Sample Eligibility Analysis

One of the first steps on clinical trials is the selection of the eligible individuals for the study. The rules that an individual must match in order to be elected as a candidate usually involve complex queries combining clinical metadata and variants data, expressed as a list of filters with intersections (AND), unions (OR) and complement (NOT).

Table of Contents:

Election query

The input query consists in a text line with all the conditions grouped by parenthesis.

An election query can be defined as a tree where each node is either a query or the intersection or union of a list of queries

To solve the query, the tree has to be explored, solve the leaves from the tree, and then join the results.

Query leaf node

The query is the basic element in the tree. When processed, will produce a list of samples that meet the query.

It consists in a list of filters joined by "AND" operators.

QUERY	`( <FILTER> [AND <FILTER> ]* )`

Each filter contains a param, the operator, and the value.

Standard variant query filters can be used to filter. Can be combined with sample and individual catalog queries pre-pending the query params with "`sample.`<sample-param>" or "`individual.`<individual-param>" respectively.

FILTER	`[sample.|individual.]?<QUERY_PARAM><OPERATOR><VALUE>`

Query Node Model

```
{
  "type" : QUERY,
  "query" : {
    "<query param 1>" : "<operator><value>",
    "<query param 2>" : "<operator><value>",
    ...
    "<query param n>" : "<operator><value>",
  }
}
```

Complement node (NOT)

A complement node will inverse the selection of the samples selected by the node.

COMPLEMENT	`( NOT <NODE> )`

Complement Node Model

```
{
  "type" : COMPLEMENT
  "nodes" : [ <node> ]
}
```

Union node (OR)

A union node will join all samples selected by the sub-nodes, returning them all.

UNION	(<NODE> [OR <NODE>]*)

Union Node Model

```
{
   "type" : UNION
   "nodes" : [ <node1>, <node2>, ... ]
}
```

Intersection node (AND)

A intersect node will return only samples selected by all sub-nodes.

INTERSECTION	(<NODE> [AND <NODE>]*)

Intersection Node Model

```
{
   "type" : INTERSECTION
  "nodes" : [ <node1>, <node2>, ... ]
}
```

Example

Natural language text

Get all samples that have (a `lof` mutation in gene A and with disorder X) or ((a missense_variant mutation on gene B) and (a missense_variant on gene C) but not (variant rs12345))

Election query

To build the election query, the natural language query can be divided into 4 queries:

- Q1
  - gene : A
  - ct : lof
  - individual.disorders : X
- Q2
  - gene : B
  - ct : missense_variant
- Q3
  - gene : C
  - ct : missense_variant
- Q4
  - id : rs12345

These queries are combined with AND/OR like this:
`(Q1 OR (Q2 AND Q3 AND (NOT Q4)))`

All together, the query can be expressed like this:

```
((gene=A AND cf=lof AND individual.disorders=X) OR ((gene=B
ct=missense_variant) AND (gene=C ct=missense_variant) AND (NOT (id=rs12345))
))
```

Structured

This can be expressed in a JSON query like this:

```
{
  "type" : U
NION
  "nodes" :
[
    {
      "type"
: QUERY,
      "query"
 : Q1
    } , {
      "type"
: INTERSECTI
ON
      "nodes"
 : [
        {
          "t
ype" : QUERY
          "q
uery" : Q2
        }, {
        {
          "t
ype" : QUERY
          "q
uery" : Q3
        }, {
          "t
ype" : COMPL
EMENT
          "n
odes" : [{
            "
type" : QUERY
            "
query" : Q4
          }]
        },
      ]
    }
  ]
}
```

Implementation

Implementation details at opencga#1474.

Input

Parameters

Output

Files