

DPM-XL Operators

Version History

Latest Version: 0.9
Release Date: 2025-11-24

Version	Date	Description
0.9	2025-11-24	<p>Version 0.9</p> <p>Changes from 0.8:</p> <p>New Operators:</p> <ul style="list-style-type: none">- sub clause (11.4): Filters a Recordset based on a Key Component value and drops that Key Component <p>Selection Operators (with):</p> <ul style="list-style-type: none">- Extended with syntax to support where_expression: with partial_selection [where_expression]: expression- Added extensive examples demonstrating where clause inheritance and override behavior- sub clause inside expression overrides the with's where_expression <p>Conditional Operators (if-then-else):</p> <ul style="list-style-type: none">- Changed: Scalar condition + Recordset then + no else is now allowed (was forbidden) <p>Clause Operators:</p> <ul style="list-style-type: none">- rename syntax now supports multiple renames: [rename a to b, c to d] <p>Other fixes:</p> <ul style="list-style-type: none">- Fixed link to logical operators section- Fixed isnull behavior reference (unary, not binary)- Renamed power operator parameters from num/den to base/exponent- Lowercase examples (abs, max, min instead of Abs, Max, Min)- Various typo and formatting fixes
0.8	2025-11-24	<p>Version 0.8</p> <p>Changes from 0.7:</p> <p>General Behavior:</p> <ul style="list-style-type: none">- Added clarification that Standard Keys must contain the same values for binary operators <p>Selection Operators:</p> <ul style="list-style-type: none">- Fixed X/Y axis mapping for DPM-ML (X = Row, Y = Column)- Added clarification for range selections (alphabetical order of DPM Header codes) <p>Numeric Operators:</p> <ul style="list-style-type: none">- Improved division formula for intervals (added abs() to radius calculation) <p>Time Operators:</p> <ul style="list-style-type: none">- Updated time_shift syntax: changed parameter order to time_shift(op, period,

Version	Date	Description
		numberPeriods, var) - Clarified constraints for Scalar vs Recordset operands Information Model: - Added data type mapping table (DPM to DPM-XL types) - Clarified that a Recordset may have no records - Added section on implicit nulls generation for open/closed tables
0.7	2025-08-07	Latest version published in the old format

Table of Contents

1 Introduction

2 Common General Behaviours for the Operators

2.1 Unary Operators

2.1.1 Scalars

2.1.2 Recordsets

2.1.2.1 Constraints

2.1.2.2 Structure

2.1.2.3 Records

2.2 Binary Operators

2.2.1 Scalars

2.2.2 Recordset and Scalar

2.2.2.1 Examples

2.2.3 Recordsets

2.2.3.1 Constraints

2.2.3.2 Structure

2.2.3.3 Records

2.2.3.4 Examples

2.2.3.4.1 Same Identifiers

2.2.3.4.2 Subset of Identifiers

2.2.4 Intervals Treatment

3 Selection Operators

3.1 Selection Operator

3.1.1 Syntax

3.1.2 Input Parameters

3.1.3 Output

3.1.4 Semantics

3.1.5 Additional Constraints

3.1.6 Behaviour

3.1.7 DPM-ML Metamodel Representation

3.2 With

3.2.1 Syntax

3.2.2 Input Parameters

3.2.3 Output

3.2.4 Semantics

3.2.5 Additional Constraints

3.2.6 Behaviour

3.2.7 Examples

Example 1

Example 2

Example 3

Example 4

Example 5

Example 6

Example 7

Example 8

Example 9

Example 10

3.2.8 Metamodel Representation

4 Numeric Operators

4.1 Numeric Operators' General Behaviour

4.1.1 Input Parameters

4.1.2 Result Type

4.1.3 Behaviour

4.2 Unary Arithmetic Operators

4.2.1 Plus (+)

4.2.1.1 Syntax

4.2.1.2 Input Parameters

4.2.1.3 Output

4.2.1.4 Semantics

4.2.1.5 Additional Constraints

4.2.1.6 Behaviour

4.2.1.7 Examples

4.2.2 Minus (-)

4.2.2.1 Syntax

4.2.2.2 Input Parameters

4.2.2.3 Output

4.2.2.4 Semantics

4.2.2.5 Additional Constraints

4.2.2.6 Behaviour

4.2.2.7 Examples

4.2.3 Absolute Value (abs)

4.2.3.1 Syntax

4.2.3.2 Input Parameters

4.2.3.3 Output

4.2.3.4 Semantics

4.2.3.5 Additional Constraints

- 4.2.3.6 Behaviour
 - 4.2.3.7 Examples
- 4.2.4 Exponential (exp)
 - 4.2.4.1 Syntax
 - 4.2.4.2 Input Parameters
 - 4.2.4.3 Output
 - 4.2.4.4 Semantics
 - 4.2.4.5 Additional Constraints
 - 4.2.4.6 Behaviour
 - 4.2.4.7 Examples
- 4.2.5 Natural Logarithm (ln)
 - 4.2.5.1 Syntax
 - 4.2.5.2 Input Parameters
 - 4.2.5.3 Output
 - 4.2.5.4 Semantics
 - 4.2.5.5 Additional Constraints
 - 4.2.5.6 Behaviour
 - 4.2.5.7 Examples
- 4.2.6 Square Root (sqrt)
 - 4.2.6.1 Syntax
 - 4.2.6.2 Input Parameters
 - 4.2.6.3 Output
 - 4.2.6.4 Semantics
 - 4.2.6.5 Additional Constraints
 - 4.2.6.6 Behaviour
 - 4.2.6.7 Examples
- 4.2.7 Logarithm (log)
 - 4.2.7.1 Syntax
 - 4.2.7.2 Input Parameters
 - 4.2.7.3 Output
 - 4.2.7.4 Semantics
 - 4.2.7.5 Additional Constraints
 - 4.2.7.6 Behaviour
 - 4.2.7.7 Examples
- 4.3 Binary Arithmetic Operators
 - 4.3.1 Addition (+)
 - 4.3.1.1 Syntax
 - 4.3.1.2 Input Parameters
 - 4.3.1.3 Output
 - 4.3.1.4 Semantics
 - 4.3.1.5 Additional Constraints

- 4.3.1.6 Behaviour
 - 4.3.1.7 Examples
- 4.3.2 Subtraction (-)
 - 4.3.2.1 Syntax
 - 4.3.2.2 Input Parameters
 - 4.3.2.3 Output
 - 4.3.2.4 Semantics
 - 4.3.2.5 Additional Constraints
 - 4.3.2.6 Behaviour
 - 4.3.2.7 Examples
- 4.3.3 Multiplication (*)
 - 4.3.3.1 Syntax
 - 4.3.3.2 Input Parameters
 - 4.3.3.3 Output
 - 4.3.3.4 Semantics
 - 4.3.3.5 Additional Constraints
 - 4.3.3.6 Behaviour
 - 4.3.3.7 Examples
- 4.3.4 Division (/)
 - 4.3.4.1 Syntax
 - 4.3.4.2 Input Parameters
 - 4.3.4.3 Output
 - 4.3.4.4 Semantics
 - 4.3.4.5 Additional Constraints
 - 4.3.4.6 Behaviour
 - 4.3.4.7 Examples
- 4.3.5 Maximum (max)
 - 4.3.5.1 Syntax
 - 4.3.5.2 Input Parameters
 - 4.3.5.3 Output
 - 4.3.5.4 Semantics
 - 4.3.5.5 Additional Constraints
 - 4.3.5.6 Behaviour
 - 4.3.5.7 Examples
- 4.3.6 Minimum (min)
 - 4.3.6.1 Syntax
 - 4.3.6.2 Input Parameters
 - 4.3.6.3 Output
 - 4.3.6.4 Semantics
 - 4.3.6.5 Additional Constraints
 - 4.3.6.6 Behaviour

4.3.6.7 Examples

4.3.7 Power

4.3.7.1 Syntax

4.3.7.2 Input Parameters

4.3.7.3 Output

4.3.7.4 Semantics

4.3.7.5 Additional Constraints

4.3.7.6 Behaviour

4.3.7.7 Examples

5 Comparison Operators

5.1 Comparison Operators' General Behaviour

5.1.1 Input Parameters

5.1.2 Result Type

5.1.3 Constraints

5.1.4 Behaviour

5.2 Equal (=)

5.2.1.1 Syntax

5.2.1.2 Input Parameters

5.2.1.3 Output

5.2.1.4 Semantics

5.2.1.5 Additional Constraints

5.2.1.6 Behaviour

5.2.1.7 Examples

5.3 Not Equal (!=)

5.3.1.1 Syntax

5.3.1.2 Input Parameters

5.3.1.3 Output

5.3.1.4 Semantics

5.3.1.5 Additional Constraints

5.3.1.6 Behaviour

5.3.1.7 Examples

5.4 Greater than (>)

5.4.1.1 Syntax

5.4.1.2 Input Parameters

5.4.1.3 Output

5.4.1.4 Semantics

5.4.1.5 Additional Constraints

5.4.1.6 Behaviour

5.4.1.7 Examples

5.5 Greater Equal than (>=)

- 5.5.1.1 Syntax
- 5.5.1.2 Input Parameters
- 5.5.1.3 Output
- 5.5.1.4 Semantics
- 5.5.1.5 Additional Constraints
- 5.5.1.6 Behaviour
- 5.5.1.7 Examples
- 5.6 Less than (<)
 - 5.6.1.1 Syntax
 - 5.6.1.2 Input Parameters
 - 5.6.1.3 Output
 - 5.6.1.4 Semantics
 - 5.6.1.5 Additional Constraints
 - 5.6.1.6 Behaviour
 - 5.6.1.7 Examples
- 5.7 Less Equal than (<=)
 - 5.7.1.1 Syntax
 - 5.7.1.2 Input Parameters
 - 5.7.1.3 Output
 - 5.7.1.4 Semantics
 - 5.7.1.5 Additional Constraints
 - 5.7.1.6 Behaviour
 - 5.7.1.7 Examples
- 5.8 Element of (in)
 - 5.8.1.1 Syntax
 - 5.8.1.2 Input Parameters
 - 5.8.1.3 Output
 - 5.8.1.4 Semantics
 - 5.8.1.5 Additional Constraints
 - 5.8.1.6 Behaviour
 - 5.8.1.7 Examples
- 5.9 Match Characters (match)
 - 5.9.1.1 Syntax
 - 5.9.1.2 Input Parameters
 - 5.9.1.3 Output
 - 5.9.1.4 Semantics
 - 5.9.1.5 Additional Constraints
 - 5.9.1.6 Behaviour
 - 5.9.1.7 Examples
- 5.10 Is null (isnull)

- 5.10.1.1 Syntax
- 5.10.1.2 Input Parameters
- 5.10.1.3 Output
- 5.10.1.4 Semantics
- 5.10.1.5 Additional Constraints
- 5.10.1.6 Behaviour

6 Logical Operators

6.1 Conjunction (and)

- 6.1.1.1 Syntax
- 6.1.1.2 Input Parameters
- 6.1.1.3 Output
- 6.1.1.4 Semantics
- 6.1.1.5 Additional Constraints
- 6.1.1.6 Behaviour

6.2 Disjunction (or)

- 6.2.1.1 Syntax
- 6.2.1.2 Input Parameters
- 6.2.1.3 Output
- 6.2.1.4 Semantics
- 6.2.1.5 Additional Constraints
- 6.2.1.6 Behaviour

6.3 Exclusive Disjunction (xor)

- 6.3.1.1 Syntax
- 6.3.1.2 Input Parameters
- 6.3.1.3 Output
- 6.3.1.4 Semantics
- 6.3.1.5 Additional Constraints
- 6.3.1.6 Behaviour

6.4 Negation (not)

- 6.4.1.1 Syntax
- 6.4.1.2 Input Parameters
- 6.4.1.3 Output
- 6.4.1.4 Semantics
- 6.4.1.5 Additional Constraints
- 6.4.1.6 Behaviour

7 Aggregate Operators

7.1 Aggregate Operators' General Behaviour

- 7.1.1 Syntax
- 7.1.2 Input Parameters

- 7.1.3 Output
 - 7.1.4 Semantics
 - 7.1.5 Additional Constraints
 - 7.1.6 Behaviour
 - 7.1.7 Examples
- 7.2 Sum (sum)
 - 7.2.1 Syntax
 - 7.2.2 Input Parameters
 - 7.2.3 Output
 - 7.2.4 Semantics
 - 7.2.5 Additional Constraints
 - 7.2.6 Behaviour
- 7.3 Count (count)
 - 7.3.1 Syntax
 - 7.3.2 Input Parameters
 - 7.3.3 Output
 - 7.3.4 Semantics
 - 7.3.5 Additional Constraints
 - 7.3.6 Behaviour
- 7.4 Minimum Value (min_aggr)
 - 7.4.1 Syntax
 - 7.4.2 Input Parameters
 - 7.4.3 Output
 - 7.4.4 Semantics
 - 7.4.5 Additional Constraints
 - 7.4.6 Behaviour
- 7.5 Maximum Value (max_aggr)
 - 7.5.1 Syntax
 - 7.5.2 Input Parameters
 - 7.5.3 Output
 - 7.5.4 Semantics
 - 7.5.5 Additional Constraints
 - 7.5.6 Behaviour
- 7.6 Average (avg)
 - 7.6.1 Syntax
 - 7.6.2 Input Parameters
 - 7.6.3 Output
 - 7.6.4 Semantics
 - 7.6.5 Additional Constraints
 - 7.6.6 Behaviour

7.7 Median Value (median)

7.7.1 Syntax

7.7.2 Input Parameters

7.7.3 Output

7.7.4 Semantics

7.7.5 Additional Constraints

7.7.6 Behaviour

8 Conditional Operators

8.1 If-then-else

8.1.1 Syntax

8.1.2 Input Parameters

8.1.3 Output

8.1.4 Semantics

8.1.5 Additional Constraints

8.1.6 Behaviour

8.1.7 Examples

8.2 Null Substitute (nvl)

8.2.1 Syntax

8.2.2 Input Parameters

8.2.3 Output

8.2.4 Semantics

8.2.5 Additional Constraints

8.2.6 Behaviour

8.2.7 Examples

8.3 Filter

8.3.1 Syntax

8.3.2 Input Parameters

8.3.3 Output

8.3.4 Semantics

8.3.5 Additional Constraints

8.3.6 Behaviour

8.3.7 Examples

9 String Operators

9.1 Length (len)

9.1.1 Syntax

9.1.2 Input Parameters

9.1.3 Output

9.1.4 Semantics

9.1.5 Additional Constraints

9.1.6 Behaviour

9.1.7 Examples

9.2 Concatenate (&)

9.2.1 Syntax

9.2.2 Input Parameters

9.2.3 Output

9.2.4 Semantics

9.2.5 Additional Constraints

9.2.6 Behaviour

9.2.7 Examples

10 Time Operators

10.1 Time Shift

10.1.1 Syntax

10.1.2 Input Parameters

10.1.3 Output

10.1.4 Semantics

10.1.5 Additional Constraints

10.1.6 Behaviour

10.1.7 Examples

11 Clause Operators

11.1 where

11.1.1 Syntax

11.1.2 Input Parameters

11.1.3 Output

11.1.4 Semantics

11.1.5 Additional Constraints

11.1.6 Behaviour

11.1.7 Examples

11.2 rename

11.2.1 Syntax

11.2.2 Input Parameters

11.2.3 Output

11.2.4 Semantics

11.2.5 Additional Constraints

11.2.6 Behaviour

11.2.7 Examples

11.3 get

11.3.1 Syntax

11.3.2 Input Parameters

11.3.3 Output

11.3.4 Semantics

11.3.5 Additional Constraints

11.3.6 Behaviour

11.3.7 Examples

11.4 sub

11.4.1 Syntax

11.4.2 Input Parameters

11.4.3 Output

11.4.4 Semantics

11.4.5 Additional Constraints

11.4.6 Behaviour

11.4.7 Examples

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

2. the Key Components of one Recordset (Reference Recordset) are a superset of the Key Components of the other Recordset.
3. Standard Keys contain the same values. E.g., summing {tT1, r010-020} and {tT1, r030-040} is not possible, because there are no matches in the operands.

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}

INC	f
123	1000
456	2000
789	3000

{C 28.00 c190}

INC	f
123	-100
456	-200
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} >= {F 40.02c0060}
```

Supposing that the selections yield:

{F 40.01 c0110}

LIN	TYC	f
123	x1	1
456	x1	0.8
789	x1	0.4

{F 40.02 c0060}

LIN	TYC	STC	LHC	LHO	f
123	x1	111	ABC	x1	0.3
123	x1	111	DEF	x1	0.7
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 `>=` 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 | vvariable | ooperation]1  
  [rrow [, rrow] | rrow_init-row_end | r* ]  
  [ccol [, ccol] | ccol_init-col_end | c* ]  
  [ssheet [, ssheet] | ssheet_init-sheet_end | 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 Row for DPM-XL.
- Y: If there is Column 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.

For ranges (e.g., r0010-0100), the selection is done based on the alphabetical order of the DPM Header codes. In the example, it selects all the Headers with a code between 0010 and 0100.

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 [where_expression]¹: expression

3.2.2 Input Parameters

- **partial_selection**: A selection expression (see Selection Operator)
- **expression**: An expression including selection Operators
- **where_expression**: An expression with a `where` operator

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 [selection operators](#) in the expression.

3.2.5 Additional Constraints

In case where_expression is used:

- All the operands rising from selection operators have a `where` or a `sub` clause applicable, the one in the with, or the one overriding it. This implies that it is not possible to have a selection with a table that does not have open keys.

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.

In case where `where_expression` is used, the overriding mechanism works as follows:

- Any `where` clause used inside the expression fully overrides the `where_expression`, even if the properties used in the `where` clause inside the expression are different.
- A `sub` clause used inside the expression fully overrides the `where_expression`.

3.2.7 Examples

Example 1

```
with {F 01.01 c0010, default:0, interval:false}:  
{r0010} = {r0020} + {r0030} + {r0040}
```

The partial selection applies to all the selections in the expression

Example 2

```
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

Example 3

```
with {F 01.01 c0010, default:0, interval:false}:  
{F 01.01 r0010} + {F 01.01 r0040} = {F04.01 r0010, c0010 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.

Example 4

```
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.

Example 5

```
with {tEXM, r0010}[where id1="ABC" and id2=5]:  
{c0010} = {c0020} + {c0030}
```

Where:

- **tEXM** has two Key Components: id1 and id2

The clauses applying to each selection are:

Selection	Clause
{c0010}	[where id1="ABC" and id2=5]
{c0020}	[where id1="ABC" and id2=5]
{c0030}	[where id1="ABC" and id2=5]

Example 6

```
with {tEXM, r0010}[where id1="ABC" and id2=5]:  
  {c0010} = {c0020} + {tEXM_02, c0030}[where id1="ABC"]
```

Where:

- **tEXM** has two Key Components: id1 and id2
- **tEXM_02** has one Key Component2: id1

The clauses applying to each selection are:

Selection	Clause
{c0010}	[where id1="ABC" and id2=5]
{c0020}	[where id1="ABC" and id2=5]
{tEXM_02, c0030}	[where id1="ABC"]

Example 7

```
with {r0010}[where id1="ABC"]:  
  {tEXM, c0010} = {tEXM, c0020} + {tEXM_02, c0030}
```

Where:

- **tEXM** has two Key Components: id1 and id2
- **tEXM_02** has one Key Component: id1

The clauses applying to each selection are:

Selection	Clause
{tEXM, c0010}	[where id1="ABC"]
{tEXM, c0020}	[where id1="ABC"]
{tEXM_02, c0030}	[where id1="ABC"]

Example 8

```
with {default: 0, interval: true}[where qEEA = [eba_qAE:qx2010]]:  
if  
    sum({tC_08.01.a, r0070, c0255})  
    =  
    sum({tC_08.01.a, r0010, c0255})  
then  
    sum  
    ( filter ({tC_08.02, c0255}, {tC_08.02, c0010} = 1))  
    =  
    sum({tC_09.02, r0090, c0120} [where CEG = [eba_GA:qx2014]])  
endif
```

Where:

- **C_08.01.a** has one Key Component: qEEA
- **C_08.02** has two Key Components: qEEA and OGR
- **C_09.02** has oneKey Component: CEG

The clauses applying to each selection are:

Selection	Clause
{tC_08.01.a, r0070, c0255}	[where qEEA = [eba_qAE:qx2010]]
{tC_08.01.a, r0010, c0255}	[where qEEA = [eba_qAE:qx2010]]
{tC_08.02, c0255}	[where qEEA = [eba_qAE:qx2010]]
{tC_08.02, c0010}	[where qEEA = [eba_qAE:qx2010]]
{tC_09.02, r0090, c0120}	[where CEG = [eba_GA:qx2014]]

Example 9

```
with {tEXM, r0010}[where id1="ABC" and id2=5]:  
    {c0010} = {c0020} + {tEXM_02, c0030}[sub id1="ABC"]
```

Where:

- **tEXM** has two Key Components: id1 and id2
- **tEXM_02** has one Key Component2: id1

The clauses applying to each selection are:

Selection	Clause
{c0010}	[where id1="ABC" and id2=5]
{c0020}	[where id1="ABC" and id2=5]
{tEXM_02, c0030}	[sub id1="ABC"]

Note that the `sub` clause overrides the `where` in the `with` in the `partial_selection`.

Example 10

```
with {tEXM, r0010}[where id1="ABC" and id2=5]:  
  {c0010} = {c0020} + {tEXM_03, c0030}
```

Where:

- **tEXM** has two Key Components: `id1` and `id2`
- **tEXM_03** does not have any key component

This expression raises a Semantic Error, because the `where` clause applies to the operand `{tEXM_03, c0030}`, and this table doesn't have the referred Key Components.

3.2.8 Metamodel Representation

See selection Operator.

4 Numeric Operators

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

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)`

4.2.4.2 Input Parameters

| Op: rset | scal <num>

4.2.4.3 Output

| rset | scal

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>

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](#).

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 `centre(left) + centre(right)`
- Radius is calculated `radius(left) + radius(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 `centre(left) - centre(right)`
- Radius is calculated `radius(left) + radius(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 `centre(left) * centre(right)`
- Radius is calculated as `(centre(left) * radius(right)) + abs(radius(left) * centre(right)) + (radius(left) * radius(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 `centre(left) / centre(right)` - Radius is calculated as:

```
max(  
  abs(centre(left) + radius(left)) / (centre(right) + radius(right)),  
  abs(centre(left) + radius(left)) / (centre(right) - radius(right)),  
  abs(centre(left) - radius(left)) / (centre(right) + radius(right)),  
  abs(centre(left) - radius(left)) / (centre(right) - radius(right))  
)
```

4.3.4.5 Additional Constraints

The denominator must be different from 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`

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

| base: rset | scal <num>

| exponent: 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: `abs(centre(left) - centre(right)) <= radius(left) + radius(right)`.

5.2.1.5 Additional Constraints

None.

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: `abs(centre(left) - centre(right)) > radius(left) + radius(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>

5.4.1.4 Semantics

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

For intervals: `centre(left) > centre(right) - (radius(left) + radius(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 (`>=`)

5.5.1.1 Syntax

| `left >= right`

5.5.1.2 Input Parameters

| `left: rset | scal <*>`
| `right: rset | scal <*>`

5.5.1.3 Output

| `rset | scal <boo>`

5.5.1.4 Semantics

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

For intervals: `centre(left) >= centre(right) - (radius(left) + radius(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 >= 2` results in `false`
- `3 >= null` results in `null`
- `"tez" >= "test"` results in `true`

5.6 Less than (<)

5.6.1.1 Syntax

| left < right

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: `centre(left) - centre(right) < radius(left) + radius(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: `centre(left) - centre(right) <= radius(left) + radius(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 <*>

| 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

- Binary 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

- Binary 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

`isnull(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 Operators

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	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.

Input	Result
false	true
null	null
true	false

7 Aggregate Operators

7.1 Aggregate Operators' General Behaviour

7.1.1 Syntax

aggregateOperator (op {**group by** groupingId {, groupingId}*})

7.1.2 Input Parameters

op: rset <*>
groupingId: 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 Operands.

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 groupingId {, groupingId}*})

7.5.2 Input Parameters

| op: rset <*>
| groupingId: 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** groupingId {, groupingId}*})

7.6.2 Input Parameters

| op: rset <num>
| groupingId: 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** groupingId {, groupingId}*})

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} **endif**

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	Yes	-	Recordset, with the structure of

Condition	Then	Else	Is allowed	Result structure	Remarks
					the thenExpression applying as well to the else
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>

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

- If op is a Recordset, then
- var must exist,
- must belong to the Recordset op, and
- must be of Time Interval type.
- If var is an Scalar, then var cannot exist.

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}:

refPeriod	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}, Q, 1, refPeriod)` returns:

refPeriod	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}, Q, -1, refPeriod)` returns:

refPeriod	r	c	f
2021Q4	010	010	100
2021Q4	010	020	200
2021Q4	010	010	300

refPeriod	r	c	f
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]

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}` :

refPeriod	r	CNT	f
2022Q1	010	ES	100

refPeriod	r	CNT	f
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:

refPeriod	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 {, 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}` :

refPeriod	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:

refPeriod	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} :

refPeriod	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:

refPeriod	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

11.4 sub

11.4.1 Syntax

| op [**sub** key_component = value {, key_component = value}]*

11.4.2 Input Parameters

| op: rset <*>
 | key_component: scal <prop>
 | value: scal <*>

11.4.3 Output

| rset <*>

11.4.4 Semantics

Filters a Recordset based on the value of a Key Component, and then drops the Key Component.

11.4.5 Additional Constraints

The value needs to have the same Data type as the key_component.

11.4.6 Behaviour

Returns a Recordset with the same data structure as the input Operand, except the Key Components dropped, 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.

11.4.7 Examples

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

refPeriod	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}[sub CNT = "PT"]` returns:

refPeriod	r	f
2022Q1	010	200
2022Q2	020	600