SQL Server DDL Trigger

Summary: in this tutorial, you will learn how to use the SQL Server data definition language (DDL) trigger to monitor the changes made to the database objects.

Introduction to SQL Server DDL triggers

SQL Server DDL triggers respond to server or database events rather than to table data modifications. These events created by the Transact-SQL statement that normally starts with one of the following keywords `CREATE`, `ALTER`, `DROP`, `GRANT`, `DENY`, `REVOKE`, or `UPDATE STATISTICS`.

For example, you can write a DDL trigger to log whenever a user issues a `CREATE TABLE` (https://www.sqlservertutorial.net/sql-server-basics/sql-server-create-table/) or `ALTER TABLE` statement.

The DDL triggers are useful in the following cases:

Record changes in the database schema.

Prevent some specific changes to the database schema.

Respond to a change in the database schema.

The following shows the syntax of creating a DDL trigger:

```
CREATE TRIGGER trigger_name
ON { DATABASE |  ALL SERVER}
[WITH ddl_trigger_option]
FOR {event_type | event_group }
AS {sql_statement}
```

`trigger_name`

Specify the user-defined name of trigger after the `CREATE TRIGGER` keywords. Note that you don't have to specify a schema for a DDL trigger because it isn't related to an actual database table or view.

`DATABASE | ALL SERVER`

Use `DATABASE` if the trigger respond to database-scoped events or `ALL SERVER` if the trigger responds to the server-scoped events.

`ddl_trigger_option`

The `ddl_trigger_option` specifies `ENCRYPTION` and/or `EXECUTE AS` clause. `ENCRYPTION` encrypts the definition of the trigger. `EXECUTE AS` defines the security context under which the trigger is executed.

`event_type | event_group`

The `event_type` indicates a DDL event that causes the trigger to fire e.g., `CREATE_TABLE` , `ALTER_TABLE` , etc.

The `event_group` is a group of `event_type` event such as `DDL_TABLE_EVENTS`.

A trigger can subscribe to one or more events or groups of events.

Creating a SQL Server DDL trigger example

Suppose you want to capture all the modifications made to the database index so that you can better monitor the performance of the database server which relates to these index changes.

First, create a new table (https://www.sqlservertutorial.net/sql-server-basics/sql-server-create-table/) named `index_logs` to log the index changes:

```
CREATE TABLE index_logs (
    log_id INT IDENTITY PRIMARY KEY,
    event_data XML NOT NULL,
    changed_by SYSNAME NOT NULL
);
GO
```

Next, create a DDL trigger to track index changes and insert events data into the `index_logs` table:

```
CREATE TRIGGER trg_index_changes
ON DATABASE
FOR
```

```
    CREATE_INDEX,
    ALTER_INDEX,
    DROP_INDEX
AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO index_logs (
        event_data,
        changed_by
    )
    VALUES (
        EVENTDATA(),
        USER
    );
END;
GO
```

In the body of the trigger, we used the `EVENTDATA()` function that returns the information about server or database events. The function is only available inside DDL or logon trigger.

Then, create indexes for the `first_name` and `last_name` columns of the `sales.customers` table:

```
CREATE NONCLUSTERED INDEX nidx_fname
ON sales.customers(first_name);
```

```
GO

CREATE NONCLUSTERED INDEX nidx_lname
ON sales.customers(last_name);
GO
```

After that, query data from the `index_changes` table to check whether the index creation event was captured by the trigger properly:

```
SELECT
    *
FROM
    index_logs;
```

Here is the output:

If you click on the cell of the `event_data` column, you can view XML data of the event as follows:

In this tutorial, you have learned how to create a SQL Server DDL trigger that responds to one or more DDL events.