IBM GLOBAL SERVICES

Z12

SQL Triggers in DB2 for z/OS and OS/390

Jay Yothers

IBM DB2 Information Management Technical Conference

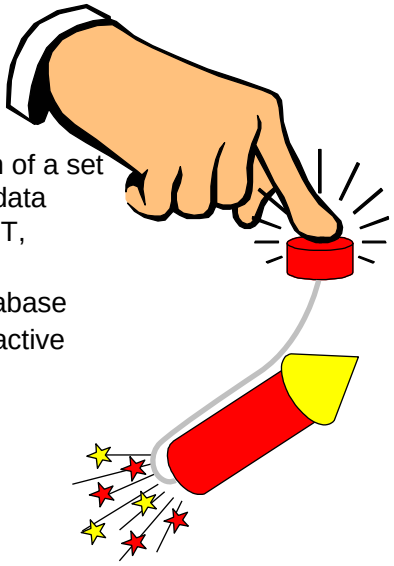
Sept. 20-24, 2004

Las Vegas, NV

Agenda

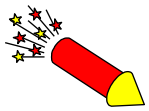
- Trigger Description
- Trigger Granularity
- Triggered Actions
- Raising Errors
- Accessing Modified Data

Triggers Overview



- Triggers provide automatic execution of a set of SQL statements when a specific data change operation (UPDATE, INSERT, DELETE) occurs
  - Bring application logic into the database
  - Transform DB2 from a passive to active DBMS
- Benefits of triggers include
  - Code reuse
  - Faster application development
  - Easier maintenance

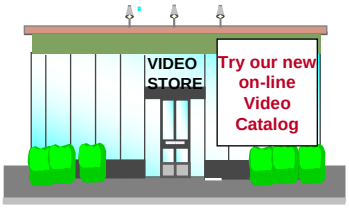
Common Uses for Triggers



- **Enforce business rules based on changing conditions**
- **Validate input data**
- **Generate new values for inserted / updated rows**
- **Cross-reference other tables**
- **Maintain audit, summary or mirror data in other tables**
- **Support "alerts"**
  - **E-mail notification**
  - **Initiate external actions**

Trigger Flow

Insert 'Z-Files' row into Video_Table



Video_Table

Title	Code	Price
Toy Glory	C28	14.95
Star Trak	S31	14.95
Indiana Bones	A07	15.95
Sixth Cents	R67	19.95
Z-Files	S31	25.95

Category_Table

Category	CatNo	Cat_Total
Adventure	A07	1
Comedy	C28	1
Drama	R67	1
Science Fiction	S31	2

AFTER TRIGGER

Update Category_Table
Set Cat_Total = Cat_Total + 1
Where CatNo = new.Code

More on Trigger Flow

Henry	44	25400	000

Inserts
Updates
Deletes

integrity constraint checking

Name	Age	Salary	Tax
Jones	23	10040	A03
Smith	56	2043	A05
Fred	23	1450	A04
Johns	12	1970	B11
Henry	44	2540	A05

Before

```
Case
    When Salary <= 10000 Then Tax = A03
    When Salary <= 14000 Then Tax = A04
    When Salary <= 19000 Then Tax = B11
    When Salary <= 20000 Then Tax = A00
    Else Tax = A05
End
```

After

```
Update Tax_Table
    Set Tax_Count = Tax_Count + 1
    Where Tax_Level = new.Tax
```

Tax_Level	Tax_Count
A03	1
A04	1
A05	2
B11	1

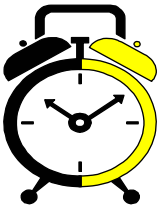
Trigger Characteristics

```
CREATE TRIGGER Payroll
    AFTER UPDATE OF salary ON Paytable
    FOR EACH STATEMENT MODE DB2SQL
    VALUES(PAYROLL_LOG(User, 'UPDATE',
    CURRENT  TIME, CURRENT DATE));
```

- Trigger Name: Limited to 8 characters in V7, 128 in V8
- Triggering Table: Table on which the trigger is defined
- Triggering Event:
  - An SQL Data Change Operation (INSERT,DELETE,UPDATE)
    - UPDATE can be qualified by column
  - ON the triggering table
- Trigger Activation Time: BEFORE or AFTER
- Trigger Granularity: for each row or for each statement

Trigger Activation Time

```
CREATE TRIGGER Purchase
    NO CASCADE BEFORE INSERT ON Order
    REFERENCING NEW AS New_Order
    FOR EACH ROW
    MODE DB2SQL
    SET New_Order.Date = CURRENT_DATE;
```



- ***BEFORE***
  - Evaluated entirely before triggering event
  - Can be considered an extension of the constraint system
    - Prevent invalid update operations
  - Useful for conditioning of input data
    - Validate or directly modify input values
  - SET allows you to modify values of affected rows
    - No UPDATE, INSERT, or DELETE statements in BEFORE trigger body

Trigger Activation Time

```
CREATE TRIGGER Purchase
    AFTER INSERT ON Order
    FOR EACH STATEMENT
    MODE DB2SQL
    CALL E-MAIL_CONFIRMATION;
```

Beep!

- ***AFTER***
  - Evaluated entirely after the triggering event
  - Can be considered an encapsulation of application logic that normally would be performed by the updating application
  - Perform audit trail logging or maintain summary data
  - Perform actions outside the database such as writing to an external data set or sending an e-mail message

Trigger Granularity

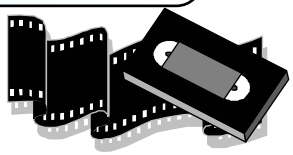
```
CREATE TRIGGER AddOrder
    NO CASCADE
    BEFORE INSERT ON Order
    REFERENCING NEW AS NewRow
    FOR EACH ROW MODE DB2SQL
    SET NewRow.Date = CURRENT_DATE;
```

```
CREATE TRIGGER Purchase
    AFTER INSERT ON Order
    FOR EACH STATEMENT
    MODE DB2SQL
    CALL E-MAIL_CONFIRMATION;
```

- Granularity controls how many times the trigger is executed
  - **FOR EACH ROW**: Executed once for each row modified by the triggering event
    - Referred to as a row trigger or a row-level trigger
  - **FOR EACH STATEMENT**: Executed once each time the triggering SQL statement is issued
    - Referred to as a statement trigger or a statement-level trigger

Triggered Action Condition

```
CREATE TRIGGER ReOrder
    AFTER UPDATE OF InStock ON Video_Table
    REFERENCING NEW AS N
    FOR EACH ROW MODE DB2SQL
    WHEN (N.InStock < 0.10 * N.MaxStock)
        CALL ORDER_VIDEO(N.MaxStock - N.InStock, N.Video_Num);
```



- **Triggered Action Condition**
  - **Optional**
  - **In the form of a WHEN clause (similar syntax to a WHERE clause)**
  - **Trigger will not fire if WHEN clause not satisfied**

Triggered SQL Statements

```
CREATE TRIGGER AddVideo
    AFTER INSERT ON Video_Table
    REFERENCING NEW AS Newrow
    FOR EACH ROW MODE DB2SQL
    BEGIN ATOMIC
      UPDATE Item_Table SET Item_cnt = Item_cnt + 1
          WHERE ItemNo = Newrow.ItemNo ;
      CALL E_MAIL_CUSTOMERS ;
    END !
```

- **Triggered SQL Statements**
  - **One or more SQL statements that are executed if WHEN clause evaluates true**
  - **Multiple statements are enclosed in BEGIN ATOMIC...END and delimited with semicolons**
    - **Use statement delimiter (!) for DSNTEP2, DSNTIAD, and SPUFI**
  - **Can include stored procedure call and functions**
  - **If trigger fails, invoking statement fails**

Statements Allowed as Triggered SQL

- Allowed in both BEFORE and AFTER triggers:
  - CALL stored-procedure
  - VALUES (expression, expression,...)
    - Normally used to invoke a user-defined function
  - SELECT
    - Used to invoke user-defined functions
  - SIGNAL SQLSTATE statement
- Allowed only in BEFORE triggers:
  - SET transition variable
- Allowed only in AFTER triggers:
  - INSERT
  - Searched UPDATE (not a cursor UPDATE)
  - Searched DELETE (not a cursor DELETE)
  - All modifications are part of triggerring statement's unit of recovery

Invoking UDFs and Stored Procedures

3 ways from within a trigger body

```
1.VALUES(UDF1(NEW.COL1),UDF2(NEW.COL2);
2.SELECT UDF1(COL1), UDF2(COL2)
        FROM NEW_TABLE
        WHERE COL1 > COL3;
3.CALL StorProc(NEW.COL1, NEW.COL2);
```

- Triggers can only perform SQL operations
- Ability to invoke stored procedures and user-defined functions expands types of possible triggered actions to include:
  - Conditional logic and looping
  - Initiation of external actions
  - Access to non-DB2 resources, including remote databases
- User-defined functions cannot be invoked as a standalone call
  - Must be part of an expression in an SQL statement

Raising Error Conditions

```
CREATE TRIGGER Creditck
    AFTER UPDATE OF Balance ON Customer
    REFERENCING NEW AS Newrow
    FOR EACH ROW MODE DB2SQL
    WHEN (Newrow.Balance > Newrow.CreditLimit)
        SIGNAL SQLSTATE '75001' ('Credit Limit Exceeded -
        Shred Card');
```

- **Triggers can be used for stopping invalid updates and for detecting other invalid conditions.**
  - **SIGNAL SQLSTATE - New SQL statement that halts processing and returns the requested SQLSTATE and message to the application. Format:**

    SIGNAL SQLSTATE sqlstate-string-constant (diagnostic-string-constant)
    - **Only valid in triggered actions**

Transition Variables

```
CREATE TRIGGER Increase
    BEFORE UPDATE OF Salary_Table ON Employee
    REFERENCING OLD AS Oldrow
                NEW AS Newrow
    FOR EACH ROW MODE DB2SQL
    WHEN (Newrow.Salary > Oldrow.Salary * 1.20)
    SET Newrow.Salary = Oldrow.Salary * 1.20;
```

- **Transition Variables:**
  - **Contain column values of row affected by triggering operation**
  - **REFERENCING clause enables a correlation name to be assigned to the before and after states of the row**
    - **OLD AS Oldrow: Value of row before triggering SQL operation**
    - **NEW AS Newrow: Value of row after triggering SQL operation**

Transition Tables

```
CREATE TRIGGER Large_Order
  AFTER INSERT ON Invoice
  REFERENCING NEW_TABLE AS N_Table
  FOR EACH STATEMENT MODE DB2SQL
  SELECT
      LARGE_ORDER_ALERT(Cust_No, Total_Price, Delivery_Date)
          FROM N_Table WHERE Total_Price > 10000
```

- **Transition Tables:**
  - **Contains entire set of rows affected by triggering operation**
  - **Apply aggregations over the set of affected rows (MAX, MIN, AVG)**
  - **REFERENCING clause specifies a table identifier**
    - **OLD_TABLE AS identifier: Table of BEFORE values**
    - **NEW_TABLE AS identifer: Table of AFTER values**
  - **Only valid for AFTER triggers**
  - **Can be referenced from invoked stored procedure or UDF**

Accessing trigger transition table

- Trigger transition table is the set of changed rows that the triggering SQL statement modifies
- Trigger can invoke UDF or stored procedure, and that UDF or stored procedure can refer to values in the transition table
  - Use ***table locators***

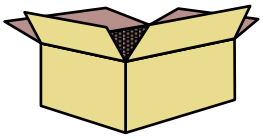
```
CREATE TRIGGER EMPRAISE
  AFTER UPDATE ON EMP
REFERENCING NEW_TABLE AS NEWEMPS
FOR EACH STATEMENT MODE DB2SQL
BEGIN ATOMIC
    VALUES (CHECKEMP(TABLE NEWEMPS));
```

```
CREATE FUNCTION CHECKEMP(TABLE LIKE EMP AS LOCATOR)
  RETURNS INTEGER
  EXTERNAL NAME 'CHECKEMP'
  PARAMETER STYLE SQL
  LANGUAGE C;
```

Valid Trigger Characteristic Combinations

Granularity	Activation Time	Triggering Operation	Transition Variables Allowed	Transition Tables Allowed
ROW	BEFORE	INSERT	NEW	NONE
		UPDATE	OLD, NEW	
		DELETE	OLD	
	AFTER	INSERT	NEW	NEW_TABLE
		UPDATE	OLD, NEW	OLD_TABLE, NEW_TABLE
		DELETE	OLD	OLD_TABLE
STATEMENT	BEFORE	INVALID TRIGGER		
	AFTER	INSERT	NONE	NEW_TABLE
		UPDATE		OLD_TABLE, NEW_TABLE
		DELETE		OLD_TABLE

Trigger packages



- When you create a trigger, DB2 creates a *trigger package*
  - Qualifier of trigger name determines package collection
    - For static, authorization ID of QUALIFIER bind option
    - For dynamic, CURRENT SQLID
  - Trigger packages are different than regular packages
    - You cannot bind them, can rebind only locally
    - They can be rebound with new REBIND TRIGGER PACKAGE command
    - Change subset of default bind options (CURRENTDATA, EXPLAIN, FLAG, ISOLATION, RELEASE)
    - Useful for picking up new access paths
    - Trigger packages cannot be freed or dropped. To delete trigger package, use DROP TRIGGER SQL statement.
    - Trigger packages cannot be copied

Trigger Performance

- SQL statements are synchronous with the application
  - All statements issued by a Trigger execute as part of the triggering statement
- After Trigger Transition Tables
  - Prior to V8, always placed in a work file
    - Even for a conditional trigger with a false condition
  - In V8, up to 4K is placed in memory