

ICS 321 Spring 2012

# Other Data Models : Unstructured, Graph, Key-Value Pairs

Asst. Prof. Lipyeow Lim

Information & Computer Science Department

University of Hawaii at Manoa

# Outline

Unstructured Data and Inverted Indexes

Web Search Engines

RDF & Linking Open Data

Big Table, CouchDB, & Cassandra

# Unstructured Data

- What are some examples of unstructured data?
- How do we model unstructured data ?
- How do we query unstructured data ?
- How do we process queries on unstructured data ?
- How do we index unstructured data ?

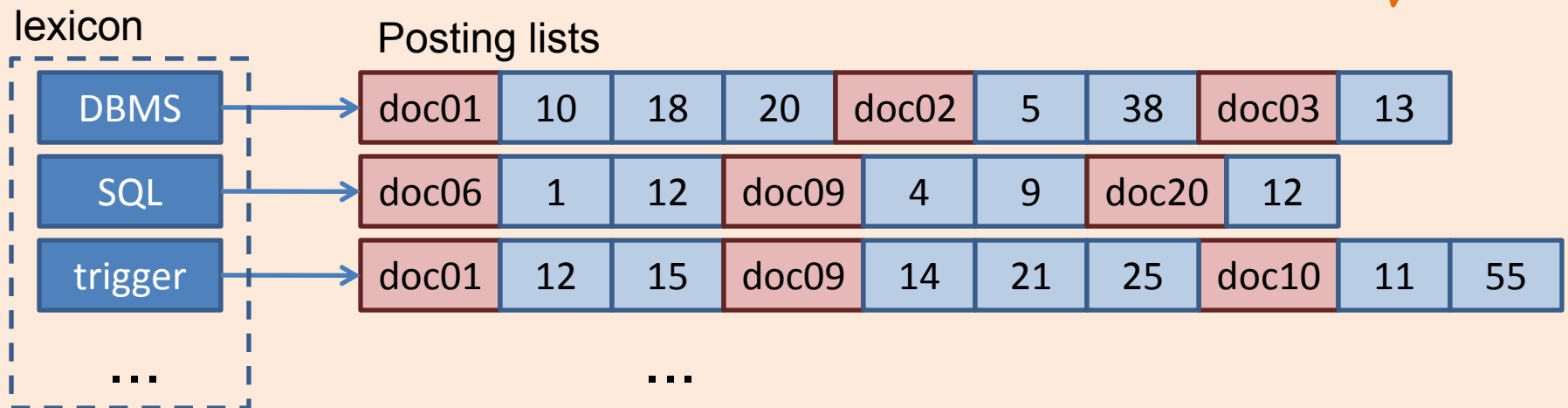
# Unstructured Text Data

- Field of “**Information Retrieval**”
- Data Model
  - Collection of documents
  - Each document is a **bag of words** (aka terms)
- Query Model
  - **Keyword** + Boolean Combinations
  - Eg. DBMS and SQL and tutorial
- Details:
  - Not all words are equal. “**Stop words**” (eg. “the”, “a”, “his” ...) are ignored.
  - **Stemming** : convert words to their basic form. Eg. “Surfing”, “surfed” becomes “surf”

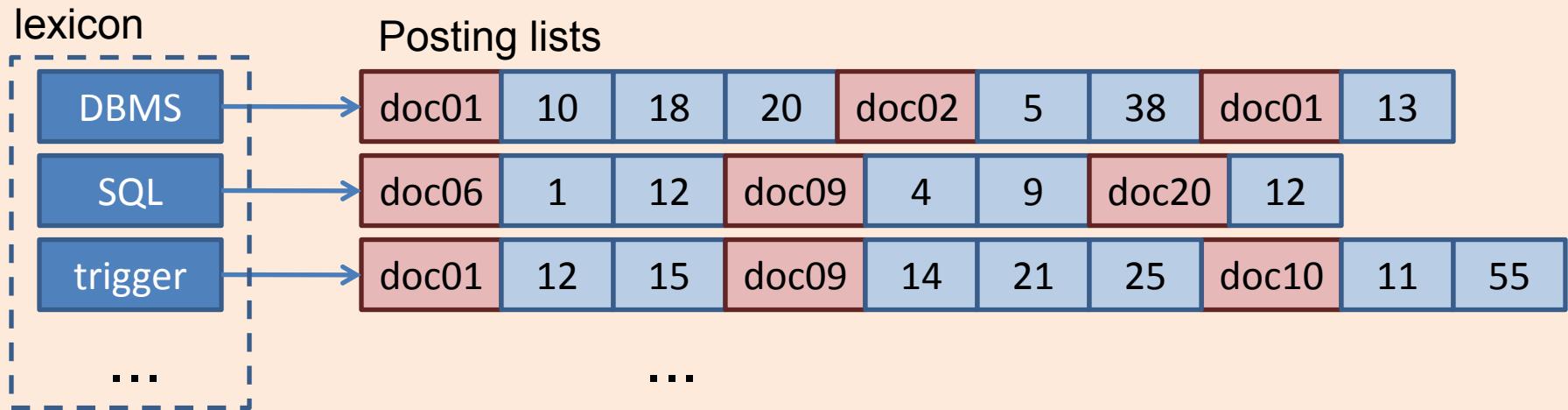
# Inverted Indexes

- Recall: an index is a mapping of search key to data entries
  - What is the search key ?
  - What is the data entry ?
- Inverted Index:
  - For each term store a list of postings
  - A posting consists of <docid,position> pairs

What is the data in an inverted index sorted on ?



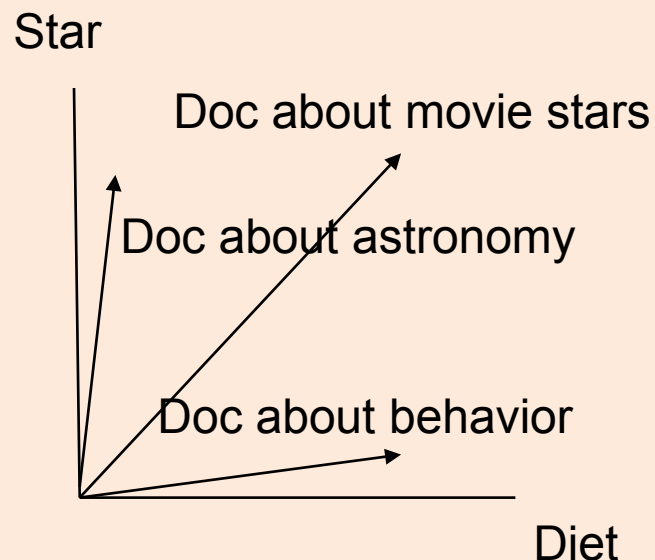
# Lookups using Inverted Indexes



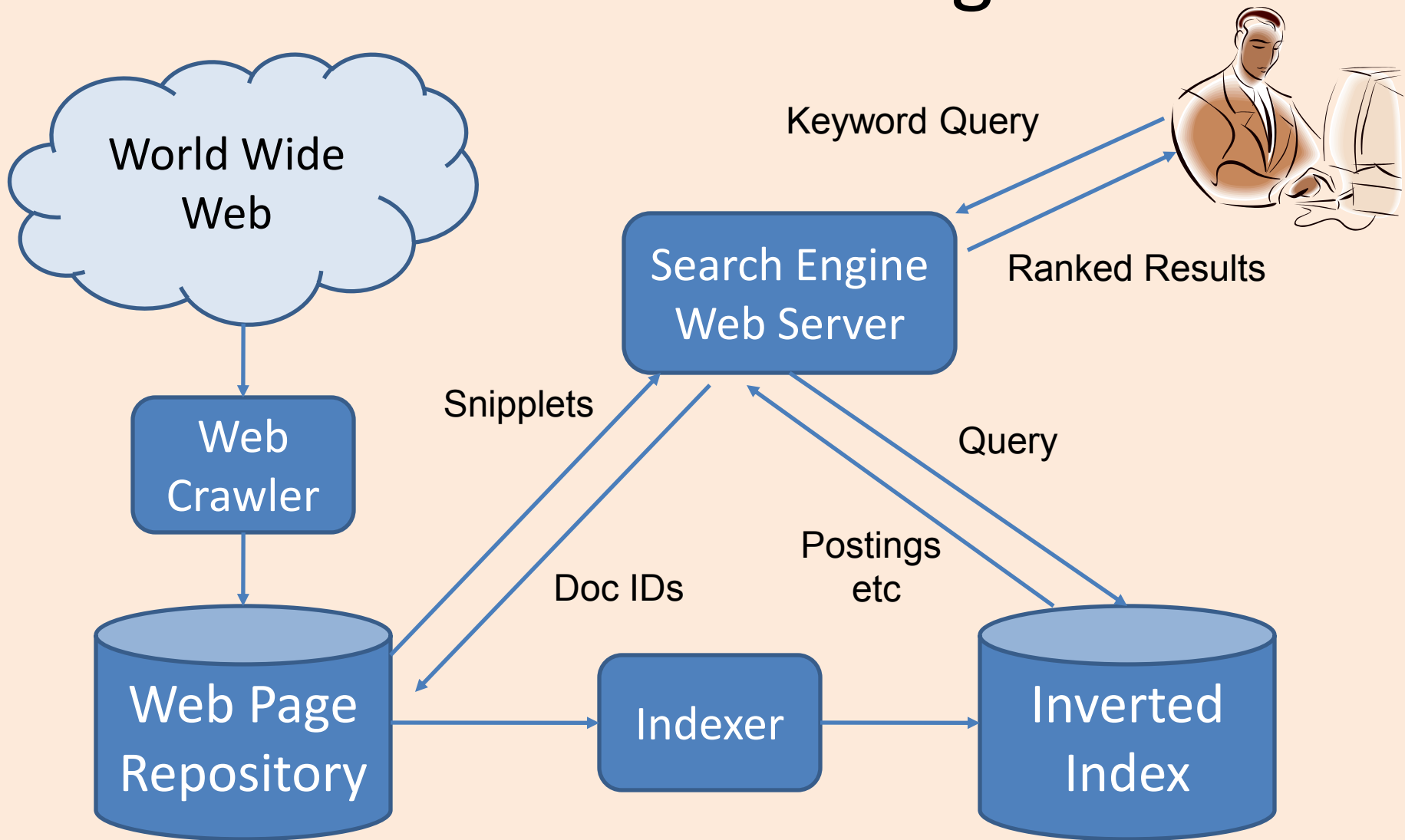
- Given a **single keyword query “k”** (eg. SQL)
  - Find k in the lexicon
  - Retrieve the posting list for k
  - Scan posting list for document IDs [and positions]
- What if the query is **“k1 and k2”** ?
  - Retrieve document IDs for k1 and k2
  - Perform intersection

# Too Many Matching Documents

- Rank the results by “relevance”!
- Vector-Space Model
  - Documents are **vectors** in hi-dimensional space
  - Each dimension in the vector represents a term
  - **Queries** are represented as **vectors** similarly
  - **Vector distance** (dot product) between query vector and document vector gives ranking criteria
  - **Weights** can be used to tweak relevance
- PageRank (later)



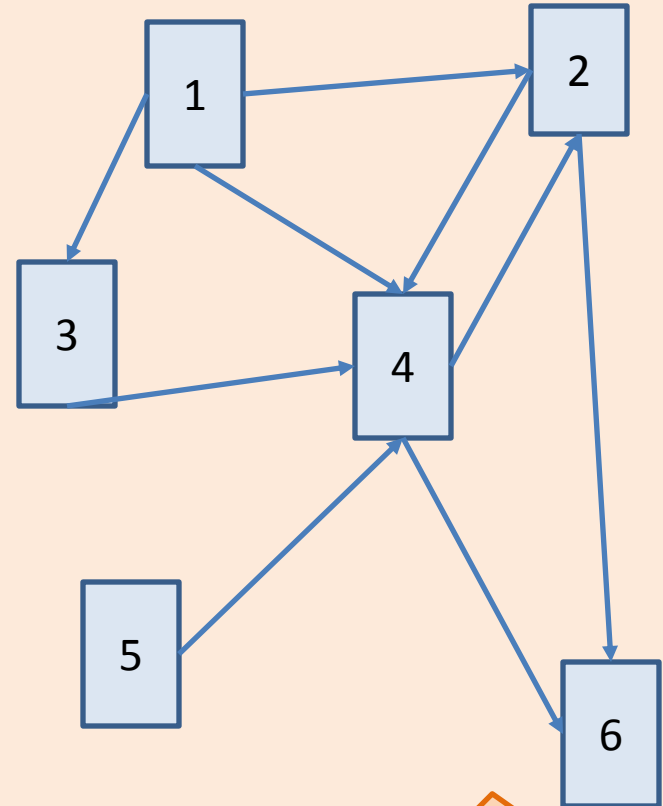
# Internet Search Engines





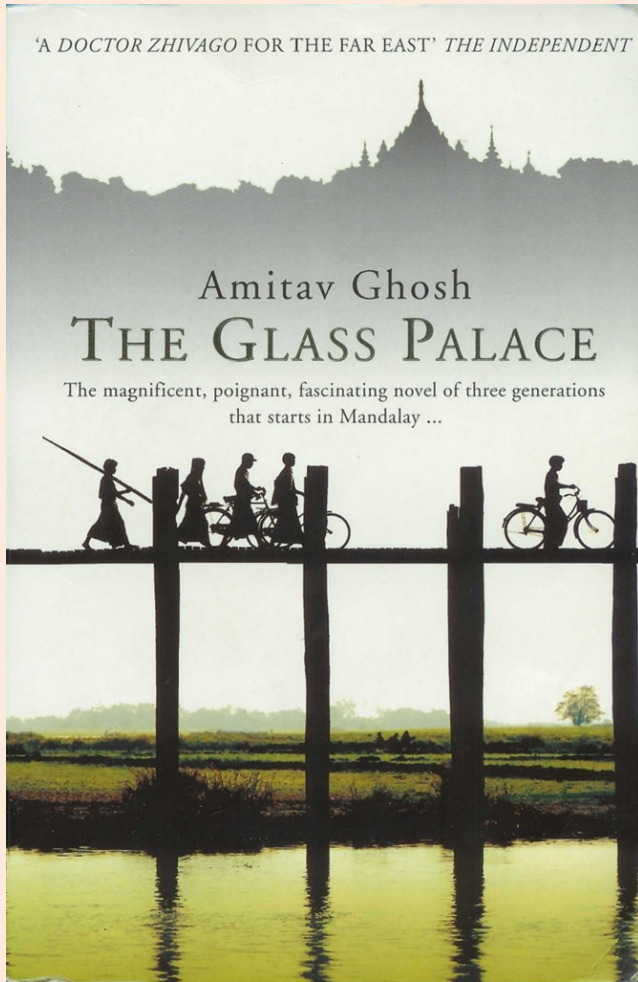
# Ranking Web Pages

- Google's **PageRank**
  - Links in web pages provide clues to **how important a webpage** is.
- Take a **random walk**
  - Start at some webpage  $p$
  - Randomly pick one of the links and go to that webpage
  - Repeat for all eternity
- The **number of times** the walker visits a page is an indication of how **important** the page is.



Vertices represent web pages.  
Edges represent web links.

# Resource Description Framework (RDF)

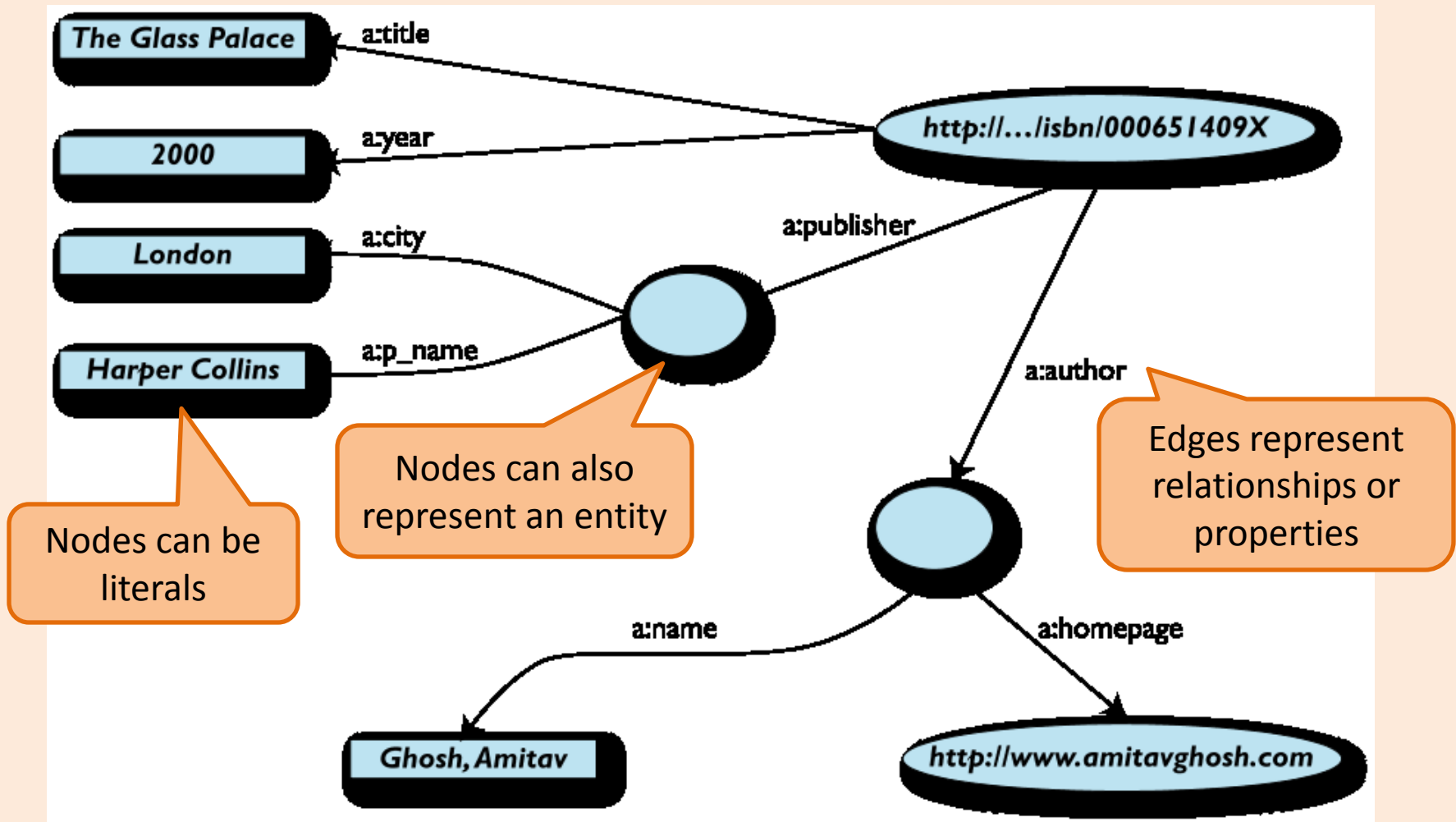


ID	Author	Title	Publisher	Year
Isbn0-00-651409-X	Id_xyz	The glass palace	Id_qpr	2000

ID	Name	Homepage
Id_xyz	Ghosh, Amitav	<a href="http://www.amitavghosh.com">http://www.amitavghosh.com</a>

ID	Publisher Name	City
Id_qpr	Ghosh, Amitav	London

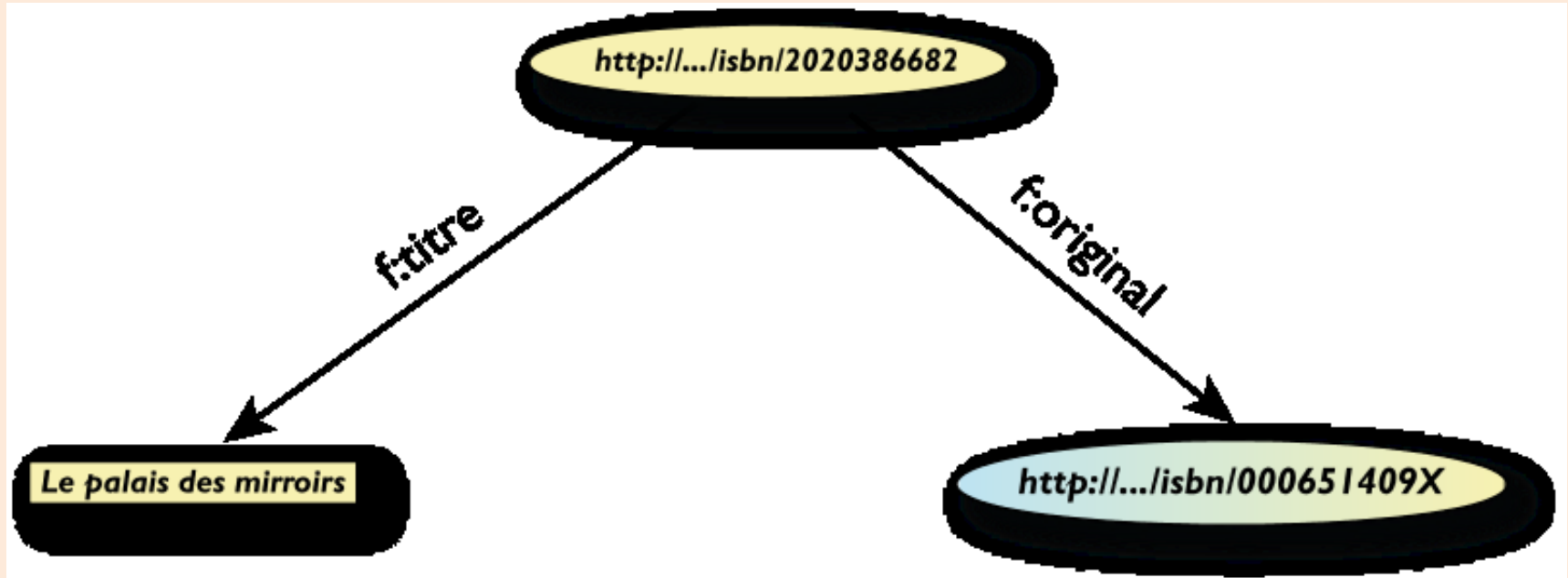
# RDF Graph Data Model



# More formally

- An **RDF graph** consists of a set of RDF triples
- An **RDF triple** (s,p,o)
  - “s”, “p” are URI-s, ie, resources on the Web;
  - “o” is a URI or a literal
  - “s”, “p”, and “o” stand for “subject”, “property” (aka “predicate”), and “object”
  - here is the complete triple: (<http://...isbn...6682>, <http://..../original>, <http://...isbn...409X>)
- RDF is a general model for such triples
- RDF can be serialized to machine readable formats:
  - RDF/XML, Turtle, N3 etc

# RDF/XML



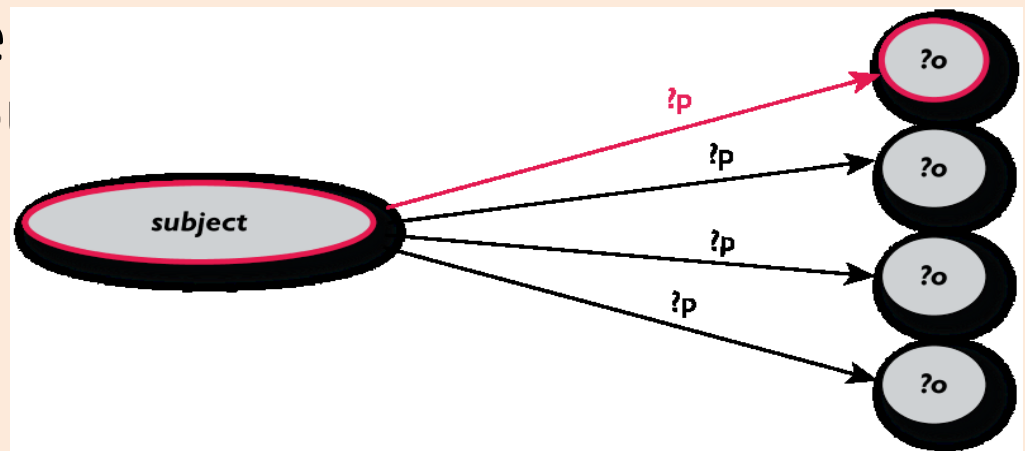
```
<rdf:Description rdf:about="http://.../isbn/2020386682">  
  <f:titre xml:lang="fr">Le palais des miroirs</f:titre>  
  <f:original rdf:resource="http://.../isbn/000651409X"/>  
</rdf:Description>
```

# Querying RDF using SPARQL

- The fundamental idea: use graph patterns
- the pattern contains unbound symbols
- by binding the symbols, subgraphs of the RDF graph are selected
- if there is such a selection the query returns bound resources

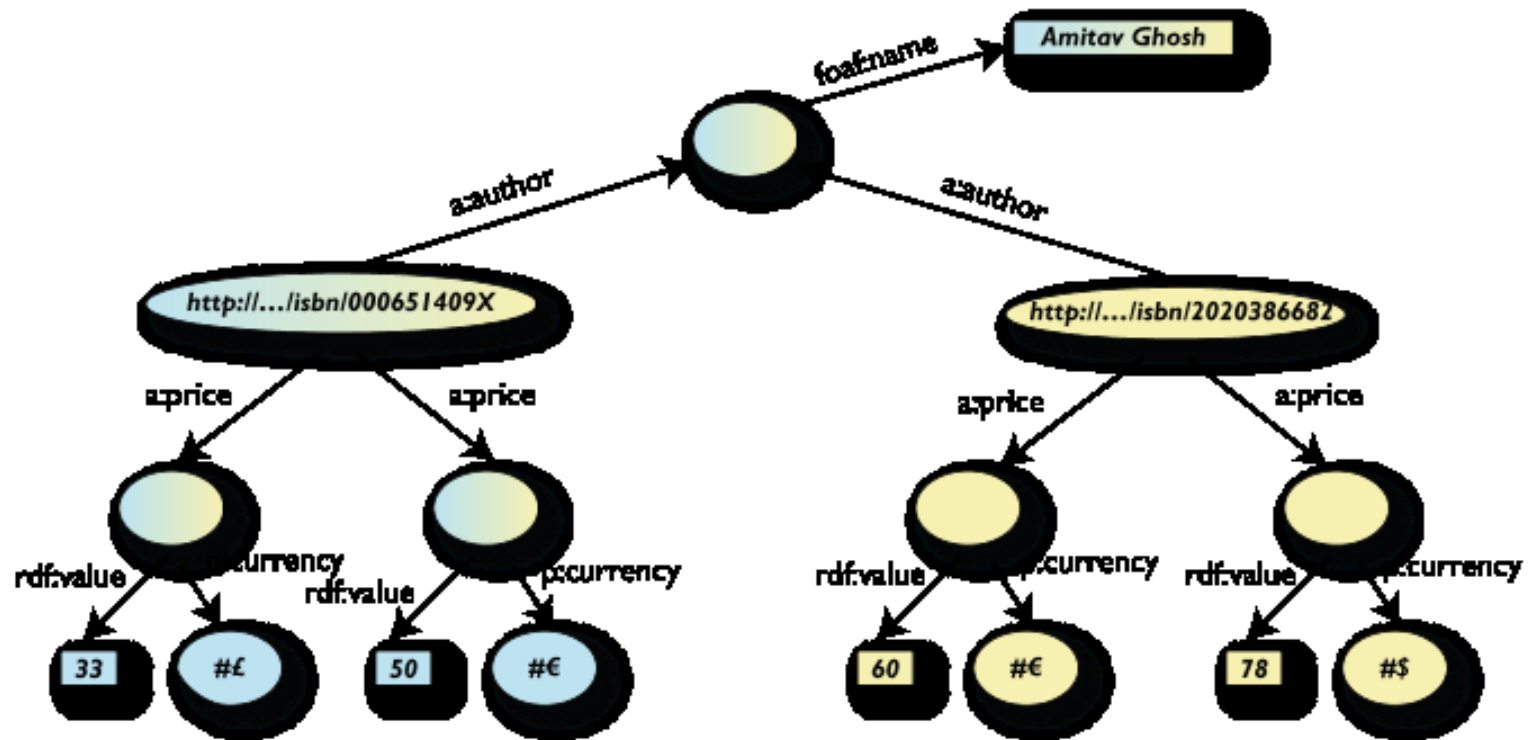
```
SELECT ?p ?o  
WHERE {subject ?p ?o}
```

Where-clause defines graph patterns. ?p and ?o denote “unbound” symbols



# Example: SPARQL

```
SELECT ?isbn ?price ?currency # note: not ?x!  
WHERE {?isbn a:price ?x.  
        ?x rdf:value ?price.  
        ?x p:currency ?currency.}
```



# Linking Open Data

- Goal: “expose” open datasets in RDF
  - Set RDF links among the data items from different datasets
  - Set up, if possible, query endpoints
- Example: DBpedia is a community effort to
  - extract structured (“infobox”) information from Wikipedia
  - provide a query endpoint to the dataset
  - interlink the DBpedia dataset with other datasets on the Web



# DBPedia

```
@prefix dbpedia
<http://dbpedia.org/resource/>.
```

```
@prefix dbterm
<http://dbpedia.org/property/>.
```

```
dbpedia:Amsterdam
```

```
dbterm:officialName "Amsterdam" ;
```

```
dbterm:longd "4" ;
```

```
dbterm:longm "53" ;
```

```
dbterm:longs "32" ;
```

```
dbterm:leaderName dbpedia:Job_Cohen ;
```

```
...
```

```
dbterm:areaTotalKm "219" ;
```

```
...
```

```
dbpedia:ABN_AMRO
```

```
dbterm:location dbpedia:Amsterdam ;
```

```
...
```

## Amsterdam




The Keizersgracht at dusk

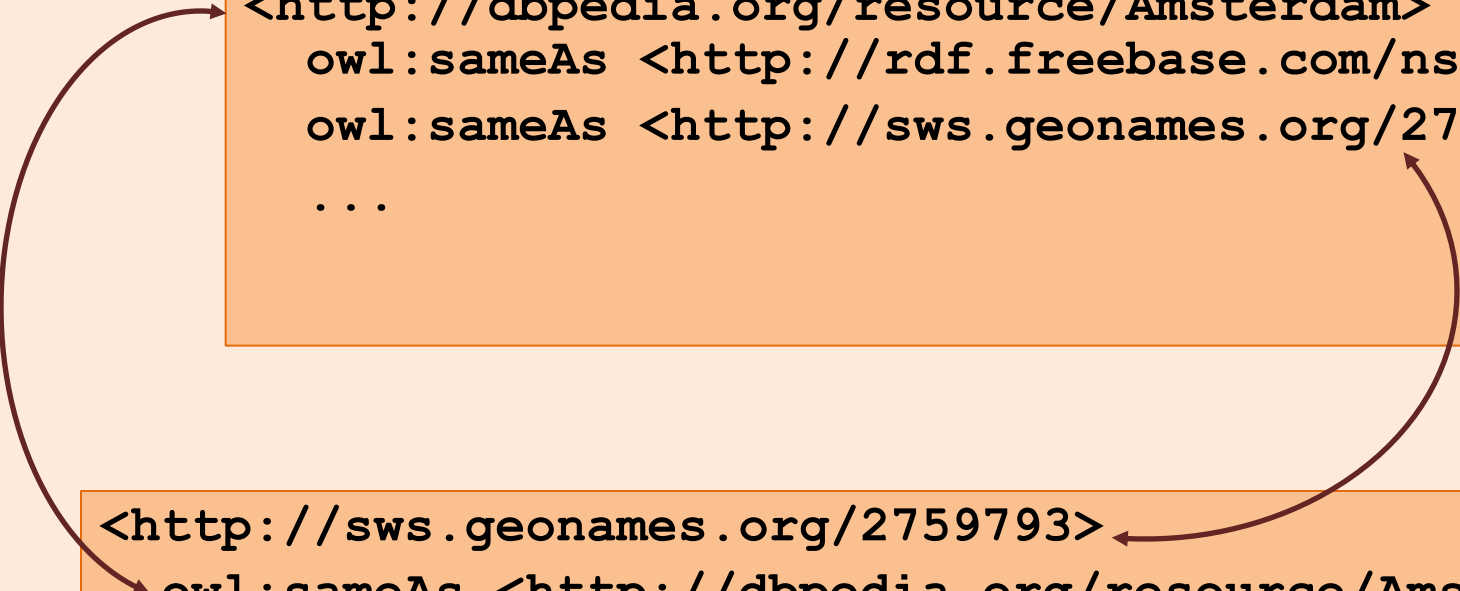
Location of Amsterdam

Coordinates:  62°22′23″N 4°53′32″E

Country	Netherlands
Province	North Holland
<b>Government</b>	
<span> </span> • <span> </span> Type	Municipality
<span> </span> • <span> </span> Mayor	Job Cohen <sup>[1]</sup> (PvdA)
<span> </span> • <span> </span> Aldermen	Lodewijk Asscher Carolien Gehrels Tjeerd Herrema Maarten van Poelgeest Marijke Vos
<span> </span> • <span> </span> Secretary	Erik Gerritsen
<b>Area</b> <sup>[2][3]</sup>	
<span> </span> • <span> </span> City	219 km <sup>2</sup> (84.6 sq mi)
<span> </span> • <span> </span> Land	166 km <sup>2</sup> (64.1 sq mi)
<span> </span> • <span> </span> Water	53 km <sup>2</sup> (20.5 sq mi)
<span> </span> • <span> </span> Urban	1,003 km <sup>2</sup> (387.3 sq mi)
<span> </span> • <span> </span> Metro	1,815 km <sup>2</sup> (700.8 sq mi)
<b>Elevation</b> <sup>[4]</sup>	2 m (7 ft)
<b>Population</b> (1 October 2008) <sup>[5][6]</sup>	
<span> </span> • <span> </span> City	755,269
<span> </span> • <span> </span> Density	4,459/km <sup>2</sup> (11,548.8/sq mi)
<span> </span> • <span> </span> Urban	1,364,422
<span> </span> • <span> </span> Metro	2,158,372
<span> </span> • <span> </span> Demonym	Amsterdammer
<b>Time zone</b>	CET (UTC+1)
<span> </span> • <span> </span> Summer (DST)	CEST (UTC+2)
<b>Postcodes</b>	1011 – 1109
<b>Area code(s)</b>	020

Website: [www.amsterdam.nl](http://www.amsterdam.nl) 

# Linking the Data



The diagram consists of two orange rectangular boxes. The top box contains an RDF snippet. The bottom box contains another RDF snippet. A curved arrow starts from the `<http://dbpedia.org/resource/Amsterdam>` URI in the top snippet and points to the `<http://dbpedia.org/resource/Amsterdam>` URI in the bottom snippet. Another curved arrow starts from the `<http://sws.geonames.org/2759793>` URI in the top snippet and points to the `<http://sws.geonames.org/2759793>` URI in the bottom snippet. This illustrates how the same URIs are used to represent the same entities in different datasets, allowing for data linking.

```
<http://dbpedia.org/resource/Amsterdam>  
  owl:sameAs <http://rdf.freebase.com/ns/...> ;  
  owl:sameAs <http://sws.geonames.org/2759793> ;  
  ...
```

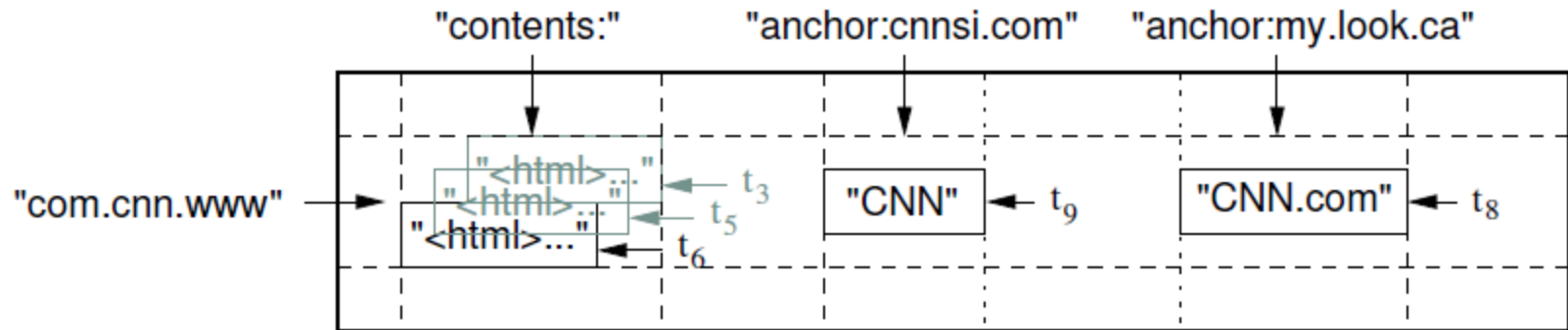
```
<http://sws.geonames.org/2759793>  
  owl:sameAs <http://dbpedia.org/resource/Amsterdam>  
  wgs84_pos:lat "52.3666667" ;  
  wgs84_pos:long "4.8833333" ;  
  geo:inCountry <http://www.geonames.org/countries/#NL>  
  ;  
  ...
```

# Google's Bigtable

*“Bigtable is a sparse, distributed, persistent multidimensional sorted map”*

- It is a type key-value store:
  - Key: (row key, column key, timestamp)
  - Value: uninterpreted array of bytes
- Read & write for data associated with a row key is atomic
- Data ordered by row key and range partition into “tablets”
- Column keys are organized into column families:
  - A column key then is specified using <family:qualifier>
- Timestamp is a 64 bit integer timestamp in microseconds

# Example: Webpages using Bigtable



- Row key = reversed string of a webpage's URL
- Column keys:
  - contents:
  - anchor:cnnsi.com
  - anchor:my.look.ca
- Timestamps: t3, t5, t6, t8, t9

# CouchDB

- A distributed document database server
  - Accessible via a RESTful JSON API.
  - Ad-hoc and schema-free
  - robust, incremental replication
  - Query-able and index-able
- A couchDB document is a set of key-value pairs
  - Each document has a unique ID
  - Keys: strings
  - Values: strings, numbers, dates, or even ordered lists and associative maps

# Example: couchDB Document

"Subject": "I like Plankton"

"Author": "Rusty"

"PostedDate": "5/23/2006"

"Tags": ["plankton", "baseball", "decisions"]

"Body": "I decided today that I don't like baseball. I like plankton."

- CouchDB enables views to be defined on the documents.
  - Views retain the same document schema
  - Views can be materialized or computed on the fly
  - Views need to be programmed in javascript

# Cassandra

- Another distributed, fault tolerant, persistent key-value store
- Hierarchical key-value pairs (like hash/maps in perl/python)
  - Basic unit of data stored in a “column”:  
(Name, Value, Timestamp)
- A **column family** is a map of columns: a set of name:column pairs. “Super” column families allow nesting of column families
- A **row key** is associated with a set of column families and is the unit of atomicity (like bigtable).
- No explicit indexing support – need to think about sort order carefully!

# Example: Cassandra

