

# DPM 2.0- Refit project

## DPM Expression Language operators

DOCUMENT VERSION AND STATUS: v.0.7, Public Working Draft

DOCUMENT DATE: 22/05/2023

# DPM Expression Language operators

## Contents

1	Introduction .....	8
2	Common general behaviours for the operators .....	8
2.1	Unary operators.....	8
2.1.1	Scalars .....	8
2.1.2	Recordsets .....	8
2.2	Binary operators.....	8
2.2.1	Scalars .....	8
2.2.2	Recordset and scalar .....	9
2.2.3	Recordsets .....	9
2.2.4	Intervals treatment.....	10
3	Selection operators .....	10
3.1	Selection operator.....	10
3.1.1	<i>Syntax</i> .....	10
3.1.2	Input parameters .....	11
3.1.3	Output.....	11
3.1.4	Semantics .....	11
3.1.5	Additional constraints .....	11
3.1.6	Behaviour.....	11
3.1.7	DPM-ML metamodel representation.....	12
3.2	With.....	12
3.2.1	Syntax.....	12
3.2.2	Input parameters .....	12
3.2.3	Output.....	12
3.2.4	Semantics .....	12
3.2.5	Additional constraints .....	12
3.2.6	Behaviour.....	12
3.2.7	Examples .....	12
3.2.8	Metamodel representation .....	13
4	Numeric.....	14
4.1	Numeric operators' general behaviour .....	14
4.1.1	Input parameters .....	14
4.1.2	Result type.....	14
4.1.3	Behaviour.....	14

4.2	Unary arithmetic operators .....	14
4.2.1	Plus (+).....	14
4.2.2	Minus (-).....	14
4.2.3	Absolute value (abs) .....	15
4.2.4	Exponential (exp).....	15
4.2.5	Natural logarithm (ln) .....	16
4.2.6	Square root (sqrt) .....	16
4.2.7	Logarithm (log) .....	17
4.3	Binary arithmetic operators .....	17
4.3.1	Addition (+) .....	18
4.3.2	Subtraction (-) .....	18
4.3.3	Multiplication (*) .....	19
4.3.4	Division (/).....	19
4.3.5	Maximum (max).....	20
4.3.6	Minimum (min).....	21
4.3.7	Power.....	21
5	Comparison operators .....	22
5.1	Comparison operators' general behaviour.....	22
5.1.1	Input parameters .....	22
5.1.2	Result type.....	22
5.1.3	Constraints .....	22
5.1.4	Behaviour.....	22
5.2	Equal (=) .....	22
5.3	Not Equal (<>) .....	23
5.4	Greater than (>).....	23
5.5	Greater Equal than (>=) .....	24
5.6	Less than (<) .....	24
5.7	Less Equal than (<=).....	25
5.8	Element of (in) .....	26
5.9	Match characters (match) .....	26
5.10	Is null (isnull) .....	27
6	Logical.....	27
6.1	Conjunction (and).....	27
6.2	Disjunction (or) .....	28
6.3	Exclusive disjunction (xor) .....	28
6.4	Negation (not) .....	29
7	Aggregate operators .....	30

7.1	Aggregate operators' general behaviour .....	30
7.1.1	Syntax .....	30
7.1.2	Input parameters .....	30
7.1.3	Output .....	30
7.1.4	Semantics .....	30
7.1.5	Additional constraints .....	30
7.1.6	Behaviour .....	30
7.1.7	Examples .....	30
7.2	Sum (sum) .....	31
7.2.1	Syntax .....	31
7.2.2	Input parameters .....	31
7.2.3	Output .....	31
7.2.4	Semantics .....	31
7.2.5	Additional constraints .....	31
7.2.6	Behaviour .....	31
7.3	Count (count) .....	32
7.3.1	Syntax .....	32
7.3.2	Input parameters .....	32
7.3.3	Output .....	32
7.3.4	Semantics .....	32
7.3.5	Additional constraints .....	32
7.3.6	Behaviour .....	32
7.4	Minimum value (min_aggr) .....	32
7.4.1	Syntax .....	32
7.4.2	Input parameters .....	32
7.4.3	Output .....	32
7.4.4	Semantics .....	32
7.4.5	Additional constraints .....	32
7.4.6	Behaviour .....	32
7.5	Maximum value (max_aggr) .....	33
7.5.1	Syntax .....	33
7.5.2	Input parameters .....	33
7.5.3	Output .....	33
7.5.4	Semantics .....	33
7.5.5	Additional constraints .....	33
7.5.6	Behaviour .....	33
7.6	Average (avg) .....	33

7.6.1	Syntax.....	33
7.6.2	Input parameters.....	33
7.6.3	Output.....	33
7.6.4	Semantics .....	33
7.6.5	Additional constraints .....	33
7.6.6	Behaviour.....	33
7.7	Median value (median).....	33
7.7.1	Syntax.....	33
7.7.2	Input parameters.....	34
7.7.3	Output.....	34
7.7.4	Semantics .....	34
7.7.5	Additional constraints .....	34
7.7.6	Behaviour.....	34
8	Conditional operators .....	35
8.1	If-then-else.....	35
8.1.1	Syntax.....	35
8.1.2	Input parameters.....	35
8.1.3	Output.....	35
8.1.4	Semantics .....	35
8.1.5	Additional constraints .....	35
8.1.6	Behaviour.....	36
8.1.7	Examples .....	36
8.2	Null substitute (nvl).....	36
8.2.1	Syntax.....	36
8.2.2	Input parameters.....	36
8.2.3	Output.....	36
8.2.4	Semantics .....	36
8.2.5	Additional constraints .....	36
8.2.6	Behaviour.....	36
8.2.7	Examples .....	36
8.3	Filter .....	36
8.3.1	Syntax.....	36
8.3.2	Input parameters.....	36
8.3.3	Output.....	36
8.3.4	Semantics .....	37
8.3.5	Additional constraints .....	37
8.3.6	Behaviour.....	37

8.3.7	Examples .....	37
9	String operators.....	37
9.1	Length (len).....	37
9.1.1	Syntax.....	37
9.1.2	Input parameters.....	37
9.1.3	Output.....	37
9.1.4	Semantics .....	37
9.1.5	Additional constraints .....	37
9.1.6	Behaviour.....	37
9.1.7	Examples .....	37
9.2	Concatenate (&).....	37
9.2.1	Syntax.....	37
9.2.2	Input parameters.....	37
9.2.3	Output.....	37
9.2.4	Semantics .....	38
9.2.5	Additional constraints .....	38
9.2.6	Behaviour.....	38
9.2.7	Examples .....	38
10	Time operators.....	39
10.1	Time shift.....	39
10.1.1	Syntax.....	39
10.1.2	Input parameters.....	39
10.1.3	Output.....	39
10.1.4	Semantics .....	39
10.1.5	Additional constraints .....	39
10.1.6	Behaviour.....	39
10.1.7	Examples .....	39
11	Clause operators.....	40
11.1	where.....	40
11.1.1	Syntax.....	40
11.1.2	Input parameters.....	41
11.1.3	Output.....	41
11.1.4	Semantics .....	41
11.1.5	Additional constraints .....	41
11.1.6	Behaviour.....	41
11.1.7	Examples .....	41
11.2	rename.....	42

11.2.1	Syntax.....	42
11.2.2	Input parameters.....	42
11.2.3	Output.....	42
11.2.4	Semantics .....	42
11.2.5	Additional constraints .....	42
11.2.6	Behaviour.....	42
11.2.7	Examples .....	42
11.3	get.....	43
11.3.1	Syntax.....	43
11.3.2	Input parameters.....	43
11.3.3	Output.....	43
11.3.4	Semantics .....	43
11.3.5	Additional constraints .....	43
11.3.6	Behaviour.....	43
11.3.7	Examples .....	43

## 1 Introduction

This documentation provides the individual description of the semantics for all the operators in the DPM Operations (DPM-XL and DPM-ML).

The document is structured from more generic to more concrete. It starts with general behaviours that apply to several operators of different types. Then it follows with the common behaviour of the operators of the same type, finalizing with the specific behaviour of the single operators.

## 2 Common general behaviours for the operators

### 2.1 Unary operators

This general behaviour is applied to operators taking as an argument one single operand.

Unless explicitly specified differently, the behaviour for the unary operators is as follows:

#### 2.1.1 Scalars

The *Operator* is applied on a scalar value and returns a scalar value.

#### 2.1.2 Recordsets

The *Operator* is applied on a *Recordset* and returns a *Recordset*.

The *Operator* is applied to the values of all the facts of the *Recordset*.

##### 2.1.2.1 Constraints

1. The application of the *Operator* is only allowed if all the facts of the *Operand Recordset* are compatible with the operator.

##### 2.1.2.2 Structure

The structure of the resulting *Recordset* contains the key and fact components of the *Operand Recordset*, the attribute components are not propagated.

##### 2.1.2.3 Records

The operators produce one output *Record* per each input *Record*, which have:

- For *Key Components*, the values unchanged.
- For the *Fact* values, the operator is applied to the input value and returns the corresponding value.

### 2.2 Binary operators

This general behaviour is applied whenever an operator takes as input two operands, which includes the following operators:

- [Arithmetic Binary operators](#)
- [Comparison operators](#)
- [Logical Binary operators](#)

Unless explicitly specified differently, the behaviour for the binary *Operators* is as follows.

#### 2.2.1 Scalars

If the two *Operands* of a binary *Operator* are *Scalars*, the result shall be the *Scalar* resulting of applying the *Operator* to the *Operands*.



### 2.2.2 Recordset and scalar

A binary *Operator* applied to a *Recordset Operand* and a *Scalar*, will result in a *Recordset* with the same key and fact components as the input *Recordset Operand* (attribute components are not propagated). The operator shall be applied to every record of the input *Recordset* and the *Scalar*.

#### 2.2.2.1 Examples

$$0.25 * \{S.26.01, r0600, cNNN\}$$

Supposing that the selection yields the following *Recordset*:

c	f
0060	100
0080	200

The operation results in:

c	f
0060	25
0080	50

### 2.2.3 Recordsets

#### 2.2.3.1 Constraints

Binary *Operators* can only be applied to two *Recordsets Operands* if they have:

1. exactly the same *Key Components*; or  
the *Key Components* of one *Recordset* (Reference *Recordset*) are a superset of the *Key Components* of the other *Recordset*.

#### 2.2.3.2 Structure

The operator yields a *Recordset* with the common *Key Components* in case 1, or the *Key Components* of the Reference *Recordset* in case 2. The resulting *Recordset* does not contain any *Attribute Component* from the Operand *Recordsets*.

#### 2.2.3.3 Records

The operator applies to the pairs of values resulting from performing an inner join of the input *Recordsets* on the common *Key Components*.

#### 2.2.3.4 Examples

##### 2.2.3.4.1 Same identifiers

$$\{C\ 28.00\ c040\} + \{C\ 28.00\ c190\}$$

Supposing that the selections yield:

{C 28.00 c040}		{C 28.00 c190}	
INC	f	INC	f
123	1000	123	-100
456	2000	456	-200
789	3000	789	-300

The result would be:

INC	f
123	900
456	1800
789	2700

#### 2.2.3.4.2 Subset of identifiers

$\{F\ 40.01\ c0110\} \geq \{F\ 40.02\ c0060\}$

Supposing that the selections yield:

{F 40.01 c0110}			{F 40.02 c0060}					
LIN	TYC	f	LIN	TYC	STC	LHC	LHO	f
123	x1	1	123	x1	111	ABC	x1	0.3
456	x1	0.8	123	x1	111	DEF	x1	0.7
789	x1	0.4	456	x1	222	ABC	x1	0.85

The result would be:

LIN	TYC	STC	LHC	LHO	f
123	x1	111	ABC	x1	true
123	x1	111	DEF	x1	true
456	x1	222	ABC	x1	false

Note that:

- An inner join on the common *Key Components* is performed. This means that there is no comparison for the record in F 40.01 with LIN = 789, because there is no match in F 40.02.
- The result of the operation has the *Structure* of the *Reference Recordset*, which is the one that has a superset of *Key Components*.
- For each match in the join the operator  $\geq$  is applied to the pairs of values for the facts.

#### 2.2.4 Intervals treatment

When one of two numeric operands is an interval, and the other is a point, the point is transformed into an interval with centre value equal to the point value, and radius 0.

## 3 Selection operators

### 3.1 Selection operator

#### 3.1.1 Syntax

$\{[ttable \mid vvariable \mid ooperation]1$

$[rrow \mid [, rrow]^* \mid rrow\_init-rrow\_end \mid r^*]^*$

$[ccol \mid [, ccol]^* \mid ccol\_init-ccol\_end \mid c^*]^*$

$[ssheet \mid [, ssheet]^* \mid ssheet\_init-ssheet\_end \mid s^*]^*$

```
    interval_arithmetics, fallback_value
}
```

### 3.1.2 Input parameters

**table:** The code of a DPM Report table.

**variable:** the code of a DPM variable.

**operation:** The code of a DPM Operation.

**row, row\_init, row\_end:** A Reference to the code of a row of a DPM table.

**col, col\_init, col\_end:** A Reference to the code of a col of a DPM table.

**sheet, sheet\_init, sheet\_end:** A Reference to the code of a sheet of a DPM table.

**interval\_arithmetics:** A Boolean value specifying whether interval arithmetics should apply. The default value, in case the parameter is omitted, is *false*.

**default\_value:** A scalar value specifying the value to be used in case of nulls.

### 3.1.3 Output

rset<\*>

### 3.1.4 Semantics

Serves to select data as defined in the DPM model.

### 3.1.5 Additional constraints

1. Existence of the parameters. The referred table, variable or operation need to be defined in the DPM.
2. Operations need to be defined in the same script.
3. Rows, columns and sheets need to exist in the table or operation on which they are defined.
4. When the selection is done at variable level, row, column and sheet parameters cannot exist.

### 3.1.6 Behaviour

Returns a *Recordset* with the following structure:

- *Key Components:*
  - *DPM-XL*
    - **Row:** If the selected table has ordinate rows and more than one row is present in the selection. The name of the *Component* is “r”.
    - **Column:** If the selected table has ordinate columns and more than one column is present in the selection. The name of the *Component* is “c”.
    - **Sheet:** If the selected table has ordinate sheets and more than one sheet is present in the selection. The name of the *Component* is “s”.
  - *DPM-ML*
    - **X:** If there is *Column* for DPM-XL.
    - **Y:** If there is *Row* for DPM-XL.

- Z: If there is *Sheet* for DPM-XL.
  - A *DPM Key Component* for each *Key Variable* belonging to the *Key* associated to the selected *Variable*. The name of the *Component* is the *Code* of the *Property* associated to the *Key Variable*. The *Data Type* of the component shall be the correspondent *Data Type* of the *Metric* or *Property* associated to the respective *Key Variable*.
- *Fact Component*. With the *Data Type* corresponding to the selected *Variable*.
- *Attribute Components*. One Component for each *Attribute Variable* associated to the selected *Variable*. The name of the Component is the Code of the Property associated to the *Attribute Variable*. The *Data Type* of the component shall be the correspondent *Data Type* of the Metric or Property associated to the respective *Attribute Variable*.

The *Records* for the *Recordset* shall be obtained from the input data according to the selection.

### 3.1.7 DPM-ML metamodel representation

There is no metamodel representation for the selection operator. It is used to select, in last instance, variables. So the result of the selection are processed to get the relevant variables for DPM-ML, as well as the *DefaultValue* and *UseIntervalArithmetics* attributes.

## 3.2 With

### 3.2.1 Syntax

**with** partial\_selection: expression

### 3.2.2 Input parameters

**partial\_selection**: A selection expression (see *Selection Operator*)

**expression**: A expression including selection operators

### 3.2.3 Output

Does not generate output; modifies the selections of the expression after the colon.

### 3.2.4 Semantics

Serves to simplify expressions by adding a single context that may apply to all the operands in the expression.

### 3.2.5 Additional constraints

None

### 3.2.6 Behaviour

The selection parameters in the partial selection applies to all the selections in the expression, unless they are overridden by a more specific parameter.

Therefore, in a selection inside the expression, whenever one of the parameters of the selection expression is not present, but that parameter is present on the partial selection, the parameter of the partial selection applies.

### 3.2.7 Examples

With {F 01.01 c0010, default:0, interval:false}:

$$\{r0010\} = \{r0020\} + \{r0030\} + \{r0040\}$$

The partial selection applies to all the selections in the expression

With {F 01.01 c0010, default:0, interval:false}:

$$\{r0010\} + \{r0040\} = \{F04.01 r0010, c0010\}$$

The partial selection applies to the two selections that do not refer to a table. Regarding the other selection, the default and intervals apply

With {F 01.01 c0010, default:0, interval:false}:

$$\{F 01.01 r0010\} + \{F 01.01 r0040\} = \{F04.01 r0010, c0010 \text{ default:null}\}$$

The partial selection applies to the two selections that refer to the same table referred to in the partial selection. Regarding the other selection, interval argument in the partial selection applies, but the default is overridden.

With {c0010, default:0, interval:false}:

$$\{F 01.01 r0010\} + \{F 01.01 r0040\} = \{F04.01 r0010\}$$

The partial selection applies to all the selections in the expression.

### 3.2.8 Metamodel representation

See selection operator.

## 4 Numeric

Numeric operators describe operations involving operands with data type Number, Integer or Number Interval.

### 4.1 Numeric operators' general behaviour

#### 4.1.1 Input parameters

Operands can be of Recordset or Scalar type. Must be defined as *Number interval, Number, or Integer*.

#### 4.1.2 Result type

Recordset or Scalar with type *Number interval, Number, or Integer*.

#### 4.1.3 Behaviour

If any operand is null, then the result is also null.

Numeric operators can be applied to any numeric type (*Number interval, Number, or Integer*) and combination of them, in which case casting to the highest type shall apply.

If the type of operands is *Integer* then the result has type *Integer*. If any of the operands is of type *Number*, and there are no *Number intervals*, then the result has type *Number*. If any operand is of *Number Interval* type, the result has type *Number Interval*.

### 4.2 Unary arithmetic operators

#### 4.2.1 Plus (+)

##### 4.2.1.1 Syntax

+ op

##### 4.2.1.2 Input parameters

Op: rset | scal <num + interval>

##### 4.2.1.3 Output

rset | scal <num>

##### 4.2.1.4 Semantics

Returns the operand unchanged.

##### 4.2.1.5 Additional constraints

None

##### 4.2.1.6 Behaviour

- Unary operators' standard behaviour.
- Numeric operators' standard behaviour.

##### 4.2.1.7 Examples

+ 1                      results in              1

+ (-3) results in              -3

#### 4.2.2 Minus (-)

##### 4.2.2.1 Syntax

- op

## EBA and EIOPA Regular Use

### 4.2.2.2 *Input parameters*

Op: rset | scal <num + interval>

### 4.2.2.3 *Output*

rset | scal <num>

### 4.2.2.4 *Semantics*

Inverts the sign of the operand.

For intervals, inverts the sign of the centre, leaving the radius unchanged.

### 4.2.2.5 *Additional constraints*

None

### 4.2.2.6 *Behaviour*

- Unary operators' standard behaviour.
- Numeric operators' standard behaviour.

### 4.2.2.7 *Examples*

- 1                      results in        -1

- (-3)   results in        3

## 4.2.3 Absolute value (abs)

### 4.2.3.1 *Syntax*

**abs(op)**

### 4.2.3.2 *Input parameters*

Op: rset | scal <num + interval>

### 4.2.3.3 *Output*

rset | scal <num>

### 4.2.3.4 *Semantics*

Calculates the absolute value of a number.

For intervals, return the absolute value of the centre, leaving the radius unchanged

### 4.2.3.5 *Additional constraints*

None

### 4.2.3.6 *Behaviour*

- Unary operators' standard behaviour.
- Numeric operators' standard behaviour.

### 4.2.3.7 *Examples*

Abs(-1)                results in        1

Abs(3) results in        3

## 4.2.4 Exponential (exp)

### 4.2.4.1 *Syntax*

**exp(op)**

## EBA and EIOPA Regular Use

### 4.2.4.2 *Input parameters*

Op: rset | scal <num>

### 4.2.4.3 *Output*

rset | scal <num>

### 4.2.4.4 *Semantics*

Returns e (base of the natural logarithm) raised to the op power.

### 4.2.4.5 *Additional constraints*

None

### 4.2.4.6 *Behaviour*

- Unary operators' standard behaviour.
- Numeric operators' standard behaviour.

### 4.2.4.7 *Examples*

exp(2) results in 7.38905

exp(1) results in 2.71828 (the e number)

## 4.2.5 Natural logarithm (ln)

### 4.2.5.1 *Syntax*

**ln(op)**

### 4.2.5.2 *Input parameters*

Op: rset | scal <num>

### 4.2.5.3 *Output*

rset | scal <num>

### 4.2.5.4 *Semantics*

Calculates the natural logarithm of a number.

### 4.2.5.5 *Additional constraints*

The numeric values must be greater than 0.

### 4.2.5.6 *Behaviour*

- Unary operators' standard behaviour.
- Numeric operators' standard behaviour.
- If any value is smaller than or equal to 0, generates a run-time error.

### 4.2.5.7 *Examples*

ln(1) results in 0

ln(148) results in 4.997

## 4.2.6 Square root (sqrt)

### 4.2.6.1 *Syntax*

**sqrt(op)**

### 4.2.6.2 *Input parameters*

Op: rset | scal <num>



## EBA and EIOPA Regular Use

### 4.2.6.3 *Output*

rset | scal <num>

### 4.2.6.4 *Semantics*

Calculates the square root of a number.

### 4.2.6.5 *Additional constraints*

The numeric values must be greater than or equal to 0.

### 4.2.6.6 *Behaviour*

- Unary operators' standard behaviour.
- Numeric operators' standard behaviour.
- If any value is smaller than 0, generates a run-time error.

### 4.2.6.7 *Examples*

sqrt(4) results in 2

sqrt(25) results in 5

## 4.2.7 Logarithm (log)

### 4.2.7.1 *Syntax*

**log**(op, base)

### 4.2.7.2 *Input parameters*

op: rset | scal <num>

base: scal <num>

### 4.2.7.3 *Output*

rset | scal <num>

### 4.2.7.4 *Semantics*

Calculates the logarithm of base op.

### 4.2.7.5 *Additional constraints*

op numeric values must be greater than 1. base numeric values must be greater than 0.

### 4.2.7.6 *Behaviour*

- Unary operators' standard behaviour.
- Numeric operators' standard behaviour.
- If the base is 1 or smaller, generates a run-time error.
- If the number is 0 or smaller, generates a run-time error.

### 4.2.7.7 *Examples*

Log(512, 2) results in 9

Log(100, 10) results in 2

## 4.3 Binary arithmetic operators

This operators group follows the [General behavior for binary operators](#).

## EBA and EIOPA Regular Use

### 4.3.1 Addition (+)

#### 4.3.1.1 Syntax

left + right

#### 4.3.1.2 Input parameters

left: rset | scal <num + interval>

right: rset | scal <num + interval>

#### 4.3.1.3 Output

rset | scal <num + interval>

#### 4.3.1.4 Semantics

Returns the sum of two numbers.

For intervals:

- Centre is calculated as  $\text{centre}(\text{left}) + \text{centre}(\text{right})$
- Radius is calculated  $\text{radius}(\text{left}) + \text{radius}(\text{right})$

#### 4.3.1.5 Additional constraints

None.

#### 4.3.1.6 Behaviour

- Binary operators' standard behaviour.
- Numeric operators' standard behaviour.

#### 4.3.1.7 Examples

3 + 2	results in	5
-7 + 3	results in	-4

### 4.3.2 Subtraction (-)

#### 4.3.2.1 Syntax

left - right

#### 4.3.2.2 Input parameters

left: rset | scal <num + interval>

right: rset | scal <num + interval>

#### 4.3.2.3 Output

rset | scal <num + interval>

#### 4.3.2.4 Semantics

Returns the difference of two numbers.

For intervals:

- Centre is calculated as  $\text{centre}(\text{left}) - \text{centre}(\text{right})$
- Radius is calculated  $\text{radius}(\text{left}) + \text{radius}(\text{right})$

#### 4.3.2.5 Additional constraints

None.

### 4.3.2.6 Behaviour

- Binary operators' standard behaviour.
- Numeric operators' standard behaviour.

### 4.3.2.7 Examples

3 - 2	results in	1
-7 - 3	results in	-10

## 4.3.3 Multiplication (\*)

### 4.3.3.1 Syntax

left \* right

### 4.3.3.2 Input parameters

left: rset | scal <num + interval>

right: rset | scal <num + interval>

### 4.3.3.3 Output

rset | scal <num + interval>

### 4.3.3.4 Semantics

Returns the product of two numbers.

For intervals:

- Centre is calculated as  $\text{centre}(\text{left}) * \text{centre}(\text{right})$
- Radius is calculated as  $(\text{centre}(\text{left}) * \text{radius}(\text{right})) + \text{abs}(\text{radius}(\text{left}) * \text{centre}(\text{right})) + (\text{radius}(\text{left}) * \text{radius}(\text{right}))$

### 4.3.3.5 Additional constraints

None.

### 4.3.3.6 Behaviour

- Binary operators' standard behaviour.
- Numeric operators' standard behaviour.

### 4.3.3.7 Examples

4 * 6	results in	24
-9 * 2	results in	-18

## 4.3.4 Division (/)

### 4.3.4.1 Syntax

num / den

### 4.3.4.2 Input parameters

num: rset | scal <num>

den: rset | scal <num>

### 4.3.4.3 Output

rset | scal <num>

### 4.3.4.4 *Semantics*

Divides two numbers.

For intervals:

- Centre is calculated as  $\text{centre}(\text{left}) / \text{centre}(\text{right})$
- Radius is calculated as  $\max((\text{centre}(\text{left}) + \text{radius}(\text{left})) / (\text{centre}(\text{right}) + \text{radius}(\text{right})), (\text{centre}(\text{left}) + \text{radius}(\text{left})) / (\text{centre}(\text{right}) - \text{radius}(\text{right})), (\text{centre}(\text{left}) - \text{radius}(\text{left})) / (\text{centre}(\text{right}) + \text{radius}(\text{right})), (\text{centre}(\text{left}) - \text{radius}(\text{left})) / (\text{centre}(\text{right}) - \text{radius}(\text{right})))$

### 4.3.4.5 *Additional constraints*

The denominator must be different to zero.

### 4.3.4.6 *Behaviour*

- Binary operators' standard behaviour.
- Numeric operators' standard behaviour.
- If the denominator is 0, generates a run-time error.

### 4.3.4.7 *Examples*

24 / 6	results in	4
-18 / 2	results in	-9

## 4.3.5 Maximum (max)

### 4.3.5.1 *Syntax*

**max**(op1, op2 {, op}\*)

### 4.3.5.2 *Input parameters*

- op1: rset | scal <num + interval>

- op2: rset | scal <num + interval>

### 4.3.5.3 *Output*

rset | scal <num + interval>

### 4.3.5.4 *Semantics*

Calculates the maximum value from a set of operands.

For intervals:

- Centre is calculated as the maximum centre value from all the intervals.
- Radius is the radius of the interval with maximum centre.

### 4.3.5.5 *Additional constraints*

None.

### 4.3.5.6 *Behaviour*

- For each pair of operands, the standard behaviour for binary operators applies

### 4.3.5.7 *Examples*

Max(1, 3, -5)	results in	3
Max(1, 3, null)	results in	null

## EBA and EIOPA Regular Use

### 4.3.6 Minimum (min)

#### 4.3.6.1 Syntax

**min**(op1, op2 {, op}\*)

#### 4.3.6.2 Input parameters

- op1: rset | scal <num + interval>

- op2: rset | scal <num + interval>

#### 4.3.6.3 Output

rset | scal <num + interval>

#### 4.3.6.4 Semantics

Calculates the minimum value from a set of operands.

For intervals:

- Centre is calculated as the minimum centre value from all the intervals.
- Radius is the radius of the interval with minimum centre.

#### 4.3.6.5 Additional constraints

None.

#### 4.3.6.6 Behaviour

- For each pair of operands, the standard behaviour for binary operators applies

#### 4.3.6.7 Examples

Min(1, 3, -5)                      results in                      -5

Min(1, 3, null)                      results in                      null

### 4.3.7 Power

#### 4.3.7.1 Syntax

**power**(base, exponent)

#### 4.3.7.2 Input parameters

num: rset | scal <num>

den: rset | scal <num>

#### 4.3.7.3 Output

rset | scal <num>

#### 4.3.7.4 Semantics

Raises the power to the exponent.

#### 4.3.7.5 Additional constraints

None.

#### 4.3.7.6 Behaviour

- Binary operators' standard behaviour.
- Numeric operators' standard behaviour.

#### 4.3.7.7 Examples

power(5,2) results in 25

power(5,-1) results in 0.2

power(-5, 3) results in -125

## 5 Comparison operators

### 5.1 Comparison operators' general behaviour

Comparison operators describe operations that compare the values of operands.

This operator group is based upon the [General behavior for binary operators](#).

#### 5.1.1 Input parameters

Operands that can be of Recordset or Scalar type. Must have the same data type. For all operators that accept numeric operands, intervals are allowed.

#### 5.1.2 Result type

Recordset or Scalar with type *Boolean*.

#### 5.1.3 Constraints

The operands for the comparison operations must be of the same type (considering implicit casting).

#### 5.1.4 Behaviour

If any operand is null, then the result is also null.

For comparison operators implying an order (>, >=, <, <=), the following rules apply:

- Boolean values: True is considered greater than false.
- Strings: Alphabetic order is followed.

### 5.2 Equal (=)

#### 5.2.1.1 Syntax

left = right

#### 5.2.1.2 Input parameters

left: rset | scal <\*>

right: rset | scal <\*>

#### 5.2.1.3 Output

rset | scal <boo>

#### 5.2.1.4 Semantics

Returns true if left is equal to right and false otherwise.

For intervals:  $\text{abs}(\text{centre}(\text{left}) - \text{centre}(\text{right})) \leq \text{radius}(\text{left}) + \text{radius}(\text{right})$ .

#### 5.2.1.5 Additional constraints

None.

## EBA and EIOPA Regular Use

### 5.2.1.6 Behaviour

- Binary operators' standard behaviour.
- Comparison operators' standard behaviour.

### 5.2.1.7 Examples

1 = 2	results in	False
3 = NULL	results in	NULL
"4" = "4"	results in	True

## 5.3 Not Equal (<>)

### 5.3.1.1 Syntax

left <> right

### 5.3.1.2 Input parameters

left: rset | scal <\*>

right: rset | scal <\*>

### 5.3.1.3 Output

rset | scal <boo>

### 5.3.1.4 Semantics

Returns false if left is equal to right and true otherwise.

For intervals:  $\text{abs}(\text{centre}(\text{left}) - \text{centre}(\text{right})) > \text{radius}(\text{left}) + \text{radius}(\text{right})$ .

### 5.3.1.5 Additional constraints

None.

### 5.3.1.6 Behaviour

- Binary operators' standard behaviour.
- Comparison operators' standard behaviour.

### 5.3.1.7 Examples

1 <> 2	results in	True
3 <> NULL	results in	NULL
"4" <> "4"	results in	False

## 5.4 Greater than (>)

### 5.4.1.1 Syntax

left > right

### 5.4.1.2 Input parameters

left: rset | scal <\*>

right: rset | scal <\*>

### 5.4.1.3 Output

rset | scal <boo>

## EBA and EIOPA Regular Use

### 5.4.1.4 *Semantics*

Returns true if left is greater than right and false otherwise.

For intervals:  $\text{centre}(\text{left}) > \text{centre}(\text{right}) - (\text{radius}(\text{left}) + \text{radius}(\text{right}))$ .

### 5.4.1.5 *Additional constraints*

None.

### 5.4.1.6 *Behaviour*

- Binary operators' standard behaviour.
- Comparison operators' standard behaviour.

### 5.4.1.7 *Examples*

12 > 2	results in	True
3 > NULL	results in	NULL
True > False	results in	True
"tez" > "test"	results in	True

## 5.5 Greater Equal than ( $\geq$ )

### 5.5.1.1 *Syntax*

left  $\geq$  right

### 5.5.1.2 *Input parameters*

left: rset | scal  $\langle * \rangle$

right: rset | scal  $\langle * \rangle$

### 5.5.1.3 *Output*

rset | scal  $\langle \text{boo} \rangle$

### 5.5.1.4 *Semantics*

Returns true if left is greater than or equal to right and false otherwise.

For intervals:  $\text{centre}(\text{left}) \geq \text{centre}(\text{right}) - (\text{radius}(\text{left}) + \text{radius}(\text{right}))$ .

### 5.5.1.5 *Additional constraints*

None.

### 5.5.1.6 *Behaviour*

- Binary operators' standard behaviour.
- Comparison operators' standard behaviour.

### 5.5.1.7 *Examples*

1 $\geq$ 2	results in	False
3 $\geq$ NULL	results in	NULL
"tez" $\geq$ "test"	results in	True

## 5.6 Less than ( $<$ )

### 5.6.1.1 *Syntax*

left  $<$  right



## EBA and EIOPA Regular Use

### 5.6.1.2 *Input parameters*

left: rset | scal <\*>

right: rset | scal <\*>

### 5.6.1.3 *Output*

rset | scal <boo>

### 5.6.1.4 *Semantics*

Returns true if left is smaller than right and false otherwise.

For intervals:  $\text{centre}(\text{left}) - \text{centre}(\text{right}) < \text{radius}(\text{left}) + \text{radius}(\text{right})$ .

### 5.6.1.5 *Additional constraints*

None.

### 5.6.1.6 *Behaviour*

- Binary operators' standard behaviour.
- Comparison operators' standard behaviour.

### 5.6.1.7 *Examples*

1 < 2	results in	True
3 < NULL	results in	NULL
"tea" < "test"	results in	True

## 5.7 Less Equal than (<=)

### 5.7.1.1 *Syntax*

left <= right

### 5.7.1.2 *Input parameters*

left: rset | scal <\*>

right: rset | scal <\*>

### 5.7.1.3 *Output*

rset | scal <boo>

### 5.7.1.4 *Semantics*

Returns true if left is smaller than or equal to right and false otherwise.

For intervals:  $\text{centre}(\text{left}) - \text{centre}(\text{right}) \leq \text{radius}(\text{left}) + \text{radius}(\text{right})$ .

### 5.7.1.5 *Additional constraints*

None.

### 5.7.1.6 *Behaviour*

- Binary operators' standard behaviour.
- Comparison operators' standard behaviour.

### 5.7.1.7 *Examples*

3 <= 2	results in	False
3 <= NULL	results in	NULL
"tea" <= "test"	results in	True

### 5.8 Element of (in)

#### 5.8.1.1 Syntax

**op in set**

#### 5.8.1.2 Input parameters

op: rset | scal <num>

set: scalar\_set <\*> | subcategory

#### 5.8.1.3 Output

rset | scal <boo>

#### 5.8.1.4 Semantics

Returns true if op belongs to the set and false otherwise.

#### 5.8.1.5 Additional constraints

op must be of the same data type as the values in the set (considering implicit casting)

#### 5.8.1.6 Behaviour

- Unary operators' standard behaviour.
- Comparison operators' standard behaviour.

#### 5.8.1.7 Examples

5 in {1,3, 5} results in True

"abc" in {"def", "ghi"} results in False

### 5.9 Match characters (match)

#### 5.9.1.1 Syntax

**match(op, pattern)**

#### 5.9.1.2 Input parameters

op: rset | scal <str>

pattern: scal <str>

#### 5.9.1.3 Output

rset | scal <boo>

#### 5.9.1.4 Semantics

Returns true if op matches the pattern, false otherwise.

#### 5.9.1.5 Additional constraints

pattern is a regex expression following the Python definition.

#### 5.9.1.6 Behaviour

- Unary operators' standard behaviour.
- Comparison operators' standard behaviour.

#### 5.9.1.7 Examples

Match("test", "[0-9]+") results in False

Match("1234", "[0-9]+") results in True

Match("hello", "[a-z]+") results in True

### 5.10 Is null (isnull)

#### 5.10.1.1 Syntax

**is\_null**(op)

#### 5.10.1.2 Input parameters

op: rset | scal <\*>

#### 5.10.1.3 Output

rset | scal <boo>

#### 5.10.1.4 Semantics

Returns true if the value of op is null, false otherwise.

#### 5.10.1.5 Additional constraints

None.

#### 5.10.1.6 Behaviour

- Unary operators' standard behaviour.
- Comparison operators' standard behaviour.

## 6 Logical

Logical operators describe operations involving two operands with Boolean data type. To deal with null values, all operations implements Kleene logic for logical operations (also known as Three-valued logic).

### 6.1 Conjunction (and)

#### 6.1.1.1 Syntax

left **and** right

#### 6.1.1.2 Input parameters

left: rset | scal <boo>

right: rset | scal <boo>

#### 6.1.1.3 Output

rset | scal <boo>

#### 6.1.1.4 Semantics

Returns true if both operands are true, otherwise false.

#### 6.1.1.5 Additional constraints

None.

#### 6.1.1.6 Behaviour

- Binary operators' standard behaviour.

	False	Null	True
False	False	False	False
Null	False	Null	Null
True	False	Null	True

### 6.2 Disjunction (or)

#### 6.2.1.1 Syntax

left **or** right

#### 6.2.1.2 Input parameters

left: rset | scal <boo>

right: rset | scal <boo>

#### 6.2.1.3 Output

rset | scal <boo>

#### 6.2.1.4 Semantics

Returns true if any operand is true, otherwise false.

#### 6.2.1.5 Additional constraints

None.

#### 6.2.1.6 Behaviour

- Binary operators' standard behaviour.

	False	Null	True
False	False	Null	True
Null	Null	Null	True
True	True	True	True

### 6.3 Exclusive disjunction (xor)

#### 6.3.1.1 Syntax

left **xor** right

#### 6.3.1.2 Input parameters

left: rset | scal <boo>

right: rset | scal <boo>

#### 6.3.1.3 Output

rset | scal <boo>

#### 6.3.1.4 Semantics

Returns true if one operand is true and the other is false, otherwise false.

#### 6.3.1.5 Additional constraints

None.

#### 6.3.1.6 Behaviour

- Binary operators' standard behaviour.

	False	Null	True
	False	Null	True
Null	Null	Null	Null
True	True	Null	False

## 6.4 Negation (not)

### 6.4.1.1 Syntax

**not** op

### 6.4.1.2 Input parameters

op: rset | scal <boo>

### 6.4.1.3 Output

rset | scal <boo>

### 6.4.1.4 Semantics

Returns true if op is false, and false if op is true.

### 6.4.1.5 Additional constraints

None.

### 6.4.1.6 Behaviour

- Binary operators' standard behaviour.

	Result
False	True
Null	Null
True	False

## 7 Aggregate operators

### 7.1 Aggregate operators' general behaviour

#### 7.1.1 Syntax

**aggregateOperator** (op {**group by** groupId {, groupId}\*})

#### 7.1.2 Input parameters

op: rset <\*>

groupId: scal <prop>

#### 7.1.3 Output

rset | scal <\*>

#### 7.1.4 Semantics

Aggregate operators perform operations on the measures of the operand recordset, calculating the required aggregated values for groups of records. The groups of records to be aggregated are specified through the grouping clause. If no grouping clause is used, the operation shall be calculated on all the records, resulting in a scalar.

#### 7.1.5 Additional constraints

The allowed data types depend on the specific operator according to the following table:

Operator	Operand type	Result type
Sum	Number	Number
Count	Any	Integer
Min	Any	Any
Max	Any	Any
Average	Number	Number
Median	Number	Number

The components in the grouping by clause shall be present in the operand.

#### 7.1.6 Behaviour

Aggregate operations generate a recordset or a scalar, depending on the grouping clause.

If the grouping clause exists, the structure of the resulting recordset has as key components the components in the group by

The result may be:

- A recordset, for which the resulting structure contains as key dimensions the components included in the group by clause.
- A scalar, if the grouping clause is omitted.

#### 7.1.7 Examples

Supposing the following recordset with name *rs1*:

r	c	CNT	f
010	010	PT	100
010	020	PT	200
020	010	PT	300
020	020	PT	400

010	010	DE	500
010	020	DE	600
020	010	DE	700
020	020	DE	800

sum(rs1 group by r) results in:

r	f
010	1400
020	2200

sum(rs1 group by r, CNT) results in:

r	CNT	f
010	PT	300
020	PT	700
010	DE	1100
020	DE	1500

count(rs1) results in: 8 (scalar)

## 7.2 Sum (sum)

### 7.2.1 Syntax

**sum**(op {**group by** groupingId {, groupingId}\*})

### 7.2.2 Input parameters

op: rset <num + interval>

groupingId: scal <prop>

### 7.2.3 Output

rset | scal <num>

### 7.2.4 Semantics

Returns the sum of the input values. Follows the general semantics of aggregate operators.

For intervals:

- The centre is calculated as the sum of the all the centers of the operands.
- The radius is calculated as the sum of all the radiuses of the operads.

### 7.2.5 Additional constraints

None.

### 7.2.6 Behaviour

Aggregate operators' general behaviour.

### 7.3 Count (count)

#### 7.3.1 Syntax

**count**(op {**group by** groupingId {, groupingId}\*})

#### 7.3.2 Input parameters

op: rset <\*>

groupingId: scal <prop>

#### 7.3.3 Output

rset | scal <num>

#### 7.3.4 Semantics

Returns the number of records in the recordset or groups of records. Follows the general semantics of aggregate operators.

#### 7.3.5 Additional constraints

None.

#### 7.3.6 Behaviour

Aggregate operators' general behaviour.

Note: Aggregate operators generally ignore null values. This behavior can be overridden by using the nvl operator.

### 7.4 Minimum value (min\_aggr)

#### 7.4.1 Syntax

**min\_aggr**(op {**group by** groupingId {, groupingId}\*})

#### 7.4.2 Input parameters

op: rset <\*>

groupingId: scal <prop>

#### 7.4.3 Output

rset | scal <num>

#### 7.4.4 Semantics

Returns the minimum value of the input values. Follows the general semantics of aggregate operators.

For intervals:

- The centre is calculated as the minimum value of the all the centers of the operands.
- The radius is the radius of the operand that has the minimum centre.

#### 7.4.5 Additional constraints

None.

#### 7.4.6 Behaviour

Aggregate operators' general behaviour.



## 7.5 Maximum value (max\_aggr)

### 7.5.1 Syntax

**max\_aggr**(op {**group by** groupId {, groupId}\*})

### 7.5.2 Input parameters

op: rset <\*>

groupId: scal <prop>

### 7.5.3 Output

rset | scal <num>

### 7.5.4 Semantics

Returns the maximum value of the input values. Follows the general semantics of aggregate operators.

For intervals:

- The centre is calculated as the maximum value of the all the centers of the operands.
- The radius is the radius of the operand that has the maximum centre.

### 7.5.5 Additional constraints

None.

### 7.5.6 Behaviour

Aggregate operators' general behaviour.

## 7.6 Average (avg)

### 7.6.1 Syntax

**avg**(op {**group by** groupId {, groupId}\*})

### 7.6.2 Input parameters

op: rset <num>

groupId: scal <prop>

### 7.6.3 Output

rset | scal <num>

### 7.6.4 Semantics

Returns the average of the input values. Follows the general semantics of aggregate operators.

### 7.6.5 Additional constraints

None.

### 7.6.6 Behaviour

Aggregate operators' general behaviour.

## 7.7 Median value (median)

### 7.7.1 Syntax

**median**(op {**group by** groupId {, groupId}\*})

## EBA and EIOPA Regular Use

### 7.7.2 Input parameters

op: rset <num>

groupingId: scal <prop>

### 7.7.3 Output

rset | scal <num>

### 7.7.4 Semantics

Returns the median of the input values. Follows the general semantics of aggregate operators.

### 7.7.5 Additional constraints

None.

### 7.7.6 Behaviour

Aggregate operators' general behaviour.

## 8 Conditional operators

### 8.1 If-then-else

#### 8.1.1 Syntax

**if** conditionExpression **then** thenExpression **{else** elseExpression **}**

#### 8.1.2 Input parameters

- conditionExpression: rset | scal <boo>

- thenExpression: rset | scal <\*>

- elseExpression: rset | scal <\*>

#### 8.1.3 Output

rset | scal <\*>

#### 8.1.4 Semantics

Returns the thenExpression if the conditionExpression evaluates to true, elseExpression otherwise.

#### 8.1.5 Additional constraints

The thenExpression and ElseExpression must be of the same data type or a compatible one. In case they are not of the same data type, there shall be implicit casting to the common data type, which will be the data type of the result

The thenExpression and elseExpression need to have the same data structure.

If the conditionExpression is of recordset type, the data structures of the thenExpression and elseExpression must contain the same identifiers as the conditionExpression, or a subset of them. The resulting structure will contain the components of the conditionExpression recordset.

The following table contains the applicability options for the if-then-else operator:

Condition	Then	Else	Is allowed	Result structure	Remarks
Scalar	Scalar	None	Yes	Scalar	
Scalar	Recordset	None	No		Forbidden because the result structure would be unknown
Scalar	Scalar	Scalar	Yes	Scalar	
Scalar	Scalar	Recordset	No		Forbidden because the result structure would be unknown
Scalar	Recordset	Scalar	No		Forbidden because the result structure would be unknown
Scalar	Recordset	Recordset	Yes	Recordset	
Recordset	Scalar	None	Yes	Recordset	
Recordset	Recordset	None	Yes	Recordset	
Recordset	Scalar	Scalar	Yes	Recordset	
Recordset	Scalar	Recordset	Yes	Recordset	
Recordset	Recordset	Scalar	Yes	Recordset	
Recordset	Recordset	Recordset	Yes	Recordset	

### 8.1.6 Behaviour

For conditionExpressions of the scalar type, returns the operand resulting from the thenExpression or the elseExpression.

For conditionExpressions of the recordset type, returns a recordset with the data structure of the conditionExpression, with one output record per input record. The fact values will be the corresponding for the thenExpression or elseExpression, depending on the evaluation result of the condition.

If the elseExpression is omitted, a null is returned, except for the case when the thenExpression is of the Boolean data type, in which case true is returned.

If the condition evaluates to null, the elseExpression is returned.

### 8.1.7 Examples

## 8.2 Null substitute (nvl)

### 8.2.1 Syntax

**nvl**(op1, op2)

### 8.2.2 Input parameters

- op1: rset | scal <\*>

- op2: rset | scal <\*>

### 8.2.3 Output

rset | scal <\*>

### 8.2.4 Semantics

Returns op2 when op1 is null, otherwise op1.

### 8.2.5 Additional constraints

op1 and op2 need to be of the same type (scalar or recordset) and same data type.

If op1 and op2 are of the recordset type, they need to have the same data structure.

### 8.2.6 Behaviour

- Binary operators' standard behaviour.

### 8.2.7 Examples

## 8.3 Filter

### 8.3.1 Syntax

**filter**(filteredOp, filteringOp)

### 8.3.2 Input parameters

- filteredOp: rset <\*>

- filteringOp: rset <bool>

### 8.3.3 Output

rset <\*>

### 8.3.4 Semantics

Returns the records of the filteredOp recordset for which their matching record in filteringOp evaluates to true.

### 8.3.5 Additional constraints

filteringOp must have the same key components as filteredOp or a subset of them.

### 8.3.6 Behaviour

Returns a recordset with the same data structure as filteredOp.

An inner join between filteredOp and filteringOp is performed. The records in filteredOp that match to a record in filteringOp with true value are returned.

### 8.3.7 Examples

## 9 String operators

### 9.1 Length (len)

#### 9.1.1 Syntax

**len(op)**

#### 9.1.2 Input parameters

- op: rset | scal <Str>

#### 9.1.3 Output

rset | scal <Int>

#### 9.1.4 Semantics

Returns the number of characters of the op string.

#### 9.1.5 Additional constraints

None.

#### 9.1.6 Behaviour

- Unary operators' standard behaviour.

#### 9.1.7 Examples

len("test")	results in	4
len(NULL)	results in	NULL

### 9.2 Concatenate (&)

#### 9.2.1 Syntax

**op1 & op2**

#### 9.2.2 Input parameters

- op1: rset | scal <Str>

- op2: rset | scal <Str>

#### 9.2.3 Output

rset | scal <Str>

## EBA and EIOPA Regular Use

### 9.2.4 Semantics

Concatenates two strings.

### 9.2.5 Additional constraints

None.

### 9.2.6 Behaviour

- Binary operators' standard behaviour.

### 9.2.7 Examples

"hello" & "world"	results in	"helloworld"
"test" & NULL	results in	NULL

## 10 Time operators

### 10.1 Time shift

#### 10.1.1 Syntax

**time\_shift**(*op*, *period*, *numberPeriods*, {*var*})

#### 10.1.2 Input parameters

- *op*: rset <\*> | scal <date>

- *period*: scal <Str>

- *numberPeriods*: scal <int>

- *var*: scal <prop>

#### 10.1.3 Output

rset <\*>

#### 10.1.4 Semantics

Changes the dates of the *var* component of the recordset *op* by adding (or subtracting) the *numberPeriods* of the *period* type.

#### 10.1.5 Additional constraints

The component *var* must belong to the recordset *op*, and has to be of *Time interval* type.

The *period* must have one of the following values:

- A for year
- S for semester
- Q for quarter
- M for month
- W for week
- D for day

#### 10.1.6 Behaviour

Returns a recordset with the same data structure and number of records as the input *op*. The dates for the component *var* are modified by adding the *numberPeriods* (subtracting, if negative) of the *period* type.

#### 10.1.7 Examples

Considering the following recordset, generated from the selection {tT1, r010-020, c010-020}:

RefDate	r	c	f
2022Q1	010	010	100
2022Q1	010	020	200
2022Q1	010	010	300
2022Q1	010	020	400
2022Q2	020	010	500
2022Q2	020	020	600
2022Q2	020	010	700
2022Q2	020	020	800

time\_shift({tT1, r010-020, c010-020}, RefDate, Q, 1) returns:

RefDate	R	c	f
2022Q2	010	010	100
2022Q2	010	020	200
2022Q2	010	010	300
2022Q2	010	020	400
2022Q3	020	010	500
2022Q3	020	020	600
2022Q3	020	010	700
2022Q3	020	020	800

time\_shift({tT1, r010-020, c010-020}, RefDate, Q, -1) returns:

RefDate	r	c	f
2021Q4	010	010	100
2021Q4	010	020	200
2021Q4	010	010	300
2021Q4	010	020	400
2022Q1	020	010	500
2022Q1	020	020	600
2022Q1	020	010	700
2022Q1	020	020	800

## 11 Clause operators

Clause operators serve to perform operations on the DPM key components of recordsets. Use of clause operators with Standard Key Components are not allowed.

### 11.1 where

#### 11.1.1 Syntax

op[**where** condition]



## EBA and EIOPA Regular Use

### 11.1.2 Input parameters

- op: rset <\*>

- condition: expression <boo>

### 11.1.3 Output

rset <\*>

### 11.1.4 Semantics

Filters a recordset based on the value of a key component.

### 11.1.5 Additional constraints

*Condition* must be a Boolean expression using as input key components of *op*.

### 11.1.6 Behaviour

Returns a recordset with the same data structure as the input operand, and with the records resulting from the evaluation of the filtering condition. When the *condition* evaluates to true, the record is kept, otherwise (including null), the record is not kept.

### 11.1.7 Examples

Considering the following recordset, generated from the selection {tT1, r010-020}:

RefDate	r	CNT	f
2022Q1	010	ES	100
2022Q1	010	PT	200
2022Q1	010	DE	300
2022Q1	010	IT	400
2022Q2	020	ES	500
2022Q2	020	PT	600
2022Q2	020	DE	700
2022Q2	020		800

{tT1, r010-020}[where CNT in {"ES", "PT"}] returns:

RefDate	r	CNT	f
2022Q1	010	ES	100
2022Q1	010	PT	200
2022Q2	020	ES	500
2022Q2	020	PT	600

## 11.2 rename

### 11.2.1 Syntax

op[**rename** compFrom **to** compTo]

### 11.2.2 Input parameters

- op: rset <\*>
- compFrom: scal <prop>
- compTo: scal <prop>

### 11.2.3 Output

rset <\*>

### 11.2.4 Semantics

Changes the name of a component.

### 11.2.5 Additional constraints

*compFrom* must belong to *op*.

*compFrom* must be a DPM key component (i.e., cannot refer to row, column or sheet).

*compTo* name cannot be already used in *op*.

*compTo* name cannot be the name of a standard key component ("r", "c" or "s").

### 11.2.6 Behaviour

Returns a recordset with the same data structure as the input operand, except the renamed components, which change name. All the records are kept.

### 11.2.7 Examples

Considering the following recordset, generated from the selection {tT1, r010-020}:

RefDate	r	CNT	f
2022Q1	010	ES	100
2022Q1	010	PT	200
2022Q1	010	DE	300
2022Q1	010	IT	400
2022Q2	020	ES	500
2022Q2	020	PT	600

2022Q2	020	DE	700
2022Q2	020		800

{tT1, r010-020}[rename CNT to country] returns:

RefDate	r	country	f
2022Q1	010	ES	100
2022Q1	010	PT	200
2022Q1	010	DE	300
2022Q1	010	IT	400
2022Q2	020	ES	500
2022Q2	020	PT	600
2022Q2	020	DE	700
2022Q2	020		800

### 11.3 get

#### 11.3.1 Syntax

op[get component]

#### 11.3.2 Input parameters

- op: rset <\*>

- component: scal <prop>

#### 11.3.3 Output

rset <\*>

#### 11.3.4 Semantics

Returns a recordset where the fact is the value of one of the key components.

#### 11.3.5 Additional constraints

*component* must belong to *op*.

#### 11.3.6 Behaviour

Returns a recordset with the same data structure as the input operand. The fact values the records are substituted for the value of the selected component.

Records with null fact value will be omitted.

#### 11.3.7 Examples

Considering the following recordset, generated from the selection {tT1, r010-020}:

## EBA and EIOPA Regular Use

RefDate	r	CNT	f
2022Q1	010	ES	100
2022Q1	010	PT	200
2022Q1	010	DE	300
2022Q1	010	IT	400
2022Q2	020	ES	500
2022Q2	020	PT	600
2022Q2	020	DE	700
2022Q2	020	IT	800

{tT1, r010-020}[get CNT] returns:

RefDate	r	CNT	f
2022Q1	010	ES	ES
2022Q1	010	PT	PT
2022Q1	010	DE	DE
2022Q1	010	IT	IT
2022Q2	020	ES	ES
2022Q2	020	PT	PT
2022Q2	020	DE	DE
2022Q2	020	IT	IT