Linked and graph data

*Introduction & lecture 1: what’s the big deal about big data?*

George Fletcher

Eindhoven University of Technology
The Netherlands

Short course in the International Doctoral School in ICT
University of Modena and Reggio Emilia, Italy

22 October 2013
introduction
Goals of this course

Participants will

1. acquire an overview of recent research developments in the management of “big” data on the web (i.e., linked (open) and graph data);

2. gain insight into challenges posed for querying in this context and current solutions in indexing and query processing for overcoming these challenges; and,

3. develop a perspective on interesting research directions in web data management.
Introduction and overview of linked and graph data, and discussion of related research in the big data field.

lecture 1. what’s the big deal about big data? (10:00-12:00)
  ▶ big data $\cup$ web data
  ▶ big data $\cap$ web data
  ▶ our focus: modeling and querying linked and graph data

lecture 2. models for web data (14:00-16:00)
  ▶ RDF and RDFS data model
  ▶ Updates on RDFS graphs
Foundations of query processing over graph and linked data.

**lecture 3.** querying graph data (11:00-13:00)
- graphs and the macro-structure of the web
- graph database models
- query languages for graphs

**lecture 4.** querying web data (14:30-16:30)
- SPARQL 1.1
- RDF & SPARQL exercises
Syllabus of the course: Thursday, 24 October

Engineering solutions for efficient query processing over graph and linked data.

lecture 5. querying web data, cont. (11:00-13:00)
  ▶ models for linked data
  ▶ querying linked data

lecture 6. indexing web data and query processing (14:30-16:30)
  ▶ value-based indexing of web data
  ▶ structural indexing of web data
Syllabus of the course: Tuesday, 29 October

Advanced topics and concluding discussion; final exam.

lecture 7. uncertain graphs & wrap up (11:00-13:00)
  ▶ uncertain graph data & query model
  ▶ annotated graphs
  ▶ recap of lectures & open discussion

final session. written exam (14:30-16:30)
Syllabus of the course

course contact

▶ **email**: g.h.l."my family name"@tue.nl
▶ **homepage**: google me (george fletcher eindhoven)
▶ **course homepage**: found on my “teaching” page
  http://wwwis.win.tue.nl/~gfletcher/modena/

course materials

▶ Online links to all background and lecture material will be provided as the course progresses.
▶ There are no specific prerequisites for the course participants, other than a general Computer Science background. All other necessary details and concepts will be introduced as needed in each lecture.
Who are we?

me

my background

▷ Assistant professor in the Web Engineering group at TU Eindhoven
▷ PhD in Computer Science from Indiana University, Bloomington, USA
▷ Bachelor’s degrees in Mathematics and Cognitive Science from the Univ. of North Florida, USA

my research

▷ in general: data management systems
▷ PhD thesis in query learning for data integration
▷ current research focus on the theory, engineering, and applications of query languages for graph and web data
me

my background

▶ Assistant professor in the Web Engineering group at TU Eindhoven
▶ PhD in Computer Science from Indiana University, Bloomington, USA
▶ Bachelor’s degrees in Mathematics and Cognitive Science from the Univ. of North Florida, USA

my research

▶ in general: data management systems
▶ PhD thesis in query learning for data integration
▶ current research focus on the theory, engineering, and applications of query languages for graph and web data

and you?
Thank you

My warmest thanks to:

- dr. Federica Mandreoli,
- The International Doctoral School in ICT,
- The University of Modena and Reggio Emilia, and
- all of the participants in this course!
What are the top two or three things you would like to take away from this course?
lecture 1

what’s the big deal about big data?
Overview of this lecture

- What does “big” data mean?
- What does “web” data mean?
- Refresher on first order logic
Our ability to generate and capture data only continues to explode

Indeed, we live in a world that is

- increasingly driven by this generation and consumption of massive ever-growing data sets (leading to an emerging “second economy”),
- which in turn is driven by both technology and a rapidly increasing “datafication” of all aspects of our public and private lives, and well, of most everything.

Managing this Big Data is now central to commerce, entertainment, government, education, ...
Funny Dilbert comic strip removed. See http://dilbert.com/strips/comic/2012-07-29/
Of course, the data management community has been intensively studying data for decades

- the prestigious Very Large Data Bases (VLDB) conference had its first edition in 1975
- “big data” < “very large data”? 
Of course, the data management community has been intensively studying data for decades

- the prestigious Very Large Data Bases (VLDB) conference had its first edition in 1975
- “big data” < “very large data”? 

(But of course, the symbiosis of technological trends and datafication does indeed raise many novel and deep challenges)
Big Data: the hype

Recently, a critical mass of awareness of the ubiquity and the enormous potential value of data has been reached, as articulated in two landmark essays:

- The end of theory (2008)
- The unreasonable effectiveness of data (2009)
Recently, a critical mass of awareness of the ubiquity and the enormous potential value of data has been reached, as articulated in two landmark essays:

- The end of theory (2008)
- The unreasonable effectiveness of data (2009)

This led to an explosion of interest and energy cutting across the scientific and business worlds, e.g.,

- journal of Science, special issue on “Dealing with Data” (2011)
  - Big Data: The Management Revolution
  - Data Scientist: The Sexiest Job of the 21st Century
- National Research Council (USA), special report on “Frontiers in massive data analysis” (2013)
Big Data: the reality

Big Data is now experiencing some backlash, tempering and nuancing this enthusiasm

- e.g., On being a data skeptic (2013)
- “Observation always involves theory.” – Edwin Hubble
Big Data: the reality

Big Data is now experiencing some backlash, tempering and nuancing this enthusiasm
▶ e.g., On being a data skeptic (2013)
  ▶ “Observation always involves theory.” – Edwin Hubble

There is, of course, still much truth in all of the enthusiasm about Big Data

Due to datafication and technology trends, applications now must face data scale, heterogeneity, and distribution as the norm, rather than as exceptions, as in the past
▶ often captured as the four V’s of “volume”, “variety”, “veracity”, and “velocity”
Big Data: the reality

Big Data is now experiencing some backlash, tempering and nuancing this enthusiasm

▶ e.g., On being a data skeptic (2013)
  ▶ “Observation always involves theory.” – Edwin Hubble

There is, of course, still much truth in all of the enthusiasm about Big Data

Due to datafication and technology trends, applications now must face data scale, heterogeneity, and distribution as the norm, rather than as exceptions, as in the past

▶ often captured as the four V’s of “volume”, “variety”, “veracity”, and “velocity”

Furthermore, there is indeed increased potential for extracting significant untapped value from data

▶ e.g., the successes of Google and Facebook are essentially built on extracting value from data
However, there is increased difficulty in extracting “value” (a fifth V), due to the other four V’s.

In data management, embracing and confronting these challenges has led to an explosion of NoSQL systems, as alternatives to the dominant paradigm of structured data (i.e., “SQL” stores).
Serious relaxations in “traditional” data management assumptions include

- uncertain data
- structural heterogeneity
  - semi-structured, denormalized data
  - query paradigms for such loosely structured data
- looser data consistency
- semantic heterogeneity

NoSQL systems typically embrace one or more of these
Big Data: NoSQL systems

Serious relaxations in “traditional” data management assumptions include

▶ uncertain data
▶ structural heterogeneity
  ▶ semi-structured, denormalized data
  ▶ query paradigms for such loosely structured data
▶ looser data consistency
▶ semantic heterogeneity

NoSQL systems typically embrace one or more of these

▶ Note, though, that many of these relaxations predate NoSQL and even “SQL” systems (e.g., CODASYL and the network data model) ...
Let’s consider data structural heterogeneity, driven by the need for so-called “polyglot persistence”.

Much modern data doesn’t cleanly map to a tight relational structure:

- missing or multi-valued attributes
  - e.g., not all site visitors have exactly one phone number
- nested/hierarchical structure
  - ontologies, folksonomies
  - JSON, XML
- loose structure
  - social networks
  - chem-/bio- networks
Four broad classes of approaches taken by “NoSQL” systems:

1. **key-value databases** (essentially scalable hash tables)
   - example systems: BerkelyDB, Tokyo Cabinet

2. **document databases** (generalization of key-value model to include nested structure, e.g., JSON and XML)
   - example systems: MongoDB, CouchDB, Couchbase

3. **column-family stores** (hybrid of key-value and relational model, where rows can have different schemas)
   - example systems: Cassandra, Amazon SimpleDB

4. **graph databases**
   - example systems: Neo4j, IBM DB2 RDF GraphStore
Big Data: NoSQL systems

Four broad classes of approaches taken by “NoSQL” systems:

1. **key-value databases** (essentially scalable hash tables)
   - example systems: BerkelyDB, Tokyo Cabinet

2. **document databases** (generalization of key-value model to include nested structure, e.g., JSON and XML)
   - example systems: MongoDB, CouchDB, Couchbase

3. **column-family stores** (hybrid of key-value and relational model, where rows can have different schemas)
   - example systems: Cassandra, Amazon SimpleDB

4. **graph databases**
   - example systems: Neo4j, IBM DB2 RDF GraphStore

Document and graph stores are firmly rooted in applications arising in web data management ...
Web Data

The explosion of the Web was both a precursor to and accelerant for the rise of Big Data.

Historically, web data has been modeled as

- hypertext and SGML
- text and HTML
- XML and JSON
- ...

Primarily focusing on the metaphor of the “document” ...
The explosion of the Web was both a precursor to and accelerant for the rise of Big Data.

Historically, web data has been modeled as
- hypertext and SGML
- text and HTML
- XML and JSON
- ...

Primarily focusing on the metaphor of the “document” ...

But early “web” scientists had even broader visions for human knowledge ...
The Mundaneum
as conceived by the Belgian author and peace activist Paul Otlet (1868-1944)
This vision has resurfaced recently in the Linked Data initiative, which is essentially a vision for modeling and sharing graph data on the web, using web standards.
Web Data: linked data

This vision has resurfaced recently in the Linked Data initiative, which is essentially a vision for modeling and sharing graph data on the web, using web standards

Linked data principles:

1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names
3. When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL)
4. Include links to other URIs so that they can discover more things

e.g., BBC, NYTimes, Wikipedia, Uniprot, PubMed, DBLP, ...
Web Data: linked data

This vision has resurfaced recently in the Linked Data initiative, which is essentially a vision for modeling and sharing graph data on the web, using web standards.

Linked data principles:

1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names
3. When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL)
4. Include links to other URIs so that they can discover more things

E.g., BBC, NYTimes, Wikipedia, Uniprot, PubMed, DBLP, ...

Linked open data: public sector data as linked data

- E.g., dati.gov.it, data.gov.uk, data.gouv.fr, data.overheid.nl, data.gov, data.eu, datameti.go.jp, ...
Our focus

In this course we study the intersection of big data and web data, namely linked data and graph data.

Our focus will be on linked and graph data management (and not information retrieval, although the distinction is increasingly fuzzy)

- i.e., scaleably implementing variations of first order logic and their application

Specifically, we will investigate the challenges of modeling and querying linked and graph data.
In this course we study the intersection of big data and web data, namely linked data and graph data.

Our focus will be on linked and graph data management (and not information retrieval, although the distinction is increasingly fuzzy)

- i.e., scaleably implementing variations of first order logic and their application

Specifically, we will investigate the challenges of modeling and querying linked and graph data.

(Equally) important issues not touched upon in this course include: data integration, security, privacy, user interfaces, data integrity and consistency, ...
a quick refresher on first order logic
First order logic: $\exists, \forall, \neg, \land, \lor, R(\bar{x}), x = y, x = a$

Most of the query languages we will consider are fragments or extensions of *first order logic* (FO)

that is, the “queries” (i.e., computable mappings from database instances to database instances) expressible in these languages are equivalently expressible as formulas in FO

we next review some basics of FO
Fix some universe $U$ of atomic values.

- A relation schema consists of a name $R$ and finite set of attribute names $\text{attributes}(R) = \{A_1, \ldots, A_k\}$. The arity of $R$ is $\text{arity}(R) = k$.

- A fact over relation schema $R$ of arity $k$ is a term of the form $R(a_1, \ldots, a_k)$, where $a_1, \ldots, a_k \in U$.
  - alternatively, a tuple over $R$ is a function from $\text{attributes}(R)$ to $U$.

- An instance of relation schema $R$ is a finite set of facts/tuples over $R$. 

Fix some universe $U$ of atomic values.

- A relation schema consists of a name $R$ and finite set of attribute names $\text{attributes}(R) = \{A_1, \ldots, A_k\}$. The arity of $R$ is $\text{arity}(R) = k$.

- A fact over relation schema $R$ of arity $k$ is a term of the form $R(a_1, \ldots, a_k)$, where $a_1, \ldots, a_k \in U$.
  - Alternatively, a tuple over $R$ is a function from $\text{attributes}(R)$ to $U$.

- An instance of relation schema $R$ is a finite set of facts/tuples over $R$.

- A database schema (aka “signature”) is a finite set $D$ of relation schemas.

- An instance of database schema $D$ is a set of relation instances, one for each $R \in D$. 
Fix some database schema $D$. 

**SQL**

```
SELECT A1, ..., Ak
FROM S1, ..., Sm
WHERE Cond
```

where $S_1, ..., S_m \in D$, $Cond$ is a well-formed selection condition over $\mathcal{A}$, and $A_1, ..., A_k \in \mathcal{A}$, for $\mathcal{A} = \bigcup_{S \in \{s_1, ..., s_m\}} \text{attributes}(S)$. 
Fix some database schema $D$.

**SQL**

```sql
SELECT A1, ..., Ak
FROM S1, ..., Sm
WHERE Cond
```

where $S_1, ..., S_m \in D$, $Cond$ is a well-formed selection condition over $\mathcal{A}$, and $A_1, ..., A_k \in \mathcal{A}$, for $\mathcal{A} = \bigcup_{S \in \{S_1, ..., S_m\}} \text{attributes}(S)$.

**Relational Algebra (RA)**

$$\pi_{A_1, ..., A_k}(\sigma_{Cond}(S_1 \times \cdots \times S_m))$$

where $S_1, ..., S_m \in D$, $Cond$ is a well-formed selection condition over $\mathcal{A}$, and $A_1, ..., A_k \in \mathcal{A}$, for $\mathcal{A} = \bigcup_{S \in \{S_1, ..., S_m\}} \text{attributes}(S)$.
Datalog

\[ \text{result}(A_1, \ldots, A_k) \leftarrow S_1(A^1), \ldots, S_m(A^m), C_1, \ldots, C_j \]

where \( S_1, \ldots, S_m \in D \), \( A^i \) is a list of variables of length \( \text{arity}(S_i) \) (for \( 1 \leq i \leq m \)), \( C_i \) is a well-formed selection condition over \( \mathcal{A} \) (for \( 1 \leq i \leq j \)), and \( A_1, \ldots, A_k \in \mathcal{A} \), for the set \( \mathcal{A} \) of all variables appearing in \( A^1, \ldots, A^m \).
List the titles of all books written by Italian speakers.
List the titles of all books written by Italian speakers.
**FO review: syntax**

```sql
SELECT book.title
FROM author, book
WHERE book.authorID = author.authorID
AND
author.language = 'Italian'
```
In RA

$$\pi_{book.title}(\sigma_C(author \times book))$$

where C is

$$book.authorID = author.authorID \land author.language = 'Italian'$$
FO review: syntax


definitions:

\[\text{author}(\text{authorID}, \text{name}, \text{birthdate}, \text{language})\]
\[\text{book}(\text{bookID}, \text{title}, \text{authorID}, \text{publisher}, \text{language}, \text{year})\]

\[\text{store}(\text{storeID}, \text{address}, \text{phone})\]

\[\text{sells}(\text{storeID}, \text{bookID})\]

In Datalog

\[\text{result}(T) \leftarrow \text{author}(A, N, D, LA), \text{book}(B, T, A, P, LB, Y),\]
\[LA = \text{‘Italian’}\]
List the names of all authors writing in Italian or born in 1985.
FO review: syntax

author(authorID, name, birthdate, language)
book(bookID, title, authorID, publisher, language, year)
store(storeID, address, phone)
sells(storeID, bookID)

SELECT A.Name
FROM Author A
WHERE A.Language = 'Italian'
UNION
SELECT A.Name
FROM Author A
WHERE A.Birthdate = 1985
FO review: syntax

\[
\text{author}(\text{authorID}, \text{name}, \text{birthdate}, \text{language})
\]
\[
\text{book}(\text{bookID}, \text{title}, \text{authorID}, \text{publisher}, \text{language}, \text{year})
\]
\[
\text{store}(\text{storeID}, \text{address}, \text{phone})
\]
\[
\text{sells}(\text{storeID}, \text{bookID})
\]

\[
\begin{align*}
\text{SELECT A.Name} \\
\text{FROM Author A} \\
\text{WHERE A.Language = 'Italian'} \\
\text{UNION} \\
\text{SELECT A.Name} \\
\text{FROM Author A} \\
\text{WHERE A.Birthdate = 1985}
\end{align*}
\]

\[
\pi_{\text{name}}(\sigma_{\text{language} = 'Italian'}(\text{author})) \cup \pi_{\text{name}}(\sigma_{\text{birthdate} = 1985}(\text{author}))
\]
FO review: syntax

\textbf{author}(\textit{authorID}, \textit{name}, \textit{birthdate}, \textit{language})
\textbf{store}(\textit{storeID}, \textit{address}, \textit{phone})
\textbf{sells}(\textit{storeID}, \textit{bookID})

\texttt{SELECT A.Name} \\
\texttt{FROM Author A} \\
\texttt{WHERE A.Language = 'Italian'} \\
\texttt{UNION} \\
\texttt{SELECT A.Name} \\
\texttt{FROM Author A} \\
\texttt{WHERE A.Birthdate = 1985}

\[\pi_{\text{name}}(\sigma_{\text{language}=\text{Italian}}(\text{author})) \cup \pi_{\text{name}}(\sigma_{\text{birthdate}=1985}(\text{author}))\]

\[\text{result}(N) \leftarrow \text{author}(A, N, B, L), L = \text{‘Italian’}\]
\[\text{result}(N) \leftarrow \text{author}(A, N, B, L), B = 1985\]
List the names of all authors writing in Italian and born in 1985.
FO review: syntax

author(authorID, name, birthdate, language)
book(bookID, title, authorID, publisher, language, year)
store(storeID, address, phone)
sells(storeID, bookID)

SELECT A.Name
FROM Author A
WHERE A.Language = ‘Italian’
INTERSECT
SELECT A.Name
FROM Author A
WHERE A.Birthdate = 1985
FO review: syntax

```sql

author(authorID, name, birthdate, language)
book(bookID, title, authorID, publisher, language, year)
store(storeID, address, phone)
sells(storeID, bookID)

SELECT A.Name
FROM Author A
WHERE A.Language = 'Italian'
INTERSECT
SELECT A.Name
FROM Author A
WHERE A.Birthdate = 1985

π_name(σ_language=‘I...’(author)) ∩ π_name(σ_birthdate=1985(author))
```
**FO review: syntax**

```sql
author(authorID, name, birthdate, language)
book(bookID, title, authorID, publisher, language, year)
store(storeID, address, phone)
sells(storeID, bookID)

SELECT A.Name
FROM Author A
WHERE A.Language = 'Italian'
INTERSECT
SELECT A.Name
FROM Author A
WHERE A.Birthdate = 1985

π_{name}(σ_{language='Italian'}(author)) \cap π_{name}(σ_{birthdate=1985}(author))

I(N) ← author(A, N, B, L), L = 'Italian'
B(N) ← author(A, N, B, L), B = 1985
result(N) ← I(N), B(N)
```
List the names of all authors writing in Italian not born in 1985.
SELECT A.Name
FROM Author A
WHERE A.Language = 'Italian'
EXCEPT
SELECT A.Name
FROM Author A
WHERE A.Birthdate = 1985
FO review: syntax

\[ \pi_{\text{name}}(\sigma_{\text{language}=\text{Italian}}(\text{author})) - \pi_{\text{name}}(\sigma_{\text{birthdate}=1985}(\text{author})) \]
FO review: syntax

\(\text{author}(\text{authorID}, \text{name}, \text{birthdate}, \text{language})\)
\(\text{book}(\text{bookID}, \text{title}, \text{authorID}, \text{publisher}, \text{language}, \text{year})\)
\(\text{store}(\text{storeID}, \text{address}, \text{phone})\)
\(\text{sells}(\text{storeID}, \text{bookID})\)

\[
\begin{align*}
\text{SELECT A.Name} \\
\text{FROM Author A} \\
\text{WHERE A.Language = 'Italian'} \\
\text{EXCEPT} \\
\text{SELECT A.Name} \\
\text{FROM Author A} \\
\text{WHERE A.Birthdate = 1985}
\end{align*}
\]

\[
\pi_{\text{name}}(\sigma_{\text{language} = 'I...'}(\text{author})) - \pi_{\text{name}}(\sigma_{\text{birthdate} = 1985}(\text{author}))
\]

\[
\begin{align*}
I(N) & \leftarrow \text{author}(A, N, B, L), L = 'Italian' \\
B(N) & \leftarrow \text{author}(A, N, B, L), B = 1985 \\
\text{result}(N) & \leftarrow I(N), \text{not } B(N)
\end{align*}
\]
List the IDs of all books which are sold in every store.
In RA

\[ \pi_{book\text{ID}}(book) - \pi_{book\text{ID}}((\pi_{store\text{ID}}(store) \times \pi_{book\text{ID}}(book)) - \text{sells}) \]
In SQL

SELECT B.bookID
FROM book B
WHERE NOT EXISTS
  (SELECT bookID, storeID
   FROM book, store
   WHERE bookID=B.bookID
   EXCEPT
   sells)
FO review: syntax

\texttt{author(authorID, name, birthdate, language)}
\texttt{book(bookID, title, authorID, publisher, language, year)}
\texttt{store(storeID, address, phone)}
\texttt{sells(storeID, bookID)}

In Datalog

\begin{align*}
  \text{all}(S, B) & \leftarrow \text{book}(B, T, A, P, L, Y), \text{store}(S, Ad, Ph) \\
  \text{missing}(B) & \leftarrow \text{all}(S, B), \text{not sells}(S, B) \\
  \text{result}(B) & \leftarrow \text{book}(B, T, A, P, L, Y), \text{not missing}(B)
\end{align*}
FO review: syntax

\begin{align*}
\text{author} & (\text{authorID}, \text{name, birthdate, language}) \\
\text{book} & (\text{bookID}, \text{title, authorID, publisher, language, year}) \\
\text{store} & (\text{storeID, address, phone}) \\
\text{sells} & (\text{storeID, bookID})
\end{align*}

in TRC, the Tuple Relational Calculus (i.e., essentially straight FO)

\[ \{ t \mid \exists b \in \text{book}(t.\text{bookID} = b.\text{bookID} \land \forall s \in \text{store} \exists \ell \in \text{sells}(\ell.\text{storeID} = s.\text{storeID} \land \ell.\text{bookID} = b.\text{bookID}) \} \]
Fact. SQL (without aggregation and other bells-and-whistles), RA, TRC, and (safe non-recursive) Datalog are all equivalent in expressive power.
Fact. SQL (without aggregation and other bells-and-whistles), RA, TRC, and (safe non-recursive) Datalog are all equivalent in expressive power.

Let
- $FO$ denote the full TRC
  - i.e., any of the above languages
**Fact.** SQL (without aggregation and other bells-and-whistles), RA, TRC, and (safe non-recursive) Datalog are all equivalent in expressive power.

Let

- **$FO$** denote the full TRC
  - i.e., any of the above languages
- **$Conj$** denote the TRC using only $\exists$ and $\land$
  - i.e., the “conjunctive” queries
  - corresponds in SQL to basic SELECT-FROM-WHERE blocks
  - corresponds to the $\{\sigma, \pi, \times\}$ fragment of RA
  - corresponds to single positive datalog rules
Fact. SQL (without aggregation and other bells-and-whistles), RA, TRC, and (safe non-recursive) Datalog are all equivalent in expressive power.

Let

- **FO** denote the full TRC
  - i.e., any of the above languages
- **Conj** denote the TRC using only $\exists$ and $\land$
  - i.e., the “conjunctive” queries
  - corresponds in SQL to basic SELECT-FROM-WHERE blocks
  - corresponds to the $\{\sigma, \pi, \times\}$ fragment of RA
  - corresponds to single positive datalog rules
- **AConj** denote the “acylic” conjunctive queries
  - queries with join trees
  - full def in a later lecture
Fact. The complexity of query evaluation is as follows

<table>
<thead>
<tr>
<th></th>
<th>FO</th>
<th>Conj</th>
<th>AConj</th>
</tr>
</thead>
<tbody>
<tr>
<td>combined</td>
<td>PSPACE-complete</td>
<td>NP-complete</td>
<td>LOGCFL-complete</td>
</tr>
<tr>
<td>data</td>
<td>Logspace</td>
<td>Logspace</td>
<td>Linear time</td>
</tr>
</tbody>
</table>

where combined means in the size of the query and the database and data means in the size of the database (i.e., for some fixed query).
In Datalog, express the following.

1. Retrieve the IDs of all books sold by stores in Modena.
2. Retrieve the IDs of all books written by an Italian speaker.
3. Retrieve the IDs of all books published in a language different from that of the book’s author.
4. Retrieve the IDs of all authors who have books published in every language (known in the database).
There is no “required” reading for this lecture. The following are just recommended background reading and credits.

  http://www.wired.com/science/discoveries/magazine/16-07/pb_theory

  http://www.mckinsey.com/insights/strategy/the_second_economy


http://www.win.tue.nl/~gfletche/Maier_MSc_thesis.pdf

http://www.nap.edu/catalog.php?record_id=18374 (register for free PDF)
