куча* — прохождение при поиске по структуре некластеризованного индекса, после чего строка извлекается, используя идентификатор строки;
- ✓ *кластеризованная таблица* — прохождение при поиске по структуре некластеризованного индекса, после чего следует прохождение по соответствующему кластеризованному индексу.

В обоих случаях количество операций ввода/вывода довольно велико, поэтому следует подходить к проектированию некластеризованного индекса с

осторожностью, и применять его только в том случае, если есть уверенность, что его использование существенно повысит производительность.

Резервное копирование базы данных

Резервное копирование (backup) означает процесс создания копии базы (или баз) данных и/или журналов транзакций на отдельных носителях, которые в случае необходимости могут быть использованы для восстановления исходных данных. **Восстановлением (recovery)** называется процесс замены неподтвержденных, несогласованных или потерянных данных данными с резервной копии.

В системе SQL-сервер есть мастер плана обслуживания — Maintenance Plan Wizard, который предоставляет набор основных задач, требуемых для содержания базы в исправном состоянии. Поэтому его можно использовать для выполнения резервного копирования и восстановления пользовательских баз данных.

Создание резервной копии базы данных представляет собой выгрузка данных (из базы данных, журнала транзакций или файла) на устройства резервного копирования, поддерживаемые системой. Устройством резервного копирования может быть жесткий диск или накопитель на магнитной ленте.

Компонент Database Engine поддерживает как **статическое**, так и **динамическое резервное копирование**. **Статическое резервное копирование означает, что в процессе создания резервной копии единственным активным сеансом, поддерживаемым системой, является сеанс, который создает данную резервную копию**. Иными словами, выполнение пользовательских процессов в течение операции резервного копирования не разрешается. **Динамическое резервное копирование означает, что резервная копия может создаваться без прекращения работы сервера базы данных, отключения пользователей или даже закрытия файлов**. Более того, пользователи даже не будут знать, что выполняется резервное копирование.

Компонент Database Engine поддерживает четыре следующих метода создания резервных копий:

1. полное резервное копирование базы данных;
2. разностное резервное копирование;
3. резервное копирование журнала транзакций;
4. резервное копирование файлов или файловых групп.

Полное резервное копирование базы данных

Полное резервное копирование базы данных (full database backup) фиксирует то состояние базы данных, которое она имела на момент начала выполнения резервного копирования. В процессе создания такой резервной копии система копирует как данные, так и схему всех таблиц базы данных и соответствующие файловые структуры. В случае если резервное копирование выполняется динамически, то копируются все действия, происходящие в процессе создания резервной копии. Таким образом, в резервную копию попадают даже неподтвержденные незафиксированные транзакции.

Разностное резервное копирование

Разностное резервное копирование (differential backup) создает копию только тех частей базы данных, которые были добавлены или изменены после выполнения последнего полного резервного копирования базы данных. (Как и в случае полного резервного копирования, любая деятельность во время

динамического разностного резервного копирования также заносится в создаваемую копию.) Достоинством разностного резервного копирования является быстрота его выполнения. Для создания такой резервной копии базы данных требуется существенно меньше времени, чем для полной копии, поскольку объем копируемых данных значительно меньше (полное резервное копирование базы данных включает копии всех ее страниц).

Резервное копирование журнала транзакций

Резервное копирование журнала транзакций (transaction log backup) учитывает только те данные, которые связаны с изменениями, записанными в журнал транзакций. Таким образом, эта форма резервного копирования основана не на физических составляющих базы данных (страницах), а на логических операциях, т. е. на изменениях, выполненных посредством DML-инструкций: INSERT, UPDATE и DELETE. Поскольку при этом копируется меньший объем данных, то этот процесс создания резервной копии может быть выполнен значительно быстрее, чем полное или разностное резервное копирование базы данных.

ПРИМЕЧАНИЕ Выполнять резервное копирование журнала транзакций не имеет смысла, если не было проведено по крайней мере одно, полное резервное копирование базы данных.

Существует две основные причины, по которым следует выполнять резервное копирование журнала транзакций. Первая, чтобы сохранить на защищенном носителе данные, которые изменились со времени последнего резервного копирования журнала транзакций, а вторая, более важная, чтобы должным образом закрыть часть журнала транзакций перед началом новой порции действий его активной части. (Активная часть журнала транзакций содержит записи обо всех неподтвержденных незафиксированных транзакциях.) Используя полное резервное копирование базы данных и надежные цепочки всех резервных копий журналов транзакций, можно распространить копию базы данных и на другой компьютер. Эту копию базы данных можно будет использовать для восстановления исходной базы данных в случае ее повреждения. (Подобным образом базу данных можно восстановить, используя полную резервную копию и последнюю разностную резервную копию.)

Компонент Database Engine не позволяет сохранять журнал транзакций в том же самом файле, в котором сохраняется база данных. Одной из причин этого является то, что в случае повреждения этого файла использовать журнал транзакций для восстановления всех изменений, выполненных после последнего резервного копирования, будет невозможно. Использование журнала транзакций для записи изменений в базе данных является достаточно широко распространенной функциональной возможностью, которая применяется почти во всех реляционных СУБД. Тем не менее, иногда могут возникнуть ситуации, в которых будет полезным эту возможность отключить. Например, обработка большого объема загружаемых данных может длиться несколько часов. В таком случае скорость выполнения программы можно увеличить, отключив протоколирование транзакций. Но с другой стороны, это чревато опасными последствиями, поскольку при этом разрушаются существующие цепочки журналов транзакций. Чтобы обеспечить успешное восстановление базы данных, после окончания загрузки данных настоятельно рекомендуется выполнить полное резервное копирование базы данных.

Одной из самых распространенных причин системных сбоев является переполнение журнала транзакций. Следует иметь в виду, что такая проблема может полностью остановить систему. Если дисковое пространство, выделенное для хранения журнала транзакций, полностью заполнится, то система будет вынуждена прекратить выполнение всех текущих транзакций до тех пор, пока для журнала не будет предоставлено дополнительное пространство. Этой проблемы можно избежать, только выполняя частое резервное копирование журнала транзакций: при каждом закрытии части журнала транзакций и сохранении его на другой носитель эта часть журнала становится повторно используемой, освобождая таким образом дисковое пространство.

Преимущество разностного резервного копирования заключается в том, что оно позволяет сэкономить время при восстановлении, поскольку для полного восстановления базы данных требуется ее полная резервная копия и только последняя разностная резервная копия. Для полного же восстановления с использованием резервных копий журнала транзакций необходимо использовать полную резервную копию базы данных и все резервные копии журнала транзакций. Недостатком разностного резервного копирования является то, что такая копия не позволяет восстановить данные на определенный момент времени, т. к. в ней не сохраняются промежуточные изменения базы данных.

Резервное копирование файлов или файловых групп

Резервное копирование файлов или файловых групп позволяет вместо полной резервной копии базы данных создать резервную копию только определенных файлов (или файловых групп) базы данных. В таком случае компонент Database Engine создает резервную копию только указанных файлов. Из резервной копии базы данных можно восстанавливать отдельные файлы (или файловые группы), что позволяет выполнять восстановление данных после сбоя, который повлиял только лишь на часть файлов базы данных. Восстановление отдельных файлов или файловых групп можно выполнять из полной резервной копии базы данных или из резервной копии файловой группы. Это означает, что в качестве процедуры резервного копирования можно применять полное резервное копирование базы данных и резервное копирование журнала транзакций и в то же время иметь возможность восстанавливать отдельные файлы (или файловые группы) из этих резервных копий.

ПРИМЕЧАНИЕ Резервное копирование файлов также называется резервным копированием на уровне файлов. Данный тип резервного копирования рекомендуется применять только в случае очень большой базы данных, когда нет времени достаточного для выполнения полного резервного копирования.

Восстановление базы данных

Компонент Database Engine поддерживает как автоматическое, так и ручное восстановление базы данных.

Автоматическое восстановление

Автоматическое восстановление представляет собой средство, позволяющее сохранять работоспособность системы при возникновении ошибок. Компонент Database Engine выполняет автоматическое восстановление при каждом его перезапуске после сбоя или после его останова. Процесс автоматического восстановления проверяет, требуется ли восстановление баз данных, и если обнаруживает такую необходимость, то возвращает каждую базу данных в ее последнее согласованное состояние, используя для этого журнал транзакций.

В процессе автоматического восстановления компонент Database Engine исследует журнал транзакций от последней контрольной точки до точки сбоя системы или останова Database Engine.

Контрольной точкой называется последняя точка, в которой все выполненные изменения записываются из оперативной памяти в базу данных. Таким образом, контрольная точка обеспечивает физическую согласованность данных.

Журнал транзакций содержит **подтвержденные (зафиксированные) транзакции**, т. е. транзакции, которые были успешно выполнены (зафиксированы в журнале транзакций), но их результаты еще не были записаны в базу данных, и **неподтвержденные (незафиксированные) транзакции**, т. е. транзакции, которые не были успешно выполнены на момент отключения или сбоя. Компонент Database Engine повторяет все подтвержденные транзакции, в результате чего делает изменения в базе данных постоянными, и отменяет часть неподтвержденных транзакций, которые были выполнены до контрольной точки.

Компонент Database Engine сначала выполняет восстановление базы данных master, после чего следует восстановление всех остальных системных баз данных. Процесс завершается восстановлением пользовательских баз данных.

Ручное восстановление

В процессе ручного восстановления применяется полная резервная копия базы данных, с последовательным применением всех резервных копий журнала транзакций в порядке их создания. Эти действия возвращают базу данных в то же самое (согласованное) состояние, в котором она находилась, когда в последний раз было выполнено последнее резервное копирование журнала транзакций. При восстановлении базы данных с применением полной резервной копии компонент Database Engine сначала воссоздает все файлы баз данных и размещает их на соответствующих физических носителях. После этого система воссоздает все объекты баз данных. Компонент Database Engine может выполнять определенные формы восстановления в динамическом режиме (т. е. в процессе работы экземпляра базы данных). Динамическое восстановление повышает уровень доступности системы, т. к. недоступными являются только восстанавливаемые данные. Восстанавливать динамически можно или весь файл базы данных, или файловую группу. В Microsoft динамическое восстановление называется «восстановлением в режиме онлайн» — **online restore**.

Проверка резервного набора на пригодность для восстановления

После выполнения инструкции BACKUP заданное устройство резервного копирования (файл на диске или магнитная лента) содержит все данные выбранного для резервного копирования объекта. Эти данные называются **набором резервного копирования** (**backup set**). Прежде чем приступать к восстановлению базы данных с резервного набора, необходимо удостовериться в том, что он: содержит все данные, которые требуется восстановить; является пригодным для использования.

Компонент Database Engine поддерживает набор инструкций Transact-SQL, посредством которых можно проверить соответствие резервного набора этим требованиям. В набор инструкций, среди прочих, входят следующие инструкции:

- ✓ RESTORE LABELONLY;
- ✓ RESTORE HEADERONLY;
- ✓ RESTORE FILELISTONLY;
- ✓ RESTORE VERIFYONLY.

Инструкция RESTORE LABELONLY. Инструкция RESTORE LABELONLY применяется для отображения информации о заголовке носителя (жесткого диска или магнитной ленты) резервной копии. Выводимая в результате выполнения этой инструкции единственная строка содержит суммарную информацию о заголовке носителя (имя носителя, описание процесса резервного копирования и дату создания резервной копии).

Инструкция RESTORE HEADERONLY. Тогда как инструкция RESTORE LABELONLY предоставляет краткую информацию о заголовке файла устройства резервного копирования, инструкция RESTORE HEADERONLY предоставляет информацию о хранящихся на этом устройстве резервных копиях. В частности, инструкция выводит одну строку суммарной информации для каждой резервной копии на устройстве резервного копирования. В отличие от инструкции RESTORE LABELONLY выполнение инструкции RESTORE HEADERONLY для носителя, содержащего несколько резервных копий, может занять достаточно длительное время. Вывод инструкции содержит столбец Compressed, в котором указывается состояние сжатия соответствующего файла резервной копии. Значение 1 в этом столбце означает, что файл сжат.

Инструкция RESTORE FILELISTONLY. Эта инструкция возвращает результирующий набор со списком файлов баз данных и журналов транзакций, содержащихся в резервном наборе. Информация возвращается только для одного резервного набора. Поэтому если носитель содержит несколько резервных копий, необходимо указать позицию той, для которой требуется информация. Инструкцию RESTORE FILELISTONLY следует использовать только в тех случаях, когда неизвестно, какие имеются резервные наборы или где находятся файлы определенного резервного набора. В обоих случаях можно проверить весь или часть носителя, чтобы получить глобальную картину о существующих резервных копиях.

Инструкция RESTORE VERIFYONLY. Определив местонахождение требуемой резервной копии, можно выполнять следующий шаг: проверить ее целостность, не запуская процесс восстановления с ее применением. Такую проверку можно выполнить с помощью инструкции RESTORE VERIFYONLY, которая проверяет наличие всех носителей резервного копирования (файлов жестких дисков или магнитных лент), а также возможность считывания находящейся на них информации.

В отличие от предшествующих трех инструкций, инструкция RESTORE VERIFYONLY поддерживает две особые опции:

- ✓ LOADHISTORY — задает добавление информации о резервной копии в таблицы историй резервного копирования;
- ✓ STATS — выводит сообщение после выполнения считывания очередной порции информации и используется для отслеживания масштаба процесса (значением по умолчанию является 10.)

Безопасность баз данных и привилегии

При хранении информации в СУБД одной из основных задач является обеспечение безопасности данных. В языке SQL используются следующие основные принципы защиты данных:

- ✓ в БД действующими лицами являются пользователи. Манипуляции с данными происходят от имени конкретного пользователя;
- ✓ в SQL используется система привилегий, т.е. прав пользователя на проведение тех или иных действий над определенным объектом базы данных.

Администратор БД создает пользователей и дает им привилегии, пользователи, которые создают таблицы, сами имеют права на управление этими таблицами.

Каждый пользователь в среде SQL имеет:

✓ специальное идентификационное имя. Команда, посланная в БД, ассоциируется с определенным пользователем;

✓ набор привилегий. Эти привилегии могут изменяться со временем – новые добавляться, старые удаляться. При этом пользователь, создавший таблицу любого вида, является ее владельцем. Это означает, что такой пользователь имеет все привилегии в этой таблице и может передавать привилегии другим пользователям для данной таблицы.

Существуют следующие привилегии:

✓ SELECT – пользователь может выполнять запросы к таблице;
✓ INSERT – пользователь может выполнять вставку записей;
✓ UPDATE – пользователь может выполнять корректировку данных;
✓ DELETE – пользователь может выполнять удаления в таблице;
✓ REFERENCES – пользователь может определять внешние ключи;
✓ INDEX – пользователь может создавать индекс в таблице;
✓ SYNONYM – пользователь может создавать синонимы для объекта;
✓ ALTER – пользователь может выполнять команду ALTER TABLE.

В SQL привилегии даются и отменяются двумя операторами – **GRANT (допуск)** и **REVOKE (отмена)**.

Синтаксис:

GRANT привилегия1, привилегия2…
ON таблица
TO пользователь1, пользователь2…;

Для оператора **GRANT** существует два аргумента, которые имеют специальное значение – это **ALL PRIVILEGES** (используется вместо имени привилегии, чтобы отдать все привилегии в таблице) и **PUBLIC** (используется вместо имени пользователя, чтобы отдать соответствующую привилегию всем пользователям). Для того чтобы пользователи могли передавать свои привилегии другим пользователям, используется оператор **WITH GRANT OPTION**.

Синтаксис:

GRANT привилегия1, привилегия2…
ON таблица
TO пользователь1, пользователь2…
WITH GRANT OPTION;

С помощью этого оператора пользователь получает особые привилегии для данной таблицы, и может предоставить эту привилегию к той же таблице другому пользователю.

Для удаления привилегии используется оператор **REVOKE**.

Синтаксис:

REVOKE привилегия
ON таблица
FROM пользователь;

При этом привилегии отменяются пользователем, который их предоставил.

Вопросы

1. Дайте определение понятию *индекс*.

2. Поясните отличия между индексом книги и индексом базы данных.
3. Поясните смысл выражения *сканирование таблицы*.
4. Дайте определение понятию *страницы индекса*.
5. Дайте определение понятию *элемент индекса*.
6. Поясните понятие B+-дерева.
7. Назовите существующие типы индексов. Поясните их.
8. Дайте определение понятиям *цепочка страниц* и *кластеризованная таблица.*
9. Дайте определение понятию *куча*.

Преподаватель С.В. Банцевич