# Optimizing the ATLAS metadata database architecture in preparation for High Luminosity LHC

*Jérôme* Odier[1,*], *Fabian* Lambert[1], *Jérôme* Fulachier[1], and *Pierre-Antoine* Delsart[1]

[1]Laboratoire de Physique Subatomique et de Cosmologie (LPSC) de Grenoble

**Abstract.** ATLAS Metadata Interface (AMI) is a generic ecosystem for metadata aggregation, transformation and cataloging, benefiting from more than 25 years of feedback in the LHC context. This paper shows how it is planned to optimize the ATLAS metadata database architecture in preparation for High Luminosity LHC. In particular, the new mechanism for propagating *physics parameters* along the ATLAS dataset inheritance chain will be presented. This paper also describes the automatic scoping mechanism, that has been introduced in AMI, to take advantage of database partitioning in a transparent way.

## 1 Introduction

ATLAS Metadata Interface [1][2][3] (AMI) is a generic ecosystem for metadata aggregation, transformation and cataloging. Benefiting from more than 25 years of feedback in the LHC context, the second major version was released in 2018.

AMI provides a wide array of tools (command line tools, lightweight clients) and Web interfaces for searching scientific data by metadata criteria. It was designed to guarantee scalability, flexibility and maintainability. It perfectly fits the needs of scientific experiments in a small or big data contexts.

AMI is currently used by scientific collaborations such as ATLAS [4], NIKA2 [5] and n2EDM [6].

ATL-SOFT-PROC-2025-038
24 February 2025

The ATLAS dataset catalogs are based on an initial Oracle [7] SQL schema that has evolved only through the addition of tables and fields. One particularity is that a new catalog (a database) is duplicated each year for data and each production campaign for Monte-Carlo (MC) simulations. This is sub-optimal in terms of maintenance and cross-referencing between catalogs.

In preparation for High Luminosity LHC (a.k.a. LHC Run 4), it was decided to create a new single database catalog, optimized for SQL query speed and maintainability, using more modern techniques such as partitioning and scoping.

The AMI ecosystem features a domain-specific language, Metadata Query Language [8] (MQL), enabling database queries to be made without knowing the underlying relationships between tables. New features have been developed to automatically manage scoping.

*e-mail: ami@lpsc.in2p3.fr

## 2 Current metadata database architecture

### 2.1 The dataset and AMI-Tag catalogs

In the current architecture, there are 28 Oracle catalogs containing the metadata for around 10 million datasets and billions of files. A dataset (respectively file) is identified by an auto-incremented primary key and a logical dataset name (respectively logical file name) string. Most of the time, the metadata is stored in dedicated columns, which is very inflexible.

AMI also maintains a single database containing a set of parameters which define the ATLAS workflows that are used to generate each dataset. This is the AMI-Tags database. An AMI-Tag is identified by a letter, representing a production step (event generation, simulation, reconstruction, ...), and an auto-incrementing number, e.g. e8504. As a dataset can be produced from another dataset, AMI-Tags can be stacked, for example e8504_s4017_s3993_r14249.

In the current architecture, the AMI-Tag chain of a dataset is represented as a single string. This approach makes data selection, based on the workflow parameters, impractical and inefficient.

### 2.2 The *global dataset* and the *dataset tree* tables

AMI has a dedicated global table (called *global dataset* table, with ~13 million entries) maintaining the list of each logical dataset name and its own catalog name. Datasets may have parent-children relationships with others which can form chains or trees of datasets. To keep track of these relations, a second table (called *dataset tree* table, with ~84 million entries) maintains a family tree using a few fields such as a double foreign key to two datasets, their distance and two flags *isroot* and *isleaf*.

Neither the *global dataset* table nor the *dataset tree* table is partitioned, which may negatively impact performance.

### 2.3 The *physics parameters* mechanism

For each MC simulation catalog, there is a mechanism that allows ATLAS physics groups to attach additional metadata (called *physics parameters*, e.g. cross-section, ...) to datasets and propagate them to children. At any point in the inheritance chain, the value of a *physics parameter* can be modified, or propagation can be stopped. A *physics parameter* can also change over time. Values are thus defined for a given interval of validity.

This mechanism suffers from several problems. The first is a significant duplication of information, one for each dataset. The second is the slowness of the propagation algorithms, sometimes over thousands of datasets, and the risk of data corruption in the event of a crash in the metadata management tasks. Finally, cross-selection across multiple dataset catalogs based on *physics parameters* may be inefficient.

## 3 Proposed metadata database architecture for Run 4

### 3.1 The new dataset and AMI-Tag table

The schema of the new tables for storing dataset and file metadata remains similar to the old one. However, all the tables are partitioned in order to contain all the data and MC simulations within a single catalog.

It was decided to use the *project* parameter (which refers to large data campaigns, e.g. mc16_13TeV, data24_calib, ...) as the partitioning scope. Indeed, this corresponds to an optimal criteria to divide the datasets into sub-categories, and thus to balance the number of entries in the different partitions.

A dedicated table is used to list the *project* values, and each partitioned table contains the following SQL fragment in its definition:

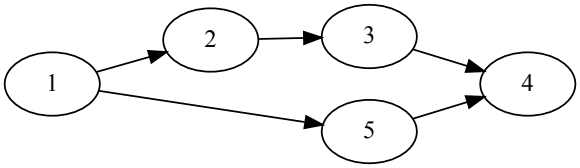
```
FOREIGN KEY ("scope") REFERENCES "project" ("project"),
PARTITION BY LIST ("scope")
```

AMI-Tags tables will benefit from the same partitioning and scoping system, and will be linked to datasets via a dedicated bridge table.

### 3.2 The new *dataset tree* table

As there is now only one single dataset table, it replaces the old *global dataset* table (see previous section). The new *dataset tree* table remains relatively similar to the old one. To represent relationships between datasets, each row contains a foreign key to a parent dataset (a.k.a. source), a second foreign key to a descendant dataset (a.k.a. destination) and the distance between them (a.k.a. generation depth). Each of the two datasets is accompanied by its respective scope (i.e. the *project* value used for partitioning in the dataset table). The table is partitioned on the *destination* scope, as we will see later that this is the most natural solution for optimizing the retrieval of the *physics parameters* associated with a dataset.

To illustrate this, consider the following lineage graph for datasets 1, 2, 3, 4 and 5:




This leads to the following entries in the *dataset tree* table where the tuple (Source, Destination, Path) is unique:

| Source | Destination | Distance | Path |
|---|---|---|---|
| 1 | 1 | 0 | /1/ |
| 1 | 2 | 1 | /1/2/ |
| 1 | 3 | 2 | /1/2/3/ |
| 2 | 2 | 0 | /2/ |
| 2 | 3 | 1 | /2/3/ |
| 3 | 3 | 0 | /3/ |
| etc... | | | |
| 1 | 4 | 3 | /1/2/3/4/ |
| 1 | 4 | 2 | /1/5/4/ |

The *path* column is a string concatenating the primary keys of each fragment of the inheritance chain. This is very useful for performing integrity checks or corrections, particularly when importing the old *dataset tree* content to the new one. This also makes it possible to distinguish the relationship $1 \rightarrow 4$ through 2 and 3 from the relationship $1 \rightarrow 4$ through 5.

This method of representing a graph allows the complete lineage of a dataset to be retrieved with a single query and without recursion.

### 3.3 The new *physics parameters* mechanism

The new *physics parameters* system must have the same functionality as the previous one. However, it must implement partitioning and scopes, and must no longer duplicate metadata along dataset inheritance chains.

To do so, we have introduced two tables. The first one represents a *physics parameter* definition, without any value, with its name, unit and description (for example a cross-section expressed in nanobarns). The second table is a sparse structure that contains either an integer, a float, a boolean, a date, a string or a JSON document. It represents a *physics parameter* value. It contains two foreign keys, one linking it to a *physics parameter* definition, and the other linking it to the dataset with which it is associated. It contains a *group* parameter to indicate which physics group is responsible for this *physics parameter* value. It also contains a date to indicate the start of validity. A new entry with a more recent date overwrites the value of the previous one.

To control propagation along the dataset inheritance chain, there are two additional parameters: *propagate* indicates whether the value should be propagated to children or only specified for the linked dataset and *overwrite* indicates whether the value should be overwritten if already present in a parent dataset. No values are duplicated. It is possible to retrieve all the *physics parameters* of a dataset with a single SQL query that joins the *dataset tree* and the two tables of *physics parameter* definitions and values.

The new database schema exposes two SQL views, one showing the latest *physics parameters* values (*current_dataset_parameters* view), and a other showing the complete history (*all_dataset_parameters* view).

The complex underlying SQL queries are described in [9].

## 4 Automatic scoping with the MQL

An important value-added feature of the AMI ecosystem is the Metadata Query Language (MQL). This domain-specific language makes it possible to perform queries without knowing the underlying relationships between tables.

It seemed necessary to be able to manage scoping automatically. To achieve this, two metadata have been added to the catalog column descriptions maintained by AMI. The first indicates whether a column is a scope, and the second is an arbitrary label. When AMI translates a MQL query into a SQL query, it automatically determines the joins. At this level, if the two joined tables have a scope field with the same label, then the scopes must be equal.

## 5 Performances

### 5.1 Test samples

A significant sample of data has been imported into the new database for testing:

- **13 million datasets** with about 360 scopes,
- **84 million parent-child relationships** stored in the dependency tree,
- **9 million *physics parameters*** linked to datasets, including their evolution over time.

### 5.2 Data access speed

The challenge is to ensure fast queries to retrieve a dataset's current parameters. For a dataset in the mc16_13TeV scope, with a generation depth of 6:

- Query time is 12–60 ms using the global dataset table from the new structure (13M datasets),
- Query time is 5–12 ms using the mc16 dataset table from the old structure (5M datasets).

Speeds are of the same order of magnitude, even though the new catalog has more entries and no longer duplicates information (which implies more computation). Overall, performance remains excellent.

### 5.3 Useful data volumes

The following comparison highlights the number of *physics parameter* records in the new structure versus the old one:

| MC Campaign | New Structure | Old Structure | Reduction (%) |
|---|---|---|---|
| mc23 | 315,063 | 1,315,168 | 76% |
| mc15 | 5,256,414 | 11,105,723 | 52% |
| mc16 | 4,192,572 | 46,696,040 | 91% |

The volume of useful data is significantly reduced. Furthermore by avoiding duplication, inconsistencies caused by copy task failures in the old schema are eliminated.

### 5.4 Query optimizations

Three Oracle-specific techniques were used to optimize the import and select queries:

• *The "PARALLEL" hint*

To parallelize queries on the database server when selecting data for import or update:

```
SELECT /*+ PARALLEL(dataset_tree, 10) */ ... FROM ... WHERE ... ;
```

It allows to reduce the query execution speed by parallelizing the load on Oracle server nodes.

• *The "NO_MERGE" hint*

This Oracle hint is particularly effective for nested queries:

```
SELECT ... FROM ... WHERE NOT EXISTS (
  SELECT /*+ NO_MERGE */ 1 FROM ... WHERE ...
);
```

It allows to reduce the size of intermediate result issued from tables crossing.

• *Partitioning and scoping*

Partitioning is primarily used in the new catalog, which contains much larger tables. However, Oracle may not always use partitioning if indexes are available.

• *Query using indexes*

When an index is used, Oracle do not need to use the partitioning, for instance:

```
SELECT * FROM "dataset" WHERE "ldn" = '<ldn>' AND "scope" = '<scope>';
```

Oracle plan:

```
| Id  | Operation                           | Name            | Rows  | Bytes | Cost (%CPU) | Time     |
|-----|-------------------------------------|-----------------|-------|-------|-------------|----------|
|  0  | SELECT STATEMENT                    |                 |  1    |  246  | 4 (0)       | 00:00:01 |
|  1  | TABLE ACCESS BY GLOBAL INDEX ROWID | dataset         |  1    |  246  | 4 (0)       | 00:00:01 |
|  2  | INDEX UNIQUE SCAN                   | SYS_C0016397672 |  1    |       | 3 (0)       | 00:00:01 |
```

When no index could be used, then Oracle use the partitioning, for instance:

```
SELECT * FROM "dataset" WHERE "ldn" LIKE 'xxx%' AND "scope" = '<scope>';
```

Oracle plan:

```
| Id  | Operation             | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
|-----|-----------------------|---------|-------|-------|------------|----------|
|  0  | SELECT STATEMENT      |         |  5512K|  1293M| 53360   (1)| 00:00:03 |
|  1  |  PARTITION LIST SINGLE|         |  5512K|  1293M| 53360   (1)| 00:00:03 |
|  2  |   TABLE ACCESS FULL   | dataset |  5512K|  1293M| 53360   (1)| 00:00:03 |
```

## 6 Conclusion

In preparation for LHC Run 4, a new database schema is currently being developed in order to store the scientific metadata of the ATLAS datasets. It has been designed as a single catalog using partitioning and scoping techniques to simplify maintainability and data cross-referencing queries. Tests on a dataset of 13 million entries suggest that performance is very good.

The AMI Metadata Query Language has been extended to automatically manage scopes in joins.

Moving forward, the challenge will be to import all metadata from the old structure into the new one, rewrite the metadata acquisition tasks to populate the new structure directly, and finally update the user interfaces.

These operations will span over several months and must be executed without any service interruptions.

## 7 Acknowledgements

Over the years, we have been helped and supported by many people at CC-IN2P3 and in the ATLAS collaboration, in particular: Osman Aidel, Luca Canali, Philippe Cheynet, Benoît Delaunay, Elizabeth Gallas, Pierre-Etienne Macchi, Mattieu Puel and Jean-René Rouet.

## References


[1] J. Fulachier, O. Aidel, S. Albrand, F. Lambert, Looking back on 10 years of the ATLAS Metadata Interface. Proceedings of the 20th International Conference on Computing in High Energy and Nuclear Physics (CHEP) J. Phys.: Conf. Ser. **513**, 042019 (2013). https://doi.org/10.1088/1742-6596/513/4/042019

[2] J. Odier, O. Aidel, S. Albrand, J. Fulachier, F. Lambert, Evolution of the architecture of the ATLAS Metadata Interface (AMI). Proceedings of the 21st International Conference on Computing in High Energy and Nuclear Physics (CHEP) J. Phys.: Conf. Ser. **664**, 042040 (2015). https://doi.org/10.1088/1742-6596/664/4/042040

[3] J. Odier, F. Lambert, J. Fulachier, The ATLAS Metadata Interface (AMI) 2.0 metadata ecosystem: new design principles and features. Proceedings of the 23rd International Conference on Computing in High Energy and Nuclear Physics (CHEP) EPJ Web of Conf.: Conf. Ser. **214**, 05046 (2019). https://doi.org/10.1051/epjconf/201921405046

[4] The ATLAS Collaboration, The ATLAS Experiment at the CERN Large Hadron Collider. JINST **3**, S08003 (2008). https://doi.org/10.1088/1748-0221/3/08/S08003

[5] M. Calvo, A. Benoît, A. Catalano et al., The NIKA2 Instrument, A Dual-Band Kilopixel KID Array for Millimetric Astronomy. J Low Temp Phys **184**, 816–823 (2016). https://doi.org/10.1007/s10909-016-1582-0

[6] N.J. Ayres, G. Ban, L. Bienstman et al., The design of the n2EDM experiment. Eur. Phys. J. C **81**, 512 (2021). https://doi.org/10.1140/epjc/s10052-021-09298-z

[7] Oracle Database [database management system]: https://www.oracle.com/database/ [accessed 2024-12-12]

[8] J. Odier, F. Lambert, J. Fulachier, Design principles of the Metadata Querying Language (MQL) implemented in the ATLAS Metadata Interface (AMI) ecosystem. Proceedings of the 24th International Conference on Computing in High Energy and Nuclear Physics (CHEP) EPJ Web of Conf.: **245**, 04044 (2020). https://doi.org/10.1051/epjconf/202024504044

[9] Description of the SQL queries for AMI *physics parameters*: https://atlas-ami.cern.ch/?subapp=document&userdata=singleCatalog [accessed 2024-12-12]