Collecting Conditions usage metadata to optimize current and future ATLAS software and processing

D Barberis, A Formica, E Gallas, S Oda, L Rinaldi, G Rybkin, M Verducci

on behalf of ATLAS Collaboration

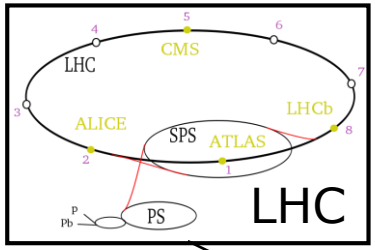
Motivation and Outline

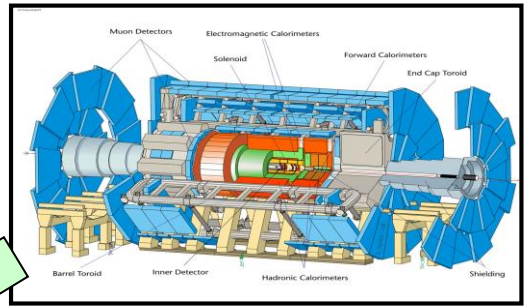
GOAL: improve understanding of Conditions data usage by event-wise processing

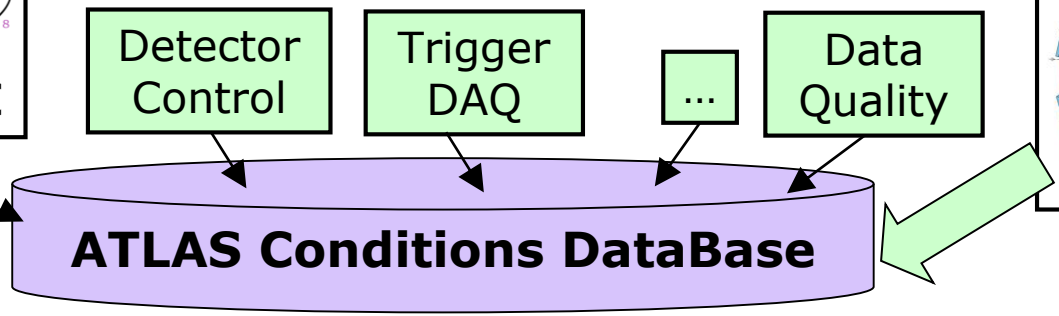
→ This helps us to develop improved tools and resources for future processing

- Introducing the ATLAS Conditions DataBase
- The need of gathering metadata information
- The ATLAS DB Release use case
  - Collecting Conditions DB metrics
  - Collecting Conditions DB access patterns by specific jobs
  - Realization of a custom-lite DB Release

ATLAS Conditions data







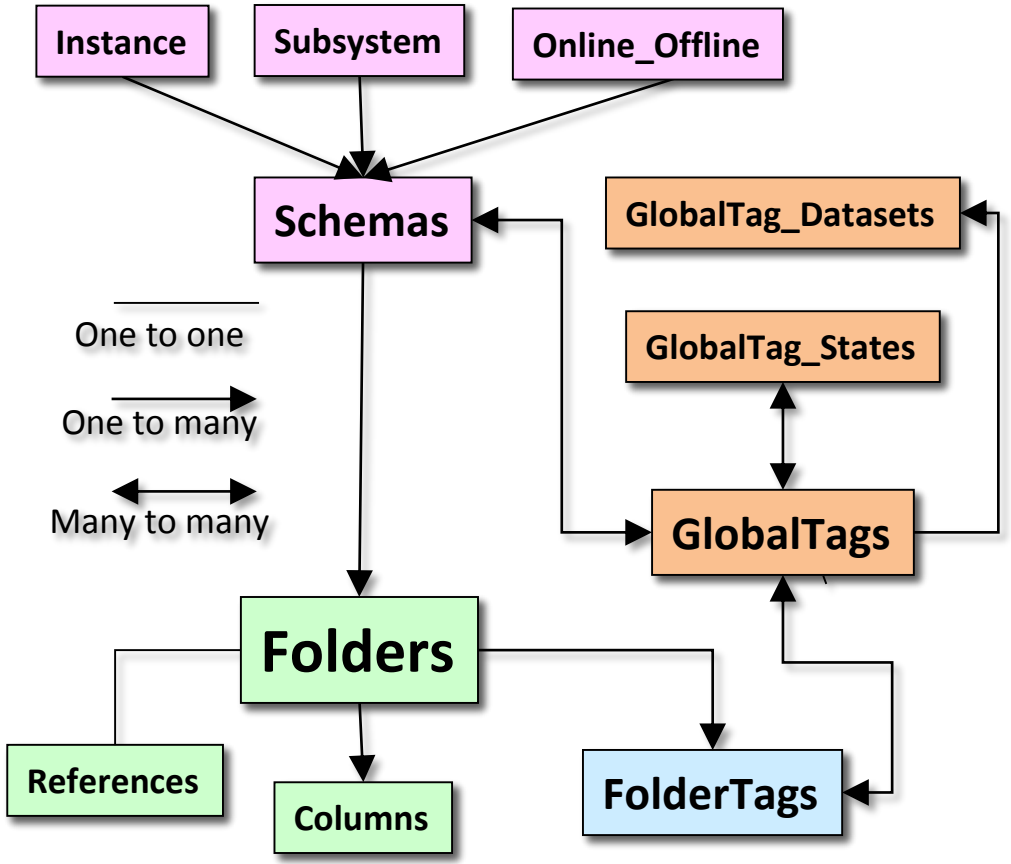
'*Conditions*' data encompass a wide variety of information

- characterize the state of all ATLAS subsystems during specific intervals
- are not generally stored event-wise but during an interval of validity
- essential for data taking and/or event processing
- are grouped by logical or physical subsystem

DataBase Design

Driven primarily by the LCG-COOL DB structure

- Folder centric: Folders represent Conditions DB tables
  - –Each Folder is owned by a specific Schema
    - •Each has subsystem, instance, and if used offline or strictly online
- Multi-version Folders have one/more FolderTags
  - –For Conditions that allow different versions over time intervals
- FolderTags may be included in one/more GlobalTags
  - –When designated to be used in event-wise processing



Conditions DB present limitations

Conditions metadata and usage information are spread over a large number of "sources"

- COOL DB : 17K tables distributed in ~30 schemas in Oracle
- GlobalTags, used to identify a set of conditions for every (sub)system, not exhaustive because information regarding single-version folder usage is not associated to any global tag
- Conditions may appear to be needed but then never really used by software

We have developed two new ways to gather metadata information from CondDB

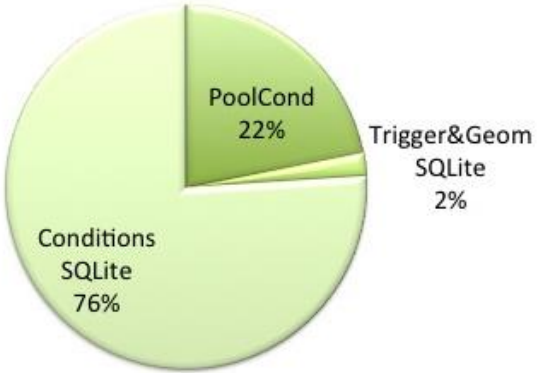
- **CondDB metrics**: extract more information at DB level which are today not stored (added new tables containing number of IOVs, payload size, ...)
- **CondDB access patterns**: analyze log files for reprocessing or MC production

Some specific tasks for investigation:

- **DB Releases** → use case discussed in this contribution
- **Overlay Event simulation** (heavy load on squid-frontier due to access to large number of Conditions)

The ATLAS DB Release Today

- ATLAS DB Releases contain:
  - file-based static SQLite replicas of the Conditions, Geometry and Trigger databases.
  - POOL ROOT payload data files for the Conditions DB (PoolCond)



- DB Releases used by the installations of ATLAS s/w when a network connection is not available on the grid node executing the job (HPC farms, offline laptops).

- DB Releases contains Conditions and Trigger data for Monte Carlo jobs only

- New DB Releases are created when a new GlobalTag is ready

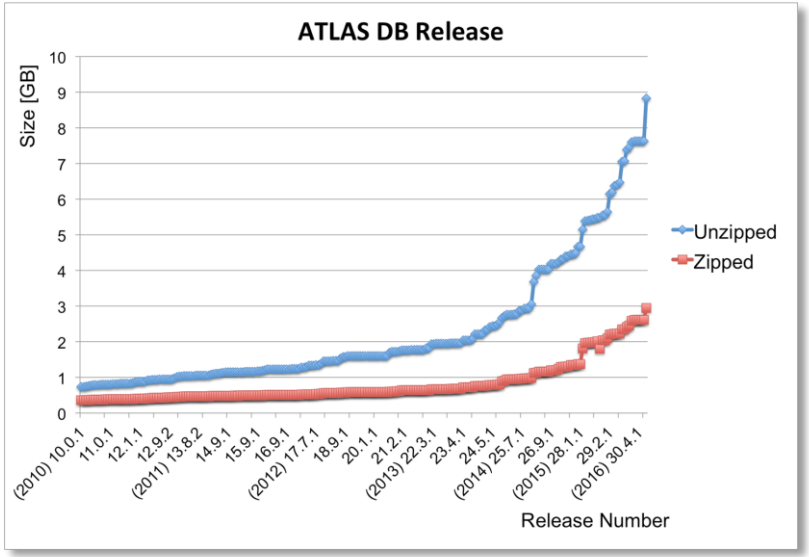
- DB Releases are available on CVMFS and DDM

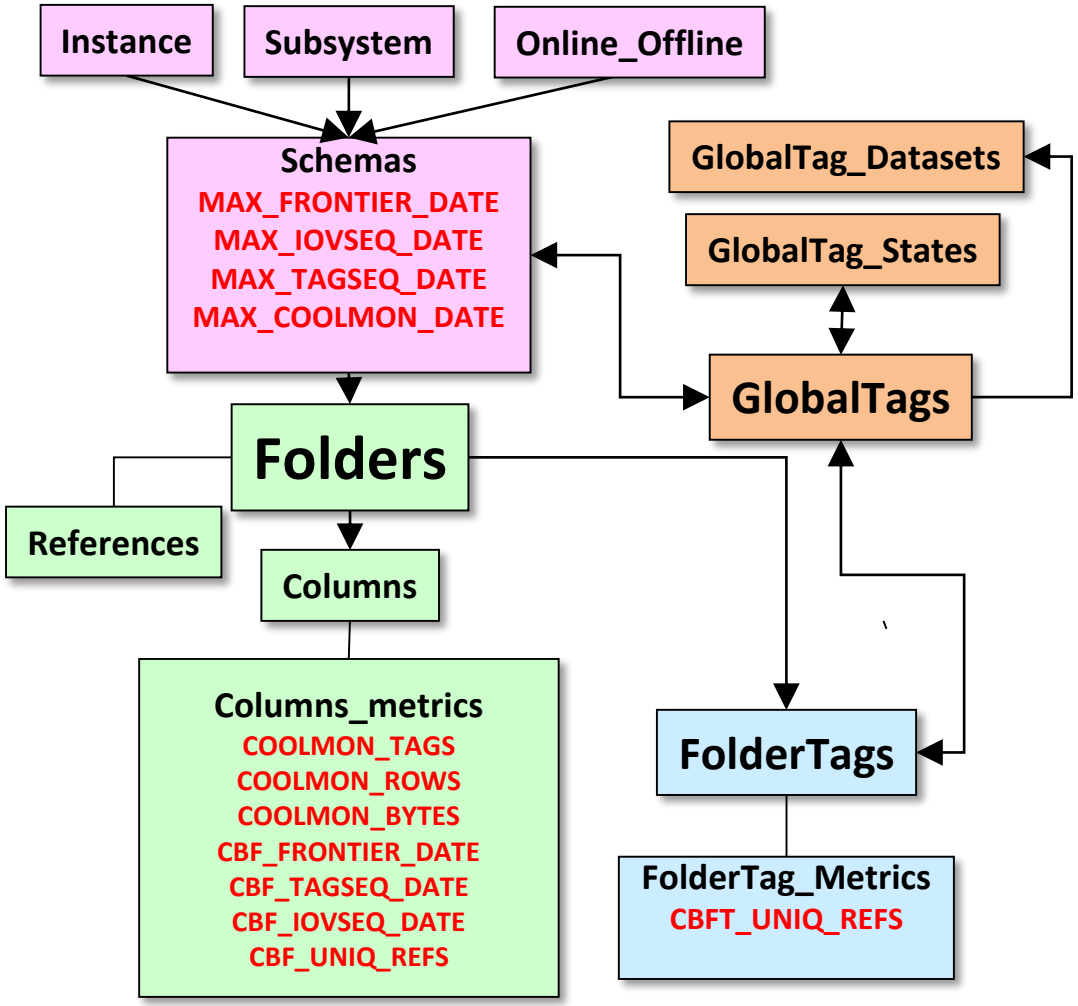
Journal of Physics: Conference Series **664** (2015) 042005

The ATLAS DB Release Today

- The present system allows to create only incremental versions of the ATLAS DB Releases
- DB Release size is steadily increasing (up to 9GB)
- Not all information stored in a DB Release is needed

GOAL: produce a custom-lite DB Release containing only information needed for dedicated production





CondDB metrics

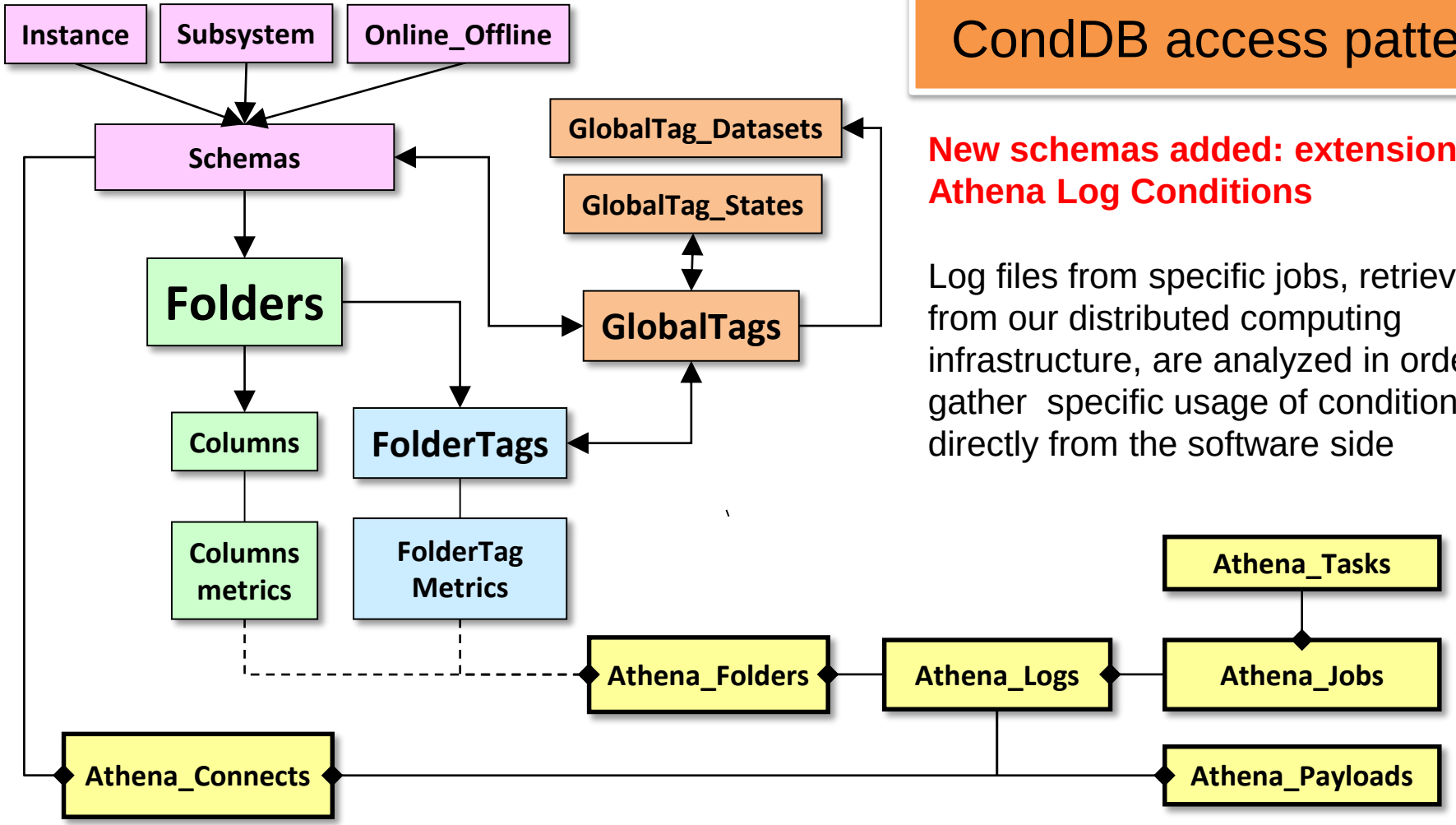
New Columns added in existing tables

- Date of latest changes
  - Helps with metadata sync
- Row counts, volumes
  - Helps to understand how much data is in each Folder and in each FolderTag

  → Useful for understanding relative scale of Athena access
- Count unique external references
  - Helps to know the number of POOL files are associated with
    - Each Folder
    - Each FolderTag

    → how many POOL files to in a customized DB Release

CondDB access pattern



New schemas added: extension to Athena Log Conditions

Log files from specific jobs, retrieved from our distributed computing infrastructure, are analyzed in order to gather specific usage of conditions directly from the software side

A custom-lite ATLAS DB Release

Several tests performed by running typical simulation transformations, accessing the default configuration

A postExec command has been used to dump all the Folders/Tags loaded during the job, by analyzing the Athena log fIles

Result: we observed that only a fraction of Folder/Tag relations were used (ranging from 10% to 20%)

This allowed to reduce the SQLite file size to a 10% - 20% of the original size

Tests with custom-lite DB Releases were successful

Ongoing analysis:

- Add information gathered by the CondDB metrics
  - Reduce the number of PoolCond files
- Set up an automatic procedure

Summary and outlook

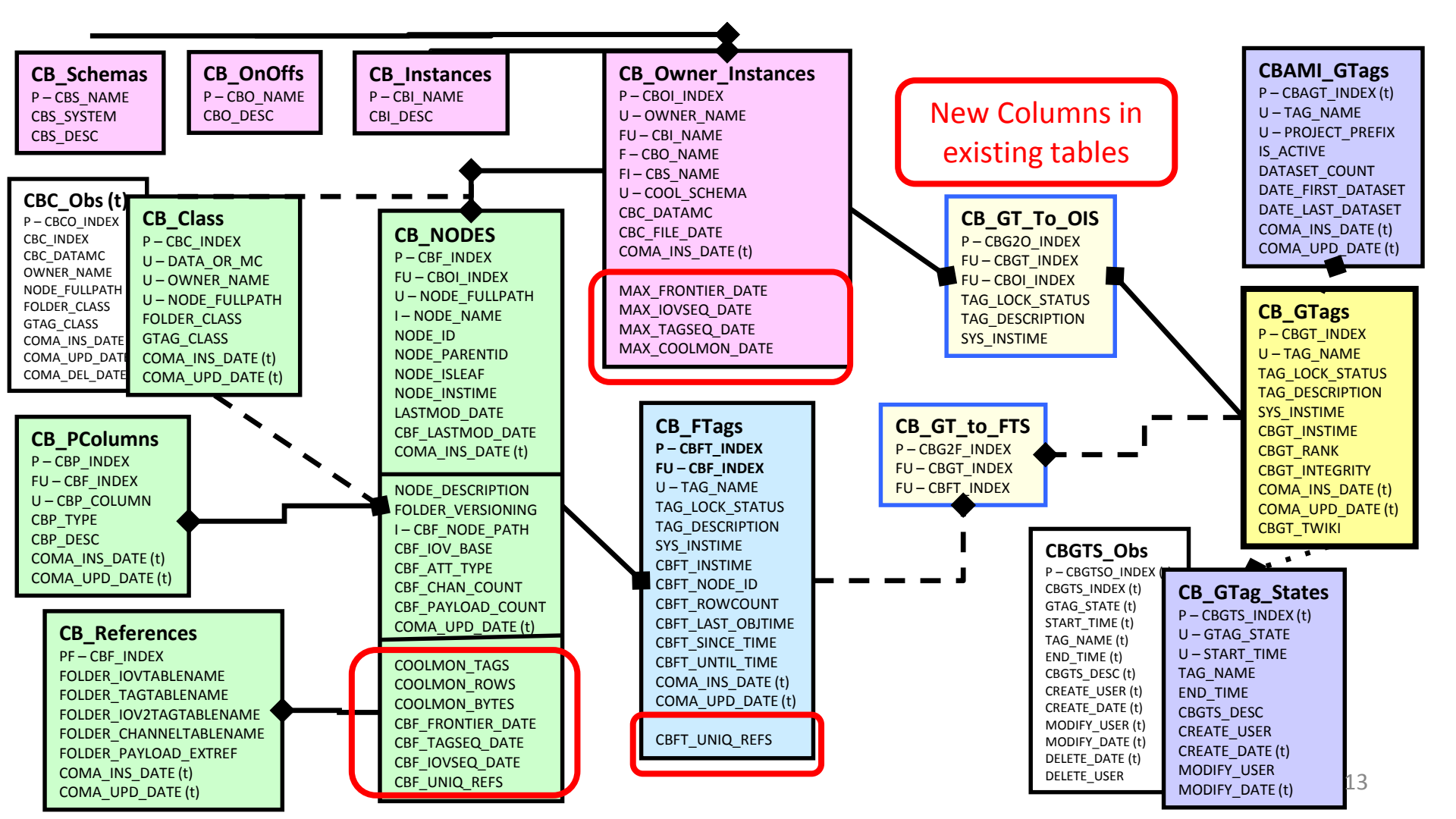
Information contained in the ATLAS Conditions DataBase is spread over a large number of sources

- Collecting such metadata is challenging in some very specific tasks
- The DB Release use case has been studied
- Custom-lite DB Releases have been successfully created by collecting metadata information with new tools based on the analysis of the CondDB access pattern
  - Results may be improved by using CondDB metrics

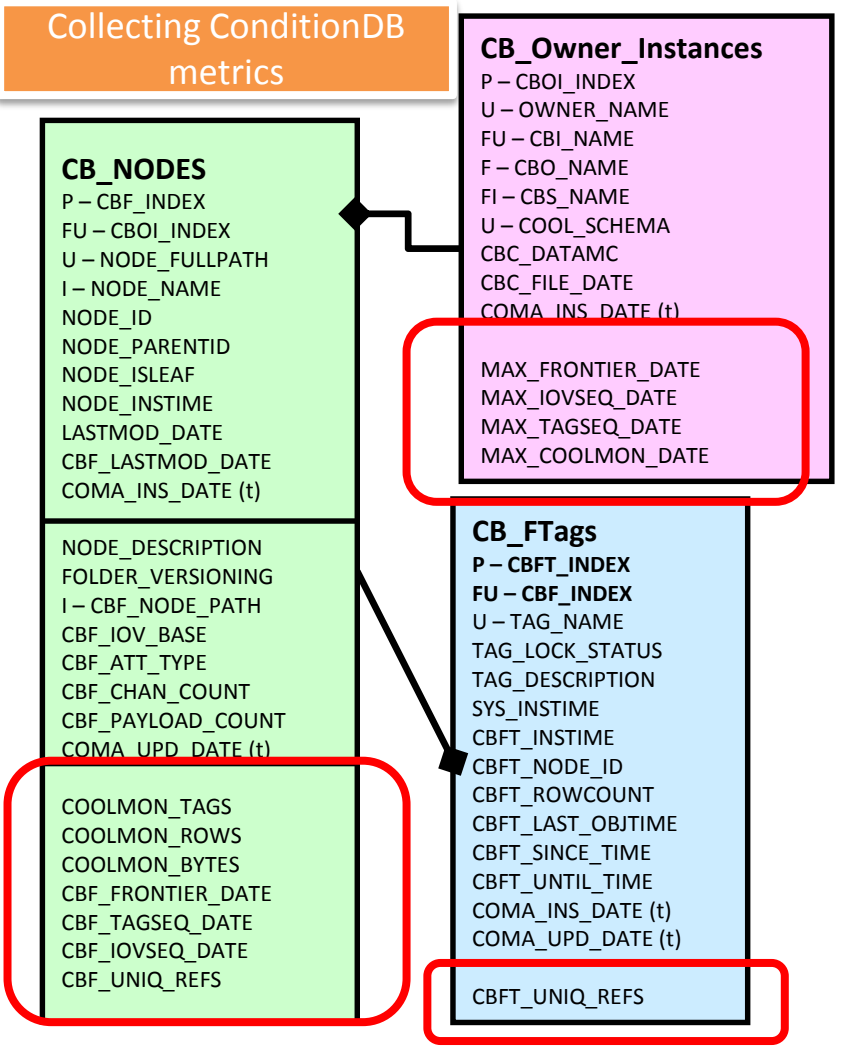
Future plans

- improve the procedures for metadata collection
- set up an automated procedure for DB Release slimming
- Investigate more use cases:
  - application on the Overlay Event simulation

Backup


CB_Schemas
P – CBS_NAME
CBS_SYSTEM
CBS_DESC
CB_OnOffs
P – CBO_NAME
CBO_DESC
CB_Instances
P – CBI_NAME
CBI_DESC
CB_Owner_Instances
P – CBOI_INDEX
U – OWNER_NAME
FU – CBI_NAME
F – CBO_NAME
FI – CBS_NAME
U – COOL_SCHEMA
CBC_DATAMC
CBC_FILE_DATE
COMA_INS_DATE (t)
MAX_FRONTIER_DATE
MAX_IOVSEQ_DATE
MAX_TAGSEQ_DATE
MAX_COOLMON_DATE
New Columns in existing tables
CBAMI_GTags
P – CBAGT_INDEX (t)
U – TAG_NAME
U – PROJECT_PREFIX
IS_ACTIVE
DATASET_COUNT
DATE_FIRST_DATASET
DATE_LAST_DATASET
COMA_INS_DATE (t)
COMA_UPD_DATE (t)
CBC_Obs (t)
P – CBCO_INDEX
CBC_INDEX
CBC_DATAMC
OWNER_NAME
NODE_FULLPATH
FOLDER_CLASS
GTAG_CLASS
COMA_INS_DATE
COMA_UPD_DATE
COMA_DEL_DATE
CB_Class
P – CBC_INDEX
U – DATA_OR_MC
U – OWNER_NAME
U – NODE_FULLPATH
FOLDER_CLASS
GTAG_CLASS
COMA_INS_DATE (t)
COMA_UPD_DATE (t)
CB_NODES
P – CBF_INDEX
FU – CBOI_INDEX
U – NODE_FULLPATH
I – NODE_NAME
NODE_ID
NODE_PARENTID
NODE_ISLEAF
NODE_INSTIME
LASTMOD_DATE
CBF_LASTMOD_DATE
COMA_INS_DATE (t)
NODE_DESCRIPTION
FOLDER_VERSIONING
I – CBF_NODE_PATH
CBF_IOV_BASE
CBF_ATT_TYPE
CBF_CHAN_COUNT
CBF_PAYLOAD_COUNT
COMA_UPD_DATE (t)
COOLMON_TAGS
COOLMON_ROWS
COOLMON_BYTES
CBF_FRONTIER_DATE
CBF_TAGSEQ_DATE
CBF_IOVSEQ_DATE
CBF_UNIQ_REFS
CB_GT_To_OIS
P – CBG2O_INDEX
FU – CBGT_INDEX
FU – CBOI_INDEX
TAG_LOCK_STATUS
TAG_DESCRIPTION
SYS_INSTIME
CB_GTags
P – CBGT_INDEX
U – TAG_NAME
TAG_LOCK_STATUS
TAG_DESCRIPTION
SYS_INSTIME
CBGT_INSTIME
CBGT_RANK
CBGT_INTEGRITY
COMA_INS_DATE (t)
COMA_UPD_DATE (t)
CBGT_TWIKI
CB_PColumns
P – CBP_INDEX
FU – CBF_INDEX
U – CBP_COLUMN
CBP_TYPE
CBP_DESC
COMA_INS_DATE (t)
COMA_UPD_DATE (t)
CB_FTags
P – CBFT_INDEX
FU – CBF_INDEX
U – TAG_NAME
TAG_LOCK_STATUS
TAG_DESCRIPTION
SYS_INSTIME
CBFT_INSTIME
CBFT_NODE_ID
CBFT_ROWCOUNT
CBFT_LAST_OBJTIME
CBFT_SINCE_TIME
CBFT_UNTIL_TIME
COMA_INS_DATE (t)
COMA_UPD_DATE (t)
CBFT_UNIQ_REFS
CB_GT_to_FTS
P – CBG2F_INDEX
FU – CBGT_INDEX
FU – CBFT_INDEX
CBGTS_Obs
P – CBGTSO_INDEX (t)
CBGTS_INDEX (t)
GTAG_STATE (t)
START_TIME (t)
TAG_NAME (t)
END_TIME (t)
CBGTS_DESC (t)
CREATE_USER (t)
CREATE_DATE (t)
MODIFY_USER (t)
MODIFY_DATE (t)
DELETE_DATE (t)
DELETE_USER
CB_GTag_States
P – CBGTS_INDEX (t)
U – GTAG_STATE
U – START_TIME
TAG_NAME
END_TIME
CBGTS_DESC
CREATE_USER
CREATE_DATE (t)
MODIFY_USER
MODIFY_DATE (t)
CB_References
PF – CBF_INDEX
FOLDER_IOVTABLENAME
FOLDER_TAGTABLENAME
FOLDER_IOV2TAGTABLENAME
FOLDER_CHANNELTABLENAME
FOLDER_PAYLOAD_EXTREF
COMA_INS_DATE (t)
COMA_UPD_DATE (t)

Collecting ConditionDB metrics



New Columns added in existing tables

- CB_Owner_Instances
  - These are the schemas
- CB_Nodes
  - These are the folders
    - Branches (nodes) also here
- CB_Ftags
  - These are folder tags

- New data is from
  - Frontier cache consistency dates
    - Date of latest change to the folder
  - COOL Oracle Sequence tables
    - Date of latest IOV or TAG insert
  - COOL MONitoring tables
    - Count of rows, size, tags, chan
      - With associated dates
      - Updated just once per day

Collecting ConditionDB metrics

- Frontier cache consistency dates
  - Date of latest change to the folder
    - → BEST source of when the last data was entered
    - → Use this to speed up loading of folder and global tags
- COOL Oracle Sequence tables
  - Date of latest IOV or TAG insert
    - → only changes when a new row is added
- COOL MONitoring tables
  - Count of rows, size, tags, chan
    - → With associated dates
    - → Updated just once per day
  - cross checking info: these are derived from Oracle dictionary tables filled by Oracle scheduler jobs !
    - → Should I just use the direct source instead (?)
  - found COOLMON tables count IOVs but not PAYLOADS
  - Accounting for data volume in dblister is confusing (outdated?)
    - → Since it is missing PAYLOAD tables

CondDB access pattern