A scalable new mechanism to store and serve the ATLAS detector description through a REST web API

**ATLAS GeoModel**

The ATLAS experiment\(^1\) uses GeoModel\(^2\) to describe the **geometry tree**: a set of **nodes** connected through **relationships** of different types. The GeoModel tree is **computed on-the-fly** when requested though the experiment framework, and stored **in-memory** only.

**Current limitations**

- An online Geometry DB stores primary numbers used to build objects. But **no structure** and **no relationships are saved** in the DB; and **no data are accessible offline**;
- **Very difficult to debug**; the detector description: have to go through the code;
- No way to open, explore and use the Detector Description without the **whole experiment's framework**;
- **Strong platform dependency**: SLC6 Linux the only platform supported by ATLAS;
- Not possible to interactively query the GeoModel and retrieve matching volumes only.

**FRESH, NEW IDEAS** for a way to easily **store**, **restore**, **access** and **serve** the experiment's geometry

**1st STEP** - **Decoupling** and GeoModel **persistification**, so that applications can use the Detector Description without the need of run the full ATLAS framework.

- **Geometry Tree**: the experiment’s framework is decoupled from the object properties.
- **Persistification**: the object properties are persistified in a DB. Only the detector description is needed to retrieve the properties.

**2nd STEP** – **NEW**! Easy, interactive access to the geometry through a REST API, and retrieval of geometry subsets

**TWO GOALS, TWO TECHNOLOGIES** to implement a **REST service to serve the geometry**, different approaches

**Graph DB (Neo4j\(^3\))**

*Works on GeoModel nodes*

To store, query and visualize the inner structure of the ATLAS **geometry tree**, and all the relationships between its **nodes**.

Neo4j is a **No-SQL** DB based on **nodes and relationships**, like GeoModel. Cypher is its **query language**, focused on relationships, labels, properties. Users are able to fast query all the instances of GeoModel objects used in the **actual ATLAS geometry**, and all connections and paths between them. The order of all nodes is stored, because it is used by GeoModel while traversing the tree to construct the detector. Also, users can interactively visualize the nodes and their connections in a graph, which is extremely useful to **debug** the Detector Description.

**Example queries** (on GeoModel nodes – easy to construct queries to query the tree structure)

- **Retrieve a specific GeoModel node** (GeoLogLog), based on a node property:
  ```cypher```
  ```
  match (n:GeoLogLog) where n.name = "FlyOutScapMag" return n;
  ```
  ```
  As above, but retrieve the GeoPhysLogLog which uses it:
  ```
  ```
  match (n:GeoPhysLogLog) where n.name = "FlyOutScapMag" return n;
  ```
  ```
  Get all "GeoShape" nodes used together with "Air" on Material:
  ```
  ```
  match (n:GeoShape)
  where n.type = "GeoShape"
  return n;
  ```
  ```
  Get all the children nodes of the RootVolume, and sort them by the "position" property of the CENOD relationship order is important in GeoModel because it affects how the tree is traversed:
  ```
  ```
  match (n:RootVolume)
  where n.re = "CENOD",
  return n;
  ```

**Search Engine (ElasticSearch / Kibana\(^4\))**

*Works on end physical volumes*

For a fast retrieval of the **final objects**: all transformations and attributes are **computed** and accumulated at **indexing time**.

ElasticSearch uses Lucene as query language. Relationships between the nodes and nodes’ order are not stored, all object are “**flattened**”: all the attributes are collected, and all the space transformations are **computed while inserting the data**. Users can then quickly retrieve the **final volumes**: physical volumes together with their **absolute positions** and all **attributes** related to it. The **goal** is to let applications **quickly retrieve geometry objects without having to traverse the geometry tree**.

**Example queries** (on final physical volumes – easy to construct queries to fast retrieve volumes)

- **Retrieve a specific volume**

  ```
  name:FlyOutScapMag
  ```
  ```
  Retrieve all Point volumes in the EndCap side C
  ```
  ```
  tags:Pixel AND tags:EndCapC
  ```
  ```
  Retrieve all box-shaped volumes whose material is "Air":
  ```
  ```
  shape:box AND material:Air
  ```

Both Neo4j and ES/Kibana have **REST** access as well as **web interfaces** for interactive queries

**Comparison**

<table>
<thead>
<tr>
<th>Offline access to geometry data</th>
<th>Storage, Query and Visualization of all GeoModel nodes and relationships</th>
<th>Partial retrieval of geometry data (subtrees)</th>
<th>Fast retrieval of final physical volumes with global positions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current implementation: Geometry DB</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td><strong>NEW</strong> - Persistent SQLite DB</td>
<td>V</td>
<td>(with SQL gimmick!)</td>
<td>(with SQL gimmick!)</td>
</tr>
<tr>
<td><strong>NEWEST</strong> - Neo4j Graph DB</td>
<td>V</td>
<td>V</td>
<td>V</td>
</tr>
<tr>
<td><strong>NEWEST</strong> - ElasticSearch/Kibana</td>
<td>V</td>
<td>X</td>
<td>V</td>
</tr>
</tbody>
</table>

\(^1\) ATLAS Collaboration (2008) JINST 3 S08009
\(^3\) www.neo4j.com
\(^5\) https://www.elastic.co

Riccardo Maria BIANCHI (Pittsburgh), Ilija VUKOTIC (Chicago) on behalf of the ATLAS Collaboration