KU LEUVEN

Streamlining iRODS:
Kafka-based Data Pipelines

Peter Verraedt
Jo Wijnant

Scope

- Needs:
  - Index collections and objects in OpenSearch for global search
  - Continuously monitor project usage (w.r.t. quota)
  - Integrate metadata in file system snapshots for easy restores
- Constraints:
  - Avoid need to periodically run heavy queries on (mysql) database
  - NOT: enforcing policies
  - Critical to trigger on all possible changes
    - Listen on all peps is probably possible but can contain duplicate peps/easy to miss specific client triggered changes
    - If changes can be missed, recreates need for periodic queries

Idea

- Don't create iRODS plugin to listen on all peps
  - because goal is explicitly not enforce a policy to e.g. restrict certain actions
- Listen on the iRODS catalog = (mysql) database instead
- Use Debezium to capture row by row changes
- Debezium supports mysql/mariadb/postgres/oracle/...
- Changes are stored in Apache Kafka

Apache Kafka

https://kafka.apache.org

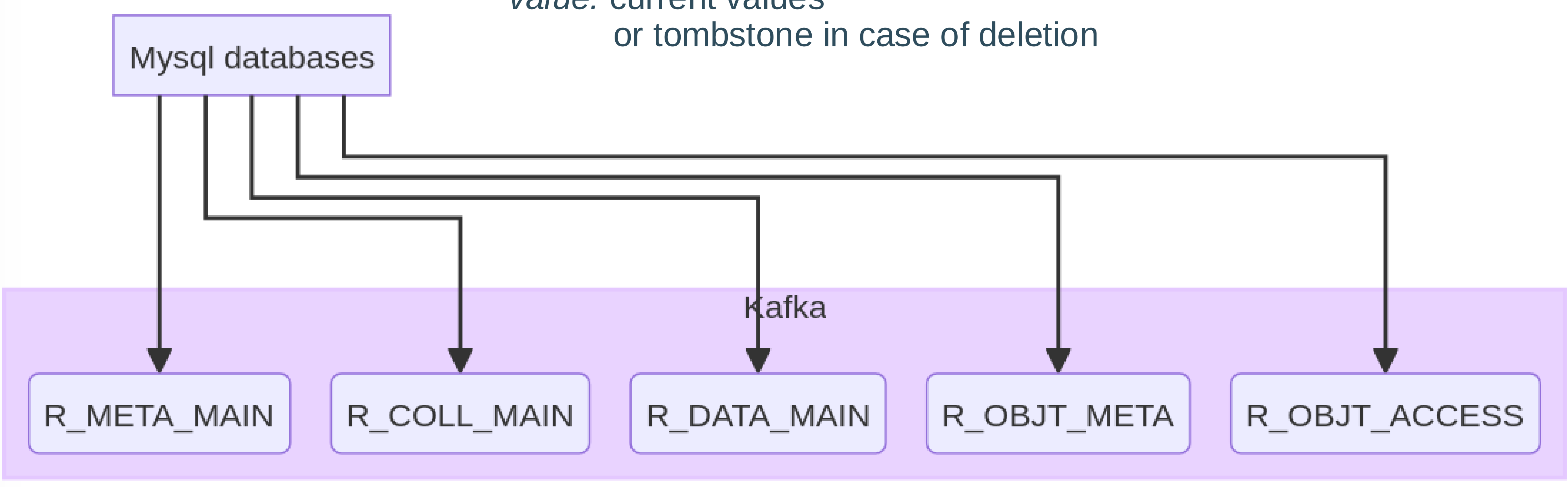
- Stores **topics** with 'messages'
- A topic is a key-value store, for each key multiple messages can be added
- A **tombstone** (null) message for a key can be stored to indicate removals
- A topic is stored in multiple **partitions**, a hash of the key is used for mapping to a partition
- One can *consume* a topic for newly appended messages = latest changes
- For today’s talk, all topics are *compacting*: from time to time, messages are cleaned up so that only the last one for each key is kept

Debezium connector

https://debezium.io

- Initial READ of tables + “slave” of mysql database to see row-by-row changes
- Output as topics:

key: database name + #object_id
value: current values
or tombstone in case of deletion



- From N databases for N zones to 5 topics containing data for all zones

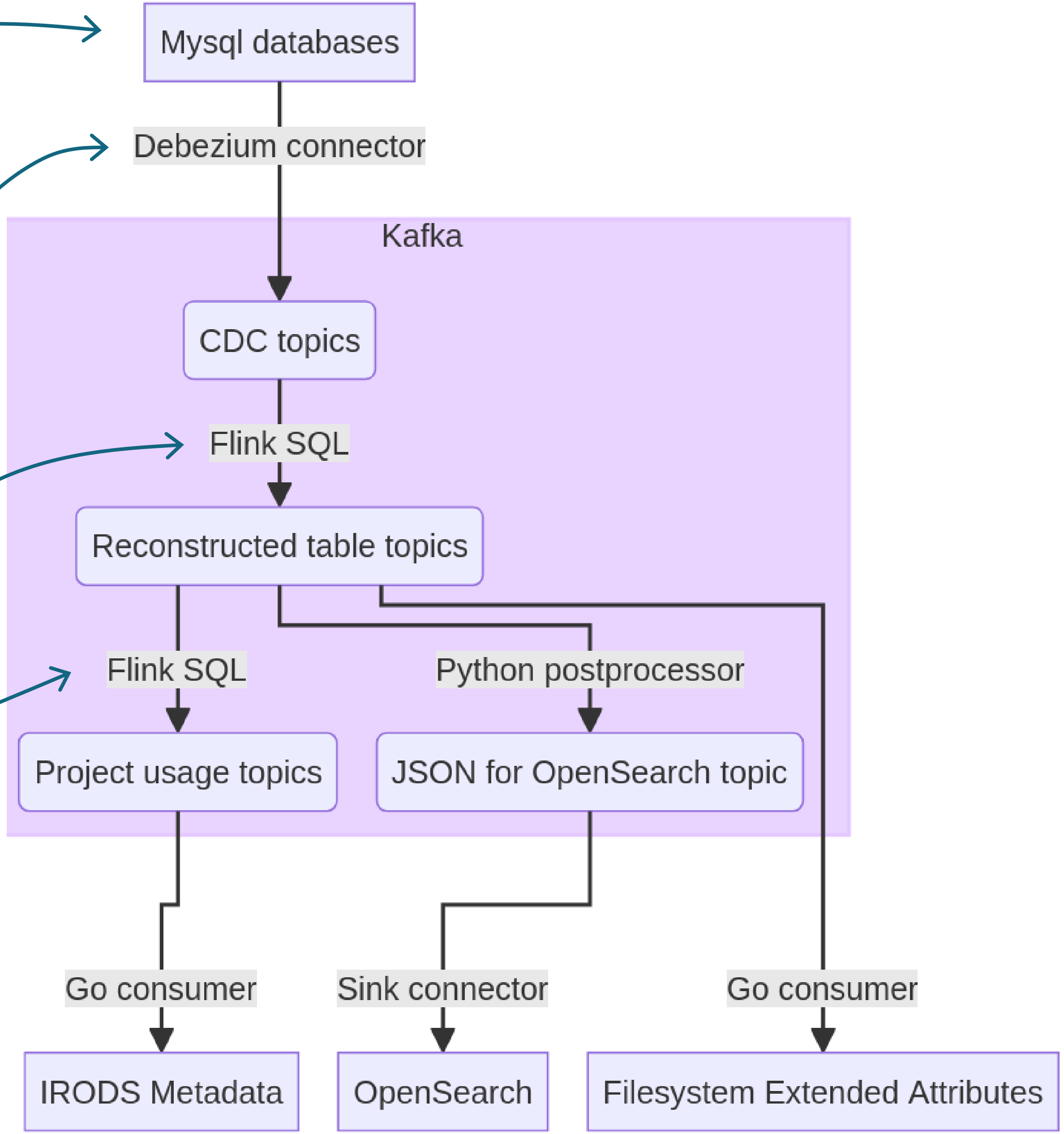
Schema

1 mysql instance
N identical structured database per irods zone

Debezium connector
Read changelog of Mysql
Output to a kafka topic per table type

Apache Flink *see next slides*

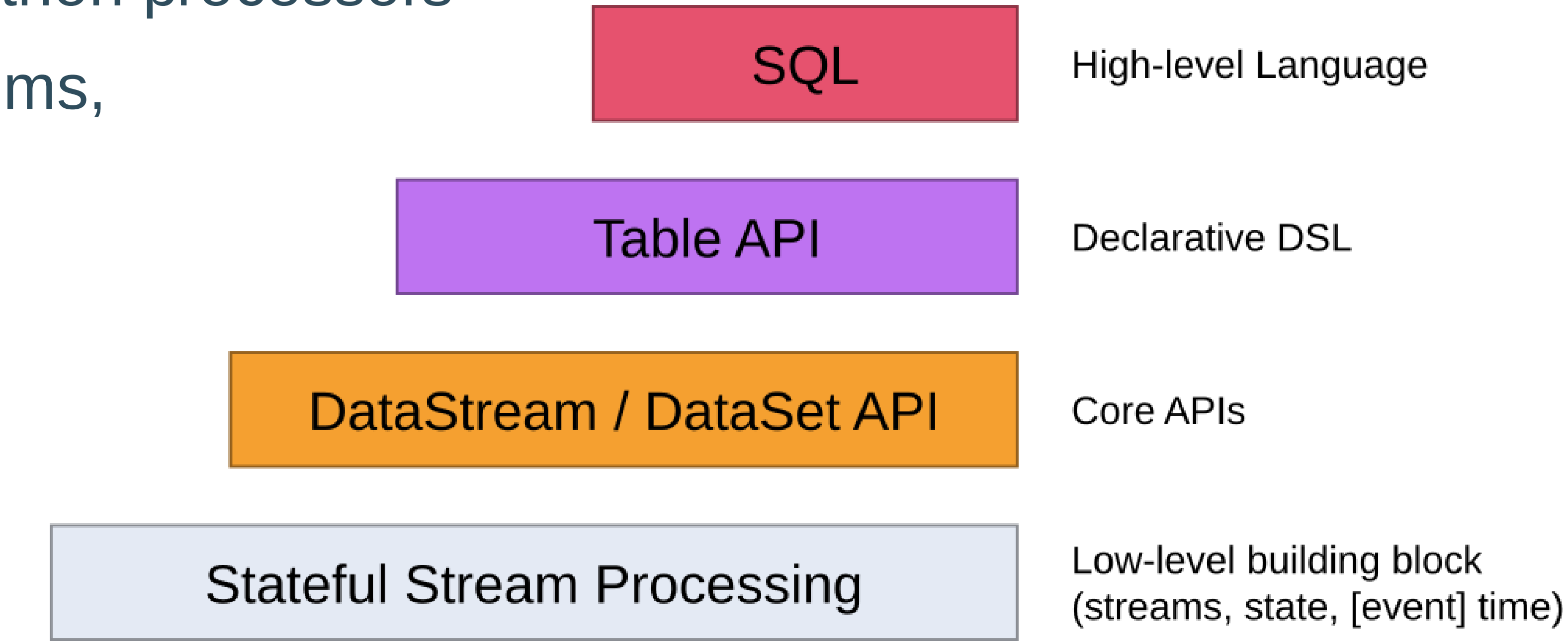
Applications



Apache Flink

https://flink.apache.org

- Data Processor
- Can take various inputs, a.o. Kafka topics
- Has SQL-like language to manipulate and combine information in topics
- Has possibility to plug in python processors
- Can output to various systems,
  a.o. Kafka, OpenSearch



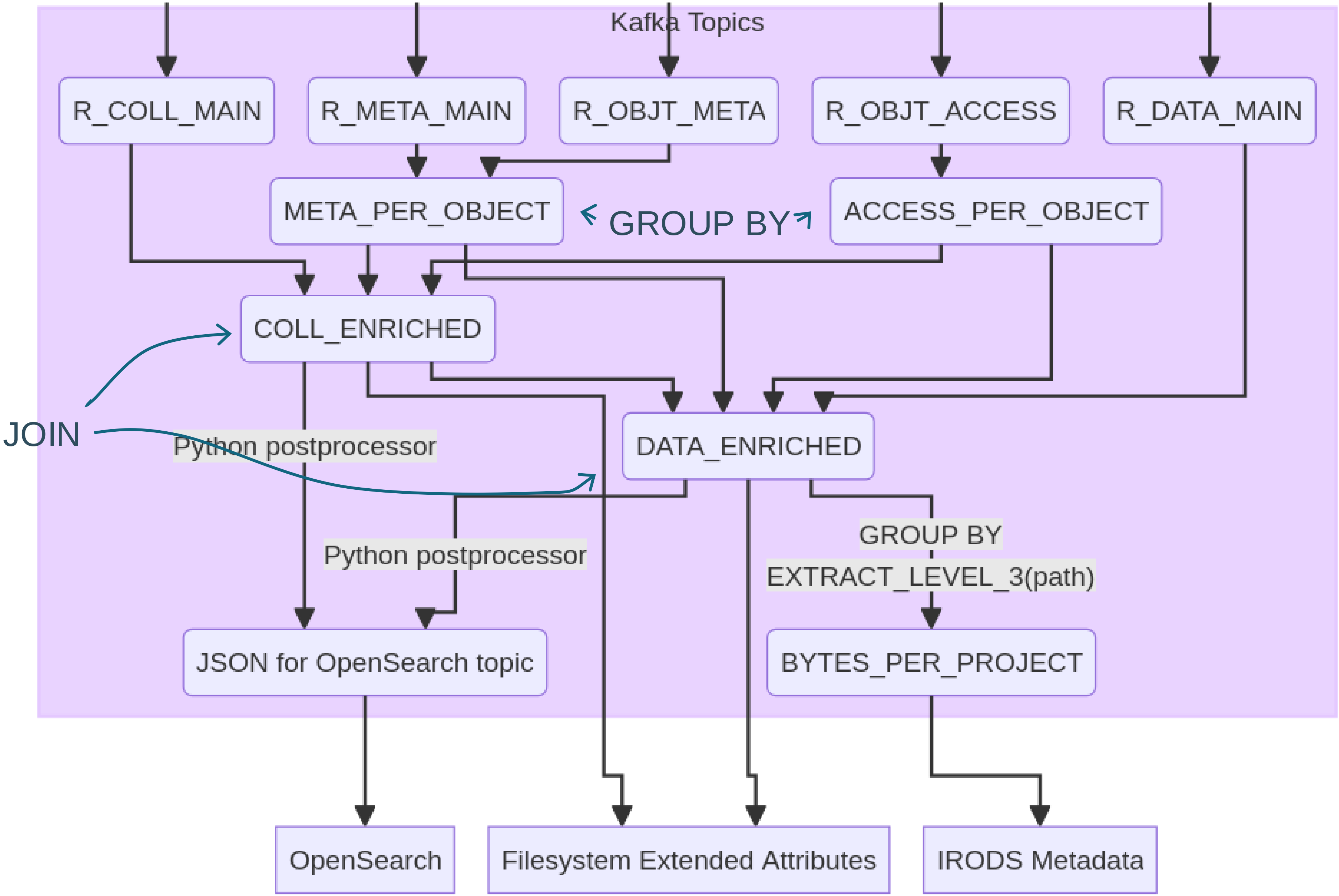
SQL — High-level Language

Table API — Declarative DSL

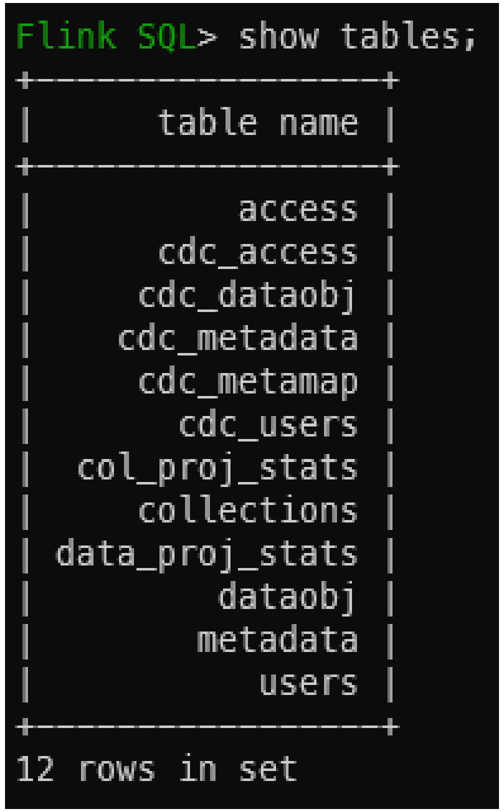
DataStream / DataSet API — Core APIs

Stateful Stream Processing — Low-level building block (streams, state, [event] time)

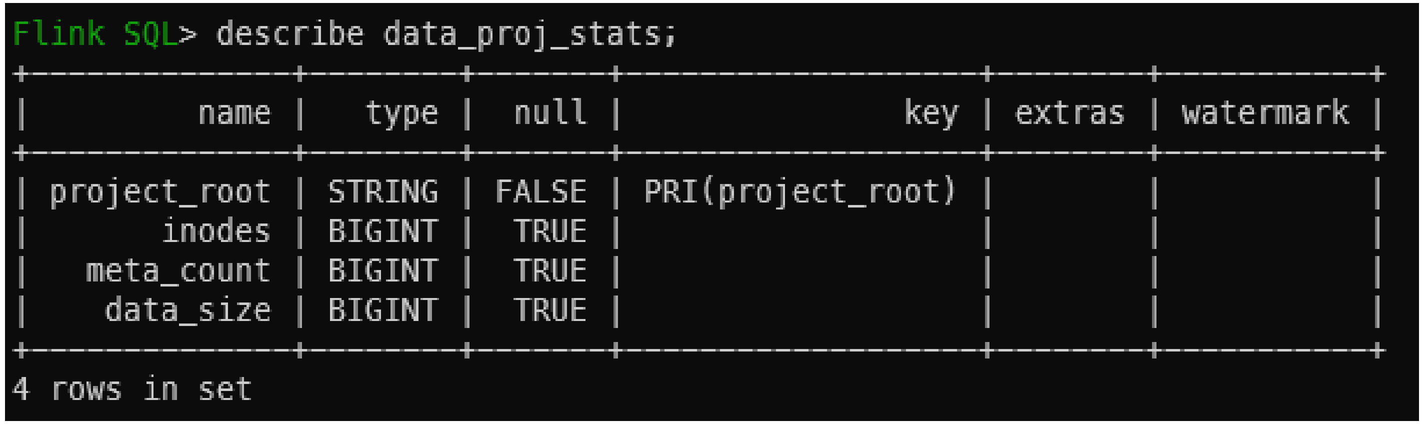
Flink SQL


Kafka Topics
R_COLL_MAIN
R_META_MAIN
R_OBJT_META
R_OBJT_ACCESS
R_DATA_MAIN
META_PER_OBJECT
GROUP BY
ACCESS_PER_OBJECT
COLL_ENRICHED
JOIN
Python postprocessor
DATA_ENRICHED
Python postprocessor
GROUP BY
EXTRACT_LEVEL_3(path)
JSON for OpenSearch topic
BYTES_PER_PROJECT
OpenSearch
Filesystem Extended Attributes
IRODS Metadata

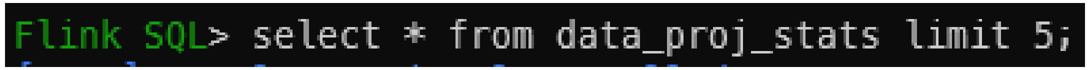
Flink SQL



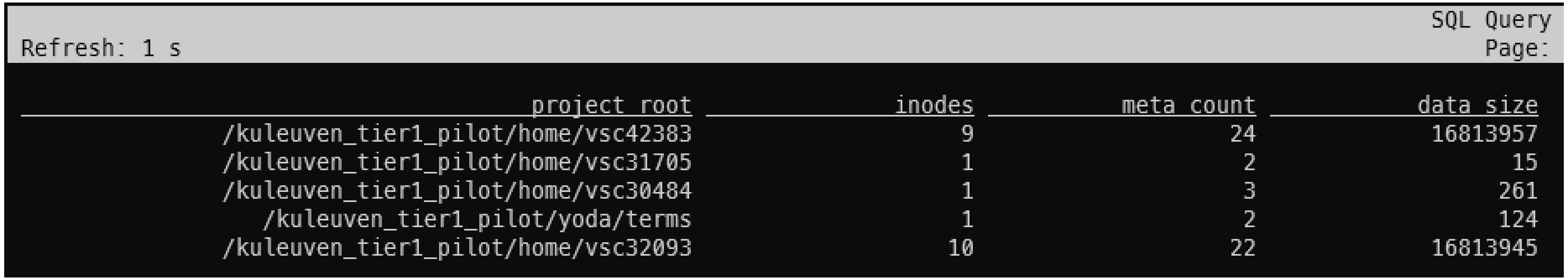
```
Flink SQL> show tables;
+-----------------+
|      table name |
+-----------------+
|          access |
|      cdc_access |
|     cdc_dataobj |
|    cdc_metadata |
|     cdc_metamap |
|       cdc_users |
|   col_proj_stats |
|     collections |
| data_proj_stats |
|         dataobj |
|        metadata |
|           users |
+-----------------+
12 rows in set
```



```
Flink SQL> describe data_proj_stats;
+--------------+--------+-------+-------------------+--------+-----------+
|         name |   type |  null |               key | extras | watermark |
+--------------+--------+-------+-------------------+--------+-----------+
| project_root | STRING | FALSE | PRI(project_root) |        |           |
|       inodes | BIGINT |  TRUE |                   |        |           |
|   meta_count | BIGINT |  TRUE |                   |        |           |
|    data_size | BIGINT |  TRUE |                   |        |           |
+--------------+--------+-------+-------------------+--------+-----------+
4 rows in set
```



```
Flink SQL> select * from data_proj_stats limit 5;
```



```
                                                                                    SQL Query
Refresh: 1 s                                                                            Page:

                                  project_root              inodes          meta_count            data_size
            /kuleuven_tier1_pilot/home/vsc42383                  9                  24             16813957
            /kuleuven_tier1_pilot/home/vsc31705                  1                   2                   15
            /kuleuven_tier1_pilot/home/vsc30484                  1                   3                  261
               /kuleuven_tier1_pilot/yoda/terms                  1                   2                  124
            /kuleuven_tier1_pilot/home/vsc32093                 10                  22             16813945
```

Flink SQL

```
SET 'pipeline.name' = 'irods-cdc-data';  SET 'parallelism.default' = '4';
EXECUTE STATEMENT SET
BEGIN
INSERT INTO data_enriched SELECT
  d.zone,        d.data_id as `object_id`,   d.coll_id,     d.data_name as name,
  c.path || '/' || d.data_name as `path`,      d.data_owner_name as owner_name,
  d.data_size,     d.data_is_dirty,           d.create_ts,   d.modify_ts,
  a.reader_ids,    a.owner_ids,               m.metadata,    c.metadata as col_metadata
FROM
  collections_enriched c INNER JOIN cdc_r_data_main d ON d.zone = c.zone and d.coll_id = c.object_id
  LEFT JOIN metadata_by_object m ON d.zone = m.zone and d.data_id = m.object_id
  LEFT JOIN access_by_object a ON d.zone = a.zone and d.data_id = a.object_id;
END;
```

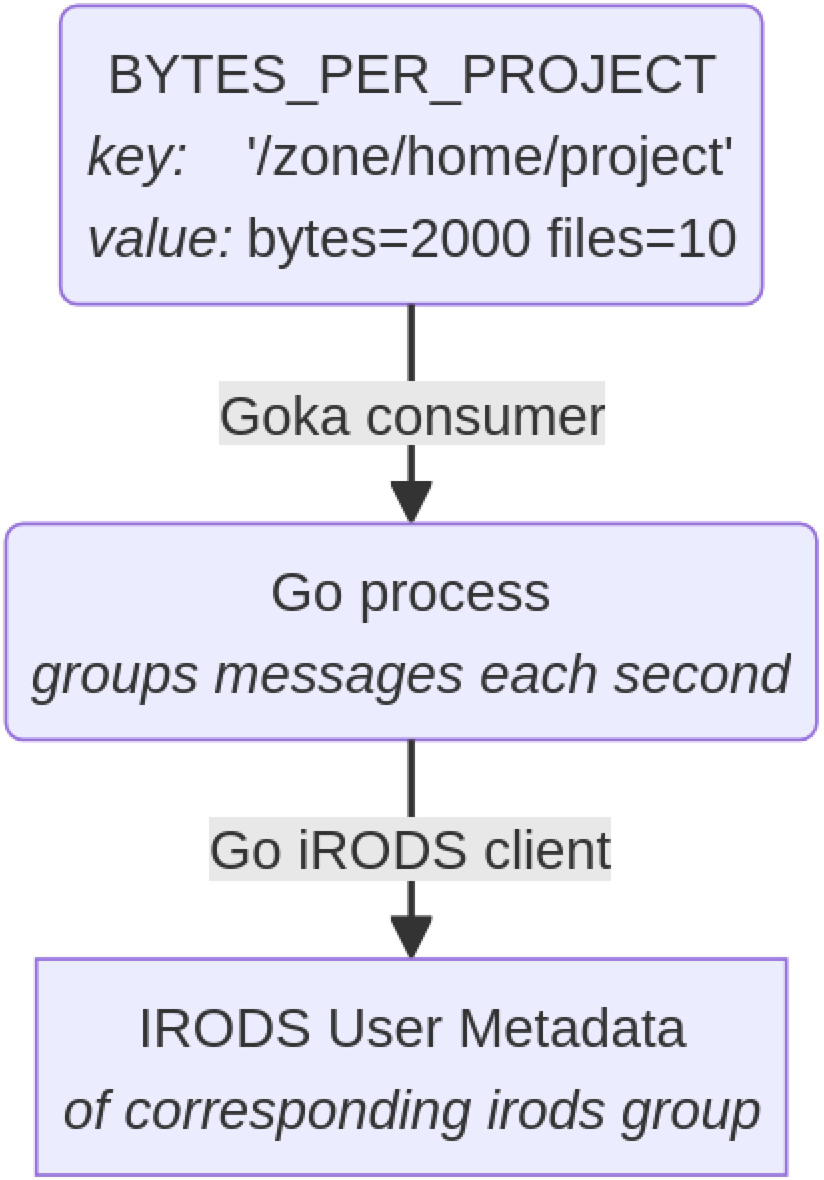
Application 1: OpenSearch

- Benefits over indexing rule engine:
  - Uniform flow across all iRODS zones
  - In principal no re-indexing needed
  - No additional load on iRODS server

- 5 million collections
- 193 million data objects
- < 19 hours initial processing time (can scale with resources)

Application 2: Project usage

- We have a Kafka topic continuously containing the current project usage
- Every second the last message per project is written as irods metadata of corresponding group
- For quota enforcing/reporting, irods metadata can be looked up



BYTES_PER_PROJECT
key: '/zone/home/project'
value: bytes=2000 files=10
Goka consumer
Go process
groups messages each second
Go iRODS client
IRODS User Metadata
of corresponding irods group

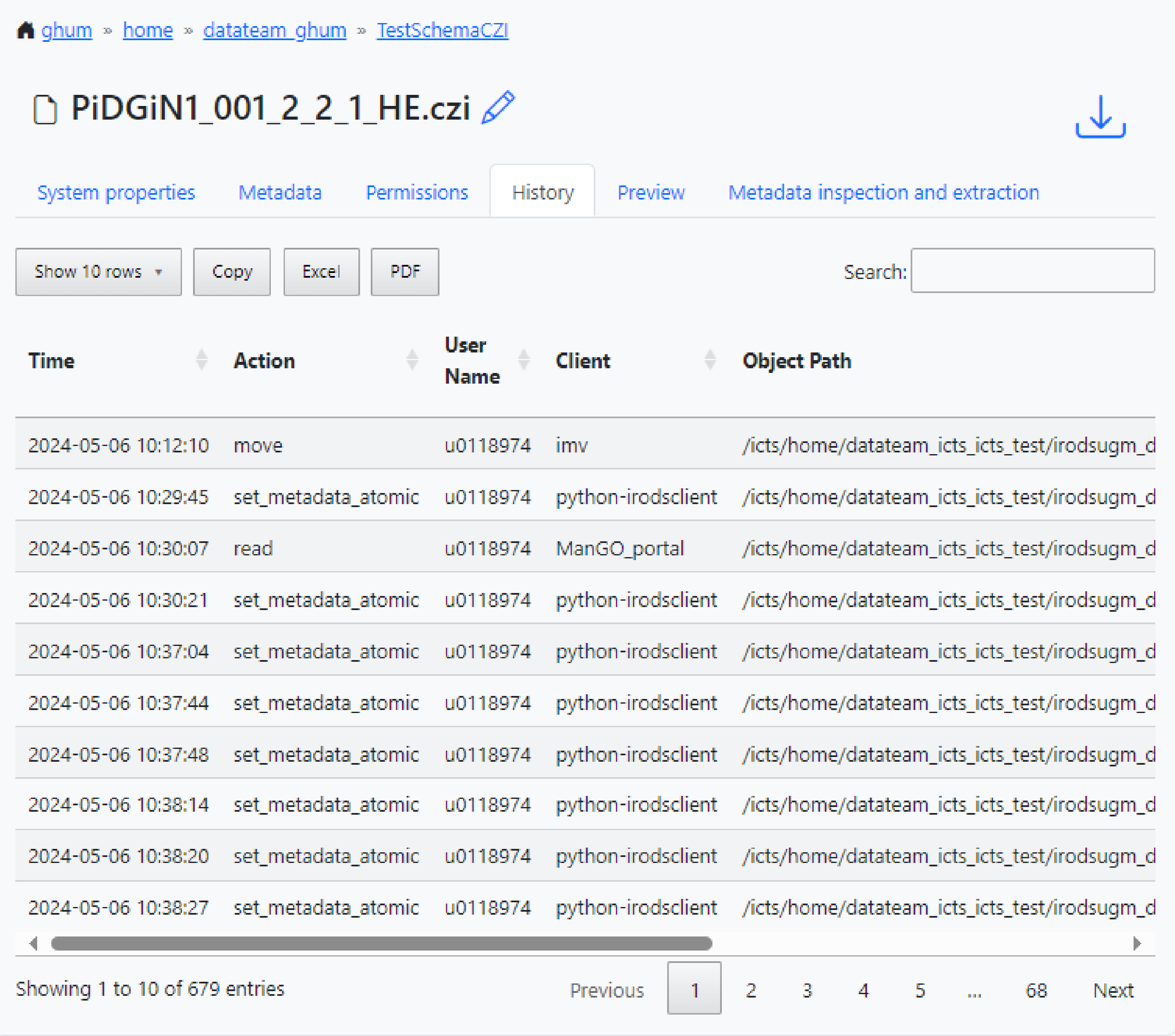
Application 3: Metadata as file system extended attributes

- Goal: incorporate iRODS metadata in file system snapshots
- Not for: backup/restore of complete iRODS zone
- But for: partial restores of data (no manual extraction of database dumps)

- Consumer of DATA_ENRICHED and COLL_ENRICHED
- Format metadata as extended attributes and set it on corresponding file on disk if it exists
- Some additional logic to account for the fact that processing can be delayed

Next steps: Audit pipeline

- From the database changelog, we know all changes
  - Lacks who triggered change
- We run the audit plugin (kafka instead of rabbitmq)
  - Lacks all changes in database (e.g. recursive chown)
  - Avoid MSI, use PRC
- Future work: try to link both
  - “Guess” which logs correspond
  - Expectation for audit logs is to be 100% accurate



ghum » home » datateam_ghum » TestSchemaCZI

PiDGiN1_001_2_2_1_HE.czi

Time	Action	User Name	Client	Object Path
2024-05-06 10:12:10	move	u0118974	imv	/icts/home/dataeam_icts_icts_test/irodsugm_d
2024-05-06 10:29:45	set_metadata_atomic	u0118974	python-irodsclient	/icts/home/dataeam_icts_icts_test/irodsugm_d
2024-05-06 10:30:07	read	u0118974	ManGO_portal	/icts/home/dataeam_icts_icts_test/irodsugm_d
2024-05-06 10:30:21	set_metadata_atomic	u0118974	python-irodsclient	/icts/home/dataeam_icts_icts_test/irodsugm_d
2024-05-06 10:37:04	set_metadata_atomic	u0118974	python-irodsclient	/icts/home/dataeam_icts_icts_test/irodsugm_d
2024-05-06 10:37:44	set_metadata_atomic	u0118974	python-irodsclient	/icts/home/dataeam_icts_icts_test/irodsugm_d
2024-05-06 10:37:48	set_metadata_atomic	u0118974	python-irodsclient	/icts/home/dataeam_icts_icts_test/irodsugm_d
2024-05-06 10:38:14	set_metadata_atomic	u0118974	python-irodsclient	/icts/home/dataeam_icts_icts_test/irodsugm_d
2024-05-06 10:38:20	set_metadata_atomic	u0118974	python-irodsclient	/icts/home/dataeam_icts_icts_test/irodsugm_d
2024-05-06 10:38:27	set_metadata_atomic	u0118974	python-irodsclient	/icts/home/dataeam_icts_icts_test/irodsugm_d

Showing 1 to 10 of 679 entries

Lessons learned

- Not so easy to set up
- Avoid exceeding disk space (or start over)
- Assign enough RAM space (or things are slow)
- Turn off snapshot locking (or mysql database hangs)
- Check whether mysql connector still runs
    - Apply trick when no user activity happens