(T16)討論 DataManipulationLanguage(DML)Trigger
CourseGUID: e48417fc-9db5-4e99-822c-706c5ccef6cc
==========
(T16)討論 DataManipulationLanguage(DML)Trigger
==========

==========

0. Summary

1.
Trigger is a special stored procedure
which will be executed automatically when an event occurs.
There are 3 types of trigger.
1.1.
Data Manipulation Language (DML) triggers
1.2.
Data Definition Language (DDL) triggers
1.3.
Logon trigger
2.

There are 2 Types of Data Manipulation Language (DML) triggers
2.1.
After/For Trigger:
After/For Triggers fires after the INSERT/UPDATE/DELETE event happened.
After/For Trigger Syntax:

```
--CREATE TRIGGER {TriggerName} ON {TableName}
--{ After/For Insert | AFTER/For DELETE | AFTER/For UPDATE }
--AS
--    BEGIN
--        ...
--    END
```

2.2.
INSTEAD OF Trigger:
Syntax:

```
--CREATE TRIGGER {TriggerName} ON {TableName}
--{ INSTEAD OF Insert | INSTEAD OF DELETE | INSTEAD OF UPDATE }
--AS
--    BEGIN
--        ...
--    END
```

INSTEAD OF Triggers fires when the Table/View run INSERT/UPDATE/DELETE event,
instead of running the default behaviour, it will run the query in the trigger body.
INSTEAD OF triggers normally correct updating views that are based on multiple tables.
3.
INSERTED table V.S. DELETED table
3.1.
AFTER/FOR INSERT Trigger / INSTEAD OF INSERT Trigger:
INSERTED table structure is same as {TableName} structure,
and it contains the copy version of data rows we inserted.
This table can only be accessed by the Trigger.
3.2.
AFTER/FOR DELETE Trigger / INSTEAD OF DELETE Trigger:
DELETED table structure is same as {TableName} structure,
and it contains the copy version of data rows we deleted.
This table can only be accessed by the Trigger.
3.3.
AFTER/FOR UPDATE Trigger / INSTEAD OF UPDATE Trigger:
The DELETED table contains the copy version of old data.
The INSERTED table contains the copy version of new data.
Both table structures are same as {TableName} structure
and can can only be accessed by the Trigger.
4.
Syntax:

```
--RAISERROR ( { msg_str | @local_variable }
--    { ,severity ,state }
--    [ ,argument [ ,...n ] ] )
--    [ WITH option [ ,...n ] ]
```

E.g.

```
--RAISERROR('ErrorMessage', 16, 1);
```

Reference:
https://docs.microsoft.com/en-us/sql/t-sql/language-elements/raiserror-transact-sql
4.1.
The first parameter, msg_str, is the error message.
4.2.
the second parameter, severity, is the severity level.
Severity level 16 means general errors and can be corrected by the user.
4.3.
The third parameter is state, and we should set default to 1.
RAISERROR only generates errors with state from 1 through 18.
Because the PDW engine may raise errors with state 0,
using a unique state number for different location
can help find which section of code is raising the errors.
5.

```
--IF ( UPDATE(ColumnName) )
```

if someone tries to update or insert {ColumnName}, then ...

5.1.
UPDATE(column)
Reference:
https://docs.microsoft.com/en-us/sql/t-sql/functions/update-trigger-functions-transact-sql
Returns a Boolean value that indicates whether an
INSERT or UPDATE attempt was made on a specified column of a table or view.

==========

1. Create Sample Data

```
--==========================================================
--T016_01_Create Sample Data
--==========================================================
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE      TABLE_NAME = 'Gamer' ) )
    BEGIN
        TRUNCATE TABLE Gamer;
        DROP TABLE Gamer;
    END;
GO -- Run the previous command and begins new batch
CREATE TABLE Gamer
    (
      ID INT PRIMARY KEY ,
      [Name] NVARCHAR(100) ,
      Email NVARCHAR(100) ,
      GenderID INT ,
      GameScore INT
    );
GO -- Run the prvious command and begins new batch
INSERT  INTO Gamer
VALUES  ( 4, 'Name4', '4@4.com', 1, 43000 );
INSERT  INTO Gamer
VALUES  ( 2, 'Name2', '2@2.com', 2, 44000 );
INSERT  INTO Gamer
VALUES  ( 1, 'Name1', '1@1.com', 1, 43000 );
INSERT  INTO Gamer
VALUES  ( 5, 'Name5', '5@5.com', 1, 42000 );
INSERT  INTO Gamer
VALUES  ( 3, 'Name3', '3@3.com', 2, 41000 );
GO -- Run the prvious command and begins new batch
SELECT  *
FROM     Gamer;
GO -- Run the prvious command and begins new batch
```

	ID	Name	Email	GenderID	GameScore
1	1	Name1	1@1.com	1	43000
2	2	Name2	2@2.com	2	44000
3	3	Name3	3@3.com	2	41000
4	4	Name4	4@4.com	1	43000
5	5	Name5	5@5.com	1	42000

==========

2. DMLTriggers

```
--==========================================================
--T016_02_DMLTriggers
--==========================================================
```

2.1. CREATE TABLE GamerAudit

```
--==========================================================
--T016_02_01
--CREATE TABLE GamerAudit
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE      TABLE_NAME = 'GamerAudit' ) )
   BEGIN
       TRUNCATE TABLE GamerAudit;
       DROP TABLE GamerAudit;
   END;
GO -- Run the previous command and begins new batch
CREATE TABLE GamerAudit
   (
      ID INT IDENTITY(1, 1)
            PRIMARY KEY ,
      AuditData NVARCHAR(1000)
   );
GO -- Run the prvious command and begins new batch
```

2.2. DML Trigger: After/For Insert Trigger

```
--==========================================================
--T016_02_02
--DML Trigger: After/For Insert Trigger
IF EXISTS ( SELECT  *
            FROM    sys.triggers
            WHERE   name = 'trgGamerForInsert' )
   BEGIN
       DROP TRIGGER trgGamerForInsert;
   END;
GO -- Run the previous command and begins new batch
CREATE TRIGGER trgGamerForInsert ON Gamer
   --AFTER INSERT
   FOR INSERT
AS
   BEGIN
       DECLARE @ID INT;
       SELECT  @ID = ID
       FROM    inserted;

       INSERT  INTO GamerAudit
       VALUES  ( 'New Gamer with Id  = ' + CAST(@ID AS NVARCHAR(10))
                 + ' is inserted at ' + CAST(GETDATE() AS NVARCHAR(20)) );
```

```
    END;
GO -- Run the prvious command and begins new batch
/*
2.1.
After/For Trigger:
After/For Triggers fires after the INSERT/UPDATE/DELETE event happened.
After/For Trigger Syntax:
--CREATE TRIGGER {TriggerName} ON {TableName}
--{ After/For Insert | AFTER/For DELETE | AFTER/For UPDATE }
--AS
--    BEGIN
--        ...
--    END
3.1.
AFTER/FOR INSERT Trigger / INSTEAD OF INSERT Trigger:
INSERTED table structure is same as {TableName} structure,
and it contains the copy version of data rows we inserted.
This table can only be accessed by the Trigger.
*/
```

2.3. DML Trigger: After/For DELETE Trigger

```
--==============================================================
--T016_02_03
--DML Trigger: After/For DELETE Trigger
IF EXISTS ( SELECT  *
            FROM    sys.triggers
            WHERE   name = 'trgGamerForDelete' )
    BEGIN
        DROP TRIGGER trgGamerForDelete;
    END;
GO -- Run the previous command and begins new batch
CREATE TRIGGER trgGamerForDelete ON Gamer
        --AFTER DELETE
    FOR DELETE
AS
    BEGIN
        DECLARE @ID INT;
        SELECT  @ID = ID
        FROM    deleted;

        INSERT  INTO GamerAudit
        VALUES  ( 'An existing Gamer with Id  = ' + CAST(@ID AS NVARCHAR(10))
                  + ' is deleted at ' + CAST(GETDATE() AS NVARCHAR(20)) );
    END;
GO -- Run the prvious command and begins new batch
/*
2.1.
After/For Trigger:
After/For Triggers fires after the INSERT/UPDATE/DELETE event happened.
After/For Trigger Syntax:
--CREATE TRIGGER {TriggerName} ON {TableName}
--{ After/For Insert | AFTER/For DELETE | AFTER/For UPDATE }
--AS
--    BEGIN
--        ...
--    END
3.2.
AFTER/FOR DELETE Trigger / INSTEAD OF DELETE Trigger:
DELETED table structure is same as {TableName} structure,
and it contains the copy version of data rows we deleted.
```

```
This table can only be accessed by the Trigger.
*/
```

2.4. DML Trigger: After/For UPDATE Trigger

```
--==========================================================
--T016_02_04
--DML Trigger: After/For UPDATE Trigger
IF EXISTS ( SELECT  *
            FROM    sys.triggers
            WHERE   name = 'trgGamerForUpdate' )
    BEGIN
        DROP TRIGGER trgGamerForUpdate;
    END;
GO -- Run the previous command and begins new batch
CREATE TRIGGER trgGamerForUpdate ON Gamer
    FOR UPDATE
AS
    BEGIN
        SELECT  *
        FROM    deleted;
        SELECT  *
        FROM    inserted;
              DECLARE @ID INT;
        SELECT  @ID = ID
        FROM    deleted;
        INSERT  INTO GamerAudit
        VALUES  ( 'An existing Gamer with Id  = ' + CAST(@ID AS NVARCHAR(10))
                  + ' is updated at ' + CAST(GETDATE() AS NVARCHAR(20)) );
    END;
GO -- Run the prvious command and begins new batch
/*
2.1.
After/For Trigger:
After/For Triggers fires after the INSERT/UPDATE/DELETE event happened.
After/For Trigger Syntax:
--CREATE TRIGGER {TriggerName} ON {TableName}
--{ After/For Insert | AFTER/For DELETE | AFTER/For UPDATE }
--AS
--    BEGIN
--        ...
--    END
3.3.
AFTER/FOR UPDATE Trigger / INSTEAD OF UPDATE Trigger:
The DELETED table contains the copy version of old data.
The INSERTED table contains the copy version of new data.
Both table structures are same as {TableName} structure
and can can only be accessed by the Trigger.
*/
```

2.5. AFTER/FOR INSERT Trigger fires

```
--==========================================================
--T016_02_05
--AFTER/FOR INSERT Trigger fires
INSERT  INTO Gamer
VALUES  ( 7, 'Name7', '7@7.com', 2, 48000 );
GO -- Run the prvious command and begins new batch
```

```
--AFTER/FOR DELETE Trigger fires.
DELETE  dbo.Gamer
WHERE   ID = 7;
GO -- Run the prvious command and begins new batch
--AFTER/FOR DELETE UPDATE fires.
UPDATE  Gamer
SET     Name = 'NewName5' ,
             --Email = '5_1@5_1.com'.
        GenderID = 1 ,
        GameScore = 58000
WHERE   ID = 5;
GO -- Run the prvious command and begins new batch
```

	ID	Name	Email	GenderID	GameScore
1	5	Name5	5@5.com	1	42000

	ID	Name	Email	GenderID	GameScore
1	5	NewName5	5@5.com	1	58000

```
SELECT  *
FROM    GamerAudit;
GO -- Run the prvious command and begins new batch
```

	ID	AuditData
1	1	New Gamer with Id = 7 is inserted at Nov 9 2017 3:24AM
2	2	An existing Gamer with Id = 7 is deleted at Nov 9 2017 3:24AM
3	3	An existing Gamer with Id = 5 is updated at Nov 9 2017 3:24AM

2.6. DML Trigger: After/For UPDATE Trigger

```
--==============================================================
--T016_02_06
--DML Trigger: After/For UPDATE Trigger
ALTER TRIGGER trgGamerForUpdate ON Gamer
    FOR UPDATE
AS
    BEGIN
             -- Declare variables to hold old and updated data
        DECLARE @ID INT;
        DECLARE @OldName NVARCHAR(100) ,
             @NewName NVARCHAR(100);
        DECLARE @OldEmail NVARCHAR(100) ,
             @NewEmail NVARCHAR(100);
        DECLARE @OldGenderID INT ,
             @NewGenderID INT;
        DECLARE @OldGameScore INT ,
             @NewGameScore INT;

             -- Audit string variable
        DECLARE @AuditString NVARCHAR(1000);

             -- inserted contains updated data
             -- Load the updated data into local temporary table
        SELECT  *
        INTO    #TempTable
        FROM    inserted;
```

```sql
    -- Loop through the data row in temp table
    WHILE ( EXISTS ( SELECT ID
                     FROM   #TempTable ) )
        BEGIN
                        --Initialize the audit string
            SET @AuditString = '';

                        -- Select first row data from local temp table
                        -- which contains updated data.
            SELECT TOP 1
                    @ID = ID ,
                    @NewName = Name ,
                    @NewEmail = Email ,
                    @NewGenderID = GenderID ,
                    @NewGameScore = GameScore
            FROM    #TempTable;

                        -- Select correspond row data from deleted tables
                        -- which contains old data.
            SELECT  @OldName = Name ,
                    @OldEmail = Email ,
                    @OldGenderID = GenderID ,
                    @OldGameScore = GameScore
            FROM    deleted
            WHERE   ID = @ID;

                        -- Build the audit string
            SET @AuditString = 'Gamer with Id = '
                + CAST(@ID AS NVARCHAR(4)) + ' changed';
            IF ( @OldName <> @NewName )
                SET @AuditString = @AuditString + ' NAME from ' + @OldName
                    + ' to ' + @NewName;

            IF ( @OldEmail <> @NewEmail )
                SET @AuditString = @AuditString + ' Email from '
                    + @OldEmail + ' to ' + @NewEmail;

            IF ( @OldGenderID <> @NewGenderID )
                SET @AuditString = @AuditString + ' GenderID from '
                    + CAST(@OldGenderID AS NVARCHAR(10)) + ' to '
                    + CAST(@NewGenderID AS NVARCHAR(10));

            IF ( @OldGameScore <> @NewGameScore )
                SET @AuditString = @AuditString + ' GameScore from '
                    + CAST(@OldGameScore AS NVARCHAR(10)) + ' to '
                    + CAST(@NewGameScore AS NVARCHAR(10));

            INSERT  INTO GamerAudit
            VALUES  ( @AuditString );

                        -- Delete the row from temp table,
                        -- so we can move to the next row
            DELETE  FROM #TempTable
            WHERE   ID = @ID;
        END;
END;
```

```
GO -- Run the prvious command and begins new batch
UPDATE  Gamer
SET     Name = 'NewName3' ,
             --Email = '3_1@3_1.com'.
        GenderID = 1 ,
        GameScore = 58000
WHERE   ID = 3;
GO -- Run the prvious command and begins new batch
SELECT  *
FROM    GamerAudit;
/*
Gamer with Id = 3 changed NAME from Name3 to NewName3 GenderID from 2 to 1 GameScore from 41000 to 58000
*/
```

	ID	AuditData
1	1	New Gamer with Id = 7 is inserted at Nov 9 2017 3:24AM
2	2	An existing Gamer with Id = 7 is deleted at Nov 9 2017 3:24AM
3	3	An existing Gamer with Id = 5 is updated at Nov 9 2017 3:24AM
4	4	Gamer with Id = 3 changed NAME from Name3 to NewName3 GenderID from 2 to 1 GameScore from 41000 to 58000

2.7. Clean up

```
--==========================================================================
--T016_02_07
--Clean up
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'Gamer' ) )
   BEGIN
       TRUNCATE TABLE Gamer;
       DROP TABLE Gamer;
   END;
GO -- Run the previous command and begins new batch
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'GamerAudit' ) )
   BEGIN
       TRUNCATE TABLE GamerAudit;
       DROP TABLE GamerAudit;
   END;
GO -- Run the previous command and begins new batch
IF EXISTS ( SELECT  *
            FROM    sys.triggers
            WHERE   name = 'trgGamerForInsert' )
   BEGIN
       DROP TRIGGER trgGamerForInsert;
   END;
GO -- Run the previous command and begins new batch
IF EXISTS ( SELECT  *
            FROM    sys.triggers
            WHERE   name = 'trgGamerForDelete' )
   BEGIN
       DROP TRIGGER trgGamerForDelete;
   END;
GO -- Run the previous command and begins new batch
```

```
IF EXISTS ( SELECT  *
            FROM    sys.triggers
            WHERE   name = 'trgGamerForUpdate' )
    BEGIN
        DROP TRIGGER trgGamerForUpdate;
    END;
GO -- Run the previous command and begins new batch
```

3. Create Sample Data

```
--==============================================================
--T016_03_Create Sample Data
--==============================================================
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE      TABLE_NAME = 'vwPlayerTeam' ) )
    BEGIN
        DROP VIEW vwPlayerTeam;
    END;
GO -- Run the previous command and begins new batch
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE      TABLE_NAME = 'Player' ) )
    BEGIN
        TRUNCATE TABLE Player;
        DROP TABLE Player;
    END;
GO -- Run the previous command and begins new batch
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE      TABLE_NAME = 'Team' ) )
    BEGIN
        TRUNCATE TABLE Team;
        DROP TABLE Team;
    END;
GO -- Run the previous command and begins new batch
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE      TABLE_NAME = 'Gender' ) )
    BEGIN
        TRUNCATE TABLE Gender;
        DROP TABLE Gender;
    END;
GO -- Run the previous command and begins new batch
CREATE TABLE Gender
(
  GenderID INT IDENTITY(1, 1)
               PRIMARY KEY
               NOT NULL ,
```

```sql
  Gender NVARCHAR(50) NOT NULL,
 );
GO -- Run the previous command and begins new batch
INSERT Gender VALUES (N'Male')
INSERT Gender VALUES (N'Female')
INSERT Gender VALUES (N'Unknow')
CREATE TABLE Team
    (
       TeamID INT IDENTITY(1, 1)
                        PRIMARY KEY
                        NOT NULL ,
       TeamName NVARCHAR(100)
    );
GO -- Run the prvious command and begins new batch
INSERT  INTO Team
VALUES  ( 'Team1' );
INSERT  INTO Team
VALUES  ( 'Team2' );
INSERT  INTO Team
VALUES  ( 'Team3' );
INSERT  INTO Team
VALUES  ( 'Team4' );
GO -- Run the prvious command and begins new batch
CREATE TABLE Player
(
  PlayerId INT IDENTITY(1, 1)
               PRIMARY KEY
               NOT NULL ,
  Name NVARCHAR(100) ,
  GenderID INT FOREIGN KEY REFERENCES Gender ( GenderID ),
  TeamID INT FOREIGN KEY REFERENCES Team ( TeamID )
);
GO -- Run the prvious command and begins new batch
INSERT  INTO Player
VALUES  ( 'Name1', 1, 2 );
INSERT  INTO Player
VALUES  ( 'Name2', 2, 2 );
INSERT  INTO Player
VALUES  ( 'Name3', 2, 1 );
INSERT  INTO Player
VALUES  ( 'Name4', 1, 4 );
INSERT  INTO Player
VALUES  ( 'Name5', 3, 2 );
GO -- Run the prvious command and begins new batch
CREATE VIEW vwPlayerTeam
AS
    SELECT  p.PlayerId ,
            p.Name ,
            p.GenderID ,
                    p.TeamID,
            t.TeamName
    FROM    dbo.Player p
            INNER JOIN dbo.Team t ON p.TeamID = t.TeamID;
GO -- Run the prvious command and begins new batch
```

```
SELECT  *
FROM    Player;
SELECT  *
FROM    Team;
SELECT  *
FROM    Gender;
SELECT  *
FROM    vwPlayerTeam;
GO -- Run the prvious command and begins new batch
```

	PlayerId	Name	GenderID	TeamID
1	1	Name1	1	2
2	2	Name2	2	2
3	3	Name3	2	1
4	4	Name4	1	4
5	5	Name5	3	2

	TeamID	TeamName
1	1	Team1
2	2	Team2
3	3	Team3
4	4	Team4

	GenderID	Gender
1	1	Male
2	2	Female
3	3	Unknow

	PlayerId	Name	GenderID	TeamID	TeamName
1	1	Name1	1	2	Team2
2	2	Name2	2	2	Team2
3	3	Name3	2	1	Team1
4	4	Name4	1	4	Team4
5	5	Name5	3	2	Team2

==========

4. InsteadOfInsertTrigger : Fix Incorrectly Insert VIEW

```
--==============================================================
--T016_04_InsteadOfInsertTrigger : Fix Incorrectly Insert VIEW
--==============================================================
```

4.1. Incorrectly Insert VIEW

```
--==============================================================
--T016_04_01
--Incorrectly Insert VIEW
INSERT  INTO vwPlayerTeam
        ( Name, GenderID, TeamName )
VALUES  ( 'Name6', 1, 'NotExistTeam' );
```

```
GO -- Run the prvious command and begins new batch
--Retturn Error
```

```
Messages
Msg 4405, Level 16, State 1, Line 572
View or function 'vwPlayerTeam' is not updatable because the modification affects multiple base tables.
```

4.2. Fix the incorrectly Insert VIEW

```
--==========================================================
--T016_04_02
--Fix the incorrectly Insert VIEW
IF EXISTS ( SELECT  *
            FROM    sys.triggers
            WHERE   name = 'trVwPlayerTeam_InsteadOfInsert' )
    BEGIN
        DROP TRIGGER trVwPlayerTeam_InsteadOfInsert;
    END;
GO -- Run the previous command and begins new batch
CREATE TRIGGER trVwPlayerTeam_InsteadOfInsert ON vwPlayerTeam
    INSTEAD OF INSERT
AS
    BEGIN
        DECLARE @TeamID INT;
        DECLARE @GenderID INT;

             --check if the input TeamName is valid and exists in the TeamTable.
        SELECT  @TeamID = Team.TeamID
        FROM    dbo.Team
                JOIN inserted ON inserted.TeamName = Team.TeamName;
             --If TeamId is in-valid, then raise Error and Stop processing
        IF ( @TeamID IS NULL )
            BEGIN
                RAISERROR('Invalid Team Name. Statement terminated', 16, 1);
                RETURN;
            END;

             --check if the input GenderID is valid and exists in the TeamTable.
        SELECT  @GenderID = inserted.GenderID
        FROM    dbo.Gender
                JOIN inserted ON inserted.GenderID = dbo.Gender.GenderID;
             --If enderID is in-valid, then raise Error and Stop processing
        IF ( @GenderID IS NULL )
            BEGIN
                RAISERROR('Invalid GenderID. Statement terminated', 16, 1);
                RETURN;
            END;
             --Insert into Gamer table
        INSERT  INTO dbo.Player
                ( Name ,
                  GenderID ,
                  TeamID
                )
                SELECT  [Name] ,
                        @GenderID ,
                        @TeamID
                FROM    inserted i;
```

```
    END;
GO -- Run the prvious command and begins new batch
/*
1.
--SELECT  @TeamID = Team.TeamID
--FROM    dbo.Team
--        JOIN inserted ON inserted.TeamName = Team.TeamName;
AFTER/FOR INSERT Trigger / INSTEAD OF INSERT Trigger:
INSERTED table structure is same as {TableName} structure,
and it contains the copy version of data rows we inserted.
This table can only be accessed by the Trigger.
2.
--IF ( @TeamID is null )
--    BEGIN
--        RAISERROR('Invalid Team Name. Statement terminated', 16, 1)
--        RETURN
--    END
2.1.
If TeamId is in-valid, then raise Error and Stop processing.
IF ( @TeamID is null ) means
the input TeamName is NOT exist in the TeamTable.
Then raise the Error
--RAISERROR('Invalid Team Name. Statement terminated', 16, 1)
RETURN will stop processing.
2.2.
--RAISERROR ( { msg_str | @local_variable }
--    { ,severity ,state }
--    [ ,argument [ ,...n ] ] )
--    [ WITH option [ ,...n ] ]
Reference:
https://docs.microsoft.com/en-us/sql/t-sql/language-elements/raiserror-transact-sql
2.2.1.
The first parameter, msg_str, is the error message.
2.2.2.
the second parameter, severity, is the severity level.
Severity level 16 means general errors and can be corrected by the user.
2.2.3.
The third parameter is state, and we should set default to 1.
RAISERROR only generates errors with state from 1 through 18.
Because the PDW engine may raise errors with state 0,
using a unique state number for different location
can help find which section of code is raising the errors.
*/
```

4.3. Fix the incorrectly Insert VIEW

```
--==============================================================
--T016_04_03
--Fix the incorrectly Insert VIEW
--T016_04_03_01
--Insert with a invalid teamName
INSERT  INTO vwPlayerTeam
        ( Name, GenderID, TeamName )
VALUES  ( 'Name6', 1, 'NotExistTeam' );
GO -- Run the prvious command and begins new batch
--Return Error,
--because 'NotExistTeam' is a invalid teamName
```

```
Messages
Msg 50000, Level 16, State 1, Procedure trVwPlayerTeam_InsteadOfInsert, Line 15 [Batch Start Line 671]
Invalid Team Name. Statement terminated

(1 row affected)
```

```
--T016_04_03_02
--Insert with a valid teamName
INSERT  INTO vwPlayerTeam
```

```sql
        ( Name, GenderID, TeamName )
VALUES  ( 'Name6', 1, 'Team1' );
GO -- Run the prvious command and begins new batch
SELECT  *
FROM    vwPlayerTeam
WHERE   PlayerId >= 5;
SELECT  *
FROM    dbo.Player
WHERE   PlayerId >= 5;
SELECT  *
FROM    dbo.Team;
GO -- Run the prvious command and begins new batch
```

	PlayerId	Name	GenderID	TeamID	TeamName
1	5	Name5	3	2	Team2
2	6	Name6	1	1	Team1

	PlayerId	Name	GenderID	TeamID
1	5	Name5	3	2
2	6	Name6	1	1

	TeamID	TeamName
1	1	Team1
2	2	Team2
3	3	Team3
4	4	Team4

==========

5. InsteadOfInsertTrigger : Fix Incorrectly Update VIEW

```sql
--==============================================================
--T016_05_InsteadOfInsertTrigger : Fix Incorrectly Update VIEW
--==============================================================
```

5.1. Incorrectly Update VIEW

```sql
--==============================================================
--T016_05_01
--Incorrectly Update VIEW
SELECT  *
FROM    vwPlayerTeam;
```

	PlayerId	Name	GenderID	TeamID	TeamName
1	1	Name1	1	2	Team2
2	2	Name2	2	2	Team2
3	3	Name3	2	1	Team1
4	4	Name4	1	4	Team4
5	5	Name5	3	2	Team2
6	6	Name6	1	1	Team1

```sql
UPDATE  vwPlayerTeam
```

```
SET      TeamName = 'Team3'
WHERE    PlayerId = 5;
SELECT  *
FROM     vwPlayerTeam;
```

	PlayerId	Name	GenderID	TeamID	TeamName
1	1	Name1	1	2	Team3
2	2	Name2	2	2	Team3
3	3	Name3	2	1	Team1
4	4	Name4	1	4	Team4
5	5	Name5	3	2	Team3
6	6	Name6	1	1	Team1

```
SELECT  *
FROM     Team;
```

	TeamID	TeamName
1	1	Team1
2	2	Team3
3	3	Team3
4	4	Team4

```
/*
In Team table,
Look at the TeamID=2,
its TeamName became "Team3"
This is wrong.
*/
--Clean up
UPDATE   Team
SET      TeamName = 'Team1'
WHERE    TeamID = 1;
UPDATE   Team
SET      TeamName = 'Team2'
WHERE    TeamID = 2;
UPDATE   Team
SET      TeamName = 'Team3'
WHERE    TeamID = 3;
UPDATE   Team
SET      TeamName = 'Team4'
WHERE    TeamID = 4;
SELECT  *
FROM     Team;
GO -- Run the prvious command and begins new batch
```

	TeamID	TeamName
1	1	Team1
2	2	Team2
3	3	Team3
4	4	Team4

5.2. Fix the incorrectly Update VIEW

```
--==========================================================
--T016_05_02
--Fix the incorrectly Update VIEW
IF EXISTS ( SELECT  *
```

```sql
            FROM    sys.triggers
            WHERE   name = 'trVwPlayerTeam_InsteadOfUpdate' )
    BEGIN
        DROP TRIGGER trVwPlayerTeam_InsteadOfUpdate;
    END;
GO -- Run the previous command and begins new batch
CREATE TRIGGER trVwPlayerTeam_InsteadOfUpdate ON vwPlayerTeam
    INSTEAD OF UPDATE
AS
    BEGIN
              -- if someone tries to update or insert PlayerId
        IF ( UPDATE(PlayerId) )
            BEGIN
                RAISERROR('PlayerId is PK and unchangeable.', 16, 1);
                RETURN;
            END;
              -- if someone tries to update or insert GenderID
        IF ( UPDATE(GenderID) )
            BEGIN
                DECLARE @GenderID INT;
                            --check if the input GenderID is valid and exist in the GenderTable
                SELECT  @GenderID = Gender.GenderID
                FROM    dbo.Gender
                        JOIN inserted ON inserted.GenderID = Gender.GenderID;
                            --(@GenderID IS NULL)=true   means input GenderID is in-valid.
                IF ( @GenderID IS NULL )
                    BEGIN
                        RAISERROR('Invalid GenderID. Statement terminated', 16, 1);
                        RETURN;
                    END;
                            --if input GenderID is valid, then process updating.
                UPDATE  Player
                SET     GenderID = @GenderID
                FROM    inserted
                        JOIN Player ON Player.PlayerId = inserted.PlayerId;
            END;
              -- if someone tries to update or insert TeamName
        IF ( UPDATE(TeamName) )
            BEGIN
                DECLARE @TeamID INT;
                            --check if the input TeamName is valid and exist in the TeamTable
                SELECT  @TeamID = Team.TeamID
                FROM    Team
                        JOIN inserted ON inserted.TeamName = Team.TeamName;
                            --(@TeamID IS NULL)=true   means input TeamName is in-valid.
                IF ( @TeamID IS NULL )
                    BEGIN
                        RAISERROR('Invalid Team Name. Statement terminated', 16, 1);
                        RETURN;
                    END;

                            --if input TeamName is valid, then process updating.
                UPDATE  Player
                SET     TeamID = @TeamID
                FROM    inserted
```

```
                    JOIN Player ON Player.PlayerId = inserted.PlayerId;
        END;
                -- if someone tries to update or insert Name
    IF ( UPDATE(Name) )
        BEGIN
                        --Name is NOT a foreign key,
                        --thus, does not need to check if valid
                        --then process updating.
            UPDATE  Player
            SET     Name = inserted.Name
            FROM    inserted
                    JOIN Player ON Player.PlayerId = inserted.PlayerId;
        END;
    END;
GO -- Run the prvious command and begins new batch
/*
1.
INSTEAD OF UPDATE Trigger:
The DELETED table contains the old data.
The INSERTED table contains the updated data.
2.
--IF ( UPDATE(PlayerId) )
if someone tries to update or insert PlayerId, then ...
2.1.
UPDATE(column)
Reference:
https://docs.microsoft.com/en-us/sql/t-sql/functions/update-trigger-functions-transact-sql
Returns a Boolean value that indicates whether an
INSERT or UPDATE attempt was made on a specified column of a table or view.
*/
```

5.3. Fix the incorrectly Update VIEW

```
--==========================================================
--T016_05_03
--Fix the incorrectly Update VIEW
```

5.3.1. Update Name

```
-----------------------------------------------------------
--T016_05_03_01
--Update Name and TeamName
SELECT  *
FROM    vwPlayerTeam
WHERE   PlayerId = 1;
SELECT  *
FROM    Player
WHERE   PlayerId = 1;
SELECT  *
FROM    Team;
```

	PlayerId	Name	GenderID	TeamID	TeamName
1	1	Name1	1	2	Team2

	PlayerId	Name	GenderID	TeamID
1	1	Name1	1	2

	TeamID	TeamName
1	1	Team1
2	2	Team2
3	3	Team3
4	4	Team4

```
UPDATE  vwPlayerTeam
SET     Name = 'Name1_1' ,
        TeamName = 'Team1'
WHERE   PlayerId = 1;
SELECT  *
FROM    vwPlayerTeam
WHERE   PlayerId = 1;
SELECT  *
FROM    Player
WHERE   PlayerId = 1;
SELECT  *
FROM    Team;
```

	PlayerId	Name	GenderID	TeamID	TeamName
1	1	Name1_1	1	1	Team1

	PlayerId	Name	GenderID	TeamID
1	1	Name1_1	1	1

	TeamID	TeamName
1	1	Team1
2	2	Team2
3	3	Team3
4	4	Team4

```
/*
Update PlayerId=1,
update his name to 'Name1_1' and TeamID to 1.
It updates correctly.
*/
```

5.3.2. Update Name and TeamName

```
-----------------------------------------------------------
--T016_05_03_02
--Update Name and TeamName
SELECT  *
FROM    vwPlayerTeam
WHERE   PlayerId = 1;
SELECT  *
FROM    Player
WHERE   PlayerId = 1;
SELECT  *
FROM    Team;
```

	PlayerId	Name	GenderID	TeamID	TeamName
1	1	Name1_1	1	1	Team1

	PlayerId	Name	GenderID	TeamID
1	1	Name1_1	1	1

	TeamID	TeamName
1	1	Team1
2	2	Team2
3	3	Team3
4	4	Team4

```
UPDATE  vwPlayerTeam
SET     Name = 'Name1_1_1' ,
        TeamName = 'Team3' ,
```

```
        GenderID = 3
WHERE   PlayerId = 1;
SELECT  *
FROM    vwPlayerTeam
WHERE   PlayerId = 1;
SELECT  *
FROM    Player
WHERE   PlayerId = 1;
SELECT  *
FROM    Team;
```

	PlayerId	Name	GenderID	TeamID	TeamName
1	1	Name1_1_1	3	3	Team3

	PlayerId	Name	GenderID	TeamID
1	1	Name1_1_1	3	3

	TeamID	TeamName
1	1	Team1
2	2	Team2
3	3	Team3
4	4	Team4

```
/*
Update PlayerId=1,
update his name to 'Name1_1_1'
and update his TeamID to 3.
and update his GenderID to 3.
It updates correctly.
*/
```

5.3.3. Clean up

```
------------------------------------------------------------
--T016_05_03_03
--Clean up
--Team
UPDATE  Team
SET     TeamName = 'Team1'
WHERE   TeamID = 1;
UPDATE  Team
SET     TeamName = 'Team2'
WHERE   TeamID = 2;
UPDATE  Team
SET     TeamName = 'Team3'
WHERE   TeamID = 3;
UPDATE  Team
SET     TeamName = 'Team4'
WHERE   TeamID = 4;
--Player
UPDATE  Player
SET     Name = 'Name1' ,
        GenderID = 1 ,
        TeamID = 2
WHERE   PlayerId = 1;
UPDATE  Player
SET     Name = 'Name2' ,
        GenderID = 2 ,
        TeamID = 2
WHERE   PlayerId = 2;
```

```sql
UPDATE  Player
SET     Name = 'Name3' ,
        GenderID = 2 ,
        TeamID = 1
WHERE   PlayerId = 3;
UPDATE  Player
SET     Name = 'Name4' ,
        GenderID = 1 ,
        TeamID = 4
WHERE   PlayerId = 4;
UPDATE  Player
SET     Name = 'Name5' ,
        GenderID = 3 ,
        TeamID = 2
WHERE   PlayerId = 5;
UPDATE  Player
SET     Name = 'Name6' ,
        GenderID = 1 ,
        TeamID = 1
WHERE   PlayerId = 6;
GO -- Run the prvious command and begins new batch
SELECT  *
FROM    Player;
SELECT  *
FROM    Team;
GO -- Run the prvious command and begins new batch
```

	PlayerId	Name	GenderID	TeamID
1	1	Name1	1	2
2	2	Name2	2	2
3	3	Name3	2	1
4	4	Name4	1	4
5	5	Name5	3	2
6	6	Name6	1	1

	TeamID	TeamName
1	1	Team1
2	2	Team2
3	3	Team3
4	4	Team4

6. InsteadOfInsertTrigger : Fix Incorrectly Delete VIEW

```sql
--==========================================================
--T016_06_InsteadOfInsertTrigger : Fix Incorrectly Delete VIEW
--==========================================================
```

6.1. Incorrectly Delete VIEW

```sql
--==========================================================
--T016_06_01
--Incorrectly Delete VIEW
DELETE  FROM vwPlayerTeam
WHERE   PlayerId = 1;
GO -- Run the prvious command and begins new batch
--Return Error
--SQL server is confused that whether it should delete Player or delete Team
```

```
Messages
Msg 4405, Level 16, State 1, Line 984
View or function 'vwPlayerTeam' is not updatable because the modification affects multiple base tables.
```

6.2. Fix incorrectly Delete VIEW

```sql
--==========================================================
--T016_06_02
--Fix incorrectly Delete VIEW
IF EXISTS ( SELECT  *
            FROM    sys.triggers
            WHERE   name = 'trVwPlayerTeam_InsteadOfDelete' )
    BEGIN
        DROP TRIGGER trVwPlayerTeam_InsteadOfDelete;
    END;
GO -- Run the previous command and begins new batch
CREATE TRIGGER trVwPlayerTeam_InsteadOfDelete ON vwPlayerTeam
    INSTEAD OF DELETE
AS
    BEGIN
              ----Join
        --Delete  dbo.Player
        --FROM    dbo.Player
        --        join deleted on Player.PlayerId = deleted.PlayerId
              --SubQuery
        DELETE  FROM dbo.Player
        WHERE   PlayerId IN ( SELECT  PlayerId
                              FROM    deleted );
    END;
GO -- Run the prvious command and begins new batch
/*
In most cases JOINs are faster than SUB-QUERIEs.
However, in this case, you only need one subset of data rows which is PlayerId.
thus, in this case, SubQuery is slightly faster.
*/
```

6.3. Fix incorrectly Delete VIEW

```sql
--==========================================================
--T016_06_03
--Fix incorrectly Delete VIEW
SELECT  *
FROM    vwPlayerTeam
WHERE   PlayerId <= 2;
SELECT  *
FROM    Player
WHERE   PlayerId <= 2;
```

```
SELECT  *
FROM    Team;
GO -- Run the prvious command and begins new batch
```

	PlayerId	Name	GenderID	TeamID	TeamName
1	1	Name1	1	2	Team2
2	2	Name2	2	2	Team2

	PlayerId	Name	GenderID	TeamID
1	1	Name1	1	2
2	2	Name2	2	2

	TeamID	TeamName
1	1	Team1
2	2	Team2
3	3	Team3
4	4	Team4

```
DELETE  FROM vwPlayerTeam
WHERE   PlayerId = 1;
GO -- Run the prvious command and begins new batch
SELECT  *
FROM    vwPlayerTeam
WHERE   PlayerId <= 2;
SELECT  *
FROM    Player
WHERE   PlayerId <= 2;
SELECT  *
FROM    Team;
GO -- Run the prvious command and begins new batch
```

	PlayerId	Name	GenderID	TeamID	TeamName
1	2	Name2	2	2	Team2

	PlayerId	Name	GenderID	TeamID
1	2	Name2	2	2

	TeamID	TeamName
1	1	Team1
2	2	Team2
3	3	Team3
4	4	Team4

==========

7. Clean up

```
--==========================================================
--T016_07_Clean up
--==========================================================
IF ( EXISTS ( SELECT    *
```

```sql
            FROM      INFORMATION_SCHEMA.TABLES
            WHERE      TABLE_NAME = 'vwPlayerTeam' ) )
    BEGIN
        DROP VIEW vwPlayerTeam;
    END;
GO -- Run the previous command and begins new batch
IF ( EXISTS ( SELECT    *
            FROM      INFORMATION_SCHEMA.TABLES
            WHERE      TABLE_NAME = 'Player' ) )
    BEGIN
        TRUNCATE TABLE Player;
        DROP TABLE Player;
    END;
GO -- Run the previous command and begins new batch
IF ( EXISTS ( SELECT    *
            FROM      INFORMATION_SCHEMA.TABLES
            WHERE      TABLE_NAME = 'Team' ) )
    BEGIN
        TRUNCATE TABLE Team;
        DROP TABLE Team;
    END;
GO -- Run the previous command and begins new batch
IF ( EXISTS ( SELECT    *
            FROM      INFORMATION_SCHEMA.TABLES
            WHERE      TABLE_NAME = 'Gender' ) )
    BEGIN
        TRUNCATE TABLE Gender;
        DROP TABLE Gender;
    END;
GO -- Run the previous command and begins new batch
------------------------------------------------------------
IF EXISTS ( SELECT  *
           FROM    sys.triggers
           WHERE   name = 'trVwPlayerTeam_InsteadOfInsert' )
    BEGIN
        DROP TRIGGER trVwPlayerTeam_InsteadOfInsert;
    END;
GO -- Run the previous command and begins new batch
IF EXISTS ( SELECT  *
           FROM    sys.triggers
           WHERE   name = 'trVwPlayerTeam_InsteadOfUpdate' )
    BEGIN
        DROP TRIGGER trVwPlayerTeam_InsteadOfUpdate;
    END;
GO -- Run the previous command and begins new batch
IF EXISTS ( SELECT  *
           FROM    sys.triggers
           WHERE   name = 'trVwPlayerTeam_InsteadOfDelete' )
    BEGIN
        DROP TRIGGER trVwPlayerTeam_InsteadOfDelete;
    END;
GO -- Run the previous command and begins new batch
```