FILTER

Applies to: CELONIS 4.0 CELONIS 4.2 CELONIS 4.3 CELONIS 4.4

Description

Filters can be defined as Analysis filters, Sheet filters or Component filters. If a query is sent to Celonis, all active filters are propagated to the input table(s). Multiple filters on a table are merged together by a logical AND.

Filters are applied to the input tables before the actual query. If the filter expression evaluates to false for a row of the input table(s), this row is excluded before the query is applied.

Comparison against NULL

Applying a filter which compares a column against null or not null always returns an empty result. To filter on all not null values use the ISNULL function. The reasoning behind it is the same as in the previous example. Null represents an unknown value. Celonis can not be sure if two unknown values are not the same.

Syntax

```
FILTER [FORCED] condition;
```

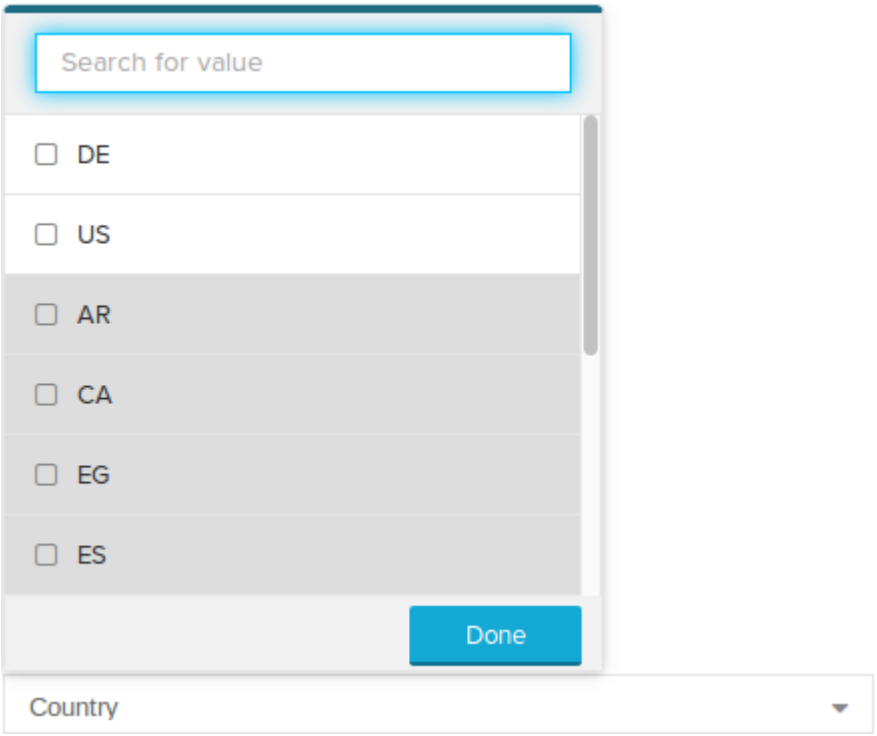
Forced Filter

If a regular filter is set as a sheet or component filter, the affected Dropdown and Button Dropdown components still show all available values. If only those values which respect the filter should be displayed and selectable, a forced filter can be used. Analysis filters are forced by default.

Example

Dropdown component using this regular sheet or component filter:

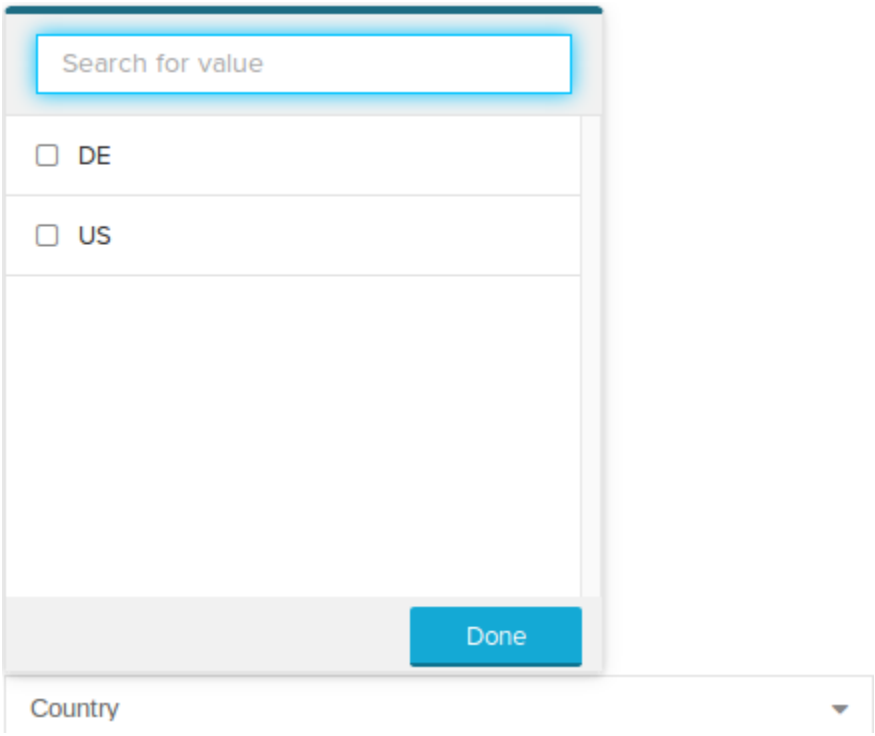
```
FILTER "Table"."Country" IN ('DE','US');
```



All values are available in the dropdown menu. Values which do not match the filter condition are displayed in gray color.

Dropdown component using this forced sheet or component filter:

```
FILTER FORCED "Table"."Country" IN ('DE','US');
```



Only values which match the filter condition are available.

Examples

Example where one filter is applied to the query. The filter condition excludes the second input row.

Query

Filter
`FILTER "Numbers"."number" != 22`
Column1
`"Numbers"."id"`
Column2
`"Numbers"."number"`

Input

Numbers

id : INT	number : INT
1	13
2	22
3	34

Output

Result

Column1 : INT	Column2 : INT
1	13
3	34

Example where one filter is applied to the query. The filter condition excludes the second input row.

Query

Filter

```
FILTER "Numbers"."number" IN (13, 34)
```

Column1

```
"Numbers"."id"
```

Column2

```
"Numbers"."number"
```

Input

Numbers

id : INT	number : INT
1	13
2	22
3	34

Output

Result

Column1 : INT	Column2 : INT
1	13
3	34

Example where two filters are applied to the query. Both filter conditions are merged together by a logical AND. The first filter condition excludes the second input row, and the second filter condition excludes the first input row. Therefore, only the third row appears in the result.

Query

Filter

```
FILTER "Numbers"."number" IN (13, 34)
```

Filter

```
FILTER "Numbers"."id" IN (2, 3)
```

Column1

```
"Numbers"."id"
```

Column2

```
"Numbers"."number"
```

Input

Numbers

id : INT	number : INT
1	13
2	22
3	34

Output

Result

Column1 : INT	Column2 : INT
3	34

Example where one filter is applied to the query. The SUM aggregate function is applied after the filter has been applied to the input table.

Query

Filter

```
FILTER "Numbers"."number" IN (13, 34)
```

Column1

```
SUM("Numbers"."number")
```

Input

Numbers

id : INT	number : INT
1	13
2	22
3	34

Output

Result

Column1 : INT
47

Example where two filters are applied to the query. Both filter conditions are merged together by a logical AND. The first filter condition excludes the first and third input row, and the second filter condition excludes the second and third input row. Therefore, the result is empty.

Query

Filter

```
FILTER "Numbers"."number" NOT IN (13, 34)
```

Filter

```
FILTER "Numbers"."id" = 1
```

Column1

```
"Numbers"."id"
```

Column2

```
"Numbers"."number"
```

Input

Numbers

id : INT	number : INT
1	13
2	22
3	34

Output

Result

Column1 : INT	Column2 : INT

Example of two joined tables where one filter is applied to the query. The filter condition excludes the last row of the companyDetail input table, therefore, the last two rows of the caseTable are excluded.

Query

Filter

```
FILTER "companyDetail"."country" = 'DE'
```

Column1

```
"caseTable"."caseId"
```

Input

caseTable

caseId : INT	companyCode : STRING	value : INT
1	'001'	600
2	'001'	400
3	'001'	200
4	'002'	300
5	'003'	300
6	'003'	200

companyDetail

companyCode : STRING	country : STRING
'001'	'DE'
'002'	'DE'
'003'	'US'

Foreign Keys

caseTable.companyCode	companyDetail.companyCode

Output

Result

Column1 : INT
1
2
3
4

Example of two joined tables where one filter is applied to the query. The filter condition excludes the last four rows of the caseTable.

Query

Filter

```
FILTER "caseTable"."value" > 300
```

Column1

```
"caseTable"."caseId"
```

Column2

```
"companyDetail"."country"
```

Output

Result

Column1 : INT	Column2 : STRING
1	'DE'
2	DE

Input

caseTable

caseId : INT	companyCode : STRING	value : INT
1	'001'	600
2	'001'	400
3	'001'	200
4	'002'	300
5	'003'	300
6	'003'	200

companyDetail

companyCode : STRING	country : STRING
'001'	'DE'
'002'	'DE'
'003'	'US'

Foreign Keys

caseTable.companyCode	companyDetail.companyCode

Applying a filter which compares a column against null always returns an empty result.

Query

Filter

```
FILTER Table1.Column1 = null
```

Column1

```
Table1.Column1
```

Input

Table1

Column1 : INT
1
null

Output

Result

Column1 : INT

Applying a filter which compares a column against not null always returns an empty result.

Query

Filter

```
FILTER Table1.Column1 != null
```

Column1

```
Table1.Column1
```

Input

Table1

Column1 : INT
1
null

Output

Result

Column1 : INT