

Aggregation operator

- [Parameter](#)
- [Aggregation Functions](#)
- [Examples](#)
 - [First](#)
 - [Last](#)
 - [Changing the way this operator outputs values](#)
- [Further information](#)

The aggregation operator is an alternative implementation of [Aggregate \(and Group\) operator](#). In particular for sliding time windows with advance of 1 this operator is faster than the implementation with partial aggregates.

Differences in the use of this operator compared to [Aggregate \(and Group\) operator](#):

- This operator has a more flexible PQL interface that allows to specify key value parameters.
- This operator does not set end timestamps of the resulting data stream elements. If you need the validity of the aggregation value you need to append an element window of size 1.
- This operator outputs "empty aggregations" if no tuple is valid at a specific point in time. E.g., the sum aggregation function would output 0. This is necessary to determine the end timestamp with a subsequent element window.

These aggregation functions are still in development. Especially the keys for the parameters are preliminary and subject to change.

Parameter

- `group_by`: An optional list of attributes over which the grouping should occur.
- `aggregations`: A list of aggregate functions (see below).
- `SUPPRESS_FULL_META_DATA_HANDLING`: Boolean flag set to true if the handling of meta data other than Time Interval (e.g. Latency) should be suppressed.

The following optional boolean parameters control when a new aggregation value is transferred (see below for useful examples):

- `eval_at_new_element`: Outputs an updated aggregation value when a new element gets valid. In the case that more than one element gets valid at the same time (same start timestamp), this operator outputs for each element an output value in the order of arrival. The default value is `true`.
- `eval_at_outdating`: Outputs an updated aggregation value when one or more elements gets invalid with the value after the removal of the invalid elements. The default value is `true`.
- `eval_before_remove_outdating`: Outputs an updated aggregation value before removing the invalid elements instead of after removal. The default value is `false`.
- `eval_at_done`: Outputs the value at the time the operator gets the done signal. The default value is `false`.
- `output_only_changes`: Suppresses elements that are equal to the previous outputted element. The default value is `false`. If you want to use this, make sure the equals-method for every attribute type is implemented.

Aggregation Functions

Function Name	Description	Parameters				Examples
Count	Outputs the number of stream elements.	Name	Description	Default Value	Optional?	<pre>['FUNCTION' = 'Count'] ['FUNCTION' = 'Count', 'OUTPUT_ATTRIBUTES' = 'number_of_elements']</pre>
		OUTPUT_ATTRIBUTES	The name for the output attribute.	count	True	
Sum	Outputs the sum of elements.	Name	Description	Default Value	Optional?	<pre>['FUNCTION' = 'Sum'] ['FUNCTION' = 'Sum', 'INPUT_ATTRIBUTES' = 'value1'] ['FUNCTION' = 'Sum', 'INPUT_ATTRIBUTES' = ['value1', 'value2']]</pre>
		INPUT_ATTRIBUTES	The single string or a list of the name(s) of the attribute(s) in the input tuples. By default, all input attributes are used. This could raise an error if attributes are not numeric.	(all attributes)	True	
		OUTPUT_ATTRIBUTES	A single string or list of output attributes. By default, the string "Sum_" concatenated with the original input attribute name is used.	"Sum_" + input attribute name	True	
Avg	Average value (mean)	TODO (similar to Sum)				

Min	Min value	TODO (similar to Sum)			
Max	Max value	TODO (similar to Sum)			
First	The first element of a window. See example below.	Name	Description	Default Value	Optional?
		OUTPUT_ATTRIBUTES	The name for the output attribute.	first	True
Last	The last element of a window. See example below.	Name	Description	Default Value	Optional?
		OUTPUT_ATTRIBUTES	The name for the output attribute.	last	True
Trigger	The tuple that triggers the output.	TODO			
Variance	Calculates the variance	TODO (similar to Sum)			
TopK	Calculates the top-K list	TODO			
Nest	Nests the valid elements as list.	TODO			

You should use the following settings:

```
output_only_changes = true
```

This results in getting the first element in each window. Especially useful with a tumbling window.

You should use the following settings:

```
EVAL_AT_NEW_ELEMENT = false
EVAL_BEFORE_REMOVE_OUTPUT = true
```

This results in getting the last element in each window. Especially useful with a tumbling window.

Examples

```
counted = AGGREGATION({AGGREGATIONS = [['FUNCTION' = 'Count']], GROUP_BY = ['publisher', 'item']], windowed)
```

You can use more than one aggregation function:

```
counted = AGGREGATION({AGGREGATIONS = [['FUNCTION' = 'Count'], ['FUNCTION' = 'Sum', 'INPUT_ATTRIBUTES' = 'value1']], GROUP_BY = ['publisher', 'item']], windowed)
```

```
/// count the number of items for each publisher
counted = AGGREGATION({AGGREGATIONS = [['FUNCTION' = 'Count']], GROUP_BY = ['publisher', 'item']], windowed)
/// aggregate the 100 most frequent items for each publisher to an ordered list
TopKItemsByPublisher ::= AGGREGATION({AGGREGATIONS = [
  [
    'FUNCTION' = 'TopK',
    'TOP_K' = '100', // number of items
    'SCORING_ATTRIBUTES' = 'Count', // the attribute name that defines the order
    'INPUT_ATTRIBUTES' = 'item', // do not use the whole input tuple, just use the
'item' attribute for creating the output top-k set
    'MIN_SCORE' = '0', // remove items that reaches a score of 0 (due to the
previous aggregation these are all items that has no valid tuple)
    'UNIQUE_ATTR'='item', // use 'item' as a unique attribute. that means, a new
tuple with an known items id replaces the previous value. (this is some kind of element window in this operator)
    'descending' = true, // default is true. If you want to
have the smallest elements, use 'false', if you want to have the biggest elements, use 'true'
    'ALWAYS_OUTPUT' = true // If set to false (default),
'null' is put out instead of the result if the result is equal to the previous result.
  ]], GROUP_BY = ['publisher']}, counted)
```

First

Here, we use a tumbling window with the "First" aggregate function to only get the first element per 5-minute window.

```

// Tumbling window
tumbling = TIMEWINDOW({
    size = [5, 'MINUTES'],
    advance = [5,
'MINUTES']
    },
    selectCenter
)

// Select first of tumbling
reduce = AGGREGATION({
    aggregations = [['FUNCTION' = 'First']],
    output_only_changes = true,
    group_by = ['movingObjectId']
    },
    tumbling
)

// Remove the grouping id (because it will be in the unnested tuple)
withoutId = PROJECT({
    attributes = ['first']
    },
    reduce
)

// Unnest the tuple
output = UNNEST({
    attribute='first'
    },
    withoutId
)

```

Last

Here, we use a tumbling window and the "Last" aggregate function to only get the last element per 5-minute window.

```

// Tumbling window
tumbling = TIMEWINDOW({
    size = [5, 'MINUTES'],
    advance = [5,
'MINUTES']
    },
    selectCenter
)

// Select last of tumbling
reduce = AGGREGATION({
    aggregations = [['FUNCTION' = 'Last']],
    group_by = ['movingObjectId']
    },
    tumbling
)

// Remove the grouping id (because it will be in the unnested tuple)
withoutId = PROJECT({
    attributes = ['last']
    },
    reduce
)

// Unnest the tuple
output = UNNEST({
    attribute='last'
    },
    withoutId
)

```

Changing the way this operator outputs values

By using the default values, this operator act as [Aggregate \(and Group\) operator](#) (with the limitations explained above). Useful alternative settings are:

- Set `eval_at_new_element` to `false` and `eval_before_remove_outdating` to `true` and add a preceding window with `advance`.

Remark: In this case, the starttime of the output gets the timestamp of the value, that triggers the output (i.e. the element that states, that the current elements are outdated).

The following example calculates the number of elements in the stream impressions in one minute. It outputs the total number at the end of each minute instead of each update when a new item arrives.

```

windowed = TIMEWINDOW({size = [1, 'Minutes'], ADVANCE = [1, 'MINUTES']}, impressions)
impressions_per_minute = AGGREGATION({AGGREGATIONS = [['FUNCTION' = 'Count']], EVAL_AT_NEW_ELEMENT = false,
EVAL_BEFORE_REMOVE_OUTDATING = true}, windowed)

```

Further information

[How to create aggregation functions \(in german\)](#)