**DDL Triggers**

**SQL Server 2012**

[Other Versions](javascript:;)

Description: http://i.technet.microsoft.com/Areas/Epx/Content/Images/ImageSprite.png

* [SQL Server 2008 R2](http://technet.microsoft.com/en-us/library/ms175941(d=printer,v=sql.105).aspx)
* [SQL Server 2008](http://technet.microsoft.com/en-us/library/ms175941(d=printer,v=sql.100).aspx)
* [SQL Server 2005](http://technet.microsoft.com/en-us/library/ms175941(d=printer,v=sql.90).aspx)

DDL triggers fire in response to a variety of Data Definition Language (DDL) events. These events primarily correspond to Transact-SQL statements that start with the keywords CREATE, ALTER, DROP, GRANT, DENY, REVOKE or UPDATE STATISTICS. Certain system stored procedures that perform DDL-like operations can also fire DDL triggers.

Use DDL triggers when you want to do the following:

* Prevent certain changes to your database schema.
* Have something occur in the database in response to a change in your database schema.
* Record changes or events in the database schema.

|  |
| --- |
| **Description: Important noteImportant** |
| Test your DDL triggers to determine their responses to system stored procedures that are run. For example, the CREATE TYPE statement and the **sp\_addtype** stored procedure will both fire a DDL trigger that is created on a CREATE\_TYPE event. |

[Types of DDL Triggers](javascript:void(0))

Transact-SQL DDL Trigger

A special type of Transact-SQL stored procedure that executes one more more Transact-SQL statements in response to a server-scoped or database-scoped event. For example, a DDL Trigger may fire if a statement such as ALTER SERVER CONFIGURATION is executed or if a table is deleted by using DROP TABLE.

CLR DDL Trigger

Instead of executing a Transact-SQL stored procedure, a CLR trigger executes one or more methods written in managed code that are members of an assembly created in the .NET Framework and uploaded in SQL Server.

DDL triggers fire only after the DDL statements that trigger them are run. DDL triggers cannot be used as INSTEAD OF triggers. DDL triggers do not fire in response to events that affect local or global temporary tables and stored procedures.

DDL triggers do not create the special inserted and deleted tables.

The information about an event that fires a DDL trigger, and the subsequent changes caused by the trigger, is captured by using the EVENTDATA function.

Multiple triggers to be created for each DDL event.

Unlike DML triggers, DDL triggers are not scoped to schemas. Therefore, functions such as OBJECT\_ID, OBJECT\_NAME, OBJECTPROPERTY, and OBJECTPROPERTYEX cannot be used for querying metadata about DDL triggers. Use the catalog views instead.

Server-scoped DDL triggers appear in the SQL Server Management Studio Object Explorer in the Triggers folder. This folder is located under the Server Objects folder. Database-scoped DDL triggers appear in the Database Triggers folder. This folder is located under the Programmability folder of the corresponding database.

|  |
| --- |
| **Description: Security noteSecurity Note** |
| Malicious code inside triggers can run under escalated privileges. For more information about how to help reduce this threat, see [Manage Trigger Security](http://technet.microsoft.com/en-us/library/e94720a8-a3a2-4364-b0a3-bbe86e3ce4d5). |

[DDL Trigger Scope](javascript:void(0))

DDL triggers can fire in response to a Transact-SQL event processed in the current database, or on the current server. The scope of the trigger depends on the event. For example, a DDL trigger created to fire in response to a CREATE\_TABLE event can do so whenever a CREATE\_TABLE event occurs in the database, or on the server instance. A DDL trigger created to fire in response to a CREATE\_LOGIN event can do so only when a CREATE\_LOGIN event occurs in the server instance.

In the following example, DDL trigger safety will fire whenever a DROP\_TABLE or ALTER\_TABLE event occurs in the database.

CREATE TRIGGER safety

ON DATABASE

FOR DROP\_TABLE, ALTER\_TABLE

AS

PRINT 'You must disable Trigger "safety" to drop or alter tables!'

ROLLBACK;

In the following example, a DDL trigger prints a message if any CREATE\_DATABASE event occurs on the current server instance. The example uses the EVENTDATA function to retrieve the text of the corresponding Transact-SQL statement. For more information about how to use EVENTDATA with DDL triggers, see [Use the EVENTDATA Function](http://technet.microsoft.com/en-us/library/675b8320-9c73-4526-bd2f-91ba42c1b604).

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

The lists that map the Transact-SQL statements to the scopes that can be specified for them are available through the links provided in the section "Selecting a Particular DDL Statement to Fire a DDL Trigger," later in this topic.

Database-scoped DDL triggers are stored as objects in the database in which they are created. DDL triggers can be created in the **master** database and behave just like those created in user-designed databases. You can obtain information about DDL triggers by querying the **sys.triggers** catalog view. You can query **sys.triggers** within the database context in which the triggers are created or by specifying the database name as an identifier, such as **master.sys.triggers**.

Server-scoped DDL triggers are stored as objects in the **master** database. However, you can obtain information about server-scoped DDL triggers by querying the **sys.server\_triggers** catalog view in any database context.

[Specifying a Transact-SQL Statement or Group of Statements](javascript:void(0))

**Selecting a Particular DDL Statement to Fire a DDL Trigger**

DDL triggers can be designed to fire after one or more particular Transact-SQL statements are run. In the previous example, trigger safety fires after any DROP\_TABLE or ALTER\_TABLE event. For lists of the Transact-SQL statements that can be specified to fire a DDL trigger, and the scope at which the trigger can fire, see [DDL Events](http://technet.microsoft.com/en-us/library/62ef24b4-3553-4aed-b62a-670980bae501).

**Selecting a Predefined Group of DDL Statements to Fire a DDL Trigger**

A DDL trigger can fire after execution of any Transact-SQL event that belongs to a predefined grouping of similar events. For example, if you want a DDL trigger to fire after any CREATE TABLE, ALTER TABLE, or DROP TABLE statement is run, you can specify FOR DDL\_TABLE\_EVENTS in the CREATE TRIGGER statement. After CREATE TRIGGER is run, the events that are covered by an event group are added to the **sys.trigger\_events** catalog view.

In SQL Server 2005, if a trigger is created on an event group, **sys.trigger\_events** does not include information about the event group, **sys.trigger\_events** includes information only about the individual events covered by that group. In SQL Server 2008 and higher, **sys.trigger\_events** persists metadata about the event group on which the triggers is created, and also about the individual events that the event group covers. Therefore, changes to the events that are covered by event groups in SQL Server 2008 and higher do not apply to DDL triggers that are created on those event groups in SQL Server 2005.

For a list of the predefined groups of DDL statements that are available for DDL triggers, the particular statements the event groups cover, and the scopes at which these event groups can be programmed, see [DDL Event Groups](http://technet.microsoft.com/en-us/library/12b45cc3-2f91-4609-bb8a-3e82e28bf642).

[Related Tasks](javascript:void(0))

|  |  |
| --- | --- |
| **Task** | **Topic** |
| Describes how to create, modify, delete or disable DDL triggers. | [Implement DDL Triggers](http://technet.microsoft.com/en-us/library/f44e5340-1d18-40e9-828e-0ffcca091ae3) |
| Describes how to create a CLR DDL trigger. | [Create CLR Triggers](http://technet.microsoft.com/en-us/library/31f41703-134d-49fc-9850-76c297351c2c) |
| Describes how to return information about DDL triggers. | [Get Information About DDL Triggers](http://technet.microsoft.com/en-us/library/462becea-292a-4b9e-bb98-533e89733911) |
| Describes how to return information about an event that fires a DDL trigger by using the EVENTDATA function. | [Use the EVENTDATA Function](http://technet.microsoft.com/en-us/library/675b8320-9c73-4526-bd2f-91ba42c1b604) |
| Describes how to manage trigger security. | [Manage Trigger Security](http://technet.microsoft.com/en-us/library/e94720a8-a3a2-4364-b0a3-bbe86e3ce4d5) |

**DML Triggers**

**SQL Server 2012**

[Other Versions](javascript:;)

Description: http://i.technet.microsoft.com/Areas/Epx/Content/Images/ImageSprite.png

* [SQL Server 2008 R2](http://technet.microsoft.com/en-us/library/ms178110(d=printer,v=sql.105).aspx)
* [SQL Server 2008](http://technet.microsoft.com/en-us/library/ms178110(d=printer,v=sql.100).aspx)
* [SQL Server 2005](http://technet.microsoft.com/en-us/library/ms178110(d=printer,v=sql.90).aspx)

DML triggers is a special type of stored procedure that automatically takes effect when a data manipulation language (DML) event takes place that affects the table or view defined in the trigger. DML events include INSERT, UPDATE, or DELETE statements. DML triggers can be used to enforce business rules and data integrity, query other tables, and include complex Transact-SQL statements. The trigger and the statement that fires it are treated as a single transaction, which can be rolled back from within the trigger. If a severe error is detected (for example, insufficient disk space), the entire transaction automatically rolls back.

[DML Trigger Benefits](javascript:void(0))

DML triggers are similar to constraints in that they can enforce entity integrity or domain integrity. In general, entity integrity should always be enforced at the lowest level by indexes that are part of PRIMARY KEY and UNIQUE constraints or are created independently of constraints. Domain integrity should be enforced through CHECK constraints, and referential integrity (RI) should be enforced through FOREIGN KEY constraints. DML triggers are most useful when the features supported by constraints cannot meet the functional needs of the application.

The following list compares DML triggers with constraints and identifies when DML triggers have benefits over .

* DML triggers can cascade changes through related tables in the database; however, these changes can be executed more efficiently using cascading referential integrity constraints. FOREIGN KEY constraints can validate a column value only with an exact match to a value in another column, unless the REFERENCES clause defines a cascading referential action.
* They can guard against malicious or incorrect INSERT, UPDATE, and DELETE operations and enforce other restrictions that are more complex than those defined with CHECK constraints.

Unlike CHECK constraints, DML triggers can reference columns in other tables. For example, a trigger can use a SELECT from another table to compare to the inserted or updated data and to perform additional actions, such as modify the data or display a user-defined error message.

* They can evaluate the state of a table before and after a data modification and take actions based on that difference.
* Multiple DML triggers of the same type (INSERT, UPDATE, or DELETE) on a table allow multiple, different actions to take place in response to the same modification statement.
* Constraints can communicate about errors only through standardized system error messages. If your application requires, or can benefit from, customized messages and more complex error handling, you must use a trigger.
* DML triggers can disallow or roll back changes that violate referential integrity, thereby canceling the attempted data modification. Such a trigger might go into effect when you change a foreign key and the new value does not match its primary key. However, FOREIGN KEY constraints are usually used for this purpose.
* If constraints exist on the trigger table, they are checked after the INSTEAD OF trigger execution but prior to the AFTER trigger execution. If the constraints are violated, the INSTEAD OF trigger actions are rolled back and the AFTER trigger is not executed.

[Types of DML Triggers](javascript:void(0))

AFTER trigger

AFTER triggers are executed after the action of the INSERT, UPDATE, MERGE, or DELETE statement is performed. AFTER triggers are never executed if a constraint violation occurs; therefore, these triggers cannot be used for any processing that might prevent constraint violations. For every INSERT, UPDATE, or DELETE action specified in a MERGE statement, the corresponding trigger is fired for each DML operation.

INSTEAD OF trigger

INSTEAD OF triggers override the standard actions of the triggering statement. Therefore, they can be used to perform error or value checking on one or more columns and the perform additional actions before insert, updating or deleting the row or rows. For example, when the value being updated in an hourly wage column in a payroll table exceeds a specified value, a trigger can be defined to either produce an error message and roll back the transaction, or insert a new record into an audit trail before inserting the record into the payroll table. The primary advantage of INSTEAD OF triggers is that they enable views that would not be updatable to support updates. For example, a view based on multiple base tables must use an INSTEAD OF trigger to support inserts, updates, and deletes that reference data in more than one table. Another advantage of INSTEAD OF triggers is that they enable you to code logic that can reject parts of a batch while letting other parts of a batch to succeed.

This table compares the functionality of the AFTER and INSTEAD OF triggers.

|  |  |  |
| --- | --- | --- |
| **Function** | **AFTER trigger** | **INSTEAD OF trigger** |
| Applicability | Tables | Tables and views |
| Quantity per table or view | Multiple per triggering action (UPDATE, DELETE, and INSERT) | One per triggering action (UPDATE, DELETE, and INSERT) |
| Cascading references | No restrictions apply | INSTEAD OF UPDATE and DELETE triggers are not allowed on tables that are targets of cascaded referential integrity constraints. |
| Execution | After:   * Constraint processing * Declarative referential actions * **inserted** and **deleted** tables creation * The triggering action | Before:   * Constraint processing   In place of:   * The triggering action   After:   * **inserted** and **deleted** tables creation |
| Order of execution | First and last execution may be specified | Not applicable |
| varchar(max) , nvarchar(max), and varbinary(max) column references in **inserted** and **deleted** tables | Allowed | Allowed |
| text , ntext, and image column references in **inserted** and **deleted** tables | Not allowed | Allowed |

CLR Triggers

A CLR Trigger can be either an AFTER or INSTEAD OF trigger. A CLR trigger can also be a DDL trigger. Instead of executing a Transact-SQL stored procedure, a CLR trigger executes one or more methods written in managed code that are members of an assembly created in the .NET Framework and uploaded in SQL Server.

[Related Tasks](javascript:void(0))

|  |  |
| --- | --- |
| **Task** | **Topic** |
| Describes how to create a DML trigger. | [Create DML Triggers](http://technet.microsoft.com/en-us/library/ms190227.aspx) |
| Describes how to create a CLR trigger. | [Create CLR Triggers](http://technet.microsoft.com/en-us/library/ms179562.aspx) |
| Describes how to create a DML trigger to handle both single-row and multi-row data modifications. | [Create DML Triggers to Handle Multiple Rows of Data](http://technet.microsoft.com/en-us/library/ms190752.aspx) |
| Describes how to nest triggers. | [Create Nested Triggers](http://technet.microsoft.com/en-us/library/ms190739.aspx) |
| Describes how to specify the order in which AFTER triggers are fired. | [Specify First and Last Triggers](http://technet.microsoft.com/en-us/library/ms189568.aspx) |
| Describes how to use the special inserted and delete tables in trigger code. | [Use the inserted and deleted Tables](http://technet.microsoft.com/en-us/library/ms191300.aspx) |
| Describes how to modify or rename a DML trigger. | [Modify or Rename DML Triggers](http://technet.microsoft.com/en-us/library/ms190671.aspx) |
| Describes how to view information about DML triggers. | [Get Information About DML Triggers](http://technet.microsoft.com/en-us/library/ms179309.aspx) |
| Describes how to delete or disable DML triggers. | [Delete or Disable DML Triggers](http://technet.microsoft.com/en-us/library/ms175506.aspx) |
| Describes how to manage trigger security. | [Manage Trigger Security](http://technet.microsoft.com/en-us/library/ms191134.aspx) |

**Logon Triggers**

**SQL Server 2012**

[Other Versions](javascript:;)

Description: http://i.technet.microsoft.com/Areas/Epx/Content/Images/ImageSprite.png

* [SQL Server 2008 R2](http://technet.microsoft.com/en-us/library/2f0ebb2f-de10-482d-9806-1a5de5b312b8(d=printer,v=sql.105))
* [SQL Server 2008](http://technet.microsoft.com/en-us/library/2f0ebb2f-de10-482d-9806-1a5de5b312b8(d=printer,v=sql.100))
* [SQL Server 2005](http://technet.microsoft.com/en-us/library/2f0ebb2f-de10-482d-9806-1a5de5b312b8(d=printer,v=sql.90))

Logon triggers fire stored procedures in response to a LOGON event. This event is raised when a user session is established with an instance of SQL Server. Logon triggers fire after the authentication phase of logging in finishes, but before the user session is actually established. Therefore, all messages originating inside the trigger that would typically reach the user, such as error messages and messages from the PRINT statement, are diverted to the SQL Server error log. Logon triggers do not fire if authentication fails.

You can use logon triggers to audit and control server sessions, such as by tracking login activity, restricting logins to SQL Server, or limiting the number of sessions for a specific login. For example, in the following code, the logon trigger denies log in attempts to SQL Server initiated by login login\_test if there are already three user sessions created by that login.

Transact-SQL

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;

Note that the LOGON event corresponds to the AUDIT\_LOGIN SQL Trace event, which can be used in [Event Notifications](http://technet.microsoft.com/en-us/library/ms186376). The primary difference between triggers and event notifications is that triggers are raised synchronously with events, whereas event notifications are asynchronous. This means, for example, that if you want to stop a session from being established, you must use a logon trigger. An event notification on an AUDIT\_LOGIN event cannot be used for this purpose.

[Specifying First and Last Trigger](javascript:void(0))

Multiple triggers can be defined on the LOGON event. Any one of these triggers can be designated the first or last trigger to be fired on an event by using the [sp\_settriggerorder](http://technet.microsoft.com/en-us/library/ms186762) system stored procedure. SQL Server does not guarantee the execution order of the remaining triggers.

[Managing Transactions](javascript:void(0))

Before SQL Server fires a logon trigger, SQL Server creates an implicit transaction that is independent from any user transaction. Therefore, when the first logon trigger starts firing, the transaction count is 1. After all the logon triggers finish executing, the transaction commits. As with other types of triggers, SQL Server returns an error if a logon trigger finishes execution with a transaction count of 0. The ROLLBACK TRANSACTION statement resets the transaction count to 0, even if the statement is issued inside a nested transaction. COMMIT TRANSACTION might decrement the transaction count to 0. Therefore, we advise against issuing COMMIT TRANSACTION statements inside logon triggers.

Consider the following when you are using a ROLLBACK TRANSACTION statement inside logon triggers:

* Any data modifications made up to the point of ROLLBACK TRANSACTION are rolled back. These modifications include those made by the current trigger and those made by previous triggers that executed on the same event. Any remaining triggers for the specific event are not executed.
* The current trigger continues to execute any remaining statements that appear after the ROLLBACK statement. If any of these statements modify data, the modifications are not rolled back.

A user session is not established if any one of the following conditions occur during execution of a trigger on a LOGON event:

* The original implicit transaction is rolled back or fails.
* An error that has severity greater than 20 is raised inside the trigger body.

[Disabling a Logon Trigger](javascript:void(0))

A logon trigger can effectively prevent successful connections to the Database Engine for all users, including members of the sysadmin fixed server role. When a logon trigger is preventing connections, members of the sysadmin fixed server role can connect by using the dedicated administrator connection, or by starting the Database Engine in minimal configuration mode (-f). For more information, see [Database Engine Service Startup Options](http://technet.microsoft.com/en-us/library/ms190737).

[Related Tasks](javascript:void(0))

|  |  |
| --- | --- |
| **Task** | **Topic** |
| Describes how to create logon triggers. Logon triggers can be created from any database, but are registered at the server level and reside in the **master** database. | [CREATE TRIGGER (Transact-SQL)](http://technet.microsoft.com/en-us/library/ms189799) |
| Describes how to modify logon triggers. | [ALTER TRIGGER (Transact-SQL)](http://technet.microsoft.com/en-us/library/ms176072) |
| Describes how to delete logon triggers. | [DROP TRIGGER (Transact-SQL)](http://technet.microsoft.com/en-us/library/ms173497) |
| Describes how to return information about logon triggers. | [sys.server\_triggers (Transact-SQL)](http://technet.microsoft.com/en-us/library/ms176054)  [sys.server\_trigger\_events (Transact-SQL)](http://technet.microsoft.com/en-us/library/ms188375) |
| Describes how to capture logon trigger event data. |  |