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

1. Create Sample Data

-----------

2. DMLTriggers

2.1. CREATE TABLE GamerAudit

2.2. DML Trigger: After/For Insert Trigger

2.3. DML Trigger: After/For DELETE Trigger

2.4. DML Trigger: After/For UPDATE Trigger

2.5. AFTER/FOR INSERT Trigger fires

2.6. DML Trigger: After/For UPDATE Trigger

2.7. Clean up

-----------

3. Create Sample Data

-----------

4. InsteadOfInsertTrigger : Fix Incorrectly Insert VIEW

4.1. Incorrectly Insert VIEW

4.2. Fix the incorrectly Insert VIEW

4.3. Fix the incorrectly Insert VIEW

-----------

5. InsteadOfInsertTrigger : Fix Incorrectly Update VIEW

5.1. Incorrectly Update VIEW

5.2. Fix the incorrectly Update VIEW

5.3. Fix the incorrectly Update VIEW

5.3.1. Update Name

5.3.2. Update Name and TeamName

5.3.3. Clean up

-----------

6. InsteadOfInsertTrigger : Fix Incorrectly Delete VIEW

6.1. Incorrectly Delete VIEW

6.2. Fix incorrectly Delete VIEW

6.3. Fix incorrectly Delete VIEW

-----------

7. Clean up  
=======================================================================

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](mailto:4@4.com)', 1, 43000 );

INSERT  INTO Gamer

VALUES  ( 2, 'Name2', '[2@2.com](mailto:2@2.com)', 2, 44000 );

INSERT  INTO Gamer

VALUES  ( 1, 'Name1', '[1@1.com](mailto:1@1.com)', 1, 43000 );

INSERT  INTO Gamer

VALUES  ( 5, 'Name5', '[5@5.com](mailto:5@5.com)', 1, 42000 );

INSERT  INTO Gamer

VALUES  ( 3, 'Name3', '[3@3.com](mailto: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



==================================================

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](mailto: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](http://1.com/)'.

        GenderID = 1 ,

        GameScore = 58000

WHERE   ID = 5;

GO -- Run the prvious command and begins new batch

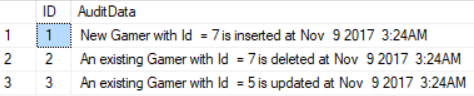
Graphical user interface, application, table

Description automatically generated

SELECT  \*

FROM    GamerAudit;

GO -- Run the prvious command and begins new batch



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;

        -- 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](http://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

\*/

Graphical user interface, text, application, Word

Description automatically generated

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 ,

  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;

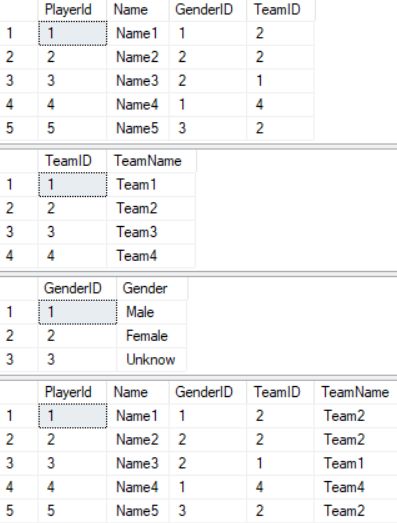
SELECT  \*

FROM    Gender;

SELECT  \*

FROM    vwPlayerTeam;

GO -- Run the prvious command and begins new batch



==================================================

4. InsteadOfInsertTrigger : Fix Incorrectly Insert VIEW

--============================================================

--T016\_04\_InsteadOfInsertTrigger : Fix Incorrectly Insert VIEW

--============================================================

4.1. Incorrectly Insert VIEW

--============================================================

--T016\_04\_01

--Incorrectly Insert VIEW

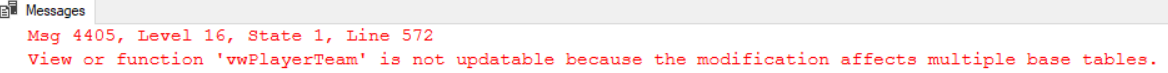
INSERT  INTO vwPlayerTeam

        ( Name, GenderID, TeamName )

VALUES  ( 'Name6', 1, 'NotExistTeam' );

GO -- Run the prvious command and begins new batch

--Retturn Error



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

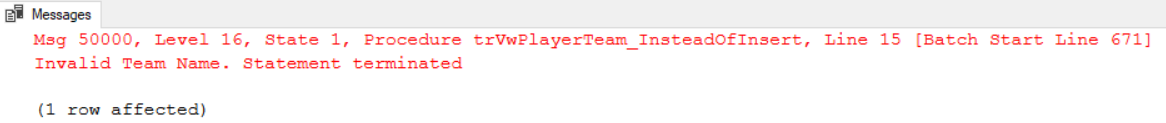
        ( Name, GenderID, TeamName )

VALUES  ( 'Name6', 1, 'NotExistTeam' );

GO -- Run the prvious command and begins new batch

--Return Error,

--because 'NotExistTeam' is a invalid teamName



--T016\_04\_03\_02

--Insert with a valid teamName

INSERT  INTO vwPlayerTeam

        ( 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

Graphical user interface, application, table, Excel

Description automatically generated

==================================================

5. InsteadOfInsertTrigger : Fix Incorrectly Update VIEW

--============================================================

--T016\_05\_InsteadOfInsertTrigger : Fix Incorrectly Update VIEW

--============================================================

5.1. Incorrectly Update VIEW

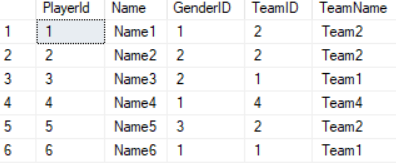
--============================================================

--T016\_05\_01

--Incorrectly Update VIEW

SELECT  \*

FROM    vwPlayerTeam;



UPDATE  vwPlayerTeam

SET     TeamName = 'Team3'

WHERE   PlayerId = 5;

SELECT  \*

FROM    vwPlayerTeam;

Graphical user interface, application

Description automatically generated

SELECT  \*

FROM    Team;

Table

Description automatically generated

/\*

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

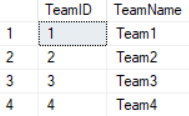
SET     TeamName = 'Team4'

WHERE   TeamID = 4;

SELECT  \*

FROM    Team;

GO -- Run the prvious command and begins new batch



5.2. Fix the incorrectly Update VIEW

--============================================================

--T016\_05\_02

--Fix the incorrectly Update VIEW

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

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;

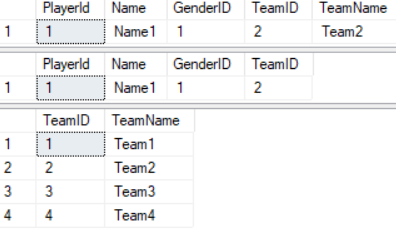
SELECT  \*

FROM    Player

WHERE   PlayerId = 1;

SELECT  \*

FROM    Team;



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;

Graphical user interface, application

Description automatically generated

/\*

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;

Shape

Description automatically generated with medium confidence

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;

Graphical user interface, application

Description automatically generated

/\*

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;

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

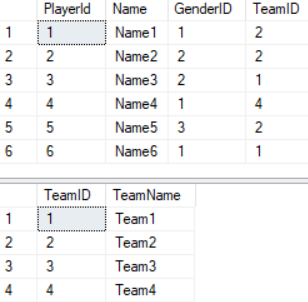
SELECT  \*

FROM    Player;

SELECT  \*

FROM    Team;

GO -- Run the prvious command and begins new batch



6. InsteadOfInsertTrigger : Fix Incorrectly Delete VIEW

--============================================================

--T016\_06\_InsteadOfInsertTrigger : Fix Incorrectly Delete VIEW

--============================================================

6.1. Incorrectly Delete VIEW

--============================================================

--T016\_06\_01

--Incorrectly Delete VIEW

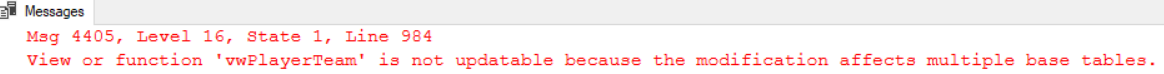
DELETE  FROM vwPlayerTeam

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



6.2. Fix incorrectly Delete VIEW

--============================================================

--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

--============================================================

--T016\_06\_03

--Fix incorrectly Delete VIEW

SELECT  \*

FROM    vwPlayerTeam

WHERE   PlayerId <= 2;

SELECT  \*

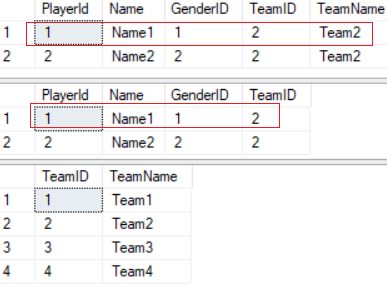
FROM    Player

WHERE   PlayerId <= 2;

SELECT  \*

FROM    Team;

GO -- Run the prvious command and begins new batch



DELETE  FROM vwPlayerTeam

WHERE   PlayerId = 1;

GO -- Run the prvious command and begins new batch

SELECT  \*

FROM    vwPlayerTeam

WHERE   PlayerId <= 2;

SELECT  \*

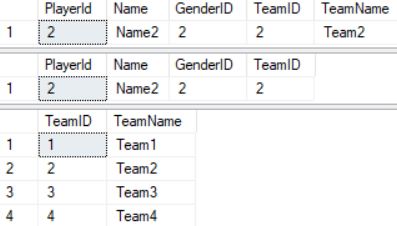
FROM    Player

WHERE   PlayerId <= 2;

SELECT  \*

FROM    Team;

GO -- Run the prvious command and begins new batch



============================================================

7. Clean up

--============================================================

--T016\_07\_Clean up

--============================================================

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

--------------------------------------------------------------

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