CREATE TRIGGER (Transact-SQL)

SQL Server 2005 | Другие версии

Изменения: **17 июля 2006 г.**

Создает триггер языка обработки данных, DDL или входа. Триггер — это особая разновидность хранимой процедуры, выполняемая автоматически при возникновении события на сервере базы данных. Триггеры языка обработки данных выполняются по событиям, вызванным попыткой пользователя изменить данные с помощью языка обработки данных. Событиями DML являются процедуры INSERT, UPDATE или DELETE, применяемые к таблице или представлению.

Примечание.

Эти триггеры срабатывают при запуске любого допустимого события независимо от того, затрагивает ли оно какие-либо строки таблицы. Это предусмотрено разработчиками.

Триггеры DDL срабатывают в ответ на ряд событий языка определения данных (DDL). Эти события прежде всего соответствуют инструкциям Transact-SQL CREATE, ALTER, DROP и некоторым системным хранимым процедурам, которые выполняют схожие с DDL операции. Триггеры входа могут срабатывать в ответ на событие LOGON, возникающее при установке пользовательских сеансов. В компоненте SQL Server 2005 Database Engine триггеры могут быть созданы непосредственно из инструкций Transact-SQL или из методов сборок, созданных в среде CLR платформы Microsoft.NET Framework, и переданы на экземпляр SQL Server. SQL Server допускает создание нескольких триггеров для любой указанной инструкции.

Примечание безопасности.

Вредоносный программный код внутри триггеров может быть запущен с расширенными правами доступа. Дополнительные сведения о снижении вероятности возникновения этой угрозы см. в разделе Управление безопасностью триггеров.

Соглашения о синтаксическом обозначении в Transact-SQL

Синтаксис

```
Trigger on an INSERT, UPDATE, or DELETE statement to a table or view (DML Trigger)
CREATE TRIGGER [ schema_name . ]trigger_name
ON { table | view }
[ WITH <dml_trigger_option> [ ,...n ] ]
{ FOR | AFTER | INSTEAD OF }
{ [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ] }
[ WITH APPEND ]
[ NOT FOR REPLICATION ]
AS { sql_statement  [ ; ] [ ,...n ] | EXTERNAL NAME <method specifier [ ; ] > }

<dml_trigger_option> ::=
    [ ENCRYPTION ]
    [ EXECUTE AS Clause ]

<method_specifier> ::=
    assembly_name.class_name.method_name

Trigger on a CREATE, ALTER, DROP, GRANT, DENY, REVOKE, or UPDATE STATISTICS statement (DDL Trigger)
CREATE TRIGGER trigger_name
ON { ALL SERVER | DATABASE }
[ WITH <ddl_trigger_option> [ ,...n ] ]
{ FOR | AFTER } { event_type | event_group } [ ,...n ]
AS { sql_statement  [ ; ] [ ,...n ] | EXTERNAL NAME < method specifier >  [ ; ] }

<ddl_trigger_option> ::=
```

```
    [ ENCRYPTION ]
    [ EXECUTE AS Clause ]

<method_specifier> ::=
    assembly_name.class_name.method_name

Trigger on a LOGON event (Logon Trigger)
CREATE TRIGGER trigger_name
ON ALL SERVER
[ WITH <logon_trigger_option> [ ,...n ] ]
{ FOR | AFTER } LOGON
AS { sql_statement  [ ; ] [ ,...n ] | EXTERNAL NAME <method specifier>  [ ; ] }
<logon_trigger_option> ::=
    [ ENCRYPTION ]
    [ EXECUTE AS Clause ]

<method_specifier> ::=
    assembly_name.class_name.method_name
```

Аргументы

schema_name
Имя схемы, которой принадлежит триггер DML. Триггеры DML ограничены областью схемы таблицы или представления, для которых они созданы. Аргумент *schema_name* не может быть указан для триггеров DDL или входа.

trigger_name
Имя триггера. Аргумент *trigger_name* должен соответствовать правилам для идентификаторов — за исключением того, что он не может начинаться с символов # или ##.

table | *view*
Таблица или представление, в которых выполняется триггер DML, иногда указывается как таблица триггера или представление триггера. Указание уточненного имени таблицы или представления не является обязательным. На представление может ссылаться только триггер INSTEAD OF. Триггеры DML не могут быть описаны в локальной или глобальной временных таблицах.

DATABASE
Применяет область действия триггера DDL к текущей базе данных. Если этот аргумент определен, триггер срабатывает всякий раз при возникновении в базе данных события типа *event_type* или *event_group*.

ALL SERVER
Применяет область действия триггера DDL или триггера входа к текущему серверу. Если этот аргумент определен, триггер срабатывает всякий раз при возникновении в любом месте на текущем сервере события типа *event_type* или *event_group*.

WITH ENCRYPTION
Затемняет текст инструкции CREATE TRIGGER. Использование аргумента WITH ENCRYPTION не позволяет публиковать триггер как часть репликации SQL Server. Параметр WITH ENCRYPTION не может быть указан для триггеров CLR.

EXECUTE AS
Указывает контекст безопасности, в котором выполняется триггер. Позволяет управлять учетной записью пользователя, используемой экземпляром SQL Server для проверки разрешений на любые объекты базы данных, ссылаемые триггером.

Дополнительные сведения см. в разделе EXECUTE AS, предложение (Transact-SQL).

FOR | AFTER
Тип AFTER указывает, что триггер срабатывает только после успешного выполнения всех операций в инструкции SQL, запускаемой триггером. Все каскадные действия и проверки ограничений, на которые имеется ссылка, должны быть успешно завершены, прежде чем триггер сработает.

Если единственным заданным ключевым словом является FOR, аргумент AFTER используется по умолчанию.

Триггеры AFTER не могут быть определены на представлениях.

INSTEAD OF
Указывает, что триггер DML срабатывает *вместо* инструкции SQL, используемой триггером, переопределяя таким образом действия выполняемой инструкции триггера. Аргумент INSTEAD OF не может быть указан для триггеров DDL или триггеров входа.

На каждую инструкцию INSERT, UPDATE или DELETE в таблице или представлении может быть определено не более одного триггера INSTEAD OF. Однако можно определить представления на представлениях, где у каждого представления есть

собственный триггер INSTEAD OF.

Триггеры INSTEAD OF не разрешены для обновляемых представлений, использующих параметр WITH CHECK OPTION. SQL Server вызывает ошибку, если триггер INSTEAD OF добавляется к обновляемому представлению с параметром WITH CHECK OPTION. Пользователь должен удалить этот параметр при помощи инструкции ALTER VIEW перед определением триггера INSTEAD OF.

{ [DELETE] [,] [INSERT] [,] [UPDATE] }

Определяет инструкции изменения данных, по которым срабатывает триггер DML, если он применяется к таблице или представлению. Необходимо указать как минимум одну инструкцию. В определении триггера разрешены любые их сочетания в любом порядке.

Для триггеров INSTEAD OF параметр DELETE не разрешен в таблицах, имеющих ссылочную связь с указанием каскадного действия ON DELETE. Точно так же параметр UPDATE не разрешен в таблицах, имеющих ссылочную связь с указанием каскадного действия ON UPDATE.

event_type

Имя события языка Transact-SQL, которое после выполнения вызывает срабатывание триггера DDL. Список событий, которые могут быть использованы в триггерах DDL, приведен в разделе DDL-события, вызывающие срабатывание триггеров DDL.

event_group

Имя стандартной группы событий языка Transact-SQL. Триггер DDL срабатывает после возникновения любого события языка Transact-SQL, принадлежащего к группе *event_group*. Список групп событий, которые могут быть использованы в триггерах DDL, приведен в разделе Группы событий для использования с триггерами DDL.

После завершения инструкции CREATE TRIGGER параметр *event_group* работает в режиме макроса, добавляя охватываемые им типы события в представление каталога **sys.trigger_events**.

WITH APPEND

Указывает, что требуется добавить триггер существующего типа. Использование этого необязательного предложения необходимо только при уровне совместимости 65 и ниже. Если уровень совместимости 70 и выше, для добавления дополнительного триггера существующего типа предложение WITH APPEND не требуется. Данное поведение используется по умолчанию для инструкции CREATE TRIGGER с настройкой уровня совместимости 70 и выше. Дополнительные сведения см. в разделе sp_dbcmptlevel (Transact-SQL).

Предложение WITH APPEND не может быть использовано для триггеров INSTEAD OF или при явном указании триггера AFTER. Предложение WITH APPEND может использоваться только при указании параметра FOR без INSTEAD OF или AFTER из соображений поддержки обратной совместимости. Предложение WITH APPEND не может быть указано, если указан параметр EXTERNAL NAME (в случае триггера CLR).

Важно!

Предложение WITH APPEND будет исключено в следующей версии Microsoft SQL Server. Избегайте использования предложения WITH APPEND в новых разработках и запланируйте изменение приложений, которые используют его в настоящий момент.

NOT FOR REPLICATION

Указывает, что триггер не может быть выполнен, если агент репликации изменяет таблицу, используемую триггером. Дополнительные сведения см. в разделе Управление ограничениями, идентификаторами и триггерами с помощью параметра «NOT FOR REPLICATION».

sql_statement

Условия и действия триггера. Условия триггера указывают дополнительные критерии, определяющие, какие события — DML, DDL или событие входа — вызывают срабатывание триггера.

Действия триггера, указанные в инструкциях языка Transact-SQL, вступают в силу после попытки использования операции.

Триггеры могут содержать любое количество инструкций языка Transact-SQL любого типа, за некоторыми исключениями. Дополнительные сведения см. в разделе «Примечания». Триггеры разработаны для контроля или изменения данных на основании инструкций модификации или определения данных; они не возвращают пользователю никаких данных. Инструкции языка Transact-SQL в составе триггера часто содержат выражения языка управления потоком.

Триггеры DML используют логические (концептуальные) таблицы **deleted** и **inserted**. По своей структуре они подобны таблице, на которой определен триггер, то есть таблице, к которой применяется действие пользователя. В таблицах **deleted** и **inserted** содержатся старые или новые значения строк, которые могут быть изменены действиями пользователя. Например, для запроса всех значений таблицы `deleted` можно использовать инструкцию:

```
SELECT *
FROM deleted
```

Дополнительные сведения см. в разделе Использование таблиц inserted и deleted.

Триггеры DDL и триггеры входа собирают сведения о событиях, запускаемых с помощью функции EVENTDATA (Transact-SQL). Дополнительные сведения см. в разделе Использование функции EVENTDATA.

SQL Server не допускает использование ссылок на столбцы типа **text**, **ntext** или **image** таблиц **inserted** и **deleted** в триггерах DELETE, INSERT или UPDATE, если установлен уровень совместимости 70. Доступ к значениям столбцов типа **text**, **ntext** и **image** в таблицах **inserted** и **deleted** невозможен. Для получения новых значений для триггера INSERT или UPDATE соедините таблицу **inserted** с исходной обновляемой таблицей. При уровне совместимости 65 и меньше для столбцов типа **text**, **ntext** или **image**, допускающих значения NULL, из таблиц **inserted** или **deleted** возвращаются значения NULL; если в этих столбцах значения NULL недопустимы, возвращаются пустые строки.

При уровне совместимости 80 и выше SQL Server разрешает обновление столбцов типа **text**, **ntext** или **image** с помощью триггера INSTEAD OF, применяемого к таблицам или представлениям.

Важно!

Типы данных **ntext**, **text** и **image** будут удалены в следующей версии Microsoft SQL Server. Следует избегать использования этих типов данных при новой разработке и запланировать изменение приложений, использующих их в настоящий момент. Вместо них следует использовать типы данных nvarchar(max), varchar(max) и varbinary(max). Как триггеры AFTER, так и триггеры INSTEAD OF поддерживают данные типов **varchar(MAX)**, **nvarchar(MAX)** и **varbinary(MAX)** в таблицах **inserted** и **deleted**.

< method_specifier >

Указывает метод сборки для связывания с CLR-триггером. Метод не должен иметь аргументов и возвращать значение типа void. Аргумент *class_name* должен быть допустимым идентификатором SQL Server, а в сборке должен существовать класс с таким именем, видимый во всей сборке. Если класс имеет имя, содержащее точки («.») для разделения частей пространства имен, имя класса должно быть заключено в квадратные скобки ([ ]) или двойные кавычки (" "). Класс не может быть вложенным.

Примечание.

По умолчанию возможность SQL Server запускать код CLR отключена. Можно создавать, изменять и удалять объекты базы данных, которые ссылаются на модули управляемого кода, но эти ссылаемые модули не будут выполнены на экземплярах SQL Server, пока параметр clr enabled не будет включен с помощью процедуры sp_configure.

Замечания

Триггеры DML

Триггеры DML часто используются для соблюдения бизнес-правил и целостности данных. В SQL Server декларативное ограничение ссылочной целостности обеспечивается инструкциями ALTER TABLE и CREATE TABLE. Однако декларативное ограничение ссылочной целостности не обеспечивает ссылочную целостность между базами данных. Ограничение ссылочной целостности подразумевает выполнение правил связи между первичными и внешними ключами таблиц. Для обеспечения ограничений ссылочной целостности используйте в инструкциях ALTER TABLE и CREATE TABLE ограничения PRIMARY KEY и FOREIGN KEY. Если ограничения распространяются на таблицу триггера, они проверяются после срабатывания триггера INSTEAD OF и до выполнения триггера AFTER. В случае нарушения ограничения выполняется откат действий триггера INSTEAD OF, и триггер AFTER не срабатывает.

Первые и последние триггеры AFTER, которые будут выполнены в таблице, могут быть определены с использованием процедуры **sp_settriggerorder**. Для таблицы можно определить только один первый и один последний триггер для каждой из операций INSERT, UPDATE и DELETE. Если в таблице есть другие триггеры AFTER, они будут выполняться случайным образом.

Если инструкция ALTER TRIGGER меняет первый или последний триггер, первый или последний набор характеристик измененного триггера удаляется, а порядок сортировки должен быть установлен заново с помощью процедуры **sp_settriggerorder**.

Триггер AFTER выполняется только после того, как вызывающая срабатывание триггера инструкция SQL была успешно выполнена. Успешное выполнение также подразумевает завершение всех ссылочных каскадных действий и проверки ограничений, связанных с измененными или удаленными объектами.

Если триггер INSTEAD OF, определенный для таблицы, выполняет по отношению к таблице какую-либо инструкцию, которая бы снова вызвала срабатывание триггера INSTEAD OF, триггер рекурсивно не вызывается. Вместо этого инструкция обрабатывается так, как если бы у таблицы отсутствовал триггер INSTEAD OF, и начинается применение последовательности ограничений и выполнение триггера AFTER. Например, если триггер определен в виде триггера INSTEAD OF INSERT для таблицы и выполняет инструкцию INSERT для этой же таблицы, инструкция INSERT не вызывает нового срабатывания триггера. Команда INSERT, выполняемая триггером, начинает процесс действий применения ограничений и обработки всех триггеров AFTER INSERT, определенных для данной таблицы.

Если триггер INSTEAD OF, определенный для представления, выполняет по отношению к представлению какую-либо инструкцию,

которая бы снова вызвала срабатывание триггера INSTEAD OF, триггер рекурсивно не вызывается. Вместо этого инструкция выполняет изменение базовых таблиц, на которых основано представление. В данном случае определение представления должно удовлетворять всем ограничениям, установленным для обновляемых представлений. Сведения об определении обновляемых представлений см. в разделе Изменение данных через представление.

Например, если триггер определен как INSTEAD OF UPDATE для представления и выполняет инструкцию UPDATE для этого же представления, инструкция UPDATE, выполняемая триггером, не вызывает нового срабатывания триггера. Инструкция UPDATE, выполняемая в триггере, обрабатывает представление так, как если бы в представлении не имелось триггера INSTEAD OF. Столбцы, измененные с помощью инструкции UPDATE, должны разрешаться в одну базовую таблицу. Каждая модификация базовой таблицы вызывает применение последовательности ограничений и взвод триггеров AFTER, определенных для данной таблицы.

Проверка действий инструкций UPDATE или INSERT на указанные столбцы

Триггер языка Transact-SQL можно сконструировать для выполнения конкретных действий, основанных на изменении определенных столбцов с помощью инструкций UPDATE или INSERT. Используйте для этих целей в теле триггера конструкции UPDATE() или COLUMNS_UPDATED. Конструкция UPDATE() проверяет действие инструкций UPDATE или INSERT на одном столбце. С помощью конструкции COLUMNS_UPDATED проверяются действия инструкций UPDATE или INSERT, проводимых на нескольких столбцах, и возвращается битовый шаблон, показывающий, какие столбцы были вставлены или обновлены.

Ограничения триггеров

Инструкция CREATE TRIGGER должна быть первой инструкцией в пакете и может применяться только к одной таблице.

Триггер создается только в текущей базе данных, но может, тем не менее, содержать ссылки на объекты за пределами текущей базы данных.

Если для уточнения триггера указано имя схемы, имя таблицы необходимо уточнить таким же образом.

Одно и то же действие триггера может быть определено более чем для одного действия пользователя (например, INSERT и UPDATE) в одной и той же инструкции CREATE TRIGGER.

Триггеры INSTEAD OF DELETE/UPDATE нельзя определить для таблицы, у которой есть внешний ключ, определенный для каскадного выполнения операции DELETE/UPDATE.

Внутри триггера может быть использована любая инструкция SET. Выбранный параметр SET остается в силе во время выполнения триггера, после чего настройки возвращаются в предыдущее состояние.

Во время срабатывания триггера результаты возвращаются вызывающему приложению так же, как и в случае с хранимыми процедурами. Чтобы предотвратить вызванное срабатыванием триггера возвращение результатов приложению, не следует включать инструкции SELECT, возвращающие результат, или инструкции, которые выполняют в триггере присвоение переменных. Триггер, содержащий либо инструкции SELECT, которые возвращают результаты пользователю, либо инструкции, выполняющие присвоение переменных, требует особого обращения; эти возвращаемые результаты должны быть перезаписаны во все приложения, в которых разрешены изменения таблицы триггера. Если в триггере происходит присвоение переменной, следует использовать инструкцию SET NOCOUNT в начале триггера, чтобы предотвратить возвращение каких-либо результирующих наборов.

Хотя инструкция TRUNCATE TABLE фактически является инструкцией DELETE, она не может запустить триггер, потому что операция не регистрирует в журнале удаления отдельных строк. Однако только пользователям с разрешением на выполнение в таблице инструкции TRUNCATE TABLE следует обращать внимание на непреднамеренный обход триггера DELETE с помощью инструкции TRUNCATE TABLE.

Инструкция WRITETEXT (с ведением журнала и без него) не запускает триггеры.

Следующие инструкции языка Transact-SQL не разрешены в триггерах DML:

ALTER DATABASE	CREATE DATABASE	DROP DATABASE
LOAD DATABASE	LOAD LOG	RECONFIGURE
RESTORE DATABASE	RESTORE LOG	

Кроме того, следующие инструкции языка Transact-SQL не разрешены в теле триггера DML, если он используется по отношению к таблице или представлению, которые являются целью действий триггера.

Важно!

Хотя это ограничение введено в SQL Server 2005, оно также применяется, если значение режима обратной совместимости равно 80.

CREATE INDEX	ALTER INDEX	DROP INDEX
DBCC DBREINDEX	ALTER PARTITION FUNCTION	DROP TABLE

ALTER TABLE, если используется в следующих целях: • Добавление, изменение или удаление столбцов. • Переключение секций. • Добавление или удаление ограничений PRIMARY KEY и UNIQUE.		

Примечание.

Поскольку SQL Server не поддерживает пользовательских триггеров в системных таблицах, рекомендуется не создавать пользовательские триггеры для системных таблиц.

Триггеры DDL

Триггеры DDL, как и стандартные триггеры, выполняют хранимые процедуры в ответ на какое-либо событие. В отличие от стандартных триггеров, они не срабатывают в ответ на выполнение инструкций UPDATE, INSERT или DELETE по отношению к таблице или представлению. Вместо этого триггеры срабатывают в первую очередь в ответ на инструкции языка определения данных (DDL). Это инструкции CREATE, ALTER, DROP, GRANT, DENY, REVOKE и UPDATE STATISTICS. Системные хранимые процедуры, выполняющие операции, подобные операциям DDL, также могут запускать триггеры DDL.

Важно!

Протестируйте триггеры DDL, чтобы получить ответ на выполнение системных хранимых процедур. Например, как инструкция CREATE TYPE, так и хранимая процедура **sp_addtype** вызывают срабатывание триггера DDL, созданного на событии CREATE_TYPE. Однако хранимая процедура **sp_rename** не запускает никаких триггеров DDL.

Дополнительные сведения о триггерах DDL см. в разделе Триггеры DDL.

Триггеры DDL не срабатывают в ответ на события, влияющие на локальные или глобальные временные таблицы и хранимые процедуры.

В отличие от триггеров DML, триггеры DDL не ограничены областью схемы. Поэтому для запроса метаданных о триггерах DDL нельзя воспользоваться функциями OBJECT_ID, OBJECT_NAME, OBJECTPROPERTY и OBJECTPROPERTYEX. Используйте вместо них представления каталога. Дополнительные сведения см. в разделе Получение сведений о триггерах DDL.

Примечание.

Серверные триггеры DDL находятся в папке **Триггеры** обозревателя объектов среды SQL Server Management Studio. Эта папка находится внутри папки **Объекты сервера**. Триггеры DDL, работающие в области базы данных, хранятся в папке **Триггеры базы данных**. Эта папка находится в папке **Программирование** соответствующей базы данных.

Триггеры входа

Триггеры входа срабатывают в ответ на событие LOGON. Событие вызывается при установке пользовательских сеансов. Дополнительные сведения см. в разделе Триггеры входа.

Общие соглашения о триггерах

Возвращаемые результаты

Возможность возвращать результаты из триггеров будет исключена из следующей версии SQL Server. Триггеры, возвращающие результирующие наборы, могут вызвать непредвиденное поведение тех приложений, которые не предназначены для работы с ними. Не используйте в разрабатываемых приложениях триггеры, возвращающие результирующие наборы, и запланируйте изменение приложений, которые используют их в настоящее время. Чтобы триггеры не возвращали результирующие наборы в SQL Server 2005, в параметре disallow results from triggers установите значение 1.

Триггеры LOGON всегда запрещают возврат результирующих наборов, и данное поведение нельзя настроить. Если триггер LOGON создает результирующий набор, то триггер не выполняется и попытка входа, вызванная триггером, запрещается.

Несколько триггеров

SQL Server позволяет создавать несколько триггеров для каждого события DML, DDL и LOGON. Например, если инструкция CREATE TRIGGER FOR UPDATE выполняется в таблице, уже имеющей триггер UPDATE, дополнительно создается триггер обновления. В более ранних версиях SQL Server был разрешен только один триггер в каждой таблице для каждого события изменения данных INSERT, UPDATE или DELETE.

Примечание.

При уровне совместимости 70 инструкция CREATE TRIGGER по умолчанию создает новые триггеры в дополнение к существующим,

если триггеры имеют разные имена. Если имена триггеров совпадают, SQL Server возвращает сообщение об ошибке. Однако если уровень совместимости 65 и меньше, новые триггеры, созданные инструкцией CREATE TRIGGER, заменяют существующие триггеры этого типа, даже если имена триггеров не совпадают. Дополнительные сведения см. в разделе sp_dbcmptlevel (Transact-SQL).

Рекурсивные триггеры

SQL Server разрешает рекурсивный вызов триггеров, если с помощью инструкции ALTER DATABASE включена настройка RECURSIVE_TRIGGERS.

В рекурсивных триггерах могут возникать следующие типы рекурсии:

- Косвенная рекурсия
  При косвенной рекурсии приложение обновляет таблицу **T1**. Это событие вызывает срабатывание триггера **TR1**, обновляющего таблицу **T2**. Это вызывает срабатывание триггера **T2** и обновление таблицы **T1**.
- Прямая рекурсия
  При прямой рекурсии приложение обновляет таблицу **T1**. Это событие вызывает срабатывание триггера **TR1**, обновляющего таблицу **T1**. Поскольку таблица **T1** уже была обновлена, триггер **TR1** срабатывает снова и т.д.

В следующем примере используются оба типа рекурсий: прямая и косвенная. Допустим, для таблицы **T1** определены два триггера: **TR1** и **TR2**. Триггер **TR1** рекурсивно обновляет таблицу **T1**. Инструкция UPDATE выполняет каждый из триггеров **TR1** и **TR2** один раз. В дополнение к этому срабатывание триггера **TR1** вызывает выполнение триггеров **TR1** (рекурсивно) и **TR2**. В таблицах **inserted** и **deleted** триггера содержатся строки, которые относятся только к инструкции UPDATE, вызвавшей срабатывание триггера.

Примечание.

Описанная ситуация имеет место только в том случае, если настройка RECURSIVE_TRIGGERS включена с помощью инструкции ALTER DATABASE. Определенного порядка выполнения нескольких триггеров, заданных для какого-либо конкретного события, не существует. Каждый триггер должен быть самодостаточным.

Отключение настройки RECURSIVE_TRIGGERS предотвращает выполнение только прямых рекурсий. Чтобы отключить косвенную рекурсию, с помощью хранимой процедуры **sp_configure** присвойте параметру сервера **nested triggers** значение 0.

Если один из триггеров выполняет инструкцию ROLLBACK TRANSACTION, никакие другие триггеры, вне зависимости от уровня вложенности, не срабатывают.

Вложенные триггеры

Вложенность триггеров может достигать максимум 32 уровня. Если триггер изменяет таблицу, для которой определен другой триггер, то запускается второй триггер, вызывающий срабатывание третьего и т.д. Если любой из триггеров в цепочке отключает бесконечный цикл, то уровень вложенности превышает допустимый предел, и срабатывание триггера отменяется. Чтобы отменить вложенные триггеры, присвойте значение 0 параметру **nested triggers** хранимой процедуры **sp_configure**. В конфигурации по умолчанию вложенные триггеры разрешены. Если вложенные триггеры отключены, рекурсивные триггеры тоже будут отключены, вне зависимости от настройки RECURSIVE_TRIGGERS, установленной с помощью инструкции ALTER DATABASE.

Примечание.

Если триггер на языке Transact-SQL выполняет управляемый код с помощью ссылки на метод, тип или статистическую функцию среды CLR, эта ссылка считается одним из допустимых 32 уровней вложенности. Методы, вызываемые из управляемого кода, не учитываются в этом ограничении.

Отложенная интерпретация имен

В SQL Server разрешены хранимые процедуры, триггеры и пакеты на языке Transact-SQL, которые содержат ссылки на таблицы, не существующие в момент компиляции. Такая возможность называется отложенной интерпретацией имен. Однако если хранимая процедура, триггер или пакет на языке Transact-SQL содержат ссылку на таблицу, определенную в хранимой процедуре или триггере, то во время создания выдается предупреждение, только если установлен уровень совместимости 65. Предупреждение во время компиляции возникает, если используется пакет. Если таблица, на которую имеется ссылка, не существует во время выполнения, возникает сообщение об ошибке. Дополнительные сведения см. в разделе Отсроченное разрешение и компиляция имен.

Разрешения

Для создания триггера DML требуется разрешение ALTER на таблицу или представление, в которых создается триггер.

Для создания триггера DDL с областью действия в пределах сервера (ON ALL SERVER) или триггера входа требуется разрешение CONTROL SERVER на сервер. Для создания триггера DDL с областью видимости в пределах базы данных (ON DATABASE) требуется разрешение ALTER ANY DATABASE DDL TRIGGER на текущую базу данных.

Примеры

A. Использование триггера DML с предупреждающим сообщением

Следующий триггер DML отправляет клиенту сообщение, когда кто-то пытается добавить или изменить данные в таблице `Customer`.

```
USE AdventureWorks;
GO
IF OBJECT_ID ('Sales.reminder1', 'TR') IS NOT NULL
   DROP TRIGGER Sales.reminder1;
GO
CREATE TRIGGER reminder1
ON Sales.Customer
AFTER INSERT, UPDATE
AS RAISERROR ('Notify Customer Relations', 16, 10);
GO
```

Б. Использование триггера DML с предупреждающим сообщением, отправляемым по электронной почте

В следующем примере указанному пользователю (`MaryM`) по электронной почте отправляется сообщение при изменении таблицы `Customer`.

```
USE AdventureWorks;
GO
IF OBJECT_ID ('Sales.reminder2','TR') IS NOT NULL
    DROP TRIGGER Sales.reminder2;
GO
CREATE TRIGGER reminder2
ON Sales.Customer
AFTER INSERT, UPDATE, DELETE
AS
   EXEC msdb.dbo.sp_send_dbmail
        @profile_name = 'AdventureWorks Administrator',
        @recipients = 'danw@Adventure-Works.com',
        @body = 'Don''t forget to print a report for the sales force.',
        @subject = 'Reminder';
GO
```

B. Использование триггера DML AFTER для применения бизнес-правил между таблицами PurchaseOrderHeader и Vendor

Поскольку ограничение CHECK может содержать ссылки только на столбцы, для которых определены ограничения на уровне столбцов или таблицы, любые межтабличные ограничения (в данном случае бизнес-правила) должны быть заданы в виде триггеров.

В следующем примере создается триггер DML. Этот триггер проверяет уровень кредитоспособности поставщика при попытке добавить новый заказ на покупку в таблицу `PurchaseOrderHeader`. Для получения сведений о кредитоспособности поставщика требуется ссылка на таблицу `Vendor`. В случае слишком низкой кредитоспособности выводится соответствующее сообщение, и вставка не производится.

Примечание.

Примеры триггеров DML AFTER, которые обновляют несколько строк, см. в разделе Замечания по работе с несколькими строками в триггерах DML. Примеры триггеров DML INSTEAD OF INSERT см. в разделе Триггеры INSTEAD OF INSERT.

```
IF OBJECT_ID ('Purchasing.LowCredit','TR') IS NOT NULL
   DROP TRIGGER Purchasing.LowCredit;
GO
CREATE TRIGGER LowCredit ON Purchasing.PurchaseOrderHeader
AFTER INSERT
AS
DECLARE @creditrating tinyint,
   @vendorid int
SELECT @creditrating = v.CreditRating, @vendorid = p.VendorID
FROM Purchasing.PurchaseOrderHeader AS p
```

```
    INNER JOIN inserted AS i ON p.PurchaseOrderID =
   i.PurchaseOrderID
    JOIN Purchasing.Vendor AS v on v.VendorID = i.VendorID
IF @creditrating = 5
BEGIN
   RAISERROR ('This vendor''s credit rating is too low to accept new
      purchase orders.', 16, 1)
ROLLBACK TRANSACTION
END
GO
```

Г. Использование отложенной интерпретации имен

В следующем примере для демонстрации отложенной интерпретации имен создаются два триггера DML.

```
USE AdventureWorks;
GO
IF OBJECT_ID ('HumanResources.trig1','TR') IS NOT NULL
   DROP TRIGGER HumanResources.trig1;
GO
-- Creating a trigger on a nonexistent table.
CREATE TRIGGER HumanResources.trig1
on HumanResources.Employee
AFTER INSERT, UPDATE, DELETE
AS
   SELECT e.EmployeeID, e.BirthDate, x.info
   FROM HumanResources.Employee AS e INNER JOIN does_not_exist AS x
      ON e.EmployeeID = x.xID
GO
-- Here is the statement to actually see the text of the trigger.
SELECT t.object_id, m.definition
FROM sys.triggers AS t INNER JOIN sys.sql_modules AS m
   ON t.object_id = m.object_id
WHERE t.type = 'TR' and t.name = 'trig1'
AND t.parent_class = 1
GO

-- Creating a trigger on an existing table, but with a nonexistent
-- column.
USE AdventureWorks;
GO
IF OBJECT_ID ('HumanResources.trig2','TR') IS NOT NULL
   DROP TRIGGER HumanResources.trig2
GO
CREATE TRIGGER HumanResources.trig2
ON HumanResources.Employee
AFTER INSERT, UPDATE
AS
   DECLARE @fax varchar(12)
   SELECT @fax = 'AltPhone'
   FROM HumanResources.Employee
GO
-- Here is the statement to actually see the text of the trigger.
SELECT t.object_id, m.definition
FROM sys.triggers AS t INNER JOIN sys.sql_modules AS m
   ON t.object_id = m.object_id
WHERE t.type = 'TR' and t.name = 'trig2'
AND t.parent_class = 1
GO
```

Д. Использование триггера DDL с областью видимости в пределах базы данных

В следующем примере триггер DDL используется для предотвращения удаления синонимов в базе данных.

```
USE AdventureWorks;
GO
IF EXISTS (SELECT * FROM sys.triggers
    WHERE parent_class = 0 AND name = 'safety')
DROP TRIGGER safety
ON DATABASE;
GO
CREATE TRIGGER safety
```

```
ON DATABASE
FOR DROP_SYNONYM
AS
   RAISERROR ('You must disable Trigger "safety" to drop synonyms!',10, 1)
   ROLLBACK
GO
DROP TRIGGER safety
ON DATABASE;
GO
```

E. Использование триггера DDL с областью видимости в пределах сервера

В следующем примере триггер DDL используется для вывода сообщения при возникновении на данном экземпляре сервера любого из событий CREATE DATABASE, а функция **EVENTDATA** используется для получения текста соответствующей инструкции на языке Transact-SQL.

Примечание.

Примеры использования функции EVENTDATA в триггерах DDL см. в разделе Использование функции EVENTDATA.

```
IF EXISTS (SELECT * FROM sys.server_triggers
    WHERE name = 'ddl_trig_database')
DROP TRIGGER ddl_trig_database
ON ALL SERVER;
GO
CREATE TRIGGER ddl_trig_database
ON ALL SERVER
FOR CREATE_DATABASE
AS
    PRINT 'Database Created.'
    SELECT EVENTDATA().value('(/EVENT_INSTANCE/TSQLCommand/CommandText)[1]','nvarchar(max)')
GO
DROP TRIGGER ddl_trig_database
ON ALL SERVER;
GO
```

Ж. Использование триггера входа

В следующем примере триггера входа выполняется запрет попытки подключения к SQL Server в качестве члена *login_test* учетной записи, если под этой учетной записью уже запущено три сеанса.

```
USE master;
GO
CREATE LOGIN login_test WITH PASSWORD = '3KHJ6dhx(0xVYsdf' MUST_CHANGE,
    CHECK_EXPIRATION = ON;
GO
GRANT VIEW SERVER STATE TO login_test;
GO
CREATE TRIGGER connection_limit_trigger
ON ALL SERVER WITH EXECUTE AS 'login_test'
FOR LOGON
AS
BEGIN
IF ORIGINAL_LOGIN()= 'login_test' AND
    (SELECT COUNT(*) FROM sys.dm_exec_sessions
            WHERE is_user_process = 1 AND
                original_login_name = 'login_test') > 3
    ROLLBACK;
END;
```

3. Просмотр событий, вызвавших срабатывание триггера

В следующем примере выполняются запросы к представлениям каталога `sys.triggers` и `sys.trigger_events` с целью определения, какие события языка Transact-SQL вызывали срабатывание триггера `safety`. Создание триггера `safety` показано в предыдущем примере.

```
SELECT TE.*
FROM sys.trigger_events AS TE
JOIN sys.triggers AS T
ON T.object_id = TE.object_id
WHERE T.parent_class = 0
AND T.name = 'safety'
GO
```

См. также

Справочник
ALTER TABLE (Transact-SQL)
ALTER TRIGGER (Transact-SQL)
COLUMNS_UPDATED (Transact-SQL)
CREATE TABLE (Transact-SQL)
DROP TRIGGER (Transact-SQL)
ENABLE TRIGGER (Transact-SQL)
DISABLE TRIGGER (Transact-SQL)
TRIGGER_NESTLEVEL (Transact-SQL)
EVENTDATA (Transact-SQL)
sp_depends (Transact-SQL)
sys.sql_dependencies (Transact-SQL)
sp_help (Transact-SQL)
sp_helptrigger (Transact-SQL)
sp_helptext (Transact-SQL)
sp_rename (Transact-SQL)
sp_settriggerorder (Transact-SQL)
UPDATE() (Transact-SQL)
sys.triggers (Transact-SQL)
sys.trigger_events (Transact-SQL)
sys.sql_modules (Transact-SQL)
sys.assembly_references (Transact-SQL)
sys.server_triggers
sys.server_trigger_events
sys.server_sql_modules
sys.server_assembly_modules (Transact-SQL)

Другие ресурсы
Создание хранимых процедур (компонент Database Engine)
Программирование триггеров CLR
Использование идентификаторов в качестве имен объектов
Получение сведений о триггерах DML
Получение сведений о триггерах DDL
Управление ограничениями, идентификаторами и триггерами с помощью параметра «NOT FOR REPLICATION»
Использование типов данных больших значений

Справка и поддержка
Получение помощи по SQL Server 2005

Журнал изменений

Версия	Журнал
12 декабря 2006 г.	**Новое содержимое** • Добавлены сведения по всему разделу о триггерах входа, введенных в SQL Server 2005 с пакетом обновления 2.
17 июля 2006 г.	**Изменения** • Обновленные примеры Д и Е.
14 апреля 2006 г.	**Добавления** • Добавлено важное примечание в раздел «Примечания», которое рекомендует протестировать триггеры DDL для проверки ответов на выполнение хранимых процедур.

	• Добавлено примечание в пример B со ссылками на примеры триггеров DML AFTER, которые обновляют несколько строк и триггеры DML INSTEAD OF INSERT. • Добавлено примечание в пример E со ссылками на большее количество примеров использования функции EVENTDATA в триггерах DDL. **Изменения** • Обновлен пример Б для использования компонента Database Mail.
5 декабря 2005 г.	**Добавления** • Добавленные триггеры DML не могут быть описаны в локальной или глобальной временных таблицах. • Добавленные триггеры DDL не срабатывают в ответ на события, влияющие на локальные или глобальные временные таблицы и хранимые процедуры.

Добавления сообщества ДОБАВИТЬ