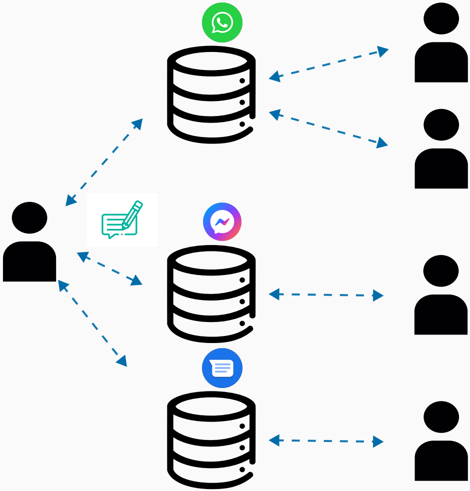
# Revisiting Link Prioritization for Efficient Traversal in Structured Decentralized Environments

Ruben Eschauzier, Ruben Taelman, Ruben Verborgh

October 13, 2025

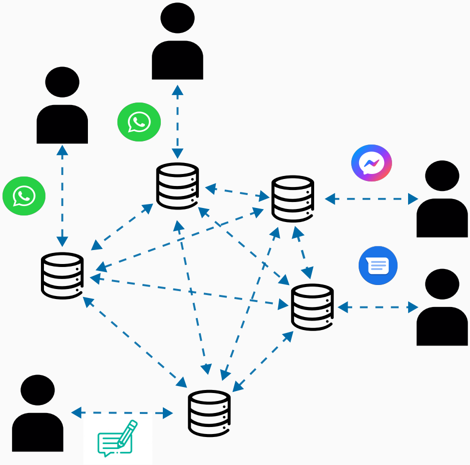
Department of Electronics and Information Systems, Ghent University – Imec

# The Need for Decentralized Personal Data Storage



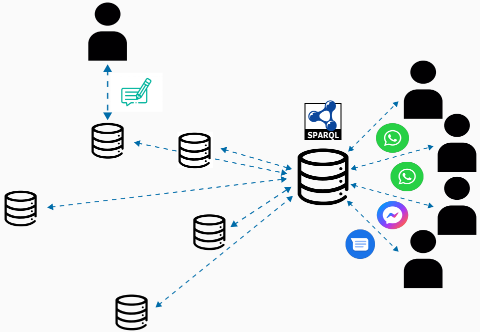
- Each application has its own data
- Stiffles innovation
- Causes vendor lock-in

# The Need for Decentralized Personal Data Storage



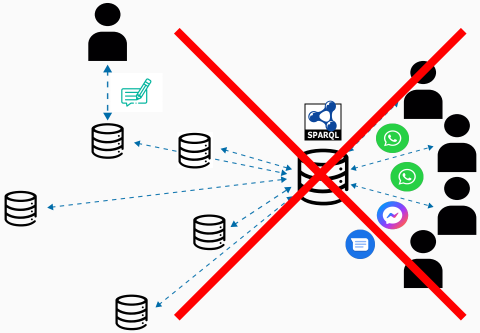
- Each application uses common data storage
- Easy to switch vendors
- Promotes innovation

# The Problem with Centrally Aggregating and Querying




- Why not aggregate data and query it?
- Impossible in case of personal data due to privacy concerns

# The Problem with Centrally Aggregating and Querying

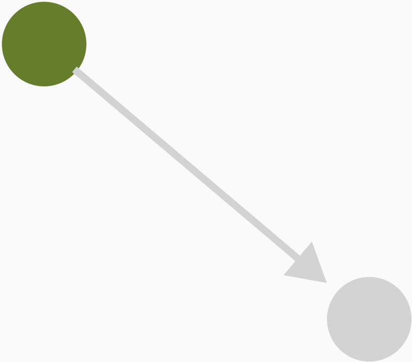



- Why not aggregate data and query it?
- Impossible in case of personal data due to privacy concerns

# Link Traversal-based Query Processing

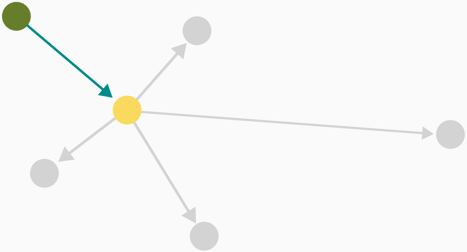
- Link Traversal iteratively dereferences data to query over
- Continuously produces results
- Can enforce fine-grained (document-level) access-control

## Link Traversal: Seed Document



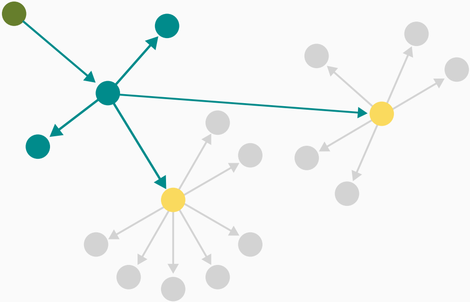
- Link Traversal starts from seed documents (URIs)
- These are provided by the user or in the query.

# Link Traversal: Traversal



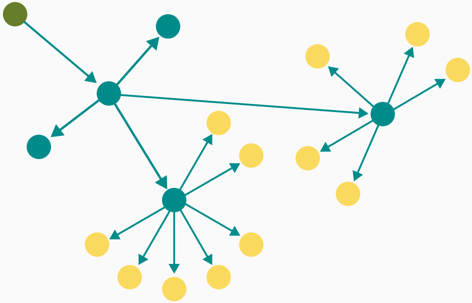
- New URIs are extracted from the seed document
- URIs are extracted in accordance with reachability criterions

# Link Traversal: Traversal



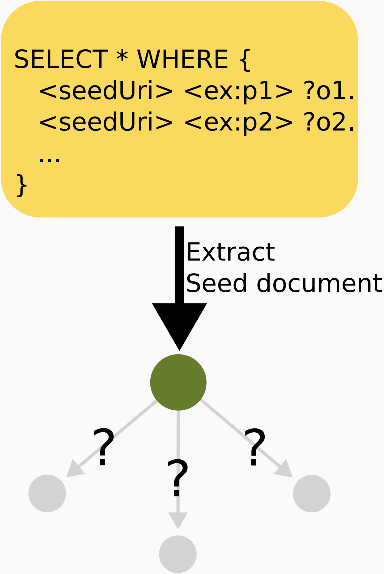
- New URIs are dereferenced and the process is repeated

# Link Traversal: Termination



- This continues untill all links are dereferenced

# Query Optimization for Link Traversal

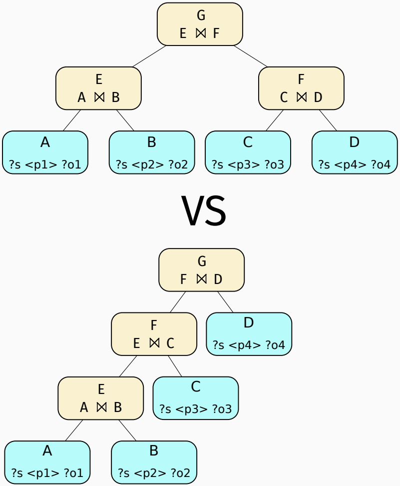



The query (partly) determines:

- The queried data
- The topology of the queried data
- The query-relevant documents

Result: limited prior knowledge for query optimization

# Query Optimization for Link Traversal: Traditional Query Planning

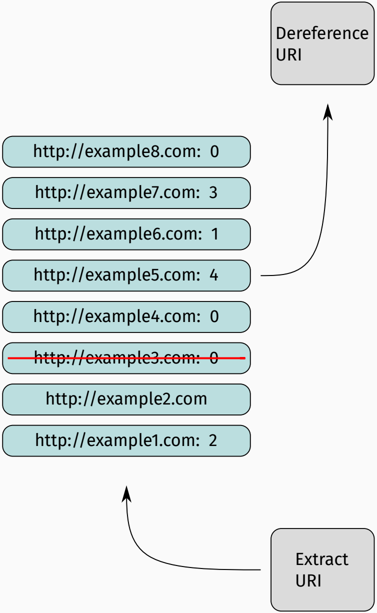



- Query optimization for link traversal involves traditional (zero knowledge) query planning

# Query Optimization for Link Traversal: Traversal optimization


- URIs are put into a link queue
- Link traversal uses a FiFo queue by default

# Query Optimization for Link Traversal: Traversal optimization




Optimizations:

- Prioritize query-relevant URIs
- Prune irrelevant URIs
- Pruning can lead to missing results without prior knowledge

# Problem Statement

Investigate the performance of link prioritization algorithms in literature in new structured decentralized environments

# Problem Statement

- These algorithms are implemented in an engine that is no longer maintained
- The baseline performance depends on design choices orthogonal to the prioritization algorithm
- The baseline should measure marginal prioritization performance

# $R^3$ metric: Motivation

**Context**
During link traversal, the engine traverses a fixed topology entirely

**Goal**
The goal of prioritization is to find query-relevant (*where-provenance*) documents as soon as possible.

**Challenge**
Find the **optimal traversal order** of a given query in hindsight to compare the traversal order taken by the engine to.

## $R^3$ metric: Steiner trees

- The problem of finding the optimal traversal order can be translated to solving the directed graph steiner tree problem.
- The traversed topology admits directed graph $G = (V, A)$ with root $r$
- Query-relevant documents $D_T$ serve as terminals $T \subseteq V$
- Find minimum cost sub-graph $X = (V', A')$ starting at root $r$ and spanning all vertexes $T$.
- With cost: $C(X) = \sum_{a \in A'} c(a)$,, with $c(a)$ the cost of an edge in the topology

# $R^3$ metric

### Definition

$$R^3 = \frac{C(X)}{C(O_{\mathcal{T}})}$$

- $X$: the **optimal traversal order** (minimal-cost path)
- $O_{\mathcal{T}}$: traversal order produced by the link prioritization algorithm
- $C(\cdot)$: The total cost of all arcs in the traversal path

### Interpretation

- $R^3 = 1 \rightarrow$ perfect match with the optimal traversal
- $R^3 < 1 \rightarrow$ deviation from optimal performance
- **Higher values indicate better prioritization quality.**

# $R^3$ Metric Illustration

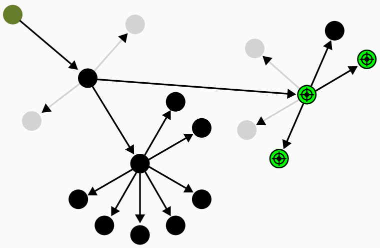


**Figure 1:** Actual traversal

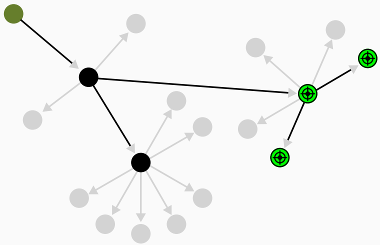


**Figure 2:** Optimal traversal: minimal cost path

$$R^3 = \frac{5}{13} = 0.38$$

# Prioritization algorithms

**Non-adaptive**

- Breadth-first (default), depth-first, random prioritization

**Graph-based**

- In-degree, PageRank score

**Result-based**

- Uses result contribution count (RCC) of each node (URI) in the topology
- Priority set based on the sum or count of non-zero RCC of the 1 or 2-hop in-neighbours of a node
- Called rcc-1, rcc-2, rel-1, rel-2 respectively.

# Prioritization algorithms

**Intermediate-results**

- Uses intermediate solutions in the engine
- Priority of URI is determined by the largest intermediate result with that URI bound
- *IS* sets initial priorities to 0, while *ISdcr* sets priority to the priority of the parent node - 1

**Hybrid**

- Multiply intermediate and full result scoring functions
- *is-rcc1*, *is-rcc2*, *is-rel1*, *is-rel2*

**TypeIndex**

- TypeIndex points to location for resource of specific type
- Prioritize TypeIndex

# Prioritization algorithms

**Oracle**

- Compute RCC in hindsight
- Scores are propegated through the shortest path
- Serves as optimal performance oracle