Views

Relational Databases Basics

View

Read and remember!

View – virtual relational variable (usually – a named result of query execution).

Materialized View – virtual relational variable **with stored data** (usually – a **stored** named result of query execution).

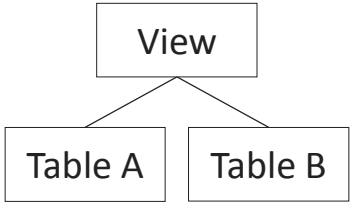
Such views are **extremely** DBMS-specific (and have almost nothing in common in different DBMSes, so refer to the documentation).

View vs materialized view

View

1. View creation is just simple fixation of a fact that from now on an SQL query may be called by a specified name

2. During such a call the real query is executed, and all necessary data is collected

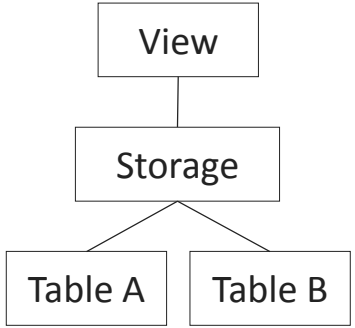


Materialized view

1. View creation gives an SQL query an alias and forces DBMS to make a temporary storage

2. This storage is updated according to specified plan

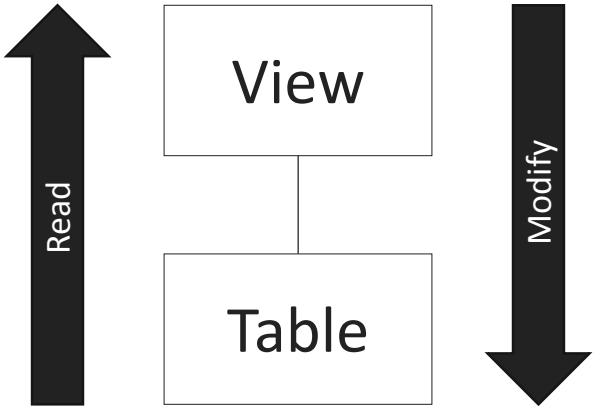
3. During the view access DBMS reads the storage to collect data



Bi-directional views

Some views allow both data reading and data modification operations. This mechanism is highly DBMS-specific, so read the manual carefully.

Such views are neither good, nor bad themselves. Bad things happen, if you expect one behavior, but get another.



Using views: pros

- Complex queries management simplification
- Good approach to simple and powerful API creation
- Business-logic simplification
- Minimum storage/speed overhead
- Performance optimization (with materialized views)
- Additional security level

Using views: cons

Redundancy (sometimes you just… don’t need views)

Additional code to write and maintain

DBMS limitations on available operations via views

So: views are neither good, nor bad – its just a tool. Use it wisely.

How to create a view

Imagine, we have to get a list files uploaded on the first Saturday and Sunday of every month.

And we have to retrieve this data from a lot of application parts.

```
SELECT *
FROM
        (
                SELECT *,
                CASE
                 WHEN WEEKDAY(FROM_UNIXTIME(`f_upload_dt`) –
                 INTERVAL DAY(FROM_UNIXTIME(`f_upload_dt`))-1 DAY) <=
                      WEEKDAY(FROM_UNIXTIME(`f_upload_dt`))
                 THEN WEEK(FROM_UNIXTIME(`f_upload_dt`), 5) -
                      WEEK(FROM_UNIXTIME(`f_upload_dt`) –
                 INTERVAL DAY(FROM_UNIXTIME(`f_upload_dt`))-1 DAY, 5) + 1
                 ELSE WEEK(FROM_UNIXTIME(`f_upload_dt`), 5) -
                      WEEK(FROM_UNIXTIME(`f_upload_dt`) –
                INTERVAL DAY(FROM_UNIXTIME(`f_upload_dt`))-1 DAY, 5)
                END AS `W`,
                WEEKDAY(FROM_UNIXTIME(`f_upload_dt`)) + 1 AS `D`
        FROM `file`
        ) AS `prepared_data`
WHERE (`W` = 1)
  AND ((`D` = 6) OR (`D` = 7))
```

The SQL query

How to create a view

But we may create a view…

The SQL query to create a view

```
CREATE VIEW `files_on_first_days_off` AS
SELECT *
FROM    ( SELECT *,
                CASE
                 WHEN WEEKDAY(FROM_UNIXTIME(`f_upload_dt`) –
                 INTERVAL DAY(FROM_UNIXTIME(`f_upload_dt`))-1 DAY) <=
                      WEEKDAY(FROM_UNIXTIME(`f_upload_dt`))
                 THEN WEEK(FROM_UNIXTIME(`f_upload_dt`), 5) -
                      WEEK(FROM_UNIXTIME(`f_upload_dt`) –
                 INTERVAL DAY(FROM_UNIXTIME(`f_upload_dt`))-1 DAY, 5) + 1
                 ELSE WEEK(FROM_UNIXTIME(`f_upload_dt`), 5) -
                      WEEK(FROM_UNIXTIME(`f_upload_dt`) –
                INTERVAL DAY(FROM_UNIXTIME(`f_upload_dt`))-1 DAY, 5)
                END AS `W`,
                WEEKDAY(FROM_UNIXTIME(`f_upload_dt`)) + 1 AS `D`
        FROM `file`
        ) AS `prepared_data`
WHERE (`W` = 1)
  AND ((`D` = 6) OR (`D` = 7))
```

The SQL query to access data

Now data access looks like this:

```
SELECT * FROM `files_on_first_days_off`
```

Managing access permissions with views

Imagine, we have to limit access some stored files (by extension).

```
CREATE VIEW `files_with_jpg_jpeg_png_gif_extensions` AS
SELECT *
  FROM `file`
 WHERE LOWER(`f_original_extension`) IN ('jpg', 'jpeg', 'png', 'gif')
WITH CHECK OPTION
```

This is how

Making a view non-bi-directional

Just add an expression to the original query to block a view from becoming a bi-directional one.

```
CREATE VIEW `ro_files_with_jpg_jpeg_png_gif_extensions` AS
SELECT `f_id`,
       `f_owner`,
       `f_size` + 0,
       `f_upload_dt`,
       `f_exp_dt`,
       `f_original_name`,
       `f_original_extension`,
       `f_name`,
       `f_control_sum`,
       `f_delete_link`
  FROM `file`
 WHERE LOWER(`f_original_extension`) IN ('jpg', 'jpeg', 'png', 'gif')
```

This is how

Live demo in Sparx Enterprise Architect and MySQL Workbench

Views

Relational Databases Basics