Querying Software Heritage

Challenges behind a query language

DIVERSE SEMINAR
12/12/2023

This presentation is based on my understanding of SWH APIs, it’s not an official presentation

Software Heritage in a nutshell

Collect **Share** **Preserve**

- Collect, preserve, and share *all* software source code
- Find and reference (SWHIDs) all software source code
- Enable analysis of all software source code

Preserving our heritage, enabling better software and better science for all.

A universal software archive, as a shared infrastructure

- One infrastructure for Cultural Heritage, Industry, Research, Public Administration
- Open, transparent, FLOSS, replicable

The largest archive ever built

Source files

17,548,413,238

Commits

3,724,310,091

Projects

273,655,292

Directories

14,081,620,820

Authors

68,736,671

Releases

81,086,168

Metadata : ~ 15TB

Wall archive : almost 1 PB

The largest archive ever built

Bitbucket		git	R	debian		GitHub	gitiles	GitLab
2,489,011 origins	56,983 origins	23,961 origins	26,416 origins	135,568 origins	52,811 origins	193,816,991 origins	10,021 origins	4,137,619 origins
git	Gogs	GO	GNU	heptapod	launchpad	Maven	npm	
2,619 origins	172 origins	899,728 origins	354 origins	1,196 origins	500,444 origins	312,461 origins	3,485,607 origins	4,986 origins
	Packagist	fedora PAGURE	Phabricator	pub.dev	python Package Index	SOURCEFORGE	stagit	
	300,175 origins	67,590 origins	202 origins	47,587 origins	497,141 origins	381,330 origins	254 origins	

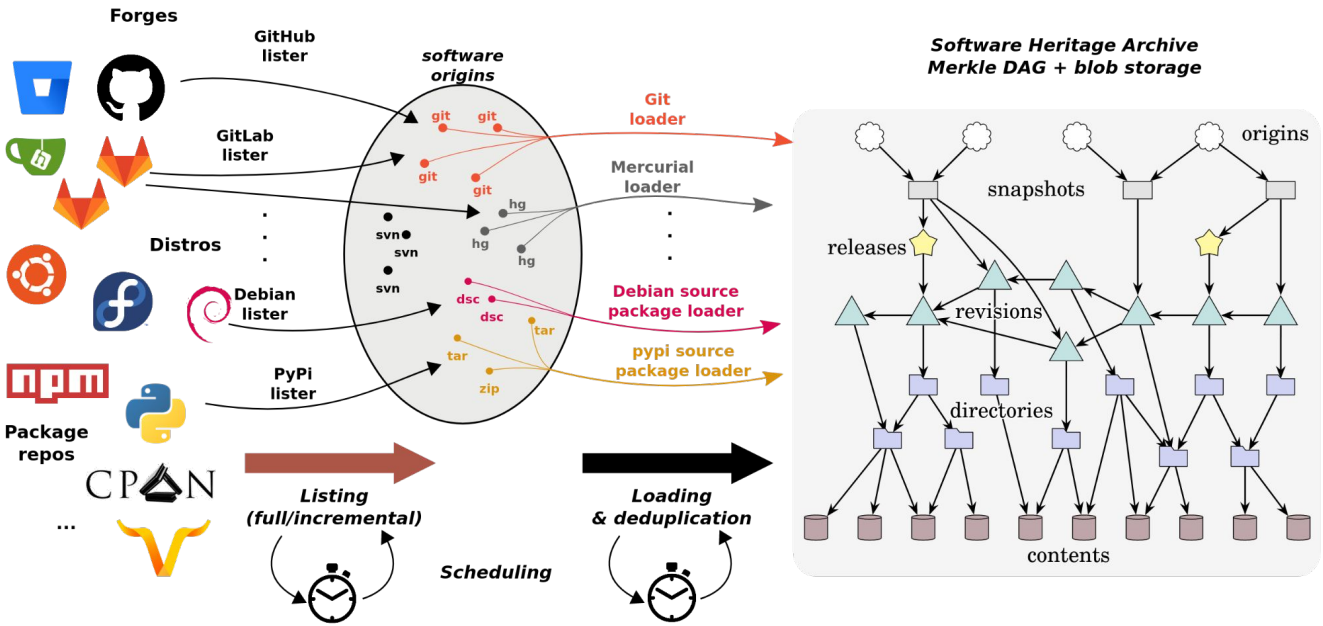
On demand archival

eLife	HAL science ouverte	IPOL Journal
12 origins	570 origins	193 origins

Discontinued hosting

GITORIOUS	Google code	Bitbucket
122,014 origins	790,026 origins	336,795 origins

Harvest and archive



Current rate: 10 origins are visited per second

Reference (25 billion SWHIDs)

SoftWare Hash Identifiers (swhid.org)

Intrisic, decentralised, cryptographically strong

Why are we interested in querying SWH ?

- For empirical research in Software Engineering
  - Mining Software Repository community
- For building (reproducible) source code datasets
  - Benchmarks
  - Machine learning training dataset (LLM etc ...).
- For building refined datasets ie. extending the SWH model with metrics of interests
  - Adding extra metadata (labelization of vulnerable commits, code metrics, energy consumption)
  - At different level: origin, snapshot, commit, file (... and why not AST node in the future ?)

Forges do not provide appropriate tooling for large scale mining

Heterogeneous information sources with heterogeneous API

SOURCEFORGE

GitLab

Bitbucket

WIKIMEDIA FOUNDATION

... At the end you will choose github

- Query Expressivity Limitation
- Rate Limitation
- Complex API

Why using Software Heritage ?

Availability

Multistakeholder infrastructure

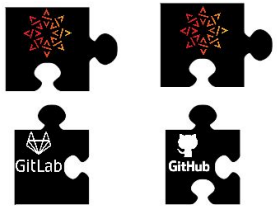
Traceability

Intrinsic unique identifiers (SWHID)

Immutability

Append only model (except law requirement)

Uniformity



Uniform API

~~REST~~ GraphQL API for tiny requests

- Navigate through the archive, node by node
- Request the download of files / commit / snapshots
- Limited to 10k requests by hour
- Request the archive of repository

Internal API - SWH storage

- Can access to the entirety of the archive (File etc ...)
- Query/traversal are performed on the production environment
- Backed by relational DB, non-adapted to resources intensive graph traversal

Need a separation of concerns, archiving / searching

SWH - Search

- Independent
- Backed by elastic search

- Limited to origin search
- Accessible through web-api

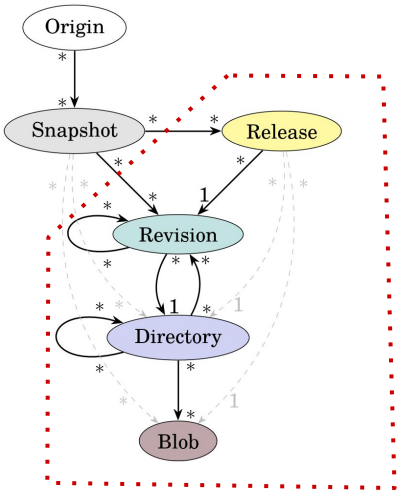
```
origin : plasma and language in [python] and visits >= 5
last_visit > 2021-01-01 or last_visit < 2020-01-01
visited = false and metadata : "kubernetes" or origin : "minikube"
keyword in ["orchestration", "kubectl"] and license in ["GPLv3+", "GPLv3"]
(origin : debian or visit_type = ["deb"]) and license in ["GPL-3"]
```

Not suitable for graph traversal request

The property Graph Dataset

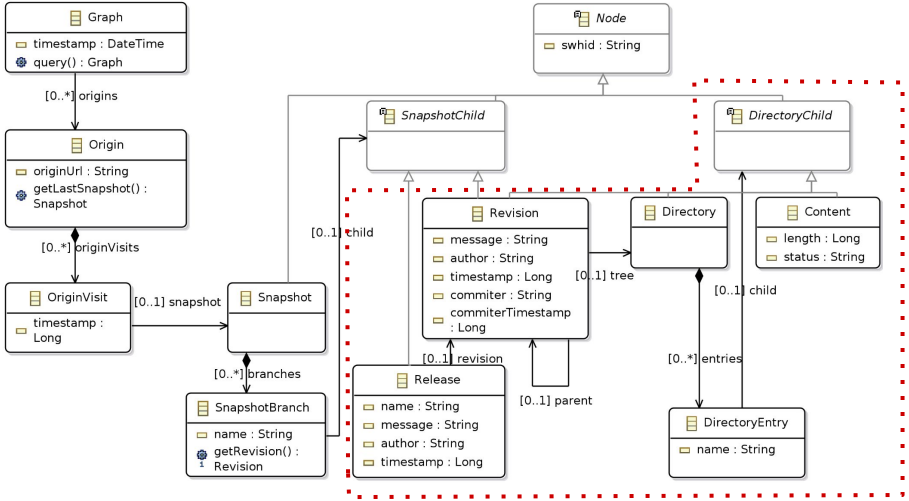
- Provide a fully independent service to access graph metadata
- **Compress** the graph and perform request in its compressed version
- Can be used through different API :
  - JAVA, REST, web-rpc
- Provided as an external services

No API limitation in terms of expressivity



Software Heritage Graph Dataset [1]

Git structure



Object model of the Software Heritage Graph Dataset

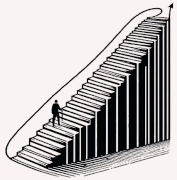
[1] Antoine Pietri. Organizing the graph of public software development for large-scale mining. Université Paris Cité, 2021.

SWH graph in practice - JAVA API

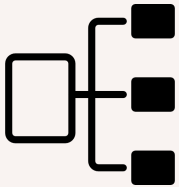
Advantages

- Expressivity : all queries can be designed
- Performant graph traversal
- Performant transitive closure

Constraints

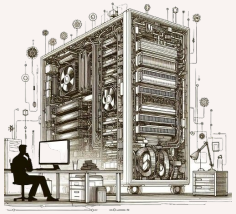


Steep learning curve



Parallelization

Self hosted



Resource Intensive
500 GB - 1TB+ RAM
10TB of SSD

SWH graph in practice - gRPC API

+

gRPC

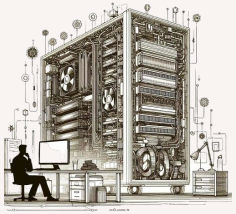
Advantages

- Ease of use
- Provide high performance graph traversal method
  - returning node / edge properties
  - performing BFS traversals
  - finding shortest paths
  - common ancestors

Constraints

Less expressive than Java API

Self hosted

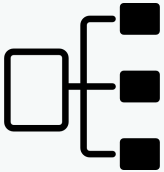


Ressource Intensive
500 GB - 1TB+ RAM
6TB of SSD

SWH graph dataset - columnar export

Column based model

Advantages



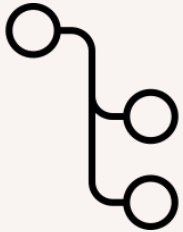
Parallelization

Queryable via cloud provider

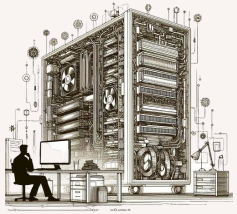
Ease of use

Performant property access

Constraints



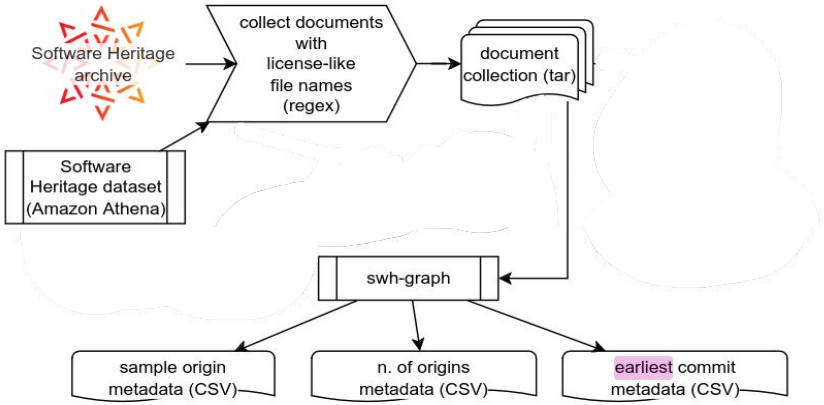
Graph Traversal



Ressource Intensive

Combining swh-graph + column based version

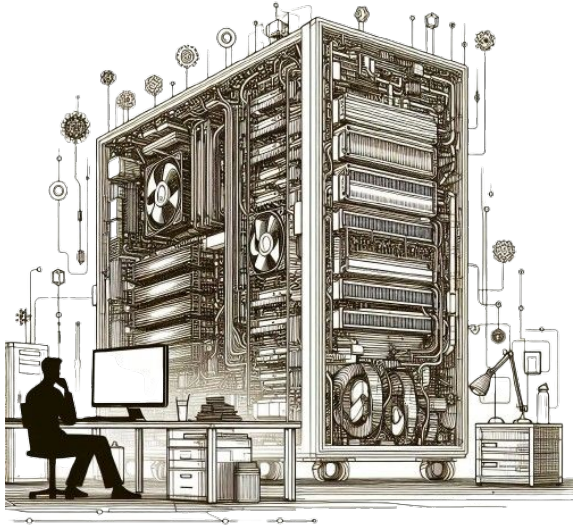
- Executing non optimized query on swh-graph can be costly
- 2-steps process, querying both SWH-graph and the column based version



Example: Licence extraction (partial workflow)

Querying SWH is complicated

- Time consuming / difficult to learn
  how to deploy + run a query on SWH-graph
- Even with time, resources are needed
- A query language is needed :
  - Easy to use
  - Hide complex query between
    column based / compressed version

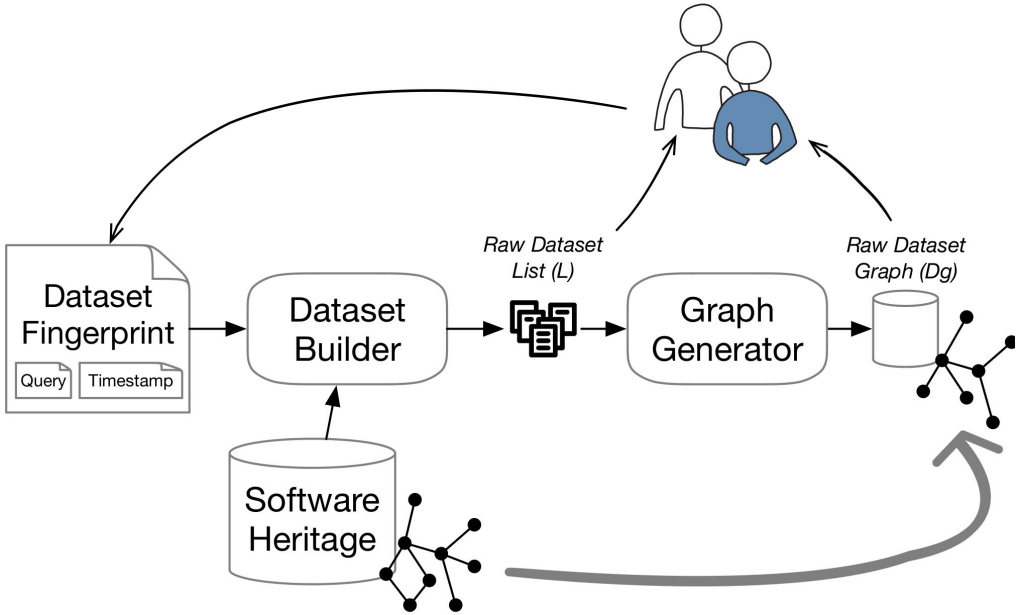


The fingerprint approach

1) A query on the data model of the source code

2) A timestamp to freeze the state of the archive

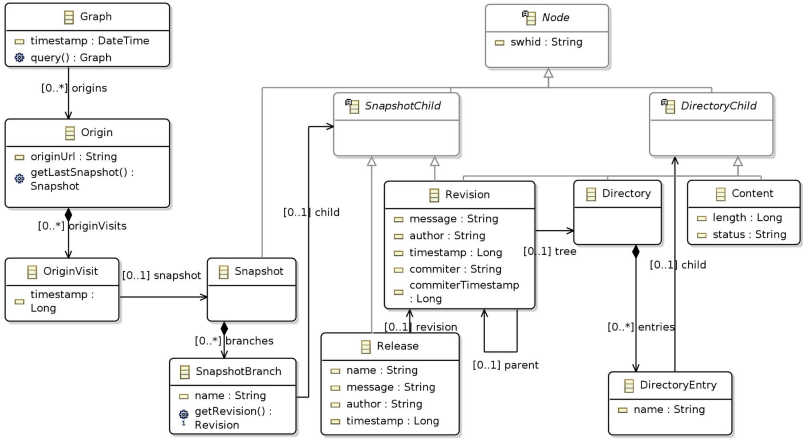
3) *A hash to prevent any corruption*



Operationalization of our approach :

Fingerprint Query Specification

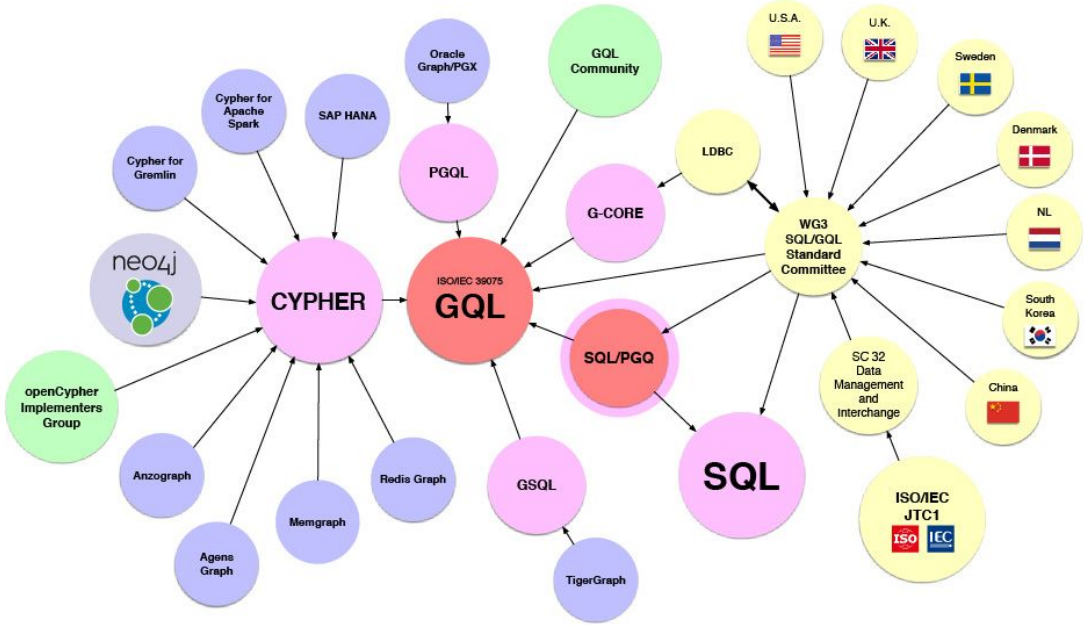
Object Constraint Language (OCL)



Object model of the SWH Graph Dataset

Fingerprint query = constraint on the SWH Graph Dataset

Towards a graph query language



GQL examples

Given two arbitrary revisions, return the shortest path between them in the undirected graph if it exists.[1]

```
MATCH (a:Revision) WHERE a.swhid = "swh:1:abc123..."
WITH a
MATCH (b:Revision) WHERE b.swhid = "swh:1:def456..."
WITH a, b
MATCH p = shortestPath((a)-[*]-(b))
RETURN nodes(p)
```

Given an origin, return all the objects reachable from it, but not reachable from any other origin[1]

```
MATCH (repo:Origin) WHERE repo.url = "github.com/..."
WITH repo
MATCH (allother:Origin) WHERE allother.url <> "github.com/..."
```

[1] Antoine Pietri. Organizing the graph of public software development for large-scale mining. Université Paris Cité, 2021.

Discussion

- Have you ever had the need to mine repositories? What were your requirements? Did you succeed without any problems ?

- Do you have any ideas of research questions that involve searching for information / mining SWH ?

- What would be the interesting properties of a query language? Any ideas on a particular syntax?

Thanks