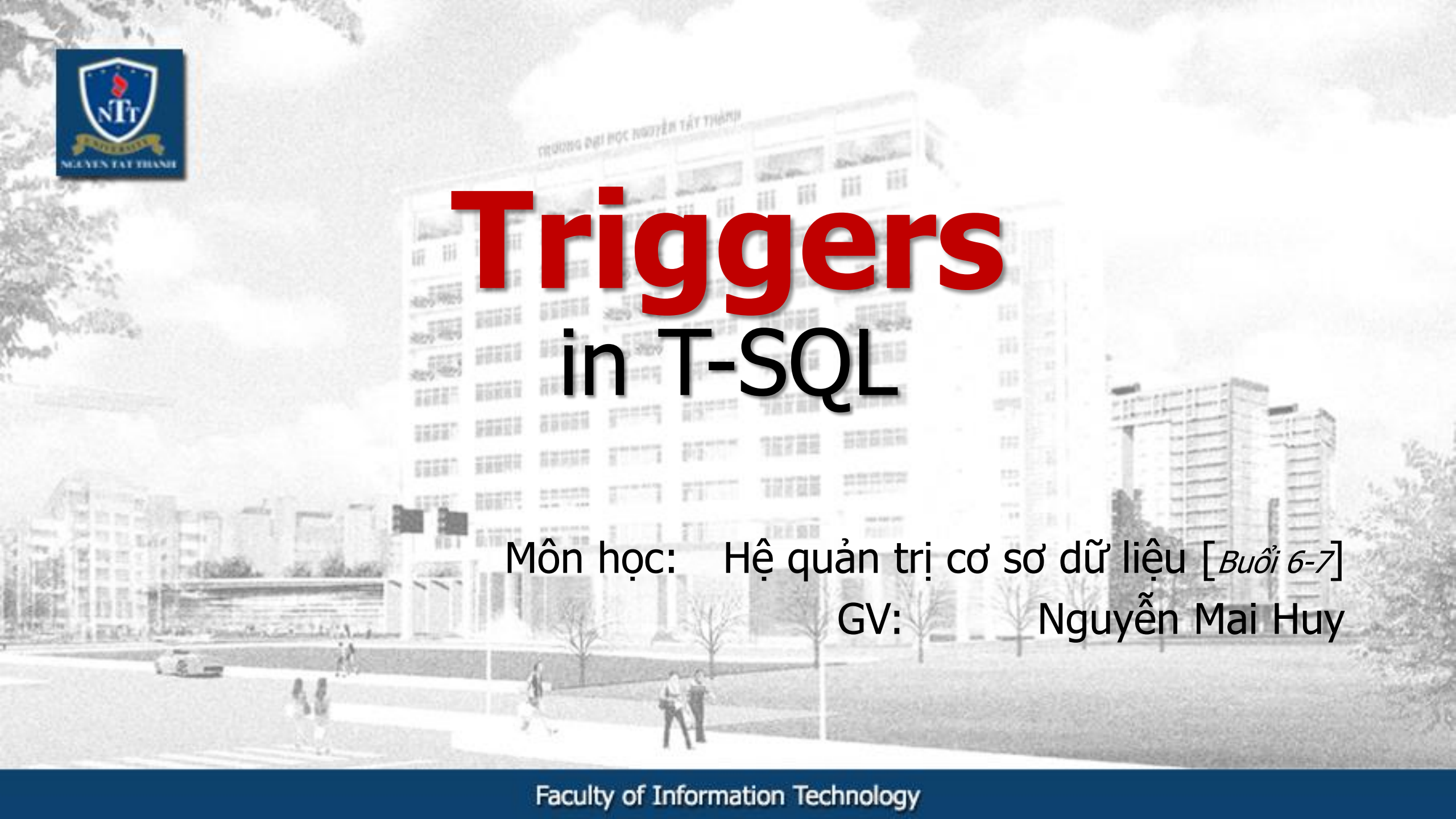


Triggers
in T-SQL

Môn học: Hệ quản trị cơ sơ dữ liệu [*Buổi 6-7*]

GV: Nguyễn Mai Huy

Triggers
on database

About **triggers**

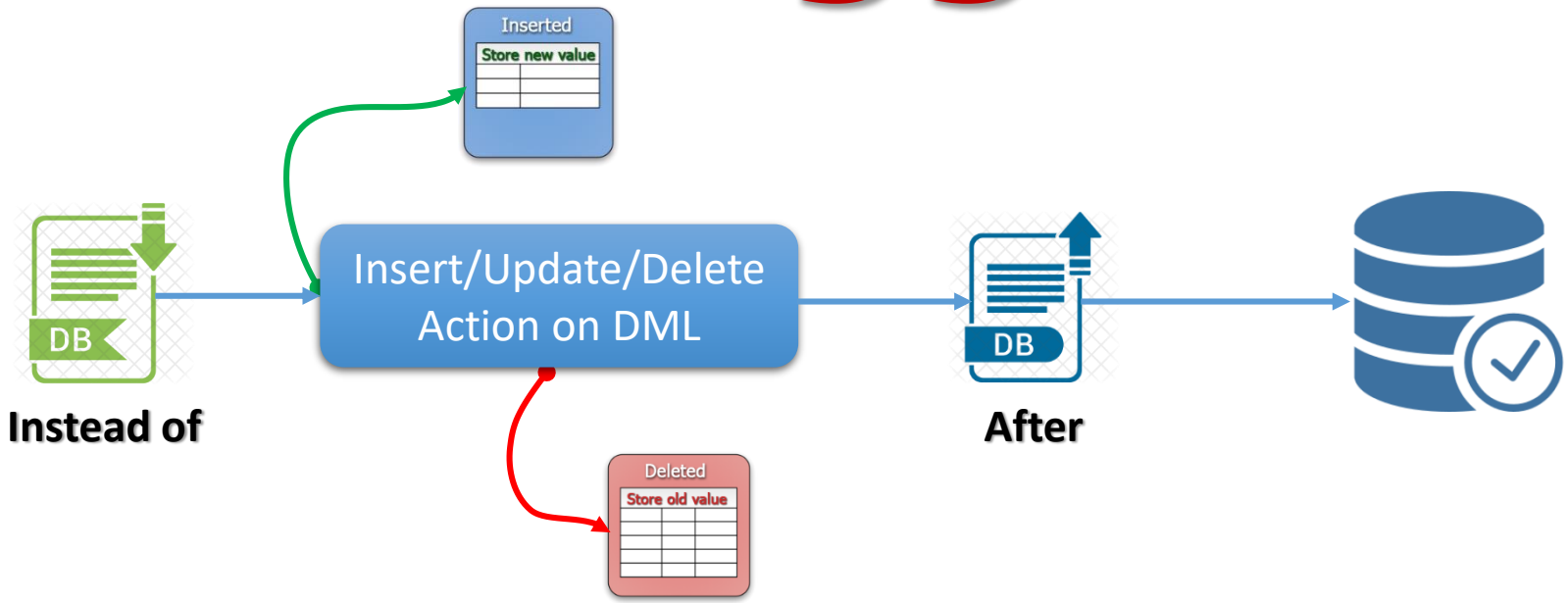
Trong SQL Server, **triggers** là đối tượng thuộc loại Database object, Những đối tượng này tồn tại dưới dạng Stored Procedure đặc biệt với mục tiêu là "***phản ứng lại***" các hành động mà người dùng tác động trên Database của SQL Server. Thông thường, các mã lệnh được đặt trong **Triggers sẽ tự động được kích hoạt** khi có *các sự kiện xảy ra* trên Database để thực thi một số các hành động nào đó. VD: Kiểm tra dữ liệu, Ngăn cản các hành vi làm tổn hại tới Db hoặc đơn giản chỉ là ghi nhật ký đối với các hành động của người dùng.

- Khi bạn cập nhật thông tin về đơn hàng đã được giao thành công cho khách, **trigger** *cập nhật số lượng tồn kho* sẽ tự động được gọi thi hành.
- Nếu thực hiện xóa một đơn hàng đã được kích hoạt (*hay thời gian đặt hàng được hơn 24 giờ*) thì **Trigger** phục vụ cho mục tiêu tương ứng sẽ được kích hoạt để ngăn cản hành động này
- Nếu *trạng thái của một bài viết* trong CSDL của một tờ báo được kích hoạt không phải do nhóm tài khoản có quyền biên tập thì dữ liệu này sẽ *tự động được ghi nhật ký* phục vụ cho việc điều tra (*nếu có*)

Types of SQL **Triggers**

- **DML triggers** – Đây là các **Triggers** được tạo ra để phản ứng lại với các lệnh được thực hiện đối với các thao tác trên dữ liệu lưu trữ trong Database: ***Insert***, ***Update***, ***Delete***
- **DDL triggers** – Trigger thuộc loại này sẽ tự động được kích hoạt khi có hành vi tác động lên đối tượng thuộc loại Database object: ***Create***, ***Alter***, ***Drop***
- **Logon triggers** – Thường dùng cho mục tiêu ghi nhận các kết nối vào database và được chứng thực bởi quyền làm việc thông qua Login name của người truy cập

DML Triggers



DML Triggers

Như đã giới thiệu trong phần trước, SQL Server **DML Trigger** là loại thủ tục đặc biệt được lưu trữ trên CSDL để "***thực hiện các phản ứng***" đối với các hành động thuộc nhóm DML: ***Insert***, ***Update*** hoặc ***Delete*** nhằm làm thay đổi dữ liệu đã lưu trữ trong CSDL, *ngay cả khi các dòng (Record) trong Table/View có bị thay đổi nội dung hay không*.

Triggers thuộc loại này sẽ *tự động thi hành các lệnh theo thuật toán đã định khi có sự thay đổi đối với dữ liệu*. DML triggers *thường được dùng để đảm bảo tính rang buộc toàn vẹn đối với dữ liệu đồng thời thi hành các lô-gic mang tính nghiệp vụ của tổ chức* dựa trên việc kiểm tra hay thiết lập giá trị phù hợp trong table.

Types of DML Triggers

- **AFTER** or **FOR** trigger : sẽ được kích hoạt khi các lệnh: **INSERT**, **UPDATE** hoặc **DELETE** đã thực hiện thành công trên dữ liệu của bảng. Ngoài ra, tất cả các "phản ứng dây chuyền" và các kiểm tra ràng buộc cũng sẽ hoàn tất trước khi Trigger loại này được kích hoạt.
  Lưu ý: **AFTER** trigger chỉ có thể định nghĩa cho Table, mà không thể tạo cho View.
- **INSTEAD OF** trigger: Khác với ***After trigger***, **Instead Of trigger** sẽ được kích hoạt cùng lúc với thời điểm xảy ra các tác động đối với dữ liệu của Database.

Syntax :: DML Triggers

CREATE TRIGGER <trigger_name> **ON** <table_name>

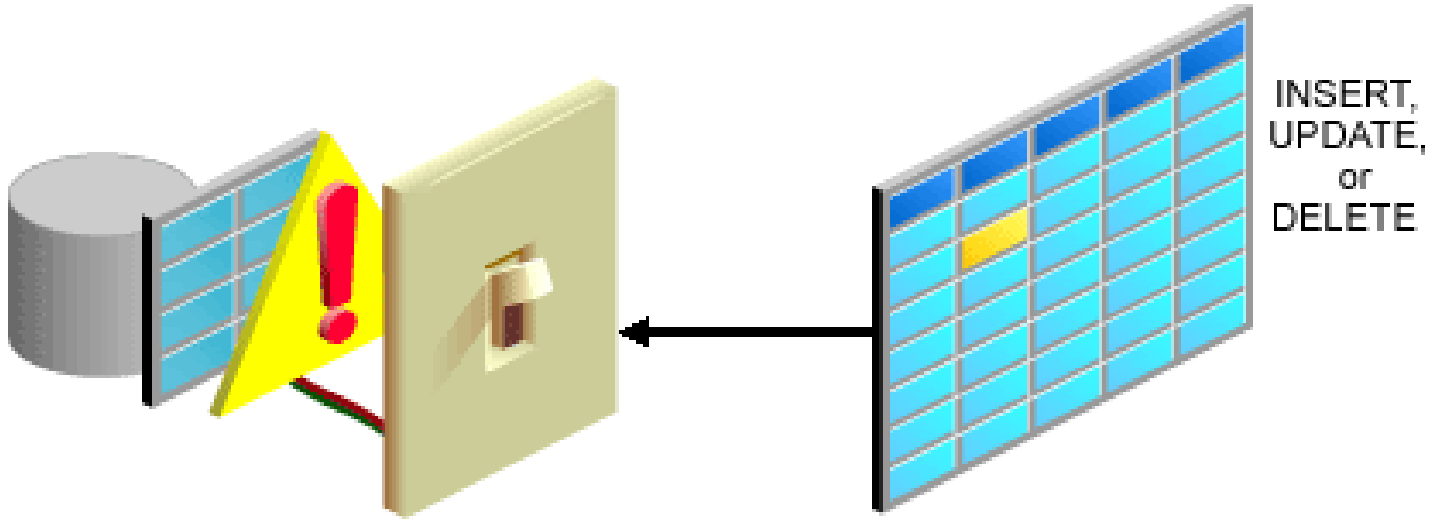
[**WITH Encryption**]

{**FOR** | **AFTER** | **INSTEAD OF**}

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

AS

{sql_statements}



Virtual tables for triggers

SQL Server provides *two virtual tables* that are available specifically for triggers called **INSERTED** and **DELETED** tables. SQL Server uses these tables to capture the data of the modified row before and after the event occurs.

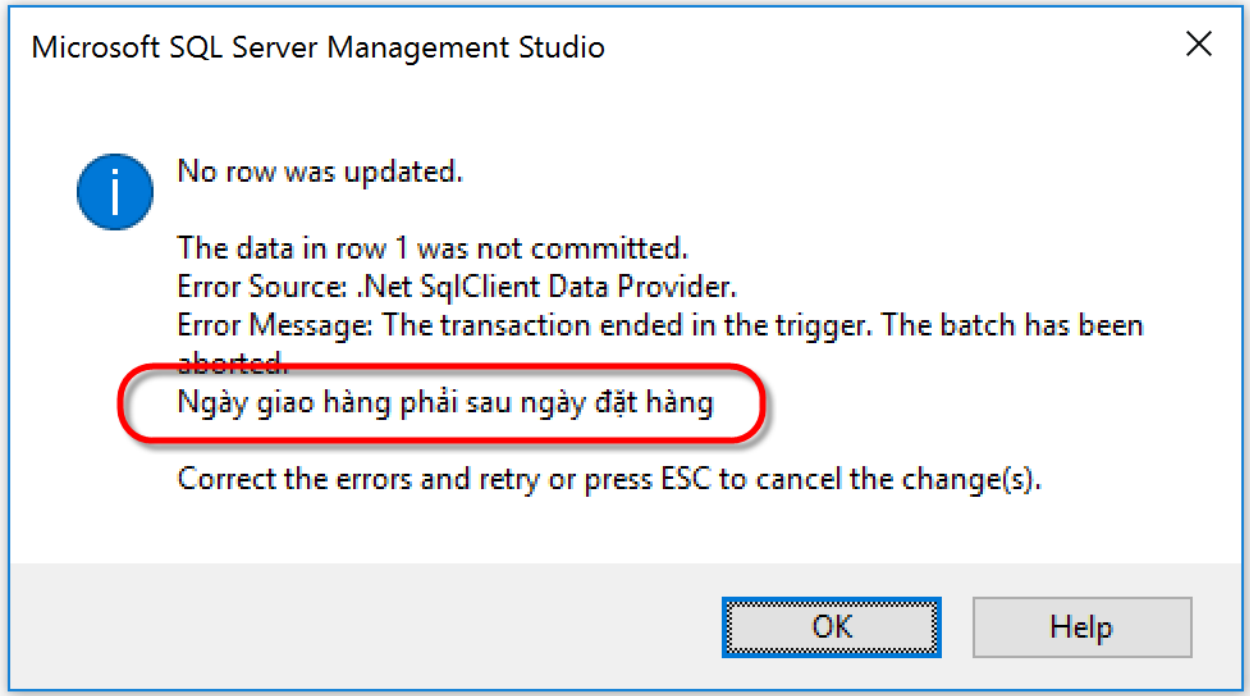
DML event	INSERTED table holds	DELETED table holds
INSERT	rows to be inserted	empty
UPDATE	new rows modified by the update	existing rows modified by the update
DELETE	empty	rows to be deleted

Example :: FOR, AFTER trigger

```
use BanHangOnline
go
-- Create MDL trigger in FOR type
Create Trigger checkOrder_delivery on donHang
With ENCRYPTION                          -- Using this keyword to secure the trigger content
FOR insert, update
as
    declare @ngayGH datetime             -- Declare a variable to store the order date
    declare @ngayDat datetime            -- Declare a variable to store the delivery date
    -- Get data from Virtual
    select @ngayDat=ngayDat, @ngayGH=ngayGH from inserted
    -- Check the delivery date whether it is valid or not
    if @ngayGH<= @ngayDat
    begin
        raiserror (N'Ngày giao hàng phải sau ngày đặt hàng', 10, 1)
        rollback;
    end
```

Example :: Check checkOrder_delivery

	soDH	maKH	taiKhoan	ngayDat	daKichHoat	ngayGH	diaChiGH	qhiChu
✎	DH001	KH001	minh	2020-07-31 00:0	False	2020-07-30 00:0	12 Trần Quốc T	*NULL*
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL



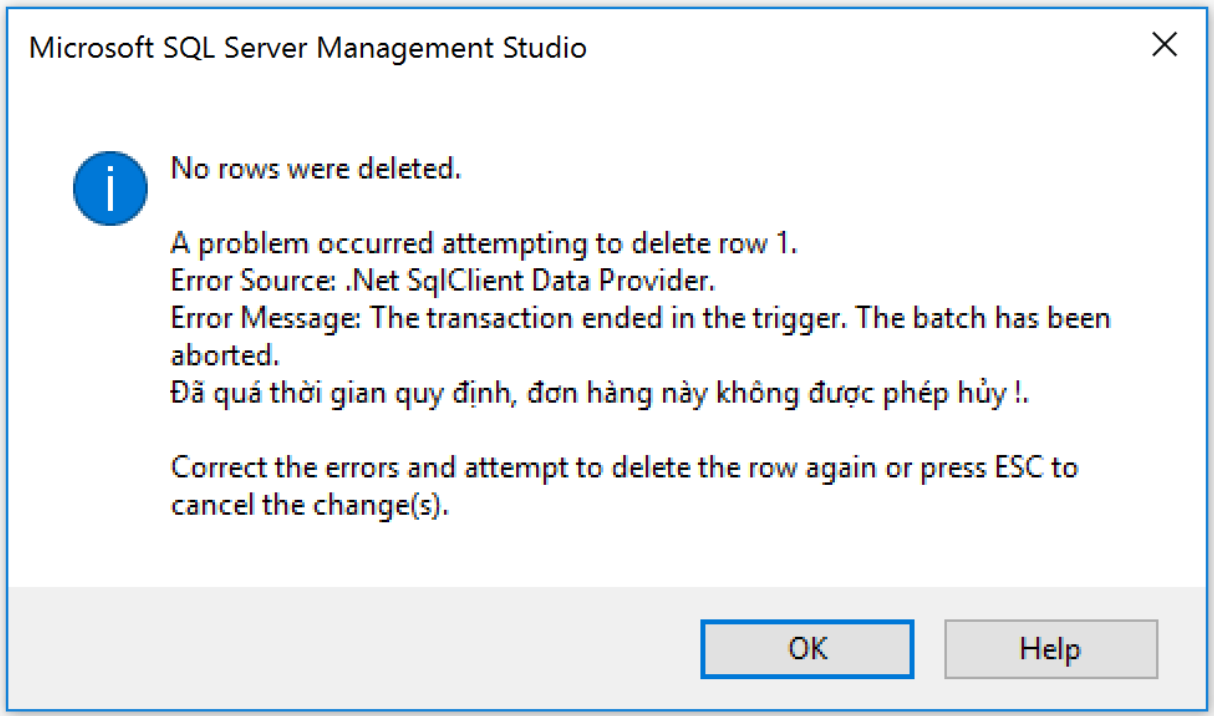
Example :: **Instead Of** trigger

```
-- Create DML Trigger INSTEAD OF
Create Trigger checkCancel_Order on donHang
Instead of delete
as
    declare @ngayGH datetime            -- Declare a variable to store the order date
    declare @ngayDat datetime           -- Declare a variable to store the delivery date
    -- Get data from Virtual
    select @ngayDat=ngayDat, @ngayGH=ngayGH from deleted
    -- Check the cancel time whether over 24 hours or not
    if DATEDIFF(hour, @ngayDat, CURRENT_TIMESTAMP)>=24
    Begin
        raiserror (N'Đã quá thời gian quy định, đơn hàng này không được phép hủy !.', 10, 1)
        rollback;
    End
go
```

Example :: Check checkCancel_Order

BODUAGROUP\BOD...e - dbo.donHang | SQLQuery1.sql - BO...angOnline (sa (57))*

	soDH	maKH	taiKhoan	ngayDat	daKichHoat	ngayGH	diaChiGH	qhiChu
▶	DH001	KH001	minh	2020-07-30 00:0...	True	2020-08-01 00:0...	123 Trần Quốc Toản, Q.3	Giao hàng - Nhận tiền
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL



Function :: DateDiff

DATEDIFF(*interval*, *date1*, *date2*)

Parameter	Description
interval	Required. The part to return. Can be one of the following values: • year, yyyy, yy = Year • quarter, qq, q = Quarter • month, mm, m = month • dayofyear = Day of the year • day, dy, y = Day • week, ww, wk = Week • weekday, dw, w = Weekday • hour, hh = hour • minute, mi, n = Minute • second, ss, s = Second • millisecond, ms = Millisecond
date1, date2	Required. The two dates to calculate the difference between

Syntax :: DML Triggers

ALTER TRIGGER <trigger_name> **ON** <table_name>

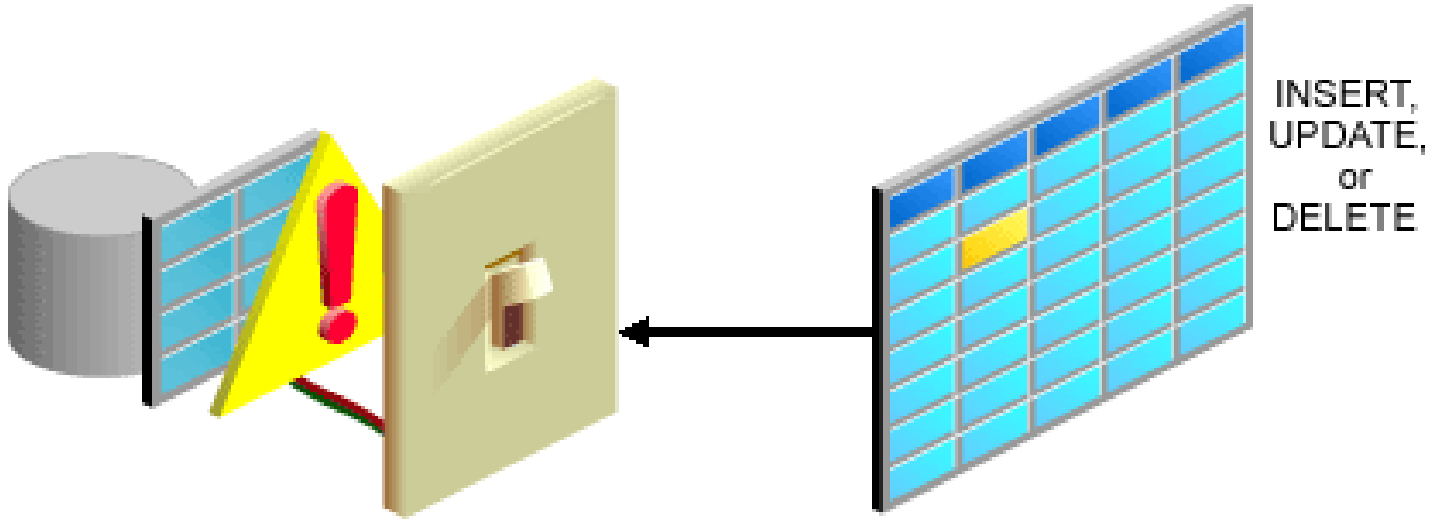
[**WITH Encryption**]

{**FOR** | **AFTER** | **INSTEAD OF**}

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

AS

{sql_statements}

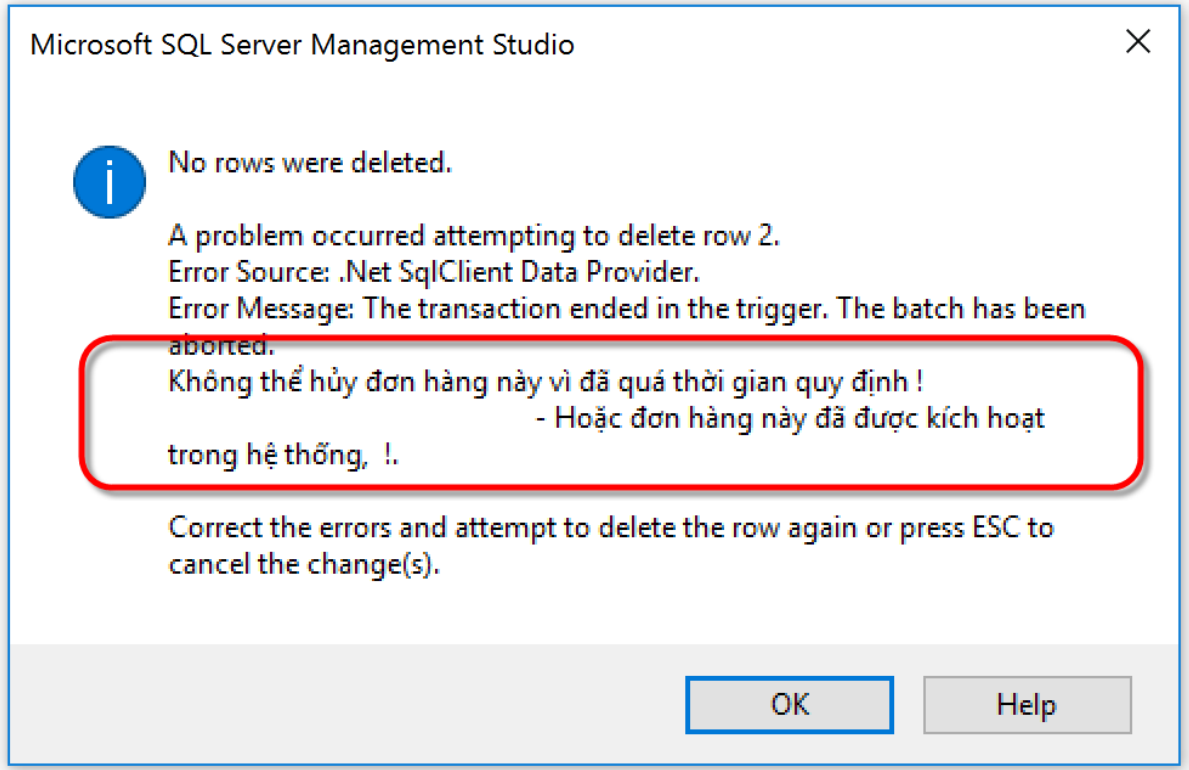


Alter checkCancel_Order trigger

```
-- Create DML Trigger INSTEAD OF
Alter Trigger checkCancel_Order on donHang
With ENCRYPTION                          -- Using this keyword to secure the trigger content
Instead of delete
as
    declare @ngayGH datetime             -- Declare a variable to store the order date
    declare @ngayDat datetime            -- Declare a variable to store the delivery date
    declare @daKichHoat bit              -- Declare a variable to store daKichHoat value
    -- Get data from Virtual
    select @ngayDat=ngayDat, @ngayGH=ngayGH, @daKichHoat=daKichHoat from deleted
    -- Check the cancel time whether over 24 hours or not
    if (DATEDIFF(hour, @ngayDat, CURRENT_TIMESTAMP)>=24) or @daKichHoat=1
    Begin
        raiserror (N'Không thể hủy đơn hàng này vì đã quá thời gian quy định !
                    - Hoặc đơn hàng này đã được kích hoạt trong hệ thống,  !.', 10, 1)
        rollback;
    End
go
```

Delete the Activated order

	soDH	maKH	taiKhoan	ngayDat	daKichHoat	ngayGH	diaChiGH	qhiChu
	DH001	KH001	minh	2020-07-30 00:0...	True	2020-08-01 00:0...	123 Trần Quốc Toản, Q.3	Giao hàng - Nhận tiền
▶	DH002	KH001	minh	2020-08-01 22:1...	True	2020-08-03 09:4...	112 Phạm Ngũ Lão, Q.1	Đã thanh toán
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL



Drop Trigger

- Syntax:

  Drop Trigger *<Trigger_Name>*;

- Example:

  Drop Trigger *checkCancel_Order*;

  Drop Trigger *checkOrder_delivery*;

RAISERROR :: Function

RAISERROR (*message_text* , *Severity* , *State*)

- **Severity**: 0 and 25
  - 0–10 Informational messages
  - 11–18 Errors
  - 19–25 Fatal errors
- **State**: is an integer from 0 through 255

DDL Triggers

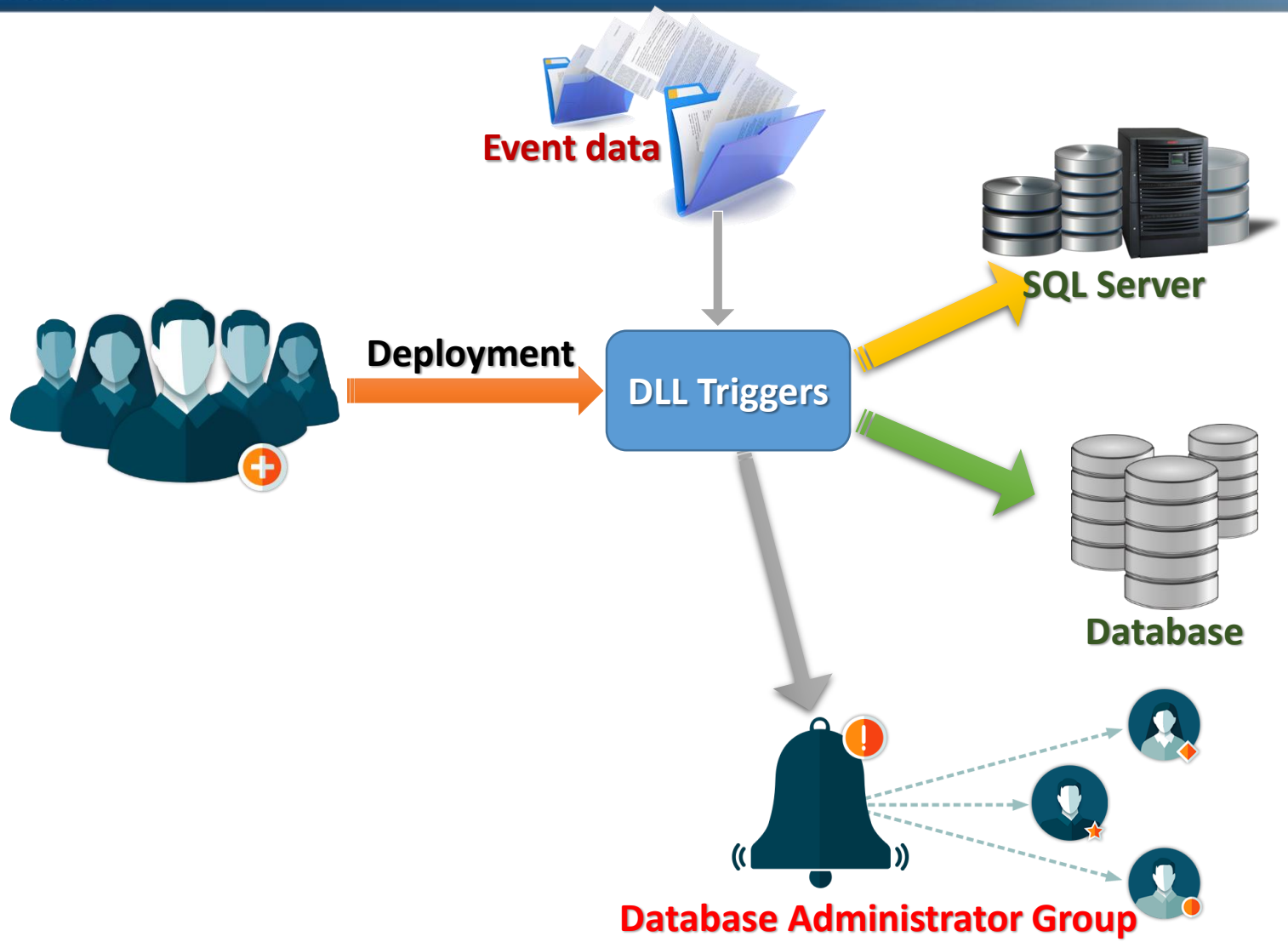
DDL Triggers

SQL Server DDL triggers phản ứng lại với các tác động của người dùng ở mức độ Server hay Database thay vì chỉ đơn thuần là sửa đổi dữ liệu trên bảng. Những sự kiện được thực hiện bởi các lệnh Transact-SQL với các từ khóa như: **Create**, **Alter**, **Drop**, **Grant**, **Deny**, **Revoke** hoặc **Update Statistics**.

Các **DDL triggers** hữu dụng trong một số tình huống sau:

- Ghi nhận những thay đổi của Database schema.
- Ngăn chặn một số tình huống nguy hiểm khi có sự thay đổi đối với Database schema.
- Ghi nhật ký tương tác với Database phục vụ cho bảo trì hoặc thiết lập các chính sách bảo mật cần thiết
- . . .

How **DDL Trigger** works?



Mỗi tương tác thuộc dạng DDL được xem là một giao dịch (*Transaction*) đối với CSDL hoặc máy chủ CSDL. Máy chủ CSDL (*Database Server*) sẽ tự động phát sinh một sự kiện có liên quan tương ứng với giao dịch DDL, đồng thời tạo ra các dữ liệu chi tiết, mô tả cho sự kiện tương ứng ở dạng Metadata dựa theo định dạng XML vốn rất phổ biến hiện nay

Syntax

CREATE TRIGGER <trigger_name>

ON { **DATABASE** | **ALL SERVER**}

[**WITH ENCRYPTION**]

FOR {event_type | event_group }

AS {sql_statement}

- ✓ **event_type** indicates a DDL event that causes the trigger to fire e.g., **CREATE_TABLE**, **ALTER_TABLE**, etc.
- ✓ **event_group** is a group of event_type event such as **DDL_TABLE_EVENTS**

Example :: Create table to store logs

```
Create Table working_logs (
        log_id      Int         Identity    Primary Key,
        metadata    Xml         Not Null,
        changed_by  Sysname     Not Null
);
```

*** Note:**

Using Special Data Types

https://docs.microsoft.com/en-us/previous-versions/sql/sql-server-2008-r2/ms191240(v=sql.105)?redirectedfrom=MSDN

Example :: DDL trigger

```
Create Trigger workingLog_Trigger ON DATABASE
For   CREATE_INDEX, ALTER_INDEX, DROP_INDEX,
      CREATE_VIEW, DROP_VIEW, ALTER_VIEW,
      CREATE_PROCEDURE, DROP_PROCEDURE,
      ALTER_PROCEDURE, CREATE_FUNCTION,
      DROP_FUNCTION, ALTER_FUNCTION
AS
Begin
      Insert Into working_logs(metadata, changed_by )
      Values ( EVENTDATA(), USER );
End;
```

Example :: Test for workingLog_Trigger

```
-- Testing for workingLog_Trigger process on :: Create View - Create Index - Drop View
-- 1.1 :: Create View ------------------------------------------------------------
Create View donHang_TaiKhoan
With Encryption
As
    select tk.taiKhoan, COUNT(dh.soDH) as N'Số ĐH'
    from taiKhoanTV tk inner Join donHang dh on dh.taiKhoan = tk.taiKhoan
    group by tk.taiKhoan
go
-- 1.2 :: Create Nonclustered Index
Create Nonclustered Index tenSanPham On sanPham(tenSP);
go
-- 1.3 :: Drop view
Drop View tk_SlSanPham;
go
```

Example :: Table working_logs

- **Log_id**:
- **Metadata**:
- **Changed_by**:

Results | Messages

	log_id	metadata	changed_by
1	1	<EVENT_INSTANCE><EventType>CREATE_VIEW</EventType><PostTime>2020-08-02T09:28:40.600</PostTime><SPID>53</SPID><S...	dbo
2	2	<EVENT_INSTANCE><EventType>CREATE_INDEX</EventType><PostTime>2020-08-02T09:29:35.327</PostTime><SPID>53</SPID><...	dbo
3	3	<EVENT_INSTANCE><EventType>DROP_VIEW</EventType><PostTime>2020-08-02T09:29:53.113</PostTime><SPID>53</SPID><Serv...	dbo

Metadata for **Create View** with **Encryption**

metadata3.xml | metadata2.xml | metadata1.xml* | SQLQuery3.sql - BO...angOnline (sa (54)) | SQLQuery1.sql - BO...angOnline (sa (53))*

```
<EVENT_INSTANCE>
  <EventType>CREATE_VIEW</EventType>
  <PostTime>2020-08-02T09:28:40.600</PostTime>
  <SPID>53</SPID>
  <ServerName>BODUAGROUP\BODUA</ServerName>
  <LoginName>sa</LoginName>
  <UserName>dbo</UserName>
  <DatabaseName>BanHangOnline</DatabaseName>
  <SchemaName>dbo</SchemaName>
  <ObjectName>donHang_TaiKhoan</ObjectName>
  <ObjectType>VIEW</ObjectType>
  <TSQLCommand>
    <SetOptions ANSI_NULLS="ON" ANSI_NULL_DEFAULT="ON" ANSI_PADDING="ON" QUOTED_IDENTIFIER="ON" ENCRYPTED="TRUE" />
    <CommandText>--ENCRYPTED--</CommandText>
  </TSQLCommand>
</EVENT_INSTANCE>
```

Metadata for **Create Nonclustered Index**

metadata3.xml | metadata2.xml* | metadata1.xml* | SQLQuery3.sql - BO...angOnline (sa (54)) | SQLQuery1.sql - BO...angOnline (sa (53))*

```
<EVENT_INSTANCE>
  <EventType>CREATE_INDEX</EventType>
  <PostTime>2020-08-02T09:29:35.327</PostTime>
  <SPID>53</SPID>
  <ServerName>BODUAGROUP\BODUA</ServerName>
  <LoginName>sa</LoginName>
  <UserName>dbo</UserName>
  <DatabaseName>BanHangOnline</DatabaseName>
  <SchemaName>dbo</SchemaName>
  <ObjectName>tenSanPham</ObjectName>
  <ObjectType>INDEX</ObjectType>
  <TargetObjectName>sanPham</TargetObjectName>
  <TargetObjectType>TABLE</TargetObjectType>
  <TSQLCommand>
    <SetOptions ANSI_NULLS="ON" ANSI_NULL_DEFAULT="ON" ANSI_PADDING="ON" QUOTED_IDENTIFIER="ON" ENCRYPTED="FALSE" />
    <CommandText>Create Nonclustered Index tenSanPham On sanPham(tenSP)</CommandText>
  </TSQLCommand>
</EVENT_INSTANCE>
```

Metadata for **Drop View**

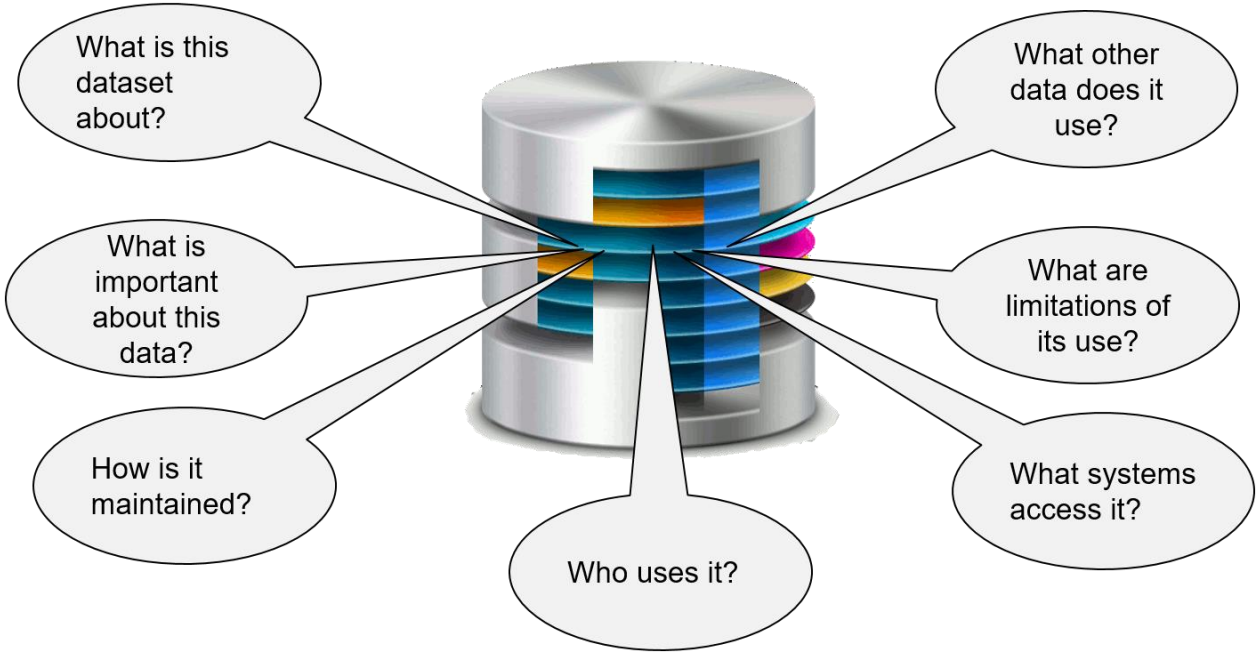
metadata3.xml | metadata2.xml* | metadata1.xml* | SQLQuery3.sql - BO...angOnline (sa (54)) | SQLQuery1.sql - BO...angOnline (sa (53))*

```
<EVENT_INSTANCE>
  <EventType>DROP_VIEW</EventType>
  <PostTime>2020-08-02T09:29:53.113</PostTime>
  <SPID>53</SPID>
  <ServerName>BODUAGROUP\BODUA</ServerName>
  <LoginName>sa</LoginName>
  <UserName>dbo</UserName>
  <DatabaseName>BanHangOnline</DatabaseName>
  <SchemaName>dbo</SchemaName>
  <ObjectName>tk_SlSanPham</ObjectName>
  <ObjectType>VIEW</ObjectType>
  <TSQLCommand>
    <SetOptions ANSI_NULLS="ON" ANSI_NULL_DEFAULT="ON" ANSI_PADDING="ON" QUOTED_IDENTIFIER="ON" ENCRYPTED="FALSE" />
    <CommandText>Drop View tk_SlSanPham;
</CommandText>
  </TSQLCommand>
</EVENT_INSTANCE>
```

Event data function

The **EVENTDATA()** is an inbuilt function of the DDL trigger in SQL Server and that would return exchange occasion subtleties with the number of the fields in XML format

EVENTDATA() returns multiple fields in XML format as shown above and using those fields, we are able to create such metrics to track various events of DDL over the objects. In general, each DDL event of the object schema changes can be appended into the table, these event types are mentioned in the header body of ä trigger with the **FOR CREATE**, **ALTER**, **DROP**,…



XML format

- **EventType** (*Create View, Alter View, Drop View, etc…*)
- **PostTime** (*Event trigger time*)
- **SPID** (*SQL Server session ID*)
- **ServerName** (*SQL Server instance name*)
- **LoginName** (*SQL Server Login name*)
- **UserName** (*username for login, by default dbo schema as username*)
- **DatabaseName** (*name of database where trigger was executed*)
- **SchemaName** (*schema name of the View*)
- **ObjectName** (*Name of the View*)
- **ObjectType** (*Object types. such as Table, view, procedure, etc…*)
- **TSQLCommand** (*Schema deployment Query which is executed by user*)
- **SetOptions** (*SET Option which are applied while Creating View or Modify it*)
- **CommandText** (*Create, Alter or Drop object command*)

Nhớ gì ?!!!

- What is triggers in T-SQL, Why we need to use it on database
- Types of Trigger and Use case for it
- Syntax for DML Triggers
- Distinguish between **AFTER**, **FOR** and **INSTEAD OF** triggers

Tài liệu tham khảo

- Itzik Ben-Gan, "**Microsoft® SQL Server ® 2012 T-SQL Fundamentals**", O'Reilly Media Inc, 2012
- Itzik Ben-Gan, Dejan Sarka, Ed Katibah, Greg Low, Roger Wolter, and Isaac Kunen, "**Inside Microsoft SQL Server 2008: T-SQL Programming**", Microsoft Press, 2010
- w3schools, "**Introduction to SQL**", https://www.w3schools.com/sql/sql_intro.asp, 10:54PM, 18/06/2020
- Microsoft SQL Server Tutorial, "**Tutorials for SQL Server**", https://docs.microsoft.com/en-us/sql/sql-server/tutorials-for-sql-server-2016, 10:54 PM, 18/06/2020