TRIGGERS are parameter less Stored Procedures that are invoked automatically by Database INSERT, UPDATE, and DELETE or (I,U,D) statements on tables. In other words triggering events are Data Manipulation Language (DML) using (I,U,D) operations on tables. Triggers are used to implement application specific business rules using procedural constraints.

A trigger can be associated with only one database table. The executions of triggers are transparent to the user. The database administrator should use triggers to enforce rules that cannot be coded through declarative constraints. Triggers may be implemented at three different levels of abstraction; Action, Level and Timing. These abstractions are also referred to as trigger characteristics.

• Action Triggers

Action triggers refers to triggering the stored procedure on the execution the SQL statements INSERT, UPDATE, or DELETE. A trigger can be designed to fire on any combination of these statements.

• Level triggering

- Level triggering refers to either statement-level or row-level triggering, statement level is the default. A Statement level triggers fire once per SQL statement while the row level trigger fire many times, once for each row affected by the data manipulation statement. Each type of level has its own unique set of constraints. Statement level triggers cannot read the values of the columns. The row level trigger can evaluate the values of each column for that row: the old value before modification and new value after modification, the column values are known as correlation values

- Decision criteria for selecting the Level Type:

(1) If there is a need to work on each row at a time, then select row level

(2) If there is a need to use both the new value and the old value, then select row level

(3) If you can decide IC violation by single row, it is row-level

(4) If you can determine a violation only by reading multiple rows or all rows, then select statement level trigger.

(5) If the syntax contains the FOR EACH ROW clause, then it should be a row-level trigger.

• **Timing**

- Trigger timing refers to whether the trigger fires BEFORE or AFTER the triggering action. When creating a trigger you need to specify four pieces of information:

- The unique trigger name
- The table to which the trigger is to be associated
- The action that the trigger should respond to (DELETE, INSERT, or UPDATE)
- When the trigger should be executed (before or after processing)

The database administrator should attempt to keep Trigger Names Unique per Database. MySQL is more flexible than other Database Management Systems (DBMS), in MySQL trigger names are only restricted to be unique per table, not per database as in many other industrial strength (DBMS). This means that two different tables in the same database can have triggers of the same name. This is not allowed in other DBMSs where trigger names must be unique per database. MySQL may make the naming rules stricter in a future release. Is this should occur it will be a good practice to establish a database-wide unique trigger name system now.

Triggers are created using the CREATE TRIGGER statement. Here is a really simple example:

• Input

```
CREATE TRIGGER newproduct AFTER INSERT ON products
FOR EACH ROW SELECT 'Product added';
```

• Analysis

CREATE TRIGGER is used to create the new trigger named newproduct triggers can be executed before or after an operation occurs, and here AFTER INSERT is specified so the trigger will execute after a successful INSERT statement has been executed. The trigger then specifies FOR EACH ROW and the code to be executed for each inserted row. In this example, the text Product added will be displayed once for each row inserted.

To test this trigger, use the I**NSERT** statement to add one or more rows to products; you'll see the Product added message displayed for each successful insertion.

Note

Only Tables Triggers are only supported on tables, not on views (and not on temporary tables). Triggers are defined per time per event per table, and only one trigger per time per event per table is allowed. As such, up to six triggers are supported per table (before and after each of INSERT, UPDATE, and DELETE). A single trigger cannot be associated with multiple events or multiple tables, so if you need a trigger to be executed for both INSERT and UPDATE operations, you'll need to define two triggers.

Triggers on DELETES will not run if you delete via cascading foreign keys. On an INSERT, you can only use NEW for column values. On a DELETE, you must use OLD.

I will post another tutorial soon on the practical use of triggers in web development. I hope this helps someone.

Links to Additional References

http://www.rustyrazorblade.com/2006/09/14/mysql-triggers-tutorial/