An example of when/what triggers are useful?

Triggers, in dbms systems are a tool for users, such as database designers and developers, to set up procedures that should fire or trigger in reaction to some events happening to the database tables. For example, if you want to monitor whenever a database administrator is making a change to a table such as an insertion, update or deletion, you can, with the help of triggers do that. For example, in the case of an 'update', you can record the data that was present in the table before and after the update, who executed the update and a timestamp of when the action occurred. This can be stored away for you to review later. This is only one of a set of trigger types that dbms offer.

How is it done?

In the case of the type just mentioned, called DML triggers[1], this is enabled by two so called virtual tables that the SQL server stores, which capture a view of a given table before and after each time one of the following 3 actions, insert, update and delete, are performed on a table. This, with the addition of using UNION ALL, you can set a trigger to pull out the information from these virtual tables automatically, after every time one of the above actions are performed, and push it into your own customary defined table. When you set up a trigger, in addition to the typical assigning it a name, associating it with a table etc., you would also specify which tables it should pull from, the table and column it should push to and how the data should be selected or combined.

Another example of a trigger of this type is 'INSTEAD OF' which allows you to skip or rollback an insert, update or delete statement and instead do something else.

A second type of trigger is called DDL, for data definition language, they are set to respond to events at a higher level than table modifications, to database or server events such as to prevent or record changes to a schema.[2]

A third type of trigger are LOGON triggers. They are set to respond to user logins and fire in between when the user logged in and the session is started. They can be used to restrict the number of active sessions a particular login can have at once, track login activity and restrict access to an sql server[3].

To what extent do vendors support the various kinds of triggers?

Oracle

Oracle offers a fourth type of trigger, which reacts to system events such as startup, shutdown and errors. Oracle also allows you to set triggers **before** as well as after events. It also allows you to do compound triggers that react before and after changes per row as well as before and after changes per statement[4]. But as a result, some of these cannot access both images, of the virtual 'inserted' and 'deleted' tables discussed above, the before and after images, at the same time[5].

Microsoft SQL Server

Ms Sql mostly doesn't support **before** event triggers[6] (nor compound triggers) but the same can be achieved with an INSTEAD OF. Also, all DML triggers in Ms are executed as a transaction, which is good in the event the event errored[7] [8]. But notice, whereas in Oracle, for eg, the INSTEAD OF will prevent the action from happening in the first place, here this will be achieved by rolling back the action.

---

[1] https://www.sqlservertutorial.net/sql-server-triggers/
[2] https://www.sqlservertutorial.net/sql-server-triggers/sql-server-ddl-trigger/
[3] https://docs.microsoft.com/en-us/sql/relational-databases/triggers/logon-triggers?view=sql-server-ver15
[4] https://docs.oracle.com/cd/B28359_01/appdev.111/b28370/triggers.htm#LNPLS020
[5] https://www.codeproject.com/Articles/621532/Comparison-of-Triggers-in-MS-SQL-and-Oracle
[6] https://www.guru99.com/oracle-vs-sql-server.html
[7] https://docs.microsoft.com/en-us/sql/relational-databases/triggers/dml-triggers?view=sql-server-ver15
[8] https://codingsight.com/sql-server-triggers-understanding-alternatives/

IBM DB2

DB2 also offers the basic DML triggers, albeit with a few limitations. You cannot specify two triggers for the same operation and fine grained conditioning is not always allowed (eg when using INSTEAD OF or BEFORE[9]). DDL and LOGON triggers are also not supported although for the latter they are said to be offering something that isn't quite a trigger but can be set to a stored procedure.

MySQL

MySQL's support for triggers is limited and DML triggers were introduced fairly late. Although, you can now define two triggers on the same timing and action, eg two triggers for 'before insert'[10], their triggers are still only per row changes and not per statement. Also, MySQL doesn't support the INSTEAD OF[11]. This can be done by using REPLACE instead but has some other shortcomings if you don't know how REPLACE works[12]. DDL and LOGON triggers are not offered at all[13] On the other hand, the error handling, in terms of rollback, is similar to Ms[14].

Problems or disadvantages associated with triggers

It is not always obvious to all users or programmers that triggers are in place. For example in MySQL a programmer would need TRIGGER privilege on a table[15] to be able to use the SHOW TRIGGERS statement. Even then, a table being modified by a trigger may not itself have a trigger that will show why it is being changed and may potentially confuse a programmer working just with the other table.

Order of triggers for the same event

Multiple triggers can be defined for the same event on the same table (e.g. in MySQL). The order of execution may be important. This can be done using PRECEDES or FOLLOWS. If left to default, in the order in which they were created (MySQL[16]) or random (Oracle[17]), a resulting bug could be difficult to find. For example, two triggers are set up on the same event/table with the assumption that the order doesn't matter. The problem occurs when later one trigger is changed, in such a way that the order now becomes important. Consider:

t2: Before updating in T, do d+ = a + b <-- executes 2nd (where a, b, c, d are database fields in table T)
t1: Before updating in T, do d+ = b + c <-- executes 1st

Six months later someone else comes along and change to:

t2: Before updating in T, do d+ = a + b + d <-- executes 2nd
t1: Before updating in T, do d+ = b + c <-- executes 1st

Expecting d to have the value from a previous update (not the one from t1).

Trigger rollback in the case of an error

For non-transactional statements, rollback cannot be done, so if the trigger is of 'before insert' type and the actual insert statement fails, any changes performed prior to the point of the error, i.e. the trigger, remain in effect[18]. This can mean that changes made by a trigger could be made even though the main modification to the database being attempted has failed. This emphasises the importance of using transactional statements with triggers so that triggered updates are rolled back in the case of a failure.

Finally, triggers can also be confusing when views include underlying tables with triggers.

---

[9] https://www.ibm.com/support/knowledgecenter/en/SSEPGG_9.7.0/com.ibm.db2.luw.sql.ref.doc/doc/r0000931.html
[10] https://dev.mysql.com/doc/refman/8.0/en/trigger-syntax.html
[11]https://www.surekhatech.com/blog/table-trigger-in-mysql
[12] https://falseisnotnull.wordpress.com/2015/09/25/mariadbmysql-on-replace-triggers/
[13] https://forums.mysql.com/read.php?99,664025,664026#msg-664026
[14] https://stackoverflow.com/questions/548541/insert-ignore-vs-insert-on-duplicate-key-update/548570#548570
[15] https://dev.mysql.com/doc/refman/8.0/en/show-triggers.html
[16] https://dev.mysql.com/doc/refman/8.0/en/trigger-syntax.html
[17] https://web.stanford.edu/dept/itss/docs/oracle/10gR2/server.102/b14220/triggers.htm
[18] https://dev.mysql.com/doc/refman/8.0/en/trigger-syntax.html