Principles of Distributed Database Systems

TS. Phan Thị Hà

Outline

- Introduction
- Distributed and Parallel Database Design
- Distributed Data Control
- Distributed Query Processing
- Distributed Transaction Processing
- Data Replication
- Database Integration – Multidatabase Systems
- Parallel Database Systems
- Peer-to-Peer Data Management
- Big Data Processing
- NoSQL, NewSQL and Polystores
- Web Data Management

Outline

- Web Data Management
  - Web Models
  - Web Search
  - Web Querying
  - Hidden Web
  - Web Data Integration

Web Overview

- Two segments
  - Publicly indexable web (PIW)
  - Deep web (hidden web)
- Most Internet users gain access to the web using search engines
  - Keyword search
  - Question-Answer (QA) systems
- Separate communities with little commonality

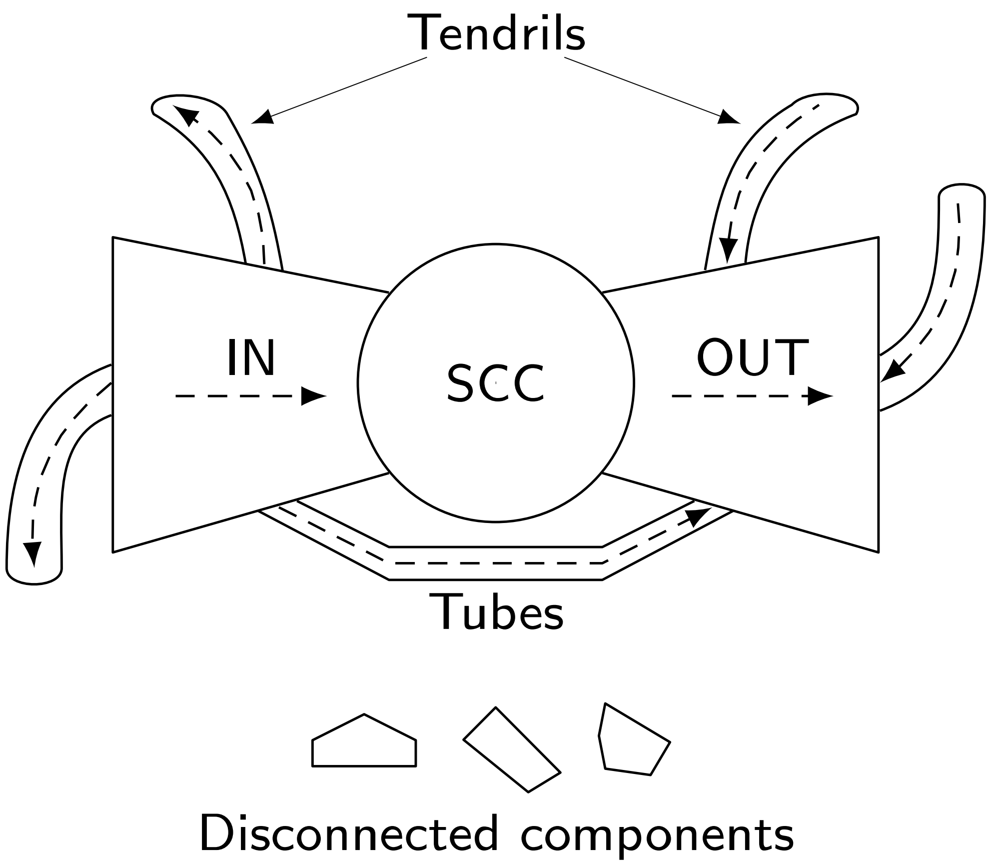
Outline

- Web Data Management
  - Web Models
  - Web Search
  - Web Querying
  - Hidden Web
  - Web Data Integration

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

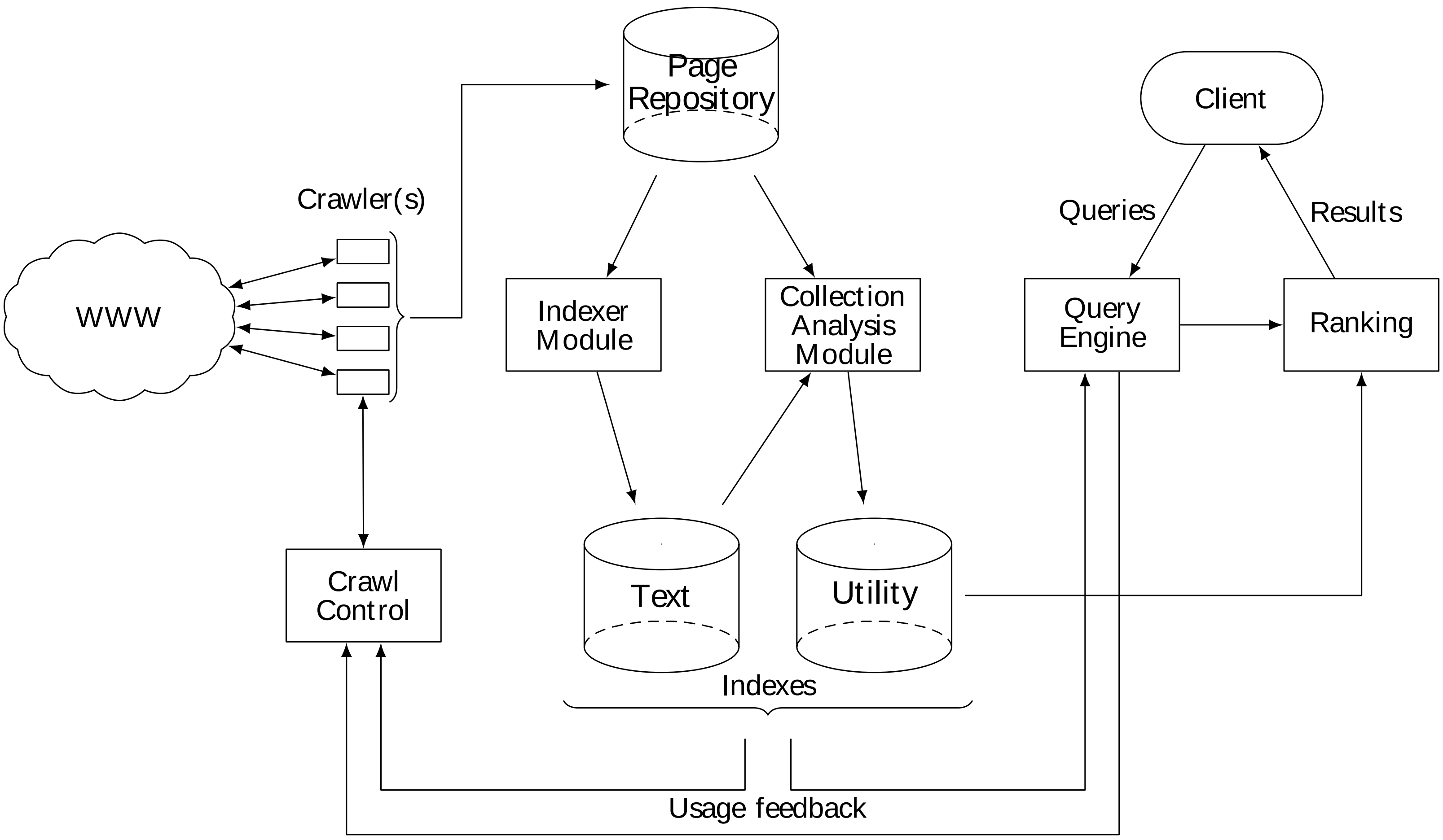
Structure of the Web



Outline

- Web Data Management
  - Web Models
  - Web Search
  - Web Querying
  - Hidden Web
  - Web Data Integration

Search Engine Architecture

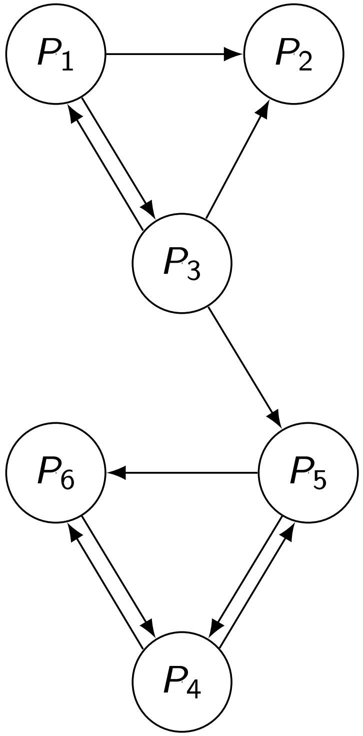


Web Crawling

- 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

Static Ranking – Recall PageRank

- Quality of a page determined by the number of incoming links and the importance of the pages of those links
- PageRank of page p_i:
  - B_{p_i}: back link pages of p_i (i.e., pages that point to p_i)
  - F_{p_i}: forward link pages of p_i (i.e., pages that p_i points to)

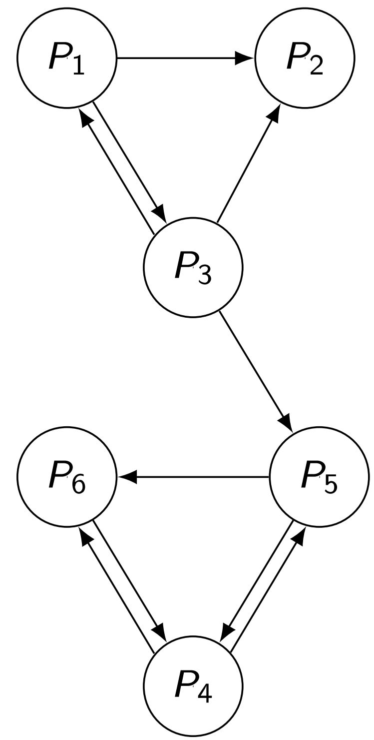


$$PR(P_i) = \sum_{P_j \in B_{P_i}} \frac{PR(P_j)}{|F_{P_j}|}$$

Ordering Metrics

- Breadth-first search
  - Visit URLs in the order they were discovered
- Random
  - Randomly choose one of the unvisited pages in the queue
- Incorporate importance metric
  - Random surfer model: On page P_i, choose one of the URLs on that page with equal probability d or jump to a random page with probability $(1-d)$

$$PR(P_i) = (1-d) + d \sum_{P_j \in B_{P_i}} \frac{PR(P_j)}{|F_{P_j}|}$$



Web Crawler Types

- Many Web pages change frequently, so the 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. It is important to coordinate these 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 vs. performance efficiency

Outline

- Web Data Management
  - Web Models
  - Web Search
  - Web Querying
  - Hidden Web
  - Web Data Integration

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.

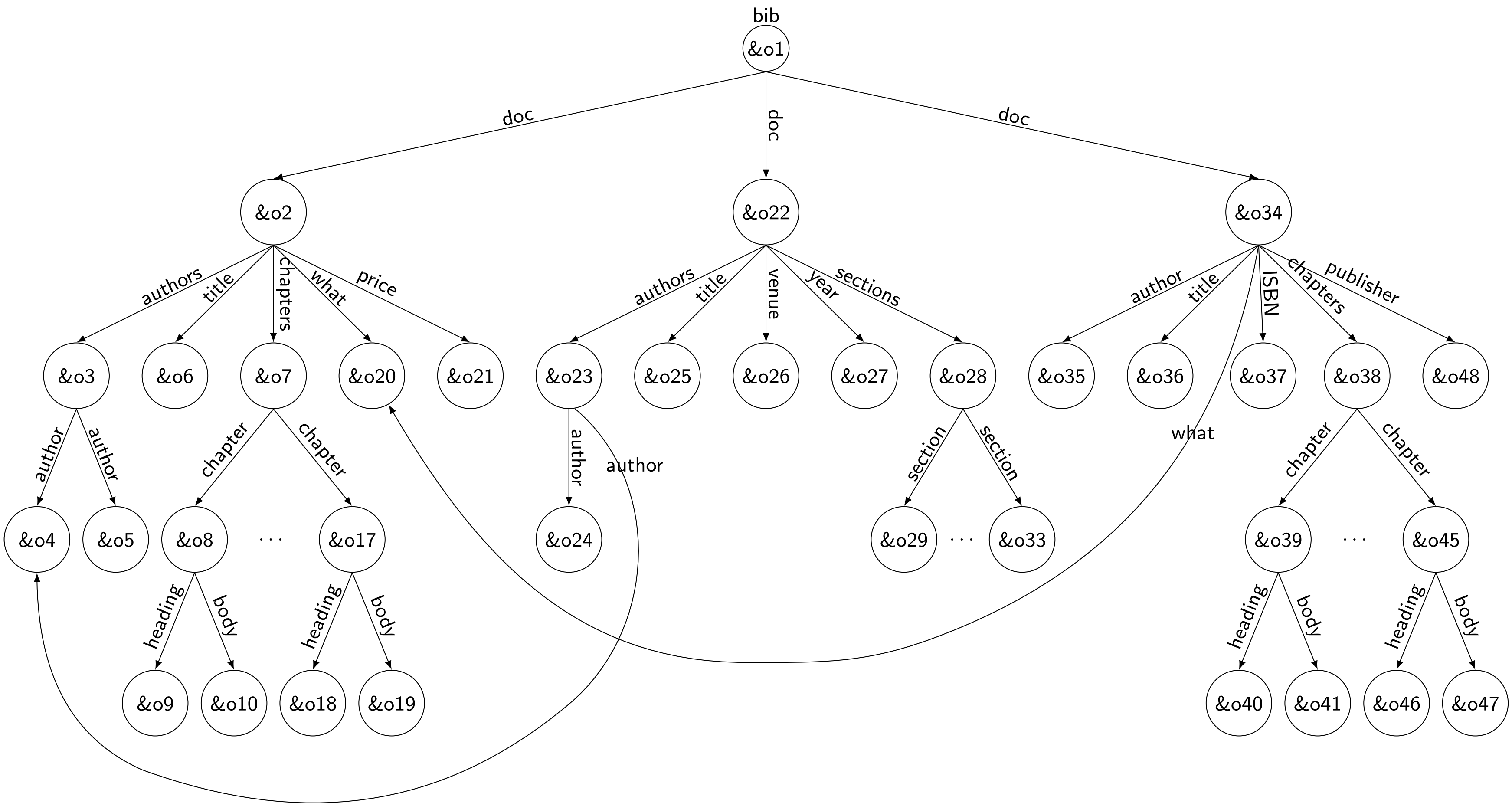
Approaches to Web Querying

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

Semistructured Data Querying

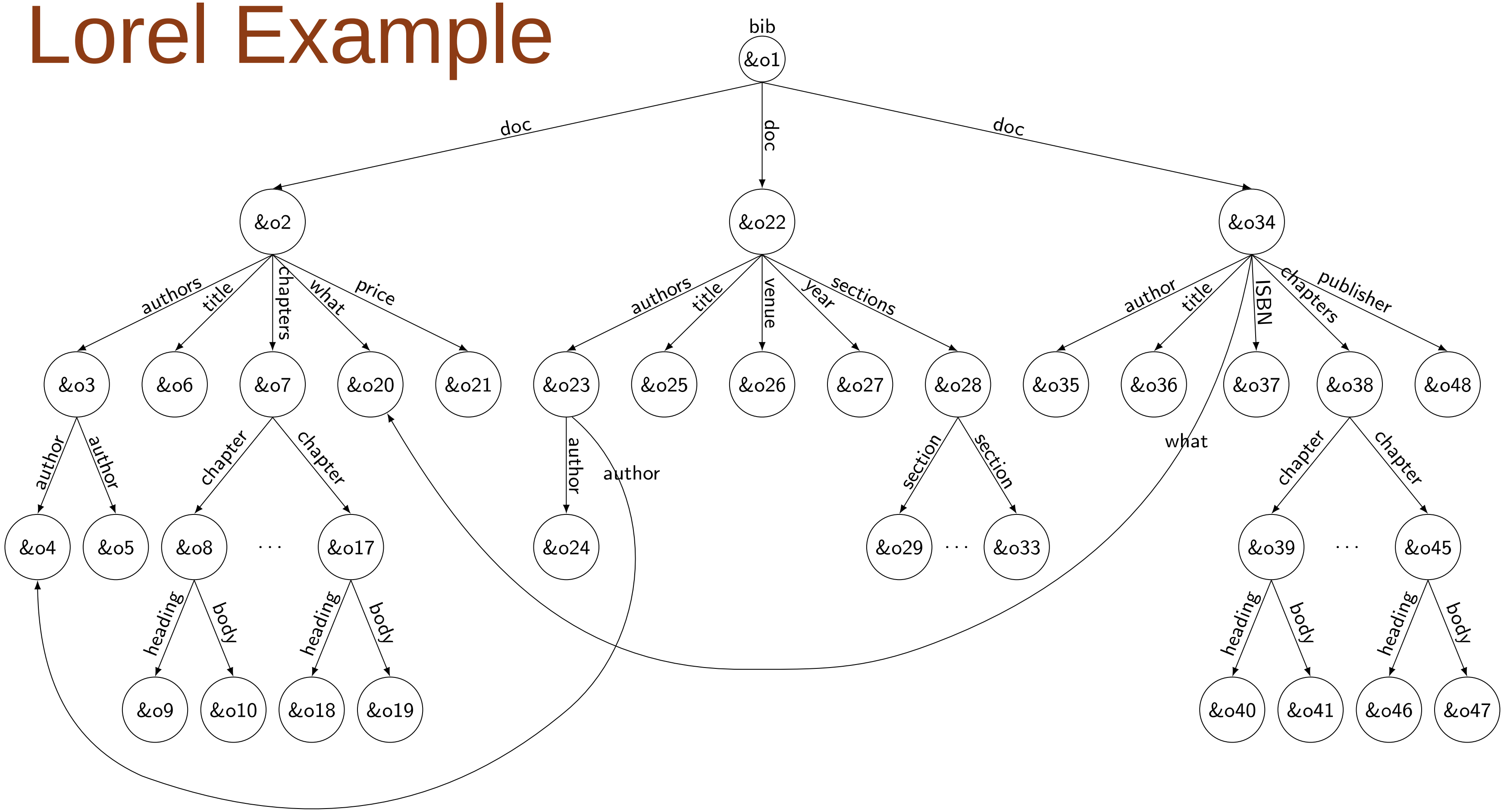
- Basic principle: Consider Web as a collection of semistructured data and use those techniques
- Uses an edge-labeled graph model of data
- Example systems & languages:
  - Lore/Lorel
  - UnQL
  - StruQL

OEM Model



bib
&o1
doc
doc
doc
&o2
&o22
&o34
authors
title
chapters
what
price
authors
title
venue
year
sections
author
title
ISBN
chapters
publisher
&o3
&o6
&o7
&o20
&o21
&o23
&o25
&o26
&o27
&o28
&o35
&o36
&o37
&o38
&o48
author
author
chapter
chapter
author
author
section
section
what
chapter
chapter
&o4
&o5
&o8
· · ·
&o17
&o24
&o29
· · ·
&o33
&o39
· · ·
&o45
heading
body
heading
body
heading
body
heading
body
&o9
&o10
&o18
&o19
&o40
&o41
&o46
&o47

Lorel Example



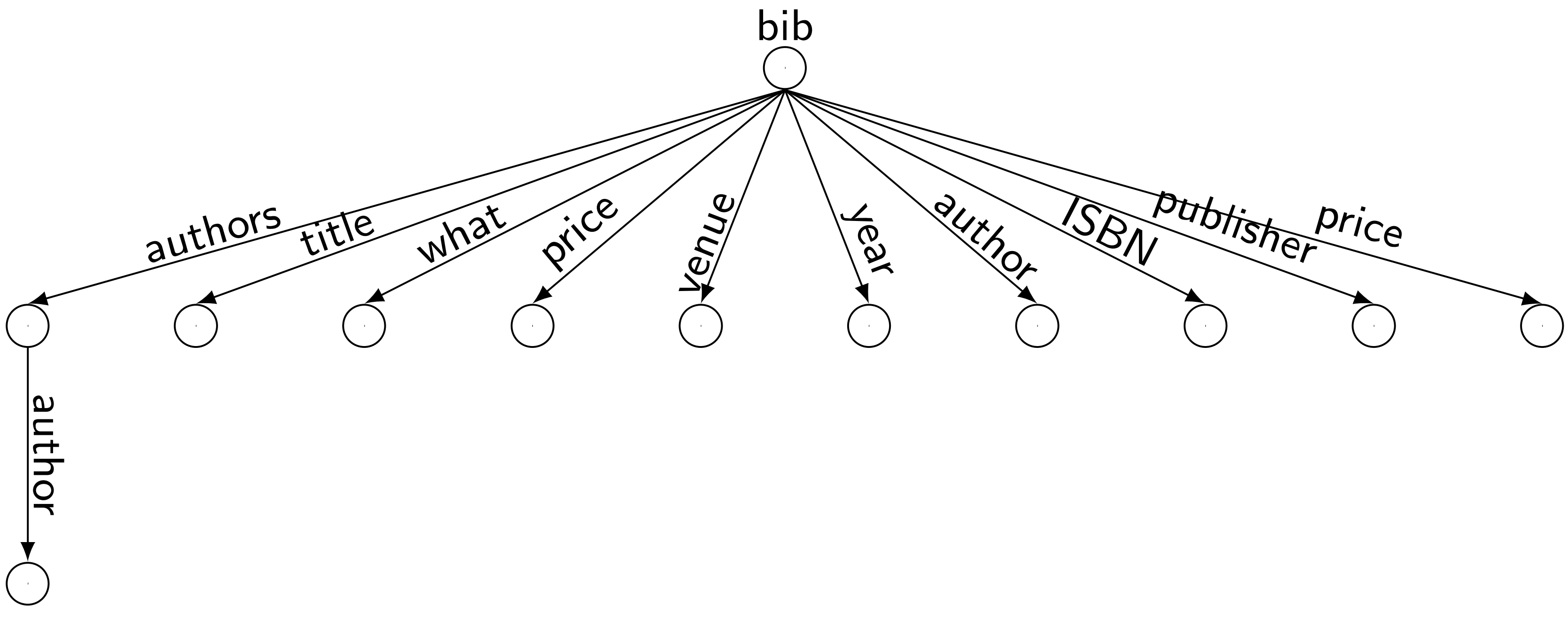
Find the authors of all books whose price is under $100

```
SELECT D(.authors)?.author
FROM   bib.doc D
WHERE  D.what = "Books" AND D.price < 100
```

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

DataGuide – Graph Summary



Web Query Languages

- Basic principle: Take into account the documents' content and internal structure as well as external links
- The 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

WebSQL Examples

```
DOCUMENT(URL,TITLE,TEXT,TYPE,LENGTH,MODIF)
LINK(BASE,HREF,LABEL)
```

- Simple search for all documents about “hypertext”

```
SELECT D.URL, D.TITLE
FROM   DOCUMENT D
           SUCH THAT D MENTIONS "hypertext"
WHERE  D.TYPE = "text/html"
```

- Find all links to applets from documents about “Java”

```
SELECT A.LABEL, A.HREF
FROM   DOCUMENT D
           SUCH THAT D MENTIONS "Java"
       ANCHOR A
           SUCH THAT BASE = X
WHERE  A.LABEL = "applet"
```

Demonstrates two scoping methods and a search for links.

WebSQL Examples (cont'd)

```
DOCUMENT(URL,TITLE,TEXT,TYPE,LENGTH,MODIF)
LINK(BASE,HREF,LABEL)
```

- Find documents that have string "database" in their title that are reachable from the ACM Digital Library home page through paths of length ≤ 2 containing only local links.

```
SELECT D.URL, D.TITLE
FROM   DOCUMENT D
            SUCH THAT "http://www.acm.org/dl"=|->|->-> D
WHERE  D.TITLE CONTAINS "database"
```

Demonstrates the use of different link types.

WebSQL Examples (cont'd)

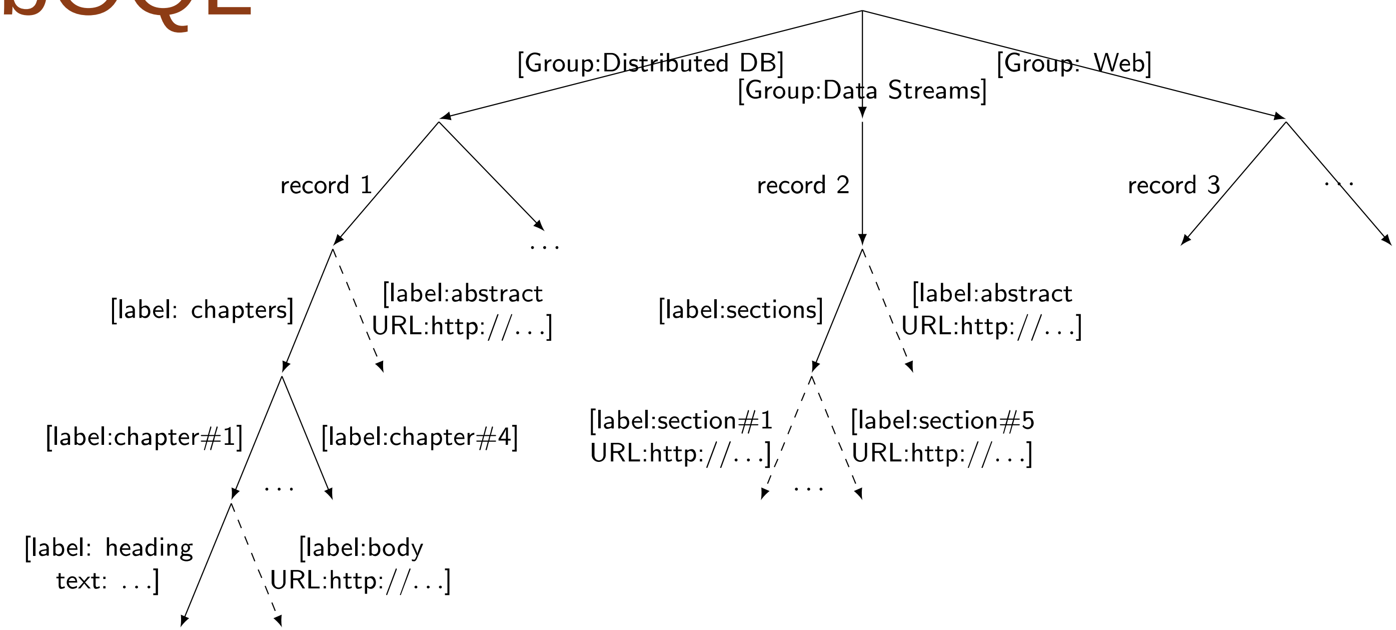
```
DOCUMENT(URL,TITLE,TEXT,TYPE,LENGTH,MODIF)
LINK(BASE,HREF,LABEL)
```

- Find documents mentioning "Computer Science" and all documents that are linked to them through paths of length ≤ 2 containing only local links

```
SELECT D1.URL, D1.TITLE, D2.URL, D2.TITLE
FROM   DOCUMENT D1
           SUCH THAT D1 MENTIONS "Computer Science"
       DOCUMENT D2
           SUCH THAT D1=|->|->-> D2
```

Demonstrates combination of content and structure specification in a query

WebOQL



record 1:
[authors: M. Tamer Ozsu, Patrick Valduriez
title: Principles of Distributed . . .
what: Book,
price:98.50]

record 2:
[authors: Lingling Yan, M. Tamer Ozsu
title: Mining Data Streams . . .
venue: CIKM,
year:2009]

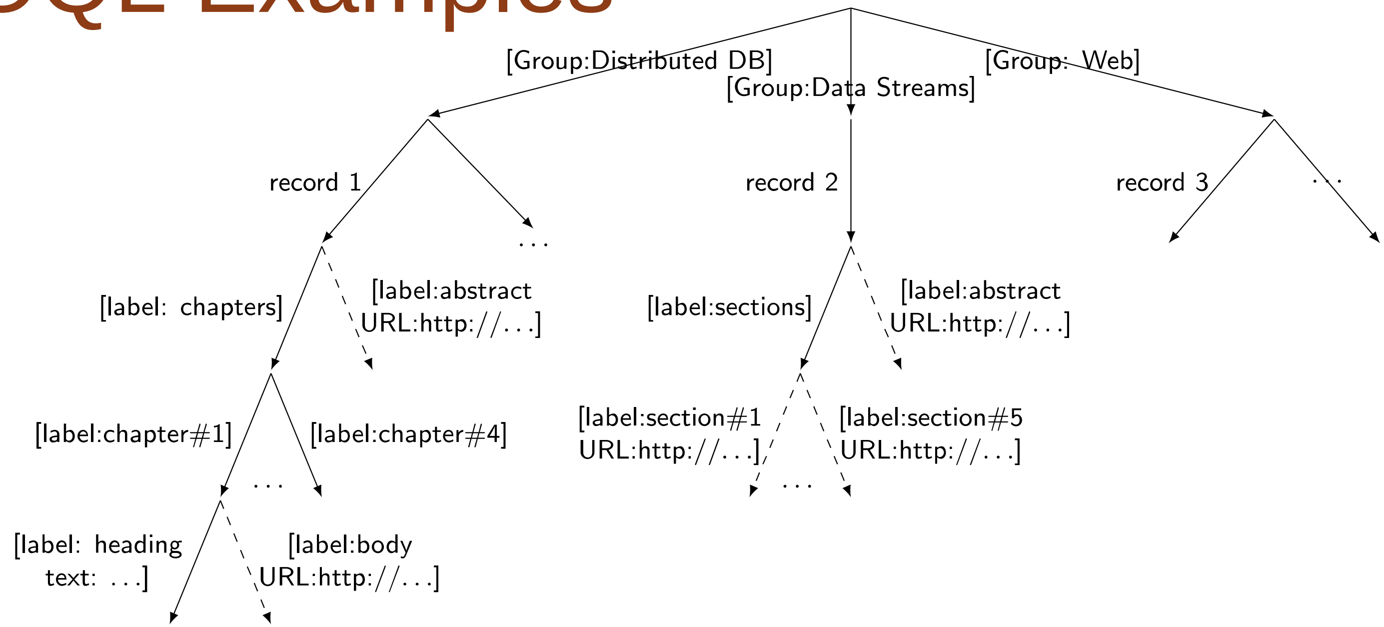
record 3:
[author: Anthony Bonato
title: A Course on the Web Graph
what: Book,
ISBN: TK5105.888.B667
publisher:AMS]

- Prime
- Peek
- Hang

- Concatenate
- Head
- Tail

- String pattern match (~)

WebOQL Examples



record 1:
[authors: M. Tamer Ozsu, Patrick Valduriez
title: Principles of Distributed ...
what: Book,
price:98.50]

record 2:
[authors: Lingling Yan, M. Tamer Ozsu
title: Mining Data Streams ...
venue: CIKM,
year:2009]

record 3:
[author: Anthony Bonato
title: A Course on the Web Graph
what: Book,
ISBN: TK5105.888.B667
publisher:AMS]

Find the titles and abstracts of all documents authored by "Ozsu"

```
SELECT [y.title, y'.URL]
FROM   x IN dbDocuments, y in x'
WHERE  y.authors ~ "Ozsu"
```

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

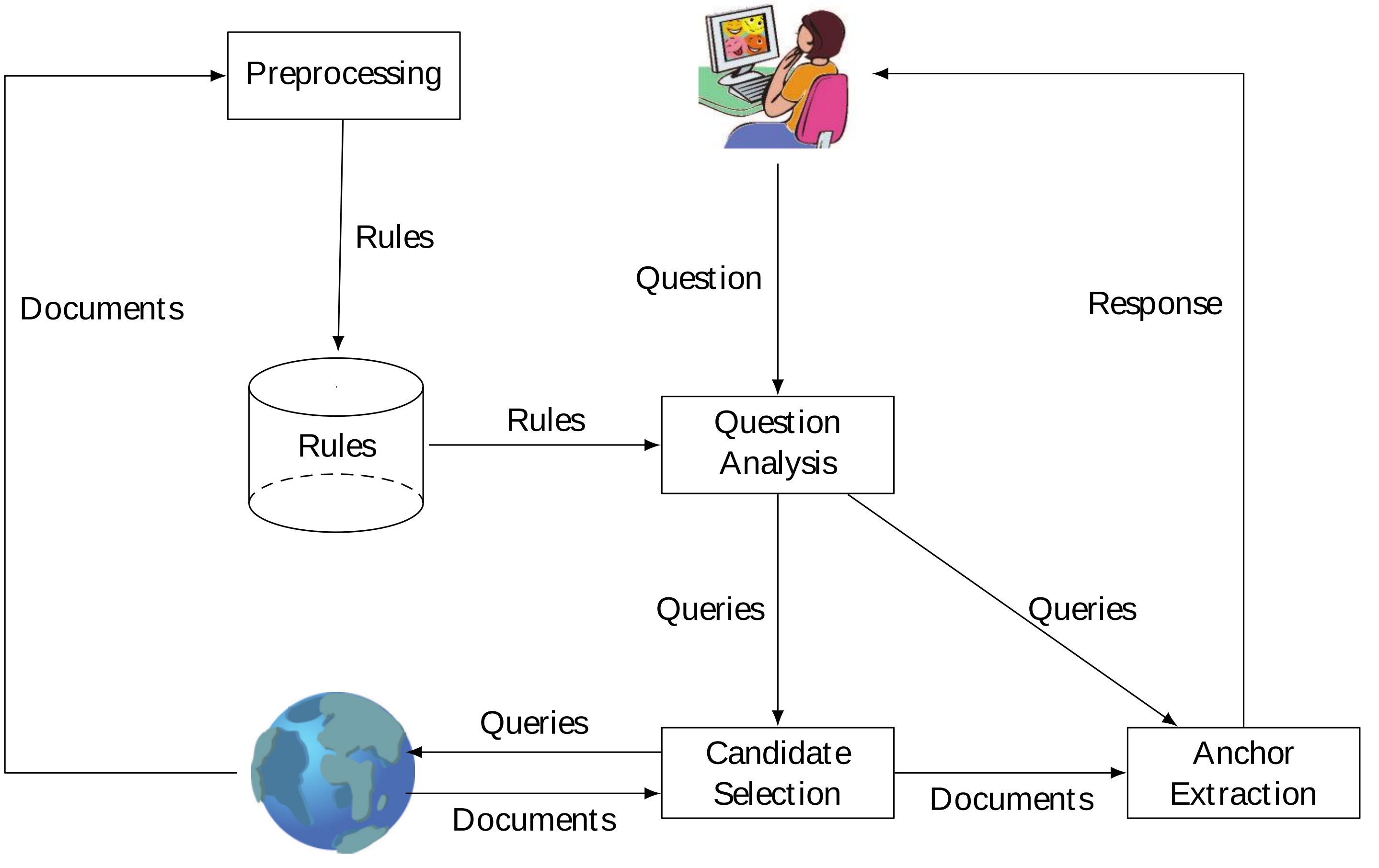
Question-Answer Approach

- 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

QA 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



Outline

- Web Data Management
  - Web Models
  - Web Search
  - Web Querying
  - Hidden Web
  - Web Data Integration

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

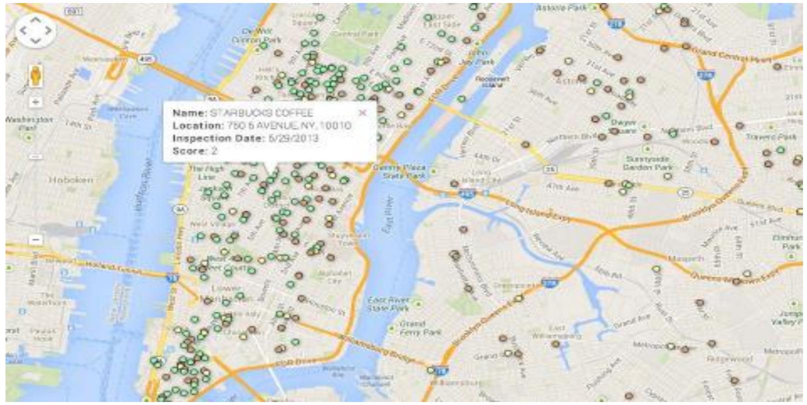
Outline

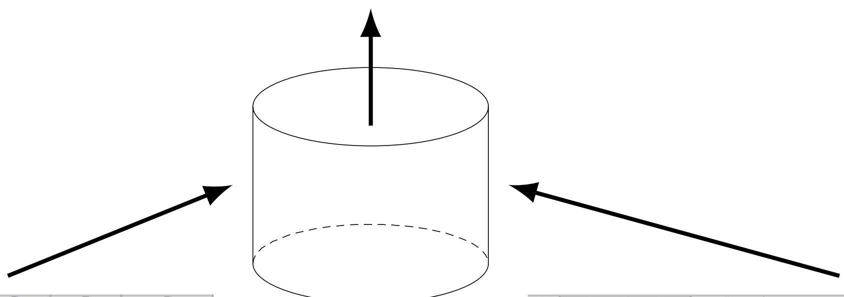
- Web Data Management
  - Web Models
  - Web Search
  - Web Querying
  - Hidden Web
  - Web Data Integration

Web Data Integration

- "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	3/11/2013	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

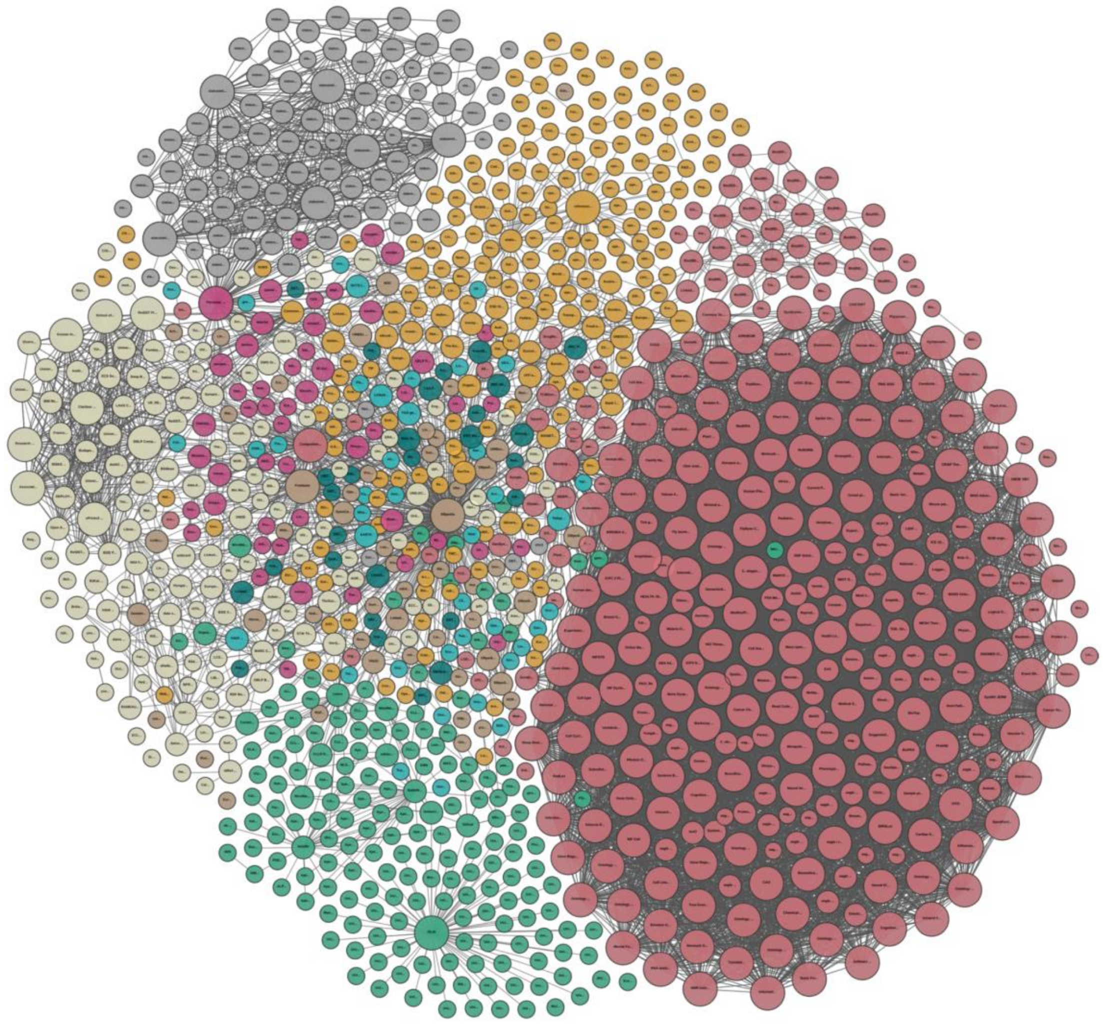
Semantic Web

- 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

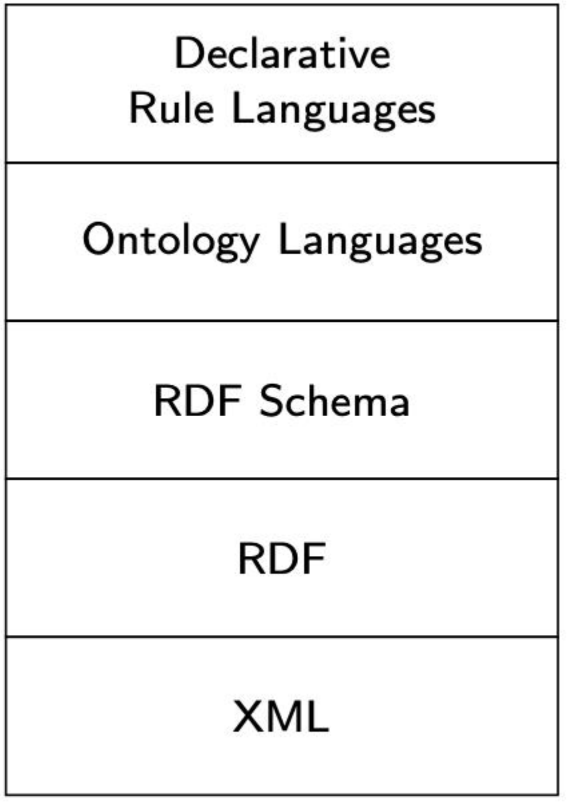
Linked Open Data (LOD)

- 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 (as of 2018)



Semantic Web Technologies



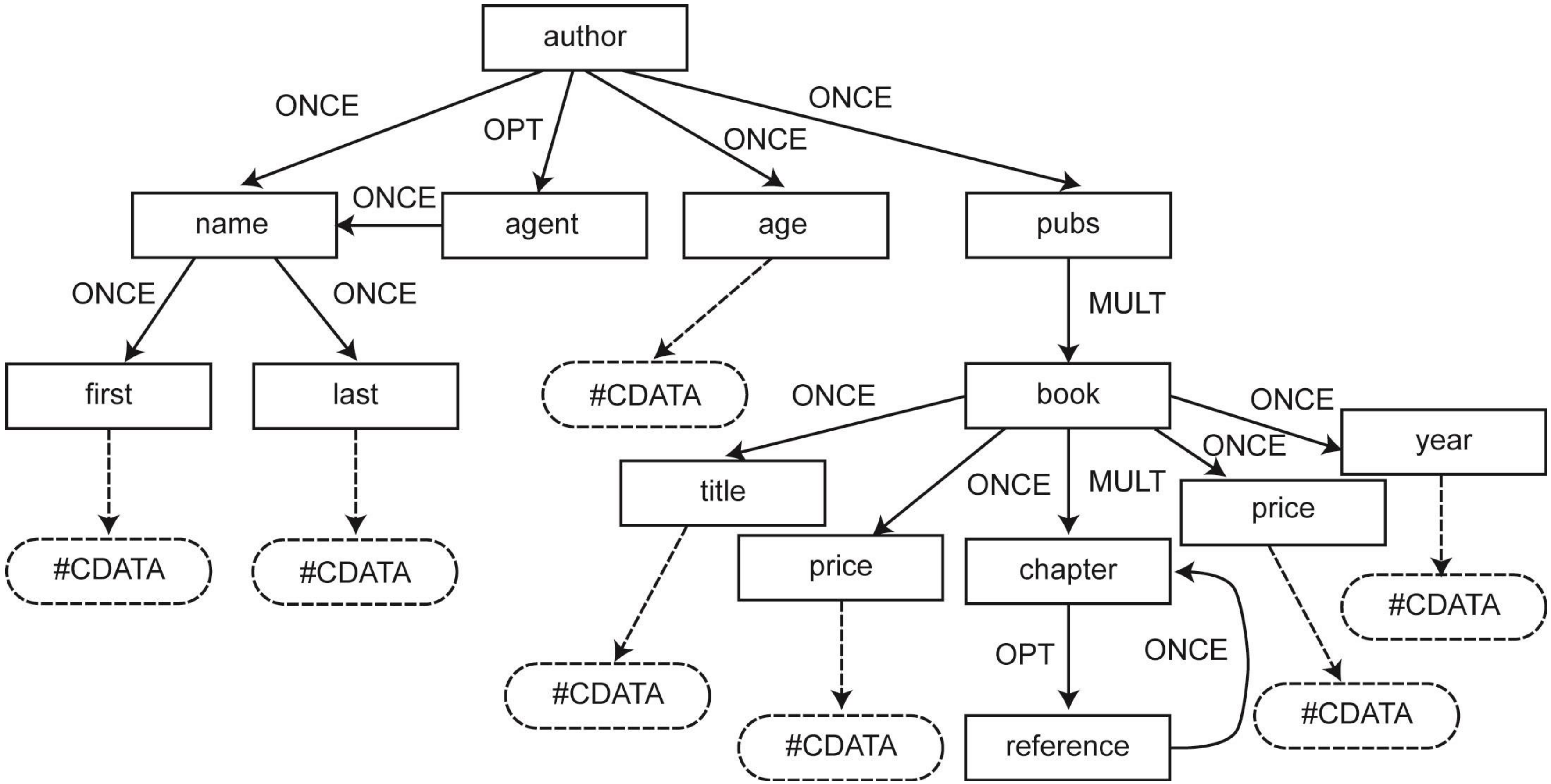
XML Overview

- 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 Schema

- Can be represented by a Document Type Definition (DTD) or XMLSchema
- Simpler definition using a schema graph: 5-tuple $\langle\Sigma, \Psi, s, m, \rangle\rangle$
  - Σ is an alphabet of XML document node types
  - $\Psi \subseteq \Sigma \times \Sigma$ is a set of edges between node types
    - e=(⌠$_1$, ⌠$_2$) $\in \Psi$ denotes item of type ⌠$_1$ may contain an item of type ⌠$_2$
  - s: $\Psi \rightarrow$ {ONCE, OPT, MULT}
    - ONCE: item of type ⌠$_1$ must contain exactly one item of type ⌠$_2$
    - OPT: item of type ⌠$_1$ may or may not contain an item of type ⌠$_2$
    - MULT: item of type ⌠$_1$ may contain multiple items of type ⌠$_2$
  - m: $\Sigma \rightarrow$ {`string`}
    - m(⌠) denotes the domain of text content of an item of type ⌠
  - $\rangle$ is the root node type

Example



author
ONCE
OPT
ONCE
ONCE
name
ONCE
agent
age
pubs
ONCE
ONCE
MULT
first
last
#CDATA
ONCE
book
ONCE
ONCE
year
title
ONCE
MULT
price
price
chapter
#CDATA
#CDATA
#CDATA
OPT
ONCE
#CDATA
#CDATA
#CDATA
reference
#CDATA

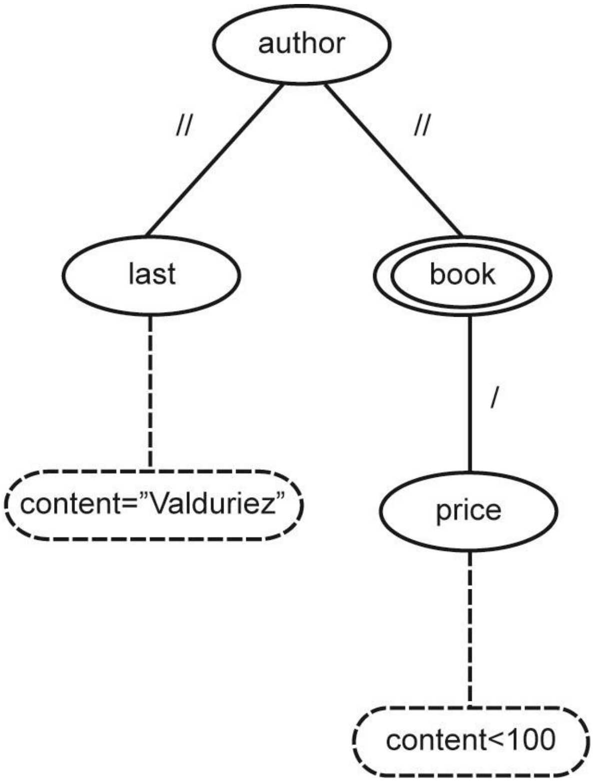
XML Query Languages

- Xpath
  - Based on path expressions
  - A list of steps
  - Each step consists of
    - An axis (13 of them) →
    - A name test
    - Zero or more qualifiers
    - Last step is called a return step
- Xquery
  - FLWOR expression
    - “for”, “let”, “where”, “order by”, “return” clauses
    - Each clause can reference path expressions or other FLWOR expressions
    - Similar to SQL select-from-where-aggregate but operating on a list of XML document tree nodes

child (/)
descendent
descendent-or-self (//)
parent
attribute (/@)
self (.)
ancestor
ancestor-or-self
following-sibling
following
preceding-sibling
preceding
namespace

XPath Example

- /author[.//last = "Valduriez"]//book[price < 100]
- Name constraints
  - Correspond to name tests
- Structural constraints
  - Correspond to axes
- Value constraints
  - Correspond to value comp
- Query pattern tree (QTP)



Xquery Example

- Return a list of books with their title and price ordered by their attribute names

```
let $col := collection(“bib”)
for $author in $col/author
         order by $author.name
         for $b in $author/pubs/book
               let $title := $b/title
               let $price := $b/price
               return $title, $price
```

Resource Description Framework (RDF)

- 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 Dataset Example

Prefixes: mdb=http://data.linkedmdb.org/resource/; geo=http://sws.geonames.org/
bm=http://wifo5-03.informatik.uni-mannheim.de/bookmashup/
lexvo=http://lexvo.org/id/;wp=http://en.wikipedia.org/wiki/

URI

Literal

URI

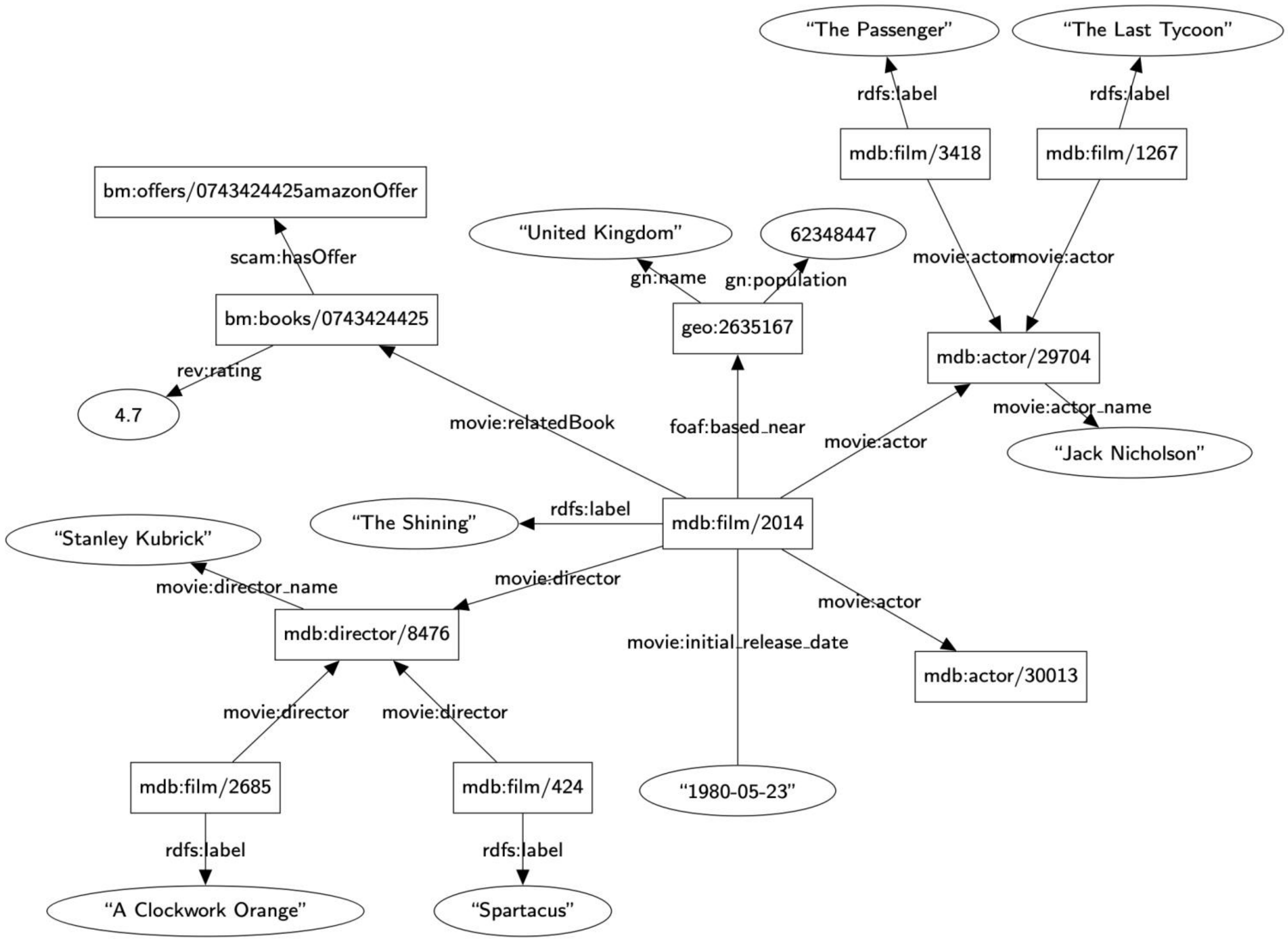
Subject	Predicate	Object
mdb: film/2014	rdfs:label	"The Shining"
mdb:film/2014	movie:initial_release_date	"1980-05-23" '
mdb:film/2014	movie:director	mdb:director/8476
mdb:film/2014	movie:actor	mdb:actor/29704
mdb:film/2014	movie:actor	mdb: actor/30013
mdb:film/2014	movie:music_contributor	mdb: music_contributor/4110
mdb:film/2014	foaf:based_near	geo:2635167
mdb:film/2014	movie:relatedBook	bm:0743424425
mdb:film/2014	movie:language	lexvo:iso639-3/eng
mdb:director/8476	movie:director_name	"Stanley Kubrick"
mdb:film/2685	movie:director	mdb:director/8476
mdb:film/2685	rdfs:label	"A Clockwork Orange"
mdb:film/424	movie:director	mdb:director/8476
mdb:film/424	rdfs:label	"Spartacus"
mdb:actor/29704	movie:actor_name	"Jack Nicholson"
mdb:film/1267	movie:actor	mdb:actor/29704
mdb:film/1267	rdfs:label	"The Last Tycoon"
mdb:film/3418	movie:actor	mdb:actor/29704
mdb:film/3418	rdfs:label	"The Passenger"
geo:2635167	gn:name	"United Kingdom"
geo:2635167	gn:population	62348447
geo:2635167	gn:wikipediaArticle	wp:United_Kingdom
bm:books/0743424425	dc:creator	bm:persons/Stephen+King
bm:books/0743424425	rev:rating	4.7
bm:books/0743424425	scom:hasOffer	bm:offers/0743424425amazonOffer
lexvo:iso639-3/eng	rdfs:label	"English"
lexvo:iso639-3/eng	lvont:usedIn	lexvo:iso3166/CA
lexvo:iso639-3/eng	lvont:usesScript	lexvo:script/Latn

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
  - `rdfs:Class`, `rdfs:subClassOf`, `rdfs:label`, …
- Example

```
Movies rdf:type rdfs:Class.
ActionMovies rdfs:subClassOf Movies.
Dramas rdfs:subClassOf Movies.
```

RDF Graph


"The Passenger"
"The Last Tycoon"
rdfs:label
rdfs:label
mdb:film/3418
mdb:film/1267
bm:offers/0743424425amazonOffer
"United Kingdom"
62348447
scam:hasOffer
gn:name
gn:population
movie:actor
movie:actor
bm:books/0743424425
geo:2635167
mdb:actor/29704
rev:rating
4.7
movie:relatedBook
foaf:based_near
movie:actor
movie:actor_name
"Jack Nicholson"
rdfs:label
"The Shining"
mdb:film/2014
"Stanley Kubrick"
movie:director_name
movie:director
movie:actor
mdb:director/8476
movie:initial_release_date
mdb:actor/30013
movie:director
movie:director
mdb:film/2685
mdb:film/424
"1980-05-23"
rdfs:label
rdfs:label
"A Clockwork Orange"
"Spartacus"

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
     ```

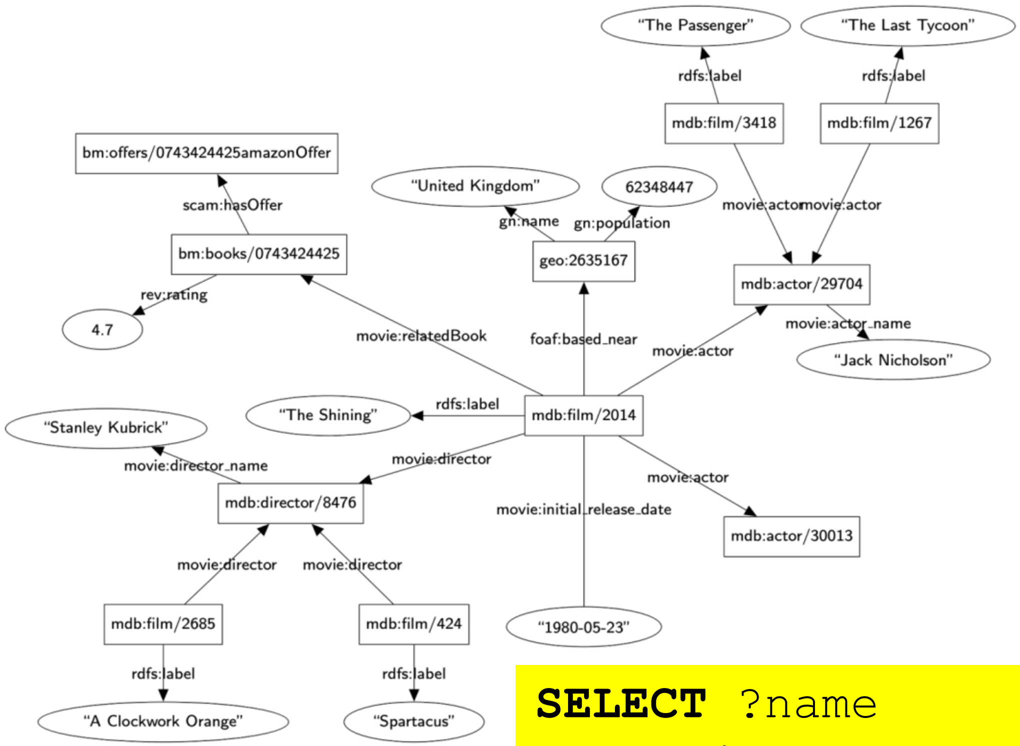
     is also a SPARQL expression
  1) If `P1` & `P2` are SPARQL expressions, then

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

     are also SPARQL expressions

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

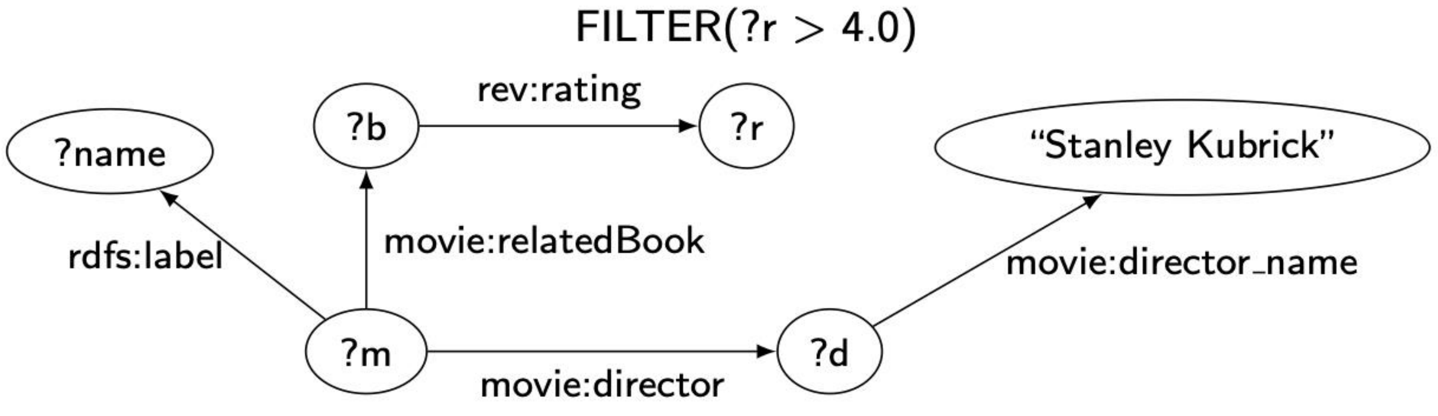
SPARQL Example



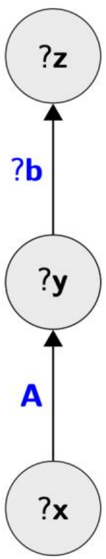

```
SELECT ?name
WHERE {
?m rdfs:label ?name. ?m movie:director ?d
?d movie:director_name "Stanley Kubrick".
?m movie:relatedBook ?b. ?b rev:rating ?r.
FILTER(?r > 4.0)
}
```

SPARQL Query Graph

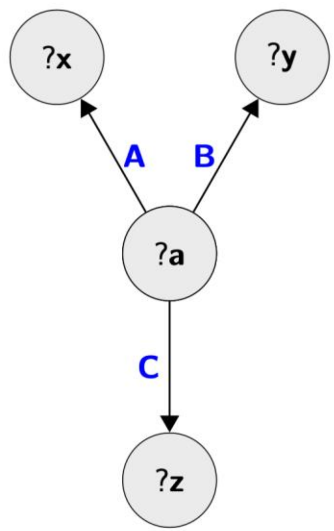
```
SELECT ?name
WHERE {
?m rdfs:label ?name. ?m movie:director ?d
?d movie:director_name "Stanley Kubrick".
?m movie:relatedBook ?b. ?b rev:rating ?r.
FILTER(?r > 4.0)
}
```



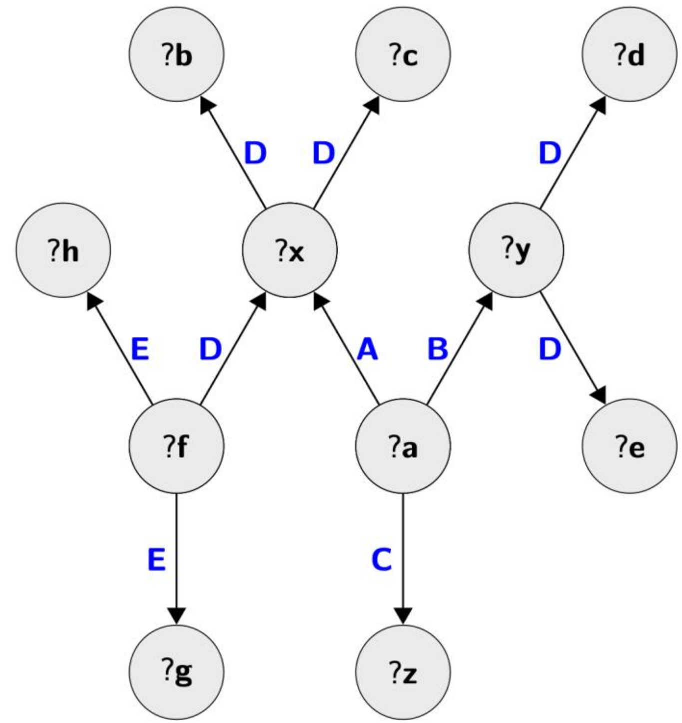
SPARQL Query Types



Linear



Star



Snowflake

RDF Data Management Approaches

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

Direct Relational Mapping

```
SELECT ?name
WHERE {
?m rdfs:label ?name. ?m movie:director ?d
?d movie:director_name "Stanley Kubrick".
?m movie:relatedBook ?b. ?b rev:rating ?r.
FILTER(?r > 4.0)
}
```

Subject	Property	Object
mdb:film/2014	rdfs:label	"The Shining"
mdb:film/2014	movie:initial_release_date	"1980-05-23"
mdb:film/2014	movie:director	mdb:director/8476
mdb:film/2014	movie:actor	mdb:actor/29704
mdb:film/2014	movie:actor	mdb: actor/30013
mdb:film/2014	movie:music_contributor	mdb: music_contributor/4110
mdb:film/2014	foaf:based_near	geo:2635167
mdb:film/2014	movie:relatedBook	bm:0743424425
mdb:film/2014	movie:language	lexvo:iso639-3/eng
mdb:director/8476	movie:director_name	"Stanley Kubrick"
mdb:film/2685	movie:director	mdb:director/8476
mdb:film/2685	rdfs:label	"A Clockwork Orange"
mdb:film/424	movie:director	mdb:director/8476
mdb:film/424	rdfs:label	"Spartacus"
mdb:actor/29704	movie:actor_name	"Jack Nicholson"
mdb:film/1267	movie:actor	mdb:actor/29704
mdb:film/1267	rdfs:label	"The Last Tycoon"
mdb:film/3418	movie:actor	mdb:actor/29704
mdb:film/3418	rdfs:label	"The Passenger"
geo:2635167	gn:name	"United Kingdom"
geo:2635167	gn:population	62348447
geo:2635167	gn:wikipediaArticle	wp:United_Kingdom
bm:books/0743424425	dc:creator	bm:persons/Stephen+King
bm:books/0743424425	rev:rating	4.7
bm:books/0743424425	scom:hasOffer	bm:offers/0743424425amazonOffer
lexvo:iso639-3/eng	rdfs:label	"English"
lexvo:iso639-3/eng	lvont:usedIn	lexvo:iso3166/CA
lexvo:iso639-3/eng	lvont:usesScript	lexvo:script/Latn

```
SELECT T1.object
FROM T AS T1, T AS T2, T AS T3,
   T AS T4, T AS T5
WHERE T1.p="rdfs:label"
AND T2.p="movie:relatedBook"
AND T3.p=:movie:director"
AND T4.p="rev:rating"
AND T5.p="movie:director_name"
AND T1.s=T2.s
AND T1.s=T3.s
AND T2.o=T4.s
AND T3.o=T5.s
AND T4.o>4.0
AND T5.o="Stanley Kubrick"
```

Too many self-joins

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

Subject	Property	Property	···	Property

Single-valued properties

Subject	Property

Multi-valued properties

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

Subject	Property	Property	···	Property	Type

Property Tables (cont'd)

- 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

Subject	Object
film/2014	“The Shining”
film/2685	“A Clockwork Orange”
film/424	“Spartacus”
film/1267	“The Last Tycoon”
film/3418	“The Passenger”
iso639-3/eng	“English”

(a) rdfs:label

Subject	Object
film/2014	actor/29704
film/2014	actor/30013
film/1267	actor/29704
film/3418	actor/29704

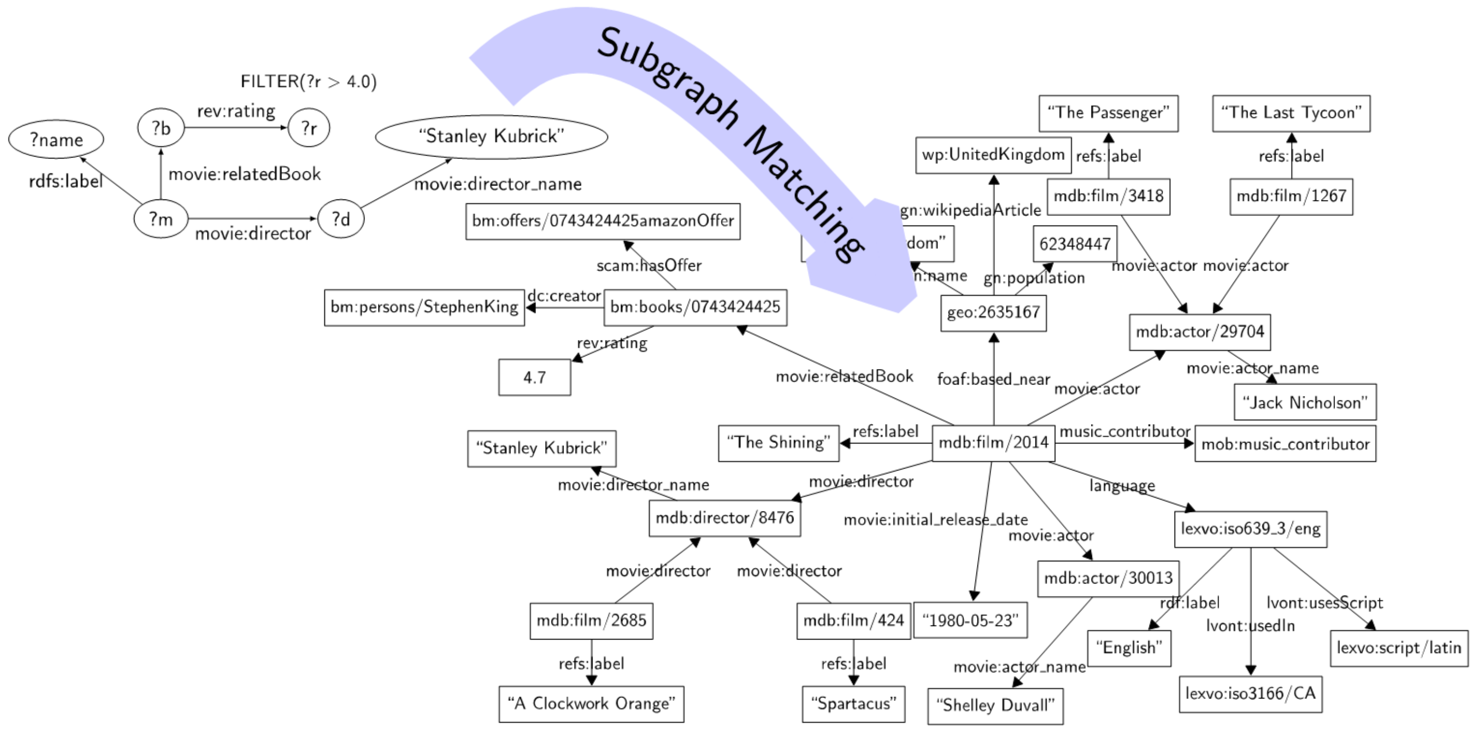
(b) movie:actor

Binary Tables Evaluation

- 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



Graph-based Processing Evaluation

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

Distributed RDF Systems

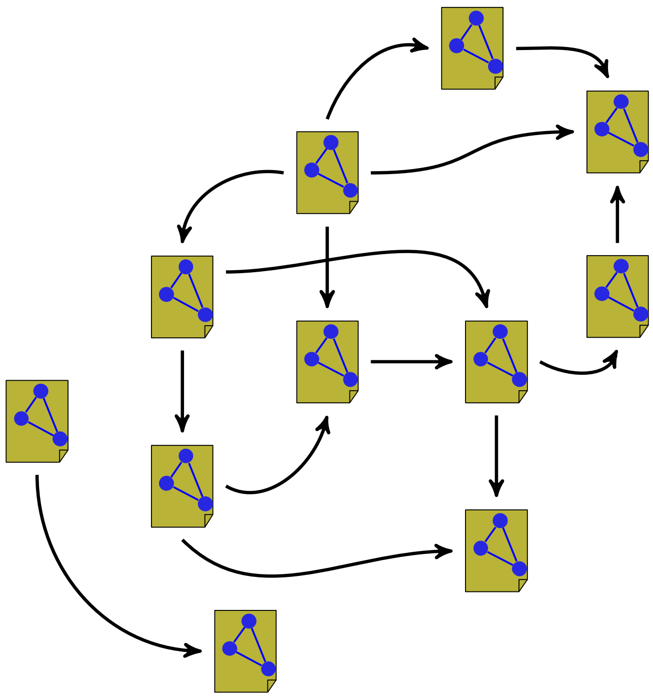
- Divide RDF graph G into set of fragments $\{G_1, \ldots, G_n\}$
- MapReduce-based solutions
  - Store $\{G_1, \ldots, G_n\}$ in HDFS
  - Each triple pattern is evaluated as a MapReduce scan job
  - Then join using a MapReduce join implementation
- Distributed query processing approaches as in book
  - Also divide SPARQL query Q into set of subqueries $\{Q_1, \ldots, Q_k\}$
  - Evaluate $\{Q_1, \ldots, Q_k\}$ over $\{G_1, \ldots, G_n\}$ using known techniques
- Partial query evaluation
  - Query Q is evaluated at each site where G_i exists
  - Get partial local matches
  - Join using a distributed join algorithm

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

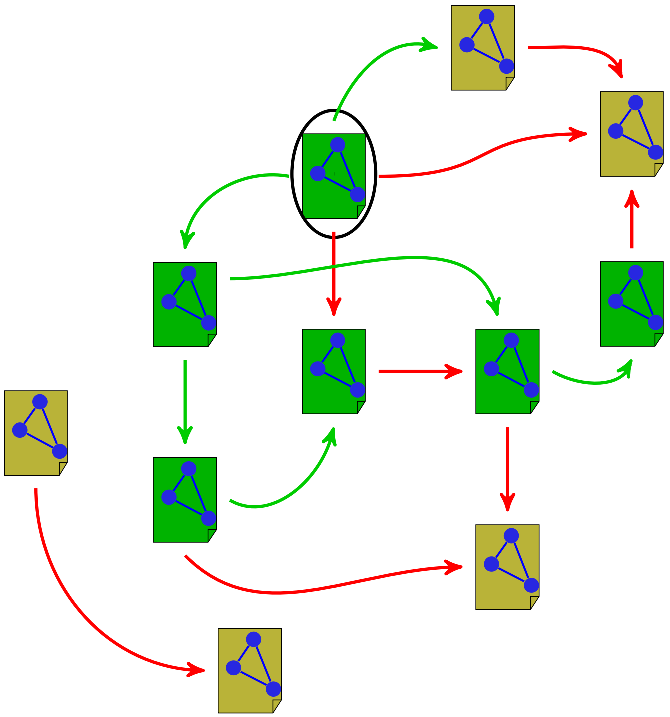
Linked Open Data (LOD)

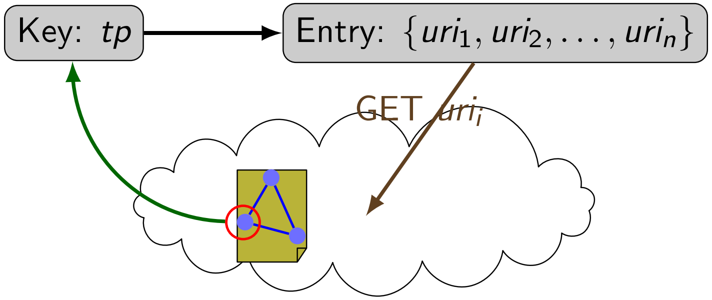
- Consists of a set of web documents each with embedded RDF triples that encode web resources
- Full web semantics: Scope of evaluating a SPARQL expression is all Linked Data
  - Query result completeness cannot be guaranteed by any (terminating) execution
- Reachability-based semantics: Scope of a query is all data along paths of data links that satisfy the condition
  - Given a set of seed URIs & reachability condition, scope is all data along paths of data links reachable from the seeds & satisfying the reachability condition
  - Computationally feasible



SPARQL Over LOD

- Traversal approach: Discover relevant URIs recursively by traversing (specic) data links at query execution runtime
  - Implements reachability-based query semantics
- Index-based approach: Use an index to determine relevant URIs
  - Different index keys possible; e.g., triple patterns
- Hybrid approach: Perform a traversal-based execution using a prioritized list of URIs to look up
  - Initial seed from the pre-populated index

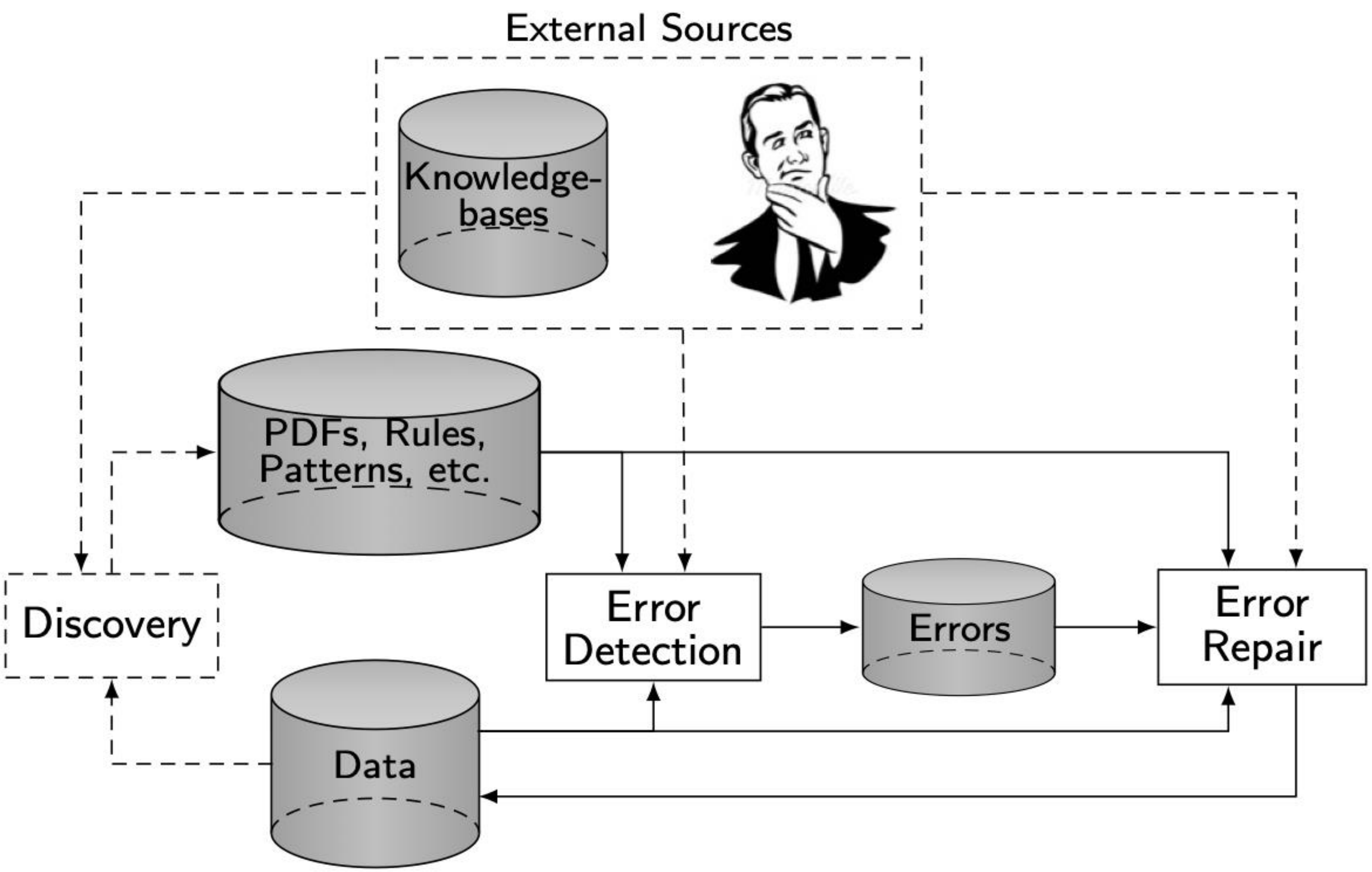




Data Quality Issues

- 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



Cleaning Structured Web Data (cont'd)

- Process works well when
  - Rich set of metadata exists
  - Enogh examples exist for automatic algorithms
- May not be true in web tables

Sevilla - Jerez de la Frontera-Cádiz	1861
Córdoba - Málaga	1865.
Bobadilla - Granada	1874
Córdoba - Bélmez	1874
Osuna	La Roda

Polaco	15.04.1983	194	84
Vini	29.09.1982	N/A	N/A
Caiao	30/11/1982	N/A	N/A
Jairo	17.02.1990	N/A	N/A
Michael	20.04.1983	N/A	N/A
Ricardinho	19.11.1975	192	94

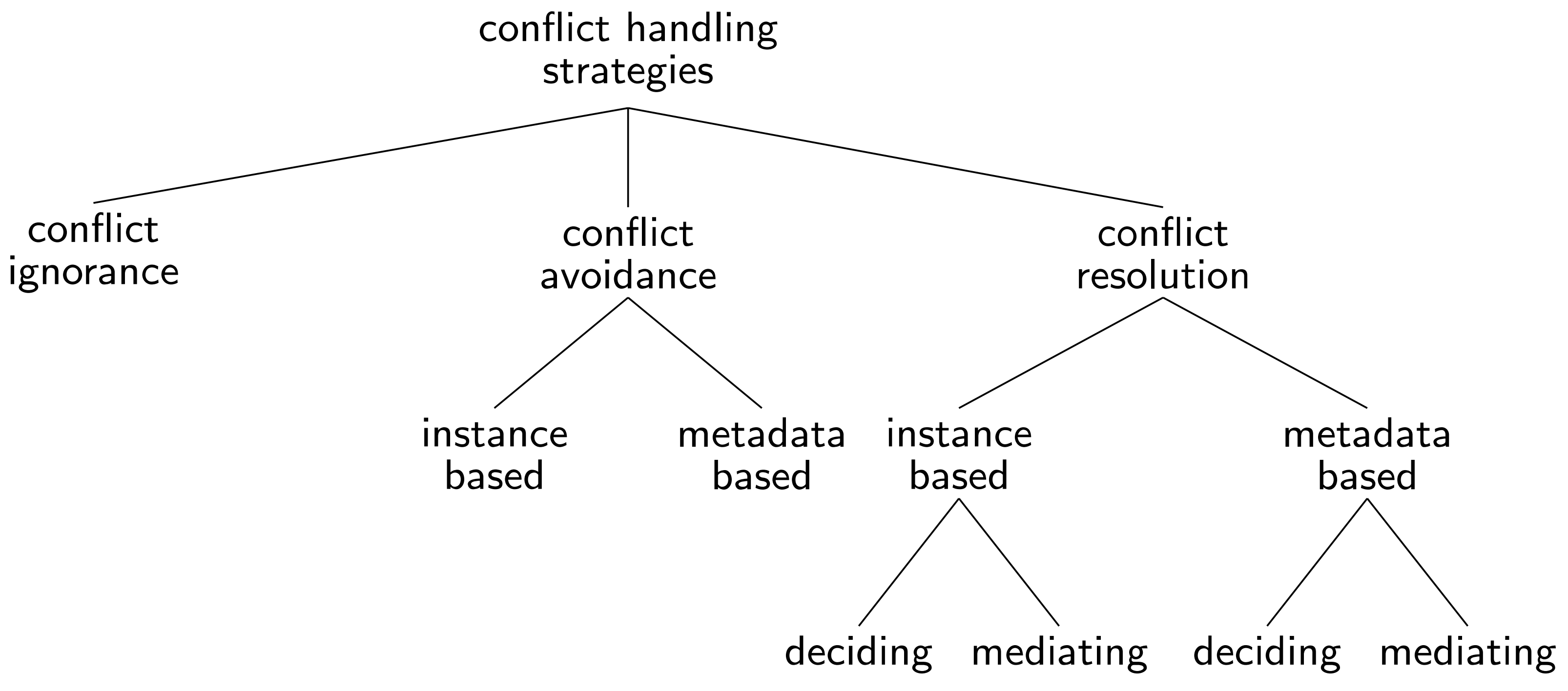
2002[12]	10.300 oz	899,500 oz
2005[13]	25.272	2.174.620 oz
2006[13]	49.354 oz	3.005.611 oz
2007[13]	48.807 oz	3.165408 oz
2008[9]	47.755 oz	3.157.837 oz
2009[2]	0.9 million oz	818.050 oz

WARRIORS@Susses Thunder	13-28	—
WARRIORS@Hampshire Thrashers	42-13	—
Essex Spartans@WARRIORS	P-P	Postponed
WARRIORS@Cambridgeshire Cats	36-44	—
East Kent Maver-icks@WARRIORS	12-18	—
WARRIORS@East Kent Mav-ericks	15-17	—

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