Query-driven Data Completeness Assessment

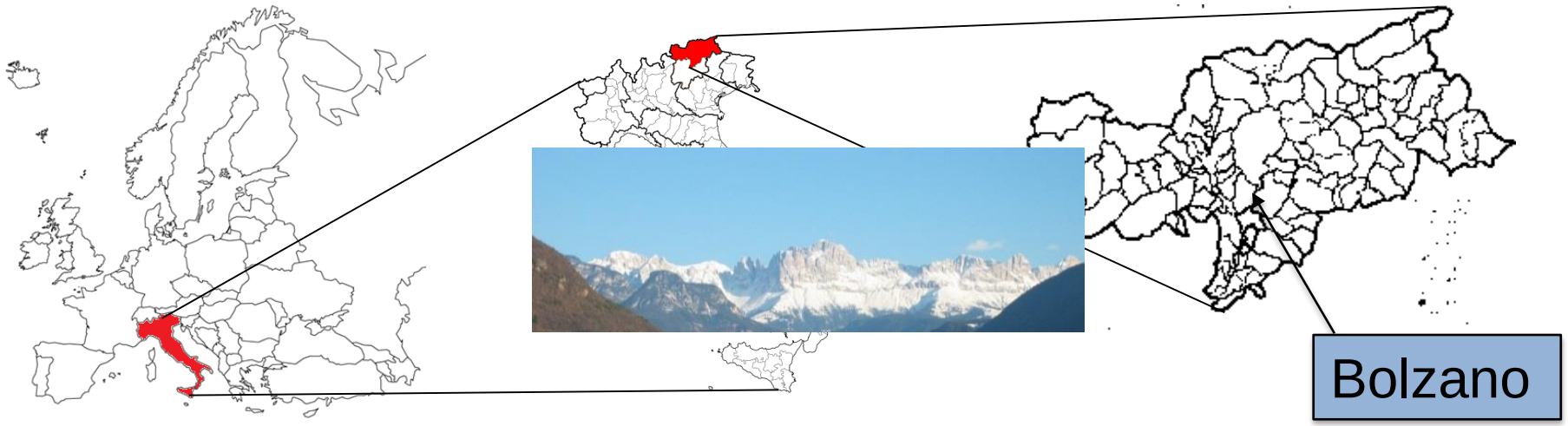
Simon Razniewski
Free University of Bozen-Bolzano, Italy

Part I joint work with
Flip Korn, Werner Nutt, Divesh Srivastava

Background

- 2011 - 2014: PhD (on reasoning about data completeness)
- 2014 - now: Assistant professor
- Research visits at UCSD (2012), AT&T Labs-Research (2013), UQ (2015), MPII (now)

unibz
Freie Universität Bozen
Libera Università di Bolzano
Università Liedia de Bulsan



Trilingual

Ötzi

1/8th of EU apples

Background (2)

- Research centered on data completeness

- Completeness often a problem in
  - Data integration
  - Complex data-generating processes
  - Large data/knowledge bases

Outline

Part I: Completeness reasoning in databases

Part II: Assessing completeness of general-purpose knowledge bases

(=Wikidata, Google Knowledge Graph, YAGO, ..)

Part I: Reasoning in databases

- Make data more complete
  - Missing value imputation
  - Information extraction
  - Data fusion
  - …
- **Reason about (in-)completeness information**
  - Missing records in relational databases (VLDB 2011)
  - Null values (CIKM 2012)
  - RDF databases (ISWC 2013)
  - Derivations from process states (BPM 2013)
  - Spatial data (SIGSPATIAL 2014)
  - **An algebra for completeness information (SIGMOD 2015)**

Motivation: Data warehouse of a telecommunication company

Admin John knows

- Team *table is complete* (HR says so)
- Maintenance *is complete for teams A, B and C*
  - their reporting systems export data automatically
- Warnings *is complete for all of Week 1, and Monday and Wednesday of Week 2*
  - Potential data loss due to a system failure on Tuesday
  - Data further than Wednesday maybe not fully loaded

Teams	
name	specialization
A	hardware
B	hardware
C	network
C	software
D	network

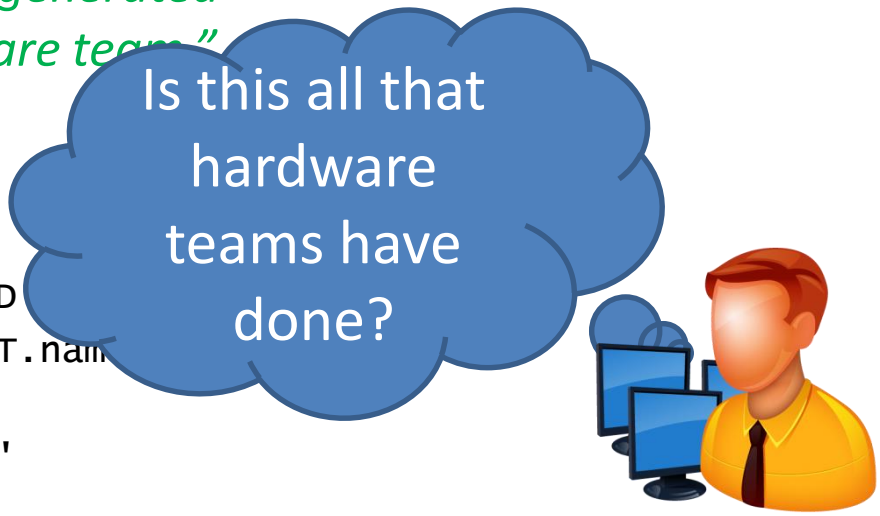
Maintenance		
ID	resp	reason
tw37	A	disk failure
tw59	D	software crash
tw83	B	unknown
tw91	C	update failure
tw91	C	network error

Warnings			
day	week	ID	message
Mon	1	tw37	high voltage
Fri	1	tw37	high voltage
Wed	2	tw37	overheat
Tue	1	tw59	auto restart
Fri	1	tw59	overheat
Mon	2	tw83	high voltage
Tue	2	tw83	auto restart

John wants to know

"Give me all warnings in Week 2 that are generated by objects in maintenance with a hardware team"

```
SELECT *
  FROM Warnings W
    JOIN Maintenance M ON W.ID = M.ID
    JOIN Teams T ON M.responsible = T.nam
  WHERE W.week = 2
    AND T.specialization = 'hardware'
```



W.Day	W.week	W.ID	W.message	M.ID	M.resp	M.reason	T.name	T.specialization
Wed	2	tw37	overheat	tw37	A	disk failure	A	hardware
Mon	2	tw83	high voltage	tw83	B	unknown	B	hardware
Tue	2	tw83	auto restart	tw83	B	unknown	B	hardware

John reasons

Teams	
name	specialization

Maintenance		
ID	resp	reason

Warnings			
day	week	ID	message

"Give me all warnings in Week 2 that are generated by objects in maintenance with a hardware team."

- Warnings *is complete for Week 1 and Monday and Wednesday of Week 2*
- Maintenance *is complete for teams A, B and C*
- Team *is complete*

→ *The query result definitely contains all warnings from*
  - *Monday for team A*
  - *Monday for team B*
  - *Monday for team C*
  - *Wednesday for team A*
  - *Wednesday for team B*
  - *Wednesday for team C*

John looks at the data

*"Give me all warnings in Week 2 that are generated by objects in maintenance with a **hardware team.**"*

→ *The query result definitely contains all warnings from*

- *Monday for team A*
- *Monday for team B*

} *= All data from Monday*

- *Monday for team C*
- *Wednesday for team A*
- *Wednesday for team B*

} *= All data from Wednesday*

- *Wednesday for team C*

Teams	
name	specialization
A	**hardware**
B	**hardware**
C	network
C	software
D	network

- HR says: The team table is **complete**!

Problems

"Warnings are complete for Week 1" *(Input)*

1. How can we formally describe complete parts of a database?

"The query result contains all warnings from Monday of Week 2 for Team A" *(Output)*

2. How can we use database completeness information to identify complete parts of query answers?

Formalism: Patterns

We have all warnings from Week 1

We have all warnings from Monday of Week 2

Warnings			
day	week	ID	message
Mon	1	tw37	high voltage
Fri	1	tw37	high voltage
Wed	2	tw37	overheat
Tue	1	tw59	auto restart
Fri	1	tw59	overheat
Mon	**2**	**tw83**	**high voltage**
Tue	2	tw83	auto restart
*	1	*	*
Mon	2	*	*

- Less expressive than previously known formalisms (views, Datalog/first-order queries, ..)
- Can be expressed in the same schema as the data
- Efficient reasoning

John's knowledge expressed by patterns

Team *table is complete*

Teams	
name	specialization
A	hardware
B	hardware
C	network
C	software
D	network
*	*

Maintenance *is complete for teams A, B and C*

Maintenance		
ID	resp	reason
tw37	A	disk failure
tw59	D	software crash
tw83	B	unknown
tw91	C	update failure
tw91	C	network error
*	A	*
*	B	*
*	C	*

Warnings *is complete for all of Week 1, and Monday and Wednesday of Week 2*

Warnings			
day	week	ID	message
Mon	1	tw37	high voltage
Fri	1	tw37	high voltage
Wed	2	tw37	overheat
Tue	1	tw59	auto restart
Fri	1	tw59	overheat
Mon	2	tw83	high voltage
Tue	2	tw83	auto restart
*	1	*	*
Mon	2	*	*
Wed	2	*	*

John's conclusions expressed by patterns

"Give me all warnings in week 2 that are generated by objects in maintenance with a hardware team."

W.Day	W.week	W.ID	W.message	M.ID	M.resp	M.reason	T.name	T.specialization
Wed	2	tw37	overheat	tw37	A	disk failure	A	hardware
Mon	2	tw83	high voltage	tw83	B	unknown	B	hardware
Tue	2	tw83	auto restart	tw83	B	unknown	B	hardware
Mon	*	*	*	*	A	*	A	*
Mon	*	*	*	*	B	*	B	*
Mon	*	*	*	*	C	*	C	*
Wed	*	*	*	*	A	*	A	*
Wed	*	*	*	*	B	*	B	*
Wed	*	*	*	*	C	*	C	*

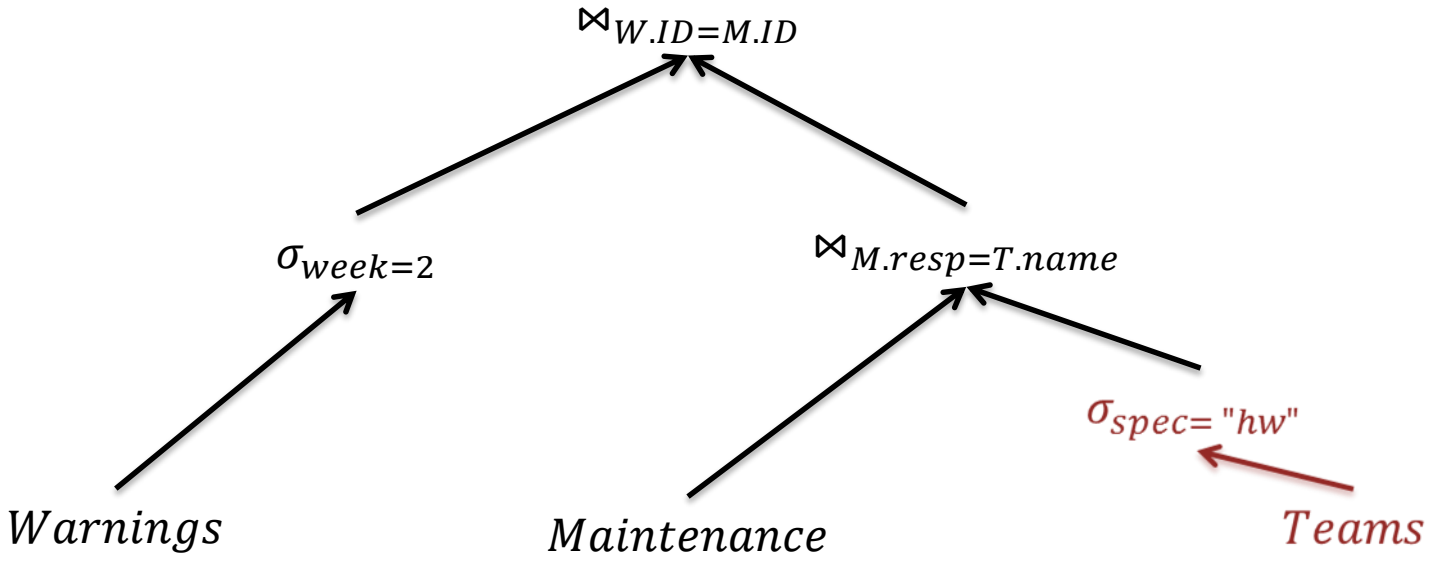
→ *The query result contains all warnings from*

- *Monday for team A*
- *…*

How to compute the completeness patterns for queries?

Queries are computed by relational algebra

Here: Select, project, equijoin



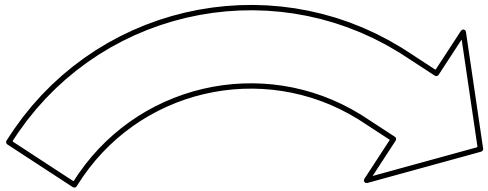
Reasoning idea:

- Apply algebra operators to completeness patterns (analogous to query result computation)

Reasoning about selections

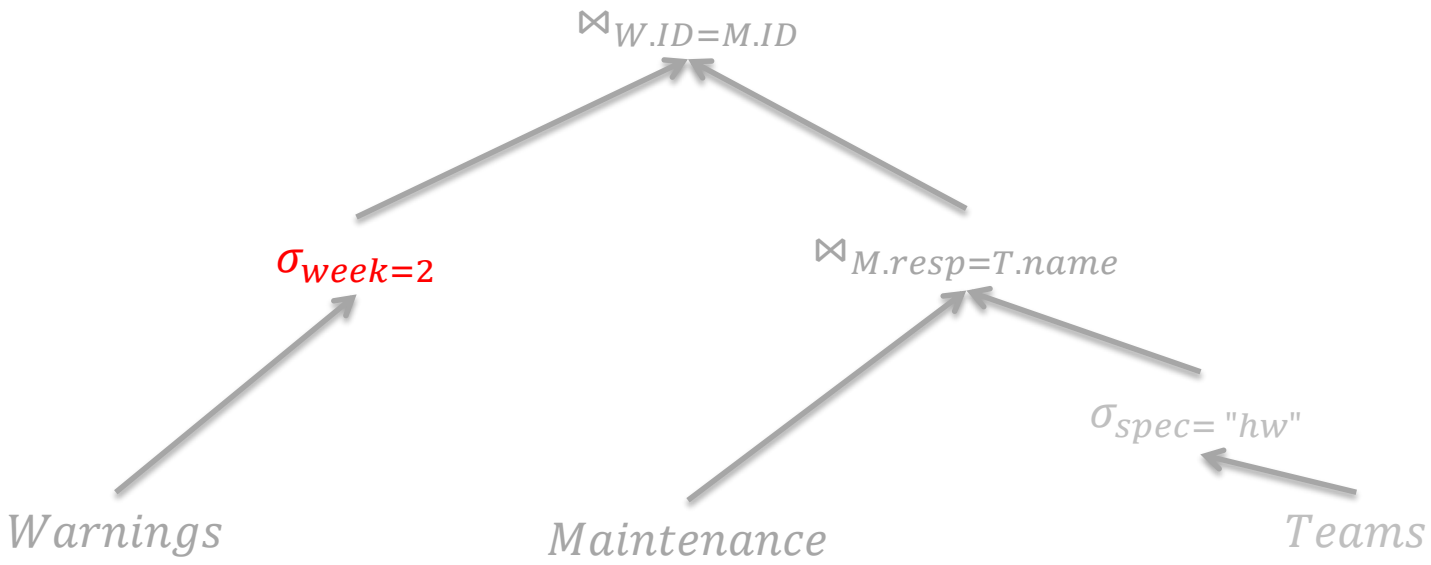
$$\sigma_{spec="hw"}(T)$$

Teams	
name	specialization
A	hardware
B	hardware
C	network
C	software
D	network
*	*



name	specialization
A	hardware
B	hardware
*	*

Rule 1: Statements with * survive


$\bowtie_{W.ID=M.ID}$
$\sigma_{week=2}$
$\bowtie_{M.resp=T.name}$
$\sigma_{spec="hw"}$
Warnings
Maintenance
Teams

Reasoning about selections (2)

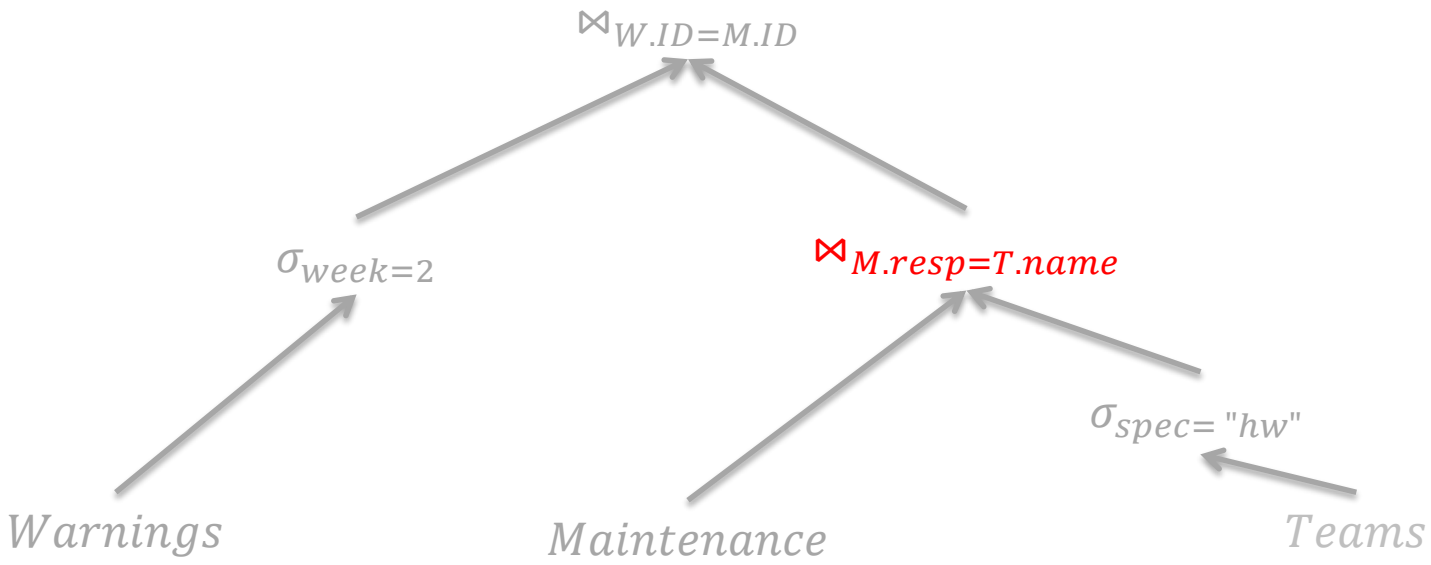
Warnings			
day	week	ID	message
Mon	1	tw37	high voltage
Fri	1	tw37	high voltage
Wed	2	tw37	overheat
Tue	1	tw59	auto restart
Fri	1	tw59	overheat
Mon	2	tw83	high voltage
Tue	2	tw83	auto restart
*	1	*	*
Mon	2	*	*
Wed	2	*	*

$\sigma_{week=2}(W)$

day	week	ID	message
Wed	2	tw37	overheat
Mon	2	tw83	high voltage
Tue	2	tw83	auto restart
Mon	*	*	*
Wed	*	*	*

Rule 2: Irrelevant constants are ignored

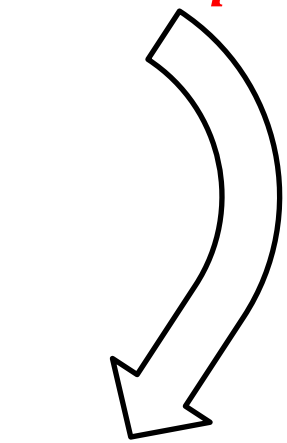
Rule 3: Selected constants survive and are promoted


$\bowtie_{W.ID=M.ID}$
$\sigma_{week=2}$
$\bowtie_{M.resp=T.name}$
$\sigma_{spec="hw"}$
Warnings
Maintenance
Teams

Reasoning about joins

Maintenance		
ID	resp	reason
tw37	A	disk failure
tw59	D	software crash
tw83	B	unknown
tw91	C	update failure
tw91	C	network error
*	A	*
*	B	*
*	C	*

$\bowtie M.resp=T.name$



$\sigma_{spec="hw"}(T)$	
name	specialization
A	hardware
B	hardware
*	*

M.ID	M.resp	M.reason	T.name	T.specialization
tw37	A	disk failure	A	hardware
tw83	B	unknown	B	hardware
*	A	*	*	*
*	B	*	*	*
*	C	*	*	*
*	*	*	A	*
*	*	*	B	*
*	*	*	C	*

Rule 1: Constants join with equal constants
Rule 2: Wildcards join with anything
Rule 3: Constants can be promoted

Algorithmic completeness

Proven: Extended algebra gives all conclusions that hold on the schema level

(reasoning only with the yellow metadata)

- Independent of the algebra tree chosen

Looking at the data

Maintenance		
ID	resp	reason
tw37	A	disk failure
tw59	D	software crash
tw83	B	unknown
tw91	C	update failure
tw91	C	network error
*	A	*
*	B	*
*	C	*

$\bowtie_{M.resp=T.name}$

$\sigma_{spec="hw"}(T)$	
name	specialization
A	hardware
B	hardware
*	*

There cannot be other hardware teams than A and B

M.ID	M.resp	M.reason	T.name	T.specialization
tw37	A	disk failure	A	hardware

M.ID	M.resp	M.reason	T.name	T.specialization
tw37	A	disk failure	A	hardware
tw83	B	unknown	B	hardware
*	*	*	*	*
*	*	*	B	*
*	*	*	C	*

Database instance allows for more promotion!

Possibly complex (coNP-complete)

So much about the theory, but…

1. How can we implement this?

2. How fast is this?
   – In comparison with query evaluation

3. How can we manage large sets of statements?

How can we implement this?

- Ideally, a plugin inside a DBMS
  - Promotion procedure benefits from fast access to data
- So far: Separate Java tool
- Schema-level algebra can also be encoded in SQL

  → Could compile normal queries into metadata queries

How fast is this? (1)

- Synthetic data

- Wikipedia has around 1000 lists declared as complete (using a template or in natural language)

This is a complete list of the 72 communities in Carmarthenshire.[Community 1]

- Abergwili
- Abernant
- Ammanford
- Betws
- Bronwydd
- Carmarthen
- Cenarth
- Cilycwm
- Llanarthney
- Llanboidy
- Llanddarog
- Llanddeusant
- Llanddowror
- Llandeilo
- Llandovery
- Llandybie
- Llangadog
- Llangain
- Llangathen
- Llangeler
- Llangennech
- Llangunnor
- Llangyndeyrn
- Llangynin

http://en.wikipedia.org/wiki/List_of_places_in_Carmarthenshire_%28categorised%29

How fast is this? (2)

- Manually extracted some and grouped them by topic
  - Recurrent topics: Sports teams, political assemblies, geographical features, songs, operas and other pieces of art
- Generated one table each about cities, schools and countries (21 statements)

city			
name	country	state	county
*	USA	Virginia	*
*	Germany	*	*
*	Ukraine	*	*
*	Bulgaria	*	*
*	USA	New York	*
*	UK	Carmarthenshire	*
*	USA	West Virginia	Hampshire County
*	Czech	Moravian-Silesia	Nový Jičín
*	Slovenia	*	*

How fast is this? (3)

```
SELECT *
FROM country, city, school
WHERE country.capital=city.name
   AND city.state=school.state
```

"All schools in capital states"

SQL runtime: 2 seconds (25891 records)

Completeness pattern runtime: 0.9 seconds (46 patterns)

Median over 7 join queries:

- SQL runtime: 2 seconds
- Completeness pattern runtime: 0.5 seconds

How to manage large sets of patterns?

Redundancies in workflows may lead to redundant patterns

John reports first that all data for Monday of Week 2 is complete, later, that the data for the whole Week 2 is complete

(Monday, 2) (*, 2)

Redundancies introduce overhead and restrict comprehensibility

→ Should be identified and possibly removed

Trivial?

(Monday, *, hardware)
(Tuesday, 2, software)
(Monday, 2, *)

(Wednesday, *, software)
(*, *, hardware)
(*, 2, software)

How to manage large sets of patterns? (2)

- Similar problems occur in term indexing for unification in classic AI
- Compared proposed data structures with classic hashing
- Time/space tradeoff:
  - Discrimination trees are most time-efficient (60 seconds for minimizing 800k patterns)
  - Hashing is marginally better in terms of space

Summary Part I

- Introduced completeness patterns to describe complete parts of databases and query answers
  - Can be expressed in the same schema as the data
- Modified the relational algebra to manipulate completeness patterns
  - Implemented join operator and evaluated scalability
- Limitation: No complete algorithm for data-dependent promotion

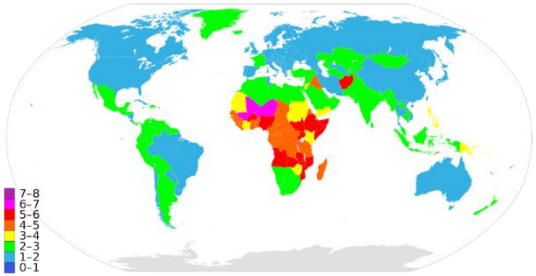
PART II: ASSESSING COMPLETENESS OF GENERAL-PURPOSE KNOWLEDGE BASES

(=Wikidata, Google Knowledge Graph, YAGO, ..)

How complete are knowledge bases?

KBs are pretty incomplete

DBpedia: contains **6 out of 35** Dijkstra Prize winners ☹

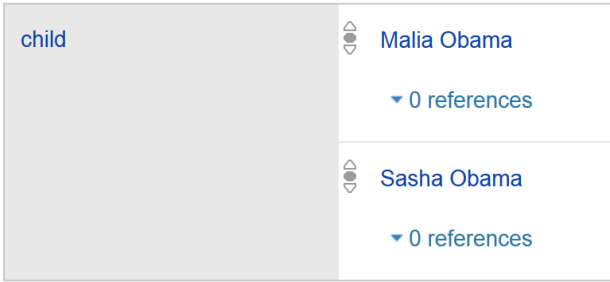


YAGO: the average number of children per person is **0.02** ☹

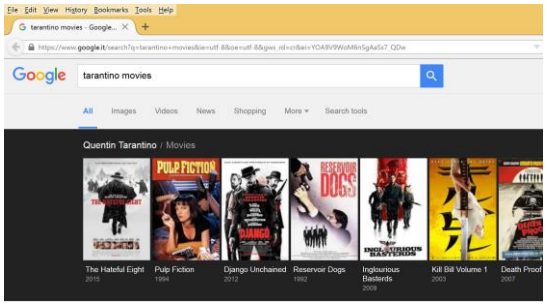
Google Knowledge Graph:
``Points of Interest'' – Completeness? ☹

KBs are pretty complete

DBpedia: **167 out of 199** Nobel laureates in Physics ☺



Wikidata: **2 out of 2** children of Obama ☺



Google Knowledge Graph: **36 out of 48** Tarantino movies ☺

So, how complete are KBs?

Name	# Entity types	# Entity instances	# Relation types	# Confident facts (relation instances)
Knowledge Vault (KV)	1100	45M	4469	271M
DeepDive [32]	4	2.7M	34	7M[a]
NELL [8]	271	5.19M	306	0.435M[b]
PROSPERA [30]	11	N/A	14	0.1M
YAGO2 [19]	350,000	9.8M	100	4M[c]
Freebase [4]	1,500	40M	35,000	637M[d]
Knowledge Graph (KG)	1,500	570M	35,000	18,000M[e]

[Dong et al., KDD 2014]

KB engineers have only tried to make KBs bigger. The point, however is to **understand** what they are actually trying to approximate.

There are **known knowns**; there are things we know we know. We also know there are **known unknowns**; that is to say we know there are some things we do not know. But there are also **unknown unknowns** – the ones we don't know we don't know.

http://www.how-complete-are-kbs.today

Facets

Occupation
- ☑ Politician 7.5%
- ❑ Soccer player 3.3%
- ❑ Lawyer 8.1%
- ❑ Other 2.2%

Nationality
- ❑ USA 3.8%
- ❑ India 2.7%
- ❑ China 2.2%
- ☑ England 5.5%
- ❑ …

Century of birth
- ❑ <15th century 1.1%
- ❑ 16th century 1.4%
- ❑ …

Gender
- ❑ Male 4.3%
- ❑ Female 3.9%

Select attribute to analyse: hasChild ▼
- date of birth
- party membership
- ….

Completeness: 2.7%

Based on:
- There are 5371 people of this kind
- For these, 231 have children
- For these, Wikipedia says there should be 750 children
- Average number of children of complete entities is 2.3
- Average number of children of unknown people is 0.01
- …..

How can we get there?

- Find patterns in data that tell about completeness
  - Extended AMIE system to mine rules about completeness [WSDM 2017]

    hockeyPlayer(x) → Incomplete(x, hasChild)
    scientist(x), hasWonNobelPrize(x) → Complete(x, graduatedFrom)

- Extract cardinality information from texts
  - Manually created patterns that allow to find information about 178% more children than Wikidata currently contains [ISWC 2016 Poster]

    *"John lives with his spouse and **five** children on a farm in Alabama."*

Are we done soon?

Challenge 1 – Modelling incompleteness

Google Knowledge Graph *knows 6 out of 7 children of President Garfield*

- RDF blank node for the 7th child?
  → No identity, thus no way to say that the 7th child is different from the previous 6

- Creating a new nameless entity?
  → Not known that the 7th child is different from all other entities in the KB

- Wikidata: New property *nr_of_children(Garfield, 7)*?
  → Semantic relation between ``*hasChild*'' and ``*nr_of_children*'' lost

Challenge 2 – Beyond triples

These are facts for humans:

Galileo Galilei:

Contrary to the dogma of the time, postulates that the earth orbits the sun [http://discovermagazine.com/2007/jul/20-things-you-didn2019t-know-about-galileo]

Reinhold Messner:

First person to climb all mountains >8000mt without supplemental oxygen [http://www.telegraph.co.uk/travel/lists/reinhold-messner-tribute-quotes-facts/]

Toothbrush? MP3-Player?

These are not KB triples

FirstPersonToClimbAllMountainsAbove8000Without(Supplemental oxygen, Reinhold Messner)

→ How incomplete are KB models?

Challenge 3 – Aggregating completeness

Can we say whether a KB knows more about Obama than about Trump?

Or about Ronaldo than about Justin Bieber?

→ Need to understand relevance, interestingness

https://www.wikidata.org/wiki/User:Ls1g/Recoin

Outlook

1. KBs contain a lot of knowledge, but little is known about how much they actually know

2. Completeness can be assessed from within KBs using pattern mining, or using external sources

3. Big challenges ahead
   - How to model incompleteness in KBs?
   - How incomplete are KB models?
   - How to aggregate completeness?