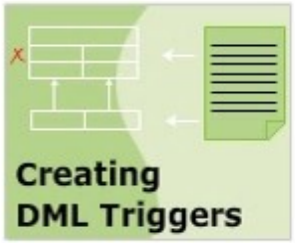
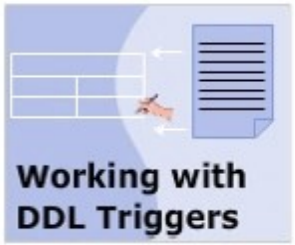
Module Summary

In this module, **Introduction to Triggers**, you learnt about:

- Introduction to Triggers
- Creating DML Triggers
- Working with DML Triggers
- Working with DDL Triggers

Click on each link for a summary of the lesson.





Working with DDL Triggers

DDL triggers execute when a table or a view is created, modified or deleted using the `CREATE`, `ALTER` or `DROP` statements. DDL triggers are defined either at the database level or at the server level. The scope of the DDL trigger depends on whether the trigger executes for database events or server events. Accordingly, the DDL triggers are classified into Database-scoped DDL triggers and Server-scoped DDL triggers. SQL Server 2005 allows you to create DDL triggers for create, alter and drop events.

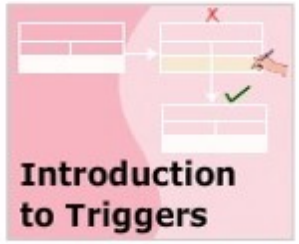
Module Overview

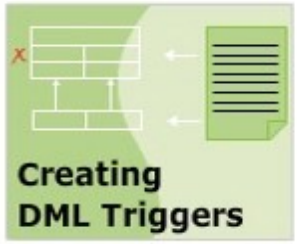
Welcome to the module **Introduction to Triggers**. A trigger is a stored procedure that is executed when data in a specified table is modified. Triggers are often created to enforce referential integrity among logically related data in different tables. Because users cannot circumvent triggers, you can use triggers to enforce complex business rules that maintain data integrity.

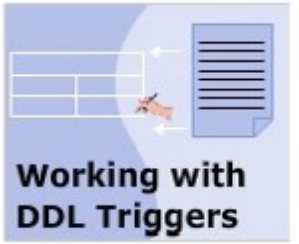
In this module, you will learn about:

- Introduction to Triggers
- Creating DML Triggers
- Working with DML Triggers
- Working with DDL Triggers

Towards the end of the module, there are demonstrations and/ or simulations for reinforcing the theoretical concepts.







Press the **RIGHT ARROW** on your keyboard to go to the next screen or **LEFT ARROW** to go to the previous screen.

Lesson Overview

In this first lesson, **Introduction to Triggers**, you will learn to:

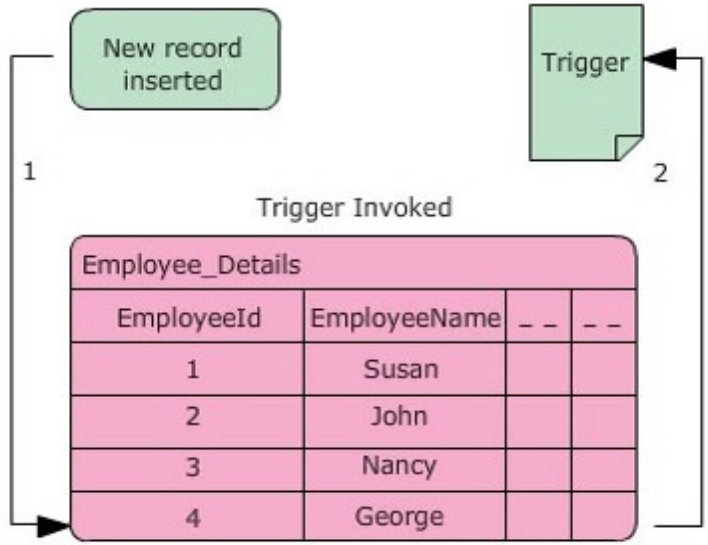
- Define triggers.
- State the uses of triggers.
- List the different types of triggers.
- Distinguish between DDL and DML triggers.

Triggers

A trigger is a stored procedure that is executed when an attempt is made to modify data in a table that is protected by the trigger. Unlike standard system stored procedures, triggers cannot be executed directly, nor do they pass or receive parameters. Triggers are defined on specific tables and these tables are referred to as trigger tables.

If a trigger is defined on the `INSERT`, `UPDATE` or `DELETE` action on a table, it fires automatically when these actions are attempted. This automatic execution of the trigger cannot be circumvented.

In SQL Server 2005, triggers are created using the `CREATE TRIGGER` statement.



Uses of Triggers

Triggers can contain complex processing logic and are generally used for maintaining low-level data integrity. The primary uses of triggers can be classified as follows:

- Cascading changes through related tables.
- Enforcing more complex data integrity than CHECK constraints.
- Defining custom error messages.
- Maintaining denormalized data.
- Comparing before and after states of data being modified.

Click on each link to learn more.

Uses of Triggers

Triggers can contain complex processing logic and are generally used for maintaining low-level data integrity. The primary uses of triggers can be classified as follows:

- Cascading changes through related tables.
- Enforcing more complex data integrity than CHECK constraints.
- Defining custom error messages.
- Maintaining denormalized data.
- Comparing before and after states of data being modified.

Click on each link to learn more.

Cascading changes through related tables

You can use a trigger to cascade changes through related tables. For example, consider a table `Salary_Details` having a `FOREIGN KEY`, `EmpID` referencing the `PRIMARY KEY`, `EmpID` of the `Employee_Details` table. If an update or a delete event occurs in the `Employee_Details` table, an update or delete trigger can be defined to cascade these changes to the `Salary_Details` table.

Uses of Triggers

Triggers can contain complex processing logic and are generally used for maintaining low-level data integrity. The primary uses of triggers can be classified as follows:

- Cascading changes through related tables.
- Enforcing more complex data integrity than CHECK constraints.
- Defining custom error messages.
- Maintaining denormalized data.

Enforcing more complex data integrity than CHECK constraints

Unlike `CHECK` constraints, triggers can reference columns in other tables. This feature can be used to apply more complex data integrity checks. Data integrity can be enforced by:

- Checking constraints before cascading updates or deletes.
- Creating multi-row triggers for actions executed on multiple rows.
- Enforcing referential integrity between databases.

Uses of Triggers

Triggers can contain complex processing logic and are generally used for maintaining low-level data integrity. The primary uses of triggers can be classified as follows:

- Cascading changes through related tables.
- Enforcing more complex data integrity than CHECK constraints.
- Defining custom error messages.
- Maintaining denormalized data.
- Comparing before and after states of data being modified.

Click on each link to learn more.

Defining custom error messages

Custom error messages are used for providing more suitable or detailed explanations in certain error situations. Triggers can be used to invoke such predefined custom error messages when the relevant error conditions occur.

Press the **RIGHT ARROW** on your keyboard to go to the next screen or **LEFT ARROW** to go to the previous screen.

Uses of Triggers

Triggers can contain complex processing logic and are generally used for maintaining low-level data integrity. The primary uses of triggers can be classified as follows:

- Cascading changes through related tables.
- Enforcing more complex data integrity than CHECK constraints.
- Defining custom error messages.
- Maintaining denormalized data.
- Comparing before and after states of data being modified.

Click on each link to learn more.

Maintaining denormalized data

Low-level data integrity can be maintained in denormalized database environments using triggers. Denormalized data generally refers to redundant or derived data. Here, triggers are used for checks that do not require exact matches. For example, if the value of the year is to be checked against complete dates, a trigger can be used to perform the check.

Uses of Triggers

Triggers can contain complex processing logic and are generally used for maintaining low-level data integrity. The primary uses of triggers can be classified as follows:

- Cascading changes through related tables.
- Enforcing more complex data integrity than CHECK constraints.
- Defining custom error messages.
- Maintaining denormalized data.
- Comparing before and after states of data being modified.

Click on each link to learn more.

Comparing before and after states of data being modified

Triggers provide the option to reference changes that are made to data by `INSERT`, `UPDATE` and `DELETE` statements. This allows you to reference the affected rows when modifications are carried out through triggers.

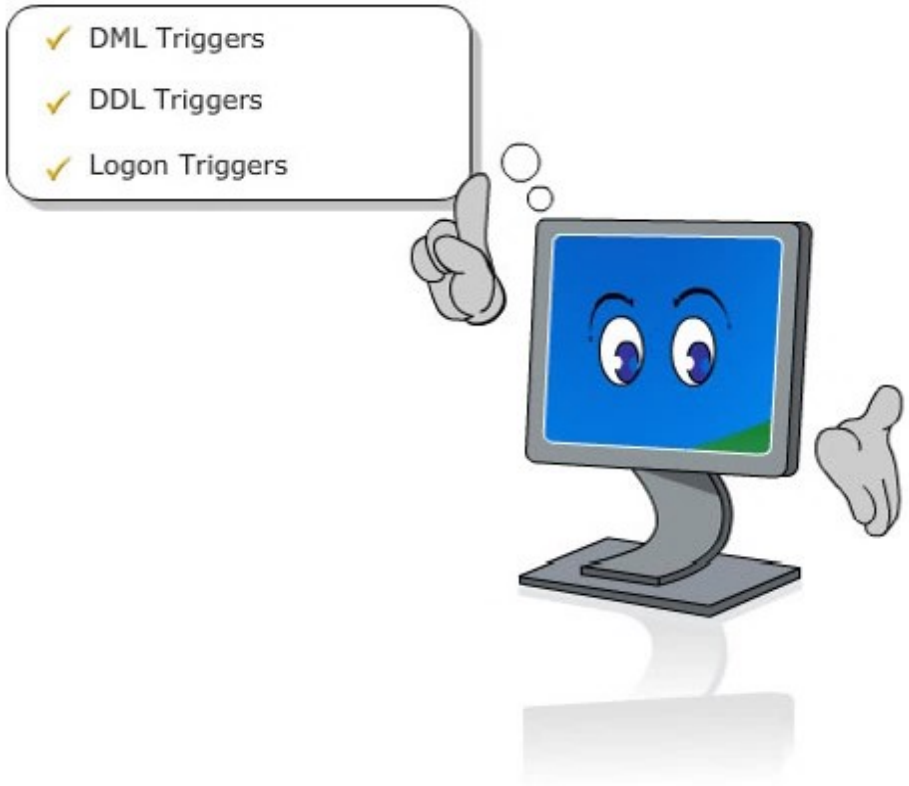
Types of Triggers

A trigger can be set to automatically execute an action when a language event occurs in a table or a view. Language events can be classified as Data Manipulation Language (DML) events and Data Definition Language (DDL) events. Triggers associated with DML events are known as DML triggers whereas triggers associated with DDL events are known as DDL triggers.

Triggers in SQL Server 2005 can be classified into three basic types:

- Data Manipulation Language (DML) Triggers
- Data Definition Language (DDL) Triggers
- Logon Triggers

Click on each link to learn more.



Types of Triggers

A trigger can be set to automatically execute an action when a language event occurs in a table or a view. Language events can be classified as Data Manipulation Language (DML) events and Data Definition Language (DDL) events. Triggers associated with DML events are known as DML triggers whereas triggers associated with DDL events are known as DDL triggers.

- ✓ DML Triggers
- ✓ DDL Triggers
- ✓ Logon Triggers

Triggers in SQL Server 2005 can be classified into three basic types:

- Data Manipulation Language (DML) Triggers
- Data Definition Language (DDL) Triggers
- Logon Triggers

Click on each link to learn more.

Data Manipulation Language (DML) Triggers

DML triggers execute when data is inserted, modified or deleted in a table or a view using the `INSERT`, `UPDATE` or `DELETE` statements.

Press the **RIGHT ARROW** on your keyboard to go to the next screen or **LEFT ARROW** to go to the previous screen.

Types of Triggers

A trigger can be set to automatically execute an action when a language event occurs in a table or a view. Language events can be classified as Data Manipulation Language (DML) events and Data Definition Language (DDL) events. Triggers associated with DML events are known as DML triggers whereas triggers associated with DDL events are known as DDL triggers.

- ✓ DML Triggers
- ✓ DDL Triggers
- ✓ Logon Triggers

Triggers in SQL Server 2005 can be classified into three basic types:

- Data Manipulation Language (DML) Triggers
- Data Definition Language (DDL) Triggers
- Logon Triggers

Click on each link to learn more.

Data Definition Language (DDL) Triggers

DDL triggers execute when a table or a view is created, modified or deleted using the `CREATE`, `ALTER` or `DROP` statements.

Types of Triggers

A trigger can be set to automatically execute an action when a language event occurs in a table or a view. Language events can be classified as Data Manipulation Language (DML) events and Data Definition Language (DDL) events. Triggers associated with DML events are known as DML triggers whereas triggers associated with DDL events are known as DDL triggers.

- ✓ DML Triggers
- ✓ DDL Triggers
- ✓ Logon Triggers

Triggers in SQL Server 2005 can be classified into three basic types:

- Data Manipulation Language (DML) Triggers
- Data Definition Language (DDL) Triggers
- Logon Triggers

Click on each link to learn more.

Logon Triggers

Logon triggers execute stored procedures when a session is established with a `LOGON` event. These triggers are invoked after the login authentication is complete and before the actual session is established. Logon triggers control server sessions by restricting invalid logins or limiting the number of sessions.

DDL Triggers versus DML Triggers

DDL and DML triggers have different uses and are executed with different database events. The differences between DDL and DML triggers are described in the table below.

DDL Triggers	DML Triggers
DDL triggers execute stored procedures on `CREATE`, `ALTER` and `DROP` statements.	DML triggers execute on `INSERT`, `UPDATE` and `DELETE` statements.
DDL triggers are used to check and control database operations.	DML triggers are used to enforce business rules when data is modified in tables or views.
DDL triggers operate only after the table or a view is modified.	DML triggers execute either while modifying the data or after the data is modified.
DDL triggers are defined either at the database or the sever level.	DML triggers are defined at the database level.

41 | 16

Score

Knowledge Check

Can you match the types of triggers in SQL Server 2005 against their corresponding descriptions?

Select an option from each drop-down list box and then click on **Submit**.

	Description	Trigger
(A)	Executes when a table or a view is created, modified or deleted.	Select Step...
(B)	Invokes a stored procedure when a session is established on the server.	Select Step...
(C)	Executes when data is inserted, modified or deleted in a table or a view.	Select Step...
(D)	Controls server sessions by limiting the number of sessions.	Select Step...
(E)	Maintains referential integrity.	Select Step...

Submit

View Answer

Knowledge Check

Can you match the types of triggers in SQL Server 2005 against their corresponding descriptions?

Select an option from each drop-down list box and then click on **Submit**.

	Description	Trigger
(A)	Executes when a table or a view is created, modified or deleted.	DDL
(B)	Invokes a stored procedure when a session is established on the server.	Logon
(C)	Executes when data is inserted, modified or deleted in a table or a view.	DML
(D)	Controls server sessions by limiting the number of sessions.	Logon
(E)	Maintains referential integrity.	DML

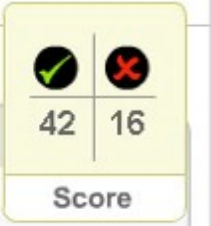
Correct

The correct answers are displayed.

Submit

View Answer

Knowledge Check



Which of these statements about the triggers of SQL Server 2005 are true and which statements are false?

Select an option for each statement and then click on **Submit**.

	Statements	True	False
(A)	Triggers retrieve information from tables of the same as well as other databases.	○	○
(B)	DDL triggers operate only after a table or a view is modified.	○	○
(C)	DML triggers execute on `INSERT`, `UPDATE` and `DELETE` statements.	○	○
(D)	DDL triggers execute either while modifying the data or after the data is modified.	○	○
(E)	Triggers enforce complex restrictions that cannot be defined by the `CHECK` constraints.	○	○

Submit

43 | 16

Score

Knowledge Check

Which of these statements about the triggers of SQL Server 2005 are true and which statements are false?

Select an option for each statement and then click on **Submit**.

	Statements	True	False
(A)	Triggers retrieve information from tables of the same as well as other databases.	◉	○
(B)	DDL triggers operate only after a table or a view is modified.	◉	○
(C)	DML triggers execute on `INSERT`, `UPDATE` and `DELETE` statements.	◉	○
(D)	DDL triggers execute either while modifying the data or after the data is modified.	○	◉
(E)	Triggers enforce complex restrictions that cannot be defined by the `CHECK` constraints.	◉	○

Correct

The correct answers are displayed.

Submit

Press the **RIGHT ARROW** on your keyboard to go to the next screen or **LEFT ARROW** to go to the previous screen.

Lesson Review

In this first lesson, **Introduction to Triggers**, you learnt to:

- Define triggers.
- State the uses of triggers.
- List the different types of triggers.
- Distinguish between DDL and DML triggers.

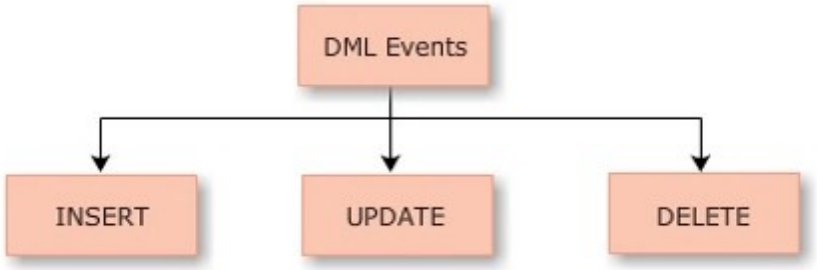
Lesson Overview

In this second lesson, **Creating DML triggers**, you will learn to:

- Describe `INSERT` Triggers.
- Describe `UPDATE` Triggers.
- Describe `DELETE` Triggers.
- Describe `AFTER` Triggers.
- Describe `INSTEAD OF` Triggers.

DML Triggers

DML triggers are executed when data manipulation language (DML) events occur in tables or views. These DML events include the `INSERT`, `UPDATE` and `DELETE` statements. DML triggers can execute either on completion of the DML events or in place of the DML events.



DML triggers enforce referential integrity by cascading changes to related tables when a row is modified. DML triggers can perform multiple actions for each modification statement.

DML triggers are of three main types:

- `INSERT` trigger
- `UPDATE` trigger
- `DELETE` trigger

Introduction to Inserted and Deleted Tables

The SQL statements in DML triggers use two special types of tables to modify data in the database. When the data is inserted, updated or deleted, SQL Server 2005 creates and manages these tables automatically. The tables temporarily store the original as well as the modified data. These tables are:

- Inserted Table
- Deleted Table

Click on each link to learn more.

More

Introduction to Inserted and Deleted Tables

The SQL statements in DML triggers use two special types of tables to modify data in the database. When the data is inserted, updated or deleted, SQL Server 2005 creates and manages these tables automatically. The tables temporarily store the original as well as the modified data. These tables are:

Inserted Table

The Inserted table contains copies of records that are modified with the `INSERT` and `UPDATE` operations on the trigger table. Trigger table is the table on which the trigger is defined. The `INSERT` and `UPDATE` operations insert new records into the Inserted and Trigger tables.

More

Introduction to Inserted and Deleted Tables

The SQL statements in DML triggers use two special types of tables to modify data in the database. When the data is inserted, updated or deleted, SQL Server 2005 creates and manages these tables automatically. The tables temporarily store the original as well as the modified data. These tables are:

Deleted Table

The Deleted table contains copies of records that are modified with the `DELETE` and `UPDATE` operations on the trigger table. Trigger table is the table on which the trigger is defined. These operations delete the records from the trigger table and insert them in the Deleted table.

Inserted and Updated Record Copies

Inserted Table

Updated and Deleted Record Copies

Deleted Table

More

Introduction to Inserted and Deleted Tables

The SQL statements in DML triggers use two special types of tables to modify data in the database. When the data is inserted, updated or deleted, SQL Server 2005 creates and manages these tables automatically. The tables temporarily store the original as well as the modified data. These tables are:

- Inserted Table
- Deleted Table

Click on each link to learn more.

More

More

The Inserted and Deleted tables do not physically remain present in the database. They are created and dropped as and when triggering events occur.

"INSERT" Triggers

An `INSERT` trigger is executed when a new record is inserted in a table. The `INSERT` trigger ensures that the value being entered conforms to the constraints defined on that table.

When you insert a record in a table, the `INSERT` trigger saves a copy of that record in the Inserted table. It then checks whether the new value in the Inserted table conforms to the specified constraints. If the record is valid, the `INSERT` trigger inserts the row in the trigger table, otherwise it displays an error message.

An `INSERT` trigger is created using the `INSERT` keyword in the `CREATE TRIGGER` and `ALTER TRIGGER` statements.

```
1  INSERT INTO Account_Transactions
2  (TransID,EmpID,CustID,TransTypeID,TransDate,
3  TransNumber,Deposit,Withdrawal)
4  VALUES(4,5,'C0013',2,'2007/4/18',5,0,60000)
```

Messages

```
Withdrawal amount cannot exceed 50000
Msg 3609, Level 16, State 1, Line 1
The transaction ended in the trigger. The batch has been aborted.
```

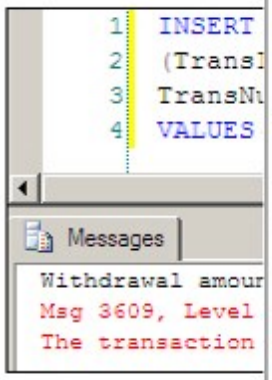
Syntax Snippet

"INSERT" Triggers

An INSERT trigger is executed when a new record i ng entered conforms to the constraints defined on that table.

When you insert a record in a table, the INSERT tri cks whether the new value in the Inserted table conforms to the sp the row in the trigger table, otherwise it displays an error messag

An INSERT trigger is created using the INSERT key



Syntax

Snippet

The following is the syntax for creating an INSERT trigger.

```
CREATE TRIGGER [schema_name.] trigger_name
ON [schema_name.] table_name
[WITH ENCRYPTION]
{FOR INSERT}
AS
[IF UPDATE (column_name)...]
    [{AND | OR} UPDATE (column_name)...]
      <sql_statements>
```

where,

schema_name: Specifies the name of the schema to which the table/trigger belongs.

trigger_name: Specifies the name of the trigger.

table_name: Specifies the table on which the DML trigger is created.

WITH ENCRYPTION: Encrypts the text of the CREATE TRIGGER statement.

FOR: Specifies that the DML trigger executes after the modification operations are complete.

Continued...

Syntax

Press the **RIGHT ARROW** on your keyboard to go to the next screen or **LEFT ARROW** to go to the previous screen.

"INSERT" Triggers

An INSERT trigger is executed when a new record i ng entered conforms to the constraints defined on that table.

When you insert a record in a table, the INSERT tri cks whether the new value in the Inserted table conforms to the sp the row in the trigger table, otherwise it displays an error messag

An INSERT trigger is created using the INSERT key

```
1  INSERT
2  (Transl
3  TransNu
4  VALUES
```

Messages

```
Withdrawal amour
Msg 3609, Level
The transaction
```

Syntax

Snippet

`INSERT`: Specifies that this DML trigger will be invoked by insert operations.

`UPDATE`: Returns a Boolean value that indicates whether an INSERT or UPDATE attempt was made on a specified column.

`column_name`: Is the name of the column to test for the UPDATE action.

`AND`: Combines two Boolean expressions and returns `TRUE` when both expressions are `TRUE`.

`OR`: Combines two Boolean expressions and returns `TRUE` if at least one expression is `TRUE`.

`sql_statement`: Specifies the SQL statements that are executed in the DML trigger.

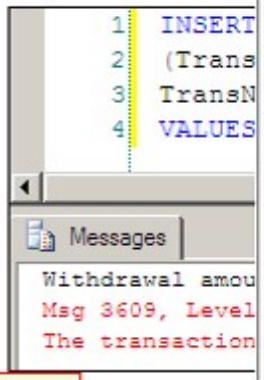
Syntax

"INSERT" Triggers

An INSERT trigger is executed when a new record
conforms to the constraints defined on that table.

When you insert a record in a table, the INSERT tr
new value in the Inserted table conforms to the sp
trigger table, otherwise it displays an error messag

An INSERT trigger is created using the INSERT key



Syntax

Snippet

The following code creates an INSERT trigger on the Account_Transactions table. When a new record is inserted, and if the withdrawal amount exceeds 50000, the insert trigger displays an error message and rolls back the transaction using the ROLLBACK TRANSACTION statement.

```
CREATE TRIGGER CheckWithdrawal_Amount
ON Account_Transactions
FOR INSERT
AS
    IF (SELECT Withdrawal From inserted) > 50000
       BEGIN
    PRINT 'Withdrawal amount cannot exceed 50000'
    ROLLBACK TRANSACTION
       END
```

Continued...

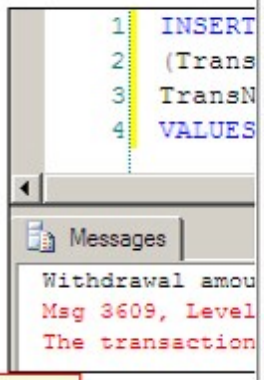
Snippet

"INSERT" Triggers

An INSERT trigger is executed when a new record
conforms to the constraints defined on that table.

When you insert a record in a table, the INSERT tr
new value in the Inserted table conforms to the sp
trigger table, otherwise it displays an error messag

An INSERT trigger is created using the INSERT key



Syntax

Snippet

The following code inserts a record where the withdrawal amount exceeds 50000.

This causes the INSERT trigger to display an error message and roll back the transaction.

```
INSERT INTO Account_Transactions
(TransID,EmpID,CustID,TransTypeID,TransDate,TransNumber,Deposi
t,Withdrawal)
VALUES(4,5,'C0013',2,'2007/4/18',5,0,60000)
```

The following error message is displayed as specified by the PRINT statement:

```
Withdrawal amount cannot exceed 50000.
```

Snippet

"UPDATE" Triggers

The `UPDATE` trigger copies the original record in the Deleted table and the new record into the Inserted table when a record is updated. It then evaluates the new record to determine if the values conform to the constraints specified in the trigger table.

If the new values are valid, the record from the Inserted table is copied to the trigger table. However, if the new values are invalid, an error message is displayed. Also, the original record is copied from the Deleted table back into the trigger table.

An `UPDATE` trigger is created using the `UPDATE` keyword in the `CREATE TRIGGER` and `ALTER TRIGGER` statements.

```
1  UPDATE Employee_Details
2  SET DateofBirth='2008/06/02'
3  WHERE EmpID=16
```

Messages

```
Date of birth cannot be greater than today's date
Msg 3609, Level 16, State 1, Line 1
The transaction ended in the trigger.
The batch has been aborted.
```

Syntax

Snippet

"UPDATE" Triggers

The UPDATE trigger copies the original record in the Deleted table and the new record into the Inserted table when a record is updated. It then evaluates the new record to determine if the values conform to the constraints specified in the trigger table.

If the new values are valid, the record from the Inserted table is copied to the trigger table. However, if
error message is displayed. Also, the or
the Deleted table back into the trigger t

An UPDATE trigger is created using the U
TRIGGER and ALTER TRIGGER statements

```
1  UPDATE Employee_Details
2  SET DateofBirth='2008/06/02'
3  WHERE EmpID=16
```

The following is the syntax for creating an UPDATE trigger at the table-level.

```
CREATE TRIGGER [schema_name.] trigger_name
ON [schema_name.] table_name
[WITH ENCRYPTION]
{FOR UPDATE}
AS
[IF UPDATE (column_name)...]
    [{AND | OR} UPDATE (column_name)...]
       <sql_statements>
```

where,

FOR UPDATE: Specifies that this DML trigger will be invoked after the update operations.

Syntax

Snippet

Syntax

"UPDATE" Triggers

The UPDATE trigger copies the original record in the Deleted table and the new record into the Inserted table when a record is updated. It then evaluates the new record to determine if the values conform to the constraints specified in the trigger table.

If the new values are valid, the record from the Inserted table is copied to the trigger table. However, if the new values are invalid, an error message is displayed. Also, the original record is copied from the Deleted table back into the trigger table.

An UPDATE trigger is created using the UPDATE keyword in the CREATE TRIGGER and ALTER TRIGGER statements.

Syntax

Snippet

The following code creates an UPDATE trigger at the table level on the Employee_Details table. When a record is modified, the UPDATE trigger gets activated. It checks whether the date of birth is greater than today's date. It displays an error message for invalid values and rolls back the modification operation using the ROLLBACK TRANSACTION statement.

```
CREATE TRIGGER CheckDateofBirth
ON Employee_Details
FOR UPDATE
AS
    IF (SELECT DateofBirth From inserted) > getDate
()
    BEGIN
    PRINT 'Date of birth cannot be greater than
today''s date'
       ROLLBACK TRANSACTION
END
```

Continued...

Snippet

"UPDATE" Triggers

The `UPDATE` trigger copies the original record in the Deleted table and the new record into the Inserted table when a record is updated. It then evaluates the new record to determine if the values conform to the constraints specified in the trigger table.

If the new values are valid, the record from the Inserted table is copied to the trigger table. However, if the new values are invalid, an error message is displayed. Also, the original record is copied from the Deleted table back into the trigger table.

An `UPDATE` trigger is created using the `UPDATE` keyword in the `CREATE TRIGGER` and `ALTER TRIGGER` statements.

Syntax

Snippet

The following code updates a record where an invalid date of birth is specified. This causes the update trigger to display the error message and rollback the transaction.

```
UPDATE Employee_Details SET
DateofBirth='2008/06/02' WHERE EmpID=16
```

The following error message is displayed as specified by the `PRINT` statement:

```
Date of birth cannot be greater than today's date.
```

Snippet

Creating "UPDATE" Triggers

The `UPDATE` triggers are created either at the column level or at the table level. The triggers at the column level execute when updates are made in the specified column. The triggers at the table level execute when updates are made anywhere in the entire table.

For creating an `UPDATE` trigger at the column level, the `UPDATE()` function is used to specify the column.

```
1  UPDATE Customer_Details
2  SET CustID='C0021'
3  WHERE CustID='C0004'
```

Messages

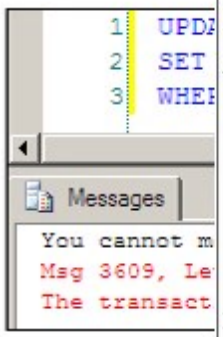
```
You cannot modify the ID of a customer
Msg 3609, Level 16, State 1, Line 1
The transaction ended in the trigger. The batch has been aborted.
```

Snippet

Creating "UPDATE" Triggers

The UPDATE triggers are created either at the co
are made in the specified column. The triggers at

For creating an UPDATE trigger at the column lev



Snippet

The following code creates an UPDATE trigger at the column level on the CustID column of Customer_Details table. When the customer ID is modified, the UPDATE trigger gets activated and an error message is displayed. The modification operation is rolled back using the ROLLBACK TRANSACTION statement.

```
CREATE TRIGGER Check_CustID
ON Customer_Details
FOR UPDATE
AS
   IF UPDATE(CustID)
   BEGIN
   PRINT 'You cannot modify the ID of a customer'
   ROLLBACK TRANSACTION
END
```

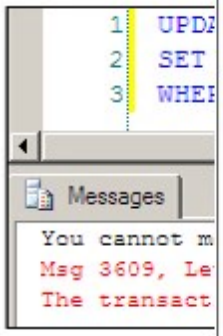
Continued...

Snippet

Creating "UPDATE" Triggers

The UPDATE triggers are created either at the co
are made in the specified column. The triggers at

For creating an UPDATE trigger at the column lev



Snippet

The following code updates a record where the value in the CustID column is being modified. This causes the update trigger to fire. The update trigger displays an error message and rolls back the transaction.

```
UPDATE Customer_Details SET CustID='C0021' WHERE CustID='C0004'
```

The following error message is displayed as specified by the PRINT statement:

```
You cannot modify the ID of a customer.
```

Snippet

"DELETE" Triggers

The DELETE trigger can be created to restrict a user from deleting a particular record in a table. The following will happen if the user tries to delete the record:

- The record is deleted from the trigger table and inserted in the Deleted table.
- It is checked for constraints against deletion.
- If there is a constraint on the record to prevent deletion, the DELETE trigger displays an error message.
- The deleted record stored in the Deleted table is copied back to the trigger table.

A DELETE trigger is created using the DELETE keyword in the CREATE TRIGGER statement.

```
1  DELETE FROM Customer_Details
2  WHERE Country='USA'
```

Messages

```
You cannot delete the customers of USA
Msg 3609, Level 16, State 1, Line 1
The transaction ended in the trigger. The batch has been aborted.
```

Syntax Snippet

"DELETE" Triggers

The DELETE trigger can be created to restrict a user from deleting a particular record in a table. The following will happen if the user tries to delete the record:

- The record is deleted from the trigger table and inserted in the Deleted table.
- It is checked for constraints against deletion.
- If there is a constraint on the record to prevent deletion, the DELETE trigger displays an error message.
- The deleted record stored in the Deleted table is copied back to the trigger table.

A DELETE trigger is created using the DELETE keyword in the CREATE TRIGGER statement.

Syntax Snippet

```
1  DELETE FROM Customer_Details
2  WHERE Country='USA'
```

Messages

The following is the syntax for creating a DELETE trigger.

```
CREATE TRIGGER <trigger_name>
ON <table_name>
[WITH ENCRYPTION]
FOR DELETE
AS <sql_statement>
```

where,

DELETE: Specifies that this DML trigger will be invoked by delete operations.

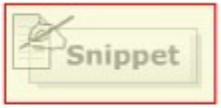
Syntax

"DELETE" Triggers

The `DELETE` trigger can be created to restrict a user from deleting a particular record in a table. The following will happen if the user tries to delete the record:

- The record is deleted from the trigger table and inserted in the Deleted table.
- It is checked for constraints against deletion.
- If there is a constraint on the record to prevent deletion, the DELETE trigger displays an error message.
- The deleted record stored in the Deleted table is copied back to the trigger table.

A `DELETE` trigger is created using the `DELETE` keyword in the `CREATE TRIGGER` statement.



The following code creates a `DELETE` trigger on the `Customer_Details` table. If a record of a customer from USA is deleted, the `DELETE` trigger gets activated and an error message is displayed. The delete operation is rolled back using the `ROLLBACK TRANSACTION` statement.

```
CREATE TRIGGER Check_Customers
ON Customer_Details
FOR DELETE
AS
   IF 'USA' IN (SELECT Country FROM deleted)
     BEGIN
   PRINT 'You cannot delete the customers of USA'
   ROLLBACK TRANSACTION
END
```

Continued...

Snippet

"DELETE" Triggers

The DELETE trigger can be created to restrict a user from deleting a particular record in a table. The following will happen if the user tries to delete the record:

- The record is deleted from the trigger table and inserted in the Deleted table.
- It is checked for constraints against deletion.
- If there is a constraint on the record to prevent deletion, the DELETE trigger displays an error message.
- The deleted record stored in the Deleted table is copied back to the trigger table.

A DELETE trigger is created using the DELETE keyword in the CREATE TRIGGER statement.

Syntax

Snippet

The following code attempts to delete records from the Customer_Details table for customers in USA.

```
DELETE FROM Customer_Details WHERE Country='USA'
```

The following error message is displayed as specified by the PRINT statement.

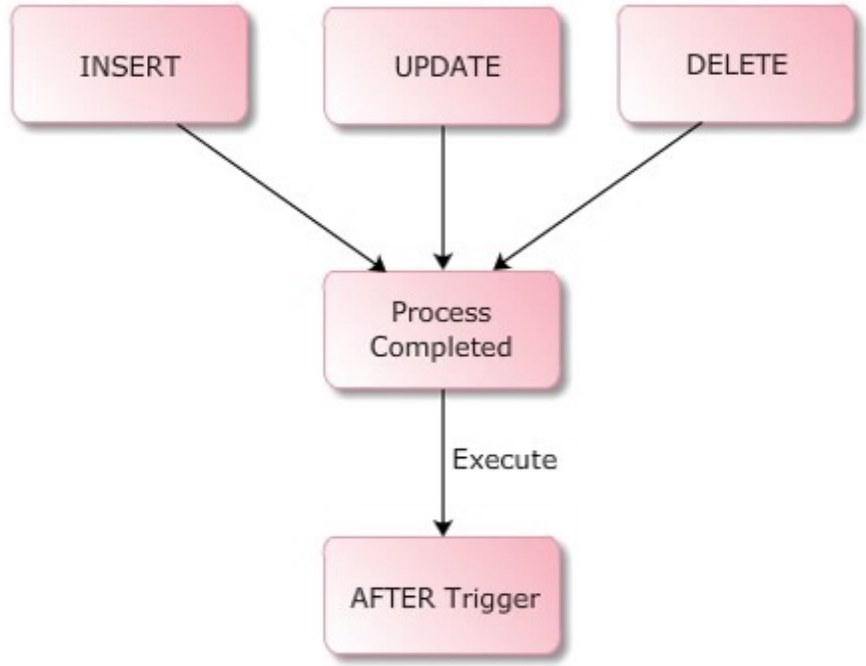
```
You cannot delete the customers of USA.
```

Snippet

"AFTER" Triggers

An `AFTER` trigger is executed on completion of `INSERT`, `UPDATE` or `DELETE` operations. `AFTER` triggers can be created only on tables. A table can have multiple `AFTER` triggers defined for each `INSERT`, `UPDATE` and `DELETE` operation. If multiple `AFTER` triggers are created on the same table, you must define the order in which the triggers must be executed.

An `AFTER` trigger is executed when the constraint check in the table is completed. Also, the trigger is executed after the Inserted and Deleted tables are created.



Syntax

Snippet

"AFTER" Triggers

An AFTER trigger is executed on completion of INSERT, UPDATE or DELETE operations. AFTER triggers can be created only on tables. A table can have multiple AFTER triggers defined for each INSERT, UPDATE and DELETE operation. If multiple AFTER triggers are created on the same table, you must define the order in which the triggers must be executed.

An AFTER trigger is executed when the constraint check in the table is completed. Also, the trigger is executed after the Inserted and Deleted tables are created.

Syntax

Snippet

The following is the syntax for creating an AFTER trigger.

```
CREATE TRIGGER <trigger_name>
ON <table_name>
[WITH ENCRYPTION]
{ FOR | AFTER }
{ [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ]
}
AS <sql_statement>
```

where,

FOR | AFTER: Specifies that the DML trigger executes after the modification operations are complete.

{ [INSERT] [,] [UPDATE] [,] [DELETE] }: Specifies the operations that invoke the DML trigger.

Syntax

"AFTER" Triggers

An AFTER trigger is executed on completion of INSERT, DELETE operations. AFTER triggers can be created only table can have multiple AFTER triggers defined for eac and DELETE operation. If multiple AFTER triggers are cr same table, you must define the order in which the tr executed.

An AFTER trigger is executed when the constraint chec completed. Also, the trigger is executed after the Inse Deleted tables are created.

Syntax

Snippet

The following code creates an AFTER DELETE trigger on the Employee_Details table. If any employee record is deleted from the table, the AFTER DELETE trigger activates. The trigger displays the number of employee records deleted from the table.

```
CREATE TRIGGER Employee_Deletion
          ON Employee_Details
          AFTER DELETE
AS
       BEGIN
       DECLARE @num int;
       SELECT @num = COUNT(*) FROM deleted
       PRINT 'No. of employees deleted = ' + CONVERT(varchar(5),
@num)
    END
```

The following code deletes a record from the Employee_Details table.

```
DELETE FROM Employee_Details WHERE EmpID=17
```

The following message is displayed as specified by the PRINT statement:

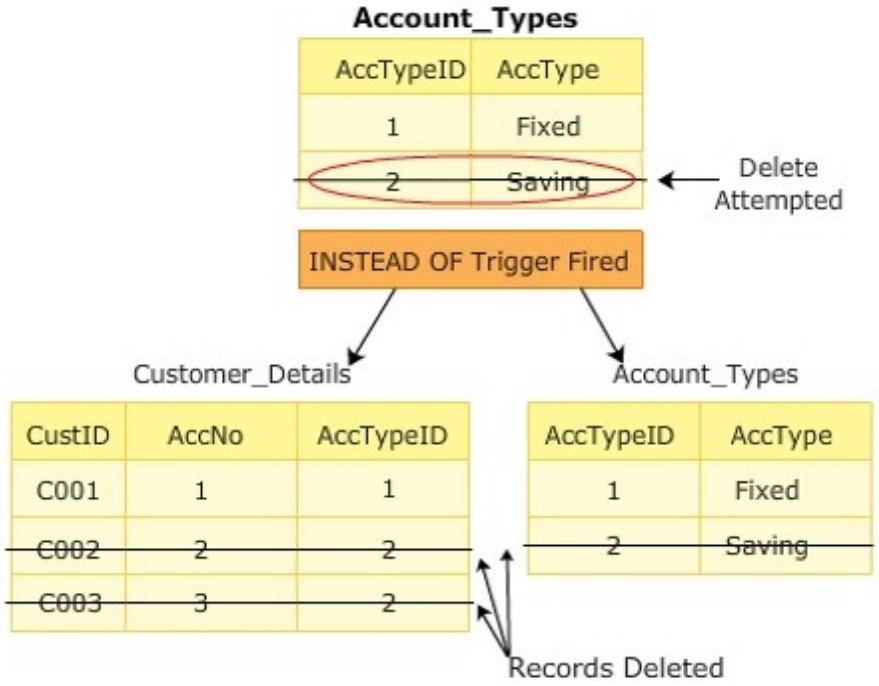
```
No. of employees deleted = 1
```

Snippet

"INSTEAD OF" Triggers

An `INSTEAD OF` trigger is executed in place of the `INSERT`, `UPDATE` or `DELETE` operation. `INSTEAD OF` triggers can be created on tables as well as views. A table or a view can have only one `INSTEAD OF` trigger defined for each `INSERT`, `UPDATE` and `DELETE` operation.

The `INSTEAD OF` triggers are executed before constraint checks are performed on the table. These triggers are executed after the creation of the Inserted and Deleted tables. The `INSTEAD OF` triggers increase the variety of types of updates that you can perform against the view.



Syntax

Snippet

More

"INSTEAD OF" Triggers

An `INSTEAD OF` trigger is executed in place of the `INSERT`, `UPDATE` or `DELETE` operation. `INSTEAD OF` triggers can be created on tables as well as views. A table or a view can have only one `INSTEAD OF` trigger defined for each `INSERT`, `UPDATE` and `DELETE` operation.

The `INSTEAD OF` triggers are executed before constraint checks are performed on the table. These triggers are executed after the creation of the Inserted and Deleted tables. The `INSTEAD OF` triggers increase the variety of types of updates that you can perform against the view.

Account_Types

AccTypeID	AccType
1	Fixed
2	Saving

← Delete Attempted

The following is the syntax for creating an `INSTEAD OF` trigger.

```
CREATE TRIGGER <trigger_name>
ON { <table_name> | <view_name> }
{ FOR | AFTER | INSTEAD OF }
{ [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ] }
AS <sql_statement>
```

where,

`view_name`: Specifies the view on which the DML trigger is created.

`INSTEAD OF`: Specifies that the DML trigger executes in place of the modification operations. These triggers are not defined on updatable views using `WITH CHECK OPTION`.

Syntax

Syntax

Snippet

More

"INSTEAD OF" Triggers

An `INSTEAD OF` trigger is execut
`DELETE` operation. `INSTEAD OF` t
well as views. A table or a view
defined for each `INSERT`, `UPDATE`

The `INSTEAD OF` triggers are exe
performed on the table. These
of the Inserted and Deleted tab
the variety of types of updates

Syntax

Sni

More

The following code creates an `INSTEAD OF DELETE` trigger on the `Account_Types` table. If any record in the `Account_Types` table is deleted, the corresponding records in the `Customer_Details` table will be removed.

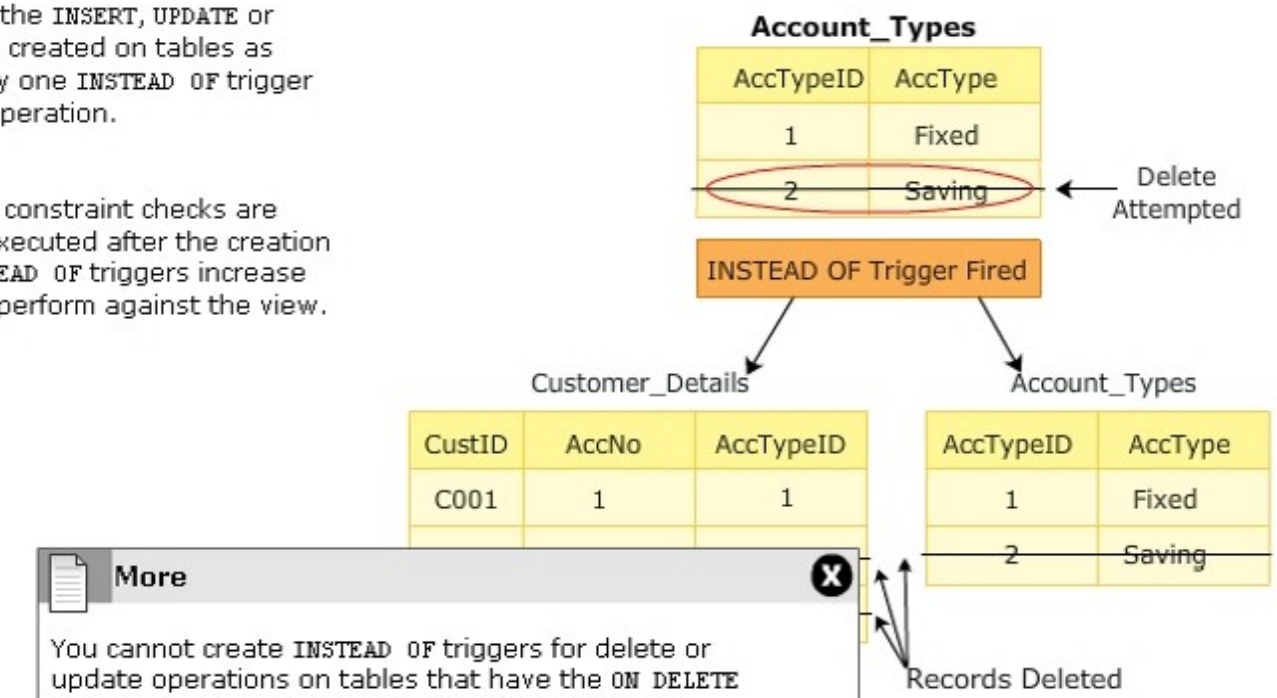
```
CREATE TRIGGER Delete_AccType
ON Account_Types
INSTEAD OF DELETE
AS
    BEGIN
    DELETE FROM Customer_Details WHERE AccTypeID IN
            (SELECT AccTypeID FROM deleted)
    DELETE FROM Account_Types WHERE AccTypeID IN
    (SELECT AccTypeID FROM deleted)
    END
```

Snippet

"INSTEAD OF" Triggers

An `INSTEAD OF` trigger is executed in place of the `INSERT`, `UPDATE` or `DELETE` operation. `INSTEAD OF` triggers can be created on tables as well as views. A table or a view can have only one `INSTEAD OF` trigger defined for each `INSERT`, `UPDATE` and `DELETE` operation.

The `INSTEAD OF` triggers are executed before constraint checks are performed on the table. These triggers are executed after the creation of the Inserted and Deleted tables. The `INSTEAD OF` triggers increase the variety of types of updates that you can perform against the view.



More

You cannot create `INSTEAD OF` triggers for delete or update operations on tables that have the `ON DELETE` cascade and `ON UPDATE` cascade options selected.

Syntax

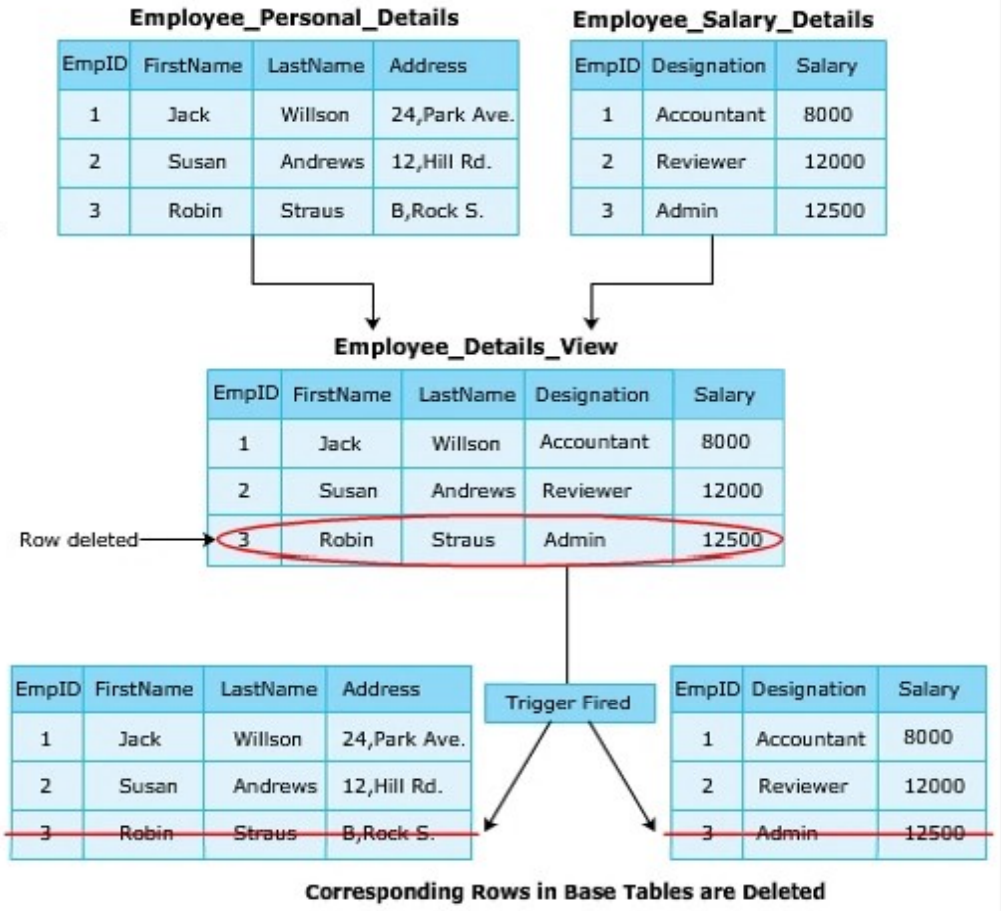
Snippet

More

Using "INSTEAD OF" triggers with Views

`INSTEAD OF` triggers can be specified on tables as well as views. This trigger executes instead of the original triggering action. `INSTEAD OF` triggers provide a wider range and types of updates that you can perform against a view. Each table or view is limited to only one `INSTEAD OF` trigger for each triggering action (`INSERT`, `UPDATE`, or `DELETE`).

You cannot create an `INSTEAD OF` trigger on views that have the `WITH CHECK OPTION` clause defined.



Snippet

Using "INSTEAD OF" triggers with Views

`INSTEAD OF` triggers can be specified on tables as well as views. This trigger executes instead of the original triggering action. `INSTEAD OF` triggers provide a wider range and types of updates that you can perform against a view. Each table or view is limited to only one `INSTEAD OF` trigger for each triggering action (`INSERT`, `UPDATE`, or `DELETE`).

You cannot create an `INSTEAD OF` trigger on views that have the `WITH CHECK OPTION` clause defined.

Snippet

The following code creates a table `Employee_Personal_Details`.

```
CREATE TABLE Employee_Personal_Details
(
   EmpID     int NOT NULL,
   FirstName varchar(30) NOT NULL,
   LastName  varchar(30) NOT NULL,
   Address   varchar(30)
)
```

The following code creates a table `Employee_Salary_Details`.

```
CREATE TABLE Employee_Salary_Details
(
   EmpID        int NOT NULL,
   Designation  varchar(30),
   Salary       int NOT NULL
)
```

Continued...

Snippet

Using "INSTEAD OF" triggers with Views

`INSTEAD OF` triggers can be specified on tables as well as views. This trigger executes instead of the original triggering action. `INSTEAD OF` triggers provide a wider range and types of updates that you can perform against a view. Each table or view is limited to only one `INSTEAD OF` trigger for each triggering action (`INSERT`, `UPDATE`, or `DELETE`).

You cannot create an `INSTEAD OF` trigger on views that have the `WITH CHECK OPTION` clause defined.

Snippet

The following code creates a view `Employee_Details_View` using columns from the `Employee_Personal_Details` and `Employee_Salary_Details` tables by joining the two tables on the `EmpID` column.

```
CREATE VIEW Employee_Details_View
AS
SELECT e1.EmpID, FirstName, LastName, Designation,
Salary
FROM Employee_Personal_Details e1
JOIN Employee_Salary_Details e2
ON e1.EmpID = e2.EmpID
```

The following code creates an `INSTEAD OF DELETE` trigger `Delete_Employees` on the view Employee_Details_View. When a row is deleted from the view, the trigger gets activated. It deletes the corresponding records from the base tables of the view, namely the `Employee_Personal_Details` and the `Employee_Salary_Details`.

```
CREATE TRIGGER Delete_Employees
ON Employee_Details_View
```

Continued...

Snippet

Using "INSTEAD OF" triggers with Views

`INSTEAD OF` triggers can be specified on tables as well as views. This trigger executes instead of the original triggering action. `INSTEAD OF` triggers provide a wider range and types of updates that you can perform against a view. Each table or view is limited to only one `INSTEAD OF` trigger for each triggering action (`INSERT`, `UPDATE`, or `DELETE`).

You cannot create an `INSTEAD OF` trigger on views that have the `WITH CHECK OPTION` clause defined.

Snippet

```
INSTEAD OF DELETE
AS
    BEGIN
      DELETE FROM Employee_Salary_Details WHERE EmpID
IN
      (SELECT EmpID FROM deleted)

      DELETE FROM Employee_Personal_Details WHERE
EmpID IN
      (SELECT EmpID FROM deleted)
END
```

The following code deletes a row from the view `Employee_Details_View` where `EmpID=2`.

```
DELETE FROM Employee_Details_View WHERE EmpID=2
```

The above code causes the `Delete_Employees` trigger to fire. The trigger deletes records from the `Employee_Personal_Details` and the `Employee_Salary_Details` tables where `EmpID=2`.

Snippet

Knowledge Check

Can you match the types of DML triggers in SQL Server 2005 against their corresponding descriptions?

Click an option on the left and its matching option on the right, and then click on **Submit**.

	Description		DML Trigger
(A)	Executes when you replace an existing record with a new value.	(1)	INSERT
(B)	Executes on completion of the modification operations.	(2)	UPDATE
(C)	Executes in place of the modification operations.	(3)	DELETE
(D)	Executes when you add a record on a table.	(4)	AFTER
(E)	Executes when you remove a record from a table.	(5)	INSTEAD OF

Submit

44 | 16

Score

Knowledge Check

Can you match the types of DML triggers in SQL Server 2005 against their corresponding descriptions?

Click an option on the left and its matching option on the right, and then click on **Submit**.

	Description		DML Trigger
(A)	Executes when you replace an existing record with a new value.	(1)	INSERT
(B)	Executes on completion of the modification operations.	(2)	UPDATE
(C)	Executes in place of the modification operations.	(3)	DELETE
(D)	Executes when you add a record on a table.	(4)	AFTER
(E)	Executes when you remove a record from a table.	(5)	INSTEAD OF

Correct

The correct answers are displayed.

Submit

Knowledge Check

Which of these statements about DML triggers in SQL Server 2005 are true and which statements are false?

Select an option for each statement and then click on **Submit**.

	Statements	True	False
(A)	DML triggers can perform multiple actions for each modification statement.	○	○
(B)	The Inserted and Deleted tables are created by SQL Server 2005 when a new table is created in the database.	○	○
(C)	`UPDATE` triggers do not use the Deleted table to update records in a table.	○	○
(D)	Deleted triggers do not use the Inserted table to delete records from a table.	○	○
(E)	`AFTER` triggers can be defined on tables and views.	○	○

Submit

45 | 16

Score

Knowledge Check

Which of these statements about DML triggers in SQL Server 2005 are true and which statements are false?

Select an option for each statement and then click on **Submit**.

	Statements	True	False
(A)	DML triggers can perform multiple actions for each modification statement.	◉	○
(B)	The Inserted and Deleted tables are created by SQL Server 2005 when a new table is created in the database.	○	◉
(C)	`UPDATE` triggers do not use the Deleted table to update records in a table.	○	◉
(D)	Deleted triggers do not use the Inserted table to delete records from a table.	◉	○
(E)	`AFTER` triggers can be defined on tables and views.	○	◉

Correct

The correct answers are displayed.

Submit

Lesson Review

In this second lesson, **Creating DML Triggers**, you learnt to:

- Describe `INSERT` Triggers.
- Describe `UPDATE` Triggers.
- Describe `DELETE` Triggers.
- Describe `AFTER` Triggers.
- Describe `INSTEAD OF` Triggers.

Lesson Overview

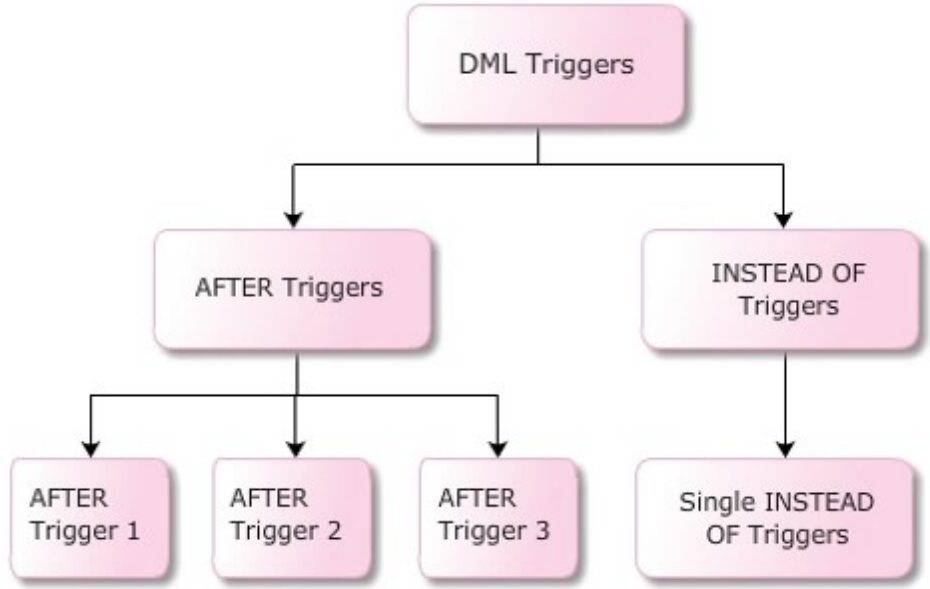
In this third lesson, **Working with DML Triggers**, you will learn to:

- Explain trigger order in DML triggers.
- Describe how to view DML triggers.
- Describe how to modify DML triggers.
- Describe how to drop DML triggers.

DML Triggers

SQL Server 2005 allows you to create multiple `AFTER` triggers for each triggering action (such as `UPDATE, INSERT and DELETE`) on a table. However, you can create only one `INSTEAD OF` trigger for each triggering action on a table.

When you have multiple `AFTER` triggers on a triggering action, all of these triggers must have a different name. An `AFTER` trigger can include a number of SQL statements that perform different functions.

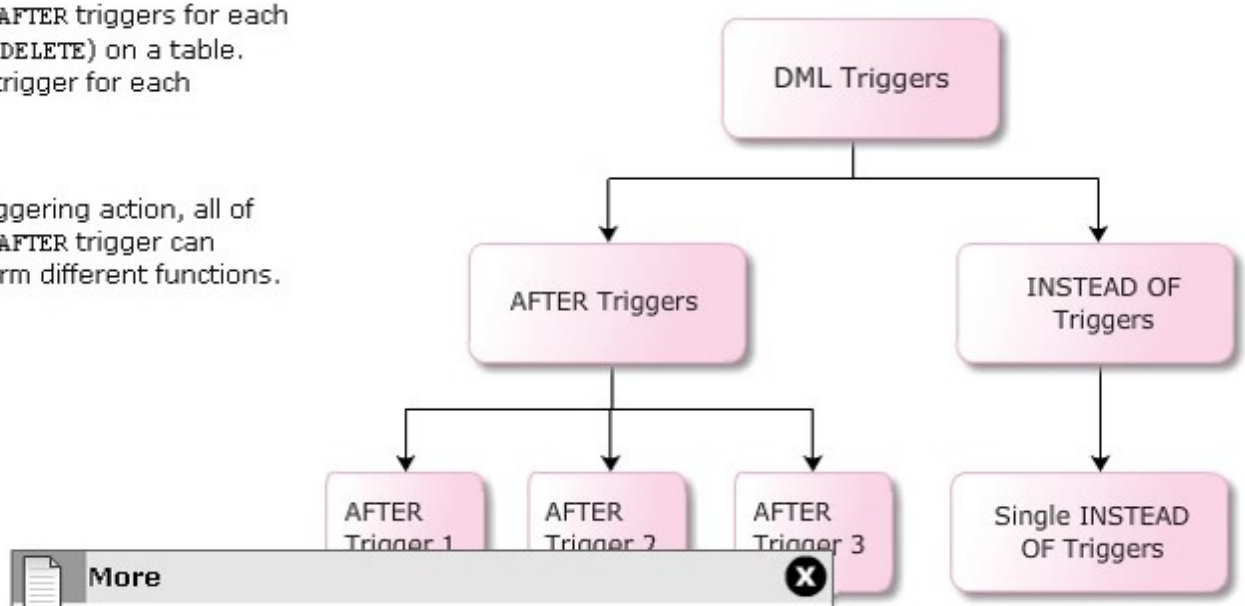


More

DML Triggers

SQL Server 2005 allows you to create multiple `AFTER` triggers for each triggering action (such as `UPDATE, INSERT and DELETE`) on a table. However, you can create only one `INSTEAD OF` trigger for each triggering action on a table.

When you have multiple `AFTER` triggers on a triggering action, all of these triggers must have a different name. An `AFTER` trigger can include a number of SQL statements that perform different functions.



More

A single `AFTER` trigger can be invoked by more than one triggering action.

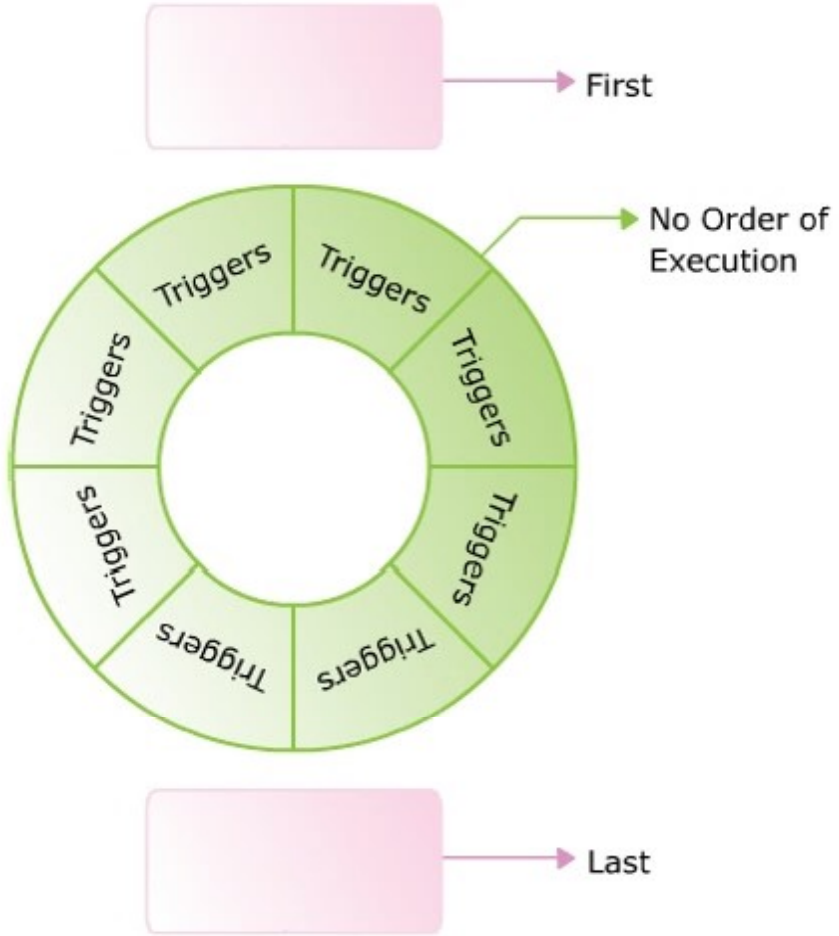
More

Execution Order of DML Trigger

SQL Server 2005 allows you to specify which `AFTER` triggers is to be executed first and which to be executed last. All `AFTER` triggers invoked between the first and last triggers have no definite order of execution.

All the triggering actions have a first and last trigger defined for them. But no two triggering actions on a table can have the same first and last triggers.

You can use the `sp_settriggerorder` stored procedure to define the order of DML `AFTER` triggers.



Syntax

Snippet

Execution Order of DML Trigger

SQL Server 2005 allows you to specify which `AFTER` triggers is to be executed first and which to be executed last. All `AFTER` triggers invoked between the first and last triggers have no definite order of execution.

All the triggering actions have a first and last trigger defined for them. But no two triggering actions on a table can have the same first and last triggers.

You can use the `sp_settriggerorder` stored procedure to define the order of DML `AFTER` triggers.

Syntax

Snippet

The following is the syntax for specifying execution order of multiple `AFTER` DML triggers.

```
sp_settriggerorder [ @triggername = ] '[
triggerschema. ] triggername'
        , [ @order = ] 'value'
        , [ @stmttype = ] 'statement_type'
```

where,

`[ triggerschema. ] triggername:` Is the name of the DML or DDL trigger and the schema to which it belongs and whose order needs to be specified.

`value:` Specifies the execution order of the trigger as `FIRST`, `LAST` or `NONE`. If `FIRST` is specified, then the trigger is fired first. If `LAST` is specified, the trigger is fired last. If `NONE` is specified, the order of the firing of the trigger is undefined.

`statement_type:` Specifies the type of SQL statement (`INSERT`, `UPDATE` or `DELETE`) that invokes the DML trigger.

Syntax

Execution Order of DML Trigger

SQL Server 2005 allows you to specify which `AFTER` triggers is to be executed first and which to be executed last. All `AFTER` triggers invoked between the first and last triggers have no definite order of execution.

All the triggering actions have a first and last trigger defined for them. But no two triggering actions on a table can have the same first and last triggers.

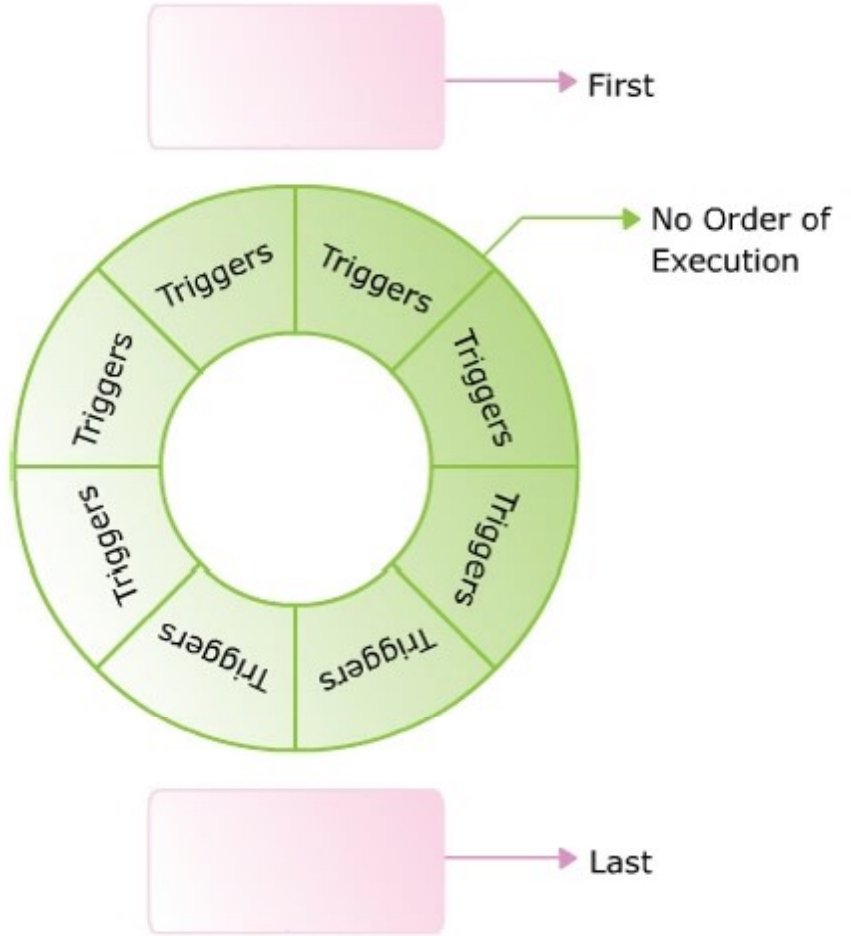
The following code first executes the `CheckWithdrawal_Amount` trigger defined on the `Account_Transactions` table when the `INSERT` operation is performed on the `Withdrawal` column of the table.

```
EXEC sp_settriggerorder @triggername =
'CheckWithdrawal_Amount', @order = 'FIRST', @stmttype =
'INSERT'
```

Snippet

Syntax

Snippet



Viewing Definitions of DML Triggers

A trigger definition includes the trigger name, the table on which the trigger is created, the triggering actions and the SQL statements that are executed. SQL Server 2005 provides `sp_helptext` stored procedure to retrieve the trigger definitions.

The DML trigger name must be specified as the parameter when executing `sp_helptext`.

```
1  sp_helptext 'CheckWithdrawal_Amount'
```

Results | Messages

	Text
1	CREATE TRIGGER CheckWithdrawal_Amount
2	ON Account_Transactions
3	FOR INSERT
4	AS
5	IF(SELECT Withdrawal From inserted) > 50000
6	BEGIN
7	PRINT 'Withdrawal amount cannot exceed 50000'
8	ROLLBACK TRANSACTION
9	END

Syntax

Snippet

More

Viewing Definitions of DML Triggers

A trigger definition includes the trigger name, the table on which the trigger is created, the triggering actions and the SQL statements that are executed. SQL Server 2005 provides `sp_helptext` stored procedure to retrieve the trigger definitions.

The following is the syntax for viewing a DML trigger.

```
sp_helptext '<DML_trigger_name>'
```

where,
`DML_trigger_name`: Specifies the name of the DML trigger whose definitions are to be displayed.

Syntax

```
1  sp_helptext 'CheckWithdrawal_Amount'
```

Results | Messages

	Text
1	CREATE TRIGGER CheckWithdrawal_Amount
2	ON Account_Transactions
3	FOR INSERT
4	AS
5	IF(SELECT Withdrawal From inserted) > 50000
6	BEGIN
7	PRINT 'Withdrawal amount cannot exceed 50000'
8	ROLLBACK TRANSACTION
9	END

Syntax | Snippet

More

Viewing Definitions of DML Triggers

A trigger definition includes the trigger name, the table on which the trigger is created, the triggering actions and the SQL statements that are executed. SQL Server 2005 provides `sp_helptext` stored procedure to retrieve the trigger definitions.

The following code displays the definitions of the trigger, `CheckWithdrawal_Amount` created on the `Account_Transactions` table.

```
sp_helptext 'CheckWithdrawal_Amount'
```

Snippet

```
1  sp_helptext 'CheckWithdrawal_Amount'
```

Results | Messages

	Text
1	CREATE TRIGGER CheckWithdrawal_Amount
2	ON Account_Transactions
3	FOR INSERT
4	AS
5	IF(SELECT Withdrawal From inserted) > 50000
6	BEGIN
7	PRINT 'Withdrawal amount cannot exceed 50000'
8	ROLLBACK TRANSACTION
9	END

Syntax | Snippet

More

Viewing Definitions of DML Triggers

A trigger definition includes the trigger name, the table on which the trigger is created, the triggering actions and the SQL statements that are executed. SQL Server 2005 provides `sp_helptext` stored procedure to retrieve the trigger definitions.

The DML trigger name must be specified as the parameter when executing `sp_helptext`.

```
1  sp_helptext 'CheckWithdrawal_Amount'
```

Results | Messages

	Text
1	CREATE TRIGGER CheckWithdrawal_Amount
2	ON Account_Transactions
3	FOR INSERT
4	AS
5	IF(SELECT Withdrawal From inserted) > 50000
6	BEGIN
7	PRINT 'Withdrawal amount cannot exceed 50000'
8	ROLLBACK TRANSACTION
9	END

More

Trigger definition cannot be viewed if the definition is encrypted.

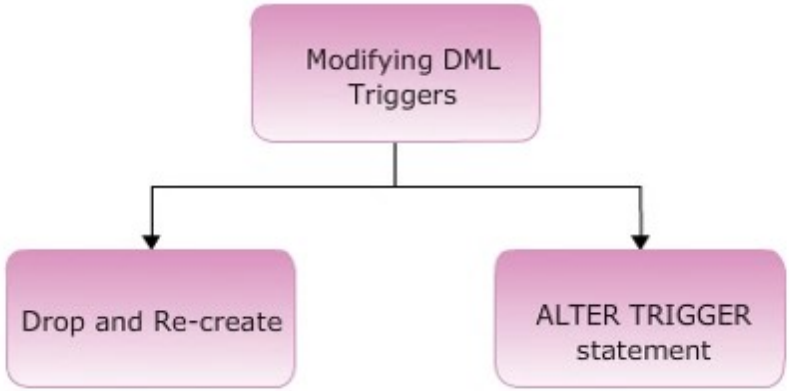
Syntax | Snippet | More

Modifying Definitions of DML Triggers

Trigger parameters are defined at the time of creating a trigger. These parameters include the type of triggering action that invokes the trigger and the SQL statements that are executed. If you want to modify any of these parameters for a DML trigger, you can do so in one of two ways:

- Drop and re-create the trigger with the new parameters.
- Change the parameters using the `ALTER TRIGGER` statement.

If the object referencing a DML trigger is renamed, the trigger must be modified to reflect the change in object name.



Syntax

Snippet

More

Modifying Definitions of DML Triggers

Trigger parameters are defined at the time of creating a trigger. These parameters include the type of triggering action that invokes the trigger and the SQL statements that are executed. If you want to modify any of these parameters for a DML trigger, you can do so in one of two ways:

- Drop and re-create the trigger with the new parameters.
- Change the parameters using the `ALTER TRIGGER` statement.

If the object referencing a DML trigger is renamed, the trigger must be modified to reflect the change in object name.

Syntax

Snippet

More

The following is the syntax for modifying a DML trigger.

```
ALTER TRIGGER <trigger_name>
ON { <table_name> | <view_name> }
[WITH ENCRYPTION]
{ FOR | AFTER | INSTEAD OF }
{ [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ] }
AS <sql_statement>
```

where,

`WITH ENCRYPTION`: Specifies that the DML trigger definitions are not displayed.

`FOR | AFTER`: Specifies that the DML trigger executes after the modification operations are complete.

`INSTEAD OF`: Specifies that the DML trigger executes in place of the modification operations.

Syntax

Modifying Definitions of DML Trig

Trigger parameters are defined at the time of creatin
These parameters include the type of triggering actio
the trigger and the SQL statements that are execute
modify any of these parameters for a DML trigger, yo
one of two ways:

- Drop and re-create the trigger with the new para
- Change the parameters using the `ALTER TRIGGER`

If the object referencing a DML trigger is renamed, th
be modified to reflect the change in object name.

Syntax

Snippet

More

The following code alters the `CheckWithdrawal_Amount` trigger created on the `Account_Transactions` table using the `WITH ENCRYPTION` option.

```
ALTER TRIGGER CheckWithdrawal_Amount
ON Account_Transactions
WITH ENCRYPTION
FOR INSERT
AS
      IF (SELECT Withdrawal From inserted) > 50000
      BEGIN
      PRINT 'Withdrawal amount cannot exceed 50000'
      ROLLBACK TRANSACTION
     END
```

Now if you try to view the definition of the `CheckWithdrawal_Amount` trigger using the sp_helptext stored procedure, the following error message is displayed:

```
The text for object 'CheckWithdrawal_Amount' is encrypted.
```

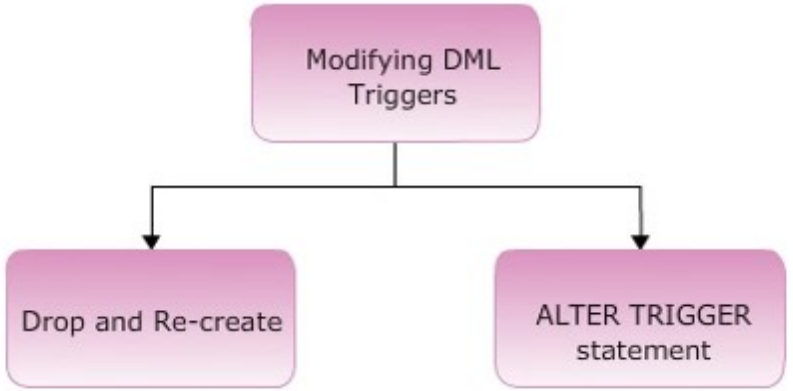
Snippet

Modifying Definitions of DML Triggers

Trigger parameters are defined at the time of creating a trigger. These parameters include the type of triggering action that invokes the trigger and the SQL statements that are executed. If you want to modify any of these parameters for a DML trigger, you can do so in one of two ways:

- Drop and re-create the trigger with the new parameters.
- Change the parameters using the `ALTER TRIGGER` statement.

If the object referencing a DML trigger is renamed, the trigger must be modified to reflect the change in object name.



Syntax

Snippet

More

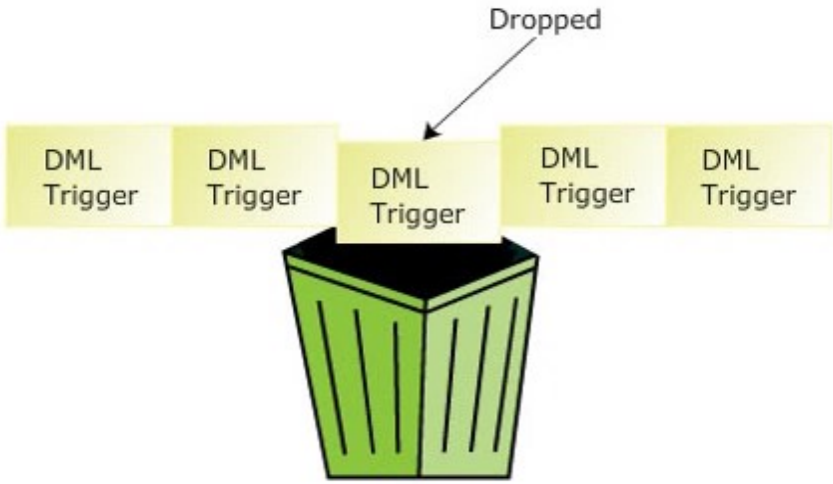
More

A DML trigger can be encrypted to hide its definition.

Dropping DML Triggers

SQL Server 2005 provides the option of dropping a DML trigger created on a table if the trigger is no longer required. The trigger can be dropped using the `DROP TRIGGER` statement. Multiple triggers can be dropped using a single drop trigger statement.

When a table is dropped, all the triggers defined on that table are also dropped.



Syntax

Snippet

More

Dropping DML Triggers

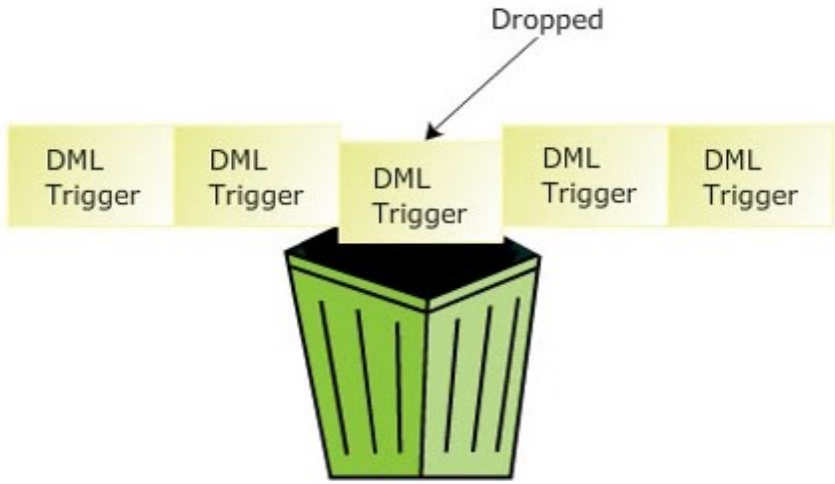
SQL Server 2005 provides the option of dropping a DML trigger created on a table if the trigger is no longer required. The trigger can be dropped using the `DROP TRIGGER` statement. Multiple triggers can be dropped using a single drop trigger statement.

The following is the syntax for dropping DML triggers.

```
DROP TRIGGER <DML_trigger_name> [ ,...n ]
```

where,
`DML_trigger_name`: Specifies the name of the DML trigger to be dropped.
`[ ,...n ]`: Specifies that multiple DML triggers can be dropped.

Syntax



Syntax

Snippet

More

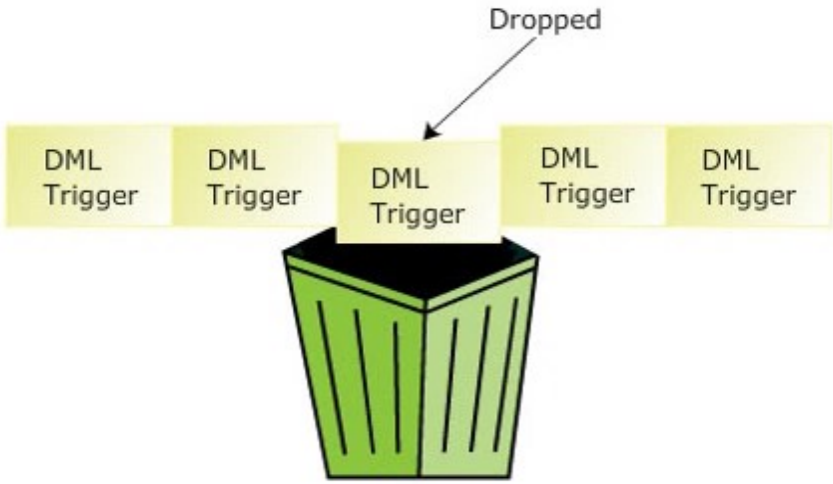
Dropping DML Triggers

SQL Server 2005 provides the option of dropping a DML trigger created on a table if the trigger is no longer required. The trigger can be dropped using the `DROP TRIGGER` statement. Multiple triggers can be dropped using a single drop trigger statement.

The following code drops the `CheckWithdrawal_Amount` trigger created on the `Account_Transactions` table.

```
DROP TRIGGER CheckWithdrawal_Amount
```

Snippet



Syntax

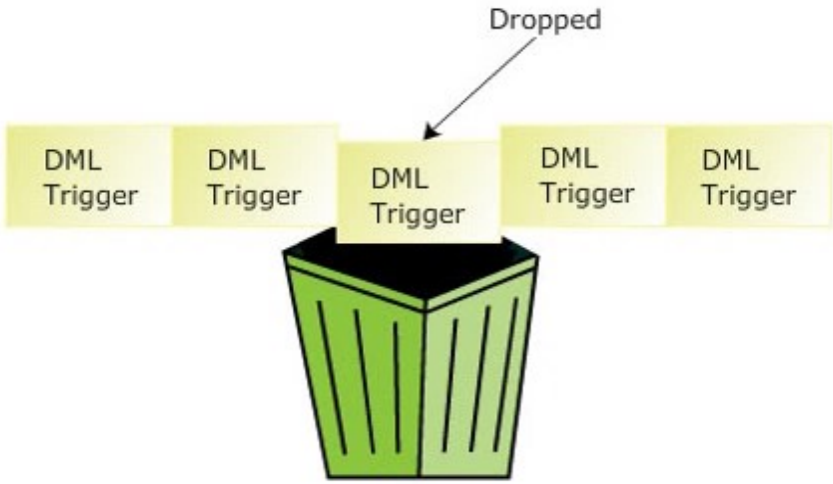
Snippet

More

Dropping DML Triggers

SQL Server 2005 provides the option of dropping a DML trigger created on a table if the trigger is no longer required. The trigger can be dropped using the `DROP TRIGGER` statement. Multiple triggers can be dropped using a single drop trigger statement.

When a table is dropped, all the triggers defined on that table are also dropped.



Syntax

Snippet

More

More

When the DML trigger is deleted from the table, the information about the trigger is also removed from the catalog views.

45 | 16

Score

Knowledge Check

Which of these statements about the working with DML triggers of SQL Server 2005 are true and which statements are false?

Select an option for each statement and then click on **Submit**.

	Statements	True	False
(A)	Each triggering action can have multiple `AFTER` triggers.	○	○
(B)	Two triggering actions on a table can have the same first and last triggers.	○	○
(C)	Trigger definition can be viewed if the information is not encrypted.	○	○
(D)	DML trigger definition can be modified by dropping and recreating the trigger.	○	○
(E)	DML trigger definition can be viewed using the `sp_helptext` stored procedure.	○	○

Submit

Knowledge Check

Which of these statements about the working with DML triggers of SQL Server 2005 are true and which statements are false?

Select an option for each statement and then click on **Submit**.

	Statements	True	False
(A)	Each triggering action can have multiple `AFTER` triggers.	◉	○
(B)	Two triggering actions on a table can have the same first and last triggers.	○	◉
(C)	Trigger definition can be viewed if the information is not encrypted.	◉	○
(D)	DML trigger definition can be modified by dropping and recreating the trigger.	◉	○
(E)	DML trigger definition can be viewed using the `sp_helptext` stored procedure.	◉	○

Correct

The correct answers are displayed.

Submit

Lesson Review

In this third lesson, **Working with DML Triggers**, you learnt to:

- Explain trigger order in DML triggers.
- Describe how to view DML triggers.
- Describe how to modify DML triggers.
- Describe how to drop DML triggers.

Lesson Overview

In this fourth lesson, **Working with DDL Triggers**, you will learn to:

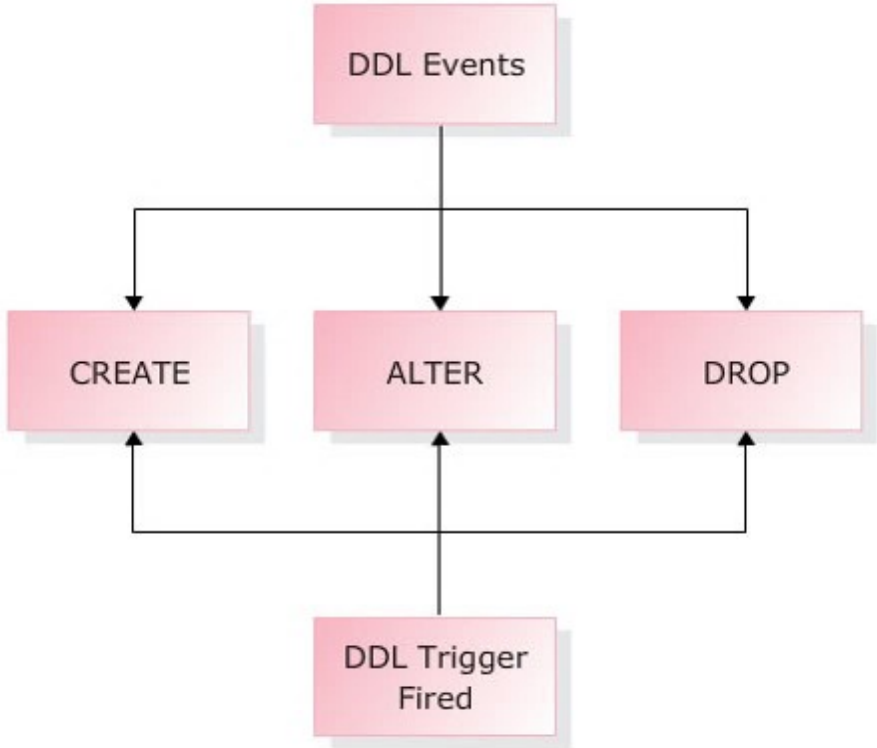
- Explain DDL trigger for `CREATE`.
- Explain DDL trigger for `ALTER`.
- Explain DDL trigger for `DROP`.
- Describe how to view DDL triggers.
- Describe how to modify DDL triggers.
- Describe how to drop DDL triggers.

DDL Triggers

A Data Definition Language (DDL) trigger is a new type of trigger introduced in SQL Server 2005. DDL triggers execute stored procedures when DDL events such as `CREATE, ALTER and DROP` statements occur in the database or the server. DDL triggers can operate only on completion of the DDL events.

DDL triggers can be used to prevent modifications in the database schema. Schema is collection of objects such as tables, views, etc in a database. DDL triggers can invoke an event or display a message based on the modifications attempted on the schema.

DDL triggers are defined either at the database level or at the server level.



DDL Triggers

A Data Definition Language (DDL) trigger is a new type of trigger introduced in SQL Server 2005. DDL triggers execute stored procedures when DDL events such as `CREATE, ALTER and DROP` statements occur in the database or the server. DDL triggers can operate only on completion of the DDL events.

DDL triggers can be used to prevent modifications in the database schema. Schema is collection of objects such as tables, views, etc in a database. DDL triggers can invoke an event or display a message based on the modifications attempted on the schema.

DDL triggers are defined either at the database level or at the server level.

Syntax

The following is the syntax for creating DDL triggers.

```
CREATE TRIGGER <trigger_name>
ON { ALL SERVER | DATABASE }
[WITH ENCRYPTION]
{ FOR | AFTER } { <event_type> }
AS <sql_statement>
```

where,

`ALL SERVER`: Specifies that the DDL trigger executes when DDL events occur in the current server.

`DATABASE`: Specifies that the DDL trigger executes when DDL events occur in the current database.

`event_type`: Specifies the name of the DDL event that invokes the DDL trigger.

Syntax

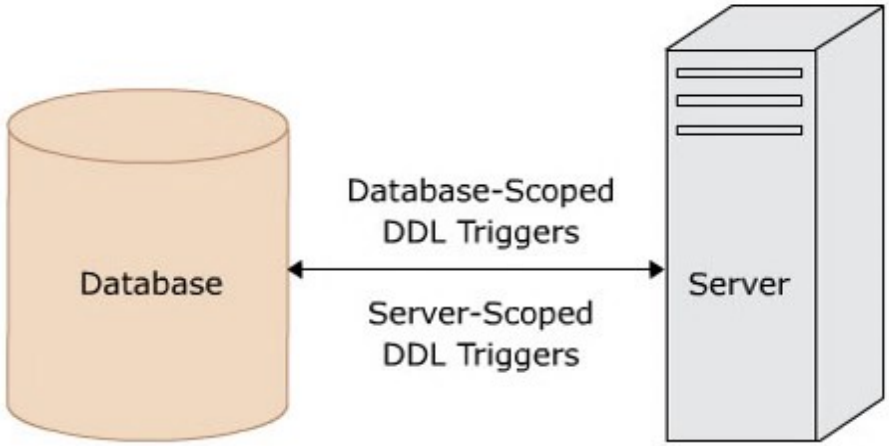
Scope of DDL Triggers

DDL triggers are invoked by SQL statements executed either in the current database or on the current server. For example, a DDL trigger created for a `CREATE TABLE` statement executes on the `CREATE TABLE` event in the database. A DDL trigger created for a `CREATE LOGIN` statement executes on the `CREATE LOGIN` event in the server.

The scope of the DDL trigger depends on whether the trigger executes for database events or server events. Accordingly, the DDL triggers are classified as:

- Database-Scoped DDL Triggers
- Server-Scoped DDL Triggers

Click on each link to learn more.

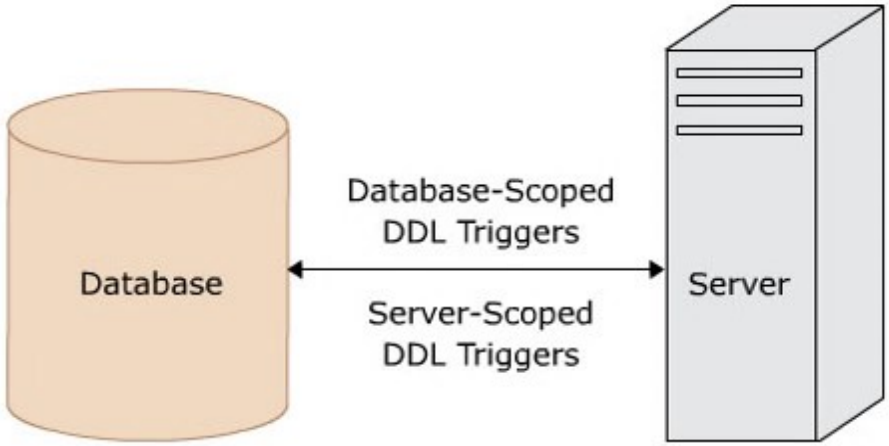


Scope of DDL Triggers

DDL triggers are invoked by SQL statements executed either in the current database or on the current server. For example, a DDL trigger created for a `CREATE TABLE` statement executes on the `CREATE TABLE` event in the database. A DDL trigger created for a `CREATE LOGIN` statement executes on the `CREATE LOGIN` event in the server.

The scope of the DDL trigger depends on whether the trigger executes for database events or server events. Accordingly, the DDL triggers are classified as:

- Database-Scoped DDL Triggers
- Server-Scoped DDL Triggers



Click on each link to learn more.

Database-Scoped DDL Triggers

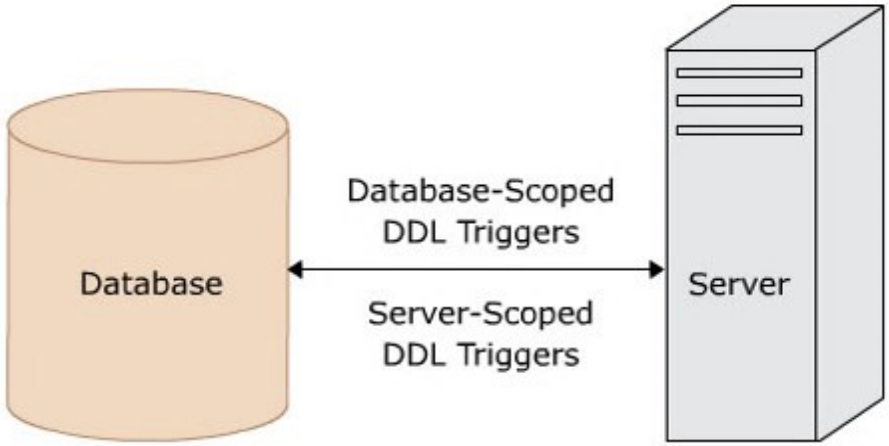
Database-scoped DDL triggers are invoked by the events that modify the database schema. These triggers are stored in the database and execute on DDL events, except those related to temporary tables.

Scope of DDL Triggers

DDL triggers are invoked by SQL statements executed either in the current database or on the current server. For example, a DDL trigger created for a `CREATE TABLE` statement executes on the `CREATE TABLE` event in the database. A DDL trigger created for a `CREATE LOGIN` statement executes on the `CREATE LOGIN` event in the server.

The scope of the DDL trigger depends on whether the trigger executes for database events or server events. Accordingly, the DDL triggers are classified as:

- Database-Scoped DDL Triggers
- Server-Scoped DDL Triggers



Click on each link to learn more.

Server-Scoped DDL Triggers

Server-scoped DDL triggers are invoked by DDL events at the server level. These triggers are stored in the `master` database.

Defining DDL Triggers for "CREATE" Events

DDL triggers defined for create events at the database-level or server-level are invoked by `CREATE` statements. While creating a DDL trigger, you must specify the name of a create event in the `CREATE TRIGGER` statement. These create events can be `CREATE_TABLE`, `CREATE_INDEX`, `CREATE_STATISTICS`, etc. The DDL trigger is invoked only after the event execution is complete.

```
1  CREATE TABLE Supplier_Details
2  (
3      SuppID  varchar(3) NOT NULL,
4      FirstName   varchar(30),
5      LastName    varchar(30),
6      City        varchar(40)
7  )
```

Messages

```
You do not have the permission to create tables
Msg 3609, Level 16, State 2, Line 1
The transaction ended in the trigger. The batch has been aborted.
```

Snippet

More

Defining DDL Triggers for "CREATE" Events

DDL triggers defined for create events at the database-level or server-level are invoked by `CREATE` statements. While creating a DDL trigger, you must specify the name of a create event in the `CREATE TRIGGER` statement. These create events can be `CREATE_TABLE`, `CREATE_INDEX`, `CREATE_STATISTICS`, etc. The DDL trigger is invoked only after the event execution is complete.

Snippet

More

The following code creates the `Create_Permission` DDL trigger at the database level. This trigger does not allow users to create tables in the database. The create operation is rolled back using the `ROLLBACK` statement.

```
CREATE TRIGGER Create_Permission
ON DATABASE
FOR CREATE_TABLE
AS
   BEGIN
      PRINT 'You do not have the permission to
create tables'
      ROLLBACK
   END
```

Continued...

Snippet

Defining DDL Triggers for "CREATE" Events

DDL triggers defined for create events at the database-level or server-level are invoked by `CREATE` statements. While creating a DDL trigger, you must specify the name of a create event in the `CREATE TRIGGER` statement. These create events can be `CREATE_TABLE`, `CREATE_INDEX`, `CREATE_STATISTICS`, etc. The DDL trigger is invoked only after the event execution is complete.

Snippet

More

The following code creates a table `Supplier_Details` that causes the `Create_Permission` DDL trigger to be fired.

```
CREATE TABLE Supplier_Details
(
      SuppID varchar(3) NOT NULL,
      FirstName varchar(30),
      LastName varchar(30),
      City varchar(40)
)
```

The following error message is displayed as specified by the `PRINT` statement:

```
You do not have the permission to create tables.
```

Snippet

Defining DDL Triggers for "CREATE" Events

DDL triggers defined for create events at the database-level or server-level are invoked by `CREATE` statements. While creating a DDL trigger, you must specify the name of a create event in the `CREATE TRIGGER` statement. These create events can be `CREATE_TABLE`, `CREATE_INDEX`, `CREATE_STATISTICS`, etc. The DDL trigger is invoked only after the event execution is complete.

```
1  CREATE TABLE Supplier_Details
2  (
3      SuppID  varchar(3) NOT NULL,
4      FirstName   varchar(30),
5      LastName    varchar(30),
6      City        varchar(40)
7  )
```

Messages

```
You do not have the permission to create tables
Msg 3609, Level 16, State 2, Line 1
The transaction ended in the trigger. The batch has been aborted.
```

More

Multiple DDL events can be specified in the `CREATE TRIGGER` statement to fire the trigger.

Snippet

More

Defining DDL Triggers for "ALTER" Events

DDL triggers defined for events that modify objects in the database or on the server are invoked by `ALTER` statements. While creating a DDL trigger, you must specify the name of an alter event in the `CREATE TRIGGER` statement. These alter events can include `ALTER_TABLE, ALTER_INDEX`, etc. The DDL trigger is invoked only after the modification operations are complete.

```
1  ALTER TABLE Supplier_Details ADD State varchar(40)
```

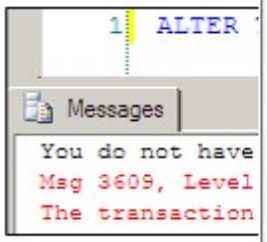
Messages

```
You do not have the permission to alter tables
Msg 3609, Level 16, State 2, Line 1
The transaction ended in the trigger. The batch has been aborted.
```

Snippet

Defining DDL Triggers for "ALTER" Events

DDL triggers defined for events that modify objects in the database or on the server are invoked by `ALTER` statements. While creating a DDL trigger, you must specify the name of an alter event in the `CREATE TRIGGER` statement. These alter events can include `ALTER_TABLE, ALTER_INDEX`, etc. The DDL trigger is invoked only after the modification operations are complete.



The following code creates the `Alter_Permission` DDL trigger at the database level. This trigger does not allow users to modify the existing tables in the database. The alter operation is rolled back using the `ROLLBACK` statement.

```
CREATE TRIGGER Alter_Permission
ON DATABASE
FOR ALTER_TABLE
AS
   BEGIN
    PRINT 'You do not have the permission to
alter tables'
   ROLLBACK
   END
```

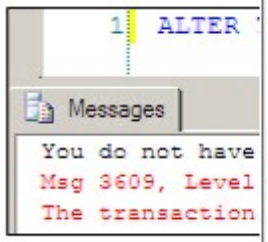
Continued...

Snippet

Snippet

Defining DDL Triggers for "ALTER" Events

DDL triggers defined for events that modify objects in the database or on the server are invoked by `ALTER` statements. While creating a DDL trigger, you must specify the name of an alter event in the `CREATE TRIGGER` statement. These alter events can include `ALTER_TABLE, ALTER_INDEX`, etc. The DDL trigger is invoked only after the modification operations are complete.



The following code tries to alter the table `Supplier_Details` that causes the `Alter_Permission` DDL trigger to be fired:

```
ALTER TABLE Supplier_Details ADD State varchar
(40)
```

The following error message is displayed as specified by the `PRINT` statement:

```
You do not have the permission to alter tables.
```

Snippet

Snippet

Defining DDL Triggers for "DROP" Events

DDL triggers defined for events that delete objects in the database or on the server are invoked by `DROP` statements. While creating a DDL trigger, you must specify the name of a drop event in the `CREATE TRIGGER` statement. These drop events can include `DROP_TABLE`, `DROP_INDEX`, `DROP_STATISTICS`, etc. The DDL trigger is invoked only after the delete operations are complete.

```
1  DROP TABLE Supplier_Details
```

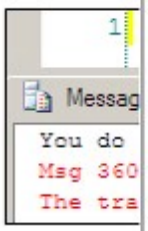
Messages

```
You do not have the permission to drop tables
Msg 3609, Level 16, State 2, Line 1
The transaction ended in the trigger. The batch has been aborted.
```

Snippet

Defining DDL Triggers for "DROP" Events

DDL triggers defined for events that delete objects in the database or on the server are invoked by `DROP` statements. While creating a DDL trigger, you must specify the name of a drop event in the `CREATE TRIGGER` statement. These drop events can include `DROP_TABLE`, `DROP_INDEX`, `DROP_STATISTICS`, etc. The DDL trigger is invoked only after the delete operations are complete.



The following code creates `Drop_Permission` DDL trigger at the database level. This trigger does not allow users to delete the existing tables in the database. The drop operation is rolled back using the `ROLLBACK` statement.

```
CREATE TRIGGER Drop_Permission
ON DATABASE
FOR DROP_TABLE
AS
   BEGIN
   PRINT 'You do not have the permission to drop
tables'
   ROLLBACK
END
```

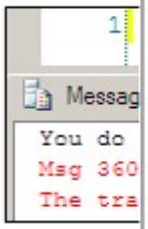
Continued...

Snippet

Snippet

Defining DDL Triggers for "DROP" Events

DDL triggers defined for events that delete objects in the database or on the server are invoked by `DROP` statements. While creating a DDL trigger, you must specify the name of a drop event in the `CREATE TRIGGER` statement. These drop events can include `DROP_TABLE`, `DROP_INDEX`, `DROP_STATISTICS`, etc. The DDL trigger is invoked only after the delete operations are complete.



The following code tries to drop the table `Supplier_Details` that causes the `Drop_Permission` DDL trigger to be fired:

```
DROP TABLE Supplier_Details
```

The following error message is displayed as specified by the `PRINT` statement:

```
You do not have the permission to drop tables
```

Snippet

Snippet

Viewing Metadata of DDL Triggers

The properties of all objects created in SQL Server 2005 are referred to as metadata. This metadata is stored in system tables.

The metadata information of DDL triggers defined at the database level can be viewed using the `sys.triggers` view. The metadata information of DDL triggers defined at the server level can be viewed using the `sys.server_triggers` view. Some of the common columns of these two views are described in the table below.

Column name	Data type	DML Triggers
`object_id`	`int`	ID of the trigger.
`name`	`sysname`	Name of the trigger.
`parent_class`	`tinyint`	Displays 0 (in sys.triggers view for database-level triggers). Displays 1 (in sys.triggers view for DML triggers). Displays 100 (in sys.server_triggers view for server-level triggers).
`parent_id`	`int`	Displays 0 in both the views for DDL triggers.
`create_date`	`datetime`	Displays the date when the trigger was created.

Viewing Metadata of DDL Triggers

The properties of all objects created in SQL Server 2005 are referred to as metadata. This metadata is stored in system tables.

The metadata information of DDL triggers defined at the database level can be viewed using the `sys.triggers` view. The metadata information of DDL triggers defined at the server level can be viewed using the `sys.server_triggers` view. Some of the common columns of these two views are described in the table below.

The following code displays the information related to the `Create_Permission` DDL trigger in the `sys.triggers` view.

```
SELECT * FROM sys.triggers WHERE name='Create_Permission'
```

Snippet

Snippet

		DML Triggers
		he trigger.
		of the trigger.
		ys 0 (in sys.triggers view for se-level triggers). ys 1 (in sys.triggers view for iggers). ys 100 (in sys.server_triggers view for server-level triggers).
parent_id	int	Displays 0 in both the views for DDL triggers.
create_date	datetime	Displays the date when the trigger was created.

Viewing Definition of DDL Triggers

The definition of a DDL trigger can be obtained using the `sys.sql_modules` view by using the object ID of the DDL trigger. The object ID of the trigger can be obtained from the `sys.triggers` or `sys.server_triggers` view by using the DDL trigger name. Hence, an `INNER JOIN` is performed on the `sys.sql_modules` and `sys.triggers` or `sys.server_triggers` views to obtain the trigger definition.

Two of the columns of the `sys.sql_modules` view are described in the table below.

Column name	Data type	Description
`object_id`	`int`	ID of the trigger.
`definition`	`nvarchar(max)`	Definition of the trigger. Displays NULL if the trigger is encrypted.

Viewing Definition of DDL Triggers

The definition of a DDL trigger can be obtained using the `sys.sql_modules` view by using the object ID of the DDL trigger. The object ID of the trigger can be obtained from the `sys.triggers` or `sys.server_triggers` view by using the DDL trigger name. Hence, an `INNER JOIN` is performed on the `sys.sql_modules` and `sys.triggers` or `sys.server_triggers` views to obtain the trigger definition.

Column name	Data type	Description
object_id	int	ID of the trigger.
definition	nvarchar(max)	Definition of the trigger. Displays NULL if the trigger is encrypted.

The following `SELECT` statement obtains the object id of the `Create_Permission` trigger from the `sys.triggers` view. The statement joins the `sys.sql_modules` and `sys.triggers` views by their object IDs to display the definition of `Create_Permission` trigger.

```
SELECT definition FROM sys.sql_modules sm INNER JOIN
sys.triggers t ON sm.object_id = t.object_id WHERE
t.name='Create_Permission'
```

Snippet

Snippet

Modifying Definitions of DDL Triggers

DDL triggers can be modified either by altering the trigger or dropping and re-creating the trigger.

When re-creating a DDL trigger, you can define new parameters.

DDL triggers can be altered using the `ALTER TRIGGER` statement. The trigger definitions can be modified with this statement.

```
1  ALTER TRIGGER Create_Permission
2  ON DATABASE
3  WITH ENCRYPTION
4  FOR CREATE_TABLE
5  AS
6      BEGIN
7         PRINT 'You do not have the permission
8               to create tables'
9         ROLLBACK
10     END
11
12 SELECT definition FROM sys.sql_modules sm
13 INNER JOIN sys.triggers t ON sm.object_id = t.object_id
14 WHERE t.name='Create_Permission'
```

Results | Messages

	definition
1	NULL

Syntax

Snippet

Modifying Definitions of DDL Triggers

DDL triggers can be modified either by altering the trigger or dropping and re-creating the trigger.

When re-creating a DDL trigger, you can define new parameters.

DDL triggers can be altered using the `ALTER TRIGGER` statement. The trigger definitions can be modified with this statement.

The following is the syntax for modifying a DDL trigger.

```
ALTER TRIGGER <trigger_name>
ON { ALL SERVER | DATABASE }
{ FOR | AFTER } { <event_type> }
AS <sql_statement>
```

Syntax

```
1  ALTER TRIGGER Create_Permission
2  ON DATABASE
3  WITH ENCRYPTION
4  FOR CREATE_TABLE
5  AS
6      BEGIN
7         PRINT 'You do not have the permission
8               to create tables'
9         ROLLBACK
10     END
11
12 SELECT definition FROM sys.sql_modules sm
13 INNER JOIN sys.triggers t ON sm.object_id = t.object_id
14 WHERE t.name='Create_Permission'
```

Results | Messages

	definition
1	NULL

Syntax | Snippet

Modifying Definitions of DDL Triggers

DDL triggers can be modified either by altering the trigger or dropping and re-creating the trigger.

When re-creating a DDL trigger, you can define new parameters.

DDL triggers can be altered using the `ALTER TRIGGER` statement. The trigger definitions can be modified with this statement.

```
1  ALTER TRIGGER Create_Permission
2  ON DATABASE
3  WITH ENCRYPTION
4  FOR CREATE_TABLE
5  AS
6      BEGIN
7          PRINT 'You do not have the permission
```

The following code creates the `Create_Permission` trigger at the database level and this trigger is encrypted using the `WITH ENCRYPTION` option. The definition for this trigger cannot be viewed as the definition is encrypted.

```
ALTER TRIGGER Create_Permission
ON DATABASE
WITH ENCRYPTION
FOR CREATE_TABLE
AS
     BEGIN
      PRINT 'You do not have the permission to
create tables'
      ROLLBACK
END
```

Snippet

Syntax

Snippet

Dropping DDL Triggers

SQL Server 2005 allows you to delete a DDL trigger if it is no longer required. Like a DML trigger, a DDL trigger is dropped using the `DROP TRIGGER` statement. You can drop multiple DDL triggers using the `DROP TRIGGER` statement.

When a DDL trigger is dropped, it is also removed from the current database. However, the objects or the data related to the deleted DDL trigger remain unaffected.

Syntax

Snippet

Dropping DDL Triggers

SQL Server 2005 allows you to delete a DDL trigger if it is no longer required. Like a DML trigger, a DDL trigger is dropped using the `DROP TRIGGER` statement. You can drop multiple DDL triggers using the `DROP TRIGGER` statement.

The following is the syntax for dropping a DDL trigger.

```
DROP TRIGGER <DDL_trigger_name> ON {DATABASE|ALL SERVER}
```

where,
`DDL_trigger_name`: Specifies the name of the DDL trigger to be deleted.

Syntax

Syntax | Snippet

Dropping DDL Triggers

SQL Server 2005 allows you to delete a DDL trigger if it is no longer required. Like a DML trigger, a DDL trigger is dropped using the `DROP TRIGGER` statement. You can drop multiple DDL triggers using the `DROP TRIGGER` statement.

The following code drops the DDL trigger `Create_Permission` created at the database level.

```
DROP TRIGGER Create_Permission ON DATABASE
```

Snippet

Syntax

Snippet

Knowledge Check

Which of these statements about DDL triggers in SQL Server 2005 are true and which statements are false?

Select an option for each statement and then click on **Submit**.

	Statements	True	False
(A)	A DDL trigger for `DROP` operation can be created as an `INSTEAD OF` trigger.	○	○
(B)	A DDL trigger for an `ALTER` event can be invoked only after the modification operations are complete.	○	○
(C)	A DDL trigger definition can be displayed using `sp_helptext` stored procedure.	○	○
(D)	A DDL trigger can be removed from the current database using the `DROP TRIGGER` statement.	○	○
(E)	Multiple DDL triggers can be dropped using a single `DROP TRIGGER` statement.	○	○

Submit

47 | 16

Score

Knowledge Check

Which of these statements about DDL triggers in SQL Server 2005 are true and which statements are false?

Select an option for each statement and then click on **Submit**.

	Statements	True	False
(A)	A DDL trigger for `DROP` operation can be created as an `INSTEAD OF` trigger.	○	◉
(B)	A DDL trigger for an `ALTER` event can be invoked only after the modification operations are complete.	◉	○
(C)	A DDL trigger definition can be displayed using `sp_helptext` stored procedure.	○	◉
(D)	A DDL trigger can be removed from the current database using the `DROP TRIGGER` statement.	◉	○
(E)	Multiple DDL triggers can be dropped using a single `DROP TRIGGER` statement.	◉	○

Correct

The correct answers are displayed.

Submit

47 | 16

Score

Knowledge Check

Can you match the types of DDL triggers in SQL Server 2005 against their corresponding descriptions?

Select an option from each drop-down list box and then click on **Submit**.

	Description	DDL Triggers
(A)	Invoked by the events that modify the database schema.	Select Step...
(B)	Invoked by DDL events at the server.	Select Step...
(C)	Stored in the master database.	Select Step...
(D)	Stored in the database.	Select Step...
(E)	Defined in the database-level or at the server-level.	Select Step...

Submit

View Answer

48 | 16

Score

Knowledge Check

Can you match the types of DDL triggers in SQL Server 2005 against their corresponding descriptions?

Select an option from each drop-down list box and then click on **Submit**.

	Description	DDL Triggers
(A)	Invoked by the events that modify the database schema.	Database-scoped
(B)	Invoked by DDL events at the server.	Server-scoped
(C)	Stored in the master database.	Server-scoped
(D)	Stored in the database.	Database-scoped
(E)	Defined in the database-level or at the server-level.	DDL triggers

Correct

The correct answers are displayed.

Submit

View Answer

Lesson Review

In this fourth lesson, **Working with DDL Triggers**, you learnt to:

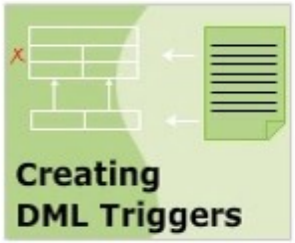
- Explain DDL trigger for `CREATE`.
- Explain DDL trigger for `ALTER`.
- Explain DDL trigger for `DROP`.
- Describe how to view DDL triggers.
- Describe how to modify DDL triggers.
- Describe how to drop DDL triggers.

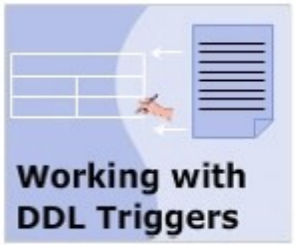
Module Summary

In this module, **Introduction to Triggers**, you learnt about:

- Introduction to Triggers
- Creating DML Triggers
- Working with DML Triggers
- Working with DDL Triggers

Click on each link for a summary of the lesson.

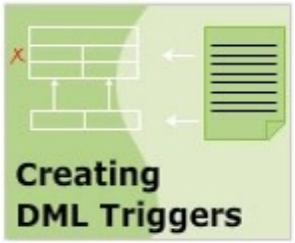


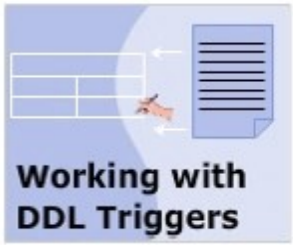


Module Summary

In this module, **Introduction to Triggers**, you learnt about:

- Introduction to Triggers
- Creating DML Triggers
- Working with DML Triggers
- Working with DDL Triggers





Click on each link for a summary of the lesson.

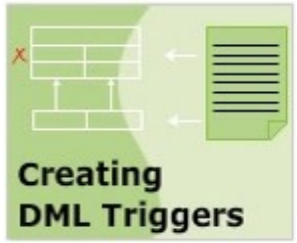
Introduction to Triggers

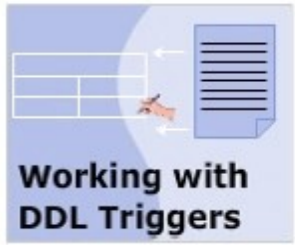
A trigger is an event that sets off an action. The `INSERT`, `UPDATE` and `DELETE` as well as the `CREATE`, `ALTER` and `DROP` statements in SQL Server 2005 invoke triggers. Triggers are used to ensure referential integrity, enforce business rules, enforce complex restrictions, display customized error messages and retrieve information from tables from the same as well as different databases. SQL Server 2005 supports three types of triggers; Data Manipulation Language (DML) triggers, Data Definition Language (DDL) triggers and Logon triggers.

Module Summary

In this module, **Introduction to Triggers**, you learnt about:

- Introduction to Triggers
- Creating DML Triggers
- Working with DML Triggers
- Working with DDL Triggers





Click on each link for a summary of the lesson.

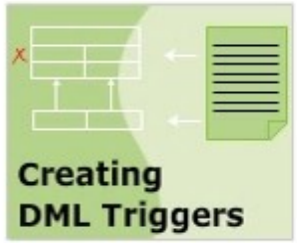
Creating DML Triggers

DML triggers execute when data is inserted, modified or deleted in a table or a view using the `INSERT`, `UPDATE` or `DELETE` statements. An `INSERT` trigger is executed when a new record is inserted in a table. An UPDATE trigger is executed when a record in the table is updated. A `DELETE` trigger is executed when a record is deleted from the table. DML triggers can be created to execute on completion or in place of the modification operations. These are referred as `AFTER` and `INSTEAD OF` triggers respectively.

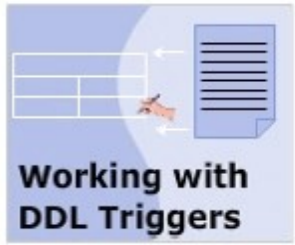
Module Summary

In this module, **Introduction to Triggers**, you learnt about:

- Introduction to Triggers
- Creating DML Triggers
- Working with DML Triggers
- Working with DDL Triggers



Click on each link for a summary of the lesson.



Working with DML Triggers

SQL Server 2005 allows you to create multiple `AFTER` triggers for each triggering action (such as `UPDATE`, `INSERT` and `DELETE`) on a table. However, you can create only one `INSTEAD OF` trigger for each triggering action on a table. If multiple `AFTER` triggers are created for a particular triggering action, you must specify only the first and the last `AFTER` triggers to be executed. SQL Server 2005 allows you to view, modify and drop the DML triggers.