Provenance as a Building Block for an Open Science Infrastructure

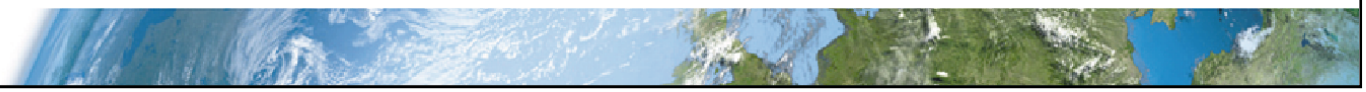
Andreas Schreiber

German Aerospace Center (DLR)
Cologne/Berlin, Germany

ISGC 2018, Taipei, Taiwan

Topics

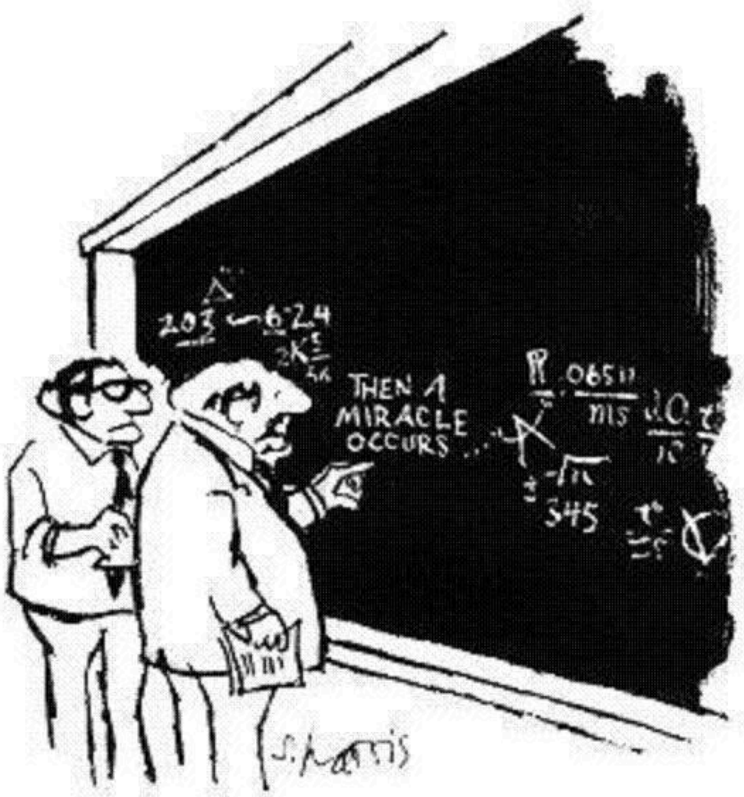
- Reproducibility
- Provenance and PROV
- Storing provenance
- Gathering provenance



Reproducibility

Reproducibility *in (data) science* is based on

- Open Source Software
- Code Reviews
- Code Repositories
- Publications with code
- Container (Docker etc.)
- Workflows
- (Electronic) laboratory notebooks
- Open data formats
- Data management
- **Metadata and Provenance**



"I think you should be more explicit here in step two."

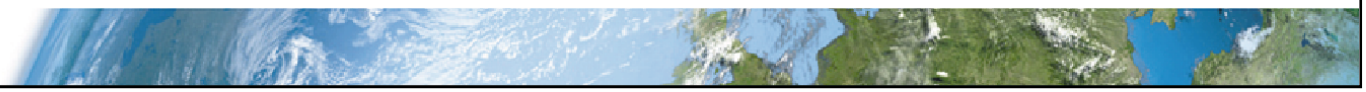
Provenance

Basics

- Provenance refers to the source of information and the process that led to its existence
  - *Where did I get this file?*
  - *How did it come to exist?*
- Provenance information is critical to users trying to understand where a particular data file came from

Other and related terms

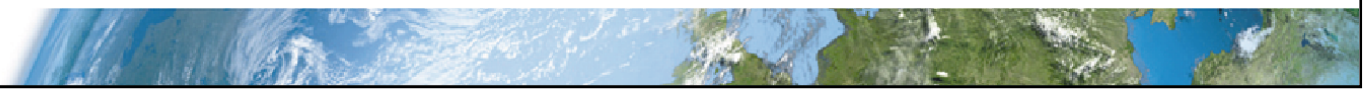
- Traceability
- Lineage
- Logging
- Monitoring



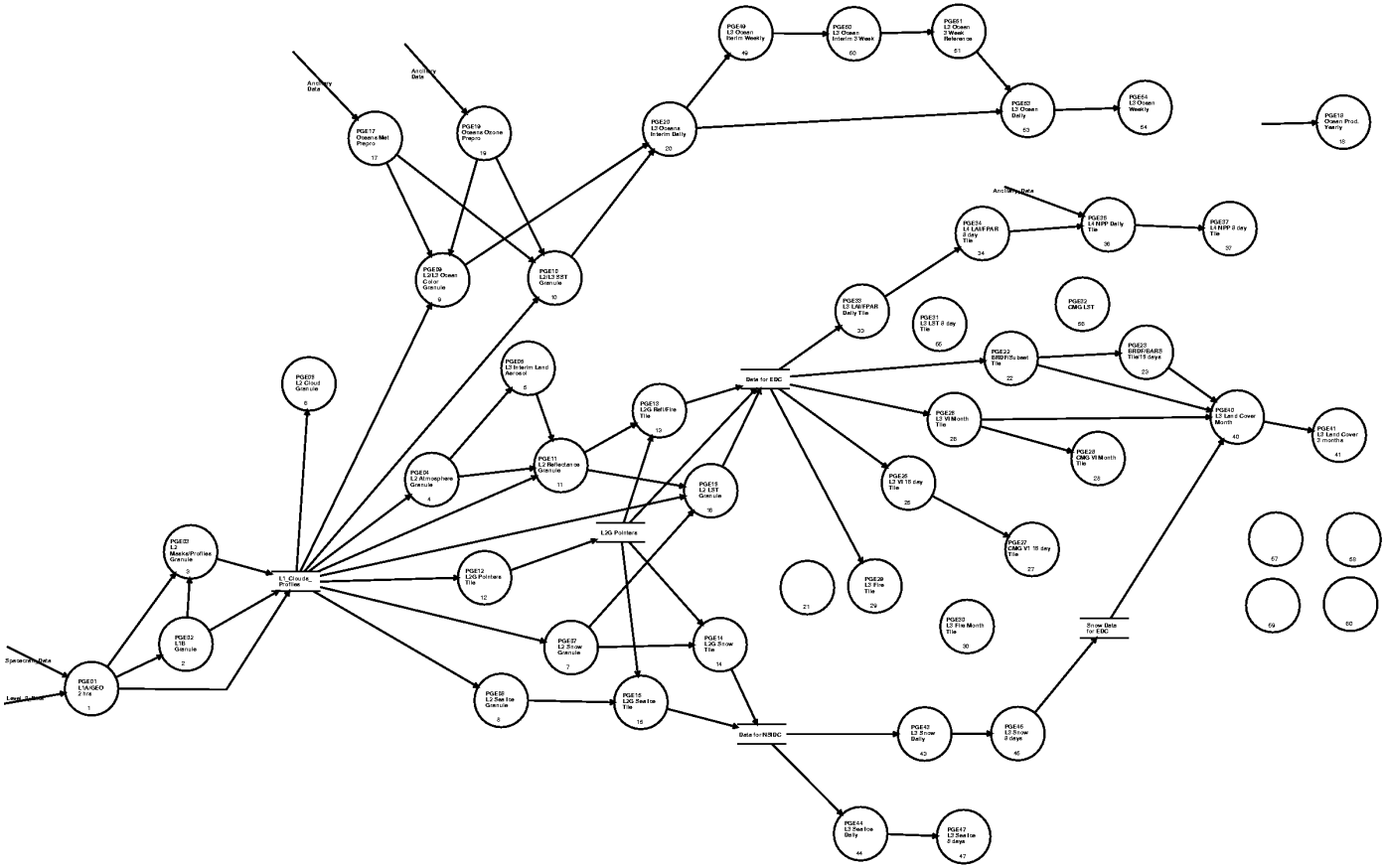
Provenance Information

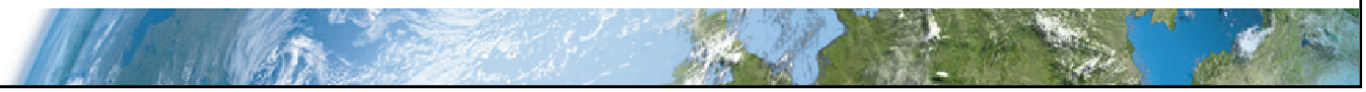
***Capture*, *archive*, and *distribute* provenance information, for example**

- The source of all externally supplied data files
- The source of the algorithms used to transform the data within the system
- The Algorithm design documents
- A complete description of the processing environment
- A complete description of the processing framework
- A record of each job's execution



Data Science Workflows





More Formal Definition of Provenance

Provenance is

information about entities, activities, and people

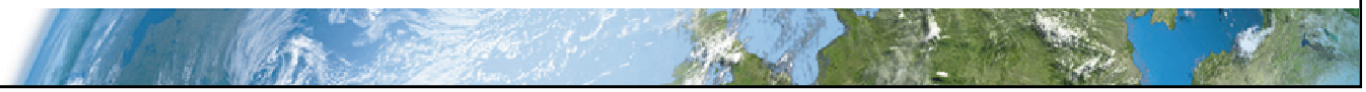
involved in

producing a piece of data or thing,

which can be used to form

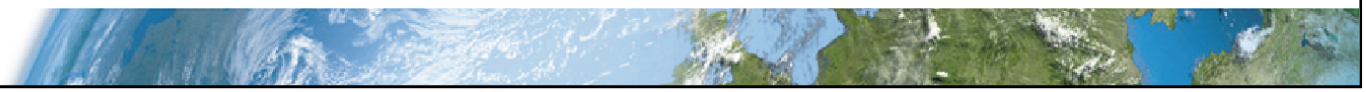
assessments about its quality, reliability or trustworthiness.

PROV W3C Working Group
https://www.w3.org/TR/prov-overview



W3C Specification „PROV“

- **PROV-O**, the PROV ontology, an OWL2 ontology allowing the mapping of the PROV data model to RDF
- **PROV-DM**, the PROV data model for provenance
- **PROV-N**, a notation for provenance aimed at human consumption
- **PROV-CONSTRAINTS**, a set of constraints applying to the PROV data model
- **PROV-XML**, an XML schema for the PROV data model
- **PROV-AQ**, mechanisms for accessing and querying provenance
- **PROV-DICTIONARY** introduces a specific type of collection, consisting of key-entity pairs
- **PROV-DC** provides a mapping between PROV-O and Dublin Core Terms
- **PROV-SEM**, a declarative specification in terms of first-order logic of the PROV data model
- **PROV-LINKS** introduces a mechanism to link across bundles



PROV Elements

Entities

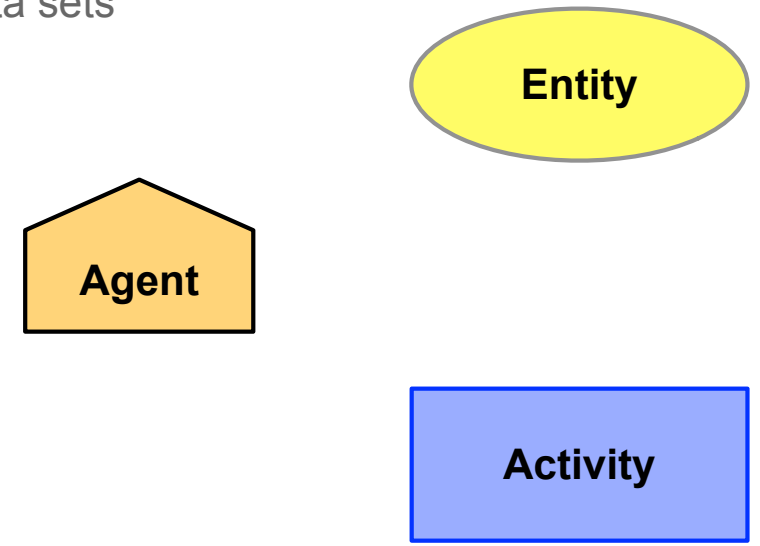
- Physical, digital, conceptual, or other kinds of things
- For example, documents, web sites, graphics, or data sets

Activities

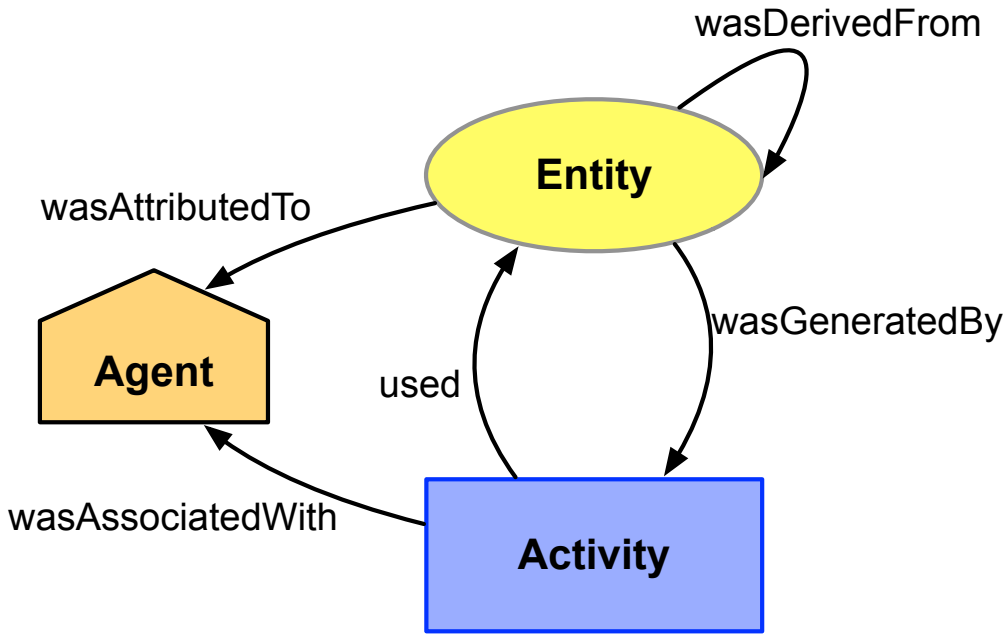
- Activities *generate* new entities or make *use* of existing entities
- Activities could be actions or processes

Agents

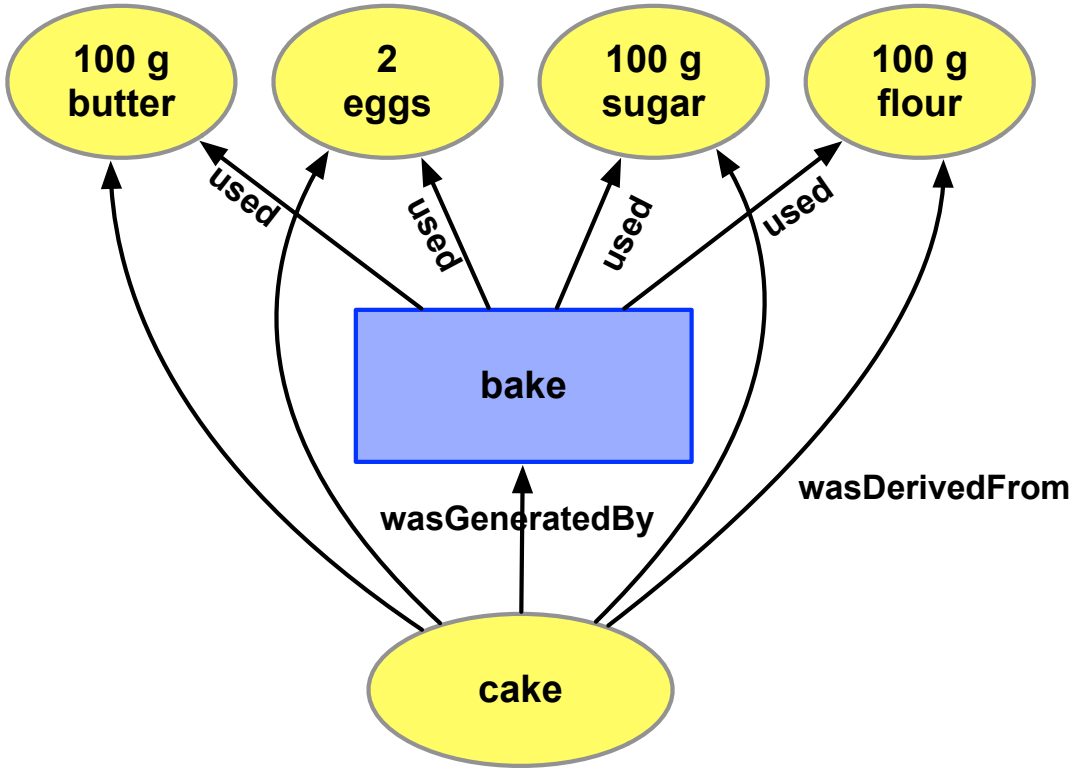
- Agents takes a role in an activity and have the responsibility for the activity
- For example, persons, pieces of software, or organizations



PROV Relations



Baking a Cake



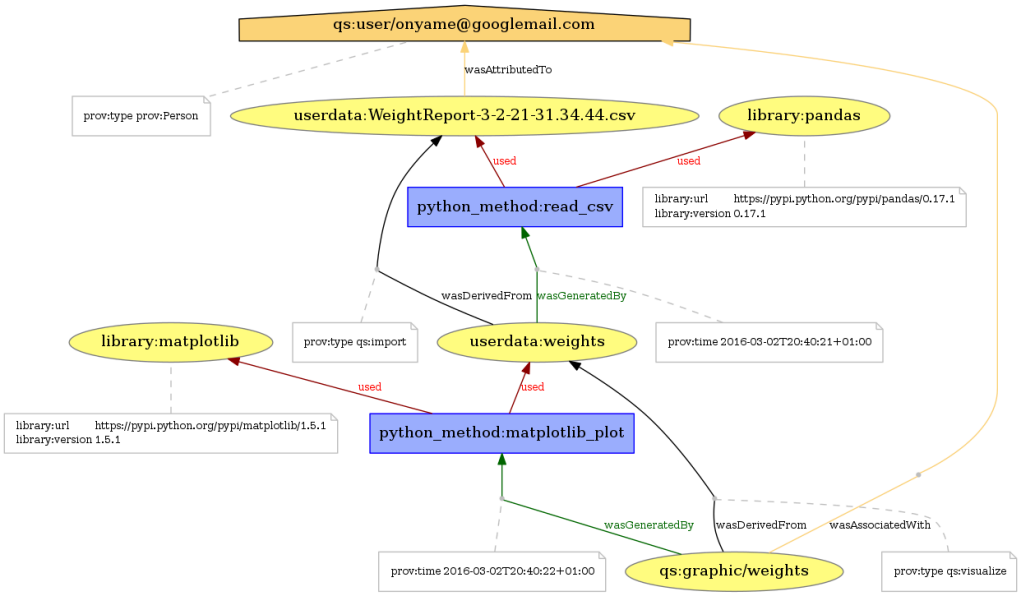
PROV Notations and Representations

Textual Representations

- Formats: *PROV-N, JSON, Turtle, XML, …*

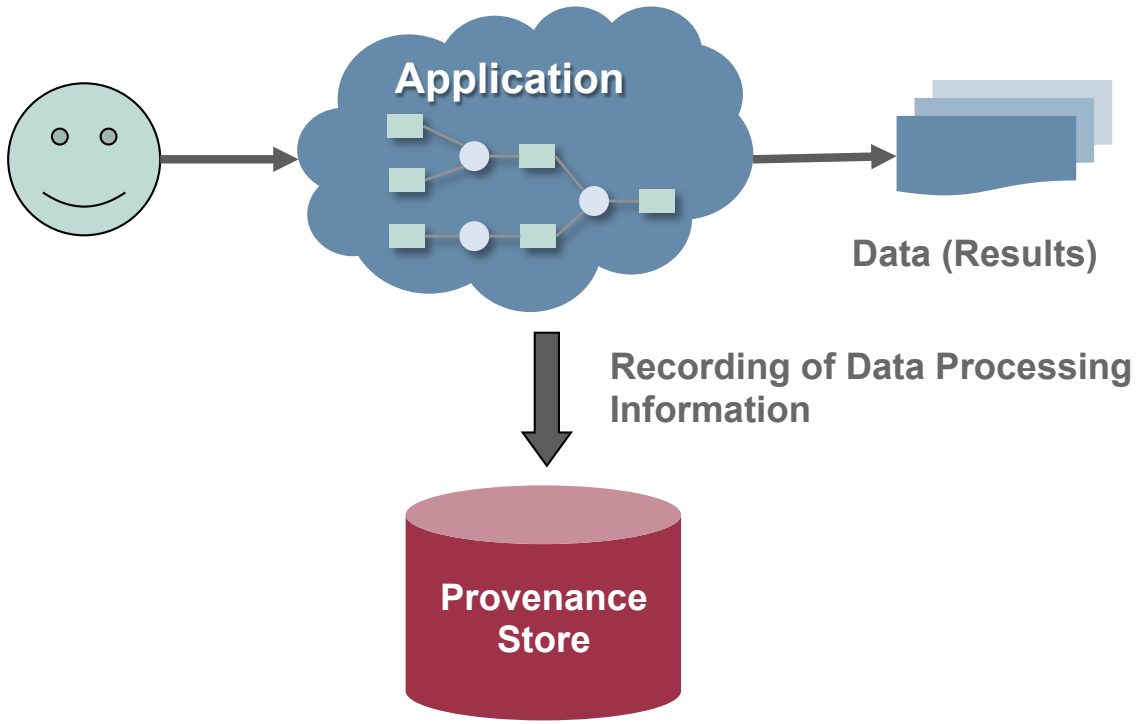
```
document
  prefix userdata http://software.dlr.de/qs/userdata/
. . .
  wasDerivedFrom(userdata:weights,
userdata:WeightReport.csv,
  wasDerivedFrom(qs:graphic/weights, userdata:weights,
  wasAssociatedWith(qs:graphic/weights, qs:user/
onyame@gmail.com, -)
  used(python_method:read_csv, library:pandas, -)
  used(python_method:matplotlib_plot, userdata:weights, -)
  used(python_method:matplotlib_plot, library:matplotlib, -)
  used(python_method:read_csv, userdata:WeightReport.csv, -)
  wasAttributedTo(userdata:WeightReport.csv, qs:user/
onyame@gmail.com)
  agent(qs:user/onyame@gmail.com, [prov:type="prov:Person"])
  entity(library:pandas, [library:version="0.17.1"])
  entity(userdata:WeightReport.csv)
  entity(userdata:weights)
. . .
endDocument
```

Visualizations



Storing and Retrieving Provenance

Provenance Architecture



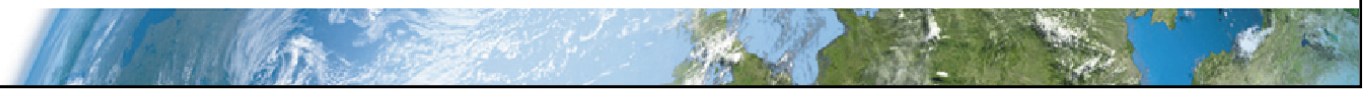
Storing and Retrieving Provenance

Some Storage Technologies

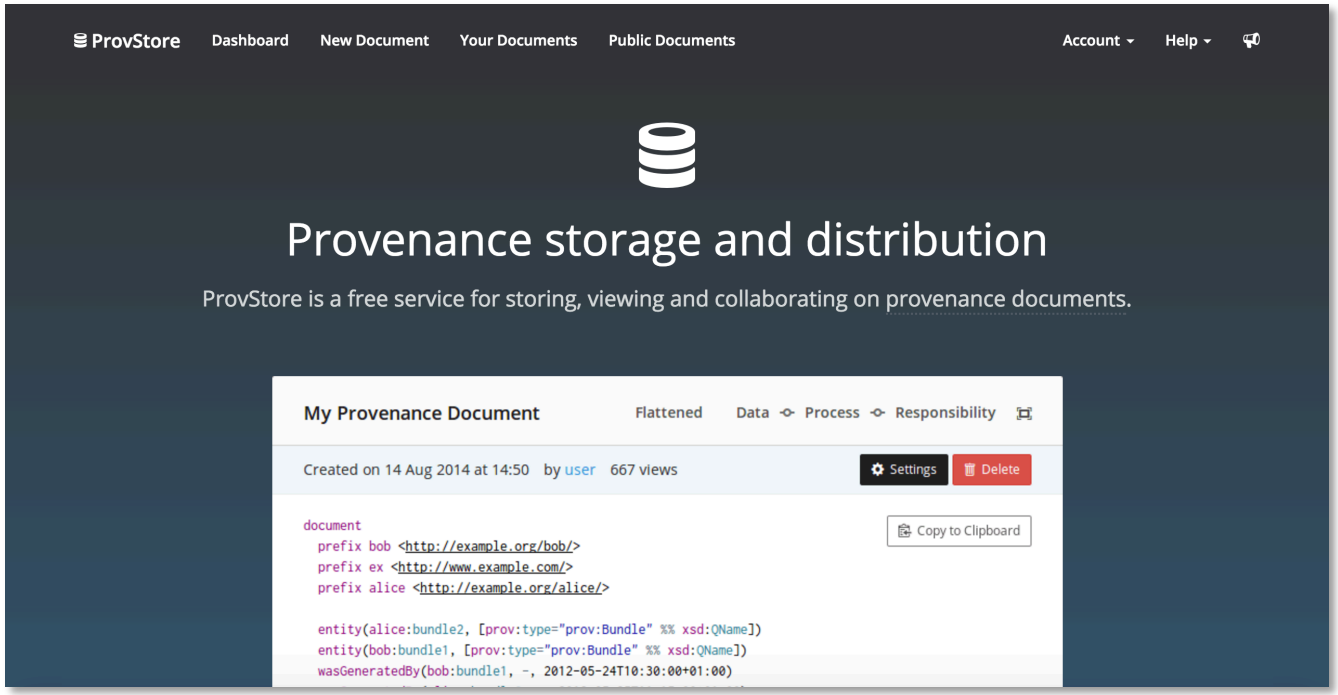
- Relational databases and SQL
- XML and Xpath
- RDF and SPARQL
- Graph databases and Gremlin/Cypher

Services

- REST APIs
- PROVSTORE



ProvStore
University of Southampton



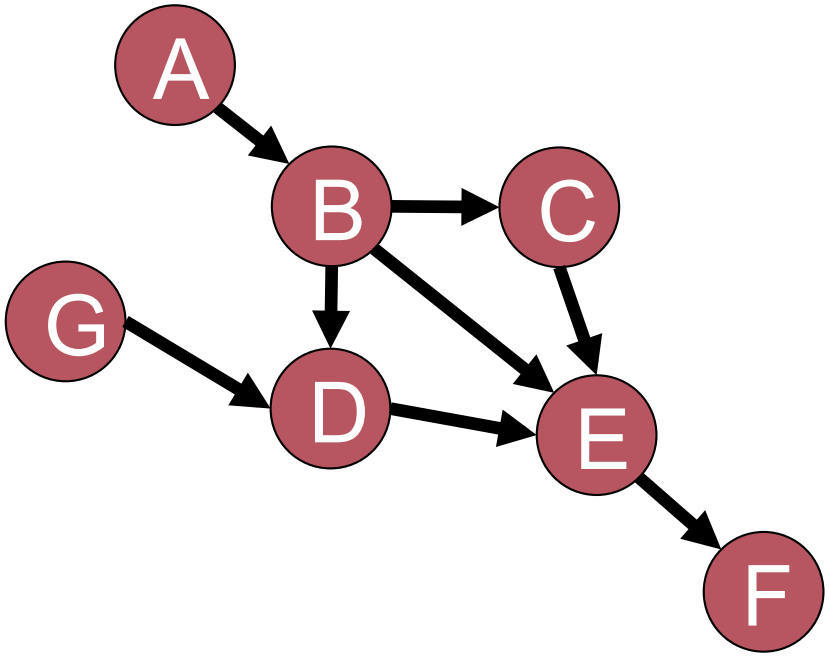
- RESTful web service
- storage and access of provenance documents
- Public and private documents
- Conversion to various text formats
- Simple visualizations
- APIs
  - Python
  - jQuery

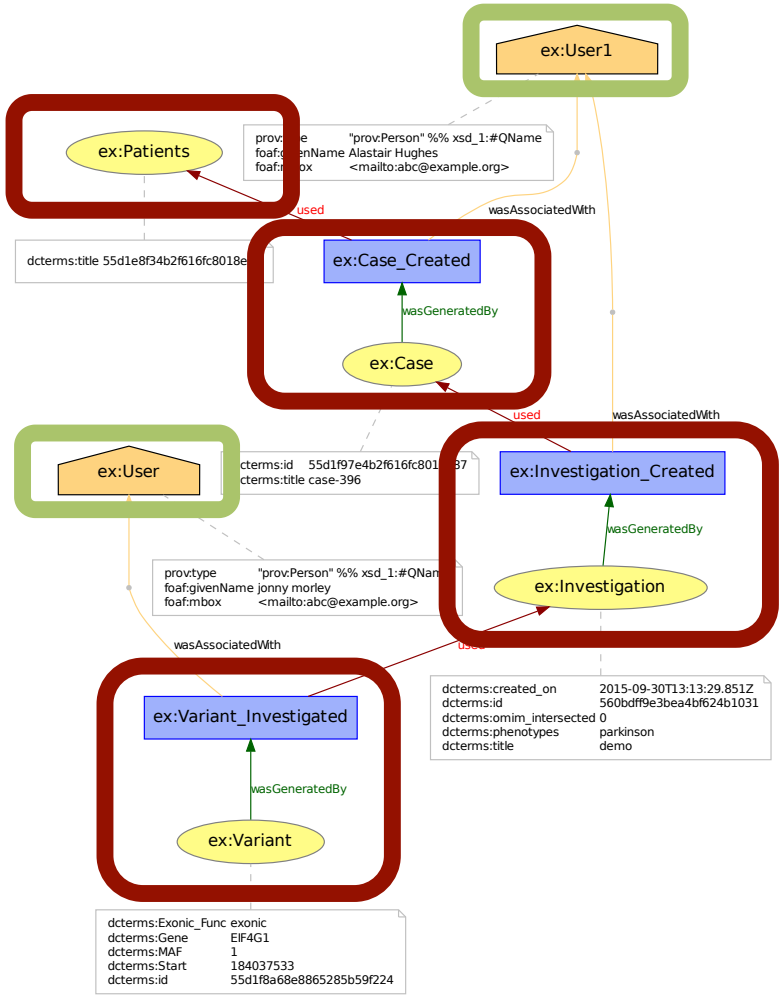
https://provenance.ecs.soton.ac.uk/store/

DLR

Graphs

Provenance is a *Directed Acyclic Graph (DAG)*





Graph Databases

Naturally, graph databases are a good technology for storing (Provenance) graphs

Many graph databases are available

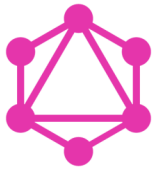
- Neo4j
- Titan
- ArangoDB
- ...

Query languages

- Cypher
- Gremlin (TinkerPop)
- GraphQL

neo4j

ArangoDB



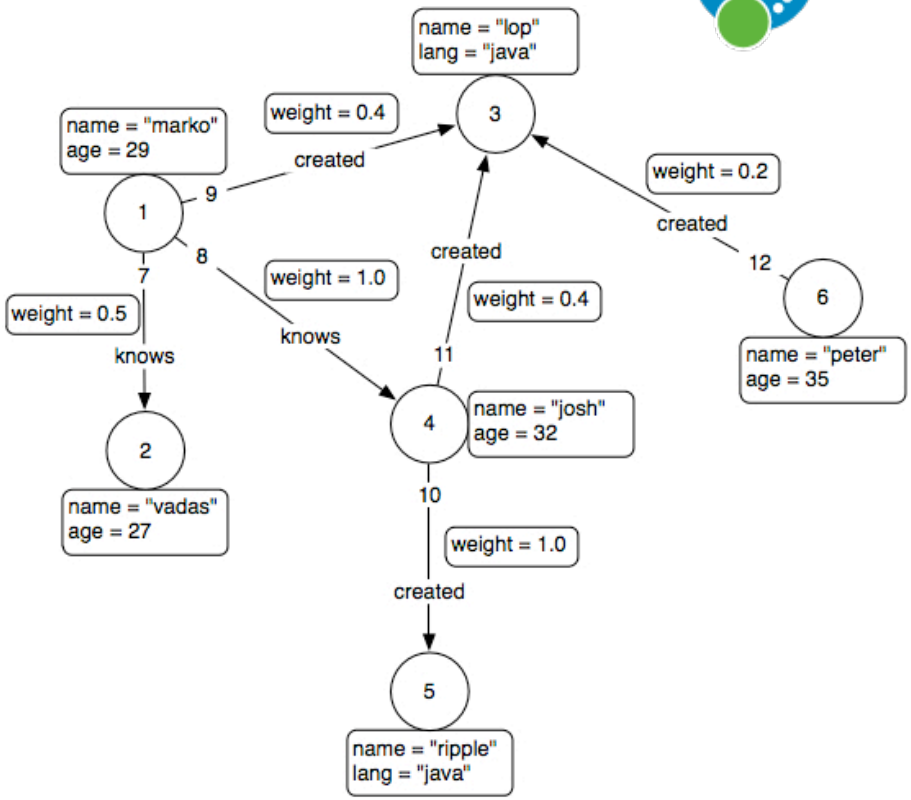
Apache TinkerPop

Gremlin
$G = (V, E)$

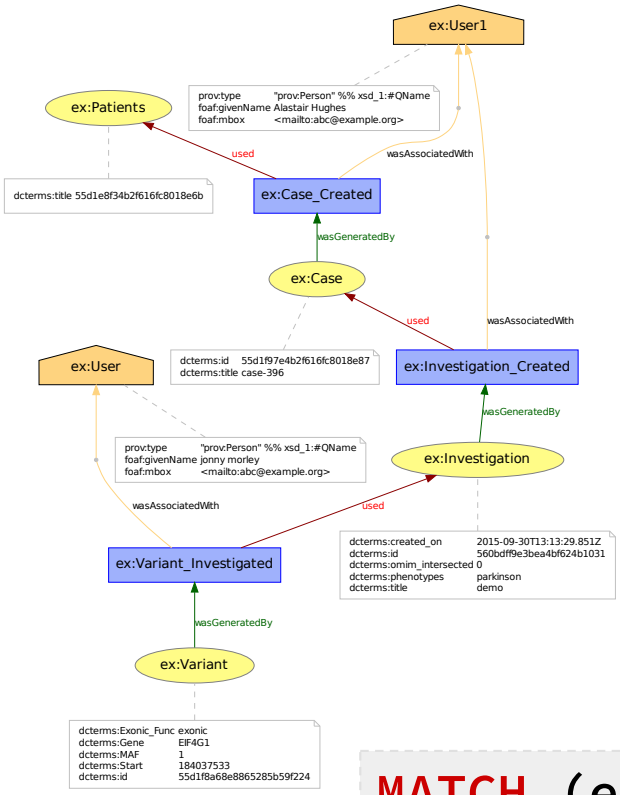
Neo4j

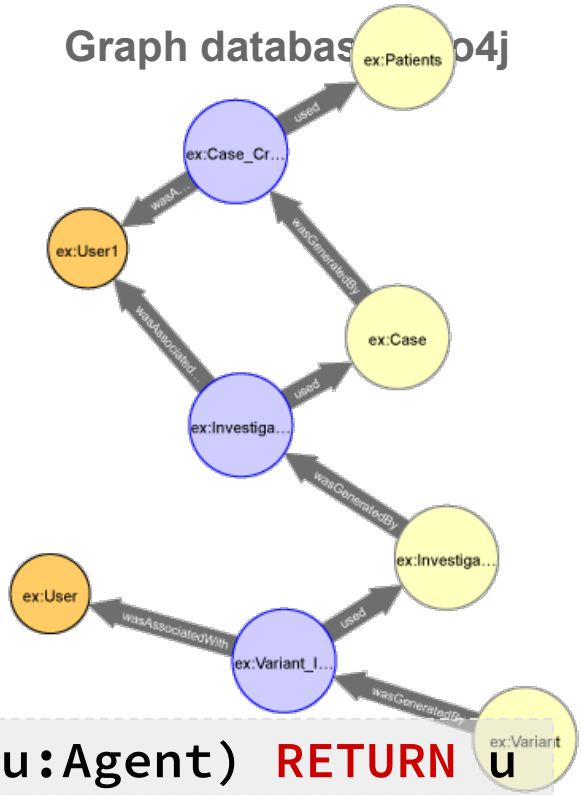
- Open-Source
- Implemented in Java
- Stores *property graphs* (key-value-based, directed)

`http://neo4j.com`

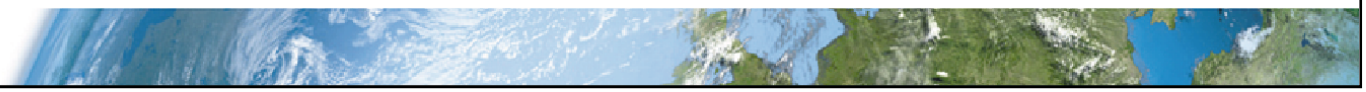


Storing Provenance in Graph Database

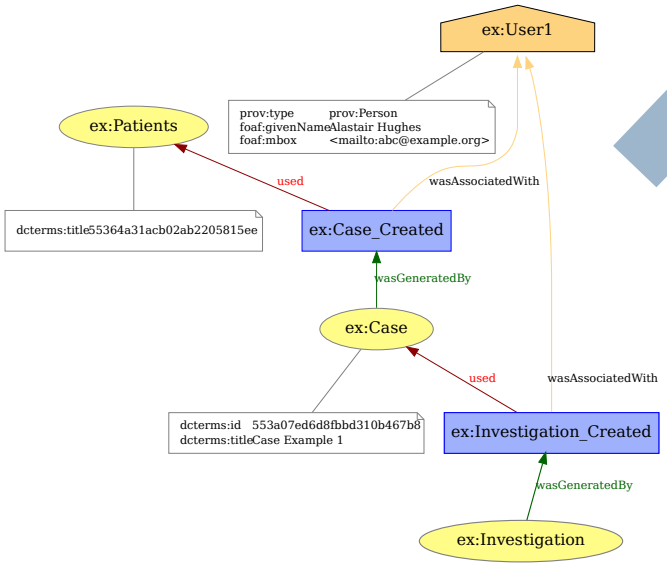





```
MATCH (e:Entity)-[*]-(u:Agent) RETURN u
```



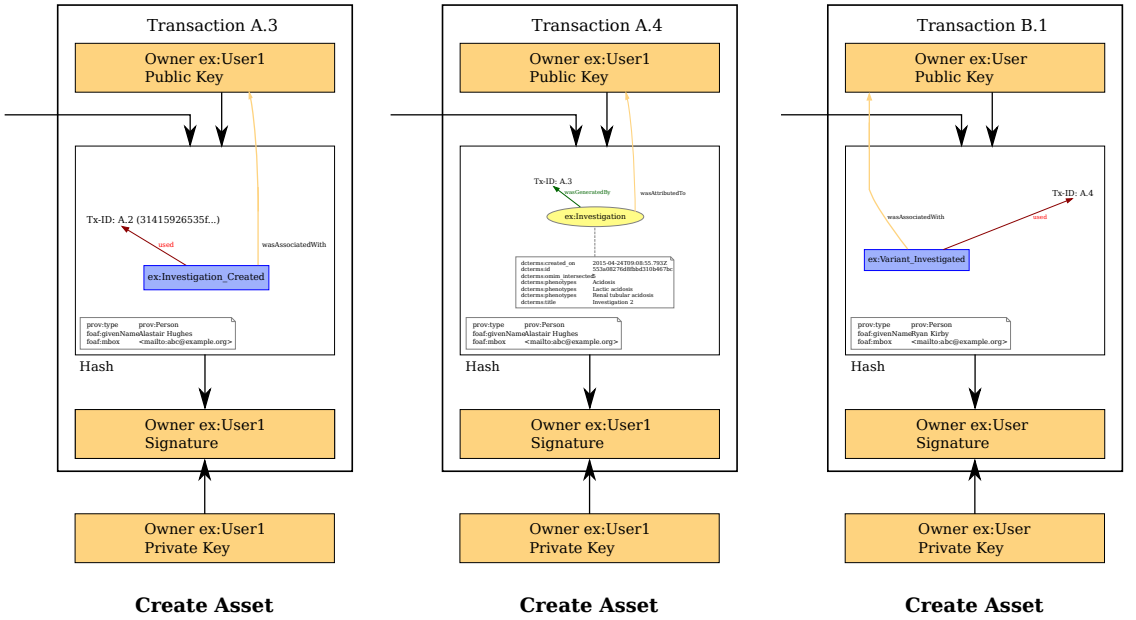
Trusted Provenance: Storing Provenance in a Blockchain



Document A

PROV2BIGCHAINDB

`https://github.com/DLR-SC/prov2bigchaindb`



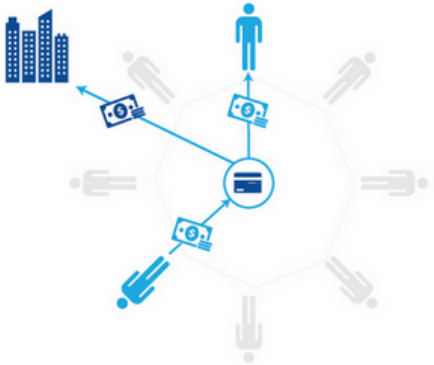
Blockchain

Combination of multiple techniques

- Peer-to-peer network
- Public/Private key signing
- Time-stamping
- Proof-of-Work
- Merkle-Trees

Proposed solutions to

- The double-spending problem
- The byzantine generals problem
- Tamper-resistant distributed database

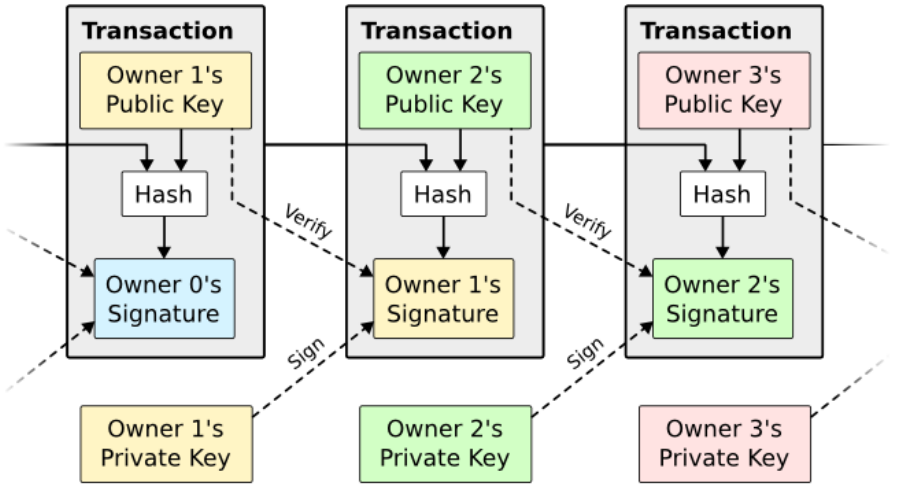


Current payment systems require third-party intermediaries that often charge high processing fees ...

... but machine-to-machine payment using the Bitcoin protocol could allow for direct payment between individuals, as well as support micropayments.

Blockchain Transactions

- Linked by hash of current and preceding transactions
- *Bitcoin*: Transfers amount of BTC
- Public/private key signing
- All transactions are broadcasted across the network



Document-based Storage of PROV Documents

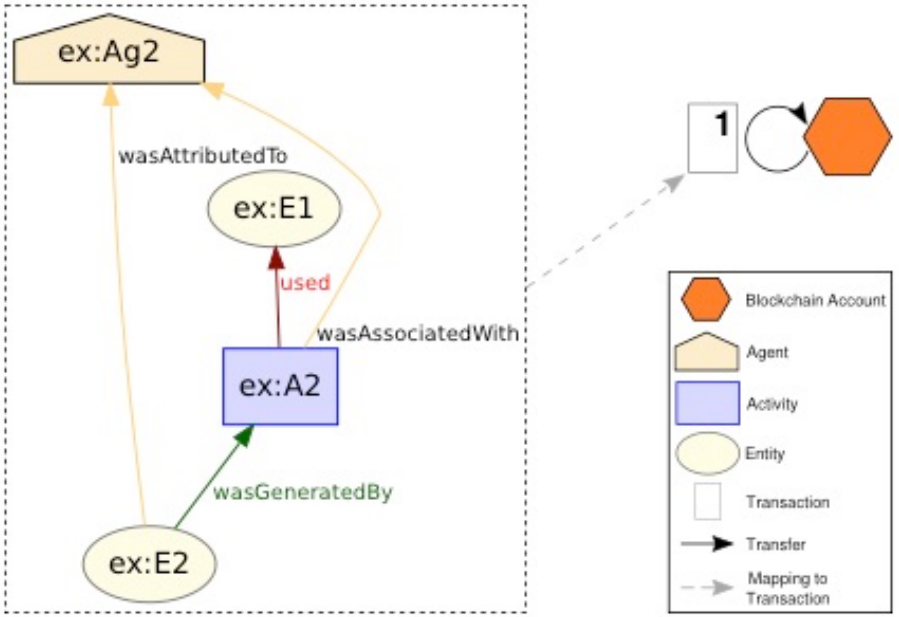
- Only one user/address on the blockchain
- Provenance is stored as one valid document

Pros

- Less complex
- Ownership restricted to one participant
- Easy to query
- Less costly, if less data is added

Cons

- Single point of failure
- Less tamper-resistant
- No chaining of transactions
- Huge amount of data in transactions



Role-based Storage of PROV Documents

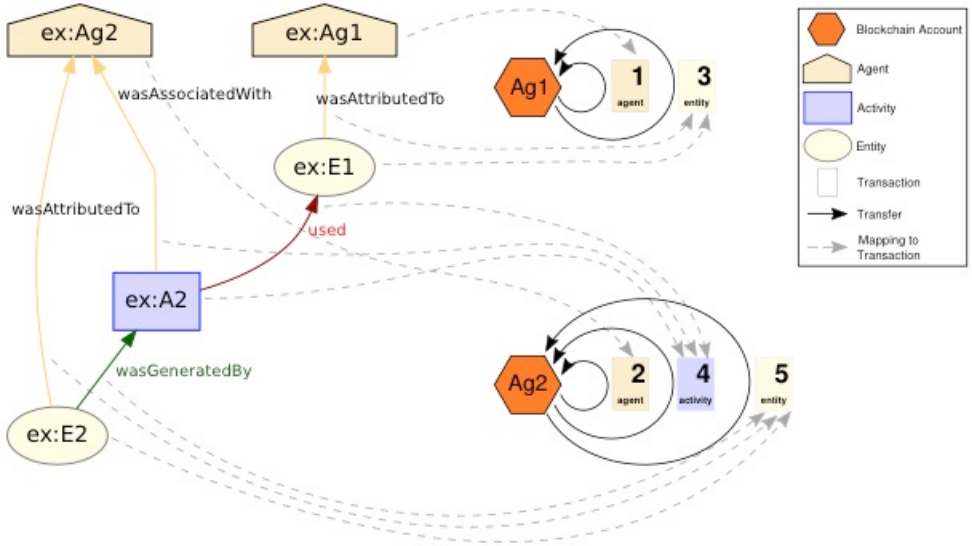
- Every agents is a blockchain user/address
- Generates transactions for its entities and activities
- Relations modeled with references to other transactions

Pros

- Close to typical process structures
- Implicit ownership and responsibility

Cons

- Agent needs to know relevant transactions for references
- Difficult to query, if no ownership transfer is used



Graph-based Storage of PROV Documents

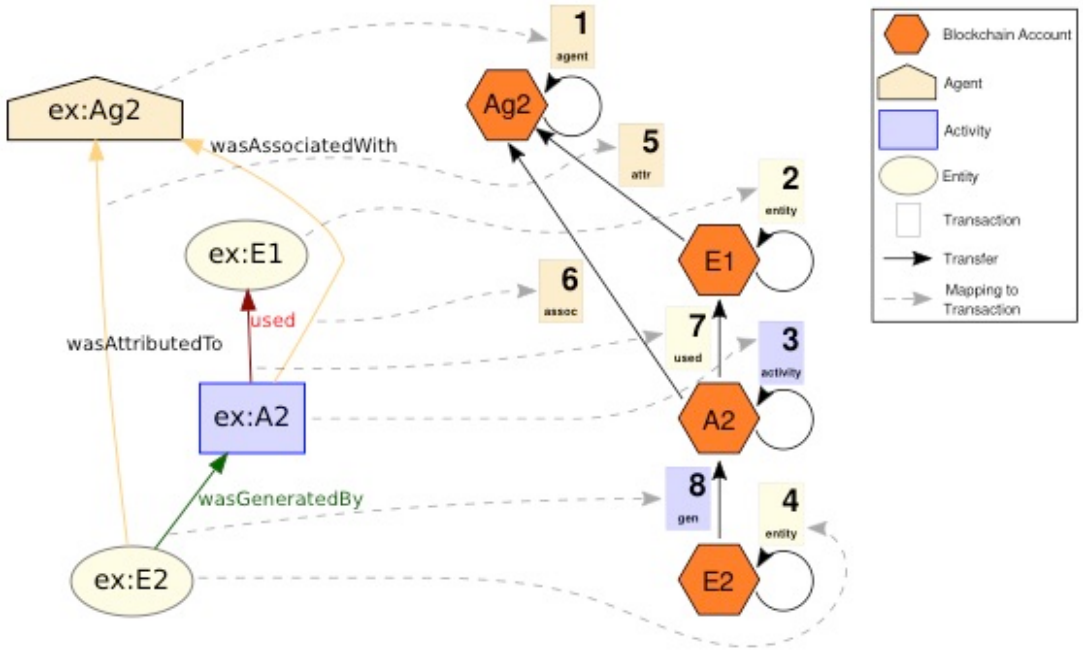
- All PROV relations are modeled as ownership transfer
- All agents, activity and entities are actual blockchain user/addresses

Pros

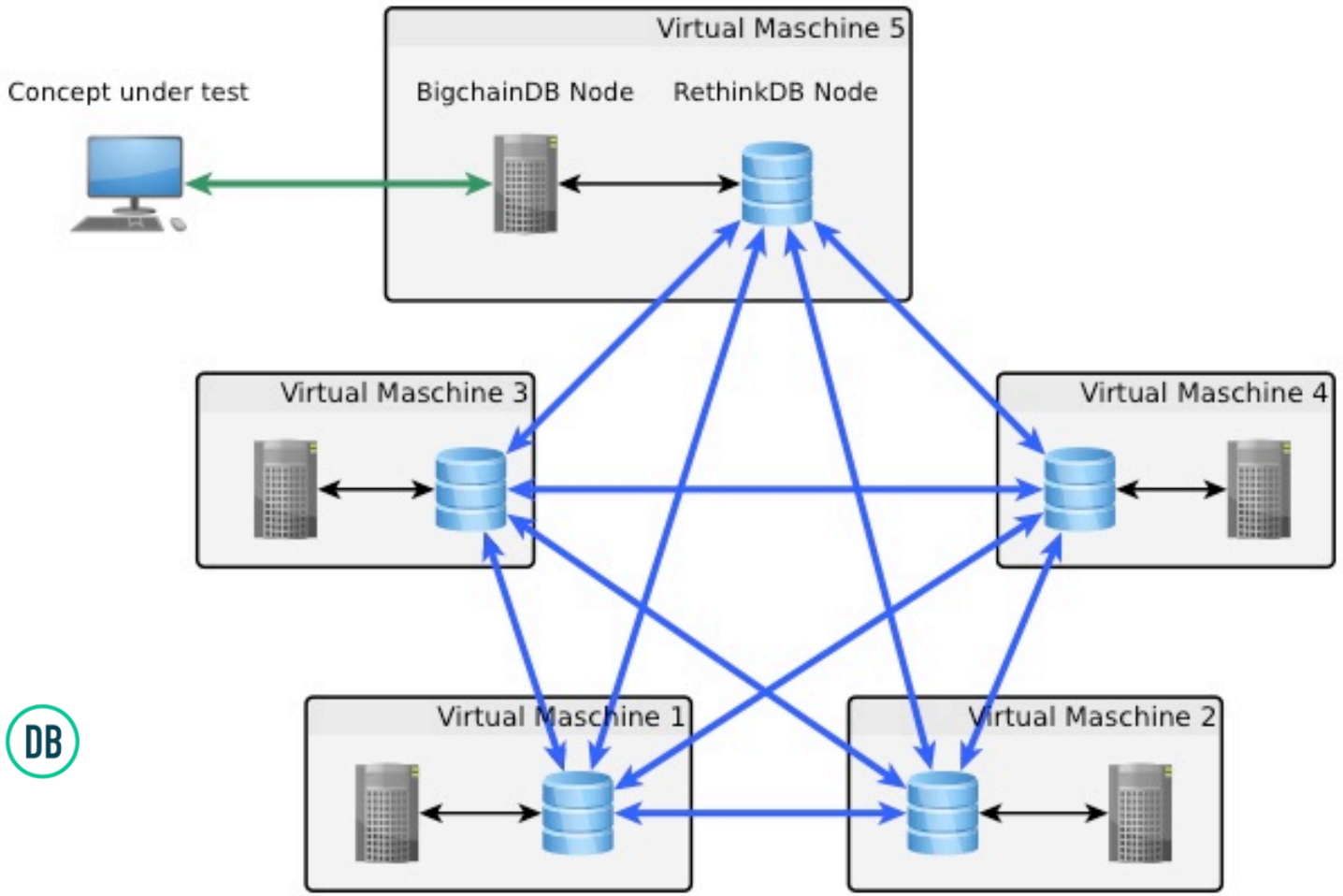
- Mapping close to PROV model
- Small amount of data per transaction
- Strong tamper-resistant due to:
  - Multiple owner
  - Large amount of transactions

Cons

- Complex implementation
- High costs due to many transactions
- Very slow in querying, if traversal of transactions is needed



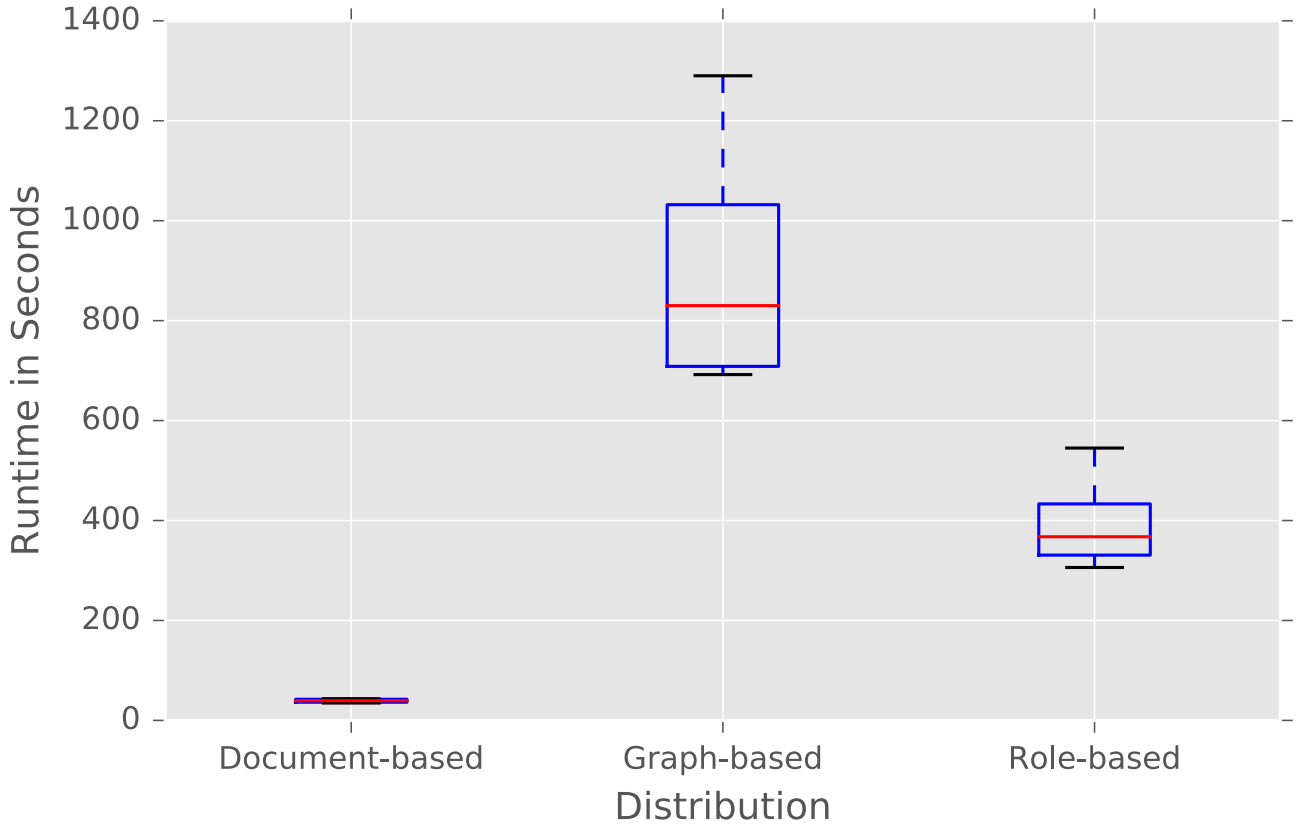
Test Setup

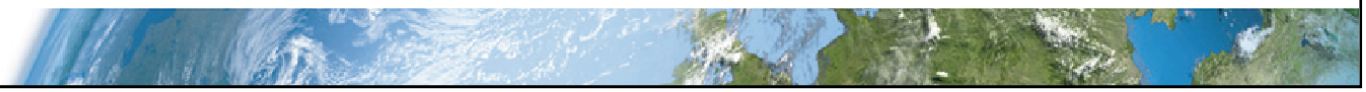


BIGCHAIN DB

DLR

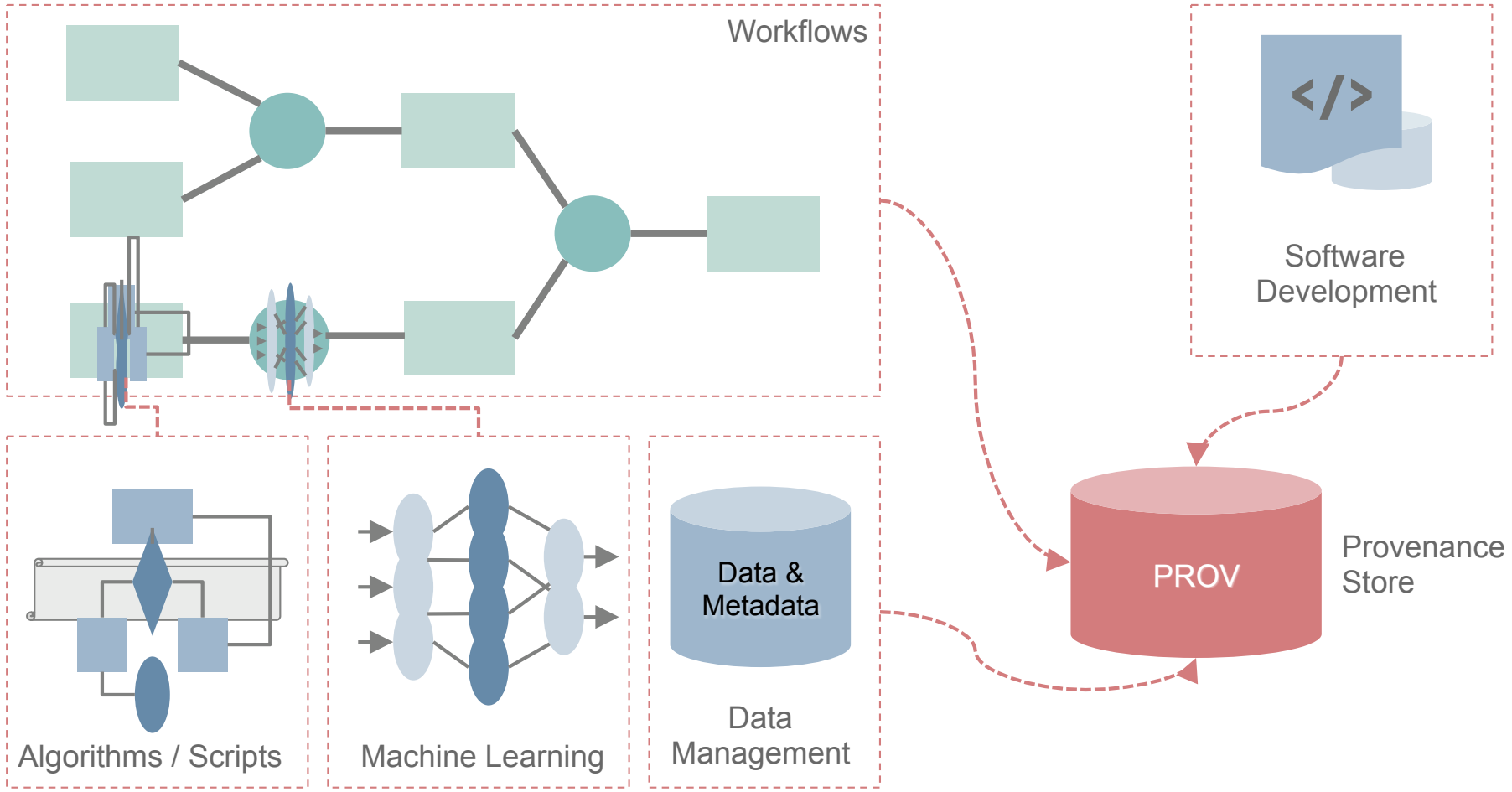
Performance Comparison





Gathering Provenance

DLR


Workflows
Software Development
Algorithms / Scripts
Machine Learning
Data & Metadata
Data Management
PROV
Provenance Store

Gather or Generate Provenance

Depends on your application (tools, languages, etc.)

- Generation at *run-time*, *compile-time*, or *retrospectively*

Runtime

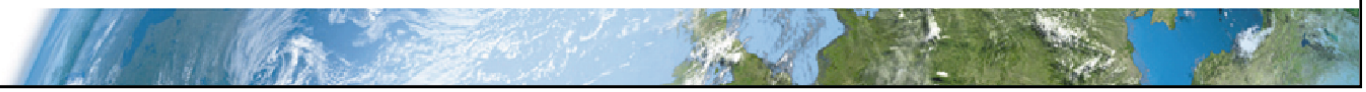
- Instrumentation of the application
- Cumbersome from software engineering perspective
- Combined with logging or with aspect-oriented approaches

Compile time

- Based on static code analysis (dependency analysis, program slicing, etc.)

Retrospectively

- Reconstructed from files or filesystem metadata



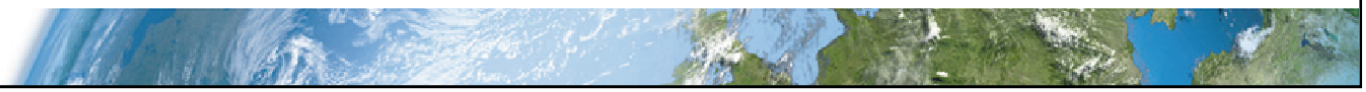
Tools and Libraries for Generating Provenance

Libraries for Python

- ProvPy
- ProvNeo4j

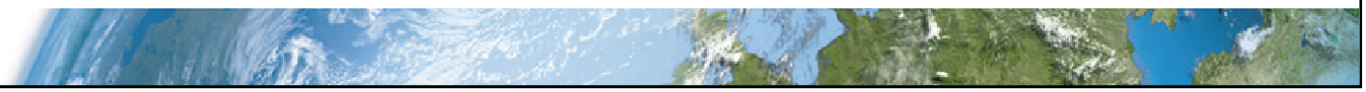
Other Tools

- NoWorkflow
- Git2PROV

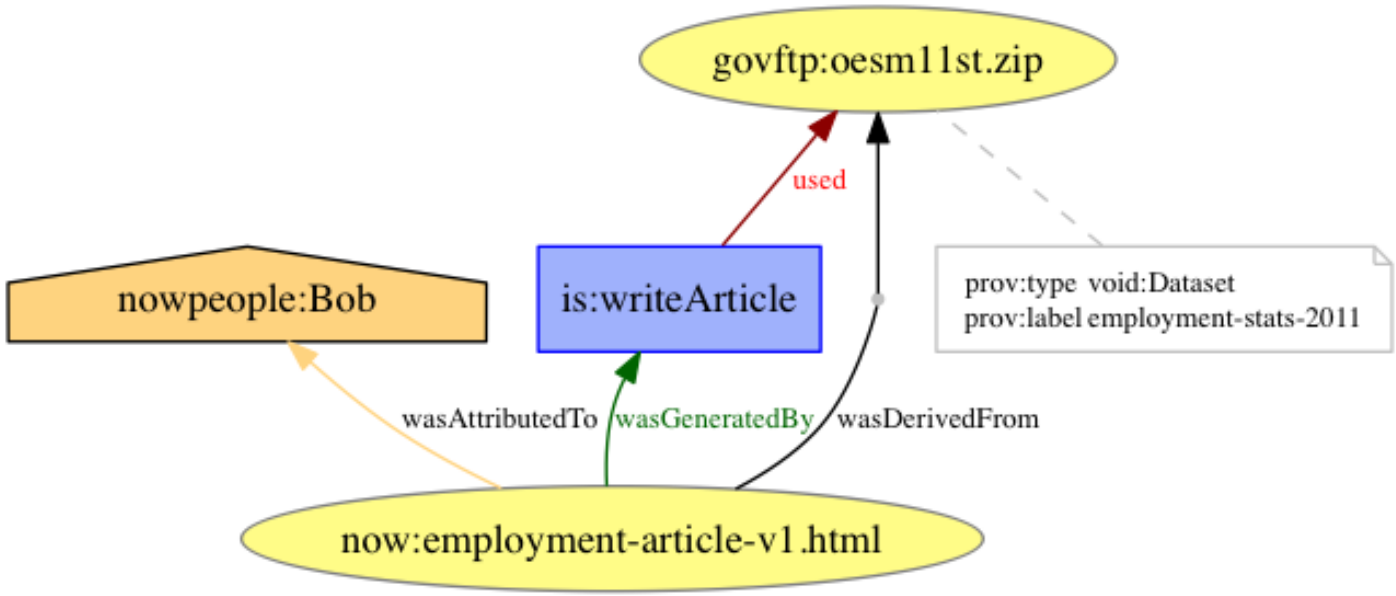


Python Library ProvPy (PROV)
https://github.com/trungdong/prov

```
from prov.model import ProvDocument
# Create a new provenance document
d1 = ProvDocument()
# Entity: now:employment-article-v1.html
e1 = d1.entity('now:employment-article-v1.html')
# Agent: nowpeople:Bob
d1.agent('nowpeople:Bob')
# Attributing the article to the agent
d1.wasAttributedTo(e1, 'nowpeople:Bob')
d1.entity('govftp:oesm11st.zip',
          {'prov:label': 'employment-stats-2011',
           'prov:type': 'void:Dataset'})
d1.wasDerivedFrom('now:employment-article-v1.html',
                  'govftp:oesm11st.zip')
# Adding an activity
d1.activity('is:writeArticle')
d1.used('is:writeArticle', 'govftp:oesm11st.zip')
d1.wasGeneratedBy('now:employment-article-v1.html', 'is:writeArticle')
```

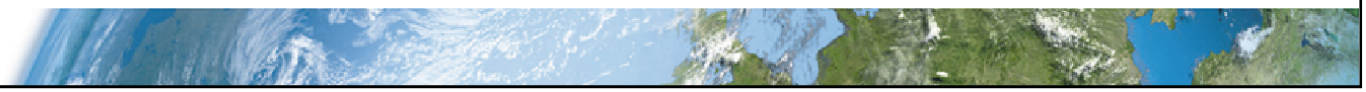


Python Library ProvPy (PROV)
https://github.com/trungdong/prov



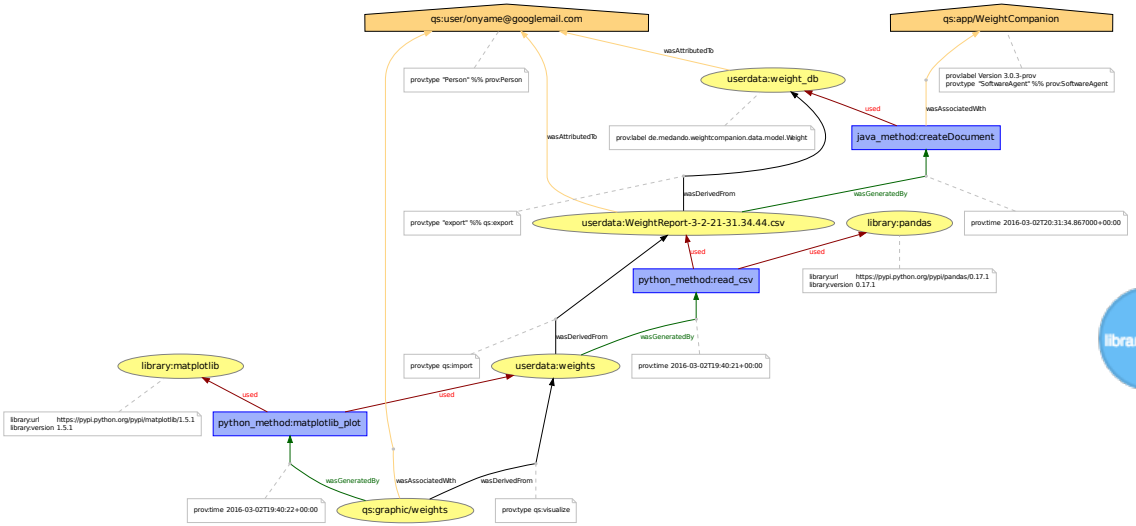
PROVNEO4J – Storing PROV Documents in Neo4j
https://github.com/DLR-SC/provneo4j

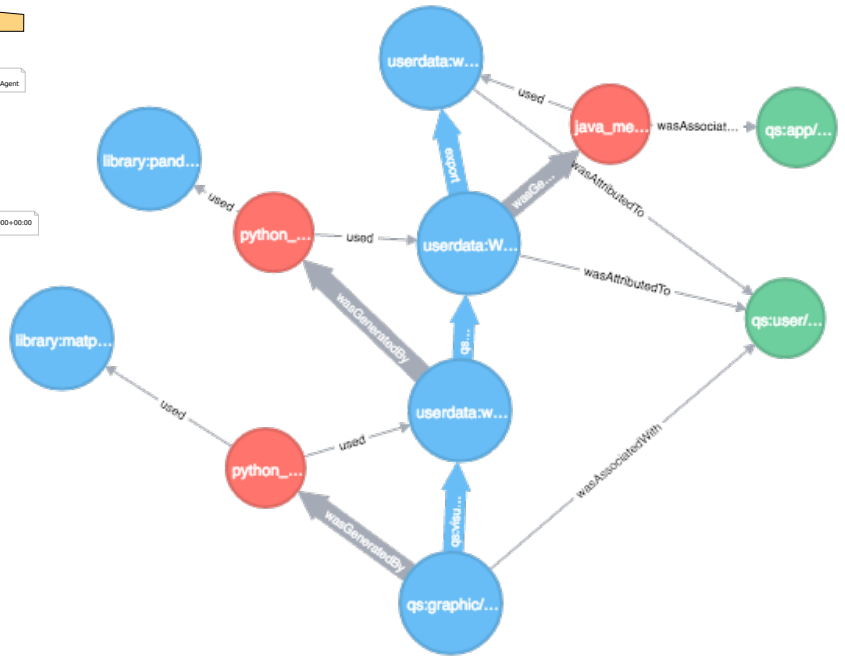
```
import provneo4j.api

provneo4j_api = provneo4j.api.Api(
        base_url="http://localhost:7474/db/data",
        username="neo4j", password="python")

provneo4j_api.document.create(prov_doc, name="MyProv")
```



PROVNEO4J – Storing PROV Documents in Neo4j

https://github.com/DLR-SC/provneo4j

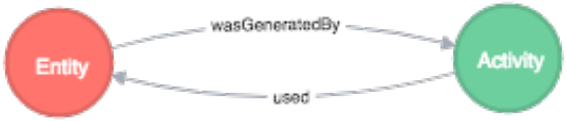




Provenance Instrumentation of TensorFlow

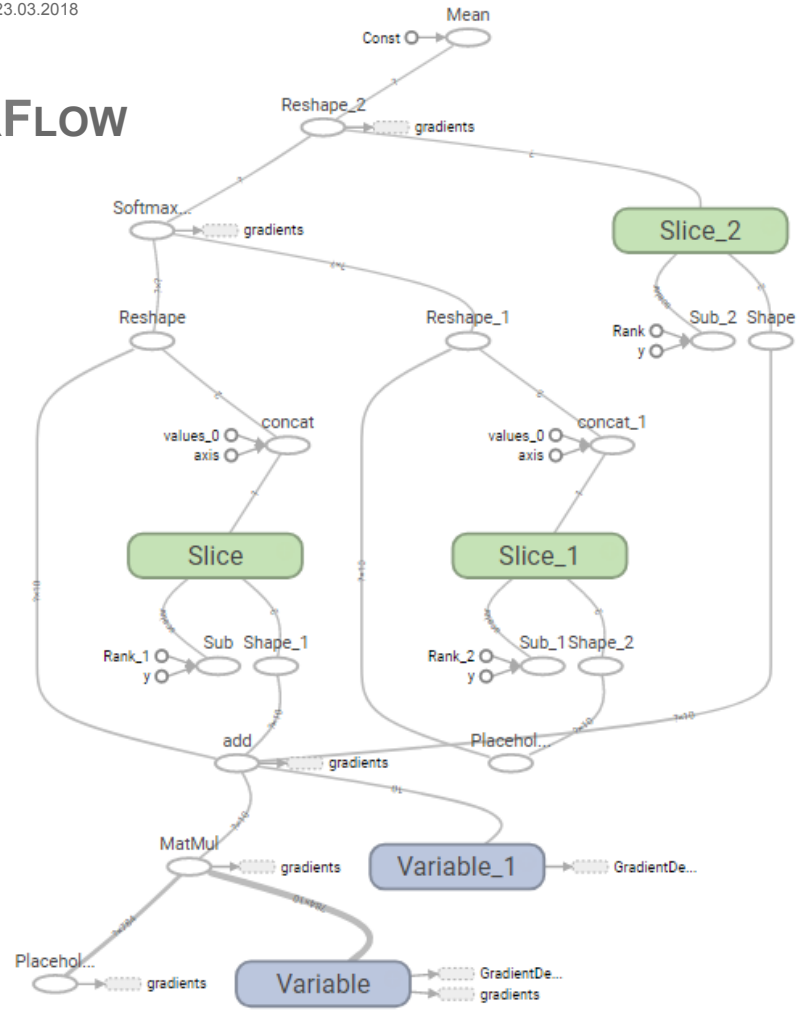
Provenance of TensorFlow workflows

- Tensor → PROV Entity
- Operations → PROV Activity



Example: MNIST with 400 training iterations

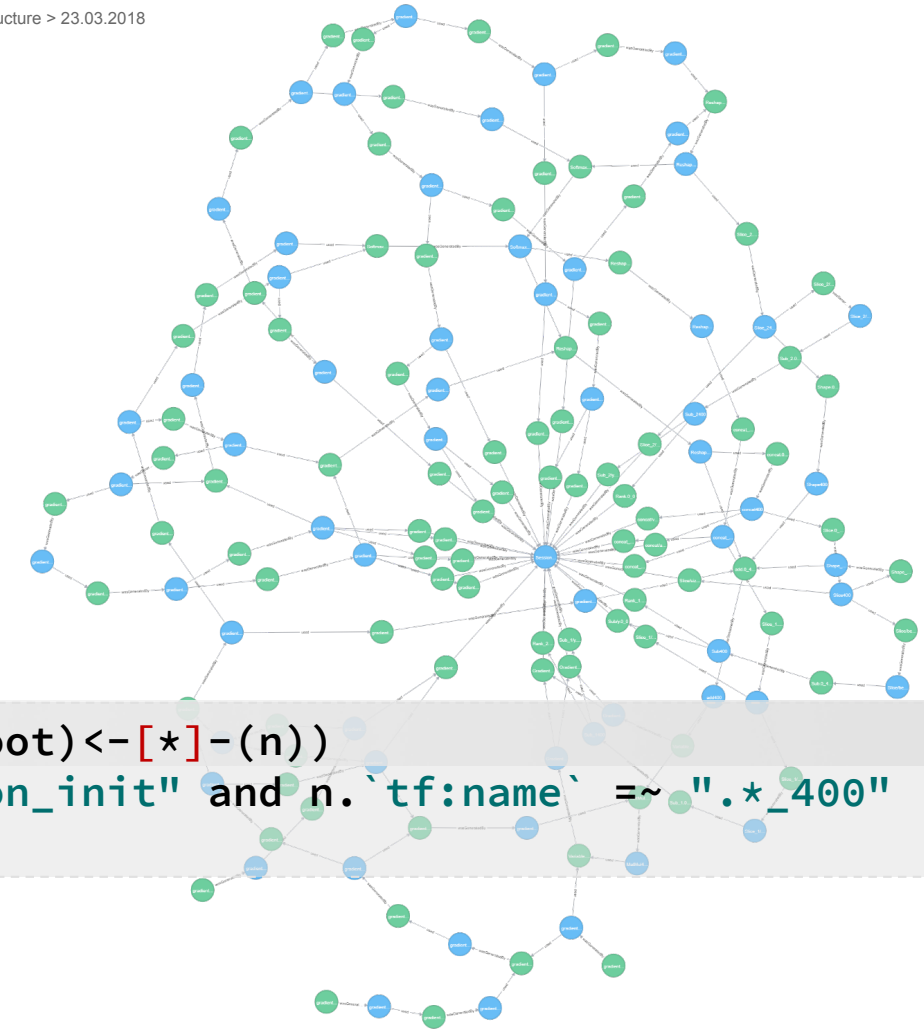
- 64581 database nodes
- 33549 Entities
- 31032 Activities



Provenance Instrumentation of TensorFlow

Example Query

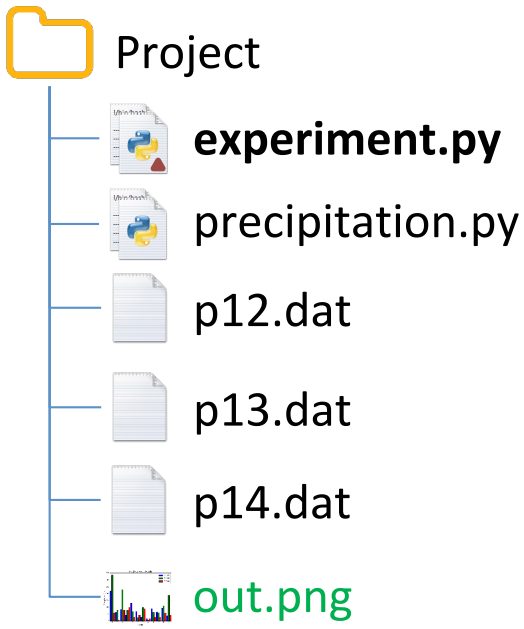
- Shortest paths from all tensors in 400. iteration to *init* operation

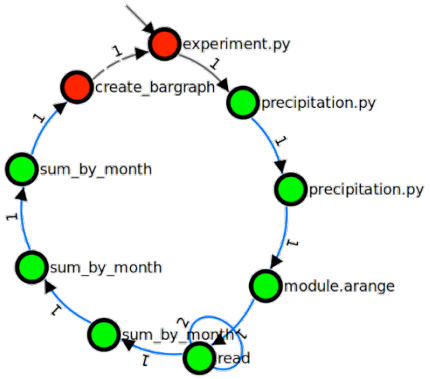


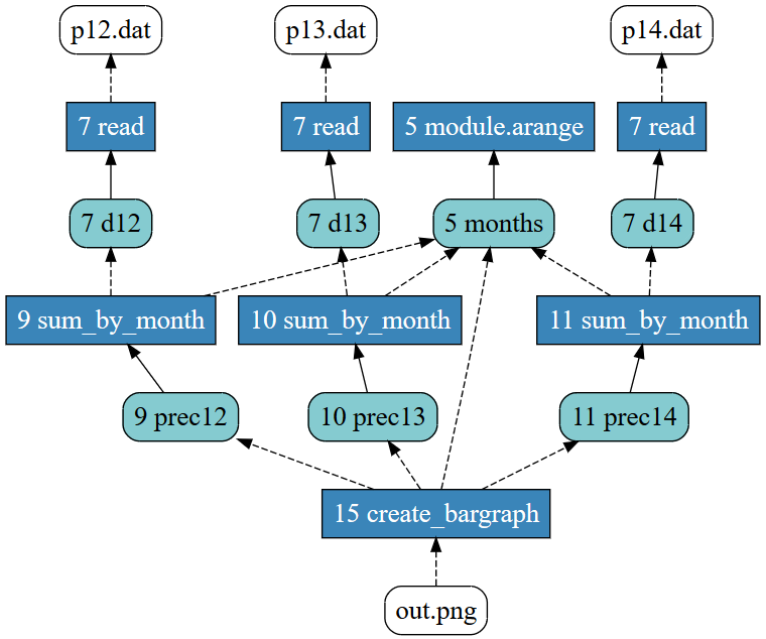
```
MATCH path=allShortestPaths((root)<-[*]-(n))
WHERE root.`tf:type`="tf:Session_init" and n.`tf:name` =~ ".*_400"
RETURN path
```

NoWorkflow – Provenance of Scripts
https://github.com/gems-uff/noworkflow

```
$ now run -e Tracker experiment.py
```

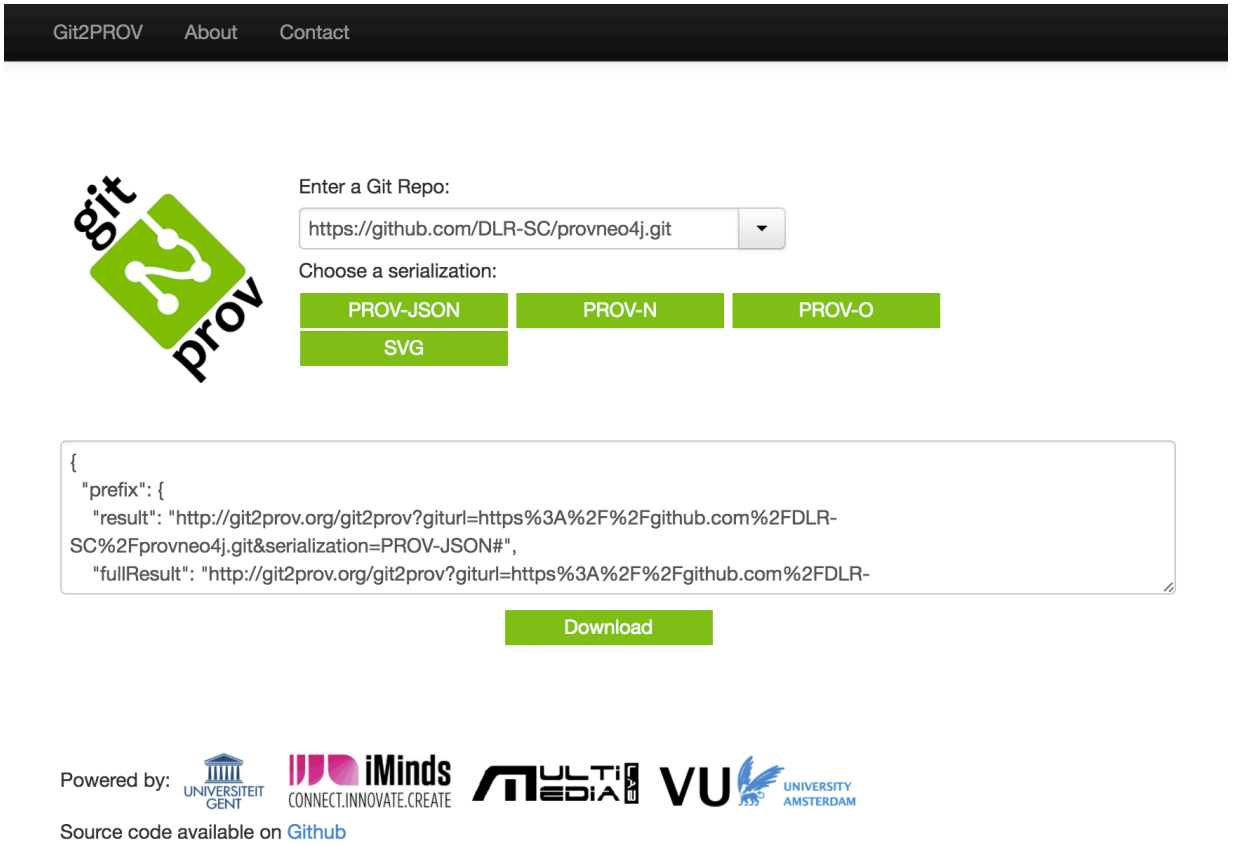






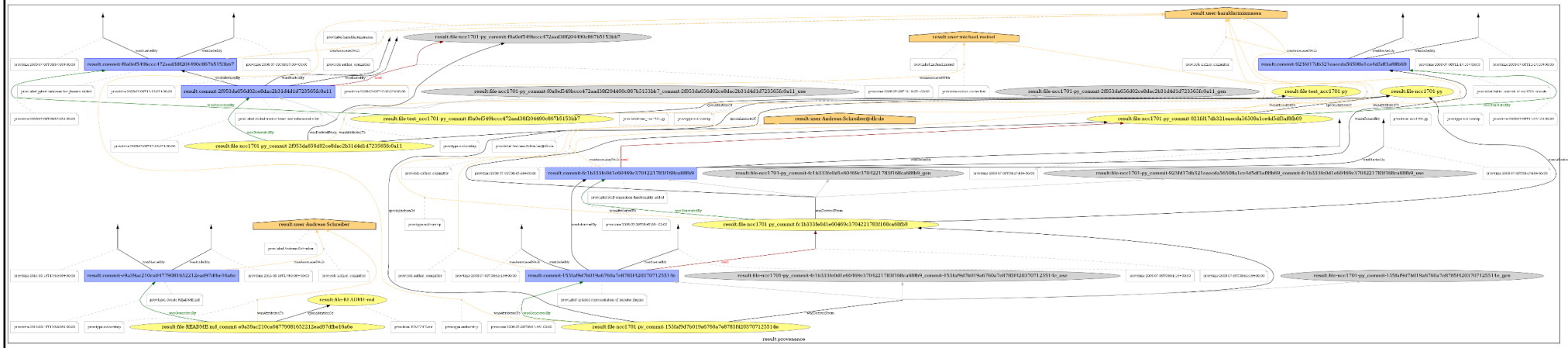
GIT2PROV
http://git2prov.org

- Generate PROV documents from git repositories



GIT2PROV Example Output

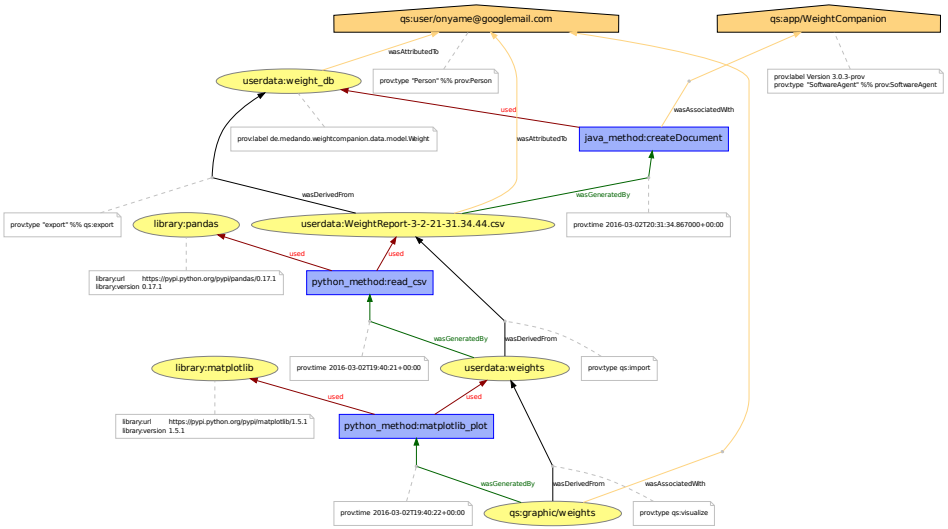
https://provenance.ecs.soton.ac.uk/store/documents/116377/

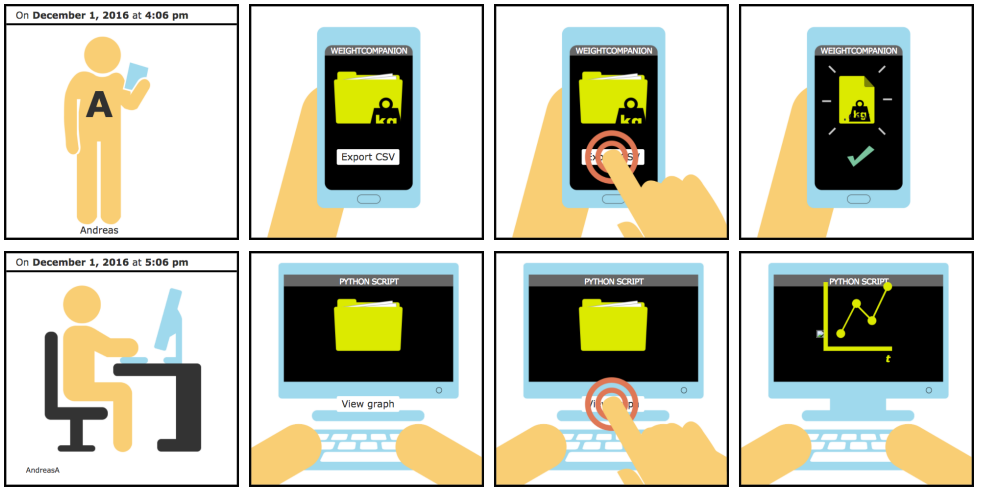


Provenance Visualization

Visualization of Provenance is an ongoing research topic

- Especially, for non-experts (“Provenance for people”)
- Example: PROV Comics





Key Messages and Summary

Recording the *Provenance* of science workflows is important

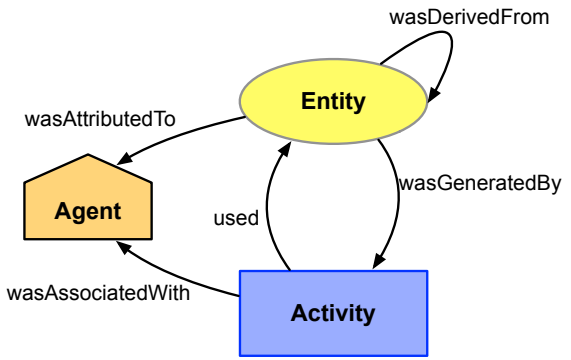
- to understand where data came from
- to reproduce data processing steps or whole workflows

Use a *standard* for Provenance

- W3C standard PROV
- Mapping to (graph) databases, allows easy querying
- A standard allow interoperability and comparison
- Storing in blockchains for increasing trust

Recording Provenance is not hard

- APIs and tools available




Thank You!
Questions?
Andreas.Schreiber@dlr.de
www.DLR.de/sc | @onyame
DLR