LDIF

Linked Data Integration Framework

LINKED DATA CHALLENGES

- Data sources that overlap in content may:
  - use a wide range of different RDF vocabularies
  - use different identifiers for the same real-world entity
  - provide conflicting values for the same properties
- Implications:
  - Queries are usually hand-crafted against individual sources – no different than an API
  - Improvised or manual merging of entities
- Integrating public datasets with internal databases poses the same problems

LDIF

- LDIF homogenizes Linked Data from multiple sources into a clean, local target representation while keeping track of data provenance

1. Collect data: Managed download and update
2. Translate data into a single target vocabulary
3. Resolve identifier aliases into local target URIs
4. Cleanse data resolving the conflicting values
5. Output

- Open source (Apache License, Version 2.0)
- Collaboration between Freie Universität Berlin and mes|semantics

LDIF PIPELINE

1 Collect data

2 Translate data

3 Resolve identities

4 Cleanse data

5 Output

Supported data sources:

- RDF dumps (various formats)
- SPARQL Endpoints
- Crawling Linked Data

LDIF PIPELINE

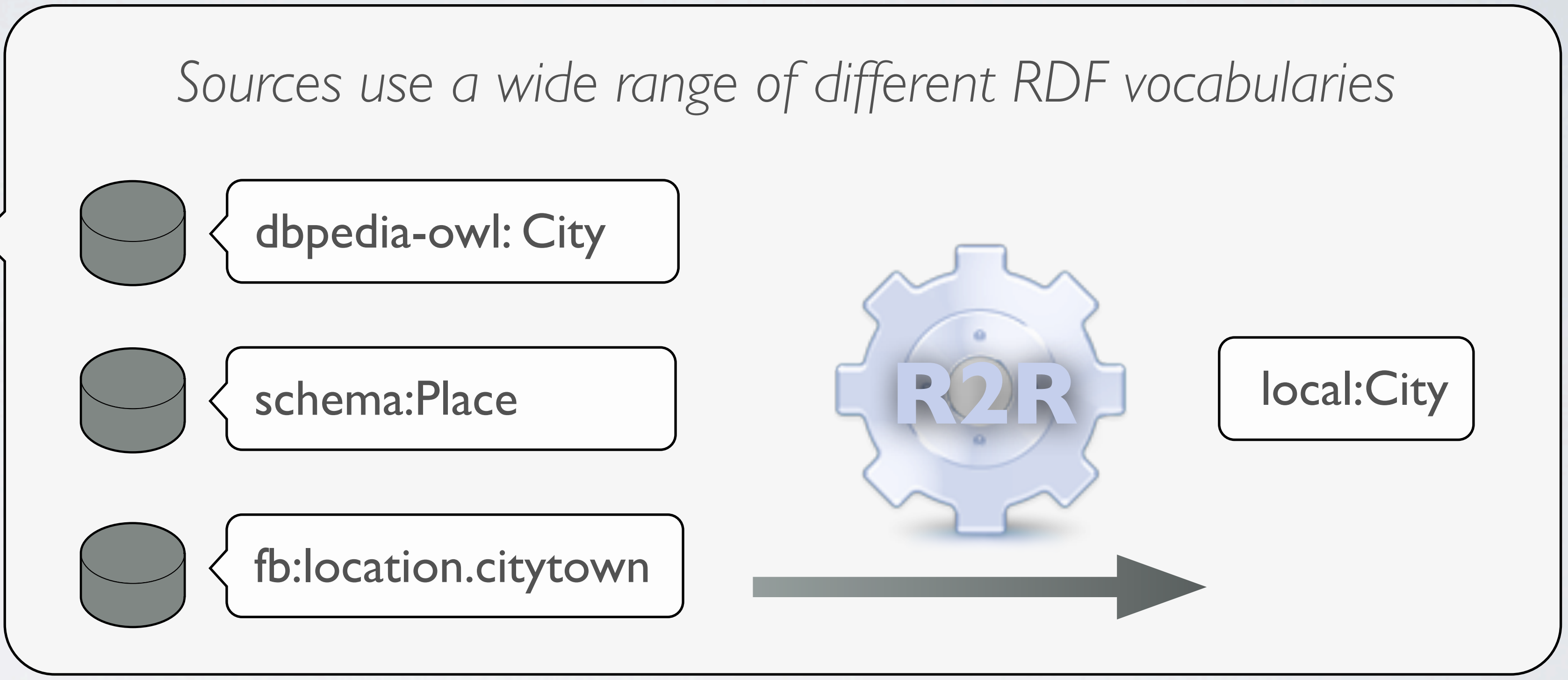
1 Collect data

2 Translate data

3 Resolve identities

4 Cleanse data

5 Output



- Mappings expressed in RDF (Turtle)
- Simple mappings using OWL / RDFs statements (x rdfs:subClassOf y)
- Complex mappings with SPARQL expressivity
- Transformation functions

Freie Universität Berlin   mes|semantics

LDIF PIPELINE

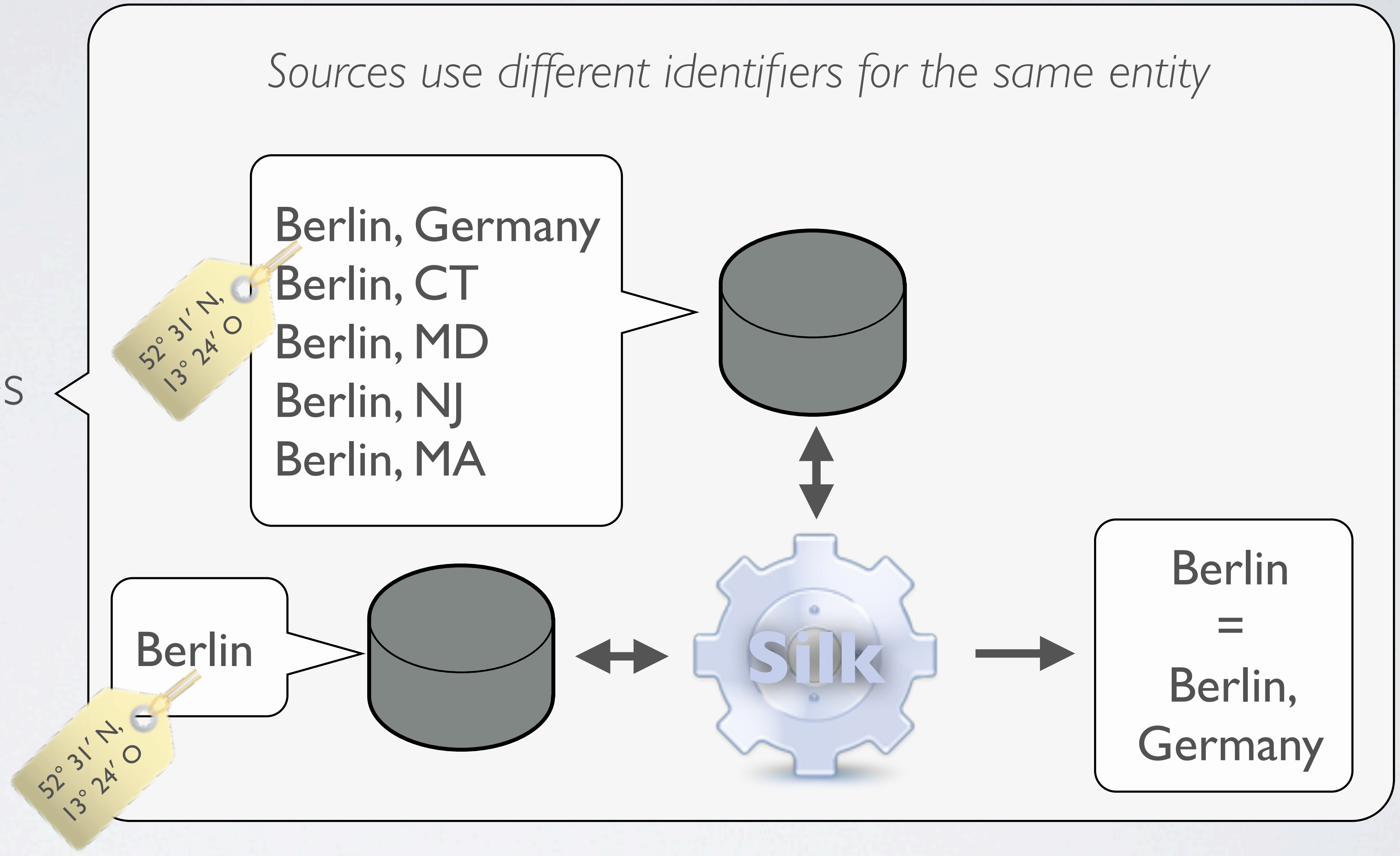
1 Collect data

2 Translate data

3 Resolve identities

4 Cleanse data

5 Output



- Profiles expressed in XML
- Supports various comparators and transformations

LDIF PIPELINE

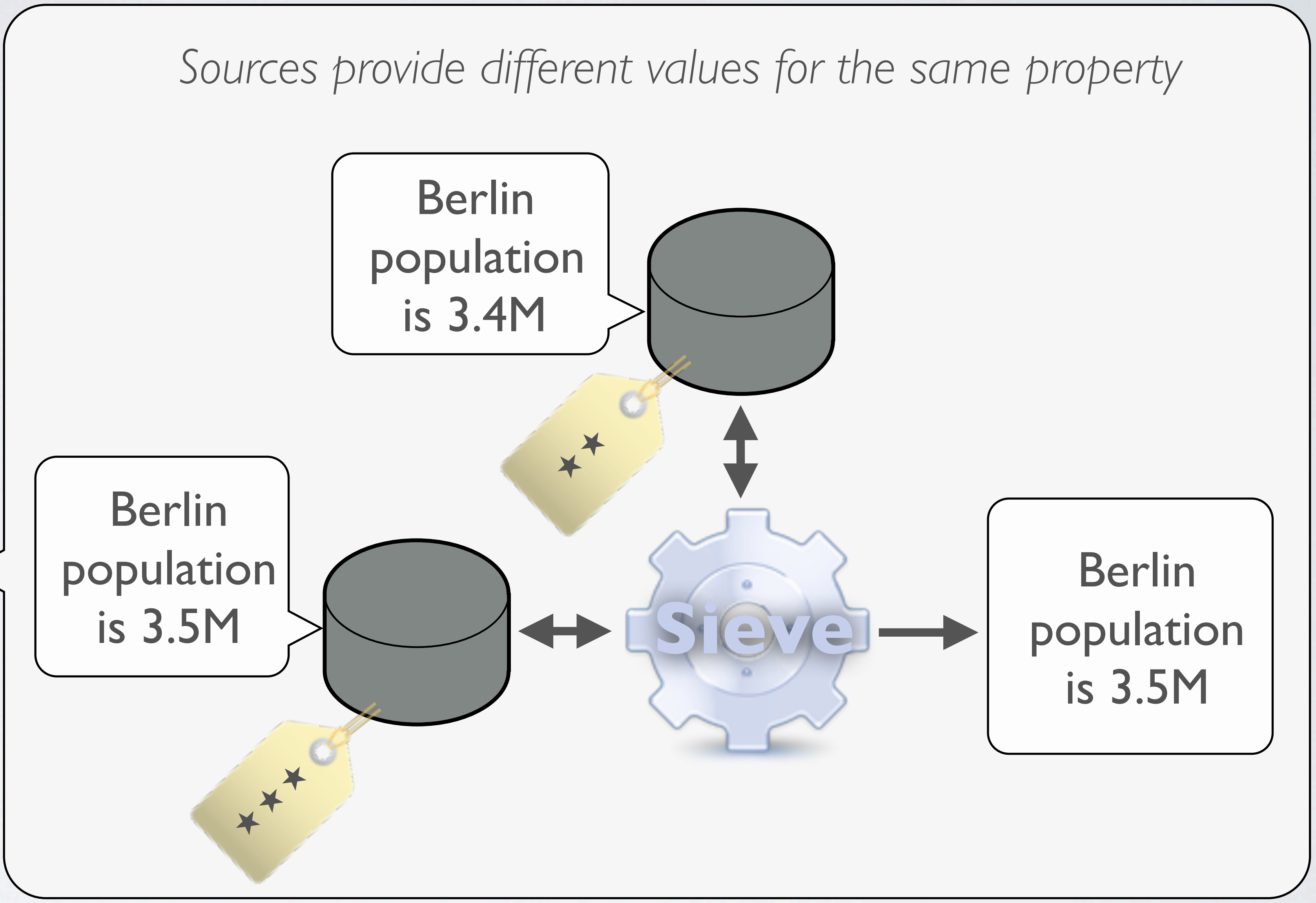
1 Collect data

2 Translate data

3 Resolve identities

4 Cleanse data

5 Output


Sources provide different values for the same property
Berlin population is 3.4M
★★
Berlin population is 3.5M
★★★
Sieve
Berlin population is 3.5M

- Profiles expressed in XML
- Supports various quality assessment policies and conflict resolution methods

LDIF PIPELINE

1 Collect data

2 Translate data

3 Resolve identities

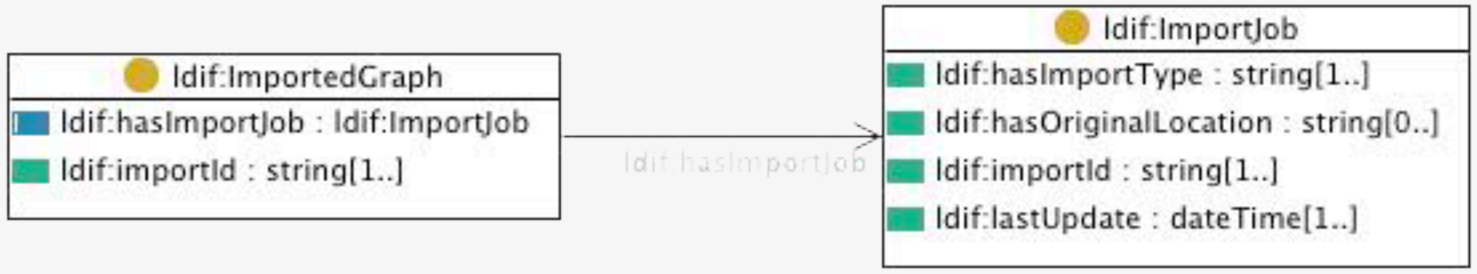
4 Cleanse data

5 Output

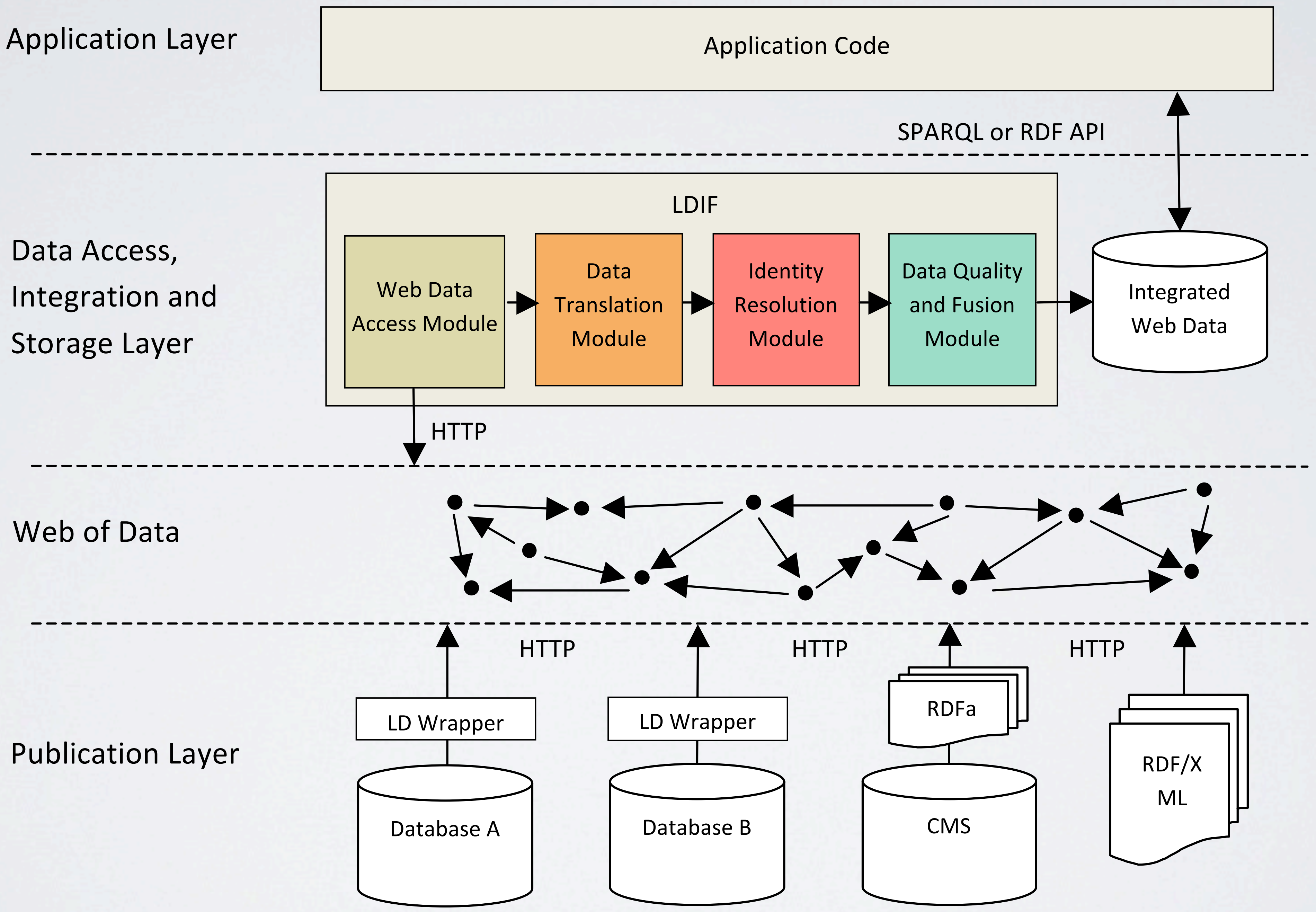
Output options:

- N-Quads
- N-Triples
- SPARQL Update Stream

- Provenance tracking using Named Graphs


ldif:ImportedGraph
ldif:hasImportJob : ldif:ImportJob
ldif:importId : string[1..]
ldif:hasImportJob
ldif:ImportJob
ldif:hasImportType : string[1..]
ldif:hasOriginalLocation : string[0..]
ldif:importId : string[1..]
ldif:lastUpdate : dateTime[1..]

LDIF ARCHITECTURE



LDIF VERSIONS

- In-memory
  - keeps all intermediate results in memory
  - fast, but scalability limited by local RAM
- RDF Store (TDB)
  - stores intermediate results in a Jena TDB RDF store
  - can process more data than In-memory but doesn't scale
- Cluster (Hadoop)
  - scales by parallelizing work across multiple machines using Hadoop
  - can process a virtually unlimited amount of data

THANK YOU

- Website: http://ldif.wbsg.de
- Google group: http://bit.ly/ldifgroup

- Supported in part by
    - Vulcan Inc. as part of its Project Halo
    - EU FP7 project LOD2 - Creating Knowledge out of Interlinked Data (Grant No. 257943)