Журнализация

Базы данных SQL Server содержат файлы трех типов:

- **Первичные файлы данных**. Первичный файл данных является отправной точкой базы данных. Он указывает на остальные файлы базы данных. В каждой базе данных имеется один первичный файл данных. Для имени первичного файла данных рекомендуется использовать расширение MDF.
- **Вторичные файлы данных**. Ко вторичным файлам данных относятся все файлы данных, за исключением первичного файла данных. Некоторые базы данных могут вообще не содержать вторичных файлов данных, тогда как другие содержат несколько вторичных файлов данных. Для имени вторичного файла данных рекомендуется использовать расширение NDF.
- **Файлы журналов**. Файлы журналов содержат все сведения журналов, используемые для восстановления базы данных. В каждой базе данных должен быть, по меньшей мере, один файл журнала, но их может быть и больше. Для имен файлов журналов рекомендуется использовать расширение LDF. Журнал транзакций нельзя ни удалять, ни изменять, если только не известны возможные последствия. Кэш журнала управляется отдельно от буферного кэша для страниц данных.

Журнал транзакций поддерживает следующие операции:

- **Восстановление отдельных транзакций.** Если приложение выполняет инструкцию ROLLBACK или если компонент Database Engine обнаруживает ошибку, такую как потеря связи с клиентом, записи журнала используются для отката изменений, сделанных незавершенной транзакцией.
- **Восстановление всех незавершенных транзакций при запуске SQL Server.** Если на сервере, где работает SQL Server, происходит сбой, базы данных могут остаться в таком состоянии, в котором часть изменений не была переписана из буферного кэша в файлы данных, и могут быть изменения в файлах данных, совершенные незаконченными транзакциями. Когда экземпляр SQL Server будет запущен, он выполнит восстановление каждой базы данных. Будет выполнен накат каждого записанного в журнал изменения, которое, возможно, не было переписано в файл данных. Чтобы сохранить целостность базы данных, будет также произведен откат каждой незавершенной транзакции, найденной в журнале транзакций.
- **Накат восстановленной базы данных, файла, файловой группы или страницы до момента сбоя.** После потери оборудования или сбоя диска, затрагивающего файлы базы данных, можно восстановить базу данных на момент, предшествующий сбою. Сначала восстановите последнюю полную резервную копию и последнюю дифференциальную резервную копию базы данных, затем восстановите последующую серию резервных копий журнала транзакций до момента возникновения сбоя. Поскольку восстанавливается каждая резервная копия журнала, компонент Database Engine повторно применяет все модификации, записанные в журнале, для наката всех транзакций. Когда последняя резервная копия журнала будет восстановлена, тогда компонент Database Engine начнет использовать данные журнала для отката всех транзакций, которые не были завершены на момент сбоя.
- **Поддержка репликации транзакций.** Агент чтения журнала следит за журналами транзакций всех баз данных, которые настроены на репликацию транзакций, и копирует отмеченные для репликации транзакции из журнала транзакций в базу данных распространителя. Дополнительные сведения см. в разделе Как работает репликация транзакций.
- **Поддержка решений с резервными серверами.** Решения резервного сервера, зеркальное отображение базы данных и доставка журналов в значительной степени полагаются на журнал транзакций. В сценарии доставки журналов основной сервер отправляет активный журнал транзакций основной базы данных одному или более адресатам. Каждый сервер-получатель восстанавливает журнал в свою локальную базу данных-получатель.

Операции резервного копирования и восстановления выполняются в контексте модели восстановления. **Модель восстановления** — это свойство базы данных, которое управляет процессом регистрации транзакций, определяет, требуется ли для журнала транзакций резервное копирование, а также определяет, какие типы операций восстановления доступны. Есть три модели восстановления:

- **Модель полного восстановления (FULL).** Обеспечивает модель обслуживания для баз данных, в которых необходима поддержка длительных транзакций. Требуются резервные копии журналов. При использовании этой модели выполняется полное протоколирование всех транзакций, и сохраняются записи журнала транзакций до момента их резервного копирования. Модель полного восстановления позволяет восстановить базу данных до точки сбоя при условии, что после сбоя возможно создание резервной копии заключительного фрагмента журнала. Кроме того, модель полного восстановления поддерживает восстановление отдельных страниц данных.
- **Модель восстановления с неполным протоколированием (BULK_LOGGED)**. Эта модель восстановления обеспечивает неполное протоколирование большинства массовых операций. Она предназначена для работы только в качестве дополнения к полной модели полного восстановления. Для ряда масштабных массовых операций (массовый импорт, создание индекса и т. п.) временное переключение на модель восстановления с неполным протоколированием повышает производительность и уменьшает место, необходимое для журналов. Тем не менее, для работы этой модели требуются резервные копии журналов. Как и в модели полного восстановления, в модели восстановления с неполным протоколированием сохраняются записи журнала транзакций после его резервного копирования. Это увеличивает объем резервных копий журналов и повышает риск потери результатов работы,

поскольку модель восстановления с неполным протоколированием не поддерживает восстановление до заданного момента времени.

- **Простая модель восстановления (SIMPLE)**. Сводит к минимуму административные затраты, связанные с журналом транзакций, поскольку его резервная копия не создается. При использовании простой модели восстановления в случае повреждения базы данных возникает риск потери значительной части результатов работы. Данные могут быть восстановлены только до момента последнего резервного копирования. Поэтому при использовании простой модели восстановления интервалы между резервным копированием должны быть достаточно короткими, чтобы предотвратить потерю значительного объема данных. В то же время они должны быть велики настолько, чтобы затраты на резервное копирование не влияли на производительность. Снизить затраты поможет использование разностного резервного копирования.

Обычно в базе данных используется модель полного восстановления или простая модель восстановления. Модели восстановления оказывают влияние на поведение журнала транзакций и способ регистрации операций, либо на и на то, и на другое.

Модель восстановления FULL подразумевает, что регистрируется каждая часть каждой операции, и это называется полной регистрацией. После выполнения полного резервного копирования базы данных в модели восстановления FULL в журнале транзакций не будет проводиться автоматическое усечение до тех пор, пока не будет выполнено резервное копирование журнала. Если вы не намерены использовать резервные копии журнала и возможность восстановления состояния базы данных на конкретный момент времени, не следует использовать модель восстановления FULL. Однако, если вы предполагаете использовать зеркальное отображение базы данных, тогда у вас нет выбора, поскольку оно поддерживает только модель восстановления FULL.

Модель восстановления BULK_LOGGED обладает такой же семантикой усечения журнала транзакций, как и модель восстановления FULL, но допускает частичную регистрацию некоторых операций, что называется минимальной регистрацией. Примерами являются повторное создание индекса и некоторые операции массовой загрузки — в модели восстановления FULL регистрируется вся операция.
Но в модели восстановления BULK_LOGGED регистрируются только изменения распределения, что радикально сокращает число создаваемых записей журнала и, в свою очередь, сокращает потенциал разрастания журнала транзакций.

Модель восстановления SIMPLE, фактически ведет себя с точки зрения ведения журнала так же, как и модель восстановления BULK_LOGGED, но имеет совершенно другую семантику усечения журнала транзакций. В модели восстановления SIMPLE невозможны резервные копии журнала, что означает, что журнал может быть усечен (если ничто не удерживает записи журнала в активном состоянии) при возникновении контрольной точки.

SQL Server поддерживает восстановление данных на следующих уровнях:

- **База данных** (*полное восстановление базы данных*). Вся база данных возвращается в прежнее состояние и восстанавливается, при этом база данных находится в автономном режиме во время операций возврата и восстановления.
- **Файл данных** (*восстановление файла*). Файл данных или набор файлов данных возвращается в исходное состояние и восстанавливается. Во время восстановления файлов файловые группы, содержащие обрабатываемые файлы, автоматически переводятся в автономный режим на время восстановления. Любые попытки подключения и работы с недоступной файловой группой приведут к ошибке.
- **Страница данных** (*восстановление страницы*). При использовании модели полного восстановления или модели восстановления с неполным протоколированием можно восстановить отдельные базы данных. Восстановление страниц может применяться для любой базы данных вне зависимости от числа файловых групп.

Логическая архитектура журнала транзакций

На логическом уровне журнал транзакций состоит из последовательности записей. Каждая запись содержит:

- *Log Sequence Number*: регистрационный номер транзакции (*LSN*). Каждая новая запись добавляется в логический конец журнала с номером *LSN*, который больше номера *LSN* предыдущей записи.
- *Prev LSN*: обратный указатель, который предназначен для ускорения отката транзакции.
- *Transaction ID number*: идентификатор транзакции.
- *Type*: тип записи *Log*-файла.
- *Other information*: Прочая информация.

Двумя основными типами записи *Log*-файла являются:

- *Transaction Log Operation Code:* код выполненной логической операции, либо
- *Update Log Record:* исходный и результирующий образ измененных данных. Исходный образ записи – это копия данных до выполнения операции, а результирующий образ – копия данных после ее выполнения.

Действия, которые необходимо выполнить для восстановления операции, зависят от типа *Log*-записи:

- Зарегистрирована логическая операция.
  - Для наката логической операции выполняется эта операция.
  - Для отката логической операции выполняется логическая операция, обратная зарегистрированной.
- Зарегистрированы исходный и результирующий образы записи.
  - ля наката операции применяется результирующий образ.
  - Для отката операции применяется исходный образ.

В журнал транзакций записываются различные типы операций:

0 BEGINXACT: Begin Transaction
4 Insert
5 Delete
6 Indirect Insert
7 Index Insert
8 Index Delete
9 MODIFY: Modify the record on the page
11 Deferred Insert (NO-OP)
12 Deferred Delete (NO-OP)
13 Page Allocation
15 Extent Allocation
16 Page Split
17 Checkpoint
20 DEXTENT
30 End Transaction (Either a commit or rollback)
38 CHGSYSINDSTAT — A change to the statistics page in sysindexes
39 CHGSYSINDPG — Change to a page in the sysindexes table

Каждая транзакция резервирует в журнале транзакций место, чтобы при выполнении инструкции отката или возникновения ошибки в журнале было достаточно места для регистрации отката. Объем резервируемого пространства зависит от выполняемых в транзакции операций, но обычно он равен объему, необходимому для регистрации каждой из операций. Все это пространство после завершения транзакции освобождается.

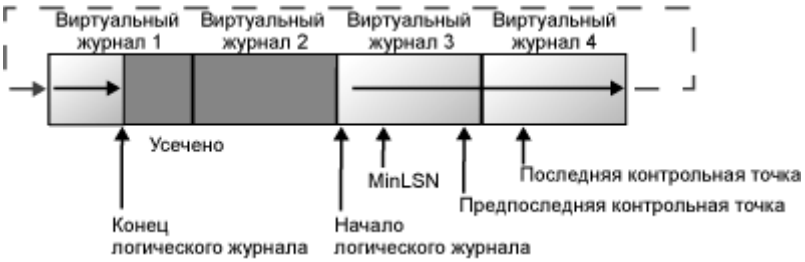
Раздел журнального файла, который начинается от первой записи, необходимой для успешного отката на уровне базы данных, до последней зарегистрированной записи называется *активной частью журнала*, или *активным журналом*. Именно этот раздел необходим для выполнения полного восстановления базы данных. Активный журнал не может быть усечен.

Физическая архитектура журнала транзакций

На физическом уровне журнал транзакций состоит из одного или нескольких физических файлов. *К*аждый физический файл журнала разбивается на несколько виртуальных файлов журнала (*VLF*). *VLF* не имеет фиксированного размера. Не существует также и определенного числа *VLF*, приходящихся на один физический файл журнала. Компонент Database Engine динамически определяет размер *VLF* при создании или расширении файлов журнала. Компонент Database Engine стремится обслуживать небольшое число *VLF*. Администраторы не могут настраивать или устанавливать размеры и число *VLF*.

Журнал транзакций является *оборачиваемым файлом*. Рассмотрим пример. Пусть база данных имеет один физический файл журнала, разделенный на четыре виртуальных файла журнала. При создании базы данных логический файл журнала начинается в начале физического файла журнала. Новые записи журнала добавляются в конце логического журнала и приближаются к концу физического файла журнала. Усечение журнала освобождает любые виртуальные журналы, все записи которых находятся перед минимальным регистрационным номером восстановления в журнале транзакций (*MinLSN*). *MinLSN* является регистрационным номером самой старой записи, которая необходима для успешного отката на уровне всей базы данных. Журнал транзакций рассматриваемой в данном примере базы данных будет выглядеть примерно так же, как на следующей иллюстрации.

Когда конец логического журнала достигнет конца физического файла журнала, новые записи журнала будут размещаться в начале физического файла журнала.



Этот цикл повторяется бесконечно, пока конец логического журнала не совмещается с началом этого логического журнала. Если старые записи журнала усекаются достаточно часто, так что при этом всегда остается место для новых записей журнала, созданных с новой контрольной точки, журнал постоянно остается незаполненным. Однако, если конец логического журнала совмещается с началом этого логического журнала, происходит одно из двух событий, перечисленных ниже:

- Если для данного журнала применена установка FILEGROWTH и на диске имеется свободное место, файл расширяется на величину, указанную в **growth_increment**, и новые записи журнала добавляются к этому расширению. Дополнительные сведения о настройке FILEGROWTH см. в разделе ALTER DATABASE (Transact-SQL).
- Если установка FILEGROWTH не применяется или диск, на котором размещается файл журнала, имеет меньше свободного места, чем это указано в **growth_increment**, формируется ошибка 9002.
- Если в журнале содержится несколько физических файлов журнала, логический журнал будет продвигаться по всем физическим файлам журнала до тех пор, пока он не вернется на начало первого физического файла журнала.

Просмотр журнала транзакций

Вариант А. Недокументированная команда DBCC LOG(*имя_базы_данных, тип_вывода*)
где *тип_вывода* принимает значение из диапазона 0..4. По умолчанию принят 0 (*Current LSN, Operation, Context, Transaction ID*). 3 выдает полный информационный дамп каждой операции.

Вариант В. Недокументированная функция fn_dblog().

Пример.

```
create table t1 (id int, name varchar(20));
insert into t1 values (1, 'test1');
checkpoint;
insert into t1 values (2, 'test2');
checkpoint;
select * from t1;
drop table t1;
-- DBCC LOG(N'dbtest', 3)
-- Выводится таблица 45*102
-- select top 1 * from fn_dblog(null, null)
-- Выводится таблица 1*97
```

Контрольные точки и активная часть журнала

Так как все изменения страниц данных происходят в страничных буферах, то изменения данных в памяти не обязательно отражаются в этих страницах на диске. Процесс кэширования происходит по алгоритму последней использованной страницы, поэтому страница, подверженная постоянным изменениям, помечается как последняя использованная, и она не записывается его на диск. Чтобы эти страницы были записаны на диск применяется контрольная точка. Все грязные страницы должны быть сохранены на диске в обязательном порядке.
Контрольная точка выполняет в базе данных следующее:

- Записывает в файл журнала запись, отмечающую начало контрольной точки.
- Сохраняет данные, записанные для контрольной точки в цепи записей журнала контрольной точки. Одним из элементов данных, регистрируемых в записях контрольной точки, является *номер LSN* первой записи журнала, при отсутствии которой успешный откат в масштабе всей базы данных невозможен. Такой *номер LSN* называется минимальным *номером LSN* восстановления (*MinLSN*). Номер *MinLSN* является наименьшим значением из:
  - *номера LSN* начала контрольной точки;
  - *номера LSN* начала старейшей активной транзакции;
  - *номера LSN* начала старейшей транзакции репликации, которая еще не была доставлена базе данных распространителя.
  - Записи контрольной точки содержат также список активных транзакций, изменивших базу данных.
- Если база данных использует простую модель восстановления, помечает для повторного использования пространство, предшествующее номеру *MinLSN*.
- Записывает все измененные страницы журналов и данных на диск.
- Записывает в файл журнала запись, отмечающую конец контрольной точки.
- Записывает в страницу загрузки базы данных *номер LSN* начала соответствующей цепи.

Действия, приводящие к срабатыванию контрольных точек

Контрольные точки срабатывают в следующих ситуациях:

- При явном выполнении инструкции CHECKPOINT. Контрольная точка срабатывает в текущей базе данных соединения.
- При выполнении в базе данных операции с минимальной регистрацией, например при выполнении операции массового копирования для базы данных, на которую распространяется модель восстановления с неполным протоколированием.
- При добавлении или удалении файлов баз данных с использованием инструкции ALTER DATABASE.
- При остановке экземпляра SQL Server с помощью инструкции SHUTDOWN или при остановке службы SQL Server (MSSQLSERVER). И в том, и в другом случае будет создана контрольная точка каждой базы данных в экземпляре SQL Server.
- Если экземпляр SQL Server периодически создает в каждой базе данных автоматические контрольные точки для сокращения времени восстановления базы данных.
- При создании резервной копии базы данных.
- При выполнении действия, требующего отключения базы данных. Примерами могут служить присвоение параметру AUTO_CLOSE значения ON и закрытие последнего соединения пользователя с базой данных или изменение параметра базы данных, требующее перезапуска базы данных.

Автоматические контрольные точки

Компонент Database Engine создает контрольные точки автоматически. Интервал между автоматическими контрольными точками определяется на основе использованного места в журнале и времени, прошедшего с момента создания последней контрольной точки. Интервал между автоматическими контрольными точками колеблется в широких пределах и может быть довольно длительным, если база данных изменяется редко. При крупномасштабных изменениях данных частота автоматических контрольных точек может быть гораздо выше.
Можно использовать параметр конфигурации сервера **recovery interval** для вычисления интервала между автоматическими контрольными точками для всех баз данных на экземпляре сервера. Значение этого параметра определяет максимальное время, отводимое компоненту Database Engine на восстановление базы данных при перезапуске системы. Компонент Database Engine оценивает количество записей журнала, которые он может обработать за время **recovery interval** при выполнении операции восстановления.
Если используется простая модель восстановления базы данных, автоматическая контрольная точка создается каждый раз, когда число записей в журнале достигает меньшего из двух предельных условий:

- журнал заполняется на 70 процентов;
- число записей в журнале достигает значения, определенного компонентом Database Engine в качестве количества записей, которое он может обработать за время, заданное параметром **recovery interval**.

Примечание. Сведения об установке интервала восстановления см. в разделе Как установить интервал восстановления (среда SQL Server Management Studio).

Активный журнал

Часть журнала, начинающаяся с номера MinLSN и заканчивающаяся последней записью, называется активной частью журнала, или активным журналом. Этот раздел журнала необходим для выполнения полного восстановления базы данных. Ни одна часть активного журнала не может быть усечена. Все записи журнала до номера MinLSN должны быть удалены из частей журнала.

На следующем рисунке изображена упрощенная схема журнала завершения транзакций, содержащего две активные транзакции. Записи контрольных точек были сжаты в одну запись.

Последней записью в журнале транзакций является запись с *номером LSN*, равным 148. На момент обработки записанной контрольной точки с *номером LSN* 147 транзакция 1 уже зафиксирована и единственной активной транзакцией является транзакция 2. В результате первая запись журнала, созданная для транзакции 2, становится старейшей записью активной транзакции на момент последней контрольной точки. Таким образом, *номером MinLSN* становится *номер LSN*, равный 142 и соответствующий записи начала транзакции 2.

Длительные транзакции

Активный журнал должен включать в себя все элементы всех незафиксированных транзакций. Приложение, инициирующее транзакцию и не выполняющее ее фиксацию или откат, не позволяет компоненту Database Engine повышать *MinLSN*. Это может привести к проблемам двух типов.

- Если система будет выключена после того, как транзакцией было выполнено много незафиксированных изменений, этап восстановления при последующем перезапуске может занять гораздо больше времени, чем указано параметром **recovery interval.**
- Журнал может достичь очень большого объема, потому что после *номера MinLSN* усечь его нельзя. Это справедливо даже в том случае, если используется простая модель восстановления, когда журнал транзакций обычно усекается при каждой автоматической контрольной точке.

Журнал транзакций с упреждающей записью

SQL Server использует журнал с упреждающей записью, который гарантирует, что до занесения на диск записи, связанной с журналом, никакие изменения данных записаны не будут. Таким образом обеспечиваются свойства ACID для транзакции. Для понимания принципов работы журнала с упреждающей записью важно знать принципы записи измененных данных на диск. SQL Server поддерживает буферный кэш, из которого система считывает страницы данных при извлечении необходимых данных. Изменения данных не заносятся непосредственно на диск, а записываются на копии страницы в буферном кэше. Изменение не записывается на диск, пока в базе данных не возникает контрольная точка, или же изменение должно быть записано на диск таким образом, чтобы для хранения новой страницы мог использоваться буфер. Запись измененной страницы данных из буферного кэша на диск называется сбросом страницы на диск. Страница, измененная в кэше, но еще не записанная на диск, называется грязной страницей.

Во время изменения страницы в буфере запись журнала строится в кэше журнала, который записывает изменение. Данная запись журнала должна быть перенесена на диск до того, как соответствующая «грязная» страница будет записана из буферного кэша на диск. Если «грязная» страница переносится на диск до записи журнала, эта страница создает изменение на диске, которое не может быть откачено, если сервер выйдет из строя до переноса записи журнала на диск. SQL Server обладает алгоритмом, который защищает «грязную» страницу от записи на диск до переноса на него соответствующей записи журнала. Содержимое журнала запишется на диск после того, как будут зафиксированы транзакции.

Управление журналом транзакций

Чтобы логический журнал не увеличивался до размера физических файлов журнала, следует периодически выполнять его усечение. Процесс усечения журнала уменьшает размер файла логического журнала, помечая виртуальные файлы журнала, которые не содержат частей логического журнала, как неактивные. В некоторых случаях может оказаться полезным физическое сжатие или расширение размера реального файла журнала.

Время усечения журнала зависит от модели восстановления базы данных. Есть три модели восстановления: простая модель восстановления, модель полного восстановления и модель восстановления с неполным протоколированием. Обычно в базе данных используется полная модель восстановления или простая модель восстановления. В качестве примера рассмотрим усечение журнала в простой модели восстановления.

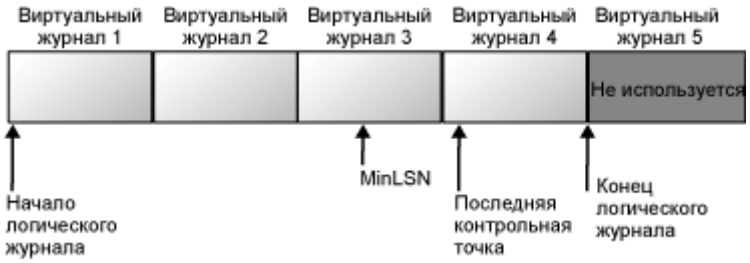
При использовании простой модели восстановления усечение журналов выполняется автоматически. Если все записи в виртуальном файле журнала неактивны, то этот логический журнал усекается обычно после контрольной точки. При этом освобождается место для повторного использования. Это относится и к контрольным точкам инструкции CHECKPOINT, и к неявным контрольным точкам, сформированным системой. Однако усечение журнала может быть отложено, если виртуальные файлы журнала остаются активными вследствие выполнения долгой транзакции или резервного копирования.

Эта модель регистрирует только минимальные сведения, необходимые для обеспечения согласованности базы данных после сбоя системы или для восстановления данных из резервной копии. Это сводит к минимуму расход места на диске под журнал транзакций по сравнению с другими моделями восстановления. Чтобы предотвратить переполнение журнала, базе данных требуется достаточно места для записи в случае задержки его усечения.

Как работает усечение журнала

Кроме прочих данных, в контрольной точке записывается *номер LSN* первой записи журнала, которую необходимо сохранить для успешного отката на уровне базы данных. Этот *номер LSN* называется минимальным *номером LSN* восстановления (*MinLSN*). Начало активной части журнала занято *VLF*, содержащим *MinLSN*. При усечении журнала транзакций освобождаются только те записи, которые находятся перед этим *VLF*.

На следующем рисунке показан журнал транзакций до усечения и после. На первом рисунке показан журнал транзакций, который никогда не усекался. В настоящий момент логический журнал состоит из четырех виртуальных файлов. Логический журнал начинается с начала первого файла виртуального журнала и заканчивается виртуальным файлом журнала 4. Запись *MinLSN* находится в виртуальном журнале 3. Виртуальные журналы 1 и 2 содержат только неактивные записи журнала. Эти записи можно усечь. Виртуальный журнал 5 пока не используется и не является частью текущего логического журнала.



На втором рисунке показан журнал после усечения. Виртуальные журналы 1 и 2 усечены и могут использоваться повторно. Логический журнал теперь начинается с начала виртуального журнала 3. Виртуальный журнал 5 все еще не используется и не является частью текущего логического журнала.

Управление размером файла журнала транзакций

Контролировать используемое пространство журнала можно с помощью процедуры DBCC SQLPERF (LOGSPACE). Она возвращает сведения об объеме пространства, используемого журналом в данный момент, и указывает, если необходимо провести усечение журнала транзакций. Для получения сведений о текущем размере файла журнала, его максимальном размере и параметре автоматического увеличения файла можно использовать столбцы **size, max_size** и **growth** для данного файла журнала в представлении **sys.database_files**.

Сжатие файла журнала

Усечение журнала освобождает место на диске для повторного использования, но не уменьшает размер физического файла журнала. Для уменьшения физического размера файл журнала должен быть сжат с целью удаления одного или более неактивных *VLF*. *VLF*, хранящий какие-либо активные записи журнала, удалить нельзя. При сжатии файла журнала транзакций в конце файла журнала удаляется достаточное количество неактивных виртуальных файлов журнала, чтобы журнал уменьшился до приблизительного целевого размера.

Примечание. Если журнал транзакций не усекался в последнее время, его сжатие может быть невозможным до тех пор, пока не выполнится усечение.