# Triggers

**Relational Databases Basics**

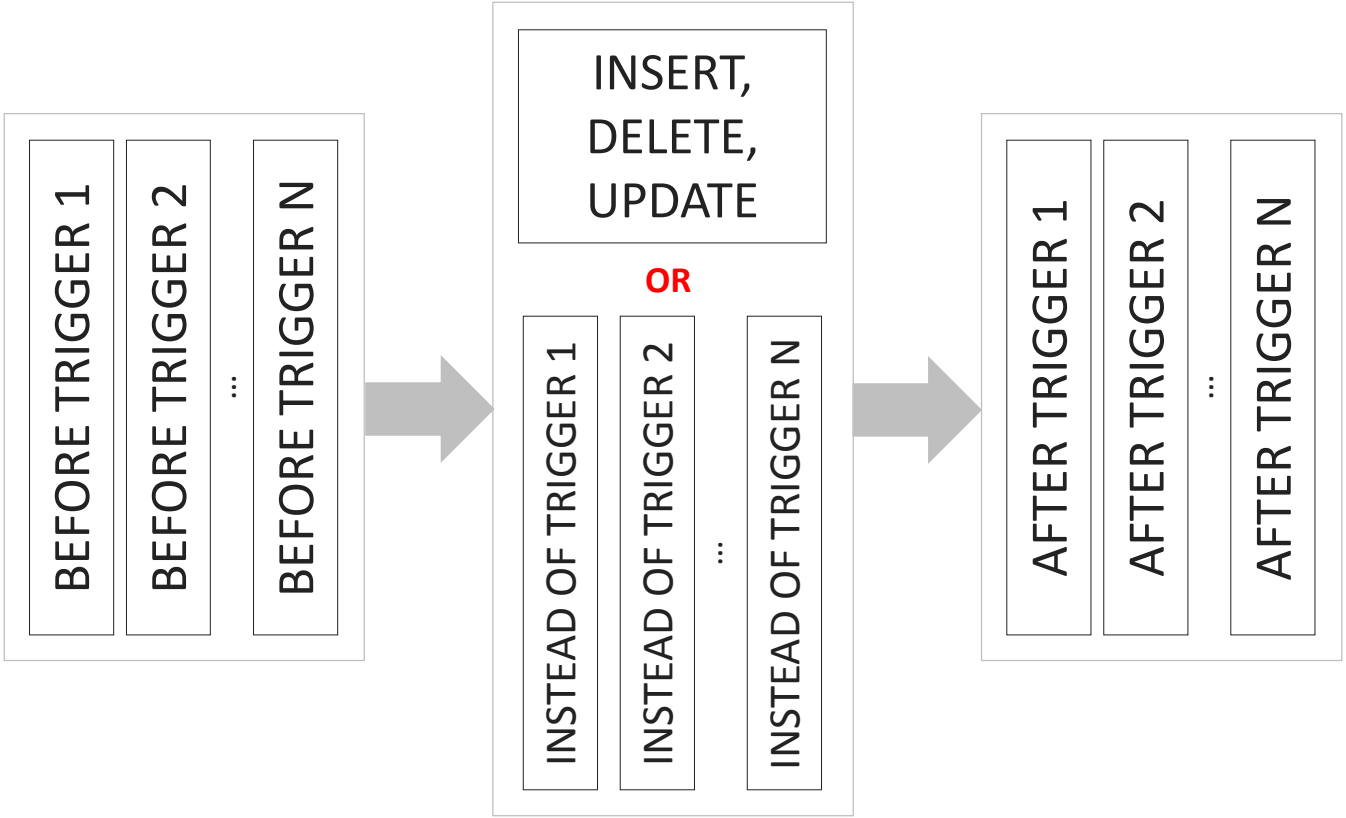
**Read and remember!**

**Trigger** – a special database object describing an action to be performed automatically in specific situations.

# Triggering actions in miscellaneous DBMSes

| Action | BEFORE, AFTER, INSTEAD OF | MySQL | MS SQL Server | Oracle |
|---|---|---|---|---|
| INSERT | BEFORE | + | - | + |
| | AFTER | + | + | + |
| | INSTEAD OF | - | + | + |
| UPDATE | BEFORE | + | - | + |
| | AFTER | + | + | + |
| | INSTEAD OF | - | + | + |
| DELETE | BEFORE | + | - | + |
| | AFTER | + | + | + |
| | INSTEAD OF | - | + | + |
| CREATE TABLE | BEFORE | - | - | See the documentation |
| | AFTER | - | + | |
| | INSTEAD OF | - | - | |
| DROP TABLE | BEFORE | - | - | See the documentation |
| | AFTER | - | + | |
| | INSTEAD OF | - | - | |
| LOGIN | BEFORE | - | - | See the documentation |
| | AFTER | - | + | |
| | INSTEAD OF | - | - | |
| Other database objects operations | BEFORE | - | - | See the documentation |
| | AFTER | - | - | |
| | INSTEAD OF | - | - | |
| Other situations | BEFORE | - | - | See the documentation |
| | AFTER | - | - | |
| | INSTEAD OF | - | - | |

# Data modification triggers logic



# Row-level triggers and statement-level triggers

## Data modification

[item]

| i_id | i_price |
|---|---|
| 345 | 25 |
| 723 | 30 |
| 991 | 15 |

[item]

| i_id | i_price |
|---|---|
| 345 | 30 |
| 723 | 36 |
| 991 | 18 |

## Triggers logic

### Row-level

| | | i_id | i_price |
|---|---|---|---|
| RUN 1 | OLD | 345 | 25 |
| | NEW | 345 | 30 |

| | | i_id | i_price |
|---|---|---|---|
| RUN 2 | OLD | 723 | 30 |
| | NEW | 723 | 36 |

| | | i_id | i_price |
|---|---|---|---|
| RUN 3 | OLD | 991 | 15 |
| | NEW | 991 | 18 |

### Statement-level

ONE SINGLE RUN FOR THE WHOLE QUERY

[inserted]

| i_id | i_price |
|---|---|
| 345 | 30 |
| 723 | 36 |
| 991 | 18 |

[deleted]

| i_id | i_price |
|---|---|
| 723 | 30 |
| 345 | 25 |
| 991 | 15 |

# Typical operations to perform with triggers

- Complex cascade operations
- Caching tables/fields update
- Data consistency implementation and control
- Business logic implementation and control
- Relationship cardinality control
- Data domain (format) control
- On-the-fly data flaws correction

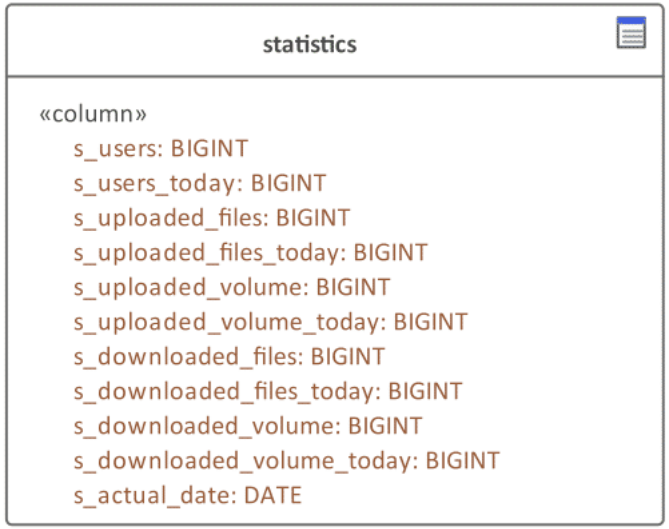
Triggers may significantly decrease database performance!

# How to create a trigger

Let’s look at some simple triggers that update some application statistics.

We may recalculate all values every time a trigger was called, but it is too slow.

Or we may just update (increase, decrease) proper fields – it’s more complex, but significantly faster.



# How to create a trigger

We shall review only `file` table triggers (you can make similar triggers for `user` table yourself).

When data in this table changes…

**file**

«column»
*PK f_id: BIGINT
*FK f_owner: BIGINT
* f_size: BIGINT
* f_upload_dt: INT
f_exp_dt: INT
* f_original_name: VARCHAR(1000)
f_original_extension: VARCHAR(1000)
* f_name: CHAR(64)
* f_control_sum: CHAR(64)
f_delete_link: CHAR(64)
f_download_count: BIGINT
f_download_count_today: BIGINT

… this table should be updated.

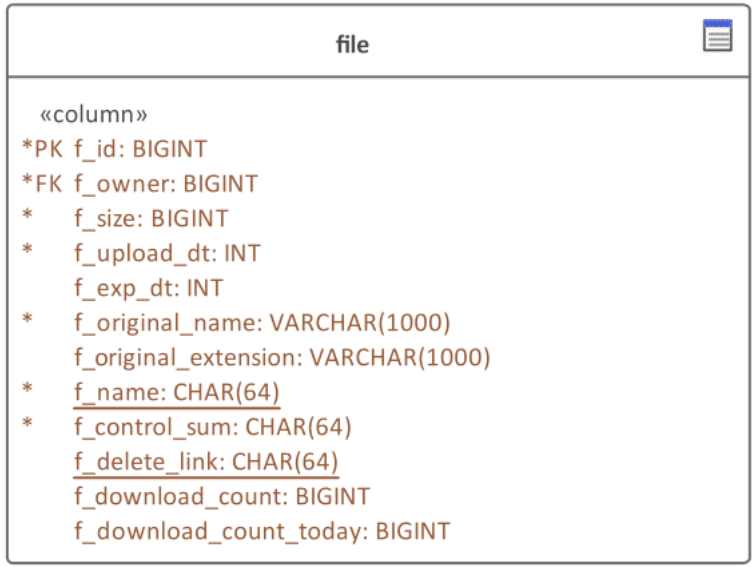
**statistics**

«column»
s_users: BIGINT
s_users_today: BIGINT
s_uploaded_files: BIGINT
s_uploaded_files_today: BIGINT
s_uploaded_volume: BIGINT
s_uploaded_volume_today: BIGINT
s_downloaded_files: BIGINT
s_downloaded_files_today: BIGINT
s_downloaded_volume: BIGINT
s_downloaded_volume_today: BIGINT
s_actual_date: DATE

# How to create a trigger

We have to create AFTER INSERT, AFTER UPDATE, AFTER DELETE triggers for this table.

Why **AFTER**?



# How to create a trigger

First, we need to clear the today's statistics each new day. It's convenient to use a stored procedure (although we may use this code directly in the trigger).

```
CREATE PROCEDURE NEW_DAY ()
BEGIN
 IF EXISTS (SELECT 1 FROM `statistics`
                    WHERE `s_actual_date` !=
                          CURRENT_DATE())
  THEN
   UPDATE `statistics` SET
    `s_users_today` = 0,
    `s_uploaded_files_today` = 0,
    `s_uploaded_volume_today` = 0,
    `s_downloaded_files_today` = 0,
    `s_downloaded_volume_today` = 0,
    `s_actual_date` = CURRENT_DATE();
   UPDATE `file` SET
    `f_download_count_today` = 0;
  END IF;
END;
```

# How to create a trigger

AFTER INSERT trigger looks like this.

```
CREATE TRIGGER `TRG_update_file_stats_after_ins` AFTER INSERT ON `file`
 FOR EACH ROW BEGIN
  CALL NEW_DAY();
  UPDATE `statistics` SET
    `s_uploaded_files` = `s_uploaded_files` + 1,
    `s_uploaded_files_today` = `s_uploaded_files_today` + 1,
    `s_uploaded_volume` = `s_uploaded_volume` + NEW.`f_size`,
    `s_uploaded_volume_today` = `s_uploaded_volume_today` + NEW.`f_size`;
 END;
```

# How to create a trigger

AFTER UPDATE trigger looks like this.

```
CREATE TRIGGER `TRG_update_file_stats_after_upd` AFTER UPDATE ON `file`
 FOR EACH ROW BEGIN
  IF (FROM_UNIXTIME(OLD.`f_upload_dt`) = CURRENT_DATE())
   THEN
    UPDATE `statistics` SET `s_uploaded_volume_today` = `s_uploaded_volume_today` - OLD.`f_size`;
    UPDATE `statistics` SET `s_uploaded_volume_today` = `s_uploaded_volume_today` + NEW.`f_size`;
    UPDATE `statistics` SET `s_downloaded_files_today` = `s_downloaded_files_today` - OLD.`f_download_count_today`;
    UPDATE `statistics` SET `s_downloaded_files_today` = `s_downloaded_files_today` + NEW.`f_download_count_today`;
    UPDATE `statistics` SET `s_downloaded_volume_today` = `s_downloaded_volume_today` - OLD.`f_size` *
                                                                                        OLD.`f_download_count_today`;
    UPDATE `statistics` SET `s_downloaded_volume_today` = `s_downloaded_volume_today` +  NEW.`f_size` *
                                                                                         NEW.`f_download_count_today`;
   END IF;

    CALL NEW_DAY();
    UPDATE `statistics` SET `s_uploaded_volume` = `s_uploaded_volume` - OLD.`f_size`;
    UPDATE `statistics` SET `s_uploaded_volume` = `s_uploaded_volume` + NEW.`f_size`;
    UPDATE `statistics` SET `s_downloaded_files` = `s_downloaded_files` - OLD.`f_download_count`;
    UPDATE `statistics` SET `s_downloaded_files` = `s_downloaded_files` + NEW.`f_download_count`;
    UPDATE `statistics` SET `s_downloaded_volume` = `s_downloaded_volume` - OLD.`f_size` * OLD.`f_download_count`;
    UPDATE `statistics` SET `s_downloaded_volume` = `s_downloaded_volume` + NEW.`f_size` * NEW.`f_download_count`;
END;
```

# How to create a trigger

AFTER DELETE trigger looks like this.

```
CREATE TRIGGER `TRG_update_file_stats_after_del` AFTER DELETE ON `file`
 FOR EACH ROW BEGIN
  IF (FROM_UNIXTIME(OLD.`f_upload_dt`) = CURRENT_DATE())
   THEN
    UPDATE `statistics` SET `s_uploaded_volume_today` = `s_uploaded_volume_today` - OLD.`f_size`;
    UPDATE `statistics` SET `s_uploaded_files_today` = `s_uploaded_files_today` - 1;
    UPDATE `statistics` SET `s_downloaded_files_today` = `s_downloaded_files_today` - OLD.`f_download_count_today`;
    UPDATE `statistics` SET `s_downloaded_volume_today` = `s_downloaded_volume_today` - OLD.`f_size` *
                                                                                          OLD.`f_download_count_today`;
   END IF;

    CALL NEW_DAY();
    UPDATE `statistics` SET `s_uploaded_volume` = `s_uploaded_volume` - OLD.`f_size`;
    UPDATE `statistics` SET `s_uploaded_files` = `s_uploaded_files` - 1;
    UPDATE `statistics` SET `s_downloaded_files` = `s_downloaded_files` - OLD.`f_download_count`;
    UPDATE `statistics` SET `s_downloaded_volume` = `s_downloaded_volume` - OLD.`f_size` * OLD.`f_download_count`;

 END;
```

# How to block an operation using a trigger

This is just a quick sample on how to prevent the DBMS from performing a prohibited action (e.g. from deleting the last user from “Administrators” group).

```
CREATE TRIGGER `TRG_preserve_last_admin_before_del` BEFORE DELETE ON
`user`
 FOR EACH ROW BEGIN
  IF (NOT EXISTS (SELECT 1
                  FROM   `m2m_user_group`
                  WHERE  (`ug_group` = 1)
                  AND (`ug_user` != OLD.`u_id`)))
   THEN
    SIGNAL SQLSTATE '45001'
    SET MESSAGE_TEXT = 'You can not delete the last user from
                        Administrators (`g_id` = 1) group.',
    MYSQL_ERRNO = 1001;
   END IF;
 END;
```

These lines make the trick

# Live demo in Sparx Enterprise Architect and MySQL Workbench

# Triggers

**Relational Databases Basics**