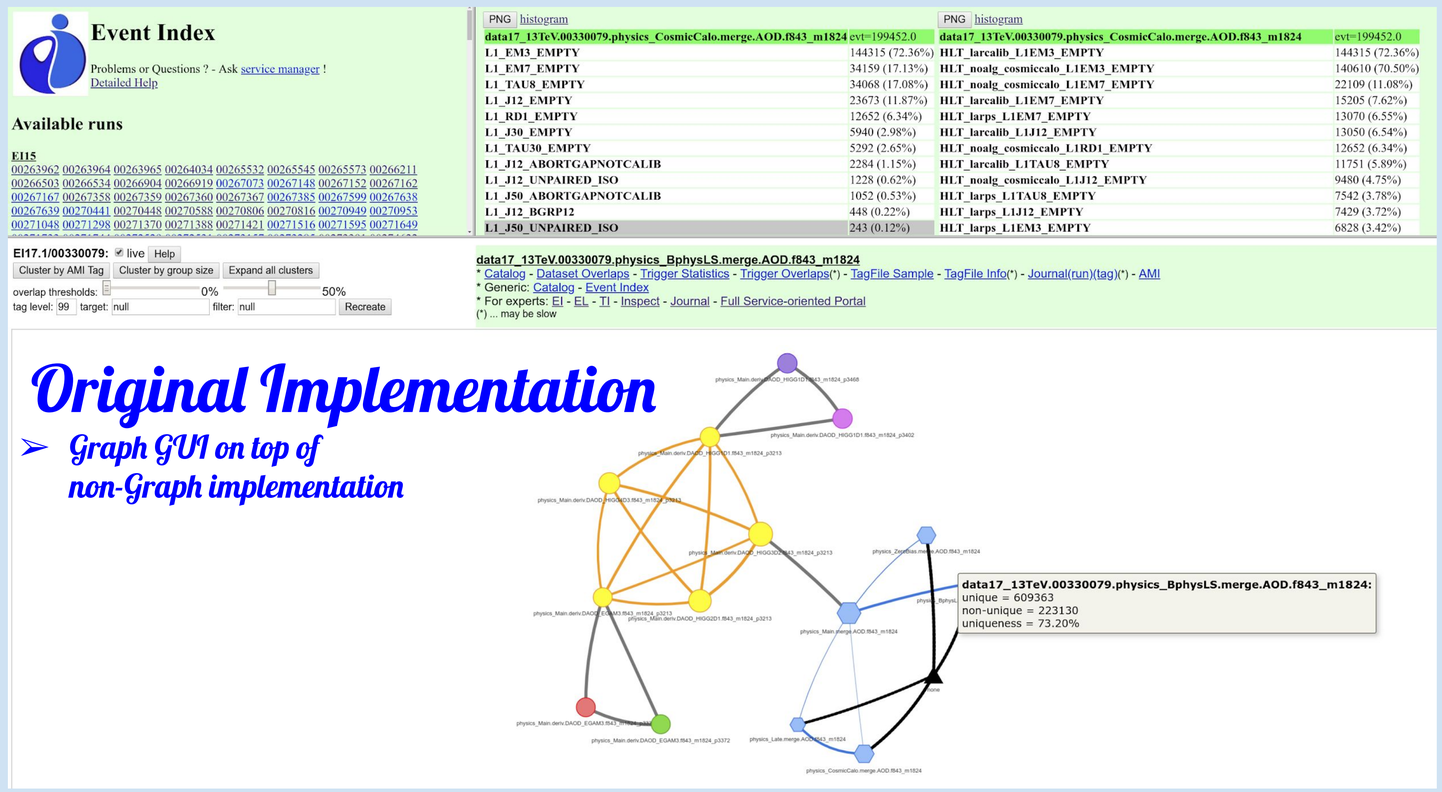
Data-centric Graphical User Interface of the ATLAS Event Index Service

Julius Hrivnac(1), Evgeny Alexandrov(2), Igor Alexandrov(2), Zbigniew Baranowski(3), Dario Barberis(4), Gancho Dimitrov(3), Alvaro Fernandez Casani(5), Elizabeth Gallas(6), Carlos García Montoro(5), Santiago Gonzalez De La Hoz(5), Andrei Kazymov(2), Mikhail Mineev(2), Fedor Prokoshin(2), Grigori Rybkin(1), Javier Sanchez(5), Jose Salt(5), Miguel Villaplana(7)
on behalf of the ATLAS Collaboration
(1) Laboratoire de l'Accelerateur Lineaire Orsay, (2) Joint Institute for Nuclear Research Dubna, (3) CERN, (4) Università e INFN Genova, (5) Instituto de Fisica Corpuscular Valencia, (6) University of Oxford, (7) Department of Physics, University of Alberta, Edmonton

ATLAS Event Index Service keeps references to all real and simulated ATLAS events. Hadoop Map files and HBase tables are used to store the Event Index data, a subset of data is also stored in the Oracle database. Several user interfaces are currently used to access and search the data. From the simple command line interface, through programmatical API to sophisticated Graphical Web Services.

- The original implementation was organized by service type (event lookup, dataset overlaps, trigger statistics,...)
  - Each service gives access to all relevant data
- Following user requests, the User Interface has evolved into a system organized by data entity (events, datasets, collections, runs,...)
  - Each entity gives access to all available services
  - While the underlying implementation was still organized by service type
- New implementation (under development for Run 3) is organized by data entity
  - Two prototypes were implemented, the ***Prototype 2*** has been selected for the final implementation
- Navigation is provided via dynamic hierarchical graph-like overview of all available data and data collections
- Data are shown together with their relations, ownership, containment or overlaps
- Some actions are provided directly by the Event Index system, others are interfaces to various external ATLAS services
- In many cases, specialised views are offered for detailed data inspection
  - Trigger histograms
  - Dataset overlaps
  - Trigger overlaps
  - ...



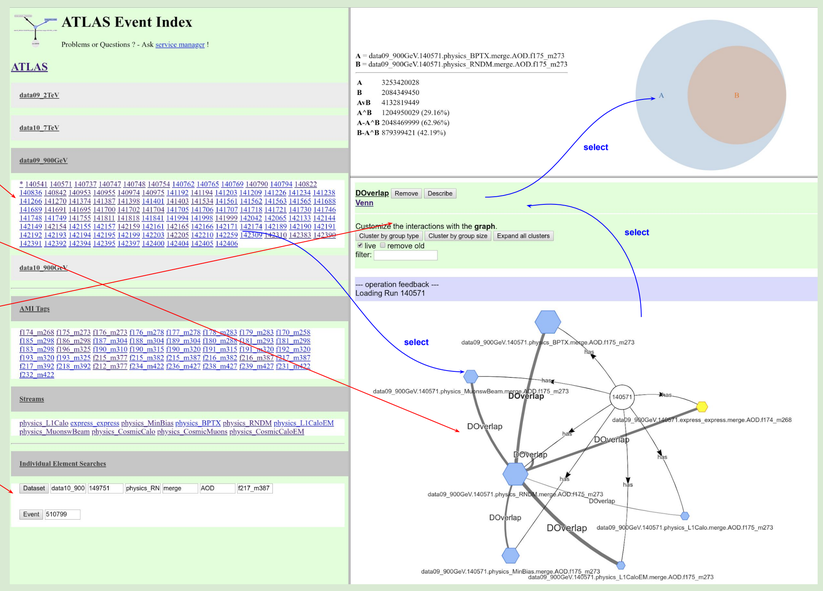
Original Implementation

- *Graph GUI on top of non-Graph implementation*

Prototype 2

- *selected for 'Run 3' Event Index implementation*

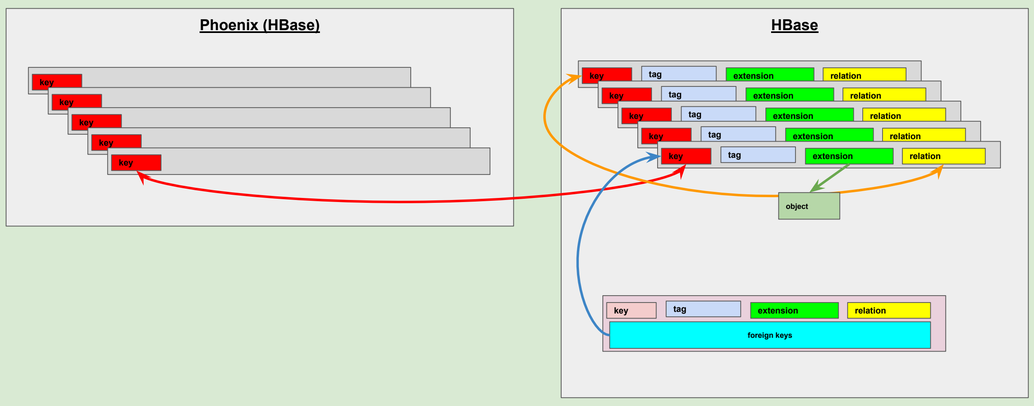
1. *Select period, run, ami tag, stream, dataset or event*
   a. *Or any other Hub (virtual collection)*
2. *Get it, together with all related entities and information stored in EI*
3. *Each entity has a set of possible actions to perform (EI or external)*
4. *Each entity can be annotated*
5. *Detailed search is possible too*
   a. *Could expose also Phoenix SQL part*



1. *Create a prototype of the Element you want to search*
2. *Fill in known values*
   a. *You can use SQL for Phoenix part*
   b. *You may choose which backend (Phoenix, HBase or both) is used for searching and data filling*
3. *Send it to the ElementFactory*
4. *Get a set of satisfying Elements, with all values filled (from both Phoenix and HBase)*
5. *Add **Tags**, **Relations** of **Extensions** to Elements*
   a. *DOverlap is_a Relation*
   b. *TStat is_a Extention*
6. *Update via ElementFactory*
   a. *HBase will be updated*

```
ElementFactory ef = …;
Dataset dprototype1 = new Dataset();
dprototype1.set("runnumber", 140571).
           .set(“project”, “data09_900GeV”).
           …;
Dataset dataset1 = (Dataset)ef.search(dprototype1).get(0);
…
Dataset dataset2 = (Dataset)ef.search(dprototype2).get(0);
dataset2.add(new DOverlap(10, 30, 50, 40, dataset1));
dataset2.add(new Tag(“mytag”, “myvalue”));
dataset2.add(new TStat(......));
ef.update(dataset2);
```

- **Prototype 1**
  - Directly storing data in JanusGraph database (on top of HBase)
  - Accessing them via standard Gremlin interface of the TinkerPop framework
- **Prototype 2 - selected for Run 3**
  - Storing data in HBase tables with Phoenix SQL API
    - To allow interoperability with other SQL-based ATLAS services
  - With additional HBase tables adding (lazy) graph structure on top
- Very generic Dynamical Web Service GUI
  - In principle re-usable for any Graph-like data
  - Customizable via stylesheets



- Both database share the same keys
- User sees one interface to both
  - All data of one key is represented by one **Element**
- HBase db is much smaller (only subset of data)
- Phoenix db is read-only
- HBase db is modifiable, it can contain
  - Simple **Tags**
    - They can be also used in search filter
  - **Extensions** with any object
    - E.g. Trigger statistics and overlap, duplicated events list,...
  - **Relations** to other elements (Graph DB emulation)
    - E.g. overlaps between datasets
- HBase can also contain Elements without Phoenix partner: **Hubs**
  - They represent pre-defined **virtual collections** of Elements
    - E.g. Amitag, Stream, Run, Project,...
  - They can be extended and searched in the same way as other Elements
- Ad-hoc virtual collections can be build also using Tags

See also Presentation Using Graph Databases in HEP (Track 4, Th, 15:15)

- *The global dynamical interactive view of all ATLAS data*
  - *With relations between data entities*
  - *Giving access to all available services*
- *User-extensible*
- *Usable for AI analyses*