Database Triggers.

1. What triggers are used for?

Triggers in Databases are used to automate a set of instructions to ensure that data consistency and integrity constraints are maintained across the DB whenever it is stored. It is used as well to automatize some tasks when some conditions are met. It is executed as a side effect if a database is modified. They are normally activated after a specific event, normally through a command issued in the Data Manipulation Language (DML) of the DB. As most DML use at least 4 basic commands (INSERT, SELECT, UPDATE and DELETE), the triggers can be used BEFORE, AFTER or INSTEAD OF these commands. It basically avoids erroneous entries and enables integrity in the DB.

Specific examples where triggers are used are: avoid invalid transactions (request validation before updating), enforce referential integrity in distributed DB, generate values automatically to populate some fields (for instance managing minimal inventory in a warehouse), prevent DML operations during certain times (e.g. out of business hours), enforce security rules log events in the DB, just to name some of them.

2. The general structure of a trigger.

The basic structure is:

a) Specify the moment when the trigger is fired. It can be decomposed into 2 steps:
   a. *An event* that activates the trigger.
   b. *Condition* to be met so the trigger is executed.
b) Actions to be executed when the trigger is fired.

Applied to a DBMS DML it is reflected in 3 steps:

a) Triggering SQL statement. As mention before, more common statements are INSERT, UPDATE, DELETE and additionally AFTER, BEFORE and INSTEAD OF the event that fires the trigger.
b) Trigger Restriction. Normally to set the frequency and scope where the trigger is applied.
c) Trigger Action. Performs a set of transactions as a consequence of the previous DML statement.

3. Description of different kinds of events and triggers.

In general, triggers can be classified in:

- *Row Level Trigger*. We have to specify when creating the trigger and its When condition.
- *Statement Level Trigger*. It is the default option. It will be executed once after the DML statement and it can be applied to one /several rows or the whole table.

Each major DBMS has its specificities. For instance, in Oracle there are 3 kinds of triggers:

- *DML triggers*. They have been formerly described: they run when a DML event is executed (normally Insert, Delete or Update). They run After, Before or Instead of when they are executed.

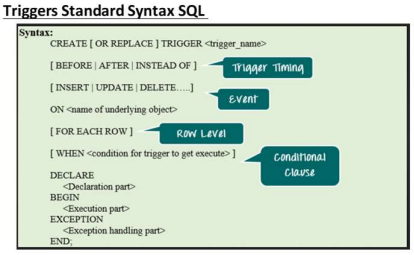
- *Schema triggers*. Fire on events that occur in objects defined within a schema, for instance: Alter, Analyse, Audit, Create, Drop, Grant, Rename, Revoke,…

- *Database triggers*: created on a Database and linked to events happening in that DB. Some examples are: After Startup, Before Shutdown, After Logon, Before Logoff, After Suspend,…

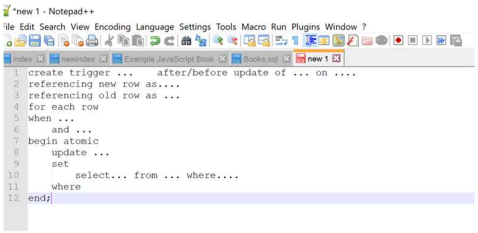
4. Databases supported by triggers.

All major DBMS support triggers but there are some differences over the SQL standard syntax. Below this standard (supported by IBM DB2 and MySQL with slight differences) is described. Other DBMS have a nonstandard syntax or may not implement all features. Below I describe some differences as well.

Triggers Standard Syntax SQL



IBM DB2 Example



Triggers in major DBMS

	Existence	Differences on SQL standard
Oracle	Yes	Keyword does not appear in referencing statement. Atomic does not appear after beguin Uses colon for the update statement. Subqueries are not allowed
Microsoft SQL Server	Yes	ON is used instead of AFTER Reference Clause is omitted Tuple used instead of New and Old rows
IBM DB2	Yes	Supports SQL standard
MySQL	Yes	Supports SQL standard

MySQL Example

```
mysql> delimiter //
mysql> CREATE TRIGGER upd_check BEFORE UPDATE ON account
       FOR EACH ROW
       BEGIN
           IF NEW.amount < 0 THEN
               SET NEW.amount = 0;
           ELSEIF NEW.amount > 100 THEN
               SET NEW.amount = 100;
           END IF;
       END;//
mysql> delimiter ;
```

5. Problems or disadvantages associated with triggers.

Triggers can be very untrusty and tricky due to the difficulties to follow the actions that they may fire. In general, it is always preferable to use relationships using foreign keys or appropriate use of stored procedures to enforce integrity rather than use to triggers. The main reason to avoid their usage is that triggers may fire other triggers, creating a 'trigger storm' with circular references.

New built-in solutions have been incorporated into new databases allow us to limit the length of these chains of triggers or to maintain replicas of databases or materialised views. They provide better solutions than to triggers.

Finally, triggers may be fired in other processes unintendedly like backups (some DB allow 'as not for replication' limits to avoid this).

In short, they can be very useful but it is always better to avoid them if proper alternatives exist.

6. References.

- *Melton, J. and Simon, A. (2002). SQL:1999. San Francisco, Calif.: Morgan Kaufmann.*
- *SILBERSCHATZ. (2019). DATABASE SYSTEM CONCEPTS: MCGRAW-HILL EDUCATION.*
- *Yarger, R., Reese, G. and King, T. (1999). MySQL & mSQL. Sebastopol (California): O'Reilly.*