



Back to Basics – Triggers and Advanced Triggers

- Frank Rhodes
- Db2 Administration Solution Architect BMC Software

I am presenting a presentation put together by a couple of my team members. A big thank you to **Jaspreet Kaur and Pradip Wagh**.

They are both developers on one of my products, Change Manager for Db2.

They put the presentation together based upon the research they performed when working on adding Advanced Trigger support to Change Manager.

Agenda

Introduction to Triggers and Event Driven Triggers

Trigger Types and Differences (Basic and Advanced Trigger)

Why Should you Care About Advanced Triggers?

Advanced Trigger Operations, Syntax and Trigger Options

Advanced Trigger Ordering and Versioning

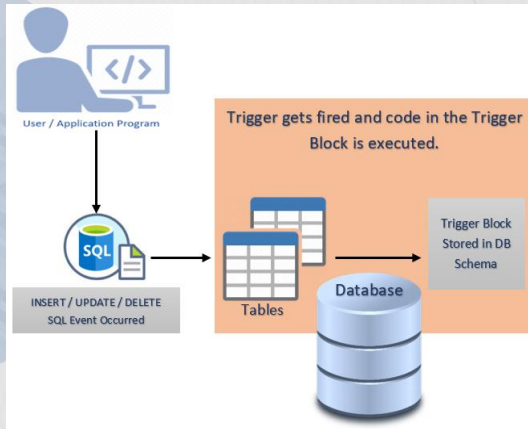
Trigger Restrictions, Considerations and Cascading Concepts

Advanced Trigger Example

Introduction to Triggers

- IBM added trigger support with Db2 Version 6.
- Definition:
Triggers are event-driven specialized procedures (a piece of code) that are stored in and managed by the RDBMS and executed in response to a data modification statement; that is, an Insert, Update, or Delete.
- Triggers are commonly used for:
 - Enforcing data integrity
 - Implementing business rules
 - Auditing changes
 - Performing tasks that require automatic actions, based on changes to the data
- Triggers move the business rule application logic into the database, which results in faster application development and easier maintenance. The business rule is centralized to the triggers and is no longer repeated in several applications.

Event-Driven Triggers



Each trigger is attached to a single, specified table

Once a trigger is created, it is always executed when its "firing" event occurs. Therefore, triggers are Automatic, Implicit, and Non-bypassable

DB2 triggers can be activated either "before" or "after" the SQL Event (for example, INSERT, UPDATE, and DELETE)

Triggers can also fire other triggers. However, it's important to design and use trigger chains carefully to avoid unintended consequences, performance issues, and potential infinite loops

4

A "before" trigger executes before activation time firing event occurs.

A "after" trigger executes after activation time firing event occurs.

Example – Suppose user is trying to delete specific data from a Table-A, and Table-A has Trigger defined to Insert this to be deleted row into another Housekeeping table before it gets deleted.

Types of Triggers

In Db2 12 for z/OS with new function mode activated, IBM enhanced support for triggers and introduced the object as an **advanced trigger**. IBM continues to support existing triggers from previous releases and refers to them as **basic triggers**.

Triggers type can be identified based on SQLPL column value stored in **SYSIBM.SYSTRIGGERS** table

Basic Trigger –
Value of SQLPL = Blank

Advanced Trigger –
Value of SQLPL = 'Y'

Advanced and Basic Trigger Differences

Basic Trigger	Advanced Trigger
Basic triggers don't have versions associated with them	Advanced triggers can define multiple versions of the trigger
Basic triggers support a limited set of SQL statements and require the MODE DB2SQL clause	Advanced triggers support a larger set of SQL statements, including SQL procedure language (SQL/PL) and must NOT specify the MODE DB2SQL clause
Changing any trigger options requires that the trigger be dropped and rebuilt	Changing trigger options can be achieved with an ALTER TRIGGER statement
Basic triggers cannot be debugged	Advanced triggers can be debugged
Basic triggers cannot include dynamic SQL statements	Advanced triggers can include dynamic SQL statements
BIND options cannot be explicitly specified for basic triggers	Various options including BIND options can be explicitly specified for advanced triggers
Only REBIND option is available for basic triggers	REBIND and REGENERATE options are available for advanced triggers

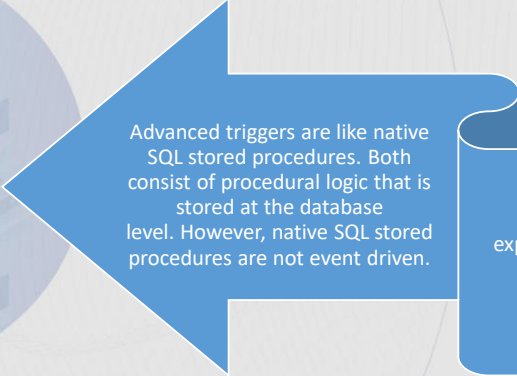
6

IBM Site Referred - https://www.ibm.com/docs/en/db2-for-zos/12?topic=concepts-triggers#db2z_triggers_sect-basicadv

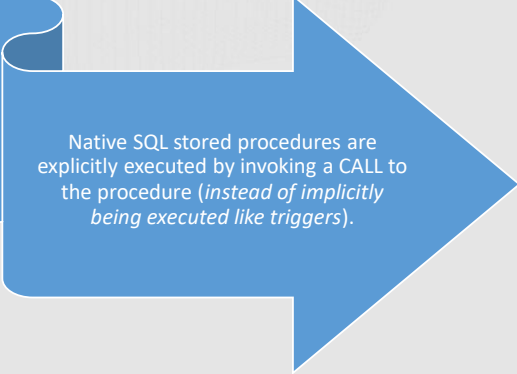
BIND Options like – ASUTIME, EXPLAIN, BUSINESS, SYSTEM AND ARCHIVE TIME SENSITIVE, ISOLATION LEVEL, etc.

Other Options like – APPLCOMPAT, DATE & TIME FORMAT, DEBUG MODE, etc.

Why Should you Care About Advanced Triggers? (1 | 2)



Advanced triggers are like native SQL stored procedures. Both consist of procedural logic that is stored at the database level. However, native SQL stored procedures are not event driven.



Native SQL stored procedures are explicitly executed by invoking a CALL to the procedure (*instead of implicitly being executed like triggers*).

Why Should you Care About Advanced Triggers? (2 | 2)

Prior to Db2 version 12, basic triggers supported a limited set of SQL statements because the trigger text had to call a stored procedure to provide this additional capability. As a result, it took increased time to develop and deploy applications.

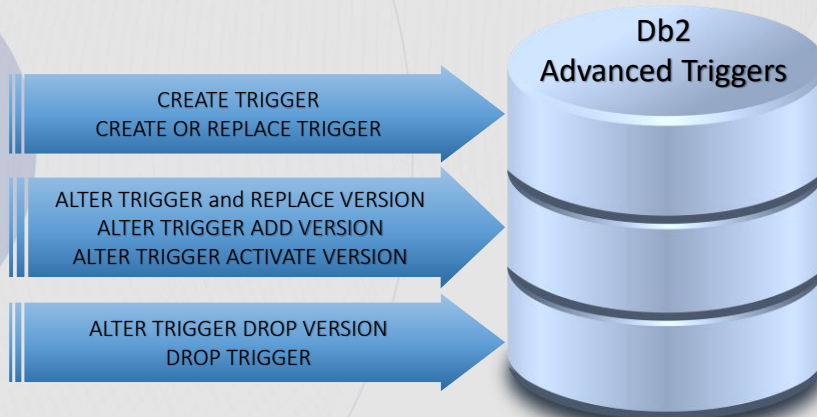
Calling a stored procedure in the code can be inefficient. This increases the cost of using Db2 for z/OS, degrades the application performance for CPU and elapsed time, and increases the maintenance cost to manage more objects and corresponding code.

With advanced triggers, users can directly use SQL/PL statements in the text instead of calling native SQL stored procedures.

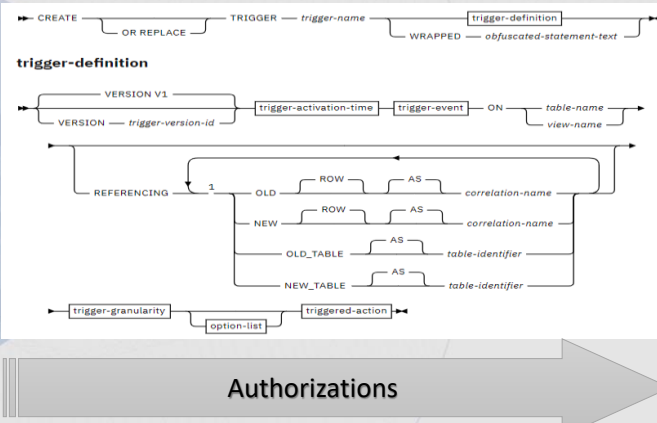
EXAMPLE

```
CREATE TRIGGER EMPSALRY
AFTER UPDATE ON EMPTABLE1
REFERENCING NEW TABLE AS NEWEMP
FOR EACH STATEMENT MODE DB2SQL
BEGIN ATOMIC
    CALL CHECKSAL(TABLE NEWEMP) ;
END
```


Operations on Advanced Triggers



CREATE (OR REPLACE) TRIGGER Syntax



The authorization set that is defined for CREATE TRIGGER must include at least one of the following:

- The CREATEIN privilege on the Schema
- SYSADM or SYSCTRL Authority
- System DBADM

Authorizations

Trigger Activation Time

– BEFORE, AFTER and INSTEAD OF

Trigger Event

– INSERT, UPDATE (of Column Name) and DELETE

Trigger Granularity

– FOR EACH ROW, FOR EACH STATEMENT (Multiple Rows)

Rows)

Trigger Actions

– WHEN (*search-condition*)

Example of Advanced Trigger - CREATE

CREATE TRIGGER

EXAMPLE

```
CREATE TRIGGER MYTRIGEXAMPLE01
VERSION V1
BEFORE INSERT ON MYTABLE1
REFERENCING NEW AS N1
FOR EACH ROW
ALLOW DEBUG MODE
QUALIFIER MVSPKW
WHEN (N1.TIMECOL IS NULL OR N1.TIMECOL > '21:00')
LBL1: BEGIN ATOMIC
  IF (N1.TIMECOL IS NULL) THEN
    SET N1.TIMECOL = N1.TIMECOL + 1 HOUR;
  END IF;
  IF (N1.TIMECOL > '21:00') THEN
    SIGNAL SQLSTATE '80000'
    SET MESSAGE_TEXT = 'Class ending time is beyond 9 pm';
  END IF;
END LBL1
```

Example of Advanced Trigger - OR REPLACE

CREATE
OR REPLACE
TRIGGER

EXAMPLE

```
CREATE OR REPLACE TRIGGER MYTRIGEXAMPLE01
VERSION V1
BEFORE INSERT ON MYTABLE1
REFERENCING NEW AS N1
FOR EACH ROW
ALLOW DEBUG MODE
QUALIFIER MVSPXW
WHEN (N1.TIMECOL IS NULL OR N1.TIMECOL > '21:00')
LBL1: BEGIN ATOMIC
  IF (N1.TIMECOL IS NULL) THEN
    SET N1.TIMECOL = N1.TIMECOL + 1 HOUR;
  END IF;
  IF (N1.TIMECOL > '21:00') THEN
    SIGNAL SQLSTATE '80000'
    SET MESSAGE_TEXT = 'Class ending time is beyond 9 pm';
  END IF;
END LBL1
```

Implicit Creation of Version "V1"

If the VERSION keyword is not specified, and the trigger does not exist

The trigger is created with the initial version (V1)

If the VERSION keyword is not specified, and the trigger exists

Version V1 does not exist

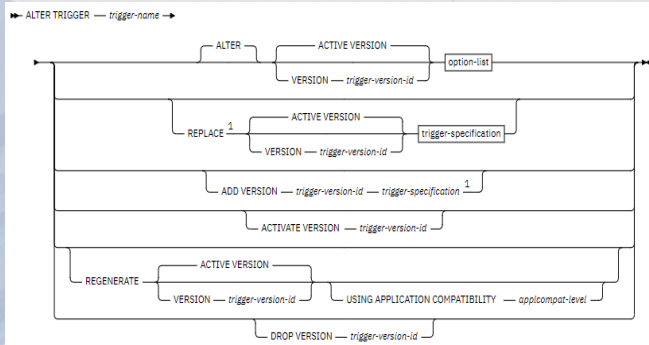
Db2 creates the trigger as version V1

Version V1 already exists

For **CREATE**, Db2 will issue an error since it will try to assign implicit Version ID as "V1"

For **CREATE OR REPLACE**, Db2 will replace the existing version V1 without issuing an error

ALTER TRIGGER Syntax



Authorizations

The authorization set that is defined for ALTER TRIGGER must include at least one of the following:

- Ownership of the Trigger
- The ALTERIN privilege on the Schema
- SYSADM authority
- SYSCtrl authority
- System DBADM

Examples of Advanced Trigger – ALTER (1 | 4)

ALTER TRIGGER VERSION
<Ver ID>

EXAMPLE

```
ALTER TRIGGER DBPXW05.MYTRIGGER_01  
VERSION V2  
ASUTIME LIMIT 20000 ;
```

ALTER TRIGGER ACTIVE
VERSION

EXAMPLE

```
ALTER TRIGGER DBPXW05.MYTRIGGER_01  
ACTIVE VERSION  
DATE FORMAT ISO  
TIME FORMAT ISO ;
```

ALTER TRIGGER
(without specifying Version ID
and ACTIVE VERSION)

EXAMPLE

```
ALTER TRIGGER DBPXW05.MYTRIGGER_01  
DATE FORMAT ISO  
TIME FORMAT ISO ;
```

Examples of Advanced Trigger – ALTER (2|4)

ALTER TRIGGER ADD
VERSION <Ver ID>

EXAMPLE

```
ALTER TRIGGER DBPXW05.MYTRIGGER_02
ADD VERSION TWO
AFTER UPDATE OF QUOTE ON DBPXW05.CURRENTQUOTE
REFERENCING NEW AS NEWQUOTE FOR EACH ROW
BEGIN ATOMIC
    INSERT INTO DBPXW05.QUOTEHISTORY VALUES
        ('TWO',NEWQUOTE.QUOTE,CURRENT_TIMESTAMP) ;
END ;
```


Examples of Advanced Trigger – ALTER (3 | 4)

ALTER TRIGGER
REPLACE VERSION

EXAMPLE

```
ALTER TRIGGER DBPXW05.MYTRIGGER_01
REPLACE VERSION V1
AFTER UPDATE OF QUOTE ON DBPXW05.CURRENTQUOTE
REFERENCING NEW AS NEWQUOTE FOR EACH ROW
ASUTIME LIMIT 2
BEGIN ATOMIC
    INSERT INTO DBPXW21.QUOTEHISTORY VALUES
        ('TWO',NEWQUOTE.QUOTE,CURRENT TIMESTAMP);
END;
```

ALTER TRIGGER
REPLACE ACTIVE
VERSION

EXAMPLE

```
ALTER TRIGGER DBPXW05.MYTRIGGER_01
REPLACE ACTIVE VERSION
AFTER UPDATE OF QUOTE ON DBPXW05.CURRENTQUOTE
REFERENCING NEW AS NEWQUOTE FOR EACH ROW
ASUTIME LIMIT 2
BEGIN ATOMIC
    INSERT INTO DBPXW21.QUOTEHISTORY VALUES
        ('TWO',NEWQUOTE.QUOTE,CURRENT TIMESTAMP);
END;
```

17

IBM Site Referred –

https://www.ibm.com/docs/en/db2-for-zos/12?topic=concepts-triggers#db2z_triggers_sect-basicadv

ALTER –

- Specifies that the trigger is to be changed. When you change one or more trigger options, any option that is not explicitly specified uses the existing value from the trigger that is being changed.

REPLACE –

- Specifies that a version of the trigger is to be replaced. When you replace a trigger, the signature of the trigger has to be maintained.
- For options that are not explicitly specified, the system default values for those options are used, even if those options were explicitly specified for the version of the trigger that is being replaced.

- Relace can not be used for the version of the trigger that is specified DISABLE DEBUG MODE. If DISABLE DEBUG MODE is specified for a version of a trigger, the option cannot be changed using the REPLACE clause.
- When a trigger definition is replaced, any existing comments in the catalog for that definition of the trigger are removed.
- Binding the replaced version of the trigger might result in a new access path even if the trigger body is not changed.

Examples of Advanced Trigger – ALTER (4|4)

ALTER TRIGGER ACTIVATE
VERSION <Ver ID>

EXAMPLE

```
ALTER TRIGGER DBPXW05.MYTRIGGER_01  
ACTIVATE VERSION V2 ;
```

ALTER TRIGGER
REGENERATE VERSION
<Ver ID>

EXAMPLE

```
ALTER TRIGGER DBPXW05.MYTRIGGER_02  
REGENERATE VERSION VERSIONID_01 ;
```

ALTER TRIGGER
REGENERATE ACTIVE
VERSION

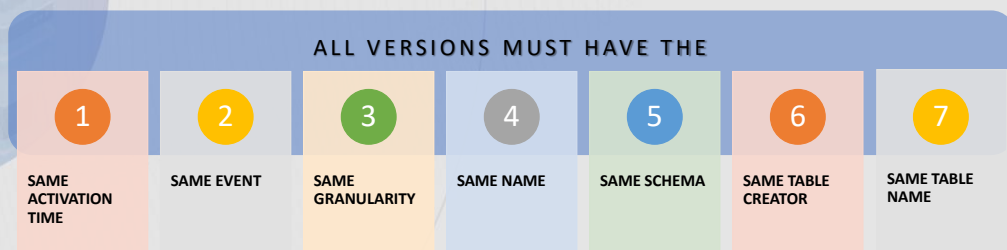
EXAMPLE

```
ALTER TRIGGER DBPXW05.MYTRIGGER_02  
REGENERATE ACTIVE VERSION ;
```

Signature of an Advanced Trigger

The signature of an advanced trigger comprises the seven properties shown in the following diagram. You must maintain the signature of an advanced trigger when you perform one of the following changes:

- Adding a new version
- Replacing an existing option
- Altering an existing option



19

```
CREATE TRIGGER MYTRIGEXAMPLE01  
VERSION V1  
BEFORE INSERT ON MYTABLE1
```

```
ALTER TRIGGER MYTRIGEXAMPLE01  
ADD VERSION V2  
AFTER INSERT ON MYTABLE1
```

Error Details –

```
DSNT408I SQLCODE = -4728, ERROR: ANOTHER VERSION OF OBJECT  
DBPXW05.TR_NEW01
```

```
EXISTS AND IS DEFINED WITH AN INCOMPATIBLE OPTION. THE OPTION  
IS TRIGGER EVENT
```

DROP TRIGGER Syntax

→ DROP →
← TRIGGER — trigger-name — →

The authorization set that is defined for DROP TRIGGER must include at least one of the following:

- The DROP privilege on the database
- DBADM or DBCTRL authority for the database
- SYSADM or SYSCTRL authority

Authorizations

Examples of Advanced Triggers - DROP

ALTER TRIGGER DROP
VERSION <Ver ID>

EXAMPLE

```
ALTER TRIGGER DBPXW05.MYTRIGGER_01  
DROP VERSION v1 ;
```

DROP TRIGGER

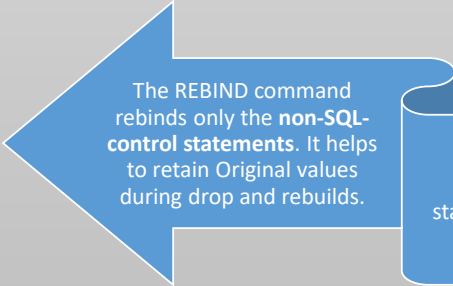
EXAMPLE

```
DROP TRIGGER DBPXW05.MYTRIGGER_02 ;
```

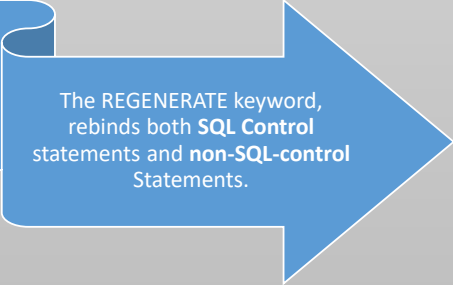
You can't drop the ACTIVE version; in that case you will need to activate the other existing version and then continue dropping with the current ACTIVE version.

Advanced Trigger – REBIND or REGENERATE

The REGENERATE keyword is available only for advanced triggers, whereas the REBIND command is for basic and advanced triggers.



The REBIND command rebinds only the **non-SQL-control statements**. It helps to retain Original values during drop and rebuilds.



The REGENERATE keyword, rebinds both **SQL Control statements** and **non-SQL-control Statements**.

Basic Trigger Detail

```
DEMO ALTER ----- Trigger Detail -----
Command ==>
WORKID . . . . : MVSPXW.BASICTRG
Timestamp : 2022-08-16-08.41.32.317968      New Values
TB Owner : RDANXA
TB Name : BASE1
Qualify TB: Y
TR Schema : DEM0323A
TR Name : TRIG_FOR_FUNC14
TR Owner : RDANXA
Version :
Type . . . . : BASIC
Add Version . . . . :
Activation Time : AFTER
Activation Event : DELETE
Granularity . . . : ROW
Referencing Correlations
OLD_AS :
NEW_AS :
OLD_TB AS:
NEW_TB AS:
Type . . . . : BASIC { Advanced, Basic }
Add Version . . . . : { N, Y }
Activation Time : AFTER { Before/After/Instead of }
Activation Event : DELETE { Insert/Update/Delete }
Granularity . . . : ROW { Row/Statement }
Referencing Correlations
OLD_AS :
NEW_AS :
OLD_TB AS:
NEW_TB AS:
Select additional panels to display then press Enter.
_ Action Text _ Update Column List _ Schema Path _ Comment _ Old Action Text
_ Trig options
```


Advanced Trigger Detail

```
DEMO ALTER ----- Trigger Detail -----
Command ==>
WORKID . . . : MVSPXW.ADVTRG03
Timestamp : 2023-03-03-05.10.04.128819      New Values
TB Owner  : DBPXW05
TB Name   : TBN1B
Qualify TB: Y
TR Schema : DBPXW05
TR Name   : TR_NEW01
TR Owner  : MVSPXW1
Version   : V1
Type      : ADVANCED
Add Version :
Activation Time : AFTER
Activation Event: UPDATE
Granularity  : ROW
Referencing Correlations
OLD AS :
NEW AS :
OLD_TB AS:
NEW_TB AS:
Select additional panels to display then press Enter.
_ Action Text _ Update Column List _ Schema Path _ Comment _ Old Action Text
_ Trig options
```

24

As far as parameters are concerned the only real addition for Advanced Triggers is the Version of the trigger

Transition Variables and Correlation Names

- A *transition variable* is a variable that references a value in the SQL for a triggered action. This variable is in the form of R.C (ROW.COLUMN).
- A *correlation name* is the row name in the transition variable.
- Two specialized aliases are available only inside of triggers (NEW and OLD).
- Each trigger can have one NEW view of the table and one OLD view of the table, to which the trigger is attached.
 - When an **INSERT** occurs, the NEW table contains the rows that were just inserted into the table
 - When a **DELETE** occurs, the OLD table contains the rows that were just deleted from the table
 - After an **UPDATE**, the NEW table contains the new values for the rows that were just updated in the table; the OLD table contains the old values for the updated rows. An **UPDATE** statement logically functions as a DELETE followed by an INSERT.

EXAMPLE

```
CREATE TRIGGER raise_limit
AFTER UPDATE OF salary ON employee
REFERENCING OLD AS oldrow
NEW AS newrow
FOR EACH ROW MODE DB2SQL
WHEN (newrow.salary > 1.1 *
oldrow.salary)
SIGNAL SQLSTATE '75000' ('Salary
increase > 10%');
```

Site Referred – <https://www.columbia.edu/sec/acis/db2/db2help/db2h2961.htm>

25

Example:

You want a triggered action to invoke an error message if a dollar amount in the SALARY column of table EMPLOYEE is updated to a figure that's over 110% of the amount. The action needs to reference two values: the original dollar amount and the figure to which it will be updated.

As you can see, we have defined the REFERENCING clause with OLD and NEW. We will give the OLD reference a correlation ID of oldrow, and the NEW reference has a correlation ID of newrow.

Please note that these correlation IDs are only valid for this trigger.

To reference the first value, you can specify a variable called OLDROW.SALARY, where OLDROW refers to the row that contains the value. To reference the second value, you can specify a variable called NEWROW.SALARY, where NEWROW refers to the row that contains the value.

INSERTS have NEW variables available to it

DELETE has the OLD variables available

UPDATE has both. The values before and after the update occurred

Basic Trigger Options

```
DEMO ALTER ----- Basic Trigger Options -----  
Command ==>  
  
WORKID . . . . : MVSPXW.BASICTRG  
                                     New Values  
TR Schema: DEM0323A  
TR Name  : TRIG_FOR_FUNC14  
TR Owner : RDANXA  
  
Secure . . . . : N . . . . (N,Y)  
  
Commands: HELP END CANCEL PF4=ZOOM
```

Advanced Trigger Options (1|4)

```

DEMO ALTER ----- Advanced Trigger Options -----
Command ==>

WORKID . . . . . MVSPXW.ADVTRG03                               New Values

TR Schema: DBPXW05
TR Name  : TR_NEW01
TR Owner : MVSPXW1
TR Ver   : V1

Active . . . . . : Y . . . . . (Y,N)
Concur Acc Res . . . . . : U . . . . . U ( ? FOR LIST )
Current Data . . . . . : N . . . . . (N,Y)
Encoding Scheme . . . . . : EBCDIC . . . . . (<DEFLT>,ASCII,EBCDIC,UNICODE)
Explain . . . . . : N . . . . . (N,Y)
Immediate Write . . . . . : N . . . . . (N,Y)
Isolation Level . . . . . : CS . . . . . (CS,RS,RR,UR)
Date Format . . . . . : EUR . . . . . EUR ( ? FOR LIST, BLANK=<DEFLT> )
Decimal Length . . . . . : 15 . . . . . 15 (<DEFLT>,15,31)
Scale . . . . . : 0 . . . . . 0 (0-9)
Time Format . . . . . : USA . . . . . USA ( ? FOR LIST, BLANK=<DEFLT> )
For Update Clause . . . . . : R . . . . . (R=REQUIRED,O=OPTIONAL)
Secure . . . . . : N . . . . . (N,Y)
Debug Mode . . . . . : DISALLOW . . . . . DISALLOW ( ? FOR LIST )
Asotime . . . . . : 0 . . . . . 0 (0=NO LIMIT,1-2147483647)
Dynamic Rules . . . . . : RUN . . . . . ( ? FOR LIST )
Release At . . . . . : COMMIT . . . . . (COMMIT,DEALLOCATE)
Rounding . . . . . : HALF_EVEN . . . . . HALF_EVEN ( ? FOR LIST )
Bus Time Sensitive: Y . . . . . (Y,N)
Sys Time Sensitive: Y . . . . . (Y,N)
Archive Sensitive : Y . . . . . (Y,N)
Appcompst . . . . . : V12R1M500 . . . . . V12R1M500 (Vn0R1, BLANK=<DEFLT> )
Concentrate Stmt : N . . . . . (N,Y)
WLMEnv . . . . . : DEMOWLM . . . . . DEMOWLM
Qualifier . . . . . : MVSPXW . . . . . MVSPXW
Opt Hint :

```

Advanced Trigger Options (2 | 4)

Trigger Options	Trigger Option Description
Trigger Schema	Trigger schema
Trigger Name	Trigger name
Trigger Owner	Trigger owner
Trigger Version	The version of the trigger
Active	Indicates whether the version is the active version (Y, N)
Defer Prepare	Indicates whether to defer the preparation of dynamic SQL statements that refer to remote objects, or to prepare them
Concur Acc Res	Indicates whether processing uses only committed data or waits for updated data (<DEFLT>, W=WAITFOR, U=USECURRcommitted)
Current Data	Indicates whether to require data currency for cursors (N, Y)
Encoding Scheme	The default encoding scheme for SQL variables (<DEFLT>, ASCII, EBCDIC, UNICODE)
Explain	Indicates whether to provide information on how SQL statements in the trigger execute (N, Y)
Immediate Write	Indicates whether to write immediately for updates (N, Y)
Isolation Level	Specifies how far to isolate the trigger from the effects of other running applications (CS=Cursor Stability, RS=Read Stability, RR=Repeatable Read, UR=Uncommitted Read)
Date Format	The date format - ISO, EUR, USA, JIS, or LOCAL
Decimal length	The maximum precision for decimals (<DEFLT>, 15, 31)
Scale	The scale for the decimal value (0-9)

28

WITH EXPLAIN or WITHOUT EXPLAIN –

Specifies whether information will be provided about how SQL statements in the trigger will execute.

DATE FORMAT - ISO, EUR, USA, JIS, or LOCAL

ISO - International Standards Organization

USA - IBM® USA standard

EUR - IBM European standard

JIS - Japanese industrial standard Christian era

LOCAL - Installation-defined

<https://www.ibm.com/docs/en/db2-for-zos/13?topic=values-string-representations-datetime>

ISOLATION LEVEL –

RR (Specifies repeatable read.)

RS (Specifies read stability.)

CS (Specifies cursor stability.) - CS is the default

UR (Specifies uncommitted read)

CONCURRENT ACCESS RESOLUTION –

Specifies whether processing uses only committed data or whether it will wait for commit or rollback of data that is in the process of being updated.

IBM Site Reference for all options –

<https://www.ibm.com/docs/en/db2-for-zos/13?topic=statements-create-trigger-advanced>

Advanced Trigger Options (3 | 4)

Trigger Options	Trigger Option Description
Time Format	The time format - ISO, EUR, USA, JIS, or LOCAL
FOR UPDATE CLAUSE OPTIONAL or FOR UPDATE CLAUSE REQUIRED	Specifies whether the FOR UPDATE clause is required for a DECLARE CURSOR statement if the cursor is to be used to perform positioned updates.
Secure	Indicates if the trigger is secured (N, Y).
Debug Mode	Specifies whether this version of the trigger can be run in debugging mode. ALLOW DEBUG MODE, DISALLOW DEBUG MODE, or DISABLE DEBUG MODE
Asutime	The asutime for the trigger (0=no limit, value between 1 and 2147483647)
Dynamic Rules	The values that apply for dynamic SQL attributes - RUN and BIND
Release At	The time to release resources that the trigger uses (COMMIT, DEALLOCATE)
Rounding	The rounding mode for DECFLOAT data - DEC_ROUND_CEILING - Specifies numbers are rounded towards positive infinity. DEC_ROUND_DOWN - Specifies numbers are rounded towards 0 (truncation). DEC_ROUND_FLOOR - Specifies numbers are rounded towards negative infinity. DEC_ROUND_HALF_DOWN - Specifies numbers are rounded to nearest; if equidistant, round down. DEC_ROUND_HALF_EVEN - Specifies numbers are rounded to nearest; if equidistant, round so that the final digit is even. DEC_ROUND_HALF_UP - Specifies numbers are rounded to nearest; if equidistant, round up. DEC_ROUND_UP - Specifies numbers are rounded away from 0.

29

TIME FORMAT ISO, EUR, USA, JIS, or LOCAL –

<https://www.ibm.com/docs/en/db2-for-zos/13?topic=values-string-representations-datetime>

FOR UPDATE CLAUSE OPTIONAL or FOR UPDATE CLAUSE REQUIRED (Default) –

Specifies whether the FOR UPDATE clause is required for a DECLARE CURSOR statement if the cursor is to be used to perform positioned updates.

ASUTIME –

Specifies the total amount of processor time, in CPU service units, that a single invocation of this version of the trigger can run. When you are debugging a trigger, setting a limit can be helpful in case the trigger gets caught in a loop.

RELEASE AT (DEALLOCATE / COMMIT) – COMMIT is the default.

Specifies when to release resources that the trigger uses: either at each commit point or when the trigger terminates.

Advanced Trigger Options (4 | 4)

Trigger Options	Trigger Option Description
Bus Time Sensitive	Determines whether references to application-period temporal tables in both static and dynamic SQL statements are affected by the value of the <code>CURRENT TEMPORAL BUSINESS_TIME</code> special register (Y/N).
Sys Time Sensitive	Determines whether references to system-period temporal tables in both static and dynamic SQL statements are affected by the value of the <code>CURRENT TEMPORAL SYSTEM_TIME</code> special register (Y/N).
Archive Sensitive	Determines whether references to archive-enabled tables in SQL statements are affected by the value of the <code>SYSIBMADM.GET_ARCHIVE</code> global variable (Y/N).
Applcompat	The bind option for the package associated with the trigger. Valid values are <code>VvvR1Mmmm</code> , <code>VvvR1</code> , or <code><DEFLT></code> . * <code>VvvR1Mmmm</code> (where <code>vv</code> is the Db2 version number and <code>mmm</code> is the Db2 maintenance release number) * <code>VvvR1</code> (where <code>vv</code> is the Db2 version number). * <code><DEFLT></code> specifies that the default value from the ZPARM will be used, and the default values for Bus Time Sensitive, Sys Time Sensitive, and Archive Sensitive will be used.
Concentrate Stmt	Indicates whether a dynamic SQL statement that specifies literal constants will be cached as a separate unique statement entry in the dynamic statement cache instead of sharing an existing statement in the cache (N, Y).
WLMEnv	The name of the WLM environment
Qualfier	The implicit qualifier for unqualified names of objects
Opt Hint	The query optimization hint character string

30

BUSINESS_TIME SENSITIVE – YES (Default) or NO

Determines whether references to application-period temporal tables in both static and dynamic SQL statements are affected by the value of the `CURRENT TEMPORAL BUSINESS_TIME` special register.

SYSTEM_TIME SENSITIVE – YES (Default) or NO

Determines whether references to system-period temporal tables in both static and dynamic SQL statements are affected by the value of the `CURRENT TEMPORAL SYSTEM_TIME` special register.

APPLCOMPAT *applcompat-level* – The default value is V12R1M500.

Specifies the application compatibility level behavior for static SQL statements in the trigger body.

Trigger Order and Versioning



You can have multiple triggers and trigger versions on the same table. Db2 records the timestamp, and the triggers and trigger versions are activated in the order in which they were created.



If the trigger is being dropped due to any alters to the trigger options, you need to ensure that the order of the versions is maintained. The order based on the creation timestamp can impact the firing order sequence.

Trigger Name and Version	OLD Created Timestamp	New - After Update Created Timestamp	OLD Trigger Order Seq	NEW Trigger Order Seq
TRIGGER1_TB1.VER3	2022-07-25-03.03.14.037256	2023-10-08-04.58.07.824041	1	2
TRIGGER2_TB1.VERSION1	2022-07-25-03.03.14.043146	2022-07-25-03.03.14.043146	2	1
TRIGGER2_TB1.VERSION2	2022-07-25-03.03.14.048613	2022-07-25-03.03.14.048613		

31

Multiple triggers:

Multiple triggers that have the same activation time and triggering event can be defined on a table. The triggers are activated in the order in which they were created. For example, the trigger that was created first is executed first; the trigger that was created second is executed second. If the OR REPLACE option is used to replace an existing trigger, the create time is changed and therefore might affect the order of trigger execution.

IBM Site Reference – <https://www.ibm.com/docs/en/db2-for-zos/13?topic=values-string-representations-datetime>

Trigger Cascading



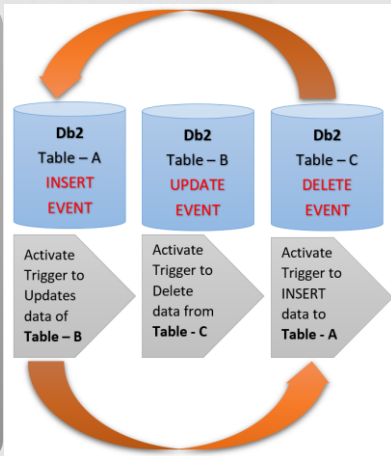
Trigger cascading is the result of the activation of one trigger that executes SQL statements that cause the activation of other triggers or even the same trigger again.



Db2 limits this cascading effect to 16 levels to prevent endless looping. If more than 16 levels of nesting occur, the transaction is aborted.



Image shows best cascading example of triggers which can lead to potential looping problems.



IBM Site Referred – <https://www.ibm.com/docs/en/db2-for-zos/13?topic=values-string-representations-datetime>

If a trigger at the 17th level is activated, Db2 returns SQLCODE -724 and backs out all SQL changes in the 16 levels of cascading. However, as with any other SQL error that occurs during trigger execution, if any action occurs that is outside the control of Db2, that action is not backed out.

Trigger Restrictions



You cannot create a basic or an advanced trigger on MQT, clone, temporary, auxiliary, accelerator only, and directory tables. In addition, you cannot create triggers on aliases, synonyms, or real-time statistics.



You cannot create triggers on views that have LOB, XML, ROWID, identity, security label, row change timestamp, row begin, row end, and transaction start ID columns.



You can nest up to 16 levels of triggers.

Trigger Considerations



Performance Impact:

Triggers can introduce performance overhead, especially if they involve complex logic, queries, or updates. Careful design and testing are required to ensure that triggers do not significantly impact database performance.



Privileges:

Triggers execute with the privileges of the user who defined the trigger, not the privileges of the user who triggered the action. This can affect the data that the trigger can access.



Transaction Control:

Triggers operate within the context of the transaction that caused them to fire. Changes made by triggers are part of the same transaction and are committed or rolled back together with the main transaction.



Isolation Levels:

Be mindful of the isolation level of transactions that use triggers. Triggers can influence the locking behavior and concurrency control of the database.



Table Alterations:

Some alterations to a table, such as adding or dropping columns, can impact triggers. Dropping a column referenced in a trigger will cause the trigger to be invalidated. Adding columns might require modifying triggers to account for the new columns – trigger is based upon or references.

Advanced Trigger – an Example

- SQL/PL statements set values to variables and show
 - ✓ DECLARE statements
 - ✓ Loops
 - ✓ Cursors
 - ✓ INSERT INTO



**Example of an
advanced trigger**

References -

- <https://www.ibm.com/docs/en/db2-for-zos/12?topic=release-advanced-trigger-support>
- <https://robertsdb2blog.blogspot.com/2017/08/db2-12-for-zos-sql-enhancements.html> – Robert Catterall
- <https://www.craigsmullins.com/triggers.htm> - Craig S. Mullins
- <https://db2portal.blogspot.com/2017/02/the-db2-12-for-zos-blog-series-part-2.html> - Craig S. Mullins



If you have any queries, feel free to contact me at –

frank_Rhodes@bmc.com

Thank you

Speaker Biography

Frank Rhodes

Solution Architect, DB2 Administration Products

Frank Rhodes started with BMC Software in 1995. In his current role he works to promote cross product coordination between the Database Administration products and the other BMC DB2 products. The products that he has primary responsibility for are ALTER for DB2, Change Manager for DB2, Catalog Manager for DB2, DASD Manager Plus for DB2, Command Center for DB2 as well as BMC AMI DevOps for Db2.

Frank was a Developer on the Change Manager for DB2 product for over 10 years. His main areas of responsibilities being the development and maintenance for the Analysis Engine as well as the product's user interface. Prior to joining the Change Manager Product

team, he worked on the TIS Install for the Administrative products. He was the lead developer in charge of the initial implementation of a common install for all the DB2 products.

Prior to BMC, Frank worked as a Systems Programmer for IBM in support of NASA JSC's administrative processors. He supported installation and maintenance of MVS products and oversaw implementing Automated Operation for IBM's internal systems as well as NASA's systems.

Frank holds a Bachelor's Degree in Computer Science from the Texas A & M University.

Jaspreet Kaur – Jaspreet holds 18+ years of IT experience and was an Application Programmer in Mainframes and DB2. She was an Associate Project Manager in Pune, Maharashtra, before joining BMC. For the last 3 years, Jaspreet has been working with BMC Pune office, as Senior Product Developer on DB2 Change Manager product.

Pradip Wagh - Pradip holds 14+ years of IT experience, and he played various roles like a Lead Application Programmer, Business and System Analyst in Mainframes and DB2. He was Lead Developer in Pune, Maharashtra, before joining BMC. For the last 2.5 years, Pradip has been working with BMC Pune office as Senior Product Developer on DB2 Change Manager product.