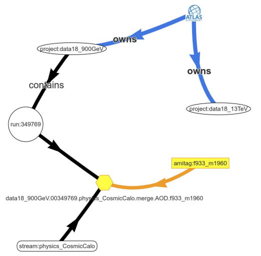
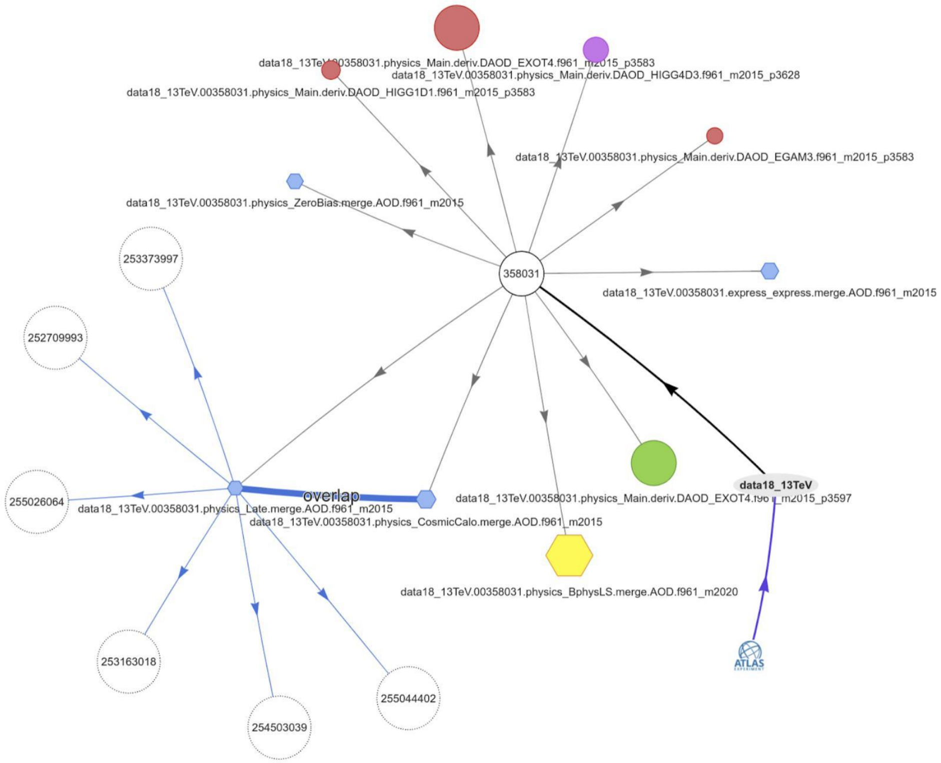
# Atlascope

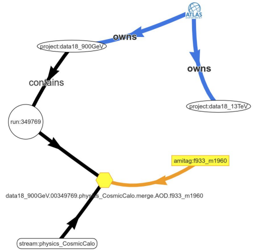



data18_13TeV.00358031.physics_Main.deriv.DAOD_EXOT4.f961_m2015_p3583
data18_13TeV.00358031.physics_Main.deriv.DAOD_HIGG4D3.f961_m2015_p3628
data18_13TeV.00358031.physics_Main.deriv.DAOD_HIGG1D1.f961_m2015_p3583
data18_13TeV.00358031.physics_Main.deriv.DAOD_EGAM3.f961_m2015_p3583
data18_13TeV.00358031.physics_ZeroBias.merge.AOD.f961_m2015
253373997
358031
data18_13TeV.00358031.express_express.merge.AOD.f961_m2015
252709993
data18_13TeV
255026064
overlap
data18_13TeV.00358031.physics_Late.merge.AOD.f961_m2015
data18_13TeV.00358031.physics_CosmicCalo.merge.AOD.f961_m2015
data18_13TeV.00358031.physics_Main.deriv.DAOD_EXOT4.f961_m2015_p3597
data18_13TeV.00358031.physics_BphysLS.merge.AOD.f961_m2020
ATLAS EXPERIMENT
253163018
255044402
254503039

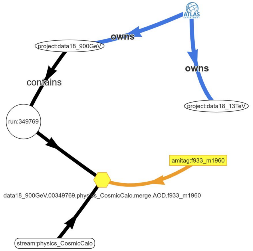
- ➢ Architecture
- ➢ API
- ➢ Web Service
- ➢ Virtual Collections
- ➢ Other Possibilities

*Julius Hrivnac, IJCLab*
*Event Index WS, Virtual, 15/6/2020*

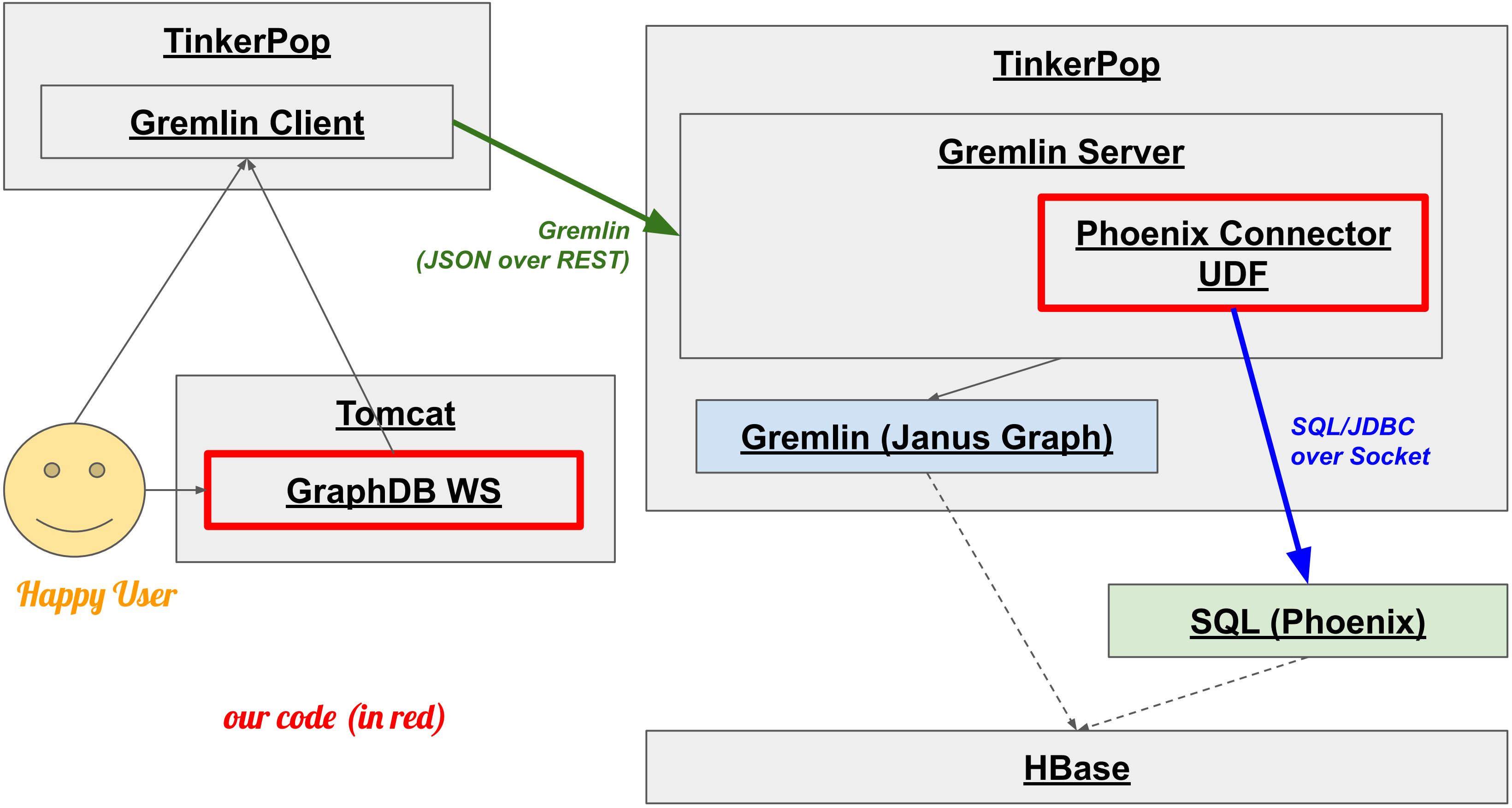
# Architecture



- The architecture is simple:
    - A **Graph layer** on top of an **SQL database**
    - A **table** corresponds to a **Vertex type** (label)
    - A **row** corresponds to an individual **Vertex**
    - Graph layer is transparent, Vertexes are created when first requested, then stored in GraphDB (**lazy creation**)
    - SQL table **relations** are automatically represented by graph **Edges**
    - New Vertexes and Edges can be freely created in the Graph layer (independent on the SQL storage)
    - **Collections** are represented by Vertexes with Edges to contained Vertexes
    - All Graph **tools** are then available for access, navigation, analyses and visualisation
- Very little of code
    - Using a lot of (mostly Apache) projects
    - Standard APIs, replaceable components



# Architecture



TinkerPop

Gremlin Client

Gremlin
(JSON over REST)

TinkerPop

Gremlin Server

Phoenix Connector
UDF

Tomcat

GraphDB WS

Gremlin (Janus Graph)

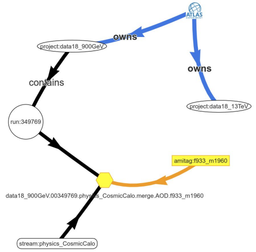
SQL/JDBC
over Socket

SQL (Phoenix)

HBase

Happy User

our code (in red)

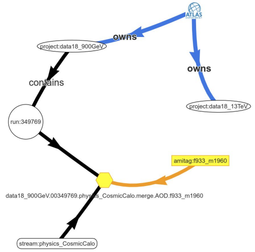


# Architecture

- TinkerPop is a Graph Database Framework for Gremlin-capable databases
  - Included components can be replaced
    - HBase with Cassandra
    - JanusGraph with Neo4J
    - …
- Gremlin client understands most functional-capable languages
  - Java, Scala, Python,...
- The only locally developed components are:
  - Phoenix Connector UDF:
    - To map Phoenix data to Objects
      - Mapping done by hand, but can be automatised
    - To (lazily) wrap them as Graph Vertexes and Edges
    - Over Socket connection
      - To isolate Phoenix & Graph frameworks
        - Originally due to incompatibilities between third-party libraries used by Phoenix & JanusGraph, but can be useful to isolate them anyway
    - Can work with any JDBC/SQL connection
  - GraphDB WS:
    - Generic (Gremlin) JS Web Service
    - Customisable by smart Stylesheet (JSON with Gremlin & JS)

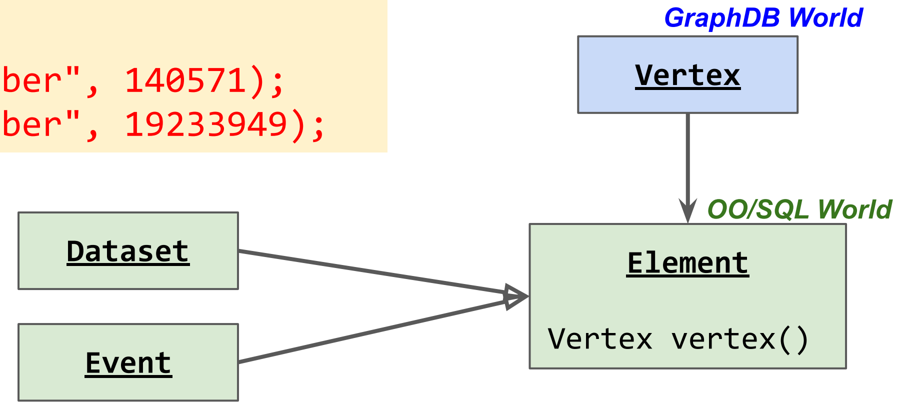
# API

```
ppc = new PhoenixProxyClient("127.0.0.1", 5000); // socket

// Dataset prototype => List<Dataset>
dataset = ppc.search(new Dataset().set("runnumber", 140571)).get(0);
// Dataset => its Vertex
vertex = dataset.vertex();

// Vertex spec => stream of Vertex (created, if needed)
vertex = g.V().has("dataset", "runnumber", 140571).next();
// Vertex => its Dataset
dataset = ppc.get(vertex);

// All vertexes (created, if needed)
vertexes = ppc.vertexes("dataset", "runnumber", 140571);
vertexes = ppc.vertexes("event", "eventnumber", 19233949);
```



- ➢ Vertexes are created lazily, i.e. only when first time asked for
- ➢ Some Edges are added automatically
- ➢ Other Vertex properties and new Edges can be added by users
- ➢ Graph layer serves as an ***extensible cache***

```
PhoenixProxyClient

List<Element> search(Element prototype)
Element get(Vertex vertex)
List<Vertex> vertexes(Object… vertexIds)
```



*GraphDB World*

Vertex

*OO/SQL World*

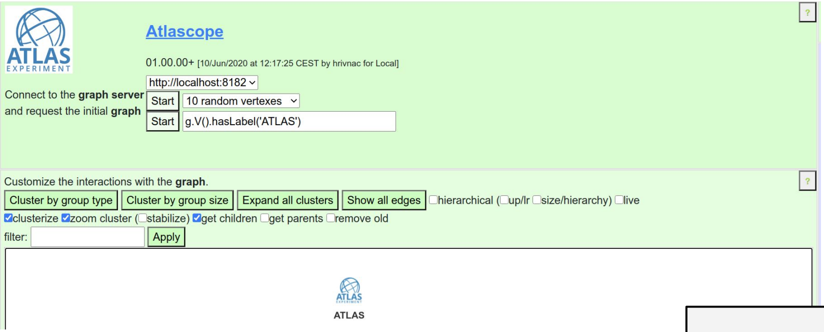
Dataset

Event

Element

Vertex vertex()

# Web Service



Atlascope

01.00.00+ [10/Jun/2020 at 12:17:25 CEST by hrivnac for Local]

Connect to the **graph server** and request the initial **graph**

http://localhost:8182

Start | 10 random vertexes

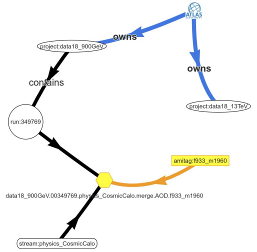
Start | g.V().hasLabel('ATLAS')

Customize the interactions with the **graph**.

☐hierarchical (☐up/lr ☐size/hierarchy) ☐live

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

filter:

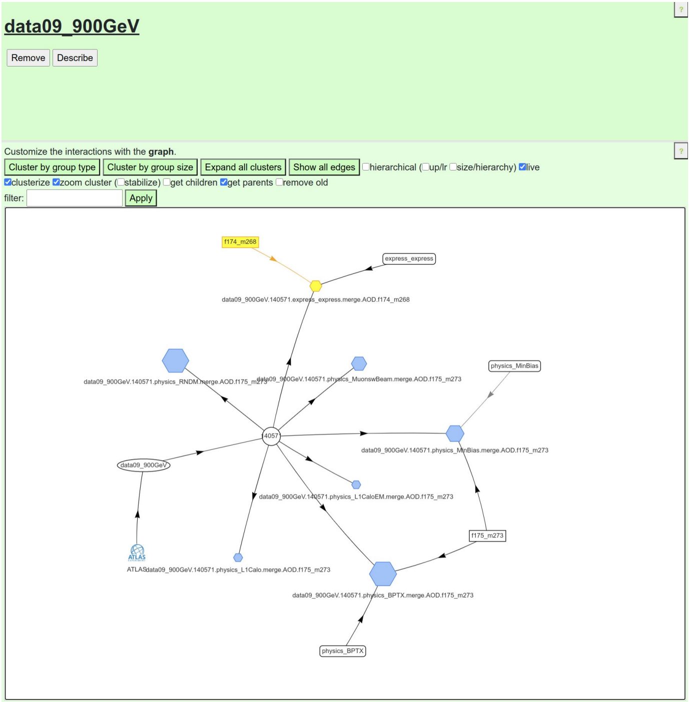


owns
owns
project:data18_900GeV
project:data18_13TeV
contains
run:349769
amitag:f933_m1960
data18_900GeV.00349769.physics_CosmicCalo.merge.AOD.f933_m1960
stream:physics_CosmicCalo

```
stylesheet.nodes.dataset = {
  graphics: {
    label:{gremlin:"sideEffect(values('prodstep').store('4'))...."},
    title:"datatype",
    subtitle:{gremlin:"values('count_events').join().toString().concat(' events')"},
    group:{gremlin:"values('version')"},
    shape:{js:"if(title == 'dataset:AOD') {shape = 'hexagon';} else {shape = 'dot';}"},
    image:" ",
    borderRadius:"0",
    borderWidth:"1",
    borderDashes:[1,0],
    value:{gremlin:"values('count_events').join().toString()"}
    },
  actions: [
    {name:"Rucio", url:{gremlin:"..."}},
    {name:"AMI", url:{gremlin:"..."}}
    ]
  }
...
```

Interactive presentation style

Actions, which can be performed on the Vertex,
may be urls to external services



data09_900GeV

Customize the interactions with the **graph**.

☐hierarchical (☐up/lr ☐size/hierarchy) ☑live

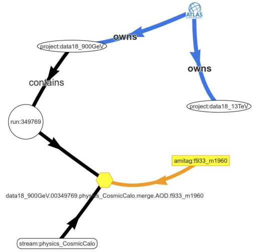
☑clusterize ☑zoom cluster (☐stabilize) ☐get children ☑get parents ☐remove old

filter:

?
Remove
Describe
Cluster by group type
Cluster by group size
Expand all clusters
Show all edges
Apply
f174_m268
express_express
data09_900GeV.140571.express_express.merge.AOD.f174_m268
data09_900GeV.140571.physics_RNDM.merge.AOD.f175_m273
data09_900GeV.140571.physics_MuonswBeam.merge.AOD.f175_m273
physics_MinBias
140571
data09_900GeV.140571.physics_MinBias.merge.AOD.f175_m273
data09_900GeV
data09_900GeV.140571.physics_L1CaloEM.merge.AOD.f175_m273
f175_m273
ATLAS
data09_900GeV.140571.physics_L1Calo.merge.AOD.f175_m273
data09_900GeV.140571.physics_BPTX.merge.AOD.f175_m273
physics_BPTX

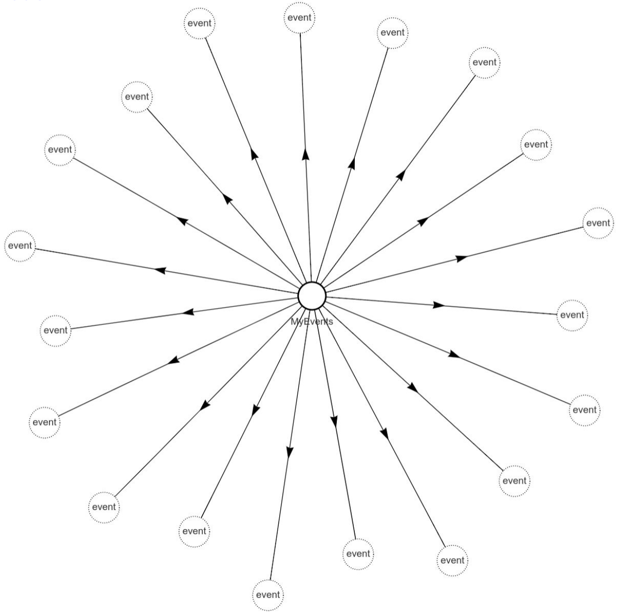
- ➢ Completely generic, connects to Gremlin server.
- ➢ Stylesheet controls graphics and context sensitive actions.
- ➢ It understands Gremlin and JavaScript.

# Virtual Collections

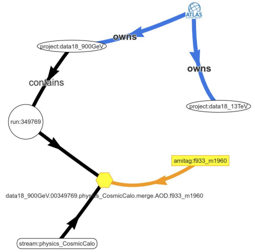


## Virtual Collection = Collection Vertex + Edges to contained Elements

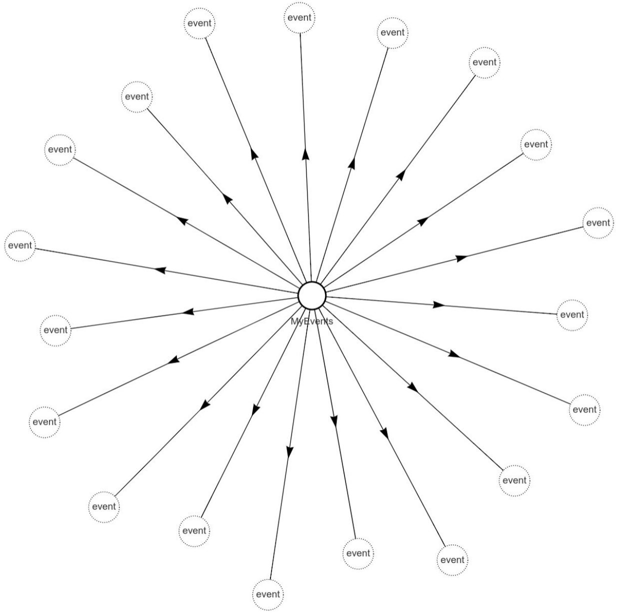
```
// Create new collection of events
eventsCollection = g.addV('ecollection')
                    .property('name','MyEvents');

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

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

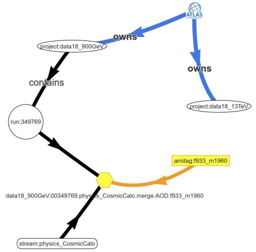


# Virtual Collections



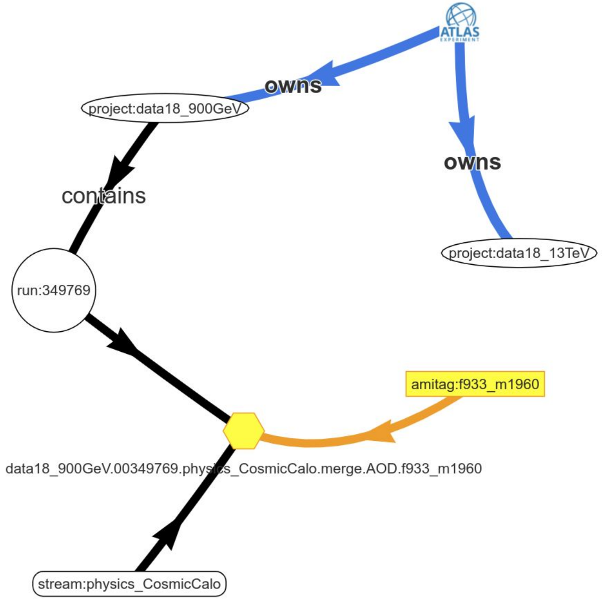
- All kinds of collections can be created
  - Manually
  - By automatic (periodic) tasks
  - By ad-hoc (exploratory) tasks
- Collections can have additional properties (annotations).
- They can be connected to other entities.
- Accessible RW remotely via Gremlin server
  - REST server with convenient clients in many languages
- Can be accessed from the Web Service.
  - Creation via Web Service can be implemented if needed.



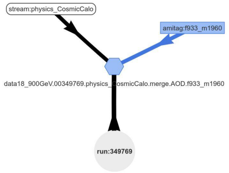

owns
owns
project:data18_900GeV
contains
project:data18_13TeV
run:349769
amitag:f933_m1960
data18_900GeV.00349769.physics_CosmicCalo.merge.AOD.f933_m1960
stream:physics_CosmicCalo

# Other Possibilities

- Creating relations (=Edges) between existing entities
  - Events, Datasets, Runs, Streams, AMItags,...
- Obvious relations are created automatically
- Others can be results of analyses tasks or added by hand
- Examples *(some are already implemented on top of the current framework, but will be more natural on top of graphs)*:
  - Edges between Datasets can carry information about overlaps
  - Trigger Statistics/Overlaps can be represented by new Vertexes, connected to their Datasets
    - They can have internal structure (Vertex=trigger, Edge=overlap,....)
- Global, structured view of all Atlas data
  - Easy navigation and manipulation
  - Natural structure (entities with relations)
  - Opens new possibilities of analyses (AI, Graph Theory,...)
- No impact on the SQL backend
- Can work on top of <u>any SQL</u> database



# Info


stream:physics_CosmicCalo
amitag:f933_m1960
data18_900GeV.00349769.physics_CosmicCalo.merge.AOD.f933_m1960
run:349769

*Using (old) Zbyszek setup @CERN*

*Need SQL schema & JDBC URL to test with new database*

**Home:** https://hrivnac.web.cern.ch/hrivnac/Activities/Packages/Atlascope
**GIT:** https://gitlab.cern.ch/atlas-event-index/GraphDB