Query Definition Language (QDL)

This document shows the steps to create a continuous query with QDL.

Name

First you need to find a name for the new query.

Example

```
query Q_1 {
        //...
}
```

Metadata

As the next step you can set metadata for the query. Many commands from Odysseus Script can be used at this point.

Example

```
query Q_1(dorewrite = false, runquery = true) {
        //...
}
```

Statements

Next you need to create statements for building the operator graph of the query. QDL offers a class for each operator of Odysseus that you can use in this context. In addition QDL offers a class for each source that is defined in Odysseus. Alternatively there are automatically implicit variables for each source. Thus it is not necessary to create new variables for reusing sources.

There are several ways to connect operators to an operator graph. One way is to use the Subscribe-expression. The arrow of a Subscribe-expression gives a hint for the direction of data flow (e.g. `auction->window`). If an operator has more than one output in this context, you can specify an output port by using the OutputPort-expression (e.g. `filter:1`). The input port is automatically increased when a source is subscribed. Another way to connect operators is by using the constructors or methods of operator classes. For example, by using the methods it is possible to manually change or set an input port.

Simple example

```
query Q_1(dorewrite = false, runquery = true) {
    TimeWindow window{size=[30, "SECONDS"], advance=[1, "SECONDS"]};

    Filter filter{predicate="bid.price <= 200"};

    ODLJoin join{predicate="auction.ID = bid.AUCTION", card = "ONE_MANY"};

    ODLProject project{attributes=[auction.ID, auction.ITEMNAME, bid.PRICE]};

    auction -> window;
    bid -> filter;
    [window, filter:1] -> join -> project;
}
```

More complex example

```
query Q_2(dorewrite = false, runquery = true) {
    double memoryInGB = runtime.totalMemory / 1073741824.0;

    long windowSize = 10;
    if (memoryInGB < 1)
        windowSize = 5;

    ElementWindow windowOp{size = windowSize};

    Aggregate aggregateOp{
        aggregations = [["AVG", bid.PRICE, "AVG_PRICE"]],
        group_by = [bid.AUCTION]};

    Operator[] sinks();
    for (int i = 1; i<=5; i++) {
        sinks += new ODLSelect{predicate = "bid.AUCTION = "+i};
    }

    bid -> windowOp -> aggregateOp -> sinks;
}
```

Multiple queries

Normally one query is installed in Odysseus when executing a QDL-query. However, sometimes it might be useful to create several queries in a loop. Therefore you can call methods in a QDL-query to manually create queries. In this context you should set the metadata `auto_create` to false.

Example

```
query Q_3 (auto_create=false){
    for (int i = 1; i<=5; i++) {
        ODLSelect select{predicate = "bid.AUCTION = "+i};
        bid -> select;
        start(select);
    }
}
```

Grammar

Grammar

```
QDLModel              ::= (Namespace)* (Query | Class | Interface)*.
Query                 ::= "query" ID ("(" (Metadata ("," Metadata))* ")")?
StatementBlock.
SubscribeExpression   ::= Expression ("->" | "<-") Expression.
PortExpression        ::= Expression ":" Expression.
```