Implementing data citation recommendations at the Earth Observation Data Centre

Bernhard Gößwein (TU Wien), Tomasz Miksa (TU Wien & SBA Research), Andreas Rauber (TU Wien), Wolfgang Wagner (TU Wien)

TECHNISCHE UNIVERSITÄT WIEN
Vienna | Austria

Introduction

Scientist
using EO client

Job

Back end

Input data

Result

Situation: Earth Observation (EO)

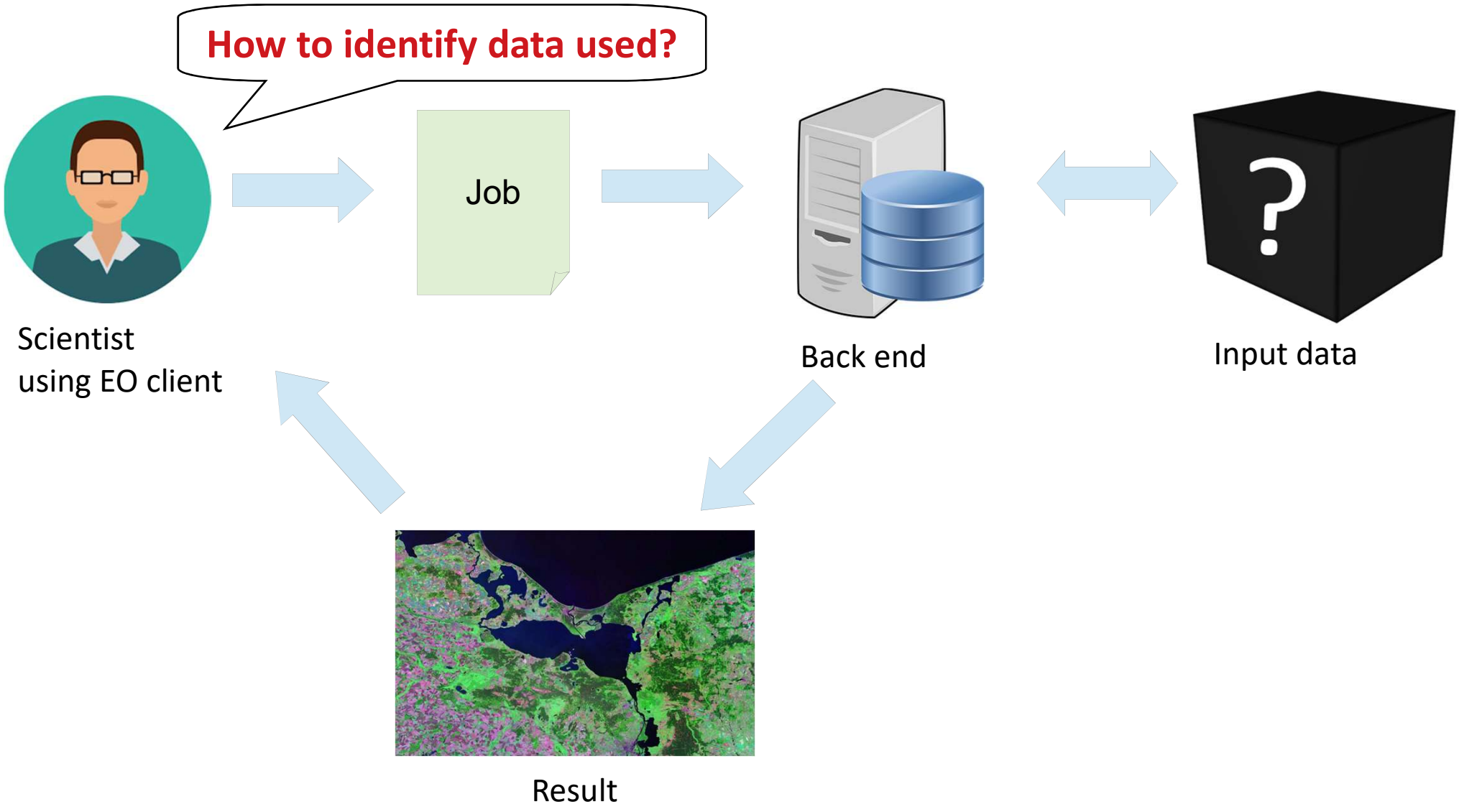
- Diverse set of data provider
- Processing happens at the data provider
- Backends provide data from similar sources e.g. ESA
- openEO provides a standardized API to access multiple backends

Introduction: EODC

- Since 2014 located in Vienna
- Provides data from the Copernicus Programme
- One of the main partners of the openEO project and will be compliant to the openEO standard.
- Connected to the HPC VSC 3

  providing >32.000 cores with >200 teraflops

Problem – Input data identification



Problem: Input data changed

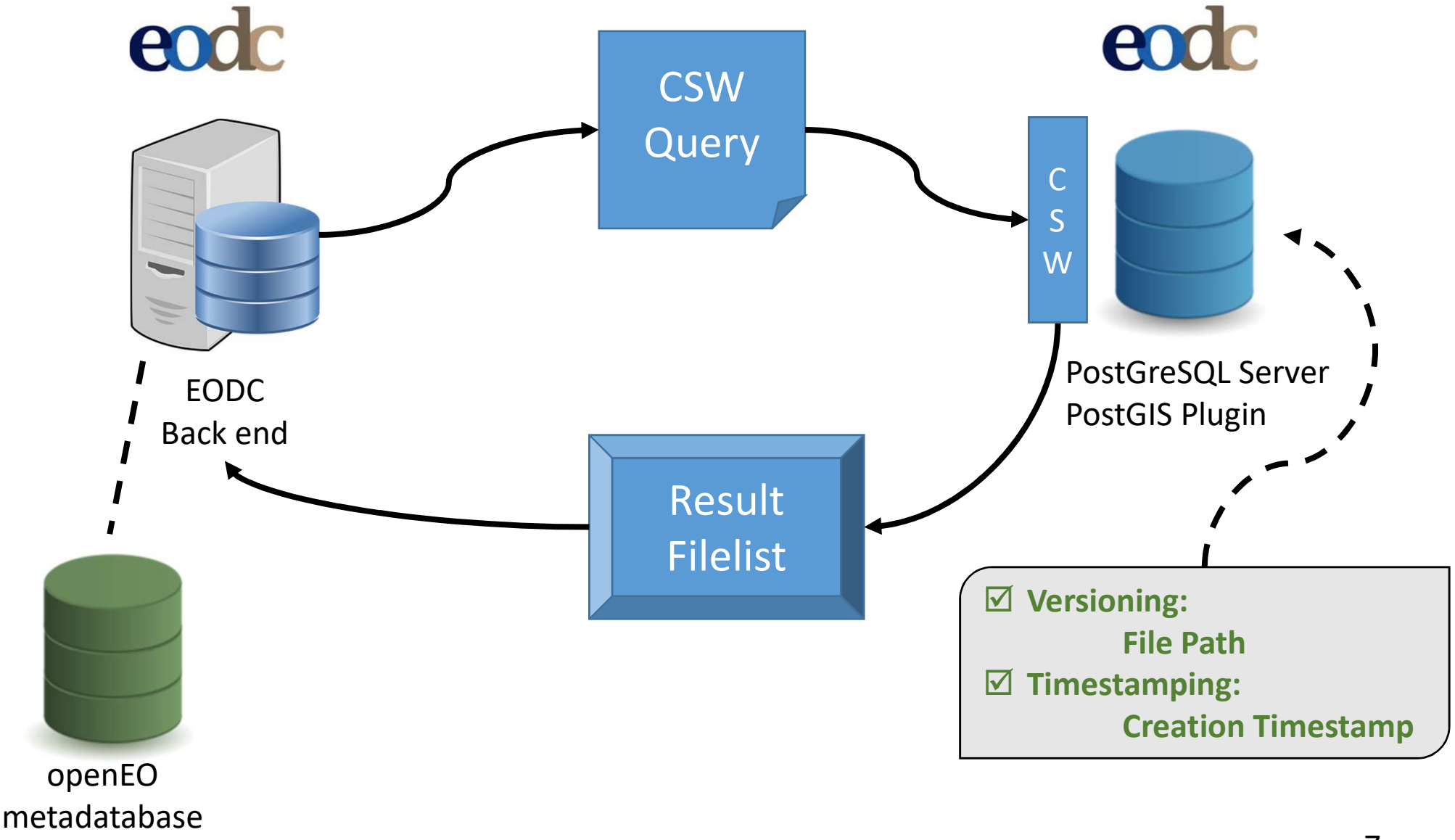
Query arguments

- **Temporal extent**: Date range of interest e.g. May of 2018
- **Spatial extent**: Geographic area of interest e.g. rectangle over Los Angeles
- **Spectral bands**: Bands of interest e.g. near infra red

Query result

- **Subset** of the backends satellite data storage
- Input data is usually big, since dimensions have not been reduced yet

Situation - EODC



Aim

Back end

- **Document** relevant software involved in processing, e.g. GDAL
- **NOT enable to** restore previous versions of the backend

Input data

- Enable identification of **CHANGING** data without making copies of subsets
- Provide easy way to cite and re-use input data

Result

- **Comparable** - Enable to identify whether differences come from data / environment or a **real** scientific phenomena

Methodology

- **RDA – Research Data Alliance**
  - Recommendations on data identification including citation and retrieval of data that existed at a certain point of time.

[DOI: 10.15497/RDA00016]

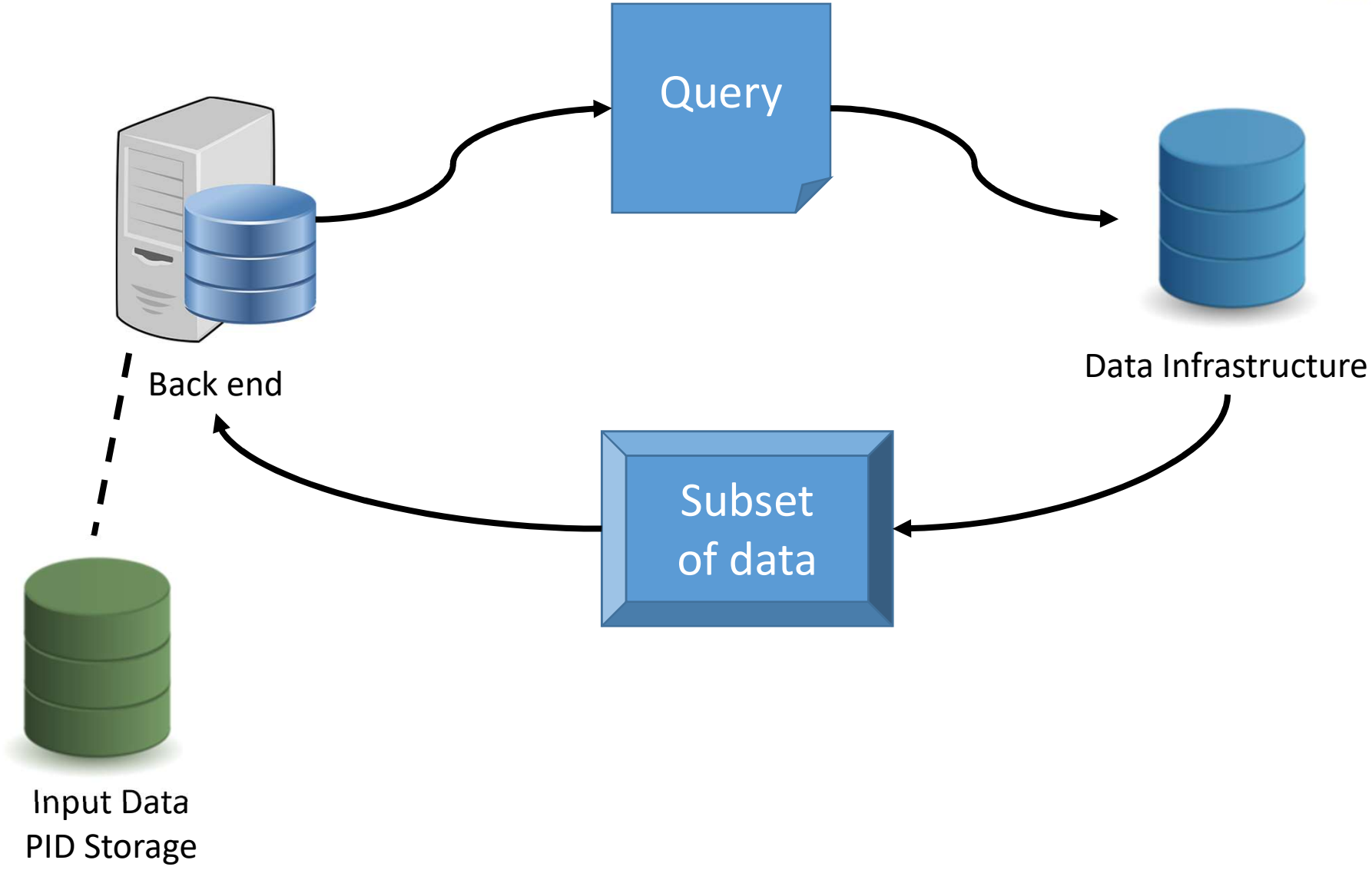
- **VFramework and Context Model**
  - Automatically document execution environments and enable their comparison.

[DOI:10.1016/j.jbi.2016.10.011]

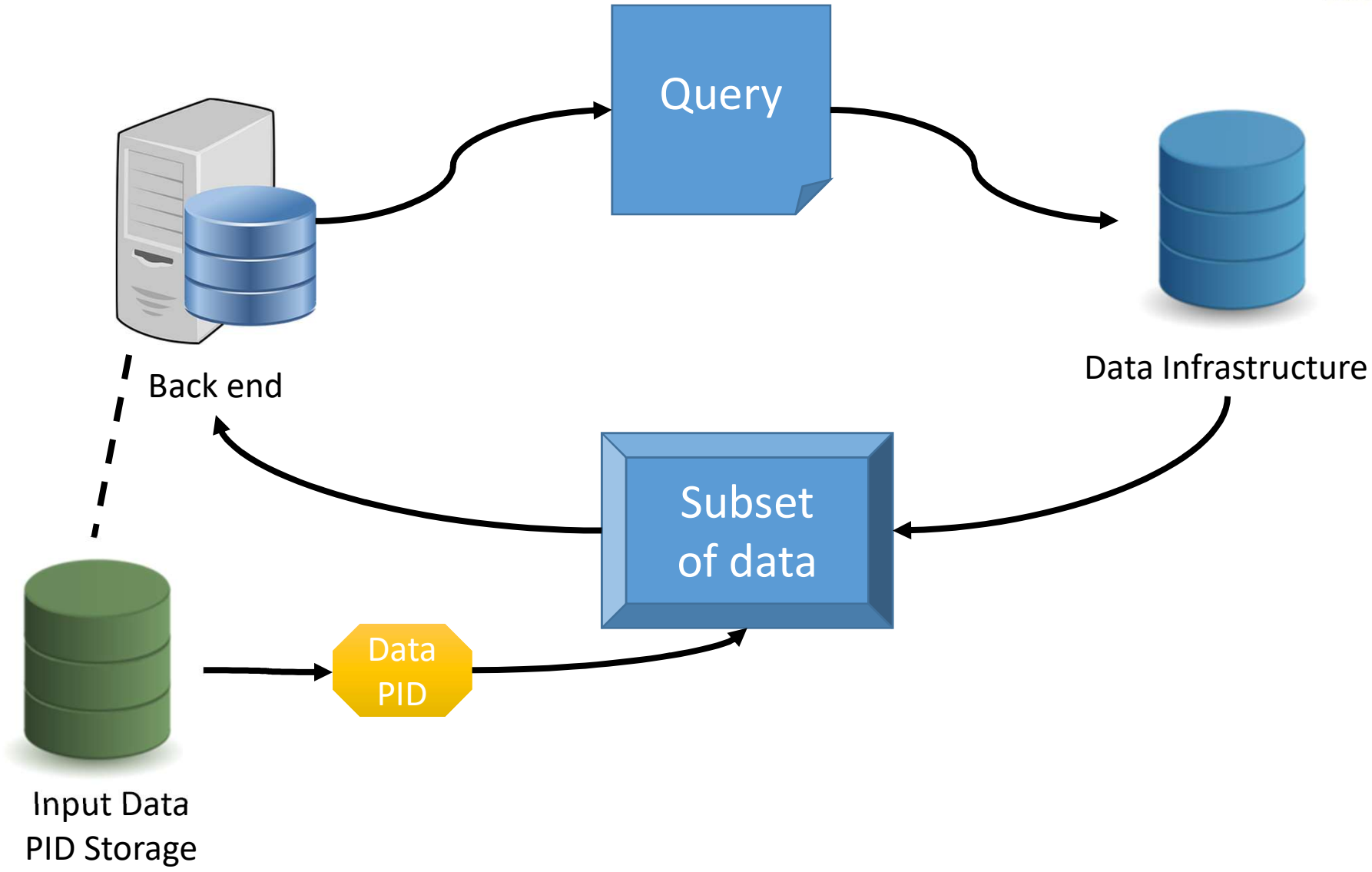
- **OpenEO Project**
  - Common EO interface enabling interoperability of EO backends. Allows researchers to run the same code on different backends.

[DOI:10.5281/zenodo.1065474]

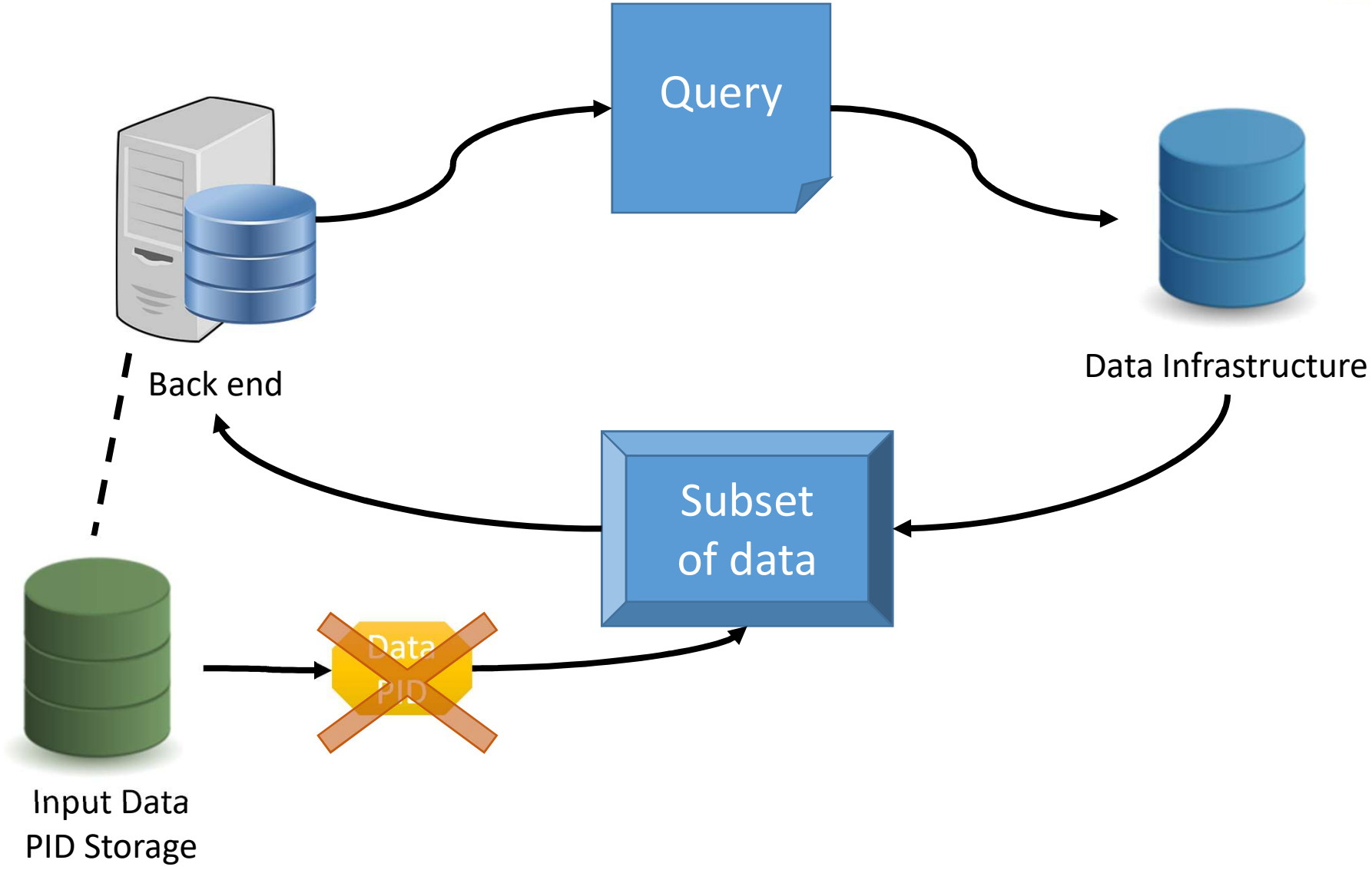
RDA - Data Identification



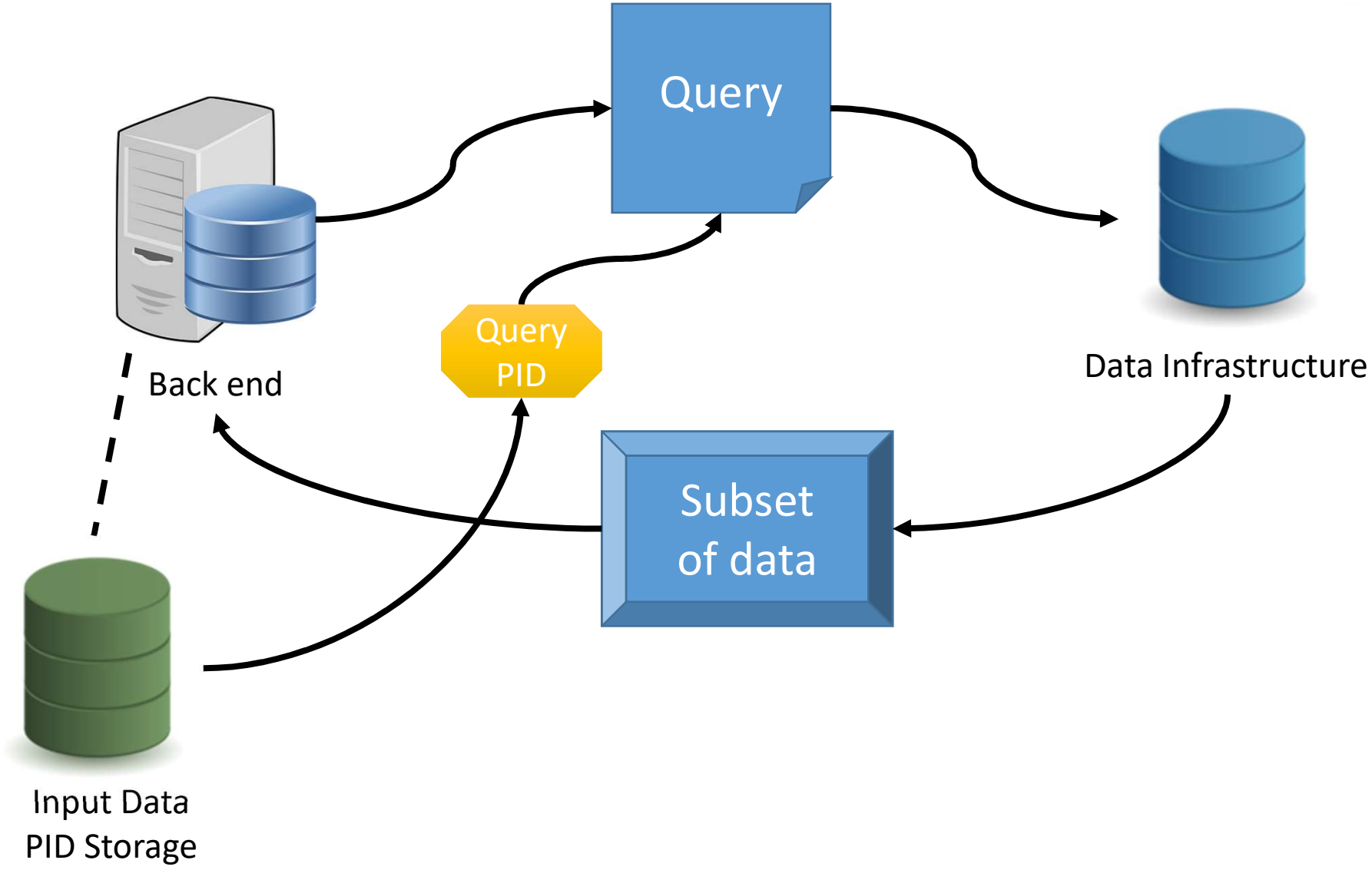
RDA - Data Identification



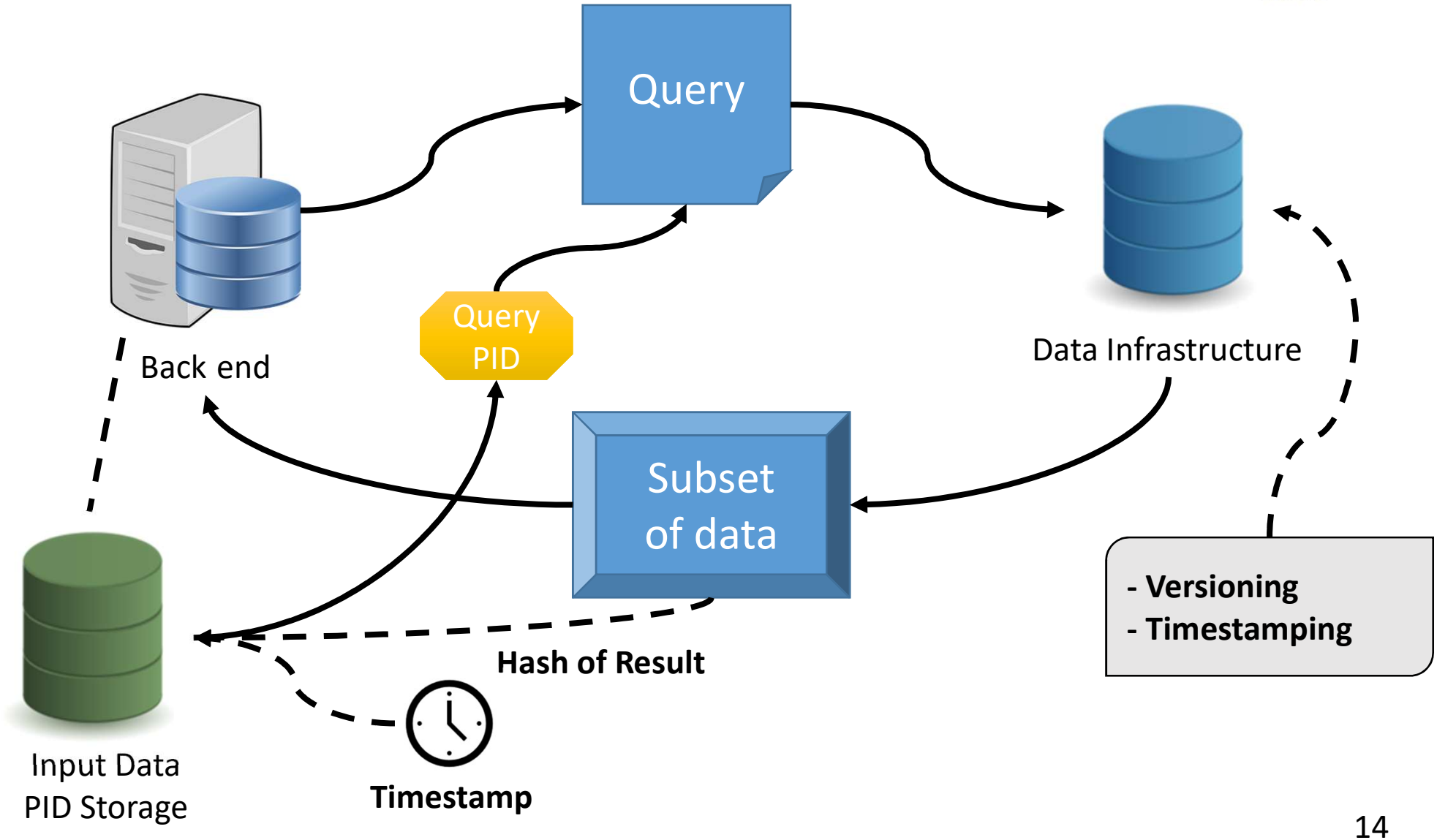
RDA - Data Identification



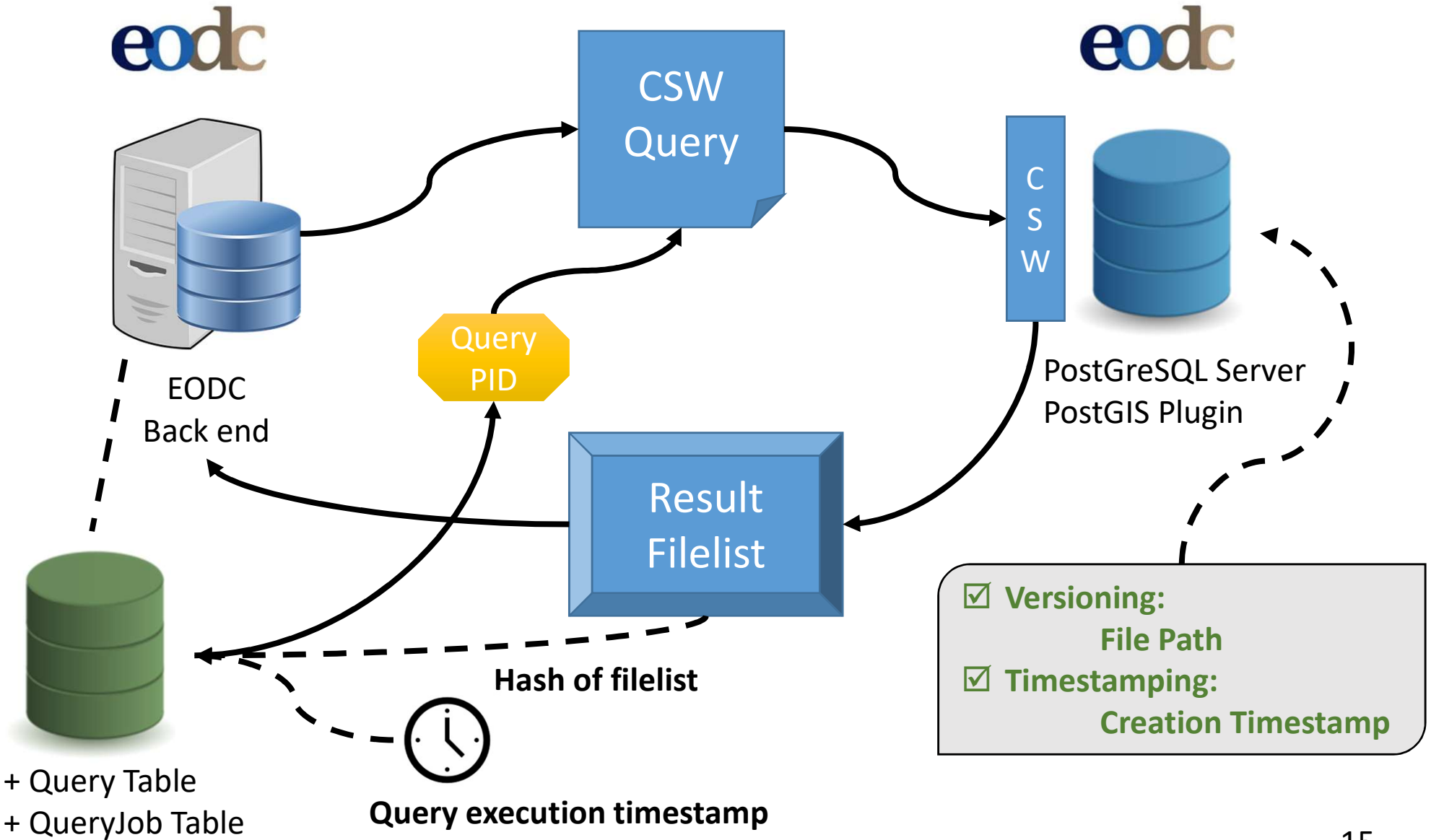
RDA - Data Identification



RDA - Data Identification



Solution - Data Identification



Solution: Query Table

- **Query Table**

Contains the elements of the Query Store defined in R3 from the RDA recommendations.

Query PID	Dataset PID	Original Query	Unique Query
VARCHAR(100)	VARCHAR(100)	TEXT	VARCHAR(300)
Query Hash	**Result Hash**	**Execution Timestamp**	**Metadata**
VARCHAR(65)	VARCHAR(65)	TIMESTAMP	TEXT

- **QueryJob Table**

Connects the Query Table with the Job Table (n : m mapping).

Solution: openEO Extensions

- **Re-use of Input Data**

The query PID can be used to define the input data of a new job execution, therefore the same data as the original execution can be used.

- **Input Data Landing Page**

Human readable and Machine readable landing page within the openEO API, to access the data per web browser or by the openEO client.

Solution: openEO Python client example

```
con = openeo.connect("http://openeo.local.127.0.0.1.nip.io")
# Choose dataset
processes = con.get_processes()
pgA = processes.get_collection(name="s2a_prd_msil1c")
pgA = processes.filter_daterange(pgA, extent=["2017-05-01", "2017-05-31"])
pgA = processes.filter_bbox(pgA, west=10.288696, south=45.935871,
east=12.189331, north=46.905246, crs="EPSG:4326")
# Choose processes
pgA = processes.ndvi(pgA, nir="B08", red="B04")
pgA = processes.min_time(pgA)
# Create and start job A out of the process graph A (pgA)
jobA = con.create_job(pgA.graph)
jobA.start_job()
# Get data PID of jobA
pidA = jobA.get_data_pid()
# Re-execute the query to print the
file_listA = con.get_filelist(pidA)
# Get state of the resultfiles, so i
# the original execution
file_listA["input_files"]["state"] #
```

```
# Take input data of job A by using the input data PID A of job A
pgC = processes.get_data_by_pid(data_pid=pidA)
# Choose processes
pgC = processes.ndvi(pgC, nir="B08", red="B04")
pgC = processes.min_time(pgC)
# Create and start Job C
jobC = con.create_job(pgC.graph)
jobC.start_job()
# re-execute query and get the resulting file list from the backend
pidC = jobC.get_data_pid()
file_listC = con.get_filelist(pidC)
# Compare resulting files with the original execution of jobA
(file_listA == file_listC) # Returns True
```

Solution: openEO Python client example

```
# Take input data of job A by using the input data PID A of job A
pgC = processes.get_data_by_pid(data_pid=pidA)
# Choose processes
pgC = processes.ndvi(pgC, nir="B08", red="B04")
pgC = processes.min_time(pgC)
# Create and start Job C
jobC = con.create_job(pgC.graph)
jobC.start_job()
# re-execute query and get the resulting file list from the backend
pidC = jobC.get_data_pid()
file_listC = con.get_filelist(pidC)
# Compare resulting files with the original execution of jobA
(file_listA == file_listC) # Returns True
```

- Reads the original CSW Query from the Query Table with the given PID.
- Adds the execution timestamp to the Query to leave out data versions that came after the original execution.
- Advantage of querying the used data exactly as it was in the original execution, compared to simply re-executing the same filter arguments.

Solution: Recommendations

R1, R2 – Data Versioning & Timestamping

Apply versioning to ensure earlier states of data sets can be retrieved.

Ensure that operations on data are timestamped, i.e. any additions, deletions are marked with a timestamp.

- Already applied by the EODC backend by having the file path as identifier and the creation date of the file as the timestamp since when it is available.

e.g. dataset:

{'timestamp': '2017-05-08 00:00:00',

'path': '/eodc/products/copernicus.eu/s2a_prd_msil1c/2017/05/04/

S2A_MSIL1C_20170504T101031_N0205_R022_T32TPR_20170504T101349.zip'}

UPDATE→

{'timestamp': '2018-04-01 10:22:03',

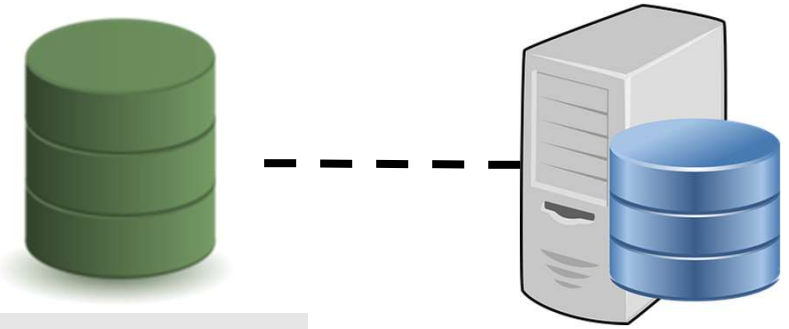
'path': '/eodc/products/copernicus.eu/s2a_prd_msil1c/2017/05/04/

S2A_MSIL1C_20170504T101031_N0205_R022_T32TPR_20170504T101349**_recalibrated20180205**.zip'}

Solution: Recommendations

R3 – Query Store Facilities

Provide means for storing queries and the associated metadata in order to re-execute them in the future.



+ Query Table
+ QueryJob Table

EODC
Back end

Query PID	Dataset PID	Original Query	Unique Query
VARCHAR(100)	VARCHAR(100)	TEXT	VARCHAR(300)

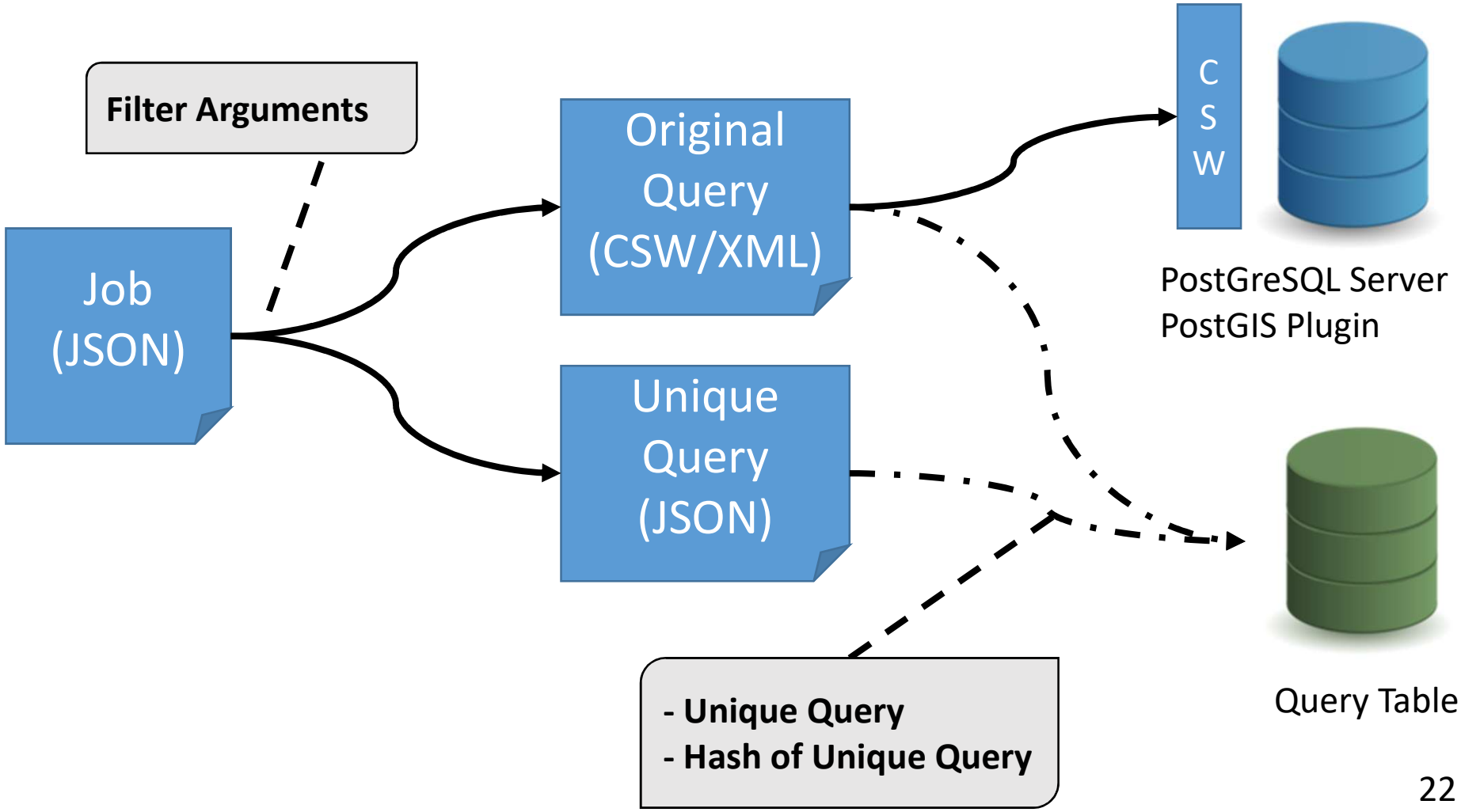
Query Hash	Result Hash	Execution Timestamp	Metadata
VARCHAR(65)	VARCHAR(65)	TIMESTAMP	TEXT

Additional Query Table in the meta database of the EODC backend.

Solution: Recommendations

R4 – Query Uniqueness

Re-write the query to a normalized form so that identical queries can be detected. Compute a checksum of the normalized query to efficiently detect identical queries.



Solution: Recommendations

R4 – Query Uniqueness

```
<?xml version="1.0" encoding="UTF-8"?>
<csw:GetRecords xmlns:csw="http://www.opengis.net/cat/csw/2.0.2" xmlns:apiso="http://www.opengis.net/cat/csw/apiso/1.0" xmlns:gmd="http://www.isotc211.org/2005/gmd"
                xmlns:gml="http://www.opengis.net/gml" xmlns:ogc="http://www.opengis.net/ogc" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" service="CSW"
                version="2.0.2" resultType="results" startPosition="1" maxRecords="1000" outputFormat="application/json" outputSchema="http://www.isotc211.org/2005/gmd"
                xsi:schemaLocation="http://www.opengis.net/cat/csw/2.0.2 http://schemas.opengis.net/csw/2.0.2/CSW-discovery.xsd">
   <csw:Query typeNames="csw:Record">
      <csw:ElementSetName>full</csw:ElementSetName>
      <csw:Constraint version="1.1.0">
         <ogc:Filter>
            <ogc:And>
               <ogc:PropertyIsEqualTo>
                  <ogc:PropertyName>apiso:ParentIdentifier</ogc:PropertyName>
                  <ogc:Literal>s2a_prd_msil1c</ogc:Literal>
               </ogc:PropertyIsEqualTo>
               <ogc:PropertyIsGreaterThanOrEqualTo>
                  <ogc:PropertyName>apiso:TempExtent_begin</ogc:PropertyName>
                  <ogc:Literal>2017-05-01T00:00:00Z</ogc:Literal>
               </ogc:PropertyIsGreaterThanOrEqualTo>
               <ogc:PropertyIsLessThanOrEqualTo>
                  <ogc:PropertyName>apiso:TempExtent_end</ogc:PropertyName>
                  <ogc:Literal>2017-05-31T23:59:59Z</ogc:Literal>
               </ogc:PropertyIsLessThanOrEqualTo>
               <ogc:BBOX>
                  <ogc:PropertyName>ows:BoundingBox</ogc:PropertyName>
                  <gml:Envelope>
                     <gml:lowerCorner>46.905246 10.288696</gml:lowerCorner>
                     <gml:upperCorner>45.935871 12.189331</gml:upperCorner>
                  </gml:Envelope>
               </ogc:BBOX>
               <ogc:PropertyIsLessThanOrEqualTo>
                  <ogc:PropertyName>apiso:Modified</ogc:PropertyName>
                  <ogc:Literal>2019-03-31 17:36:43.064445</ogc:Literal>
               </ogc:PropertyIsLessThanOrEqualTo>
            </ogc:And>
         </ogc:Filter>
      </csw:Constraint>
      <ogc:SortBy>
         <ogc:SortProperty>
            <ogc:PropertyName>dc:date</ogc:PropertyName>
            <ogc:SortOrder>ASC</ogc:SortOrder>
         </ogc:SortProperty>
      </ogc:SortBy>
   </csw:Query>
</csw:GetRecords>
```

Original CSW query at the EODC.

```
{
    "filter_bbox": {
        "left": 650000,
        "right": 672000,
        "srs": "EPSG:32632",
        "top": 5161000
        },
    "filter_daterange": {
        "from": "2018-01-01",
        "to": "2018-01-08"
        },
    "product_id": "s1a_csar_grdh_iw"
}
```

Unique Query

Unique (or Normalized) Query, by ordering the filter arguments alphabetically.

Solution: Recommendations

R5 – Stable Sorting

Ensure that the sorting of the records in the data set is unambiguous and reproducible

```
        <ogc:SortBy>
            <ogc:SortProperty>
                <ogc:PropertyName>dc:path</ogc:PropertyName>
                <ogc:SortOrder>ASC</ogc:SortOrder>
            </ogc:SortProperty>
        </ogc:SortBy>
    </csw:Query>
</csw:GetRecords>
```

Snipped of the original EODC CSW Query that is responsible for the stable sorting of the resulting file list.

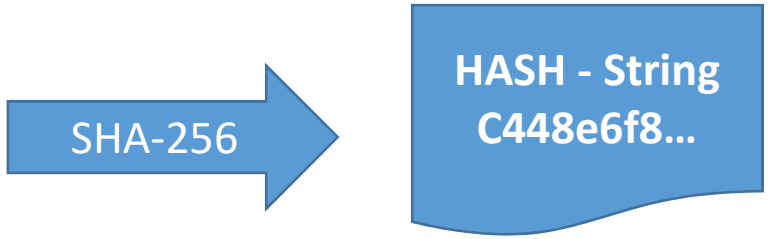
OGC®
Making location count.

Solution: Recommendations

R6 – Result Set Verification

Compute fixity information (checksum) of the query result set to enable verification of the correctness of a result upon re-execution.


```
{
    "date": "2017-01-04",
    "name": "S2A_MSIL1C_20170104T101402_N0204_R022_T32TPT_20170104T101405",
    "path": "/eodc/products/copernicus.eu/s2a_prd_msil1c/2017/01/04/S2A_MSIL1C_20170104T101402_N0204_R022_T32TPT_20170104T101405.zip"
},
{
    "date": "2017-01-04",
    "name": "S2A_MSIL1C_20170104T101402_N0204_R022_T32TPS_20170104T101405",
    "path": "/eodc/products/copernicus.eu/s2a_prd_msil1c/2017/01/04/S2A_MSIL1C_20170104T101402_N0204_R022_T32TPS_20170104T101405.zip"
},
{
    "date": "2017-01-04",
    "name": "S2A_MSIL1C_20170104T101402_N0204_R022_T32TQS_20170104T101405",
    "path": "/eodc/products/copernicus.eu/s2a_prd_msil1c/2017/01/04/S2A_MSIL1C_20170104T101402_N0204_R022_T32TQS_20170104T101405.zip"
},
{
    "date": "2017-01-04",
    "name": "S2A_MSIL1C_20170104T101402_N0204_R022_T33UVP_20170104T101405",
    "path": "/eodc/products/copernicus.eu/s2a_prd_msil1c/2017/01/04/S2A_MSIL1C_20170104T101402_N0204_R022_T33UVP_20170104T101405.zip"
},
{
    "date": "2017-01-04",
    "name": "S2A_MSIL1C_20170104T101402_N0204_R022_T33TVM_20170104T101405",
    "path": "/eodc/products/copernicus.eu/s2a_prd_msil1c/2017/01/04/S2A_MSIL1C_20170104T101402_N0204_R022_T33TVM_20170104T101405.zip"
},
{
    "date": "2017-01-04",
    "name": "S2A_MSIL1C_20170104T101402_N0204_R022_T33TUN_20170104T101405",
    "path": "/eodc/products/copernicus.eu/s2a_prd_msil1c/2017/01/04/S2A_MSIL1C_20170104T101402_N0204_R022_T33TUN_20170104T101405.zip"
},
{
    "date": "2017-01-04",
    "name": "S2A_MSIL1C_20170104T101402_N0204_R022_T33UUQ_20170104T101405",
    "path": "/eodc/products/copernicus.eu/s2a_prd_msil1c/2017/01/04/S2A_MSIL1C_20170104T101402_N0204_R022_T33UUQ_20170104T101405.zip"
},
{
    "date": "2017-01-04",
    "name": "S2A_MSIL1C_20170104T101402_N0204_R022_T33UUP_20170104T101405",
    "path": "/eodc/products/copernicus.eu/s2a_prd_msil1c/2017/01/04/S2A_MSIL1C_20170104T101402_N0204_R022_T33UUP_20170104T101405.zip"
},
{
    "date": "2017-01-04",
    "name": "S2A_MSIL1C_20170104T101402_N0204_R022_T32UQV_20170104T101405",
    "path": "/eodc/products/copernicus.eu/s2a_prd_msil1c/2017/01/04/S2A_MSIL1C_20170104T101402_N0204_R022_T32UQV_20170104T101405.zip"
},
```

Result of the query execution, the list of files needed for the processing.

Solution: Recommendations

R7, R8 – Query Timestamping & Query PID

Assign a timestamp to the query based on the last update to the entire database (or the last update to the selection of data affected by the query or the query execution time). This allows retrieving the data as it existed at the time a user issued a query.

Assign a new PID to the query if either the query is new or if the result set returned from an earlier identical query is different due to changes in the data. Otherwise, return the existing PID.

- Storing the Query execution timestamp.
- Generated by the Python library UUID, which is also used for generating other IDs at the EODC backend.

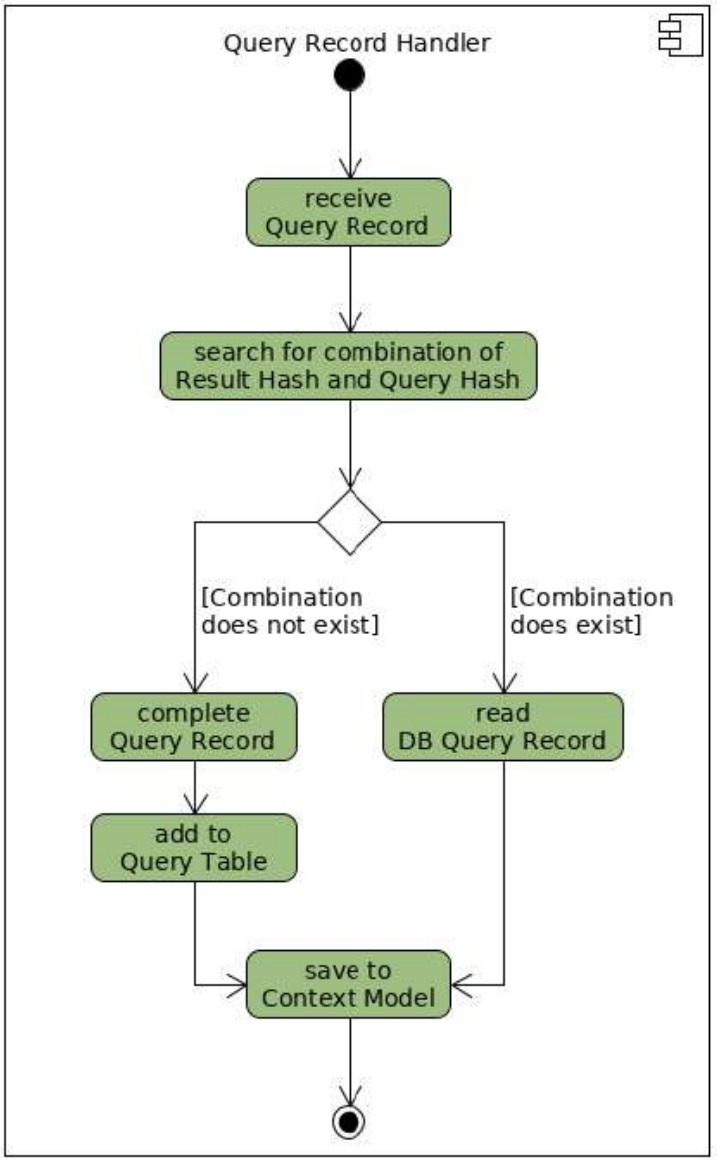
e.g. qu-5c45fa2a-52b9-4c7a-9023-37c34bdbd139

Solution: Recommendations

R9 – Store Query

Store query and metadata (e.g. PID, original and normalized query, query & result set checksum, timestamp, superset PID, data set description, and other) in the query store.

- Additional Query Table
- Additional Table for a n:m mapping of Jobs and Queries.
- Added program logic to return existing PID or generate a new one if necessary (see Figure on the right).



Solution: Recommendations

R10 – Automated Citation Texts

Generate citation texts in the format prevalent in the designated community for lowering the barrier for citing the data. Include the PID into the citation text snippet.

- Generated citation text dependent on the source dataset, the date of executing the Query and the human readable landing page URL.

> Copernicus Sentinel data (2017). Retrieved from EODC, Austria [2019-04-17], processed by ESA. PID: http://openeo.local.127.0.0.1.nip.io/data/qu-d1701f4e-e7c5-4a83-92e0-9facbd401a06

Example citation text of a used query PID.

Solution: Recommendations

R11 – Landing Page

Cite this dataset:

Using this data set or resource, you should cite it with the following citation text:

Copernicus Sentinel data (2017). Retrieved from EODC, Austria [2019-04-17], processed by ESA. PID: http://openeo.local.127.0.0.1.nip.io/data/qu-d1701f4e-e7c5-4a83-92e0-9facbd401a06

Show Result **JSON**

Source data description

Sentinel-2 is a multispectral, high-resolution, optical imaging mission, developed by the European Space Agency (ESA) in the frame of the Copernicus program of the European Commission.

Dataset Metadata

Organization	EODC
Data source (Author)	ESA - Copernicus Program
Data source Identifier	s2a_prd_msil1c
Date of creation	2019-04-17 15:46:11.728540
Spatial Extent	BoundingBox CRS: EPSG:4326 WEST: 10.288696, EAST: 45.935871 SOUTH: 45.935871, NORTH: 46.905246
Temporal Extent	from 2017-05-01 to 2017-05-31

Solution: Recommendations

R12 – Machine Actionability

Provide an API / machine actionable landing page to access metadata and data via query re-execution.

JSON representation of the information on the landing page.
Note: This Screenshot from the Firefox web browser, therefore it shows not a raw JSON format.

Solution: Recommendations

R13, R14 – Technology Migration & Migration Verification

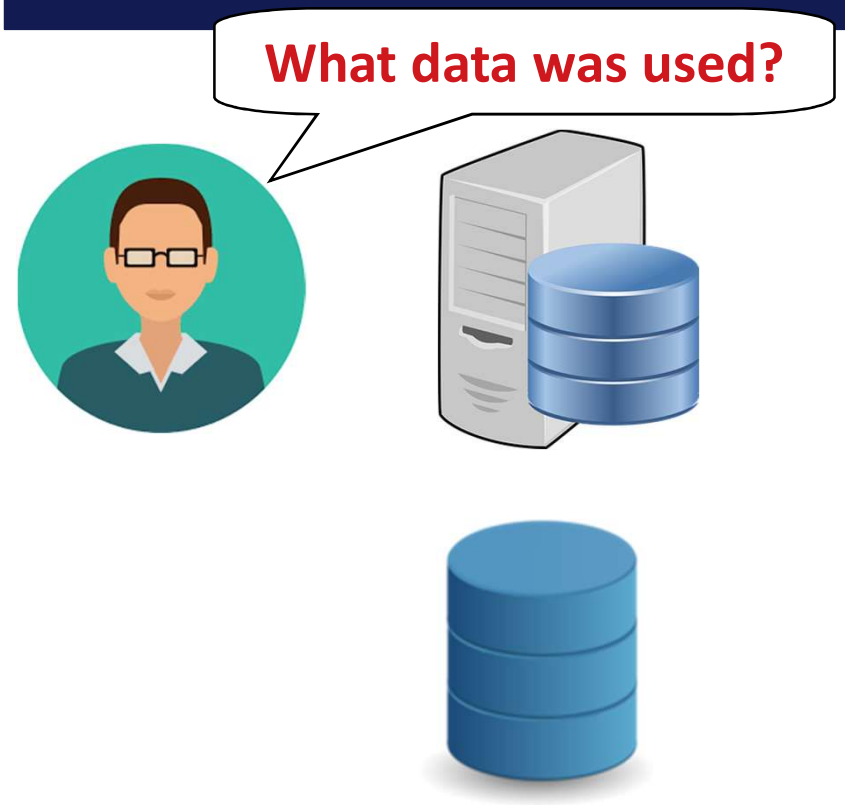
When data is migrated to a new representation (e.g. new database system, a new schema or a completely different technology), migrate also the queries and associated fixity information.

Verify successful data and query migration, ensuring that queries can be re-executed correctly.

- We did not implement R13 and R14, since the EODC has no plan to migrate to a different system. Therefore, there was no need to implement it yet. Nevertheless, this becomes relevant if a migration occurs in the future.

Not Implemented

Evaluation: Data citation



Cite input data

- Read the generated query PID via openEO client or the landing page.
- Copy the automatically generated citation text from the landing page.

Explore input data of others

- Resolvable used query PID and get additional information about the data.
- Possible via human readable landing page and via openEO client.

Re-use of input data

- Use our openEO API extension to use the same input data via query PID in your openEO client.

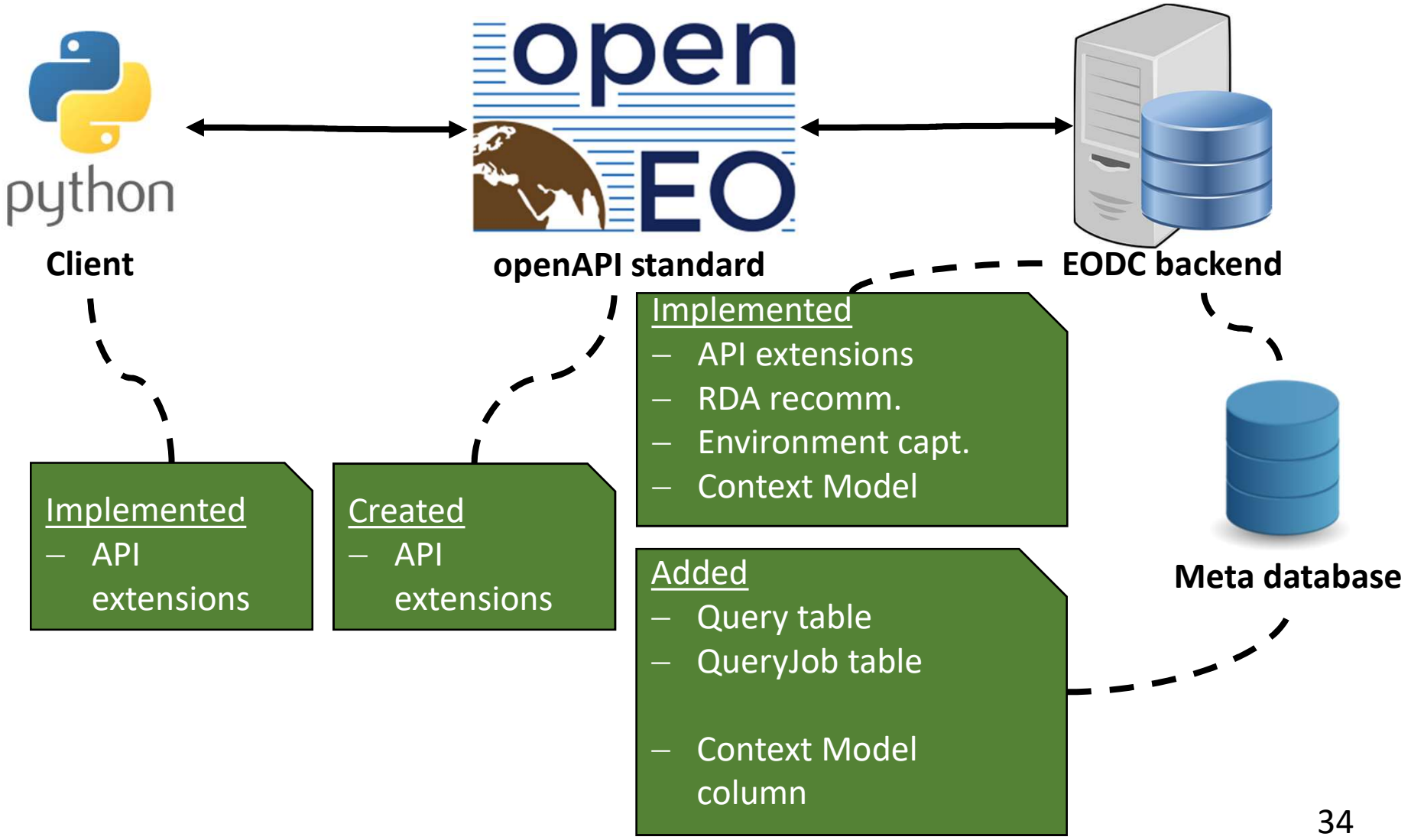
Evaluation: Performance & Storage

- **Evaluation Setup**
  - Duplicate of the EODC backend.
  - 18 test cases from publications using EODC.

- **Result – Performance**
  - The additional duration was between 20ms to 170ms with an overall computation duration between 10s to 20 minutes.

- **Result – Storage**
  - Storage is independent on job configuration and max. 2.677kB per job.

Conclusion


python
Client
openAPI standard
EODC backend
Implemented
– API extensions
Created
– API extensions
Implemented
– API extensions
– RDA recomm.
– Environment capt.
– Context Model
Added
– Query table
– QueryJob table
– Context Model column
Meta database

Conclusion

Conclusion

- Researchers do not have to change the way they work, but can describe and cite their input data.
- Solution is capable of making the input data reusable in an easy way provided by the openEO client.
- The solution implemented at the EODC backend has minimal performance and storage impact.

Future Work

- Adaption to future releases of the openEO API.
- Implementation on different backend types e.g. non-file-based.

Data identification and process monitoring for reproducible earth observation research

Bernhard Gößwein
TU Wien
Vienna, Austria

Tomasz Miksa
TU Wien & SBA Research
Vienna, Austria

Andreas Rauber
TU Wien
Vienna, Austria

***Abstract*—Earth observation researchers use specialised computing services for satellite image processing offered by various data backends. The source of data is often the same, for example Sentinel-2 satellites operated by the European Space Agency, but the way how data is pre-processed, corrected, updated, and later analysed may differ among the backends. Backends often lack mechanisms for data versioning, for example, data corrections are not tracked. Furthermore, an evolving software stack used for data processing remains a black box to researchers. Researchers have no means to identify why executions of the same code deliver different results. This hinders repeatability and reproducibility of earth observation experiments. In this paper, we present how infrastructure of existing earth observation data backends can be modified to support reproducibility. The proposed extensions are based on recommendations of the Research Data Alliance regarding data identification and the VFramework for process capturing. We implemented our approach at the Earth Observation Data Centre, which is a partner within the openEO project. We evaluated the solution on typical usage scenarios. We also provide performance and storage measures to evaluate the impact of the modifications on performance. The results indicate reproducibility can be supported with minimal performance and storage overhead.**

I. Introduction

Earth Observation (EO) data consists mostly of satellite images. Similar as in the other eScience disciplines, data is too big to be downloaded for local analysis. The solution is to store it in high-performance computational backends, process it there, and browse the results or download resulting figures or numbers [13].

Such an approach addresses the performance issues, but does not allow researchers to take a full control of the environment in which their experiments are executed. The backends act as black boxes to the researchers with no possibility of getting information on environment configuration, e.g. software libraries used in processing and their versions. Studies in different domains show that environment can have impact on reproducibility of scientific experiments and must be documented in order to ensure reproducibility [4] [1] [8]. Still the vast majority of backend providers do not share the environment information.

Another problem deals with a precise identification of data used for processing. EO backends in Europe usually obtain data from the same source, for example from the Sentinel-2 satellites operated by the European Space Agency (ESA). The ESA releases updates and corrections to data in cases when one of the instruments used for observation was wrongly calibrated or broken and raw data had to be pre-processed again. Updated data is released to backend operators. Usually there is no versioning mechanism for data. Researchers do not know which version of data was used in their study, i.e. before or after the correction was made available at the backend. This leads to a problem that scientists are not capable of precisely identifying the input data of their experiments, which hinders reproducibility and in turn undermines trust in the results.

Research Data Alliance (RDA) has identified 14 general rules [2] for identification of data used in computation that allows to cite and retrieve that data as it existed at a certain point in time. The VFramework [8] and context model [10] were proposed to automatically document environments in which computational workflows execute and to enable their comparison. The openEO project [7] works on creating a common EO interface to enable interoperability of EO backends by allowing researchers to run their experiments on different backends without reimplementing their code.

In this paper, we build on top of these developments and present a solution improving reproducibility of earth observation experiments executed at the openEO compliant backends. We follow the RDA recommendations for data identification and present how data provided by backends is made identifiable by assigning identifiers to queries made by researchers. We discuss which specific information must be captured, which interfaces must be modified, and which software components must be implemented. We also show how jobs executed at backends can be captured and compared using the VFramework to identify whether any differences in software dependencies among two executions exist. We implemented our solution for the backend of the Earth Observation Data Centre for Water Resources Monitoring (EODC). In evaluation we simulated typical use cases representing updates of data and changes in the backend environment. We also measured the performance and storage impact on the backend, which turned out to be minimal.

The remainder of this paper is structured as follows. Section II presents related work that is a basis of our solution and provides earth observation context. Section III presents architecutre of the proposed solution. Section IV presents implementation of the prototype at the EODC backend. Section V presents methods offered to researchers enhancing reproducibility. Section VII describes the experimental evaluation and discussion. Conclusion appears in Section VIII.

15th eScience 2019
INTERNATIONAL CONFERENCE

Bernhard Gößwein, Tomasz Miksa, Andreas Rauber, Wolfgang Wagner. Data Identification and Process Monitoring for Reproducible Earth Observation Research. IEEE eScience 2019, San Diego, USA.