



The AMI 2.0 metadata ecosystem: new design principles and features

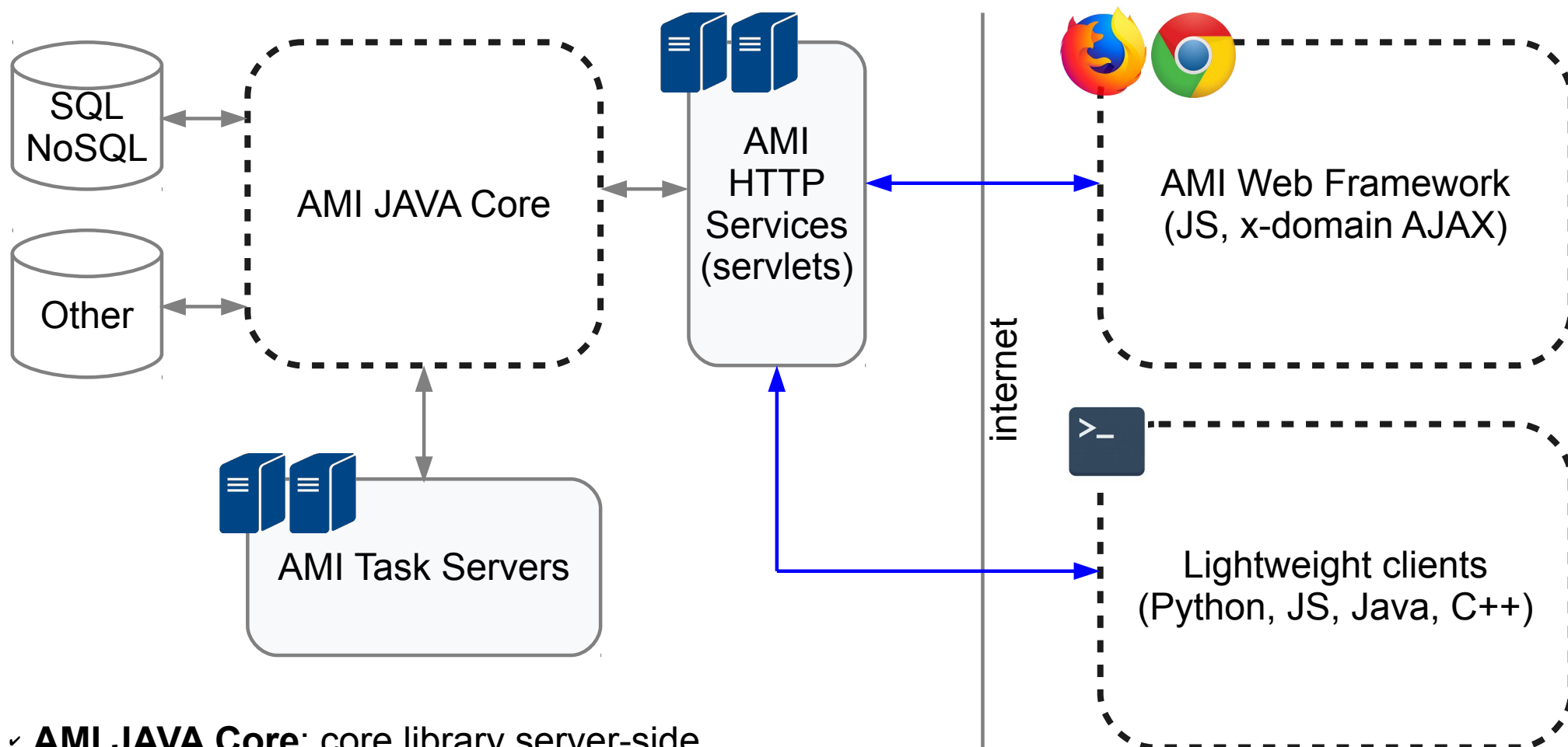
P.-A. Delsart, J. Fulachier,
F. Lambert, J. Odier

What is AMI?

- AMI (ATLAS Metadata Interface) is a generic ecosystem for metadata:
 - Primitives for metadata extraction and processing
 - Heterogenous and distributed database connectivity
 - High level tools for selecting data by metadata criteria
- The ecosystem has development kits for:
 - Developing JAVA business objects (server-side)
 - Developing metadata-oriented Web applications (client-side)
- AMI is designed for:
 - Scalability, evolutivity and maintainability

- Used by the ATLAS production system

Overview of the AMI ecosystem

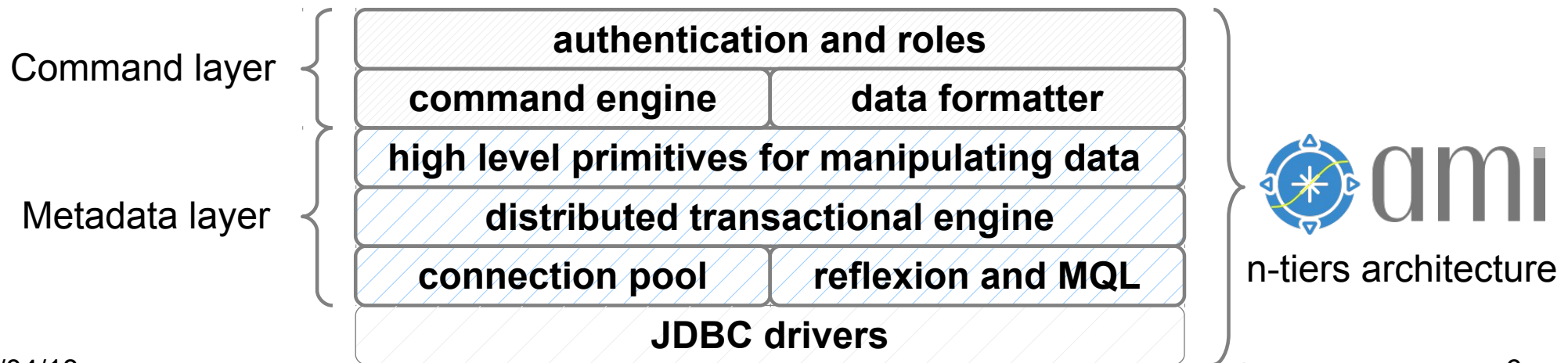


- ✓ **AMI JAVA Core:** core library server-side
- ✓ **AMI HTTP Services:** AMI Commands (proprietary) or REST API
- ✓ **AMI Task Server:** aggregating and processing metadata
- ✓ **AMI Web Framework:** developing metadata-oriented Web applications
- ✓ **Lightweight clients:** accessing AMI from anywhere

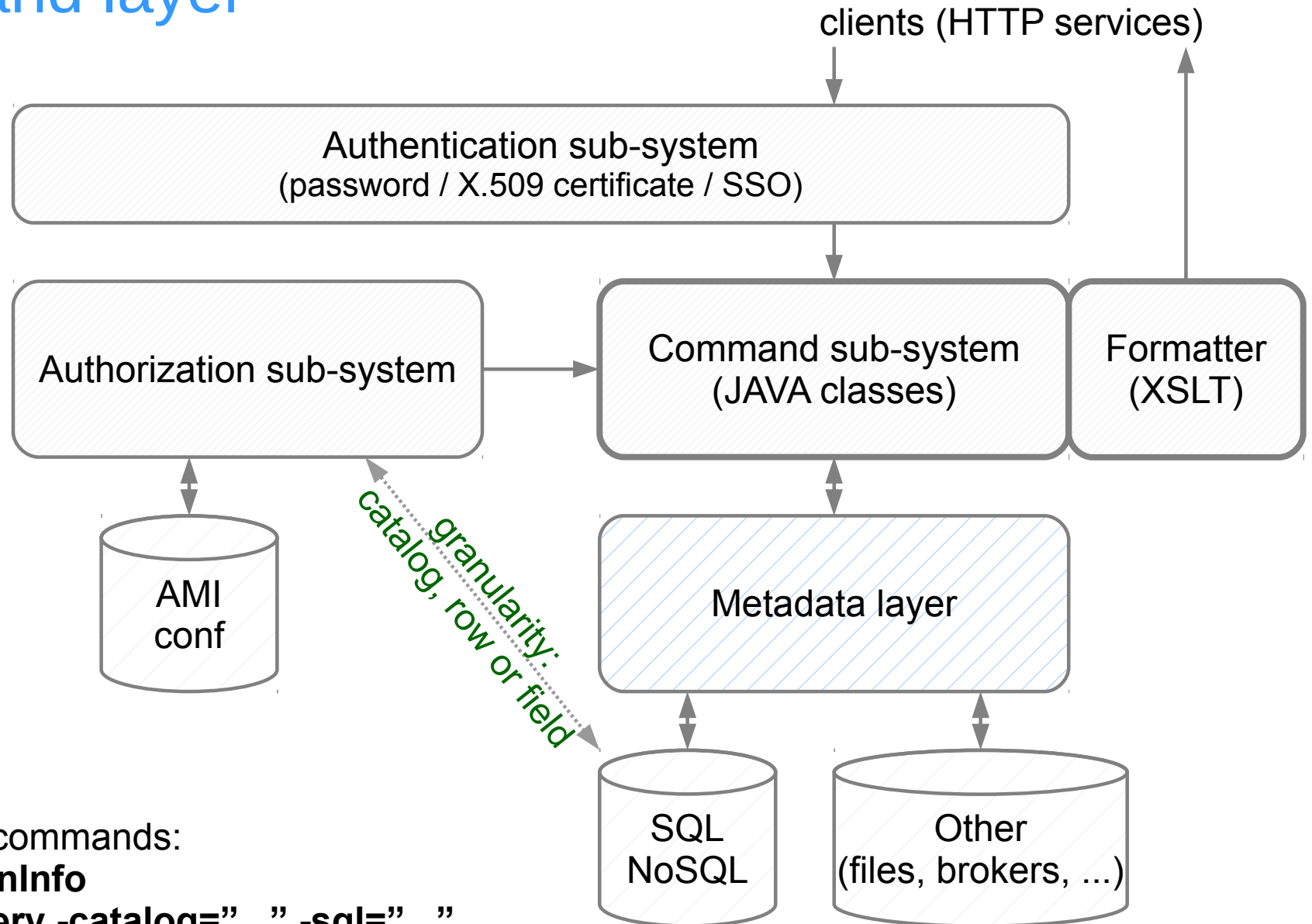
AMI JAVA Core

AMI JAVA Core features

- AMI JAVA Core is the central part of the AMI ecosystem
- Main features:
 - Authentication and authorizations
 - Command engine (~100 generic commands, ~500 ATLAS-specific commands)
 - Metadata queries (trivial [SQL, MQL] or more complex, read or write), experiment-specific commands, service administration, ...
 - High level primitives for manipulating data
 - DB rowsets, JSON documents, XML documents, remote access, ...
 - Metadata Query Language (MQL) and Structured Query Language (SQL)



Command layer



Example of commands:

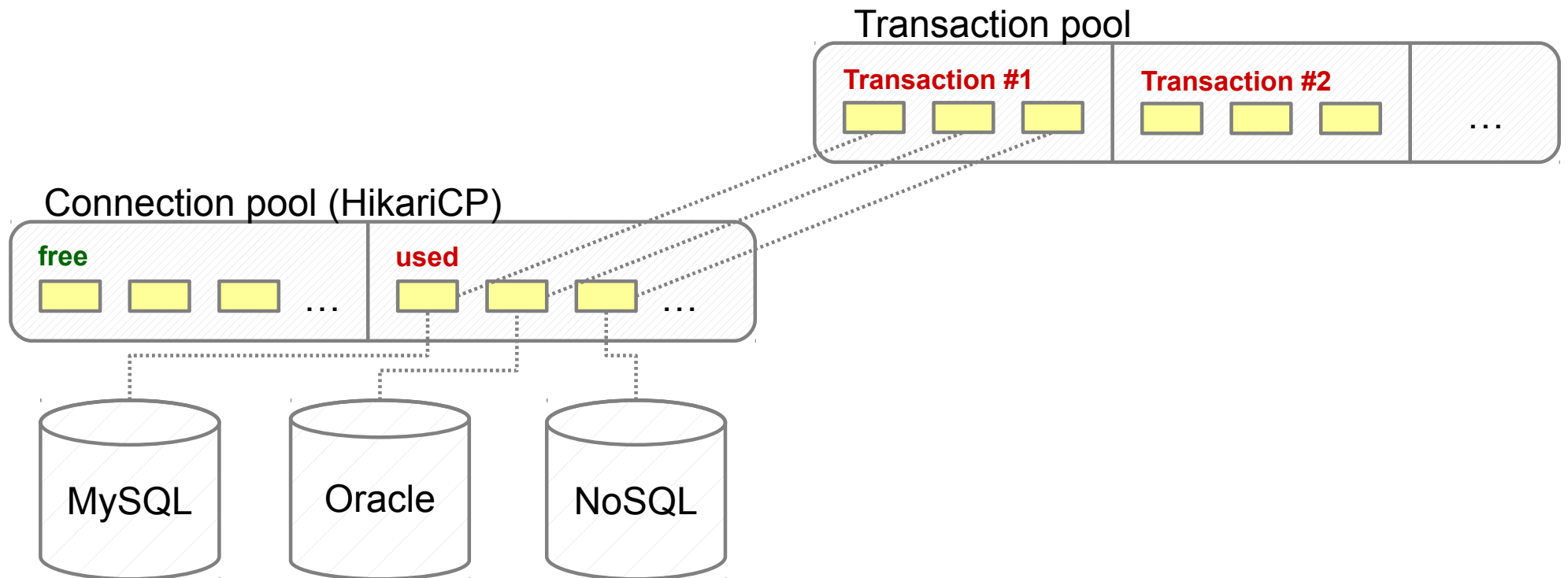
GetSessionInfo

SearchQuery -catalog="..." -sql="..."

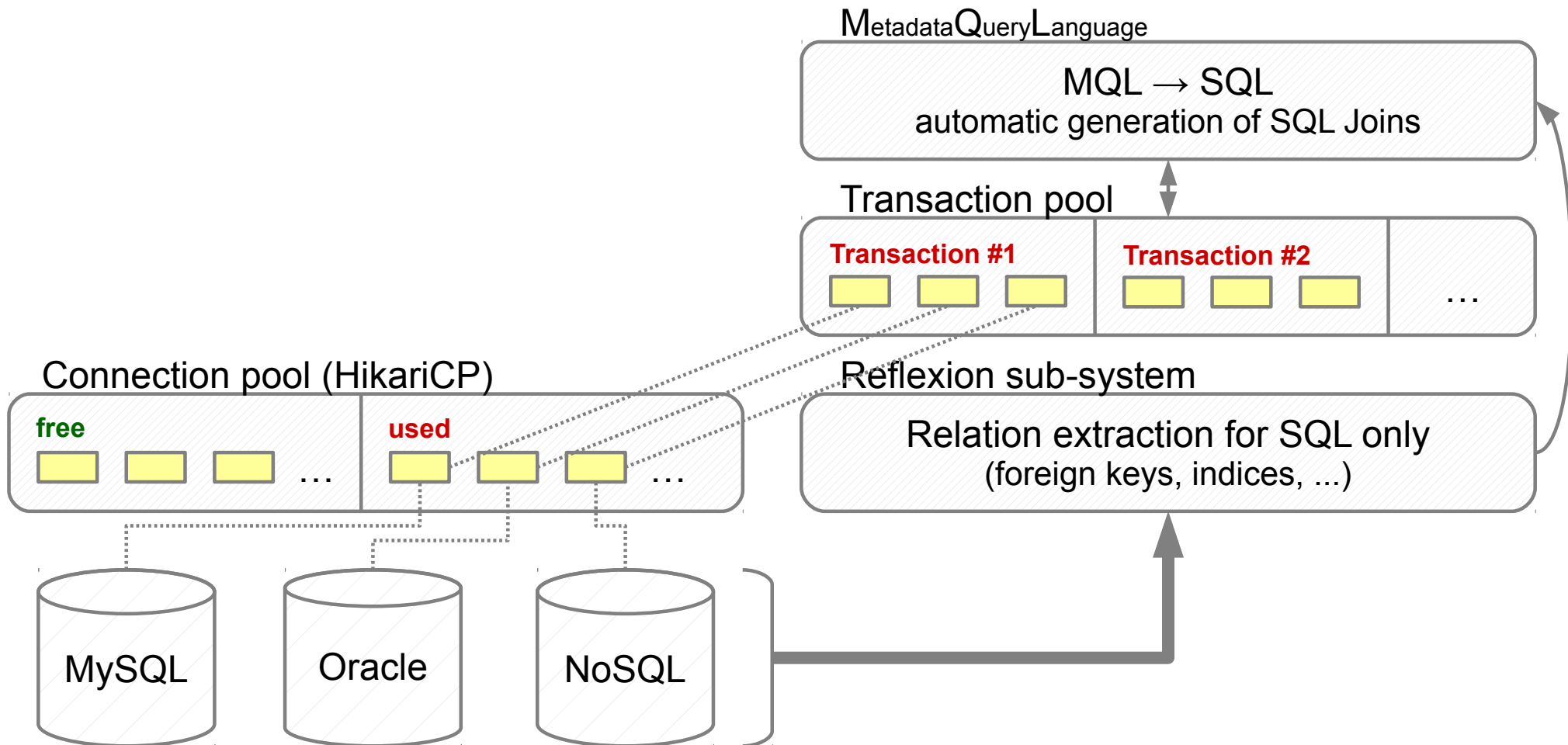
GetDatasetInfo -logicalDatasetName="..."

(for ATLAS, getting detailed dataset info)

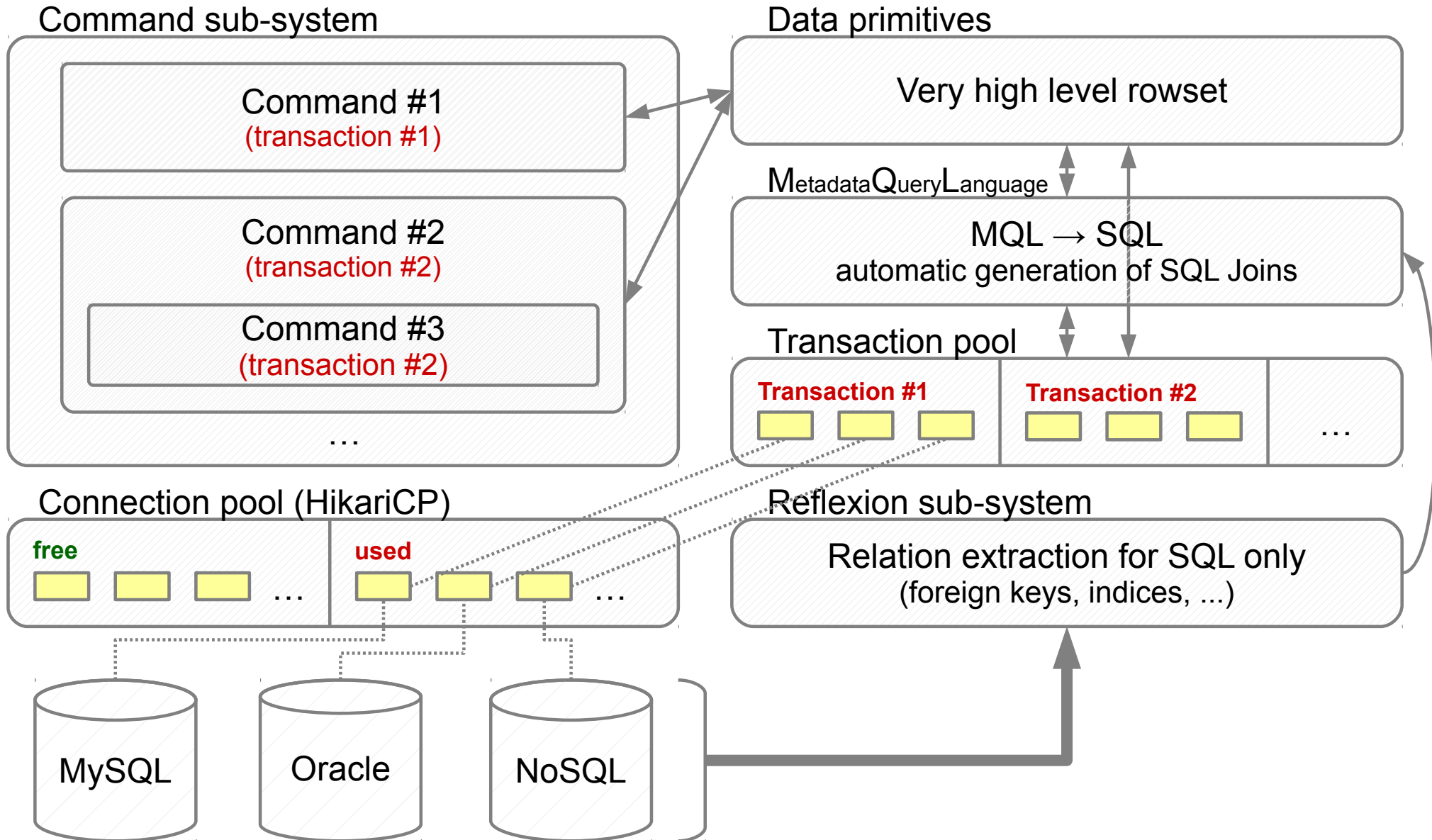
Metadata layer



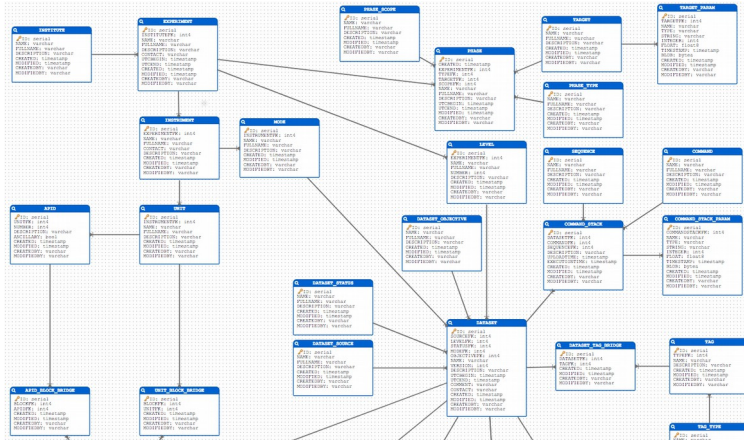
Metadata layer



Metadata layer



More about MQL



- MQL is a kind of SQL without **FROM** clause nor join
- It makes it possible to build queries without (precisely) knowing relations
- Joins are automatically generated from the reflexion sub-system info

- MQL turns DB-oriented point of view to metadata-oriented point of view
- When there are cycles in relations, there is a dedicated syntax to apply path constraints
- Example:

```
SELECT BLOCK.ID
WHERE BLOCK_PARAM.NAME='foo'
AND DATASET.NAME{BLOCK.DATASETFK}='bar'
```

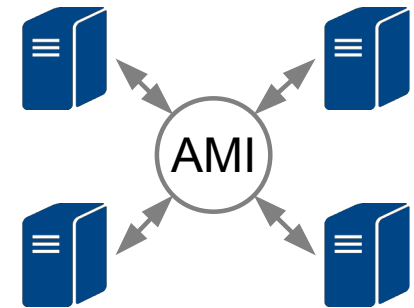
MQL
to
SQL

```
SELECT `BLOCK`.`ID`
FROM `BLOCK`
WHERE (`BLOCK`.`ID` IN
  (SELECT `BLOCK`.`ID`
   FROM `BLOCK`,
        `BLOCK_PARAM`
   WHERE `BLOCK_PARAM`.`NAME` = 'foo'
   AND (((`BLOCK_PARAM`.`ID`,
          `BLOCK`.`ID`) IN
        (SELECT `BLOCK_PARAM`.`ID`,
              `BLOCK`.`ID`
         FROM `BLOCK_PARAM`,
              `BLOCK`
         WHERE `radardb`.`BLOCK_PARAM`.`BLOCKFK` = `radardb`.`BLOCK`.`ID`))))
AND `BLOCK`.`ID` IN
  (SELECT `BLOCK`.`ID`
   FROM `BLOCK`,
        `DATASET`
   WHERE `DATASET`.`NAME` = 'bar'
   AND (((`DATASET`.`ID`,
          `BLOCK`.`ID`) IN
        (SELECT `DATASET`.`ID`,
              `BLOCK`.`ID`
         FROM `BLOCK`,
              `DATASET`
         WHERE `radardb`.`BLOCK`.`DATASETFK` = `radardb`.`DATASET`.`ID`))))))
```

AMI Task Server

AMI Task Server features

- The AMI Task Server is used for:
 - Extracting metadata from primary sources (pull mode)
 - (Re)Processing metadata
 - Storing metadata in AMI
- It can run any kind of tasks (shell, python, java, ...)
- When needed, it can benefit from the AMI Java Core library
- Main features:
 - Kind of super CRON
 - Web interfaces and monitoring
 - The AMI Task Server is distributed
 - Mutual exclusion mechanism between tasks
 - Priority lottery scheduler for avoiding starvation (not real time)
 - One shot tasks



for instance
4 AMI Task Servers

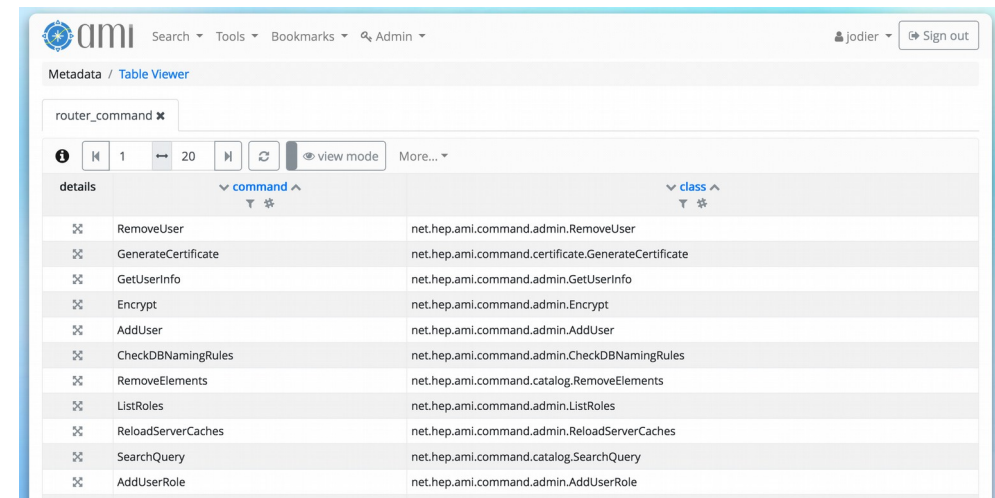
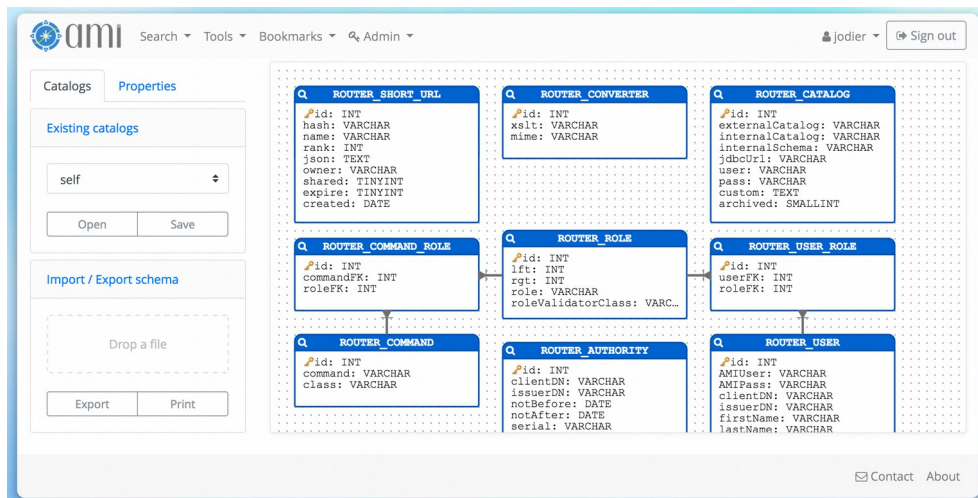
In ATLAS:

- Metadata extraction from Tier0 (real data)
- Metadata extraction from ProdSys (simulated data, reprocessing)
- ActiveMQ messages from RUCIO (data placement)
- ~50 other tasks

AMI Web Framework

AMI Web Framework (AWF)

- A Web framework for designing metadata-oriented applications
- AWF can be used without the AMI Java HTTP Service
 - Server-side, libraries AMIMini{PHP,Python,JAVA} can easily bridge AWF to existing services
- AWF is based on standard technologies:
 - JS6 (transpiled to JS5), CSS3, HTML5
 - JQuery, Twitter Bootstrap 4
 - AMI-Twig (a JavaScript homemade version of the Twig template engine)



Features and patterns

- Authentication
- URL router, short URLs
- Object paradigm (emulated)
- Sub-applications and reusable graphic controls
- Centralized resource live cycle management (CSS, JS, JSON, xml, Twig files; AMI sub-applications; AMI controls)
- Wizards for generating sub-application and control skeletons
- MVC pattern:
 - Model → AMI commands
 - View → TWIG templates
 - Controller → classes `ami.SubApp`, `ami.Control` (JavaScript)

Default controls and applications

- Controls can be embedded in external Web pages such as wikis
 - See details in poster session
- Applications are generally built by assembling controls
- Main available controls:
 - Dialog boxes
 - Controls for searching (Google-like Search, Criteria Search, ...)
 - Controls for displaying (Schema Viewer, Tab, Table, Element Info, ...)
 - Controls for annotating entities (WhiteBoard, ...)
- Main available applications:
 - Basic CMS
 - AMI command interpreter
 - Admin Dashboard and Monitoring
 - Schema Viewer, Table Viewer, Simple Search, Criteria Search, Search Modeler, ...

Screenshots

Searching ATLAS datasets by criteria

Displaying search results in AMI

Search Form

View Selection Selected datasets: 39287 (events: 44508947520, files: 7130962)

Simulated Data mc16

☒ Valid datasets

☐ projectName

☐ productionStep

☒ dataType

☒ version (AMI Tag)

☐ logicalDatasetName

☐ campaign

☐ subcampaign

☐ bunchspacing

☐ geometryVersion

☐ prodsysStatus

☐ datasetNumber

☐ generatorName

☐ ecmEnergy

☐ generatorTune

version (AMI Tag)

Any
e2623_s2997_r8957
e2623_s2997_r8957_r8996
e2623_s2997_r9191
e2623_s2997_r9191_r9128
e2623_s2997_r9370
e2623_s2997_r9370_r9315

☐ Exact

dataType

Any
AOD
DAOD_BPHY1
DAOD_BPHY4
DAOD_BPHY5
DAOD_BPHY6
DAOD_BPHY7

Select

Browse V_AMITags x

1 10 More...

details	AMITag	tagType	tagNumber	SWReleaseCache	transformationName
	e6764	e	6764	AtlasProduction_19.2.5.11	Generate_tf.py
	e6763	e	6763	MCProd_19.2.5.33.3	Generate_tf.py
	r10570	r	10570	Athena_21.0.72	Trig_reco_tf.py
	r10569	r	10569	AthenaP1_21.1.29	Trig_reco_tf.py
	m1984	m	1984	Athena_21.0.72	AODMerge_tf.py
	x566	x	566	Athena_21.0.72	Reco_tf.py
	r10568	r	10568	TrigMC_20.7.9.8.7	Reco_tf.py
	d1475	d	1475	TrigMC_20.7.9.8.7	OverlayChain_tf.py
	r10567	r	10567	AthenaP1_21.1.31-21.1-2018-06-04T2139	Trig_reco_tf.py
	r10566	r	10566	AthenaP1_21.1.31-21.1-2018-06-04T2139	Trig_reco_tf.py

Support

Webs

- Main
- Main Archive
- Plugins
- Sandbox for tests
- Public webs

Welcome Guest

Login or Register

MC16 Dibosons samples

Datasets below have their "processGroup" in the form "Diboson_XXX_YYY"

MC16 Powheg samples

dibosons processes generated by Powheg

Datasets found : 1163

Use % for wildcarding

mc16_13TeV.363893.PowhegPy8EG_CT10nloME_AZNLOCTEQ6L1_ZZqqll_mq20mi20

EVNT HITS AOD NTUP_PILEUP dataset

Details

Metadata

identifier	301372
logicalDatasetName	mc16_13TeV.363893.PowhegPy8EG_CT10nloME_AZNLOCTEQ6L1_ZZqqll_mq20mi20.simul.HITS.e5154_s3126
nFiles	5983
totalEvents	7977350
totalSize	5553506367244
dataType	HITS
prodsysStatus	EVENTS PARTIALLY AVAILABLE
completion	99.97

EL METADAT

dataset

dataset_k

dataset_o

file

jobOpt

prodsys

physicsParam

ami Datasets Files AMI-Tags Nomenclature Physics Tools Issue reporting

AMI-Tags / Add/Reset/Clone 66 new version

prod. Add a missing release yourself

AMI-Tags

Browse

Show/Edit

Compare

Add/Reset/Clone

Add tag Reset tag Clone tag

AMI-Tag name

r8410

Import

AMI-Tag type

r

Production step

recon (f, k, r, v, w, x)

Release

AtlasProduction_20.20.5.4

Transform

Reco_tf.py

Wiki about substeps

additional dpds

outputAOD_HSG2File file<AOD> outputDESIGN_ZMUMUFile file<ESD> outputNTUP_EHBIASFile file<NTUP>

outputNTUP_FASTMONFile file<NTUP> outputNTUP_FastCaloSimFile file<NTUP>

outputNTUP_HEGNOISEFile file<NTUP> outputNTUP_HIGMULTIFile file<NTUP>

outputNTUP_LARNOISEFile file<NTUP> outputNTUP_MCPScaleFile file<NTUP> outputNTUP_MCPPTFile file<NTUP>

outputNTUP_MUONCALIBFile file<NTUP> outputNTUP_PROMPTPHOTFile file<NTUP> outputNTUP_SCTFile file<NTUP>

outputNTUP_SUSYTRUTHFile file<NTUP> outputNTUP_TRKVALIDFile file<NTUP> outputNTUP_TRTFile file<NTUP>

The AMI-Tags application

A control embedded in a wiki and connected to the central AMI service

This control executes the **GetDatasetInfo** command

Conclusion

Conclusion

- AMI is mature metadata ecosystem of more than 18 years of existence
- Originally developed for the ATLAS experiment:
 - i) Official dataset discovery tool (millions of datasets, billions of files). ii) Used by the ATLAS production system (parameter definition for dataset processing [= AMI-Tags]). ...
- AMI version 2 released this year, web site soon available
- AMI Java Core
 - High level server-side JAVA library for processing metadata
 - i) High level primitives for manipulating metadata. ii) Metadata Query Language. iii) datasource connectivity.
- AMI HTTP Services + lightweight clients
 - AMI commend service (proprietary), REST interface
- AMI Task Server
 - Distributed system for extracting, processing and storing metadata
- AMI Web Framework
 - For developing metadata-oriented Web applications and graphic controls

Questions?