Task3. Execute queries over different execution models and data layout

I decided to test the following queries:

Query1 - Select

```
SELECT * FROM lineitem WHERE col1 >= 156345
```

Query2 - Project

```
SELECT col1, col2 FROM lineitem
```

Query3 – Select + Project

```
SELECT col1, col2 FROM lineitem WHERE col1 >= 156345
```

Query4 - Join

```
SELECT l.attri, o.attrj, FROM lineitem l, orders o
                                    WHERE l.l_orderkey=o.o_orderkey
```

Query 5 – ProjectAggregate

```
SELECT AGGR(attri) FROM lineitem
```

Query 6 – ProjectAggregate + Join

```
SELECT AGGR(o.attri) FROM lineitem l, orders o
                                    WHERE l.l_orderkey=o.o_orderkey
```

Here is the result obtained (time in miliseconds):

	Query 1 Select	Query 2 Project	Query 3 Select + Project	Query 4 Join	Query 5 Project Aggregate	Query 6 Project Aggregate + Join
NSM - tuple	3.98	3.57	1.24	1462.91	2.40	1334.80
PAX - tuple	5.95	4.06	2.37	6406.28	2.86	5515.31
DSM - column	5.47	0.32	4.44	841.41	2.09	793.02
DSM - vector	9.05	2.33	4.69	5327.52	8.85	5063.66

For the select query, the NSM performs the best. It is because rows are directly accessed, no overhead like when the data is stocked with columns (DSM) due to the fact that we recreate the tuples.

For the projection query, it is obvious that it is the contrary. DSM – column performs extremely good and it is the best because columns are directly accessed whereas the other layouts usually bring an overhead.

For the join, DSM – column also performs the best. Actually there are less function calls, specially due to the scan phase for DSM – column. The columns are loaded immediately wheras for NSM/PAX, the "next" method is called many times.

However, when we have both projection and selection (first selection than projection), NSM/PAX performs better than DSM – column. This could be explained by the fact that the selection is the first so what we gained with DSM – column can not be applied.

For the ProjectAggregate, DSM – column is the best and this is explained by the facts that we already mentioned earlier. We already have the column we need to do the aggregate on.

Overall, DSM – Vector performs quite like DSM – column (a little worse). However, when there are joins it performs really poorly. This is mainly due to the implementation I used.

Whereas for PAX, theoretically, it should combine both the benefits of DSM and NSM: it should avoid the columns stitching and should bring only the relevent attributes in memory. However, in the way it is used (with volcano) and due to the fact that everything is in memory, it isn't as good as we could think.

CF MAIN.JAVA for more details on the tests