SQL Trigger:

Trigger: A trigger is a stored procedure in database which automatically invokes whenever a special event in the database occurs. For example, a trigger can be invoked when a row is inserted into a specified table or when certain table columns are being updated.

Syntax:

```
create trigger [trigger_name]
[before | after]
{insert | update | delete}
on [table_name]
[for each row]
[trigger_body]
```

Explanation of syntax:

1. create trigger [trigger_name]: Creates or replaces an existing trigger with the trigger_name.
2. [before | after]: This specifies when the trigger will be executed.
3. {insert | update | delete}: This specifies the DML operation.
4. on [table_name]: This specifies the name of the table associated with the trigger.
5. [for each row]: This specifies a row-level trigger, i.e., the trigger will be executed for each row being affected.
6. [trigger_body]: This provides the operation to be performed as trigger is fired.

BEFORE and AFTER of Trigger:

BEFORE triggers run the trigger action before the triggering statement is run. AFTER triggers run the trigger action after the triggering statement is run.

Example:

Given Student Report Database, in which student marks assessment is recorded. In such schema, create a trigger so that the total and percentage of specified marks is automatically inserted whenever a record is insert.

Here, as trigger will invoke before record is inserted so, BEFORE Tag can be used.

Suppose the database Schema,

```
mysql> desc Student;

+-------+-------------+------+-----+---------+----------------+

| Field | Type        | Null | Key | Default | Extra          |

+-------+-------------+------+-----+---------+----------------+

| tid   | int(4)      | NO   | PRI | NULL    | auto_increment |
| name  | varchar(30) | YES  |     | NULL    |                |
| subj1 | int(2)      | YES  |     | NULL    |                |
| subj2 | int(2)      | YES  |     | NULL    |                |
| subj3 | int(2)      | YES  |     | NULL    |                |
| total | int(3)      | YES  |     | NULL    |                |
| per   | int(3)      | YES  |     | NULL    |                |
+-------+-------------+------+-----+---------+----------------+
7 rows in set (0.00 sec)
```

SQL Trigger to problem statement:

```
create trigger stud_marks
before INSERT
on
Student
for each row
set Student.total = Student.subj1 + Student.subj2 + Student.subj3, Student.per = Student.total * 60 / 100;
```

Above SQL statement will create a trigger in the student database in which whenever subjects marks are entered, before inserting this data into the database, trigger will compute those two values and insert with the entered values. i.e.,

```
mysql> insert into Student values(0, "ABCDE", 20, 20, 20, 0, 0);
Query OK, 1 row affected (0.09 sec)

mysql> select * from Student;
+-----+-------+-------+-------+-------+-------+------+
| tid | name  | subj1 | subj2 | subj3 | total | per  |
+-----+-------+-------+-------+-------+-------+------+
| 100 | ABCDE |    20 |    20 |    20 |    60 |   36 |
+-----+-------+-------+-------+-------+-------+------+
1 row in set (0.00 sec)
```

In this way trigger can be created and executed in the databases.

Book Management Database:

For example, given Library Book Management database schema with Student database schema. In these databases, if any student borrows a book from library then the count of that specified book should be decremented. To do so,

Suppose the schema with some data,

```
mysql> select * from book_det;
+-----+-------------+--------+
| bid | btitle      | copies |
+-----+-------------+--------+
|   1 | Java        |     10 |
|   2 | C++         |      5 |
|   3 | MySql       |     10 |
|   4 | Oracle DBMS |      5 |
+-----+-------------+--------+
4 rows in set (0.00 sec)
```

```
mysql> select * from book_issue;
+------+------+--------+
| bid  | sid  | btitle |
+------+------+--------+
1 row in set (0.00 sec)
```

To implement such procedure, in which if the system inserts the data into the book_issue database a trigger should automatically invoke and decrements the copies attribute by 1 so that a proper track of book can be maintained.

Trigger for the system –

```
create trigger book_copies_deducts
after INSERT
on book_issue
for each row
update book_det set copies = copies - 1 where bid = new.bid;
```

Above trigger, will be activated whenever an insertion operation performed in a book_issue database, it will update the book_det schema setting copies decrements by 1 of current book id(bid).

Results –

```
mysql> insert into book_issue values(1, 100, "Java");

Query OK, 1 row affected (0.09 sec)


mysql> select * from book_det;

+-----+-------------+--------+

| bid | btitle      | copies |

+-----+-------------+--------+

|   1 | Java        |      9 |

|   2 | C++         |      5 |

|   3 | MySql       |     10 |

|   4 | Oracle DBMS |      5 |

+-----+-------------+--------+

4 rows in set (0.00 sec)


mysql> select * from book_issue;

+------+------+--------+

| bid  | sid  | btitle |

+------+------+--------+

|    1 |  100 | Java   |

+------+------+--------+

1 row in set (0.00 sec)
```

As above results show that as soon as data is inserted, copies of the book deducts from the book schema in the system.