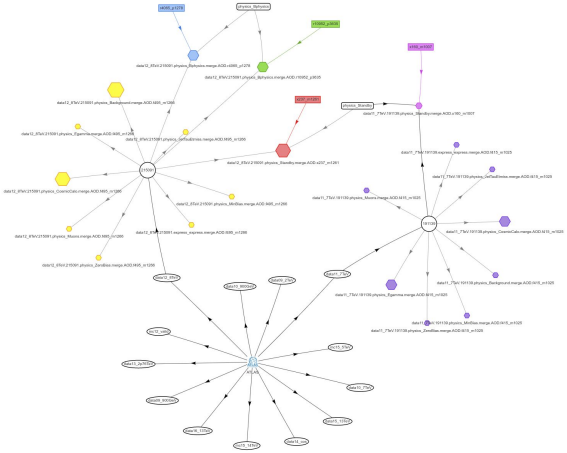
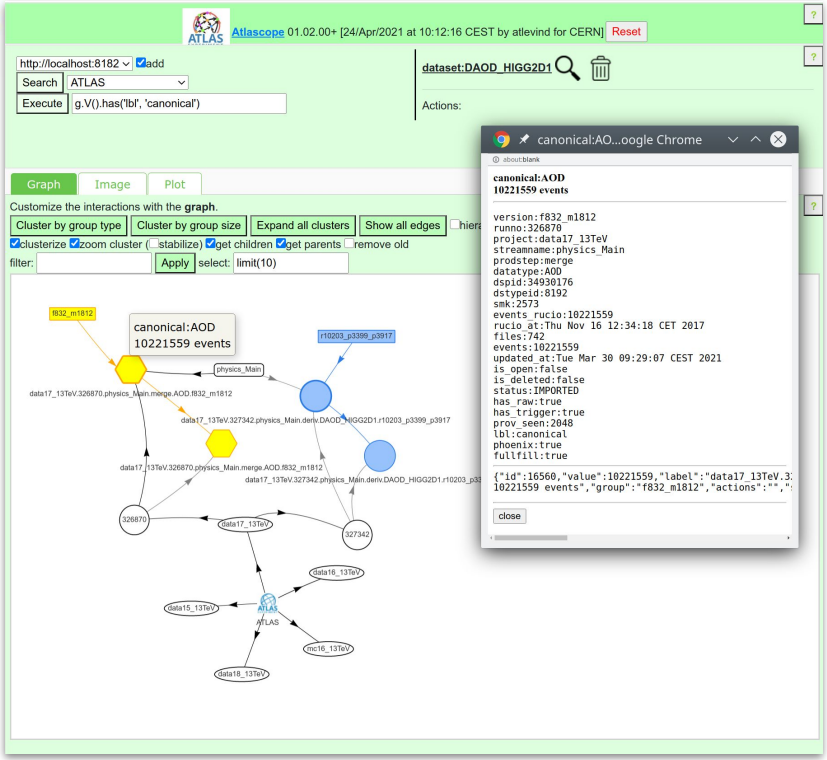
Atlascope





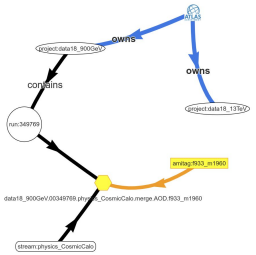
Using Demo data from Small IFIC sample @CERN

- Architecture
- Status
- Schema & Indexes
- Clients & Examples
- Web Client

Julius Hrivnac, IJCLab

EI WS, 4/Oct/2021

Architecture



import: dataset, …
+ create: amitag, run,...

JanusGraph *gremlin*
db

imported objects

proxy objects

new objects

SQL
Phoenix
db

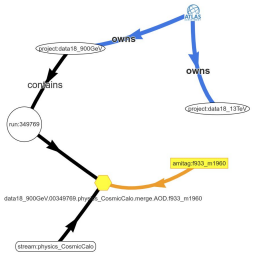
access: event,...
(import, if needed)

HBase
storage

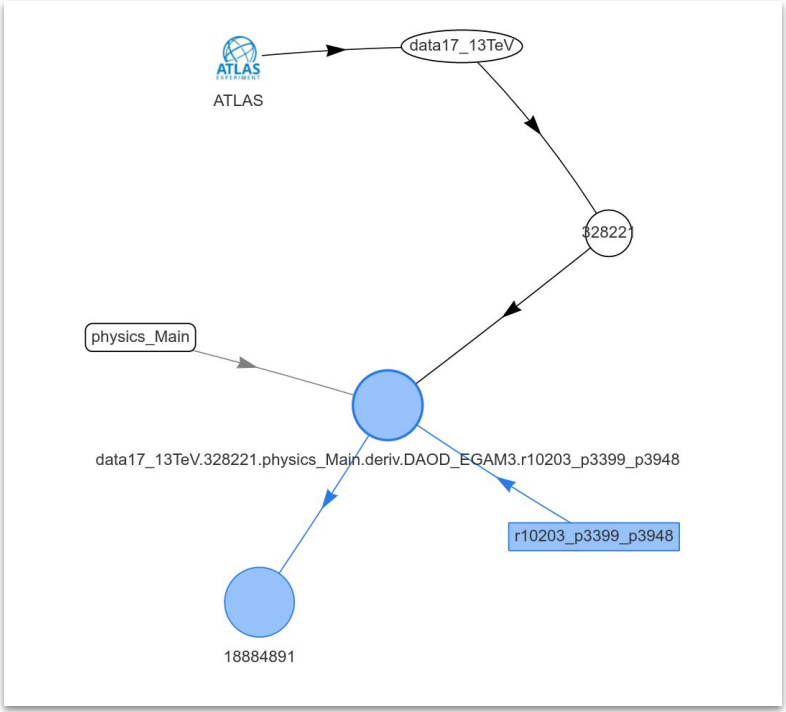
JS
Atlascope
Web Client

JSP
Atlascope
Web Application

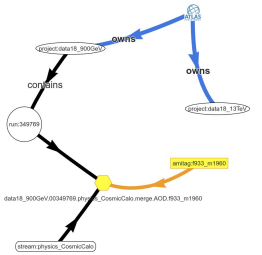
Status



- Using IFIC demo database @CERN (AEIDEV.JSZ3_*)
  - Importing @ 50Hz
    - Mostly due to Phoenix overhead and connection over Socket
    - Direct importing from files gives about 1kHz
    - Will try new Spark Scala package
  - Importing **datasets** and **canonical**
  - While importing, creating full graph structure
  - So enabling *searching by navigation*
- A lot of Phoenix table fields are in fact relations
  - They are replaced by Edges in Graph



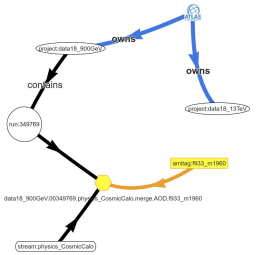
Schema & Indexes



- Graph query is executed in three steps:
  - Initial search
    - Uses underlying database technology (HBase)
    - Can profit from Indexes (HBase + Elastic Search)
  - Navigation (very fast)
  - Accumulation of properties
- Schema & Indexes created to support the Initial search
  - Graph can still contain Vertexes, Edges and properties not covered by Schema

```
// Schema
canonical = mgmt.makeVertexLabel('canonical').make()
dataset   = mgmt.makeVertexLabel('dataset'  ).make()
stream    = mgmt.makeVertexLabel('stream'   ).make()
...
tags  = mgmt.makeEdgeLabel('tags').multiplicity(ONE2MANY).make()
fills = mgmt.makeEdgeLabel('fills').multiplicity(ONE2MANY).make()
...
events_rucio = mgmt.makePropertyKey('events_rucio').dataType(Long.class   ).cardinality(Cardinality.SINGLE).make()
rucio_at     = mgmt.makePropertyKey('rucio_at'    ).dataType(Date.class   ).cardinality(Cardinality.SINGLE).make()
files        = mgmt.makePropertyKey('files'       ).dataType(Integer.class).cardinality(Cardinality.SINGLE).make()
...
mgmt.addProperties(canonical,
                   lbl,
                   importDate,
                   phoenix,
                   fullfill,
                   runno,
                   streamname,
                   prodstep,
                   datatype,
                   version,
                   dspid,
                   dstypeid,
                   smk,
...
mgmt.addConnection(has,      stream,      canonical)
mgmt.addConnection(tags,     amitag,      canonical)
mgmt.addConnection(fills,    run,         canonical)
mgmt.addConnection(gets,     project,     run)
mgmt.addConnection(owns,     ATLAS,       project)
mgmt.addConnection(contains, canonical,   dataset)
...
// Indexes
mgmt.buildIndex('byVersionES', Vertex.class).addKey(lbl).
                                             addKey(version, Mapping.TEXTSTRING.asParameter()).
                                             buildMixedIndex('search')
mgmt.buildIndex('byRunnoES',   Vertex.class).addKey(lbl).
                                             addKey(runno).
                                             buildMixedIndex('search')
```

(C)Python Client

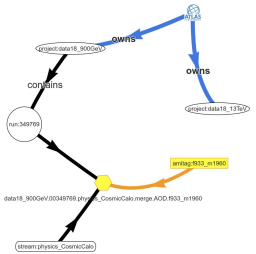



```
#pip install gremlinpython

from gremlin_python import statics
from gremlin_python.process.anonymous_traversal import traversal
from gremlin_python.process.graph_traversal import __
from gremlin_python.process.strategies import *
from gremlin_python.driver.driver_remote_connection import DriverRemoteConnection
from gremlin_python.process.traversal import T
from gremlin_python.process.traversal import Order
from gremlin_python.process.traversal import Cardinality
from gremlin_python.process.traversal import Column
from gremlin_python.process.traversal import Direction
from gremlin_python.process.traversal import Operator
from gremlin_python.process.traversal import P
from gremlin_python.process.traversal import Pop
from gremlin_python.process.traversal import Scope
from gremlin_python.process.traversal import Barrier
from gremlin_python.process.traversal import Bindings
from gremlin_python.process.traversal import WithOptions

statics.load_statics(globals())

g = traversal().withRemote(DriverRemoteConnection('ws://aiatlas073.cern.ch:8182/gremlin','g'))

x = g.V().has('lbl', 'dataset').has(...).valueMap().next()
```

Easy integration in Atlas Framework

Clients exist in most languages

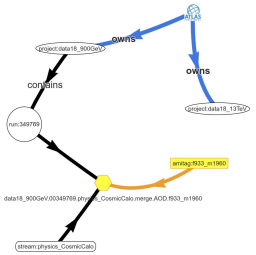
Rest Client



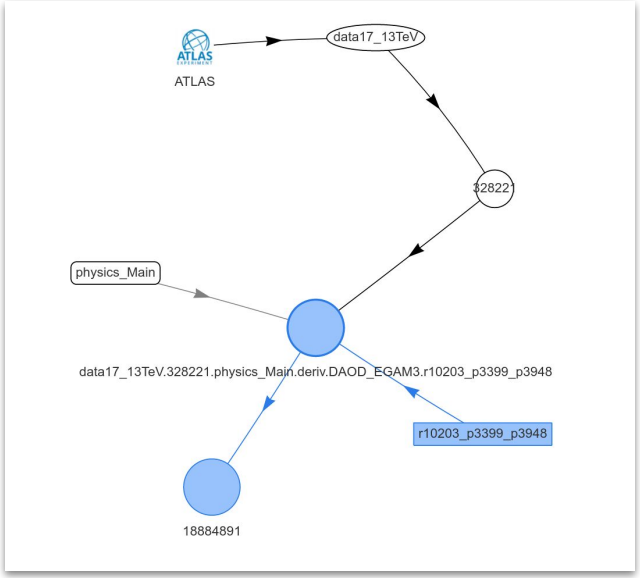

```
curl -v https://aiatlas073.cern.ch:8182/session -XPOST
     -d '{"gremlin": "g.V().has('lbl', 'dataset').has(...).valueMap()"}'
     -H "Authorization: Token YWRtaW46MTYzMDY…”
# returns JSON
```

Easy integration in Any Framework

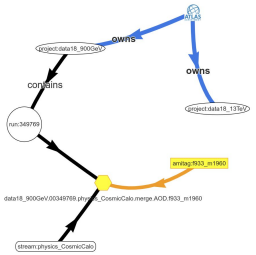
Event Lookup with Graphs




```
// only talks to JanusGraph => very fast
// (pure Gremlin)
dataset = g.V().has('lbl', 'ATLAS')
               .out().has('name',     'data17_13TeV')
               .out().has('runno',    328374)
               .out().has('prodstep', 'merge')
                     .has('datatype', 'AOD')
               .out()
               .next();
// may talks also to Phoenix
// (special method, wrapping Phoenix object as a Graph Vertex)
events = Event.getOrCreate(dataset, g, 22222, true)
```

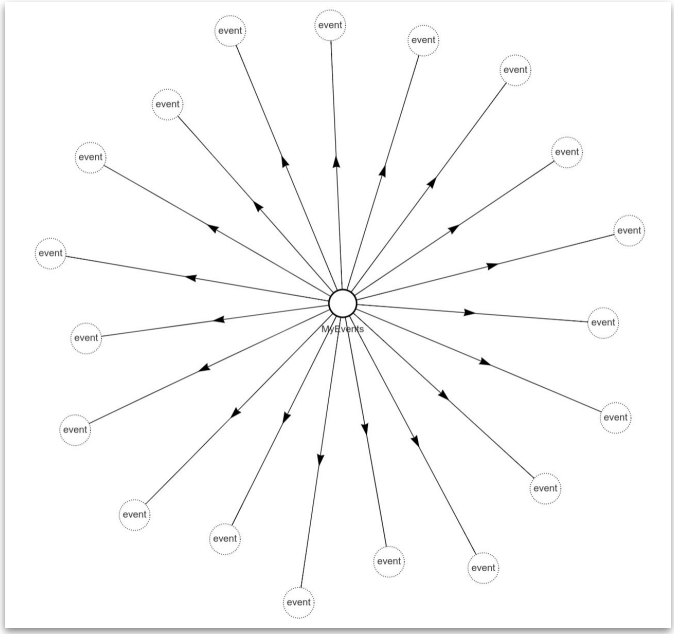


Virtual Collections

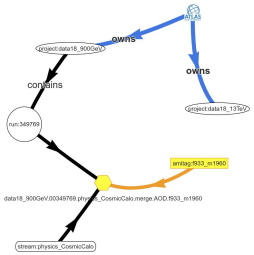


- Virtual Collection:
  - Collection Vertex
    - + Edges to contained Elements
    - + Recipy to get contained Elements

```
// Create new collection of events
eventsCollection = g.addV('ecollection')
                    .property('name','MyEvents');

// Find all events satisfying certain conditions
// and connect them to the event collection
g.V().has('lbl', 'event')
     .has(...some selection...)
     .collect {
       eventsCollection.addEdge('contains', it)
       };

graph.tx().commit();
```

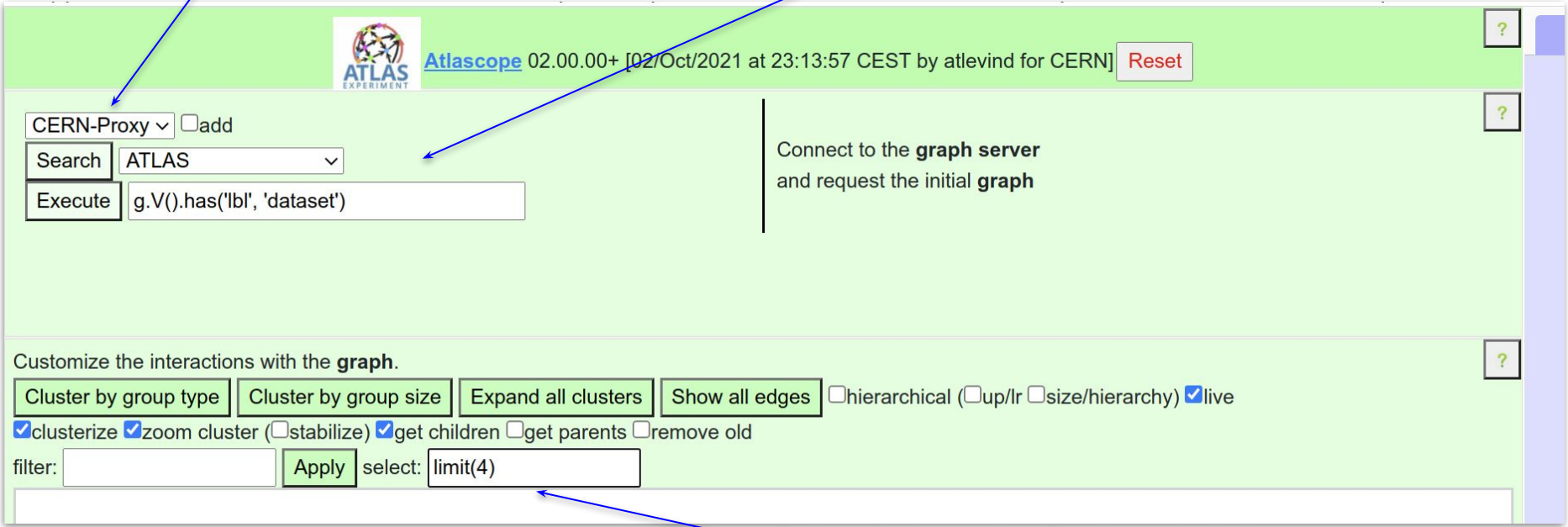


Web Client - Initial Top Panels



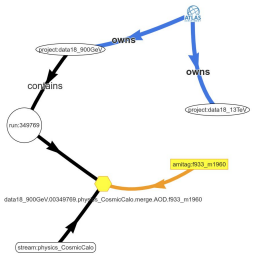
Database to use
('Proxy' tunnels requests to the database
through the Web Service to by-pass firewalls)

Initial Gremlin request



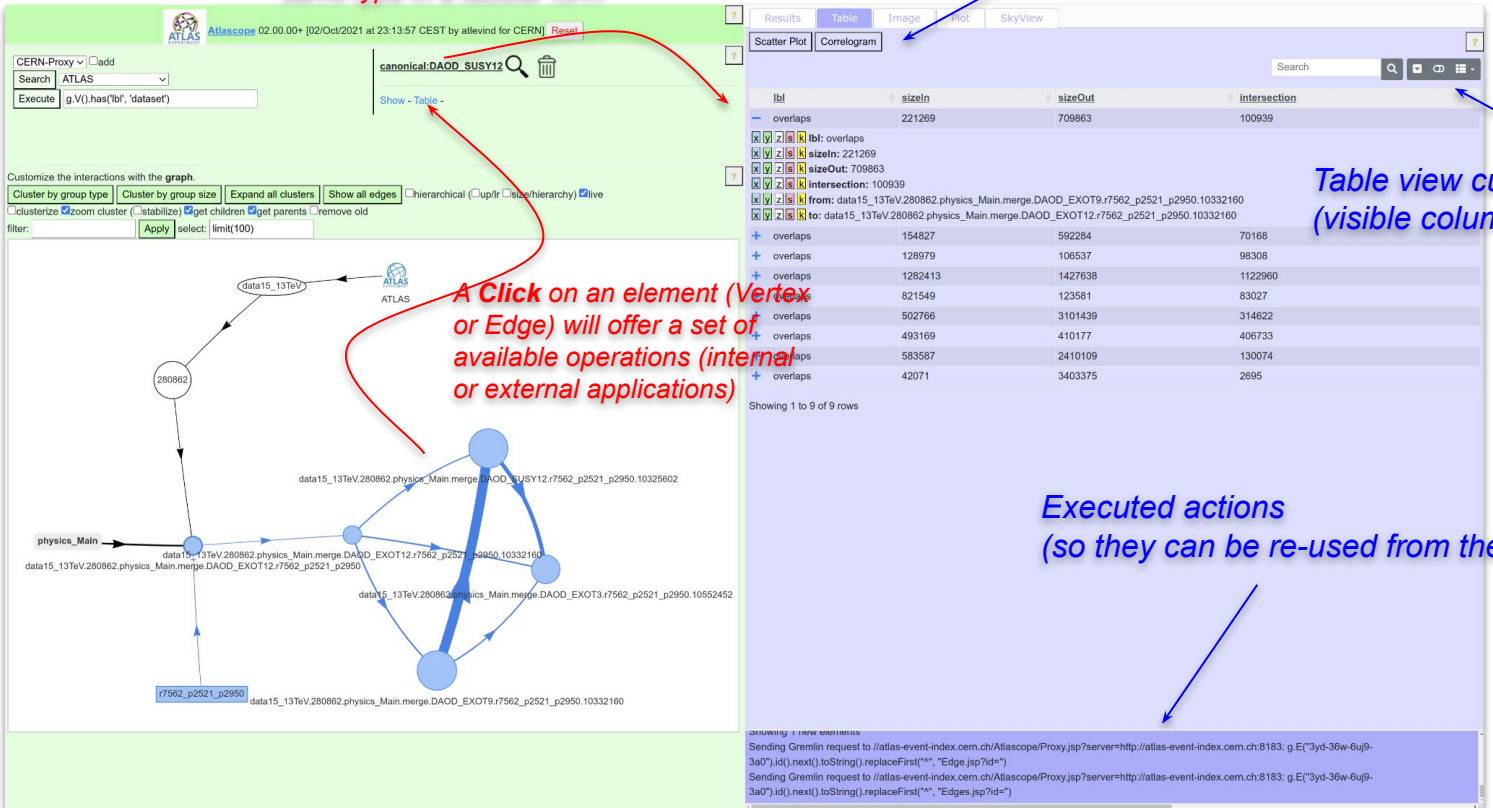
Options for interactive Graph manipulation

Web Client - Graph with Table View



*A **Table** operation will show all visible elements of the same type in a tabular form*

Table data can be plotted in various ways



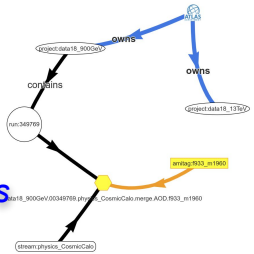
*A **Click** on an element (Vertex or Edge) will offer a set of available operations (internal or external applications)*

Table view customisation (visible columns, searches,...)

Executed actions (so they can be re-used from the command line)

Web Client - Correlogram

Various types of data views



Atlascope 02.00.00+ [02/Oct/2021 at 23:13:57 CEST by atlevind for CERN] Reset

CERN-Proxy add

Search ATLAS

Execute g.V().has('lbl', 'dataset')

canonical:DAOD_SUSY12

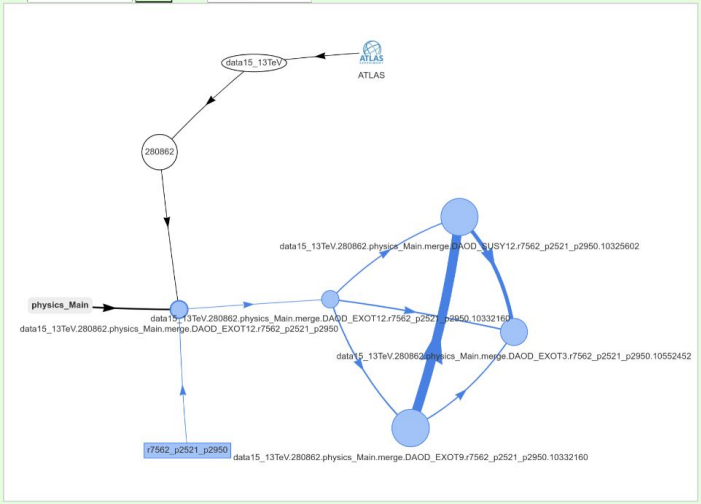
Show - Table -

Customize the interactions with the **graph**.

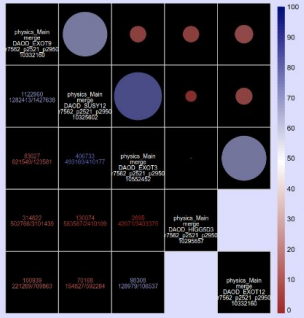
Cluster by group type | Cluster by group size | Expand all clusters | Show all edges | hierarchical (up/lr) size/hierarchy) live

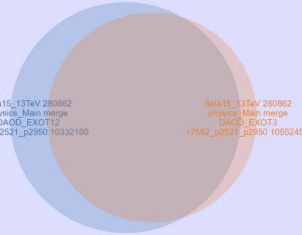
clusterize zoom cluster (stabilize) get children get parents remove old

filter: Apply select: limit(100)



Results | Table | Image | Plot | SkyView





A = data15_13TeV 280862 physics_Main merge DAOD_EXOT12 r7562_p2521_p2950 10332160
B = data15_13TeV 280862 physics_Main merge DAOD_EXOT3 r7562_p2521_p2950 10552452

A	128979	(94%)
B	106537	(78%)
A^B	98308	(72%)
A-A^B	30671	(22%)
B-A^B	30671	(6%)
AvB	137208	(100%)

Showing 1 new elements

Sending Gremlin request to //atlas-event-index.cern.ch/Atlascope/Proxy.jsp?server=http://atlas-event-index.cern.ch:8183: g.E("3yd-36w-6uj9-3a0").id().next().toString().replaceFirst("^", "Edge.jsp?id=")

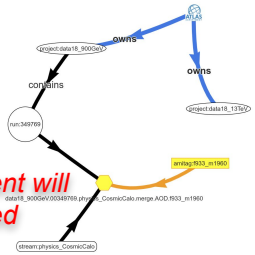
Sending Gremlin request to //atlas-event-index.cern.ch/Atlascope/Proxy.jsp?server=http://atlas-event-index.cern.ch:8183: g.E("3yd-36w-6uj9-3a0").id().next().toString().replaceFirst("^", "Edges.jsp?id=")

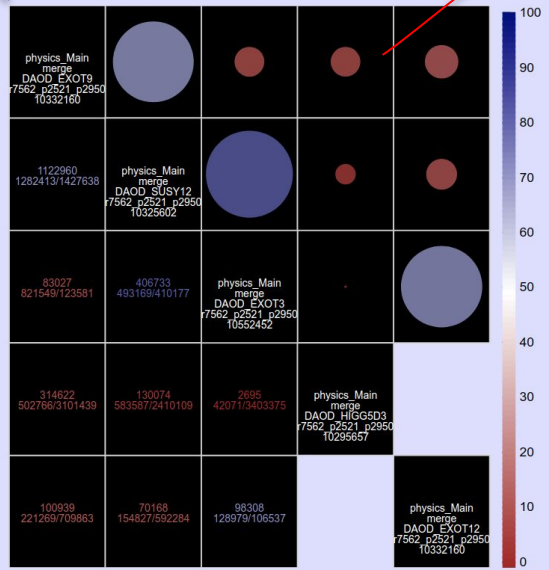
Web Client - Overlaps

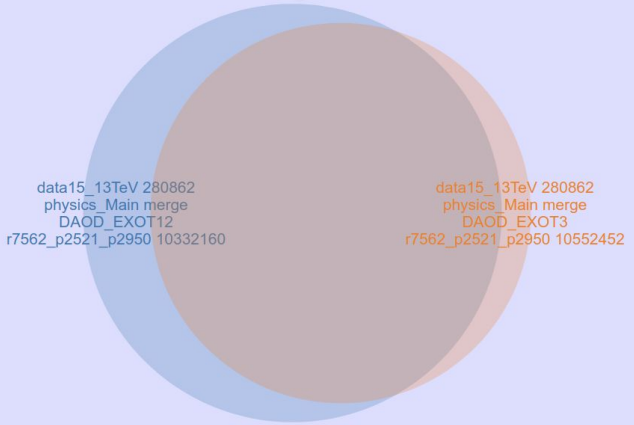
Dataset names

Overlaps as circles

***Hovering** over an overlap element will give a **Venn** diagram with detailed analyses*







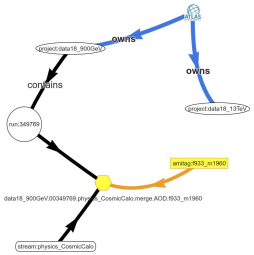
A = data15_13TeV 280862 physics_Main merge DAOD_EXOT12 r7562_p2521_p2950 10332160
B = data15_13TeV 280862 physics_Main merge DAOD_EXOT3 r7562_p2521_p2950 10552452

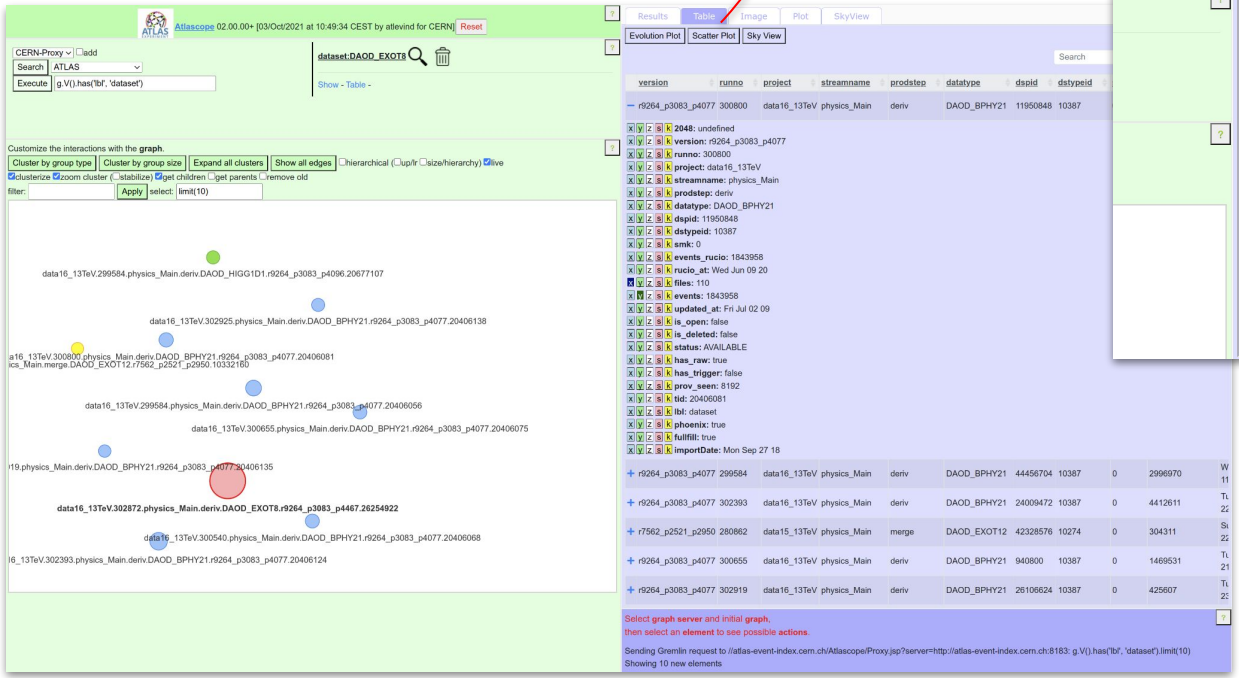
A	128979	(94%)
B	106537	(78%)
A^B	98308	(72%)
A-A^B	30671	(22%)
B-A^B	30671	(6%)
AvB	137208	(100%)

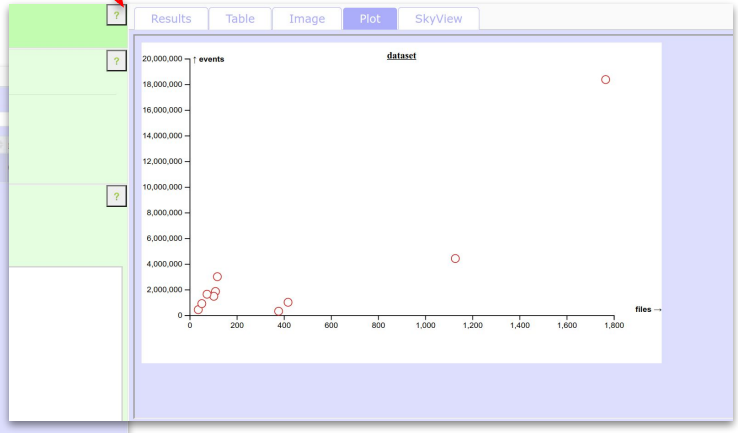
Overlaps as numbers

Web Client - Plots

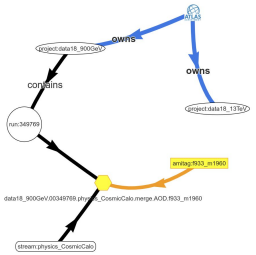
Table columns can be presented in various plot types





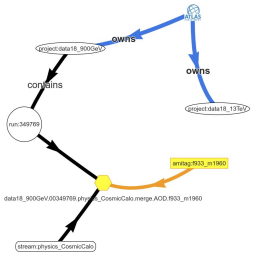


Web Client



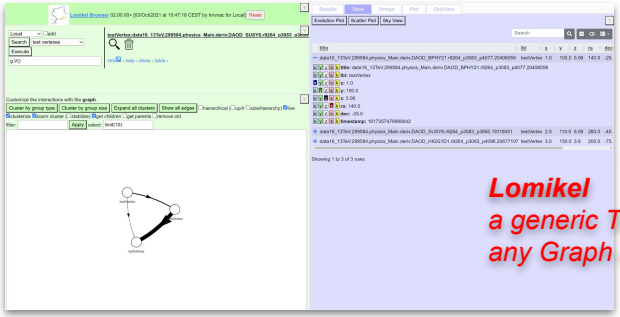
- Each element (Vertex or Edge) has defined graphical presentation, behaviour and available action/operations (internal or external)
  - Customisable by a StyleSheet
  - Applied either to the one element or to all elements of the same type on the screen
- All Views (tables, overlaps,...) are generic and customized for a particular case
  - All (groups of) Vertexes or Edges can be presented as a table (with their properties)
  - Any table fields can be presented as scatter plots, time evolution plots, correlation plots / Venn diagrams, sky views (3d spherical plots),... if they contain required data types
- All visual operations can be executed from a command line (by clients in many languages or via REST)
  - As they are shown on the Feedback pane

Web Client - Architecture

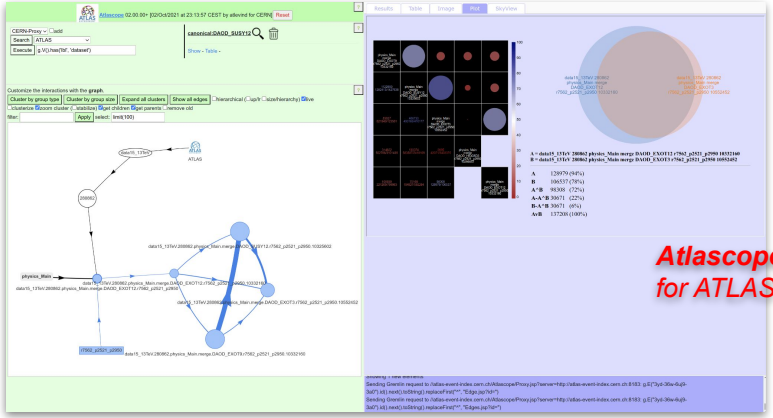



```
}
stylesheet.nodes.dataset = {
  properties:{gremlin:"valueMap('project', 'datatype', 'runno', 'stre
  graphics: {
    label:{js:"project + '.' + runno + '.' + streamname + '.' + prods
    title:"datatype",
    subtitle:{js:"events + ' events'"},
    group:"version",
    shape:{js:"datatype == 'AOD' ? 'hexagon' : 'dot'"},
    image:" ",
    borderRadius:"0",
    borderWidth:"1",
    borderDashes:[1,0],
    value:"events"
    },
  actions: [
    {name:"Show",      url:{gremlin:"id().next().toString().replaceFir
    {name:"Table",     url:{gremlin:"id().next().toString().replaceFir
    ]
  }
stylesheet.nodes.canonical = {
  properties:{gremlin:"valueMap('project', 'datatype', 'runno', 'stre
  graphics: {
    label:{js:"project + '.' + runno + '.' + streamname + '.' + prods
    title:"datatype",
    subtitle:{js:"events + ' events'"},
    group:"version",
    shape:{js:"datatype == 'AOD' ? 'hexagon' : 'dot'"}
```

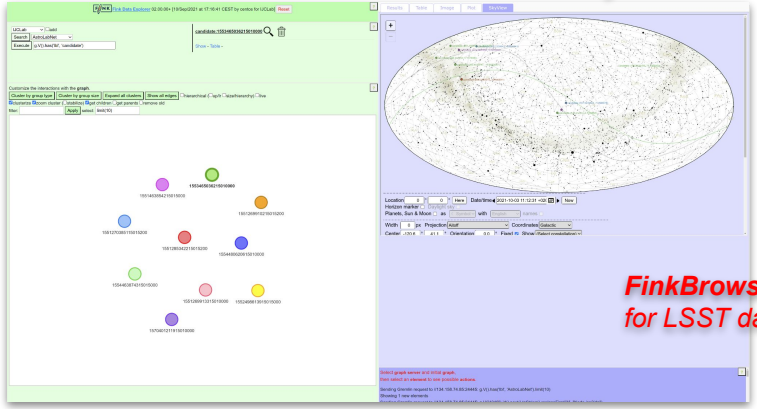
Customising stylesheet (json with embedded JS and Gremlin)



Lomikel
a generic Tool for exploring any Graph database

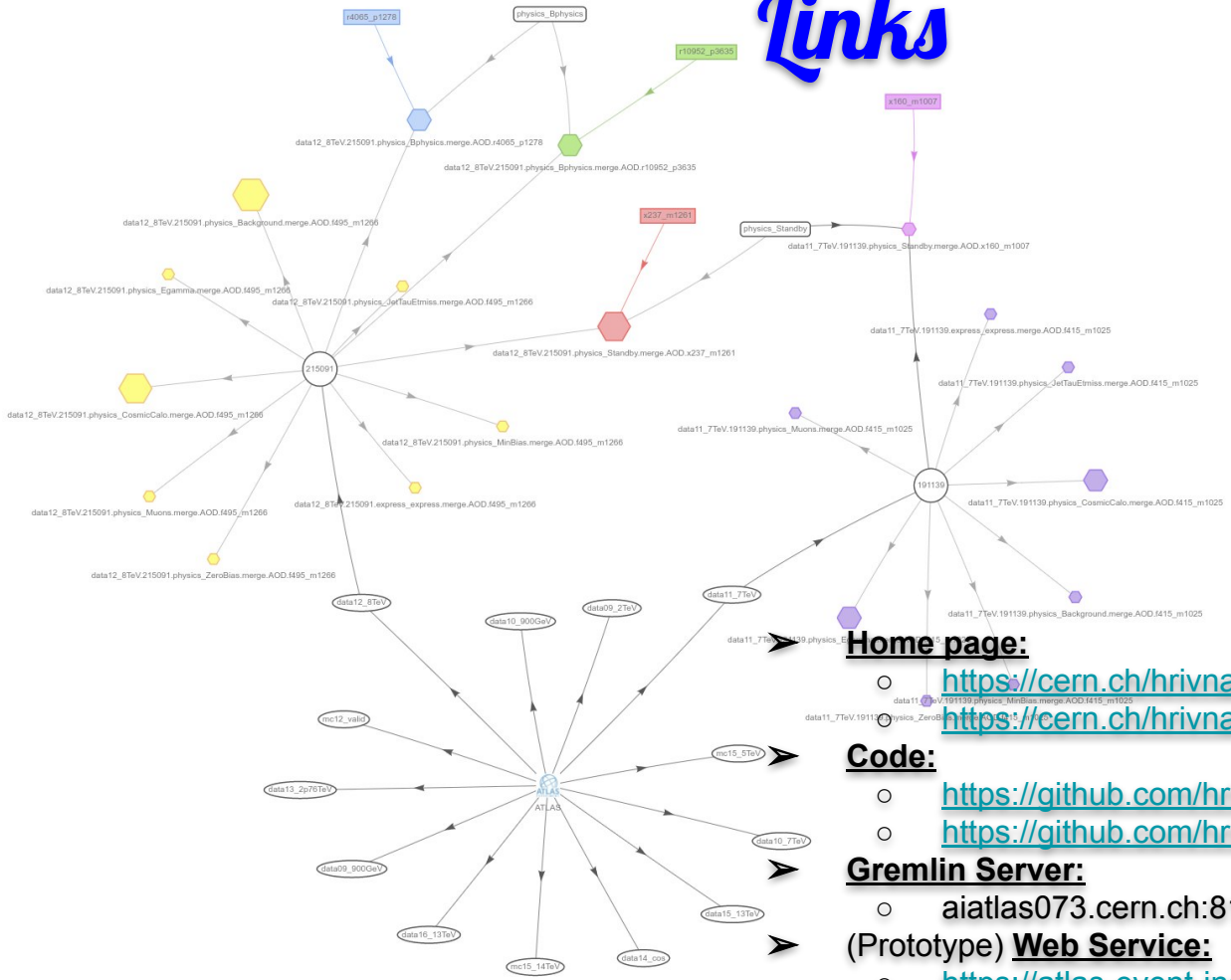


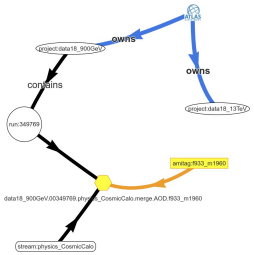
Atlascope
for ATLAS data



FinkBrowser
for LSST data

Links





Doesn't use Core Phoenix framework yet.

➤ **Home page:**
- https://cern.ch/hrivnac/Activities/Packages/Lomikel
- https://cern.ch/hrivnac/Activities/Packages/Atlascope

➤ **Code:**
- https://github.com/hrivnac/Lomikel
- https://github.com/hrivnac/Atlascope

➤ **Gremlin Server:**
- aiatlas073.cern.ch:8182

➤ (Prototype) **Web Service:**
- https://atlas-event-index.cern.ch/Atlascope/?profile=CERN