Handling of Queries

#QUERY

This command executes a query in a certain language. This might be, for example Procedural Query Language (PQL) or Continuous Query Language (CQL). There are three different commands to execute such a query: #QUERY, #ADDQUERY and #RUNQUERY. While #QUERY and #ADDQUERY (they are one and the same) only passes the definied query to Odysseus, the #RUNQUERY additionally starts the query. This means, a query that was added with #QUERY or #ADDQUERY is inactive and not started until it is explicetely started. The #RUNQUERY in contrast immediatly starts a query after it is added, e.g. by using #STARTQUERIES .

Parameters

The query command is dependent on the current parser (which is set by #PARSER). Thus, if you want to run a query in CQL that last #PARSER command before should set the parser to "CQL".

If #QName is defined before, the query will get this name.

Example

The example shows four queries after the parser is set to CQL and the transformation configuration is set to Standard. The first one uses #QUERY and it is executed as a CQL-Query, but not started. The second query is equal to the first one (it still uses CQL and is not started). The third query also uses CQL and the Standard transformation configuration, but is (in contrast to the first and second) started (it is directly running). Then, the parser is switched to PQL, so that the fourth query is parsed by the PQL-Parser and not by the CQL-Parser anymore.

```
#PARSER CQL

#QUERY
SELECT * FROM bid

#ADDQUERY
SELECT * FROM bid

#RUNQUERY
SELECT * FROM bid

#PARSER PQL

#QUERY
result =  PROJECT({ATTRIBUTES=['id','name']}, person)
```

#RUNQUERY

This command installs a query and starts it immediately. See #QUERY for parameters, examples and details.

Query can be modified via Odysseus script. For this, the queries must be named. A query can be named with #QNAME:

```
#PARSER CQL
#QNAME query1
#ADDQUERY
SELECT * FROM nexmark:person
```

#STARTQUERY

This command can be used to start a named query.

```
#STARTQUERY query1
```

#STOPQUERY

This command can be used to stop a nazmed query.

```
#STOPQUERY query1
```

#SUSPENDQUERY

This comand can be used to pause a query. Remark: received elements are stored, so this can lead to a memory problem.

```
#SUSPENDQUERY query1
```

#PARTIALQUERY

This comand can be used to reduce load of a query by throwing away some received elements. The factor is a number between 0 and 100 where 0 means keep every element and 100 throw away every element

```
#PARTIALQUERY query1 factor
```

#RESUMEQUERY

A suspended query can be resumed by this.

```
#RESUMEQUERY query1
```

#REMOVEQUERY

Remove a query from the system. Cannot be undone.

```
#REMOVEQUERY query1
```

#WAITFORQUERY

With this comand, the execution of a Odysseus Script can be paused until a query is stopped or removed. period in ms states in which time intervals the query should be checked (default is 1000). maxwaitingtime states how should be waited for the query at maxium (default wait forever).

```
#WAITFORQUERY query1 [period [maxwaitingtime]]
```

#STARTQUERIES

This command starts all installed queries that are not running at the moment.

Parameters

This command has no parameters.

Example

```
#STARTQUERIES
```

#DROPALLQUERIES

This command drops all installed queries. It does not remove andy sources or sinks, but you can use #DROPALLSINKS or #DROPALLSOURCES for this.

Parameters

It has no parameters.

Example

```
#DROPALLQUERIES
```

#BUFFERPLACEMENT

This command is used to control how buffers are (automatically) placed within the query plan if a query is transformed (e.g. by #QUERY).

Parameters

The parameter is the name of a buffer placement strategy. Since the strategies are dynamically loaded, the availability of certain strategies depends on the current system setting (which features are installed and which not). Some possible stragies are shown in the examples.

Examples

No buffers:

```
#BUFFERPLACEMENT None
```

Adds a buffer before each operator:

```
#BUFFERPLACEMENT Standard Buffer Placement
```

Adds a buffer after each source:

```
#BUFFERPLACEMENT Source Buffer Placement
```

Adds a buffer for each query:

```
#BUFFERPLACEMENT Query Buffer Placement
```

#DOQUERYSHARING

This command switches the query sharing (which tries to optimize a query be reusing parts of already installed query plans) on or off.

Parameters

The parameter is a boolean: the parameter may be either "true" or "false".

Example

```
/// query sharing off
#DOQUERYSHARING false
/// query sharing on
#DOQUERYSHARING true
```

#DOREWRITE

This command switches the rewriting (tries to optimize a query plan by switching, deleting, splitting or merging operators without changing the query's semantics) on or off.

Parameters

The parameter is a boolean: the parameter may be either "true" or "false".

Example

```
/// query rewrite off
#DOREWRITE false
/// query rewrite on
#DOREWRITE true
```

#METADATA

In the default processing scenario, all elements in Odysseus are tagged with time stamp meta data. This command can be used to define the meta data.

Hint: **This flag overwrites the standard configuration so you must provide all metadata that should be used!**

Example

```
#METADATA TimeInterval
#METADATA Latency
#METADATA Priority
```

Remark: There are some combined metadata elements available: e.g. IntervalLatency or IntervalLatencyPriority

#OPTIMIZE_PREDICATES

Enable predicate optimization for the current query. The optimizer tries to simplify the predicates used in select operations and transforms the predicate into its conjunctive normal form to move clauses in the expression down to the sources.

Example

```
#OPTIMIZE_PREDICATES true
```

#PARSER

This command sets the current parser for following commands, e.g. by #QUERY or #ADDQUERY. The according parser is used until another parser is set.

Parameters

The parser: Which parsers are available strongly depends on the current system setting and installed features. Normally in the default product, there is "PQL" for Procedural Query Language (PQL) and "CQL" for Continuous Query Language (CQL).

Example

```
#PARSER PQL
```

#QName

Set the name of the following query.

```
#QNAME Query1
```

#QPARAM

Some special processing could require query specific processing.This command can be used to set these parameters as key value pairs.

```
#QPARAM key1 value1
#QPARAM key2 value2
#QPARAM key3 value3
```

Remark for Developers: There values could be read from the LogicalQuery-Object with getUserParameter(key):

#QPRIORITY

Set the priority of the next following query

```
#QPRIORITY 10
```