Selection, Projection and Map

In this tutorial you will learn to write simple queries that filter elements and attributes.

We will use the same setting as in Simple Query Processing. So you should follow steps 1-4.

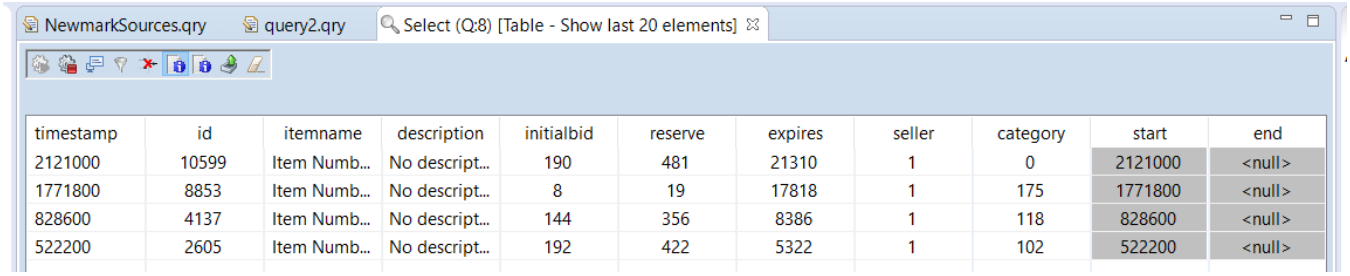
Selection

With the selection operator you can filter out elements that are not relevant for further processing. Create a new Odysseus Script file with the PQL template and name it query2.

The first example will select only those auction, that are created by the seller with the id 1:

```
out = SELECT({predicate='seller=1'}, nexmark:auction)
```

Execute the script and show the output as table. After some time, you should see only auctions opened by the seller with the id 1

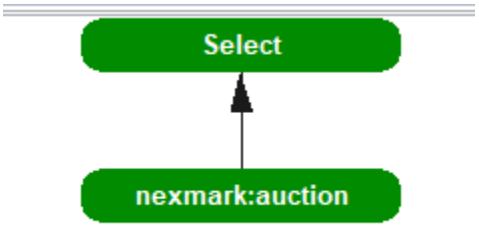


timestamp	id	itemname	description	initialbid	reserve	expires	seller	category	start	end
2121000	10599	Item Numb...	No descript...	190	481	21310	1	0	2121000	<null>
1771800	8853	Item Numb...	No descript...	8	19	17818	1	175	1771800	<null>
828600	4137	Item Numb...	No descript...	144	356	8386	1	118	828600	<null>
522200	2605	Item Numb...	No descript...	192	422	5322	1	102	522200	<null>

In this script you define a SELECT-Operator. In Procedural Query Language (PQL) each operator is identified by a name. Inside the operator there are two parts. The first part is the configuration inbetween "{" and "}". The seconds part is the source part, i.e. here the sources are listed that should deliver the input.

In this example the used source is nexmark:auction (A selection can only filter out elements from one source, so no further sources can be used in a selection). In the configuration area predicate describes the predicate that should be used in the selection. For each incoming event the predicate is evaluated and the event is send to the next operator if the predicate evaluates to true. In other cases the event will be discarded.

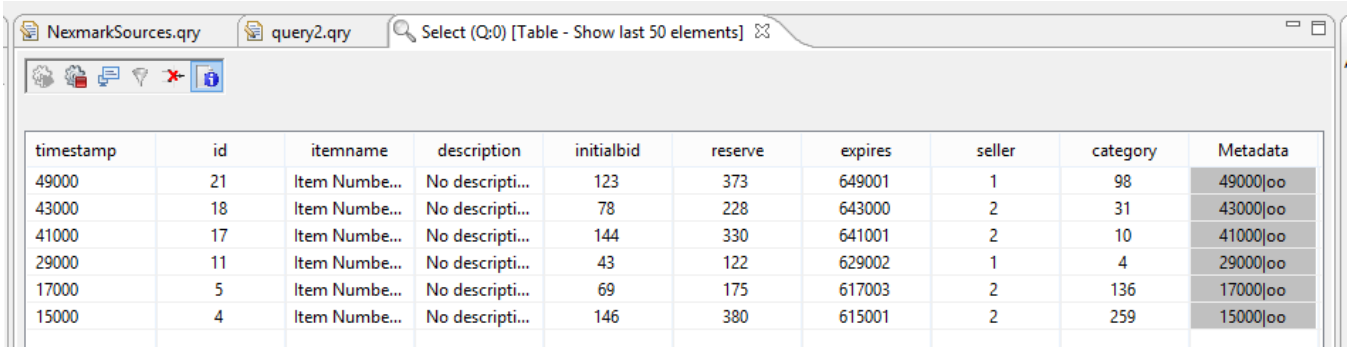
If you look at the query plan, you will see, the following



The filter step is done by the top most operator Select.

Now modify the query and change the predicate to 'seller=1 || seller=2'. This meens you are only interested in auctions opened by seller 1 or seller 2.

Remove the old query and execute the new script. The output should look like in the following.



timestamp	id	itemname	description	initialbid	reserve	expires	seller	category	Metadata
49000	21	Item Numbe...	No descripti...	123	373	649001	1	98	49000|oo
43000	18	Item Numbe...	No descripti...	78	228	643000	2	31	43000|oo
41000	17	Item Numbe...	No descripti...	144	330	641001	2	10	41000|oo
29000	11	Item Numbe...	No descripti...	43	122	629002	1	4	29000|oo
17000	5	Item Numbe...	No descripti...	69	175	617003	2	136	17000|oo
15000	4	Item Numbe...	No descripti...	146	380	615001	2	259	15000|oo

More complex predicates can be defined. See MEP: Functions and Operators for further information.

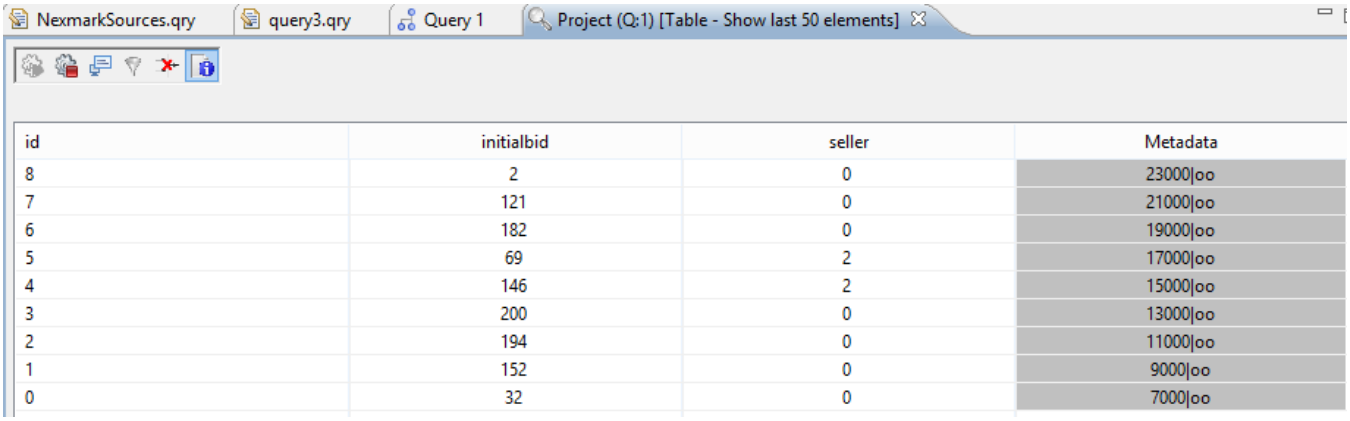
Project

With the SELECT-Operator you choose which events you will see in the output, with the PROJECT you will choose, which attributes should be in the output.

Create a new Odysseus Script file with the PQL template and name it query3.

```
out = PROJECT({ATTRIBUTES=['id', 'initialbid', 'seller']},nexmark:auction)
```

After translating the following output will be displayed. You can see, that only the selected attributes are printed.



id	initialbid	seller	Metadata
8	2	0	23000|oo
7	121	0	21000|oo
6	182	0	19000|oo
5	69	2	17000|oo
4	146	2	15000|oo
3	200	0	13000|oo
2	194	0	11000|oo
1	152	0	9000|oo
0	32	0	7000|oo

Although, the examples only contain one operator, the operators can be connected. E.g. first a selection and than a projection:

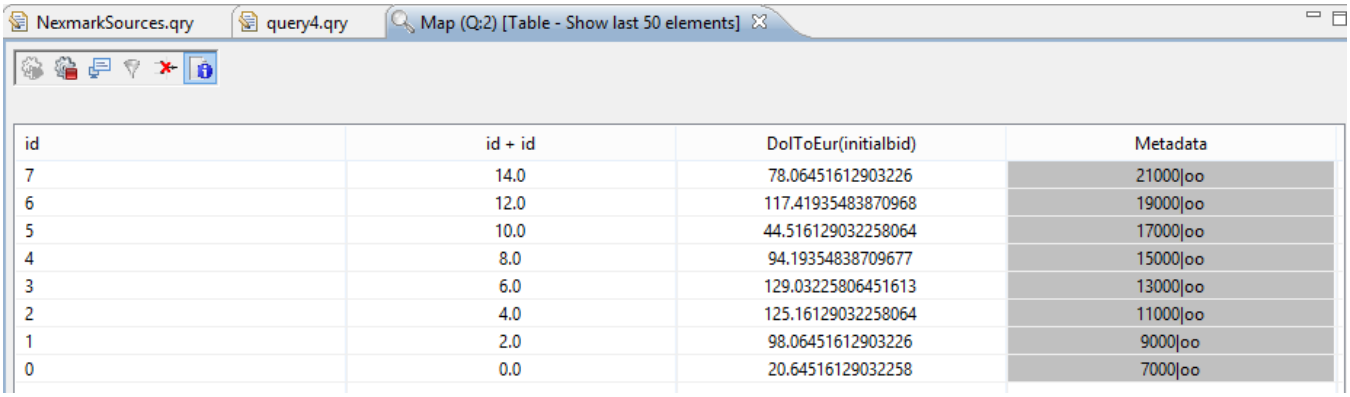
```
selected = SELECT({predicate='seller=1 || seller=2'}, nexmark:auction)
out = PROJECT({ATTRIBUTES=['id', 'initialbid', 'seller']},selected)
```

Map

PROJECT only allow the selection of attributes. With the MAP-Operator calculations can be done on the input. The simpliest calculation is the output of an input attribute, so Map is more general than Project. (Warning: You should not use Map instead of Project, because it requires more processing capabilties).

Create a new Odysseus Script file with the PQL template and name it query4.

```
out = MAP({EXPRESSIONS=['id','id+id','dolToEur(initialbid)']},nexmark:auction)
```



id	id + id	DolToEur(initialbid)	Metadata
7	14.0	78.06451612903226	21000|oo
6	12.0	117.41935483870968	19000|oo
5	10.0	44.516129032258064	17000|oo
4	8.0	94.19354838709677	15000|oo
3	6.0	129.03225806451613	13000|oo
2	4.0	125.16129032258064	11000|oo
1	2.0	98.06451612903226	9000|oo
0	0.0	20.64516129032258	7000|oo

As you can see, the output is the printed expression. Sometimes (especially when the output should be processed by another operator) this names should be more handy. You could use the RENAME Operator or an additional feature of Map. Instead of giving an expression, there can be a pair of expression and output name.

```
out = MAP({EXPRESSIONS=['id',['id+id','DoubleId'],['dolToEur(initialbid)','Bid €']]},nexmark:auction)
```

id	DoubleId	Bid €	Metadata
99	198.0	12.903225806451612	205000|oo
98	196.0	96.12903225806451	203000|oo
97	194.0	28.387096774193548	201000|oo
96	192.0	70.3225806451613	199000|oo
95	190.0	110.96774193548387	197000|oo
94	188.0	74.19354838709677	195000|oo
93	186.0	9.032258064516128	193000|oo
92	184.0	16.774193548387096	191000|oo