Sequences

- Using Sequences

  SELECT NEXT VALUE FOR MYSEQ;

- Using sequence in the insert statement.

  INSERT INTO CANDIDATE VALUES (NEXT VALUE FOR MYSEQ,'AJAY');

- Procedure using sequence to generate the candidate ID and insert the data in table.

```
CREATE PROCEDURE ADDCANDIDATE (@N AS VARCHAR(50))
AS
BEGIN
    DECLARE @A AS INT;
    DECLARE @C AS CHAR(5);
    SET @A = ( NEXT VALUE FOR MYSEQ);
    IF @A <10
            SET @C = CONCAT('C00' , @A);
    ELSE IF @A<100
            SET @C = CONCAT('C0' , @A);
    ELSE IF @A<1000
            SET @C = CONCAT('C' , @A);

    INSERT INTO  CANDIDATE VALUES (@C, @N);
END;
```

Auto Generation of ID Using Sequence

Function to generate a Alpha Numeric ID

```
CREATE FUNCTION GENID (@C CHAR (1) , @I INT)

RETURNS CHAR(5)

AS

BEGIN

    DECLARE @r  CHAR(5);

    DECLARE @ID  CHAR(5);

SELECT @R = CASE

                    WHEN @I < 10 THEN CONCAT(@C,'000')

                    WHEN @I < 100 THEN CONCAT(@C,'00')

                    WHEN @I < 1000 THEN CONCAT(@C,'0')

                    WHEN @I < 10000 THEN @C

                    ELSE 'NULL'

            END;

 SET @ID= RTRIM(@R) + LTRIM(CONVERT(CHAR(4),@I));

    RETURN @ID;

END;
```

Auto Generation of ID Using Sequence

Using user defined function with a sequence in a procedure to add an student in to the table:

CREATE PROCEDURE ADDSTU @X CHAR(20)

AS

BEGIN

   SET NOCOUNT ON;

   INSERT INTO STU

   VALUES(DBO.GENID('S',NEXT VALUE FOR MYSEQ),@X);

   SELECT * FROM STU;

   END;

ASSIGNMENT

ASSIGNMENT – 9

A-1 : CREATE A FUNCTION FOR AUTOGENERATION OF 5 CHARACTERS ALPHA NUMERIC ID. IT SHOULD ACCEPT 2 PARAMETERS A CHARACTER AND THE NUMBER AND RETURN THE ID BY CONCANATING THE CHARACTER , REQUIRED ZEROS AND THE SPECIFIED NUMBER.

RECREATE BELOW PROCEDURES IN THE INVENTORY DATABASE AS SPECIFIED (ALL THE ID s SHOULD BE AUTOMATICALLY GENERATED USING ABOVE CREATED FUNCTION AND SEQUENCES) :

ADDSUPPLIER – SHOULD ADD THE SUPPLIER IN THE SUPLIER TABLE AND DISPLAY THE DETAILS OF THE NEW SUPPLIER ADDED.

ADDPRO – SHOULD ADD THE PRODUCT IN THE PRODUCT TABLE AND DISPLAY THE DETAILS OF THE NEW PRODUCT ADDED.
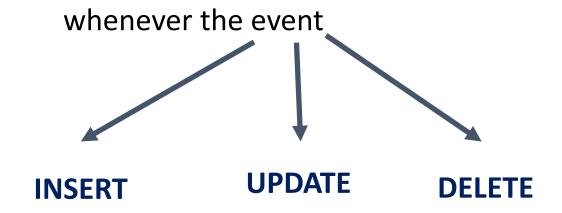
ADDCUST – SHOULD ADD THE CUSTOMER IN THE CUSTOMER TABLE AND DISPLAY THE DETAILS OF THE NEW CUSTOMER ADDED.

ADDORDER – SHOULD ADD THE ORDER IN THE ORDERS TABLE AND DISPLAY THE DETAILS OF THE ORDER. ORDER DATE SHOULD BE CURRENT DATE AND SHOULD COME AUTOMATICALLY.

TRIGGERS

Triggers

Triggers are the ***special kind*** of stored procedures

which get automatically executed

whenever the event

INSERT   **UPDATE**   **DELETE**

on the under ling table occurs.

Triggers

A trigger is a database object that is attached to a table. Triggers are often referred to as a "special kind of stored procedure". The main difference between a trigger and a stored procedure is that the trigger is attached to a table and is only fired when an INSERT, UPDATE or DELETE occurs. You specify the modification action(s) that fire the trigger when it is created.

- **Syntax**

 CREATE TRIGGER trigger_EID

ON table_EID

FOR INSERT|UPDATE |DELETE

AS

BEGIN

   SQL Statements;

END;

Triggers

Example 1: Trigger to update the stock when product is sold.

```
CREATE TRIGGER TR_INVENT_UPDATE

ON SALES

FOR INSERT

AS

BEGIN

UPDATE INVENT SET StockQty = StockQty- (SELECT QTY FROM INSERTED
    )

WHERE PID = (SELECT PID FROM INSERTED);

END;
```

Triggers

Example 2: Trigger to delete the order if the product Is deleted from the inventory.

CREATE TRIGGER TR_SALE_DELETE

ON INVENT

FOR DELETE

AS

BEGIN

DELETE FROM SALES WHERE PID = (SELECT PID FROM **DELETED**);

END;

Triggers

Example 3: Trigger to update the stock when the order quantity has been updated.

CREATE TRIGGER TR_STOCK_UPDATE2

ON SALES

FOR UPDATE

AS

BEGIN

UPDATE Stock SET SQty = SQty + (SELECT QTY FROM DELETED)

WHERE PID = (SELECT PID FROM DELETED);

UPDATE Stock SET SQty = SQty - (SELECT QTY FROM INSERTED)

WHERE PID = (SELECT PID FROM INSERTED);

END;

Triggers

Example 4: Trigger to check & update the stock when the order is placed

```
CREATE TRIGGER TR_INVENT_CHECK
ON SALES
FOR INSERT
AS
BEGIN
    DECLARE @QS AS INT;
    DECLARE @QR AS INT;
    SET @QR= ( SELECT QTY FROM INSERTED);
    SET @QS = (SELECT StockQty FROM INVENT WHERE PID=(SELECT PID FROM inserted));
    IF @QS >= @QR
            Begin
                UPDATE INVENT SET StockQty = StockQty- (SELECT QTY FROM INSERTED )
                 WHERE PID = (SELECT PID FROM INSERTED);
                 COMMIT;
            end
    ELSE
            ROLLBACK;
');
```

Thanks!

EVERY *ENDING*
IS *REALLY JUST A*
NEW BEGINNING

Rajeev Garg
Data Analytics Trainer
9899245970