Conditions data payload management with Rucio

Paul Laycock (University of Geneva)

UNIVERSITÉ DE GENÈVE
FACULTÉ DES SCIENCES
Département d'astronomie

7th Rucio Community Workshop
4th October 2024

GWSC GRAVITATIONAL WAVE SCIENCE CENTER

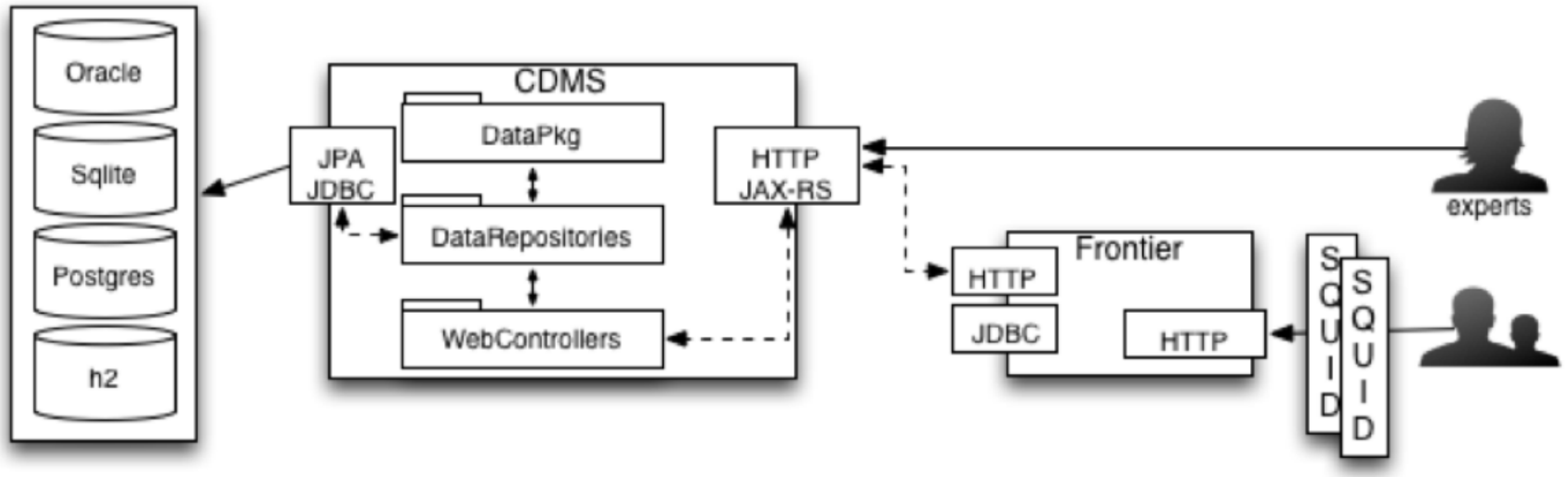
References

- https://arxiv.org/pdf/2401.16274 - The HSF CDB Reference Implementation
  - And CHEP talk https://indico.jlab.org/event/459/contributions/11326/attachments/9582/14015/conditions_db_reference_implementation_chep_2023.pdf
- https://iopscience.iop.org/article/10.1088/1742-6596/331/4/042008/pdf - Frontier
- https://arxiv.org/pdf/1901.05429 - The HSF Conditions Data white paper
- https://doi.org/10.1051/epjconf/202429501013 - Towards a new conditions data infrastructure in ATLAS

- Figures taken, context hopefully (!) retained

Conditions data

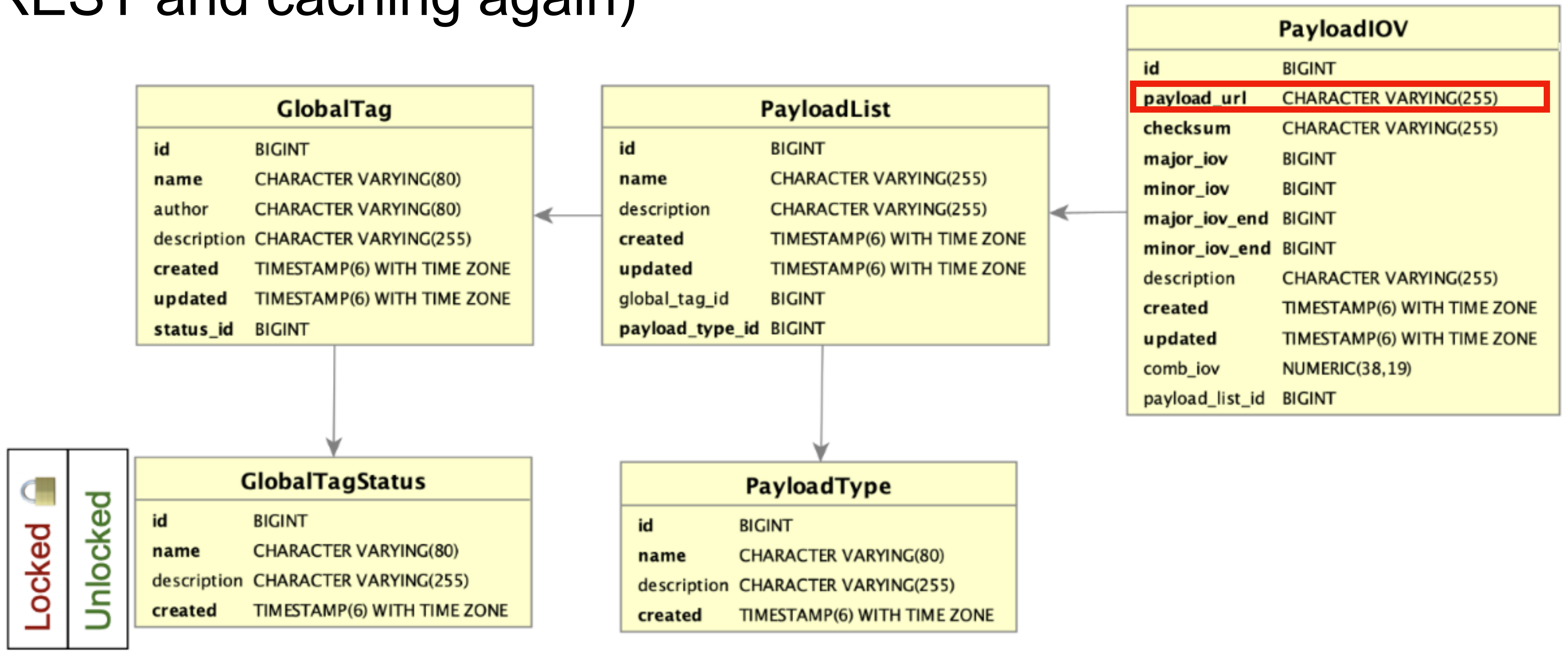
- Conditions data is… Everything apart from the “event” data that needs to be pulled into a data processing job, e.g. detector calibrations, slow control information (usually at coarse granularity)
    - Many jobs need the same conditions data
- Typically lives in a database and often still does



HSF CDB design

- Commonality between the experiment use cases led to a common understanding of best practice
  - Factorise metadata queries from payload queries (caching)
  - Keep it simple (caching again)
  - Loose coupling between client and servers (REST and caching again)

- HSF schema (right) - metadata lives in an RDB
  - Payloads are referenced by URL
- Payloads live… elsewhere (not in the DB)
- Implementation details, YMMV



GlobalTag

id	BIGINT
name	CHARACTER VARYING(80)
author	CHARACTER VARYING(80)
description	CHARACTER VARYING(255)
created	TIMESTAMP(6) WITH TIME ZONE
updated	TIMESTAMP(6) WITH TIME ZONE
status_id	BIGINT

PayloadList

id	BIGINT
name	CHARACTER VARYING(255)
description	CHARACTER VARYING(255)
created	TIMESTAMP(6) WITH TIME ZONE
updated	TIMESTAMP(6) WITH TIME ZONE
global_tag_id	BIGINT
payload_type_id	BIGINT

PayloadIOV

id	BIGINT
payload_url	CHARACTER VARYING(255)
checksum	CHARACTER VARYING(255)
major_iov	BIGINT
minor_iov	BIGINT
major_iov_end	BIGINT
minor_iov_end	BIGINT
description	CHARACTER VARYING(255)
created	TIMESTAMP(6) WITH TIME ZONE
updated	TIMESTAMP(6) WITH TIME ZONE
comb_iov	NUMERIC(38,19)
payload_list_id	BIGINT

GlobalTagStatus

id	BIGINT
name	CHARACTER VARYING(80)
description	CHARACTER VARYING(255)
created	TIMESTAMP(6) WITH TIME ZONE

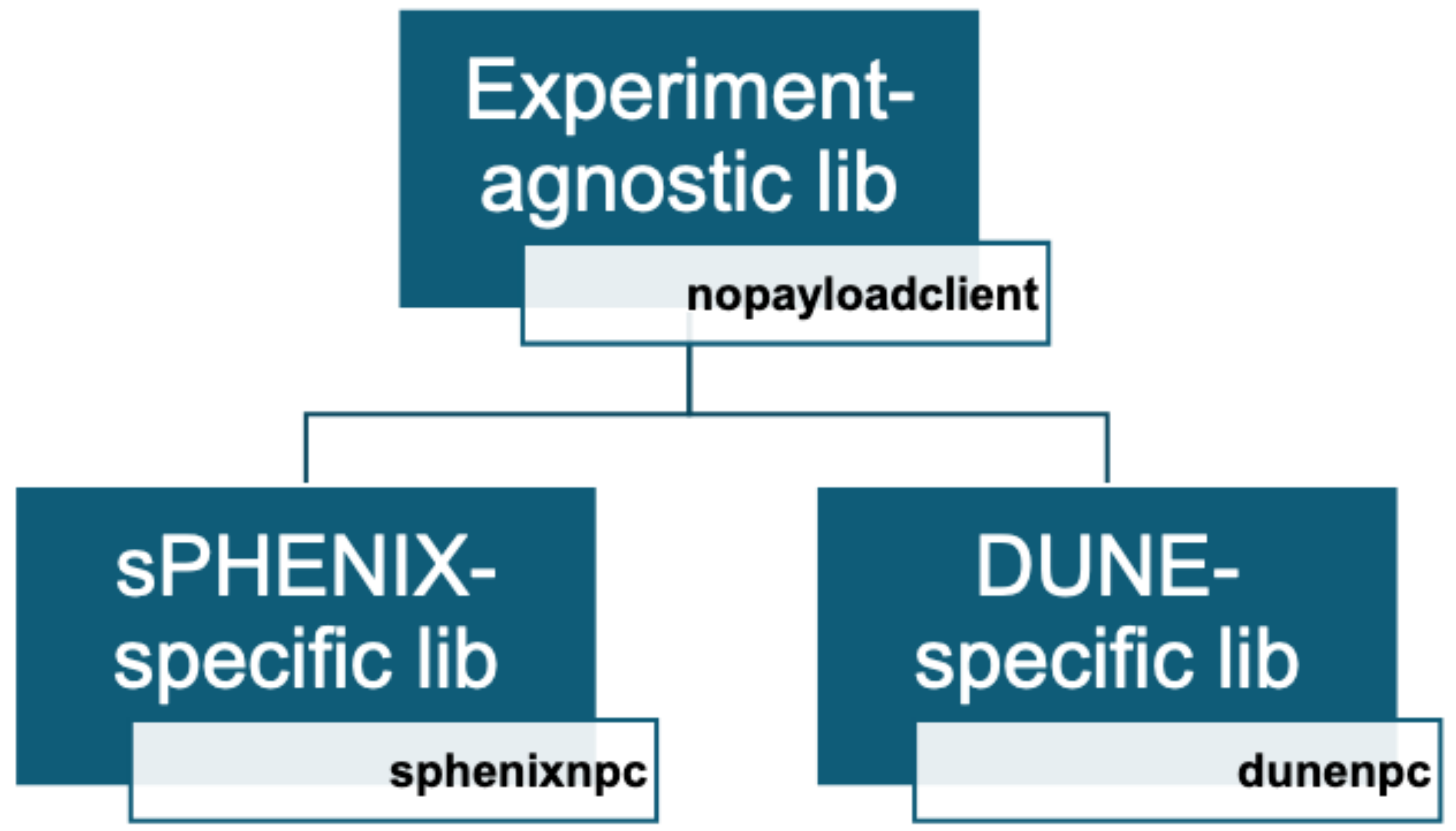
PayloadType

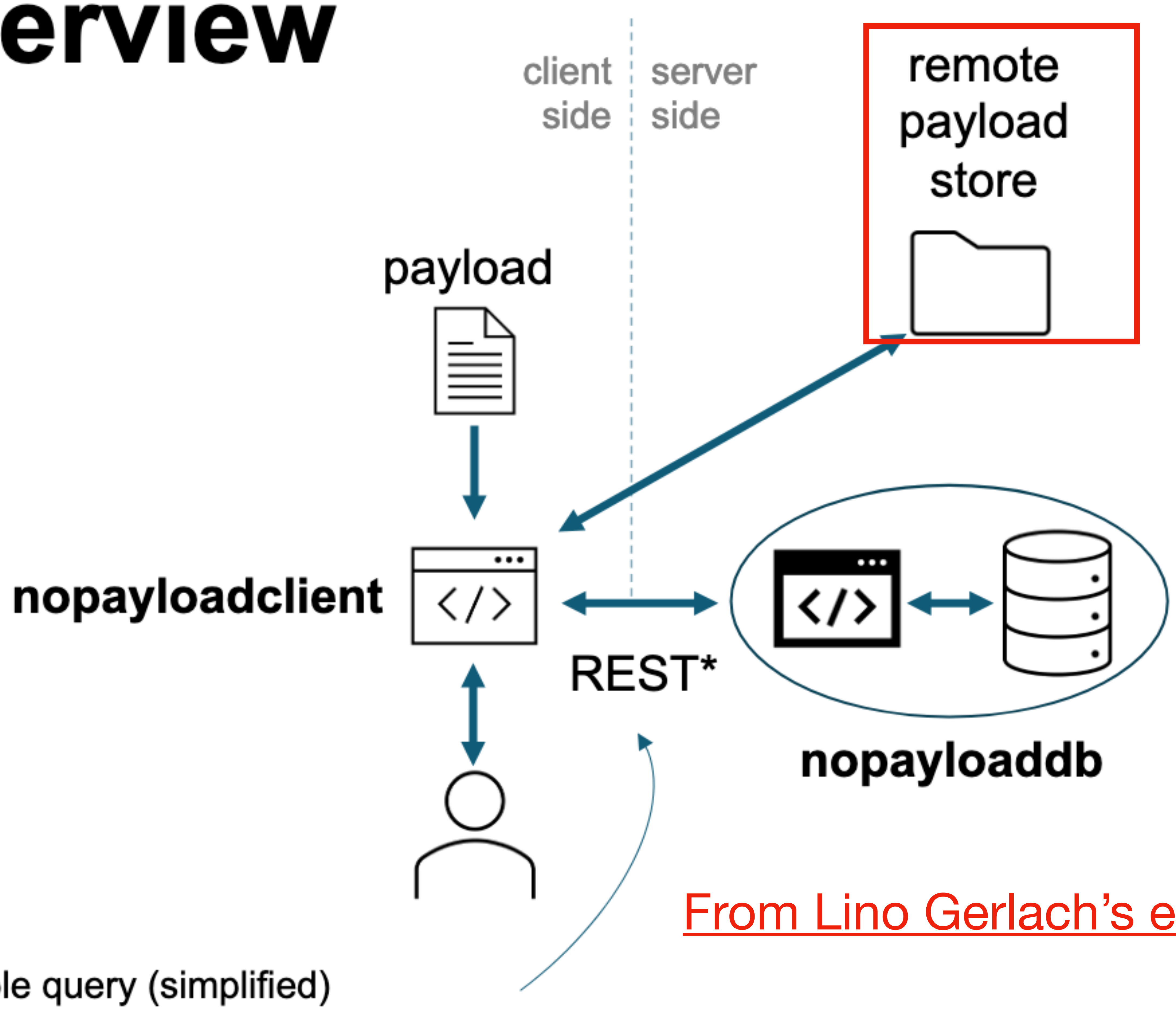
id	BIGINT
name	CHARACTER VARYING(80)
description	CHARACTER VARYING(255)
created	TIMESTAMP(6) WITH TIME ZONE

Implementation – Overview

nopayloadclient:

- Client-side stand-alone C++ tool
- Communicates with **nopayloaddb** (server)
- Local caching
- Handling of payloads





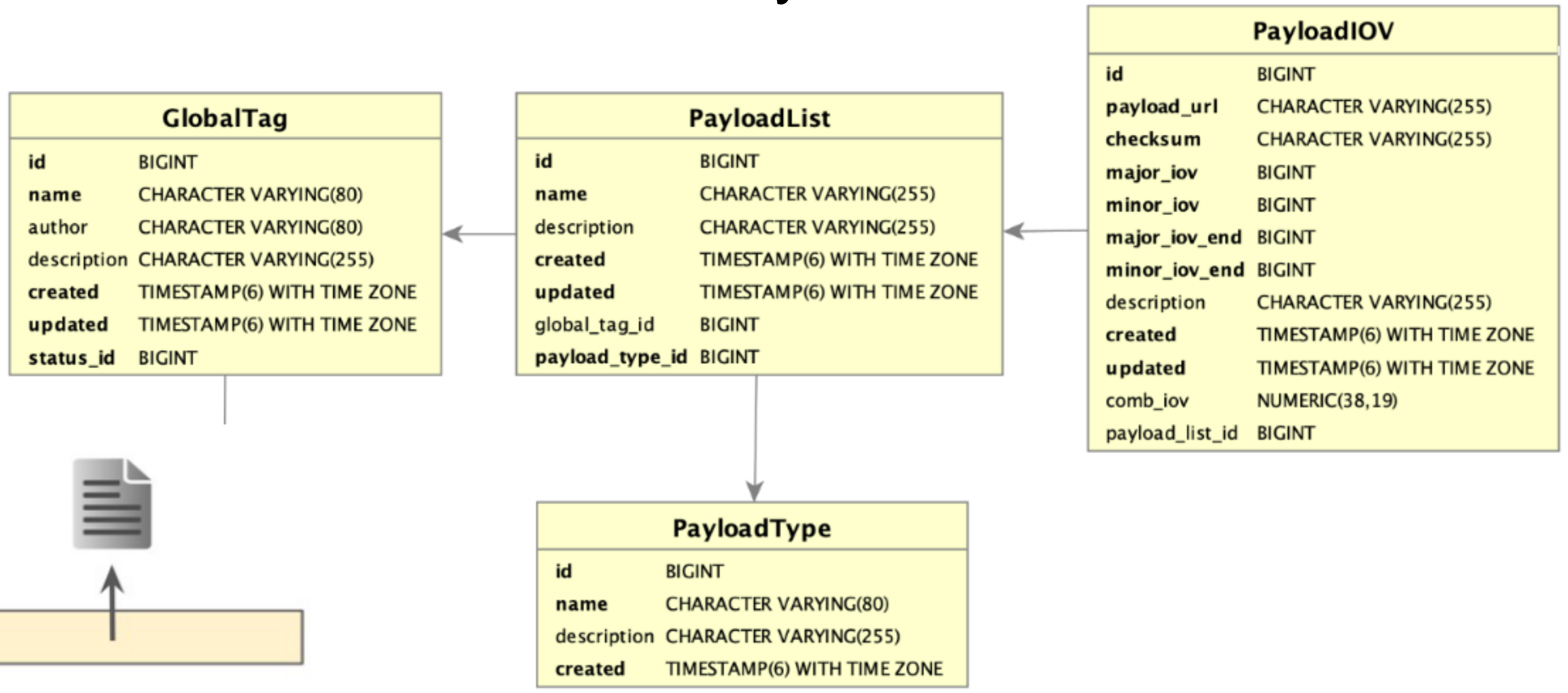
From Lino Gerlach's excellent CHEP talk

*Example query (simplified)

```
curl http://<host>/api/payloadiovs/?gtName=test_gt&iovNum=42
-> {type_1: url_1, type_2: url_2, …}
```

HSF CDB organisation

- At the job configuration level, a job needs to know which GlobalTag (global version) this job needs
- For every conditions payload type, data are organised in lists of “intervals of validity”



GlobalTag

id	BIGINT
name	CHARACTER VARYING(80)
author	CHARACTER VARYING(80)
description	CHARACTER VARYING(255)
created	TIMESTAMP(6) WITH TIME ZONE
updated	TIMESTAMP(6) WITH TIME ZONE
status_id	BIGINT

PayloadList

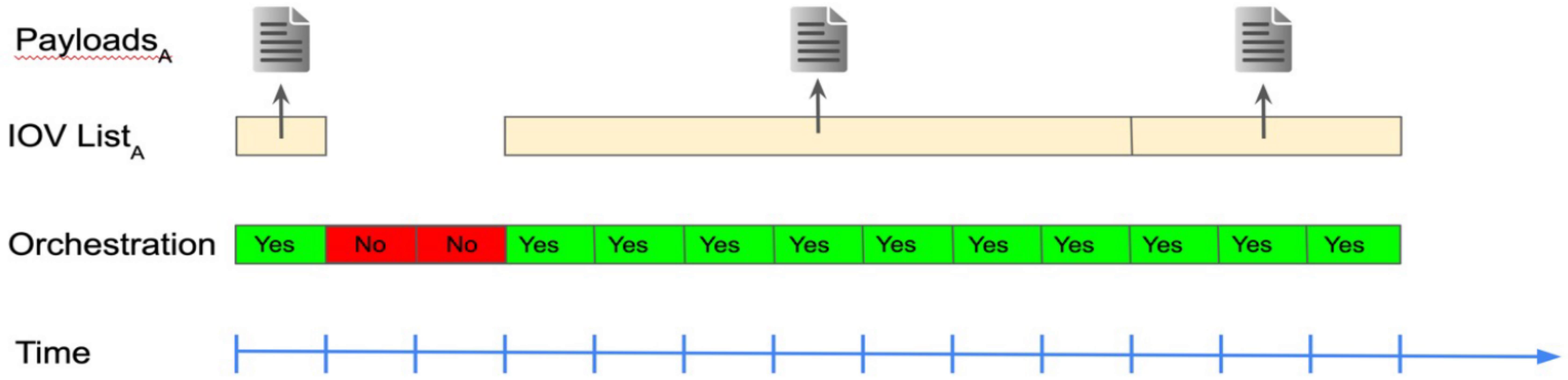
id	BIGINT
name	CHARACTER VARYING(255)
description	CHARACTER VARYING(255)
created	TIMESTAMP(6) WITH TIME ZONE
updated	TIMESTAMP(6) WITH TIME ZONE
global_tag_id	BIGINT
payload_type_id	BIGINT

PayloadIOV

id	BIGINT
payload_url	CHARACTER VARYING(255)
checksum	CHARACTER VARYING(255)
major_iov	BIGINT
minor_iov	BIGINT
major_iov_end	BIGINT
minor_iov_end	BIGINT
description	CHARACTER VARYING(255)
created	TIMESTAMP(6) WITH TIME ZONE
updated	TIMESTAMP(6) WITH TIME ZONE
comb_iov	NUMERIC(38,19)
payload_list_id	BIGINT

PayloadType

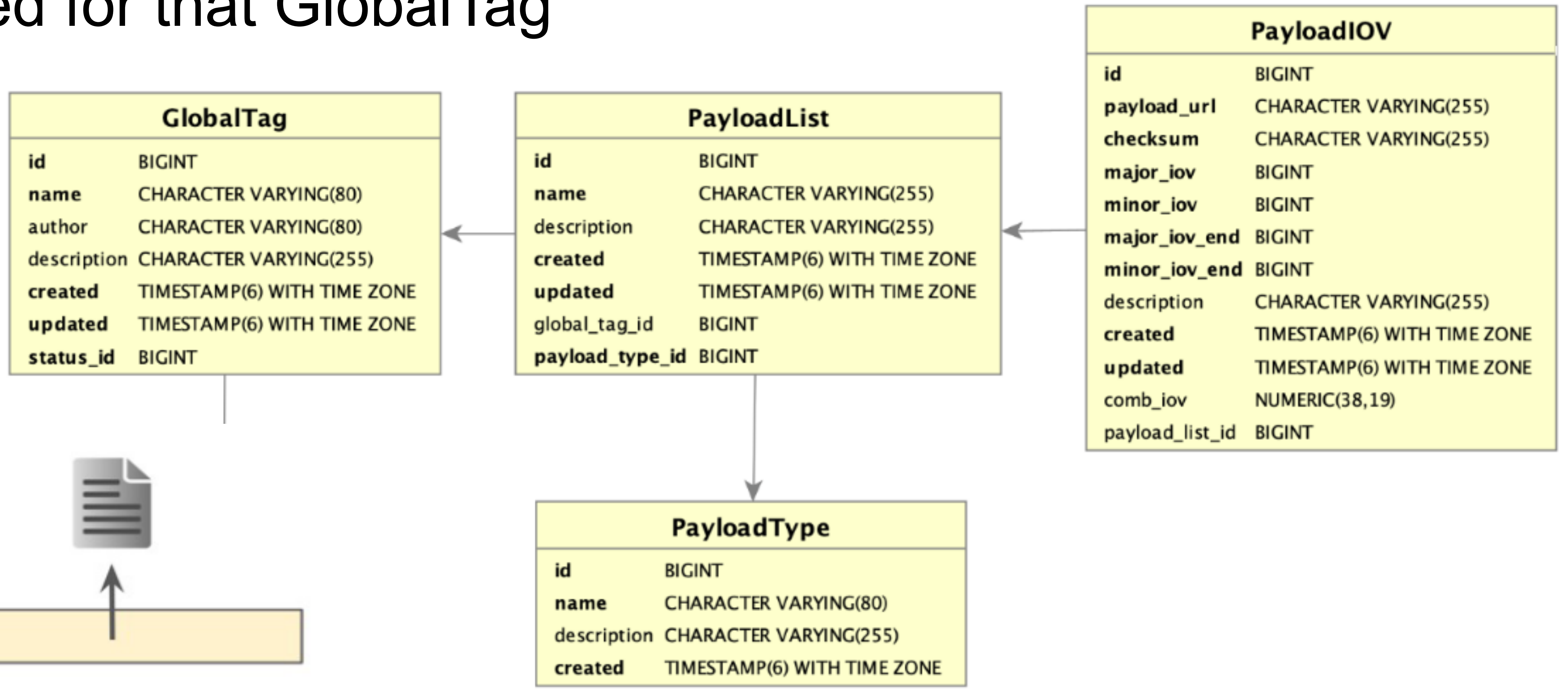
id	BIGINT
name	CHARACTER VARYING(80)
description	CHARACTER VARYING(255)
created	TIMESTAMP(6) WITH TIME ZONE



HSF CDB organisation

- In terms of data management, a GlobalTag (global version) defines all of the conditions payload files for e.g. a particular MC processing campaign: ***<u>GlobalTag -> list of files</u>***

- Could think of using Rucio to manage a dataset defined for that GlobalTag



GlobalTag

id	BIGINT
name	CHARACTER VARYING(80)
author	CHARACTER VARYING(80)
description	CHARACTER VARYING(255)
created	TIMESTAMP(6) WITH TIME ZONE
updated	TIMESTAMP(6) WITH TIME ZONE
status_id	BIGINT

PayloadList

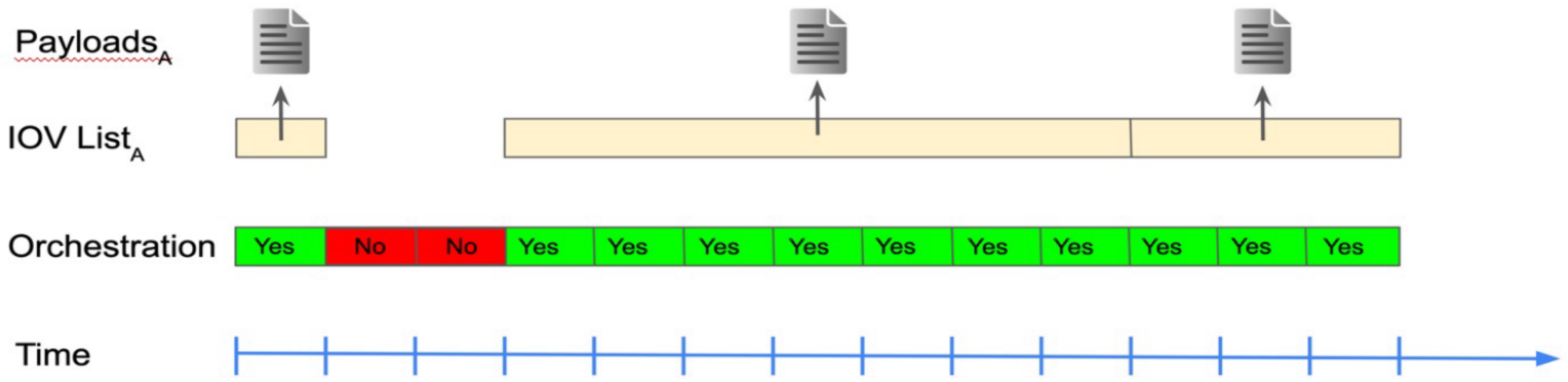
id	BIGINT
name	CHARACTER VARYING(255)
description	CHARACTER VARYING(255)
created	TIMESTAMP(6) WITH TIME ZONE
updated	TIMESTAMP(6) WITH TIME ZONE
global_tag_id	BIGINT
payload_type_id	BIGINT

PayloadIOV

id	BIGINT
payload_url	CHARACTER VARYING(255)
checksum	CHARACTER VARYING(255)
major_iov	BIGINT
minor_iov	BIGINT
major_iov_end	BIGINT
minor_iov_end	BIGINT
description	CHARACTER VARYING(255)
created	TIMESTAMP(6) WITH TIME ZONE
updated	TIMESTAMP(6) WITH TIME ZONE
comb_iov	NUMERIC(38,19)
payload_list_id	BIGINT

PayloadType

id	BIGINT
name	CHARACTER VARYING(80)
description	CHARACTER VARYING(255)
created	TIMESTAMP(6) WITH TIME ZONE



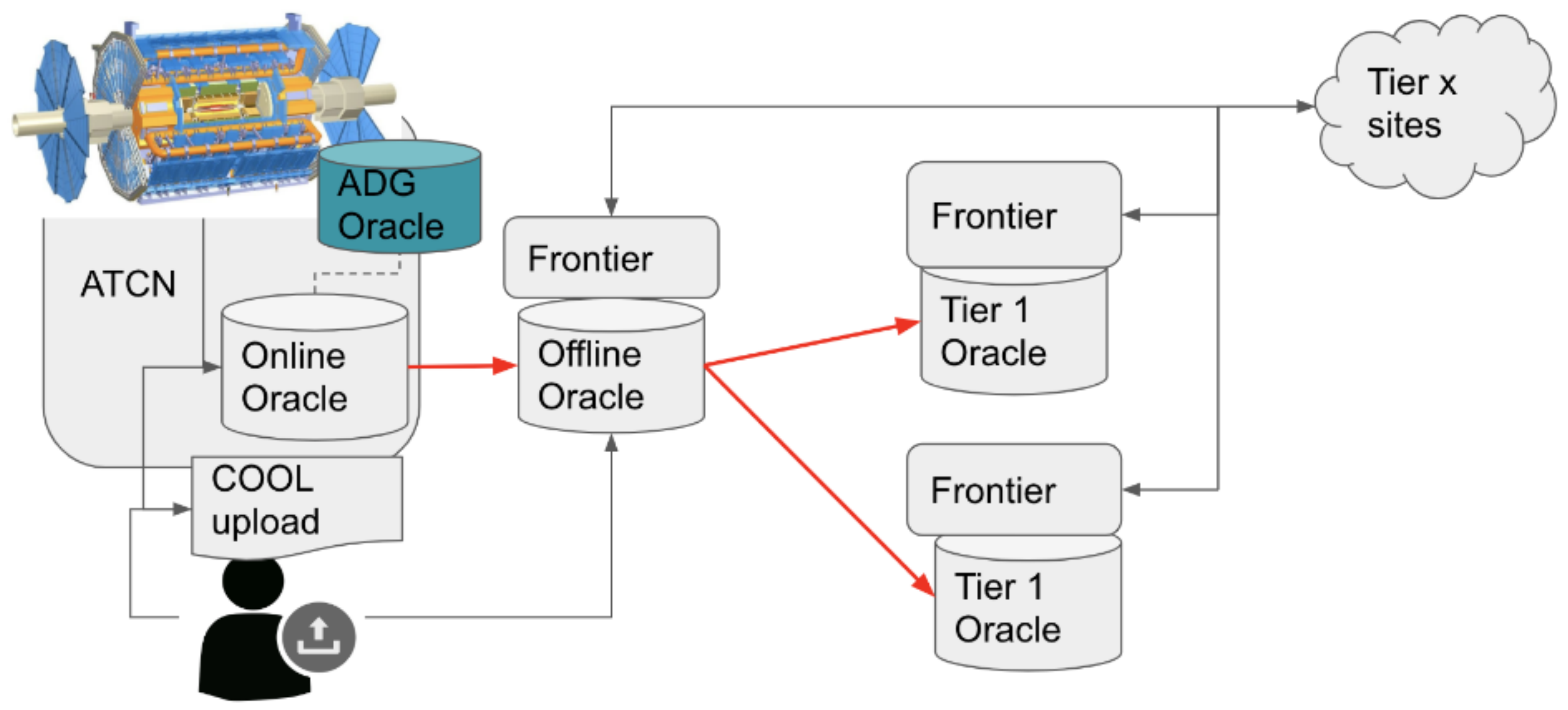
HSF CDB organisation

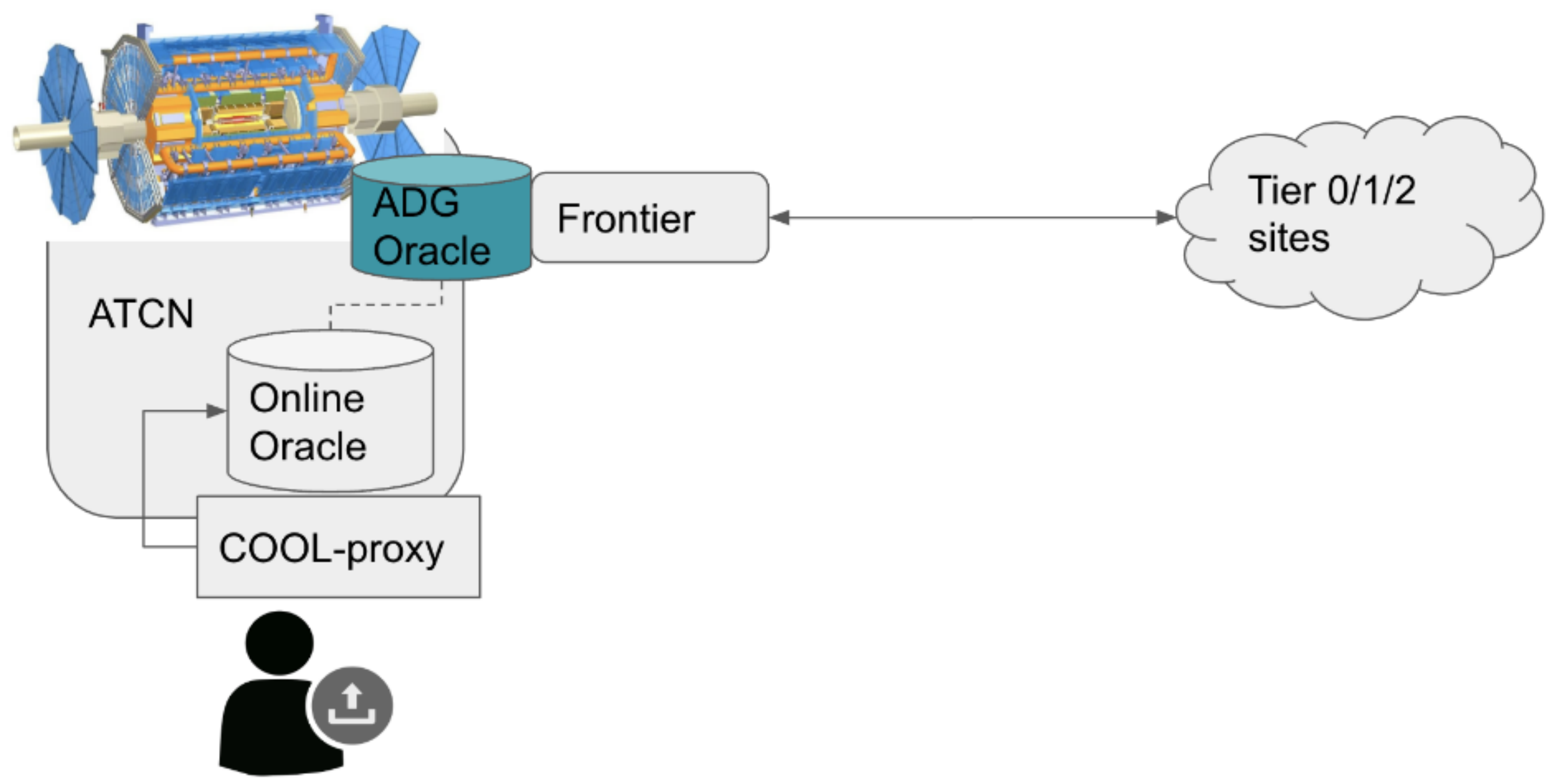
- In terms of data management, a GlobalTag (global version) defines all of the conditions payload files for e.g. a particular MC processing campaign: ***GlobalTag -> list of files***
- Could think of using Rucio to manage a dataset defined for that GlobalTag
- Used for e.g. I need to know which conditions payloads I need to send to my HPC site
  - Answer - it’s this Rucio dataset (MyConditionsGT), Rucio, please do your thing
- It could more generally be used to make replicas of conditions datasets
  - Hey Rucio, I’d like to have my conditions datasets at these sites
  - Caveat, conditions files are accessed by many jobs

HSF CDB design

- The HSF CDB design makes the payloads portable, change the storage endpoint by changing the prefix (LFN to PFN) - a configuration parameter of the HSF CDB client
- Used to determine payload service failover strategy, an example access failover pattern:
  - Check locally
  - (Check cvmfs for Belle II, works for published GTs that have been pushed there)
    - Cvmfs is a good solution for a lot of use cases, but ad hoc file storage seems like a bit of an abuse.
  - Check the main CDB service
- The server is usually protected by a cache
  - Frontier (ATLAS and CMS), or squid caches at particularly heavy sites (Belle II)
  - Many jobs hit the same files so caching is really essential

Offline ATLAS

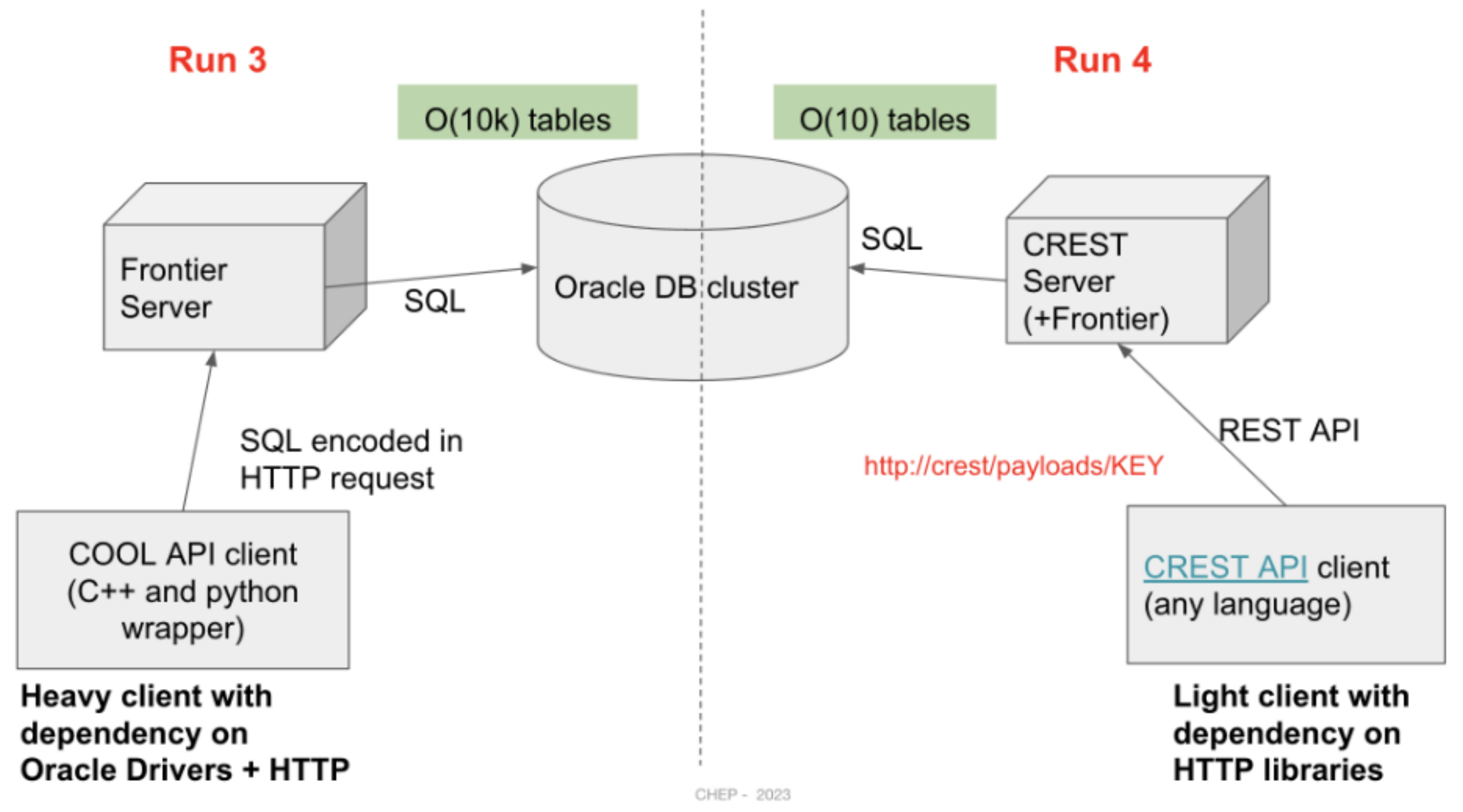




- pre-Run 3 (left) using Oracle Golden Gate, Run 3 (right) with no Golden Gate (and no Oracle outside CERN).
- In Run 3 all access from outside CERN goes via a read-only replica (ADG), protected by Frontier

- Figures from the ATLAS CHEP 2023 paper: https://doi.org/10.1051/epjconf/202429501013 - “Towards a new conditions data infrastructure in ATLAS” - mis-interpretations are mine !

Offline ATLAS


Run 3
Run 4
O(10k) tables
O(10) tables
Frontier Server
SQL
Oracle DB cluster
SQL
CREST Server (+Frontier)
SQL encoded in HTTP request
REST API
http://crest/payloads/KEY
COOL API client (C++ and python wrapper)
CREST API client (any language)
Heavy client with dependency on Oracle Drivers + HTTP
Light client with dependency on HTTP libraries
CHEP - 2023

- For Run 4, remove the dependence on COOL and replace with CREST

Offline - what if?

- Offline use case (e.g. from a software framework) looks like:
  - Query the CDB service (REST API) to ask WHICH conditions files I need
  - LFN->PFN retrieve the actual files from storage using my preferred failover strategy
- **What if the job told the CDB client where to find conditions data (known by Rucio)**
  - This is a one-off configuration at the start of a job
- **What if you could easily send conditions data to more (“all”?) of your sites (typically small data volumes)**
  - Would you ?
  - Typically have squid caches at big sites, effectively doing that at those sites (for some subset of the data)
  - These conditions files must be accessed by many jobs which is why Frontier has been so important for ATLAS and CMS
  - Latency means this is likely best used for “frozen” GlobalTags, but that’s still a lot of jobs

Online

- What it won’t do… the online use case is special and will continue to need something like the current solutions

Discussion

- At the point that conditions data payloads live on a file system, it’s intriguing to think about managing them and their access like “any other data”, and Rucio does that well !
    - The database part of CDB management is to answer the question of WHICH data do I need for this particular job, after that the retrieval of the conditions data files is just file retrieval
- At least for creating exports of conditions data for e.g. HPC processing use cases, it could be useful
- For more generally distributing replicas of conditions datasets, if this is useful, it might be “easy” (full disclosure, I have not gone further than a thought experiment !)
    - With that caveat that these files are accessed by many jobs
    - And the need for the RDB for metadata (and caching those queries) remains
- DISCUSS !