Web Data Management

Vera Goebel

Department of Informatics, University of Oslo

Web Models
Web Search
Web Querying
Web Data Exchange
Web Data Integration

Web Overview (status 2011)

- Publicly indexable web:
  - More than 25 billion static HTML pages.
  - Over 53 billion pages in dynamic web
- Deep web (hidden web)
  - Over 500 billion documents
- Most Internet users gain access to the web using search engines.
- Very dynamic
  - 23% of web pages change daily.
  - 40% of commercial pages change daily.
- Static versus dynamic web pages
- Publicly indexable web (PIW) versus hidden web (or deep web)

Current status: http://www.worldwidewebsize.com/

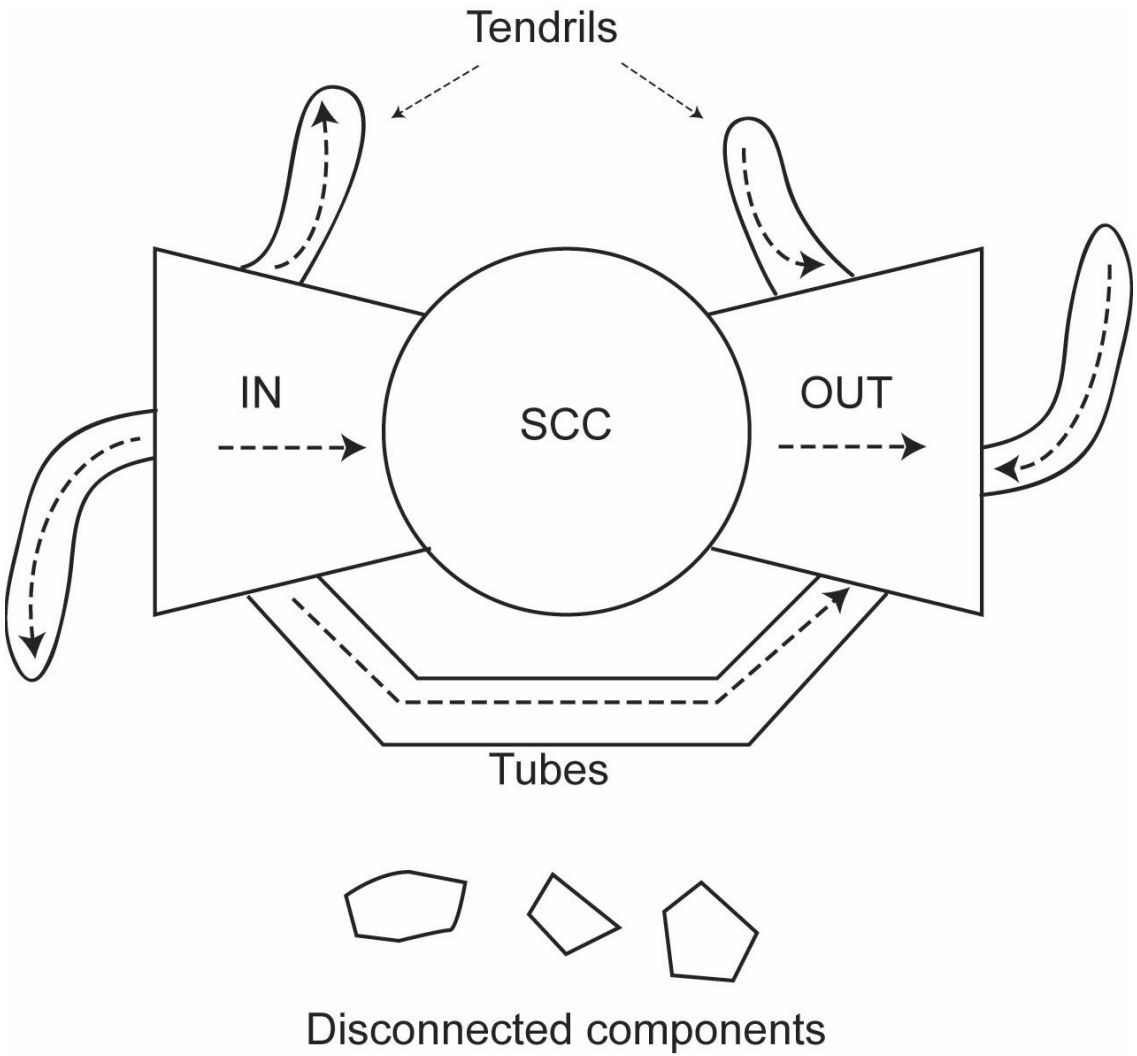
Properties of Web Data

- Lack of a schema
  - Data is at best “semi-structured”
  - Missing data, additional attributes, “similar” data but not identical
- Volatility
  - Changes frequently
  - May conform to one schema now, but not later
- Scale
  - Does it make sense to talk about a schema for Web?
  - How do you capture “everything”?
- Querying difficulty
  - What is the user language?
  - What are the primitives?
  - Search engines or metasearch engines?

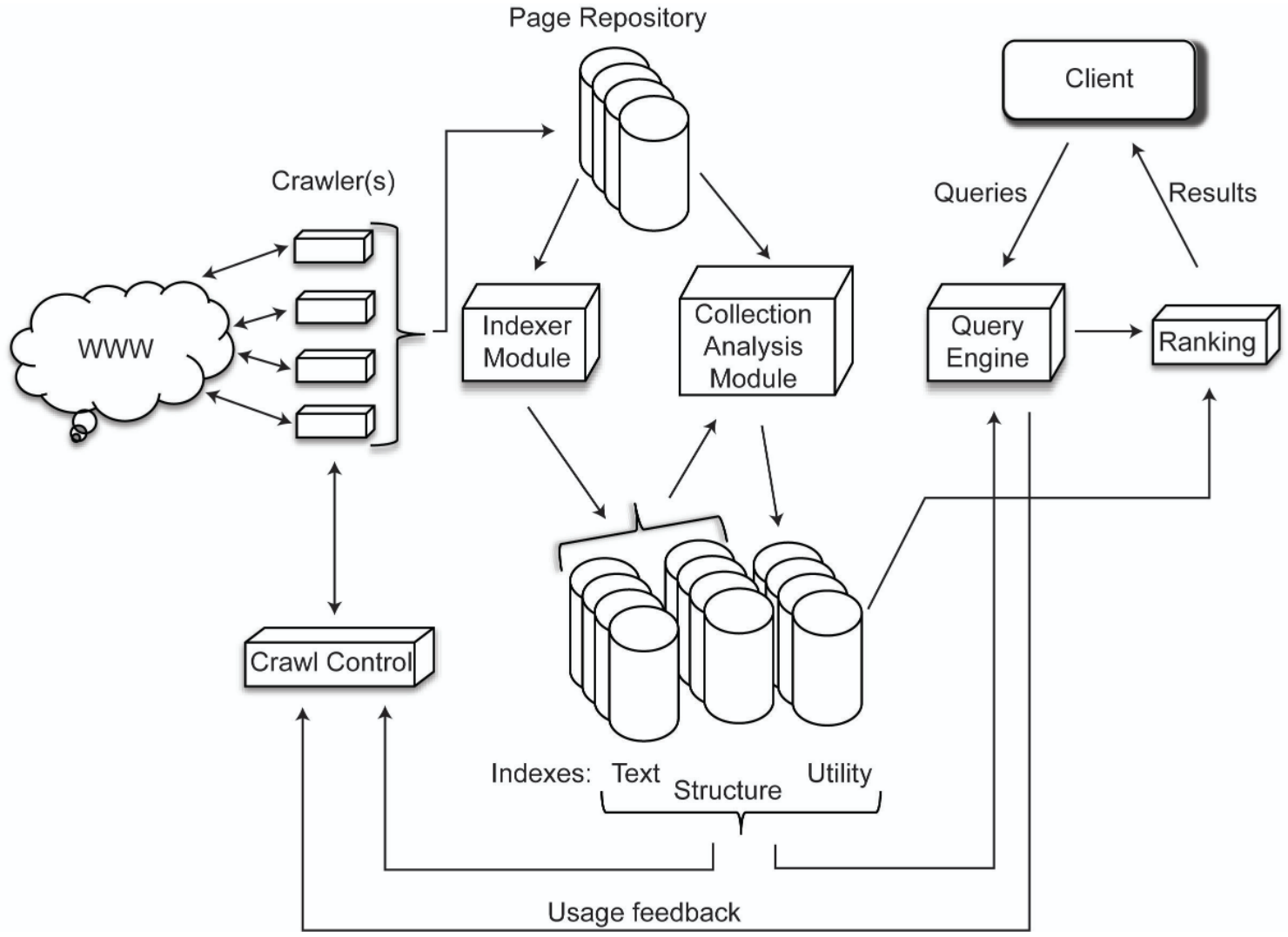
Web Model

- Web graph
  - Pages are vertices
  - Hyperlinks between pages are edges
  - Semi-structured data
- Properties
  - Volatile
  - Sparse
  - Self-organizing
  - “Small-world” graph
  - Power law graph

Structure of the Web



Search Engine Architecture



Web Crawling

- What is a crawler?
- Crawlers cannot crawl the whole Web. It should try to visit the “most important” pages first.
- Importance metrics:
  - Measure the importance of a Web page
  - Ranking
    - Static
    - Dynamic
- Ordering metric
  - How to choose the next page to crawl

Web Crawler Types

- Many Web pages change frequently
  -> crawler has to revisit already crawled pages
  → incremental crawlers

- Some search engines specialize in searching pages belonging to a particular topic
  → focused crawlers

- Search engines use multiple crawlers sitting on different machines and running in parallel
  -> coordinate parallel crawlers to prevent overlapping
  → parallel crawlers

Indexing

- Structure index
  - Link structure
- Text index
  - Indexing the content
  - Suffix arrays, inverted index, signature files
  - Inverted index most common
- Difficulties of inverted index
  - The huge size of the Web
  - The rapid change makes it hard to maintain
  - Storage versus performance efficiency

Web Querying

- Why Web Querying?
  - It is not always easy to express information requests using keywords.
  - Search engines do not make use of *Web topology* and *document structure* in queries.
- Early Web Query Approaches
  - Structured (Similar to DBMSs): Data model + Query Language
  - Semi-structured: e.g. Object Exchange Model (OEM)

Web Querying

- Question Answering (QA) Systems
  - Finding answers to natural language questions, e.g. *What is Computer?*
  - Analyze the question and try to guess what type of information that is required.
  - Not only locate relevant documents but also extract answers from them.

Approaches to Web Querying

- Search engines and metasearchers
  - Keyword-based
  - Category-based
- Semistructured data querying
- Special Web query languages
- Question-Answering

Semistructured Data Querying

- Basic principle: Consider Web as a collection of semistructured data and use those techniques
- Uses an edge-labeled graph model of data
- Example systems & languages:
  - Lore/Lorel
  - UnQL
  - StruQL

Evaluation

- Advantages
  - Simple and flexible
  - Fits the natural link structure of Web pages
- Disadvantages
  - Data model too simple (no record construct or ordered lists)
  - Graph can become very complicated
    - Aggregation and typing combined
    - DataGuides
  - No differentiation between connection between documents and subpart relationships

Web Query Languages

- Basic principle: Take into account the documents' content and internal structure and external links
- Graph structures are more complex
- First generation
  - Model the web as interconnected collection of atomic objects
  - WebSQL
  - W3QL
  - WebLog
- Second generation
  - Model the web as a linked collection of structured objects
  - WebOQL
  - StruQL

Evaluation

- Advantages
  - More powerful data model - Hypertree
    - Ordered edge-labeled tree
    - Internal and external arcs
  - Language can exploit different arc types (structure of the Web pages can be accessed)
  - Languages can construct new complex structures.
- Disadvantages
  - You still need to know the graph structure
  - Complexity issue

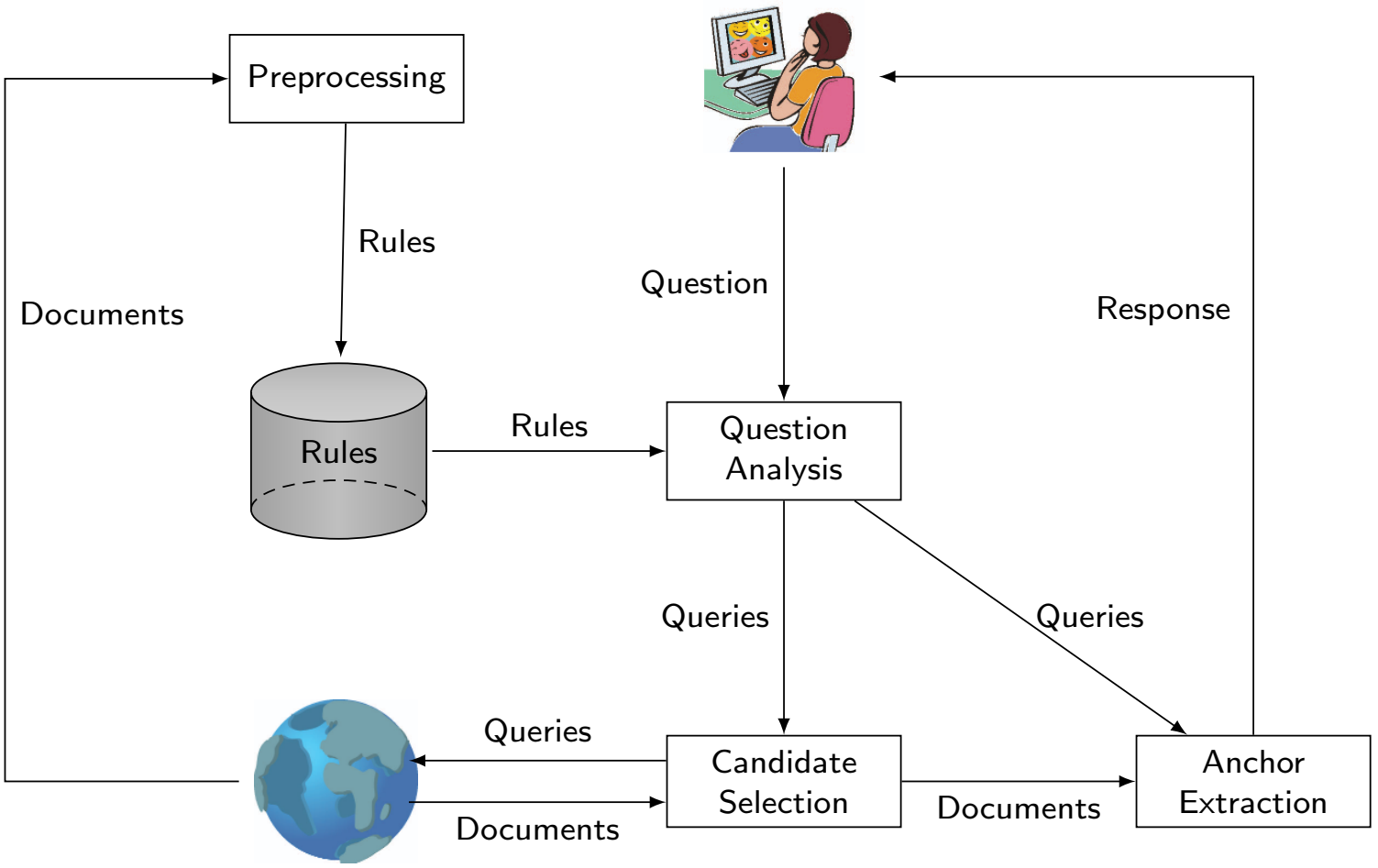
Question-Answer Approach

- Basic principle: Web pages that could contain the answer to the user query are retrieved and the answer extracted from them.
- NLP and information extraction techniques
- Used within IR in a closed corpus; extensions to Web
- Examples
  - QASM
  - Ask Jeeves
  - Mulder
  - WebQA

Question-Answer Systems

- Analyze and classify the question, depending on the expected answer type.
- Using IR techniques, retrieve documents which are expected to contain the answer to the question.
- Analyze the retrieved documents and decide on the answer.

Question-Answer System



Searching The Hidden Web

- Publicly indexable web (PIW) vs. hidden web
- Why is Hidden Web important?
  - Size: huge amount of data
  - Data quality
- Challenges:
  - Ordinary crawlers cannot be used.
  - The data in hidden databases can only be accessed through a search interface.
  - Usually, the underlying structure of the database is unknown.

Searching The Hidden Web

- Crawling the Hidden Web
  - Submit queries to the search interface of the database
    - By analyzing the search interface, trying to fill in the fields for all possible values from a repository
    - By using agents that find search forms, learn to fill them, and retrieve the result pages
  - Analyze the returned result pages
    - Determine whether they contain results or not
    - Use templates to extract information

Searching The Hidden Web

- Metasearching
  - Database selection – Query Translation – Result Merging
  - Database selection is based on *Content Summaries*.
  - Content Summary Extraction:
    - RS-Ord and RS-Lrd
    - Focused Probing with Database Categorization

Web Data Exchange

- XML basics

Why Use XML with a DBS?

Early solutions:

- XML as transport format
  - to/from database
- XML for semistructured data
  - schema to be stored not known at design time
- XML document archiving
  - retaining complete XML documents
- Application domain -> choice of XML-DBS

XML – In a Nutshell

Extensible Markup Language, World Wide Web Consortium (W3C)

- Standard of the W3C in 1998
- Documents (content)
- XML Schema or DTD (structure)
- Namespaces
- XSL, XSLT (stylesheets, transformations)
- XPath, XPointer, XLink
- XQuery

- Tagged elements with attributes
  - Extensible, self-describing
- Tag names must be unique (use namespaces)

XML Overview

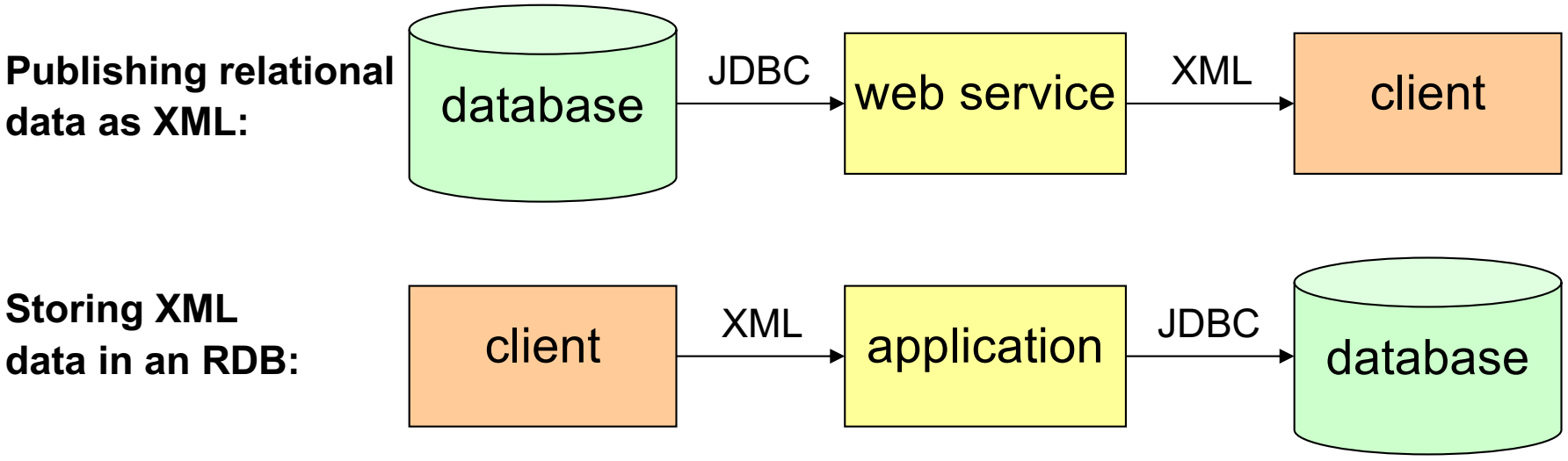
- Data is divided into pieces called elements
- Elements can be nested, but not overlapped
  - Nesting represents hierarchical relationships between the elements
- Elements have attributes
- Elements can also have relationships to other elements (ID-IDREF)
- Can be represented as a graph, but usually simplified as a tree
  - Root element
  - Zero or more child elements representing nested subelements
  - Document order over elements
  - Attributes are also shown as nodes

XML Document

```
<? xml version="1.0" ?>
<racingTeams xmlns="http://www.cart.org"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-Instance"
    xsi:schemaLocation="http:// www.cart.org team.xsd">
    <team>
        <name>"Shell</name>
        <car number=3><make>"Helix Ford"</make>
            <engine id="47456"/></car>
        <car number=4><make>"Helix Ford"</make>…</car>
        <driver><name>"Steve Johnson"</name>
            <points>2217</points></driver >
        <driver><name>"Paul Radisich"</name></driver >
        <spares><engine id="102555"/>
            <engine id="102556"/></spares>
    </team>
    . . .
</racingTeams>
<!-- Each document has one root element (e.g., racingTeams) whose name
    matches a schema element or <!DOCTYPE name> -->
```

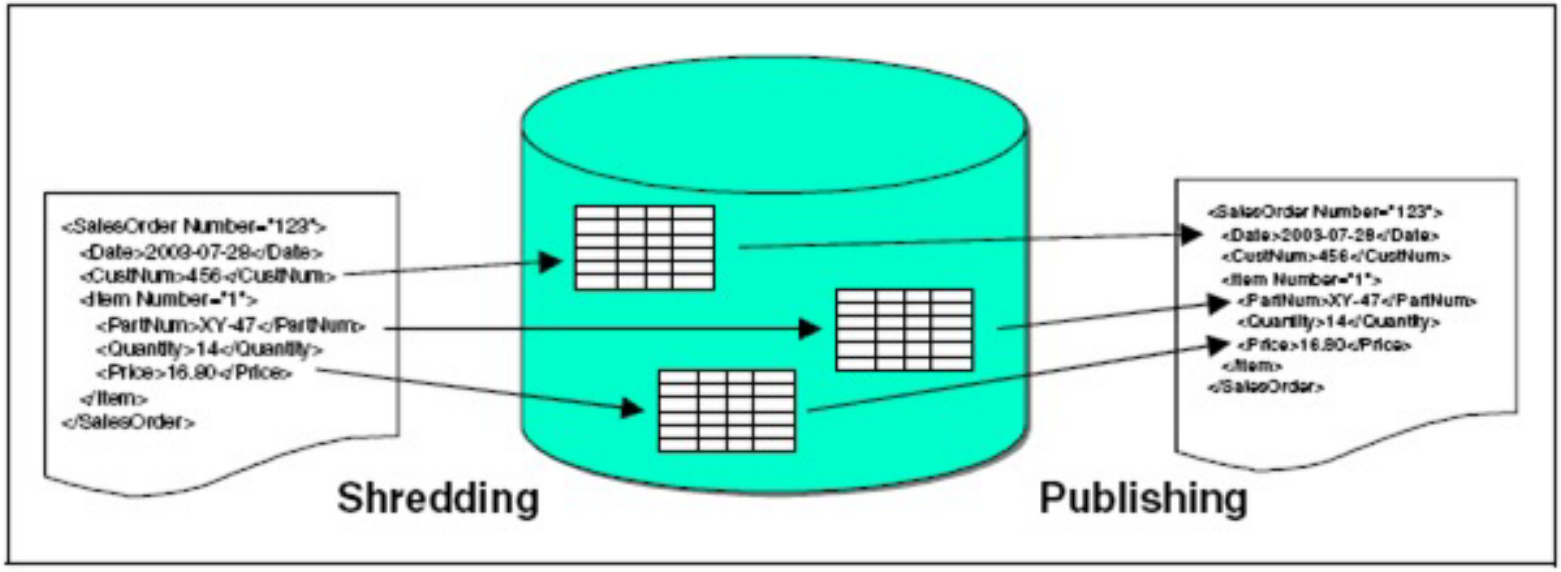
XML as Transport Format

- When
  - publishing as XML data stored in a DB, or storing data arriving as XML documents
    - between an existing DB and applications
    - between two DBs
- Why XML
  - platform independence



XML-Enabled DBS

- For data-centric XML documents
- When XML is used as a data transport format
- How
  - Store by shredding
  - Publish by composing



Properties of XML-Enabled DBS

- DB schema models primarily the *data* in the XML document
  - Data model is ”traditional”
    - relational/object-relational/...
  - No XML ”visible” in DB
- Keeps existing data and applications intact
  - adding XML functionality is adding and configuring data transfer (data mapping) software

Properties of XML-Enabled DBS

- XML mapping software
  - integrated into the DB engine or external to it
- Lossy modelling
  - XML document as such is discarded after shredding
  - In general, XML mapping software can handle only a subclass of XML documents
    - retains information that is important to the DB model:

      data itself

      hierarchical relationships (parent, child, siblings)

      not entity references, CDATA sections, comments, processing instructions, DTD, maybe not ordering of siblings,...
  - Cannot reconstruct XML document in full detail

XML for Semistructured Data

- When
  - application domains where
    - all data formats cannot be predicted in advance +
    - large volumes of data
- Why
  - why XML: extensibility
  - why DB: need to store, manage, and query data

XML Document Archiving

- When
  - retaining XML documents
- Why
  - processing
  - contracting/legislation
  - granularity: XML document or document fragment is the logical unit

NXDBS: Native XML DBS

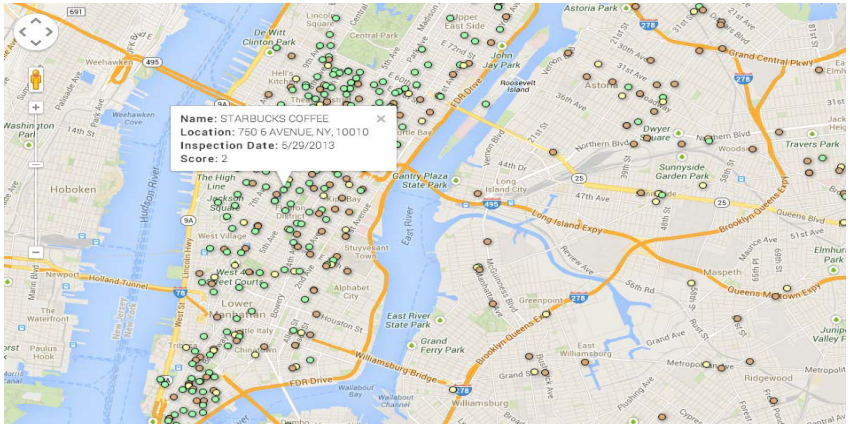
- For document-centric XML documents
  - need to be able to retain XML documents as-is
- For semi-structured data
  - when other DB models are inappropriate, e.g. since they require changing the DB schema when the XML schema evolves
- Specialized for storing XML data
  - Stores all components of the XML model intact
- XML document is the logical unit
  - (Cf. the logical unit in an RDBMS, which is a row)
- *Schema-independent NXDBS*
  - Can store any XML document regardless of its schema

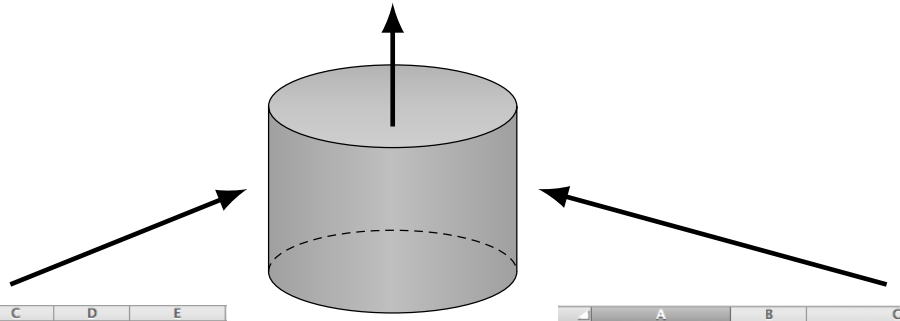
Web Data Integration

Where do we stand today? Modern solutions:

- "Big data" characteristics play a role
  - No or highly variable (among sources) schema
  - Numbers are much higher
  - Data quality is a major issue
- "Pay-as-you-go" integration
  - Data spaces → data lakes
- Some approaches
  - Web Tables / Fusion Tables
  - Semantic web & Linked Open Data (LOD)

Web Tables / Fusion Tables





	A	B	C	D	E
1	camis	inspection_d	action_code	score	current_grade
2	40363098	1/8/2013	D	11	A
3	40367790	5/1/2013	U	7	A
4	40367829	6/11/2013	D	13	A
5	40369535	4/4/2013	D	7	A
6	40369724	7/29/2013	F	28	C
7	40369782	7/1/2013	D	10	A
8	40369915	6/17/2013	U	11	A
9	40370146	11/16/2012	D	5	A
10	40370356	3/12/2013	D	6	A
11	40370781	4/30/2013	D	2	A
12	40371529	12/3/2012	D	9	A
13	40372196	4/30/2013	D	2	A
14	40372844	2/11/2013	D	11	A

	A	B	C	D	E	F
1	name	building	street	zipcode	phone	camis
2	DUNKIN' DONUTS	56	COURT STREET	11201	##########	40363098
3	MCDONALD'S	395	FLATBUSH AVENUE EXTENSION	11201	##########	40369535
4	MCDONALD'S	3267	RICHMOND AVENUE	10312	##########	40370356
5	MCDONALD'S	5804	CLARENDON ROAD	11203	##########	40373909
6	MCDONALD'S	75-50	101 AVENUE	11416	##########	40367829
7	MCDONALD'S	2154	HYLAN BOULEVARD	10306	##########	40371529
8	MCDONALD'S	3660	EAST TREMONT AVENUE	10465	##########	40372196
9	MCDONALD'S	943	FLATBUSH AVENUE	11226	##########	40367790
10	MCDONALD'S	221-28	HORACE HARDING EX	11364	##########	40372844
11	MCDONALD'S	701	UTICA AVENUE	11203	##########	40369782
12	MCDONALD'S	160-11	WILLETS POINT BOULEVARD	11357	##########	40370781
13	MCDONALD'S	106-15	71 AVENUE	11375	##########	40369724
14	MCDONALD'S	Feb-70	COPPER AVENUE	11385	##########	40377294
15	MCDONALD'S	5765-81	BROADWAY	10463	##########	40369915
16	DUNKIN' DONUTS/BASK	153-67	HORACE HARDING EXPRESSWA	11367	##########	40378035
17	MCDONALD'S	427	10 AVENUE	10001	##########	40370146

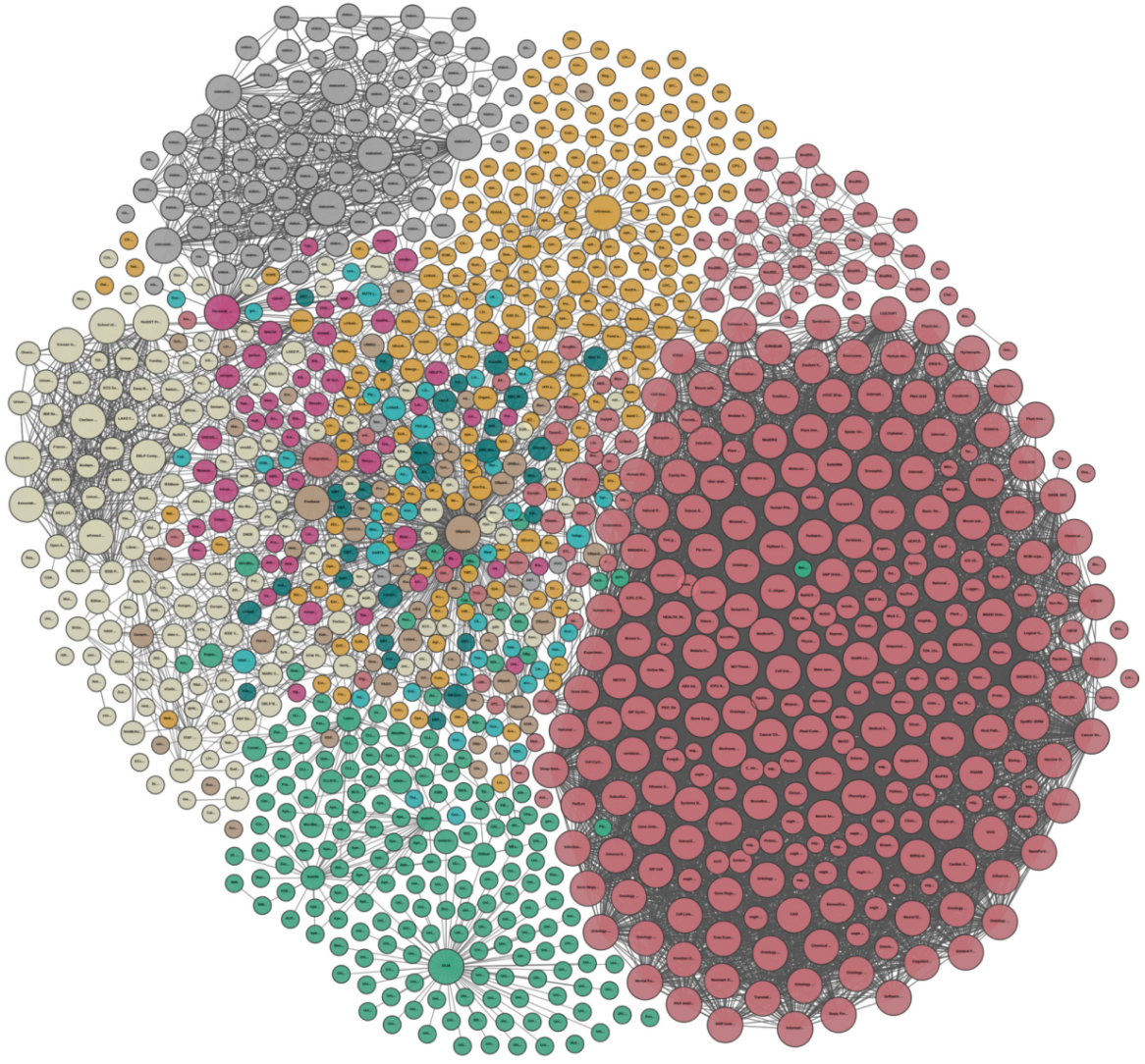
Semantic Web

- Vision: Move data from machine processable to machine understandable by integrating structured and unstructured web data and marking it up semantically
- Components:
  - Web data marked up so metadata is captured as annotations
  - Ontologies to make different data collections understandable
  - Logic-based technologies to access both metadata & ontologies

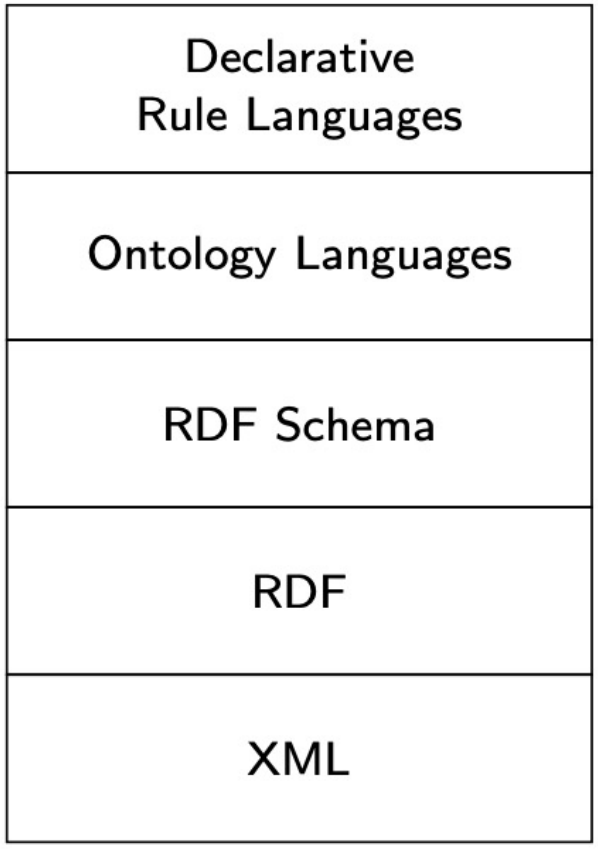
Linked Open Data (LOD)

- LOD is a realization and clarification of this vision → linkages among data is an important part of the vision
- Integrate data on the web based on four principles:
  1) All web sources (data) are locally identified by their URIs that serve as names
  2) These names are accessible by HTTP
  3) Information about data sources/entities are encoded as RDF (Resource Description Framework)
  4) Connections among datasets are established by data links

LOD Graph (status 2018)



Semantic Web Technologies



RDF: Resource Description Framework

- Data model on top of XML
- Models each fact as a set of triples
  - (subject, property (or predicate), object): ⟨s,p,o⟩
  - Subject: the entity that is described (URI or blank node)
  - Predicate: a feature of the entity (URI)
  - Object: value of the feature (URI, blank node or literal)
- Formally: an RDF triple

$$\langle s, p, o \rangle \in (\mathcal{U} \cup \mathcal{B}) \times \mathcal{U} \times (\mathcal{U} \cup \mathcal{B} \cup \mathcal{L})$$

  where
    - $\mathcal{U}$ denotes the set of URIs
    - $\mathcal{B}$ denotes the set of blank nodes
    - $\mathcal{L}$ denotes the set of literals
  - Set of RDF triples form a RDF dataset

RDF Schema (RDFS)

- Annotation of RDF data with semantic metadata
- Allows the definition of classes and class hierarchies
- Enables reasoning over RDF data (entailment)
- Built-in class definitions

RDF Query Language – SPARQL

- SPARQL Protocol and RDF Query Language
- Formally: Let $\mathcal{U}$, $\mathcal{B}$, $\mathcal{L}$, $\mathcal{V}$ denote set of all URIs, blank nodes, literals, and variables
  1) A triple pattern $(\mathcal{U} \cup \mathcal{B} \cup \mathcal{V}) \times (\mathcal{U} \cup \mathcal{V}) \times (\mathcal{U} \cup \mathcal{B} \cup \mathcal{L} \cup \mathcal{V})$ is a SPARQL expression
  2) If `P` is a SPARQL expression, then

     ```
     P FILTER R
     ```

     is also a SPARQL expression
  3) If `P1` & `P2` are SPARQL expressions, then

     ```
     P1 AND|OPT|OR P2
     ```

     are also SPARQL expressions

     2) and 3) are optional
- Basic graph pattern (BGP): set of triple patterns

RDF Data Management Approaches

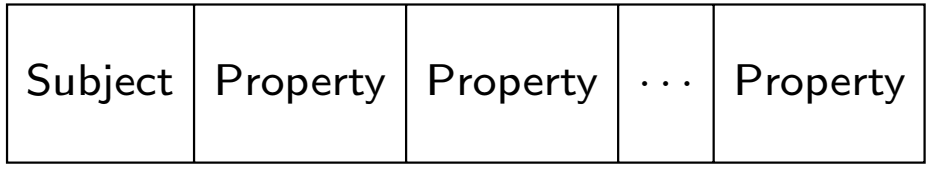
- Direct relational mapping
- Relational schema with extensive indexing
- Denormalize triples table into clustered properties
- Column-store organization
- Native graph representation

Single Table Exhaustive Indexing

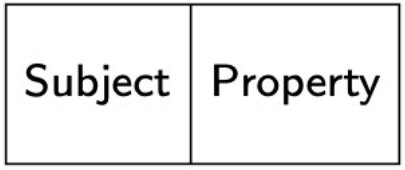
- Maintain a single table of RDF triples
- Create indexes for permutations of the three columns: SPO, SOP, PSO, POS, OPS, OSP
- RDF-3X and Hexastore
- Query processing
  - Each triple pattern can be evaluated by a range query
  - Joins between triple patterns computed using merge join
  - Join order is easy due to extensive indexing
- Advantages
  - Eliminates some of the joins { they become range queries
  - Merge join is easy and fast
- Disadvantages
  - Updates to indexes are expensive
  - Space usage

Property Tables

- Grouping by entities
- Jena
- Clustered property table: group together the properties that tend to occur in the same (or similar) subjects



Single-valued properties



Multi-valued properties

- Property class table: cluster the subjects with the same type of property into one property table

Property Tables

- DB2-RDF same approach but different structure
- Direct primary hash (DPH): organize by each subject with fixed number of (property, value) with spill
  - Properties in a given position might be different in different rows

Subject	Spill	$Prop_1$	val_1	$Prop_2$	val_2	$\cdots$	$Prop_k$	val_k

- Direct secondary hash (DSH): for multi-valued properties
  - DPH property value field has a pointer to DSH entry (*l_id*)

l_id	value

Property Tables Evaluation

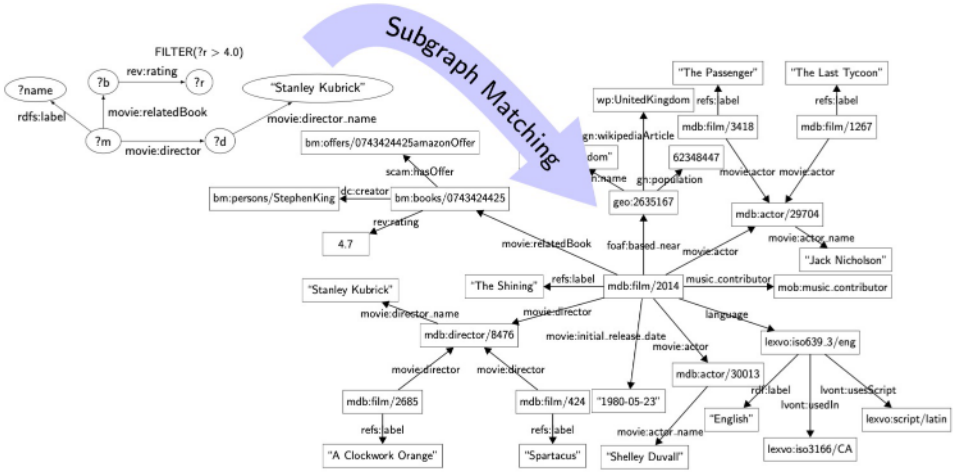
- Advantages
  - Star queries (subject-subject joins) become single table scans → fewer joins
- Disadvantages
  - Can have significant number of null values
  - Dealing with multi-valued properties requires special care
  - Is not very helpful for non-star queries
  - Clustering “similar” properties is non-trivial

Binary Tables

- Grouping by properties: For each property, build a two-column table, containing both subject and object, ordered by subjects
- *n* two-column tables
- Advantages
  - Advantages of column-oriented organization
  - Avoids null values
  - No need for (manual or automatic) clustering
  - Subject-subject joins by merge join
- Disadvantages
  - Insertions are expensive
  - More joins are required
  - Other join types do not benefit

Graph-based Processing

- Maintain graph structure of RDF data, convert SPARQL query to a query graph
- Answering SPARQL query ≡ subgraph matching using homomorphism
- gStore



- Advantages
  - Native representation → maintains the original graph structure
  - No restrictions on SPARQL queries
- Disadvantage
  - Subgraph matching is expensive

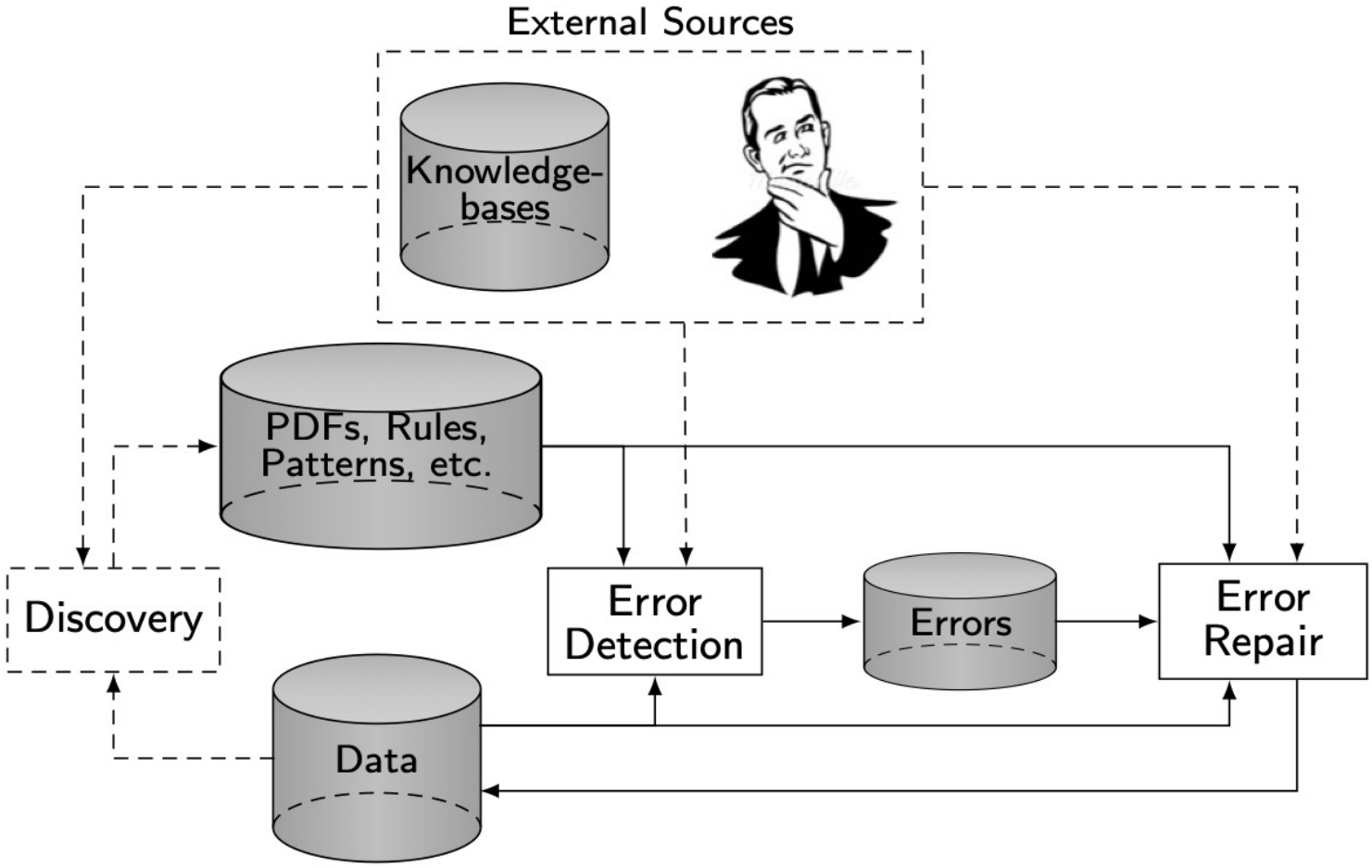
Federated RDF Systems

- Similar concerns to the relational counterparts
- Similar approaches
- Not all RDF data storage sites can execute SPARQL queries
  - SPARQL endpoints: storage sites that can execute queries
- Precompute metadata at endpoints → specify the capabilities
- Based on precomputation decompose a query and execute

Data Quality Issues

- More serious in web data integration
  - Data from far more sources
  - Data sources are not as well controlled
  - Lack of schema information in web data
    - Might have to use data-based techniques and not schema-based
- Data quality has two components
  - Data consistency
  - Veracity: authenticity and conformity of data with reality
    - Major challenge, but
    - High redundancy due to overlaps among data sources

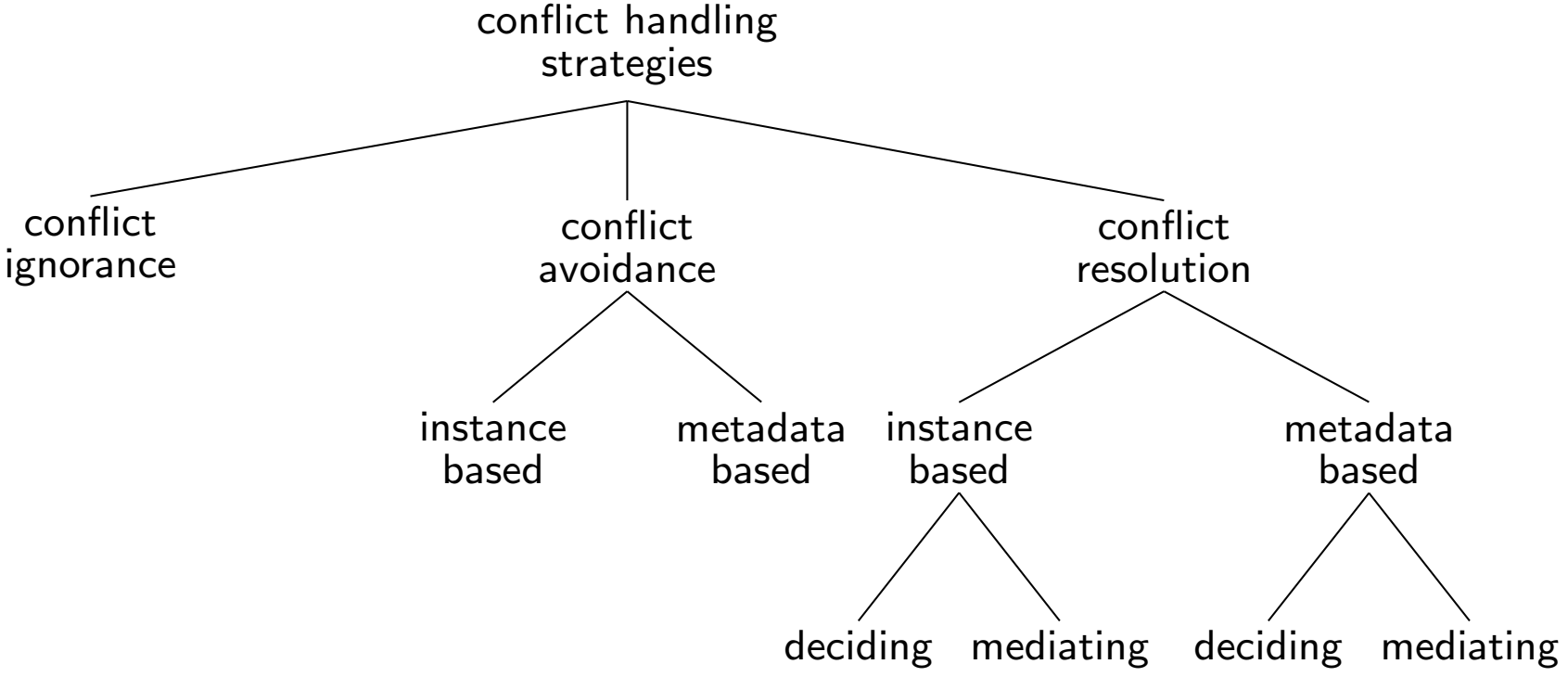
Cleaning Structured Web Data



Web Data Fusion

- Deciding the correct value for a data item that has differences among web sources
- Conflicts:
  - Uncertainty: one source has a non-null value, others have null value
    - Due to missing information (hence null values)
  - Contradiction: two or more non-null values of the same data item do not agree
- Typical enterprise data integration cleaning techniques may or may not work

Web Data Fusion Approaches



Web Source Quality

- Not all sources are equal
  - Cleaning techniques cannot assume all values to be of the same accuracy
- Web sources may copy from each other
  - Cannot ignore these dependencies
- Values can evolve over time
  - Incorrect value and outdated value are not the same thing