Sagi, T., Lissandrini, M., Pedersen, T.B. et al.
"**A design space for RDF data representations**"
The **VLDB Journal** 31, 347–373 (2022).

Understanding Graph Data Representations in Triplestores

Matteo Lissandrini,
Tomer Sagi, Torben Bach Pedersen, Katja Hose

AALBORG UNIVERSITY
DENMARK

GOALS

How to** (efficiently?) **store a Knowledge Graph

How to model the space of alternative data representations for KGs

How to evaluate alternative storage options

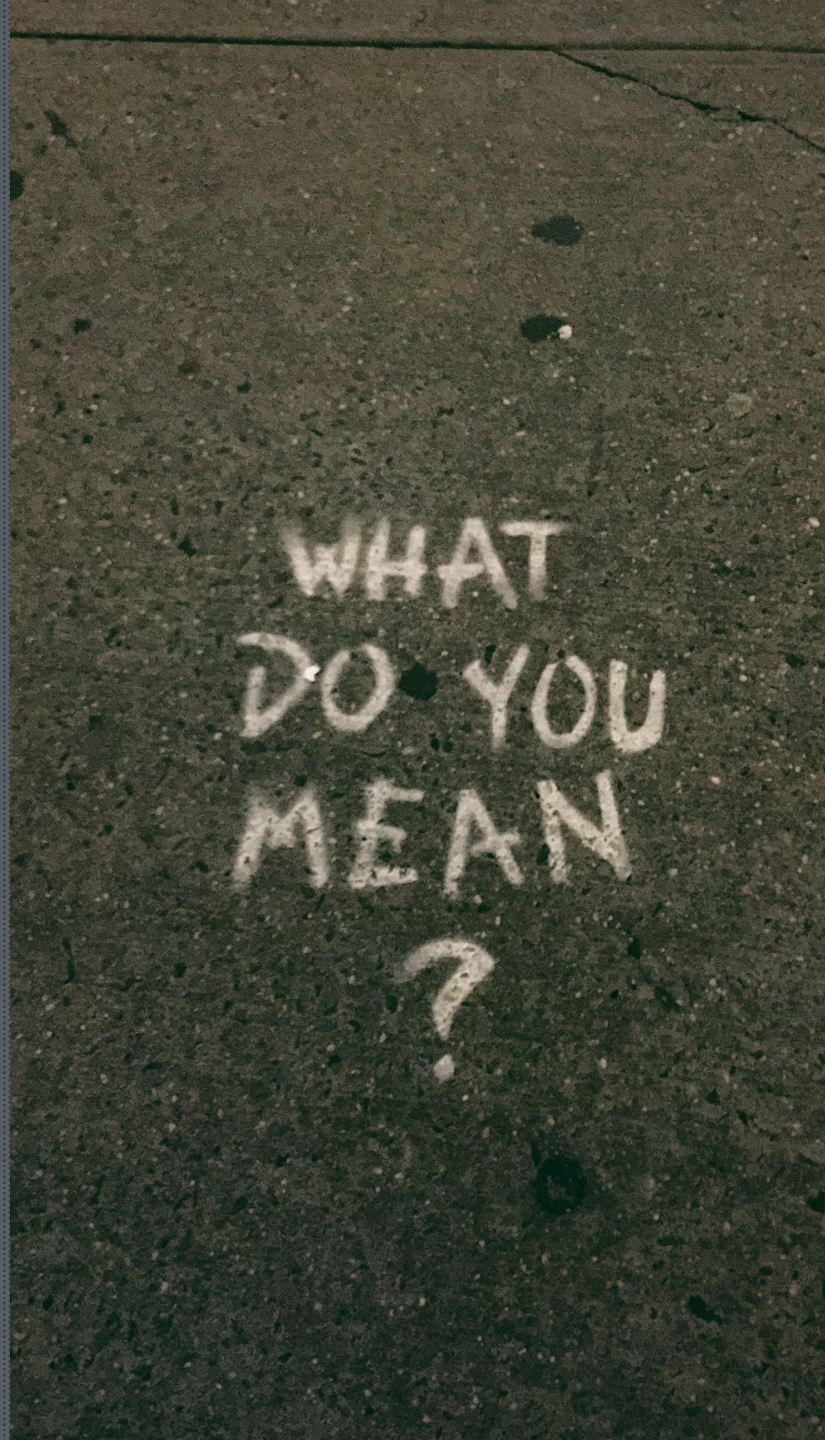
DISCLAIMER

Things to keep in mind

Intuition ≫ Technical Details

Summary of a 27pp survey with 23 systems (both prototypes & commercial)

Generate discussion of applicability beyond Triplestores



Knowledge Graphs

Heterogeneous Data
&Heterogeneous Connections

Multiple node / edge types

WikiData/Dbpedia has 9K/1.3K edge types

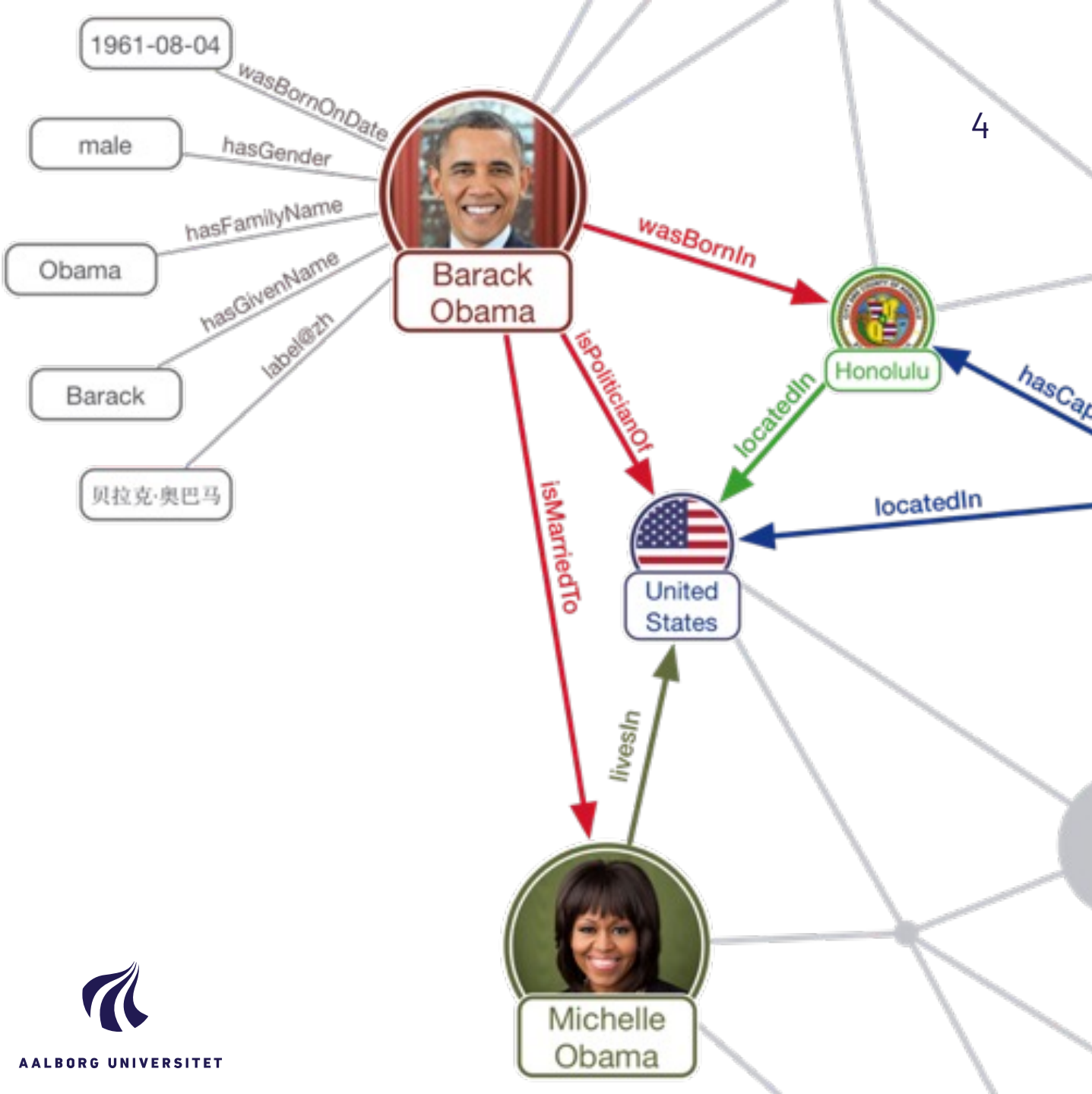
Literals with Data Types & Annotations

@en , ^^xsd:integer

Taxonomic data is part of the graph

WikiData/DBpedia has 82K/0.4K classes

Incomplete/Inconsistent Schema



AALBORG UNIVERSITET

Triplestores

Representing a KG as a Collection of Facts

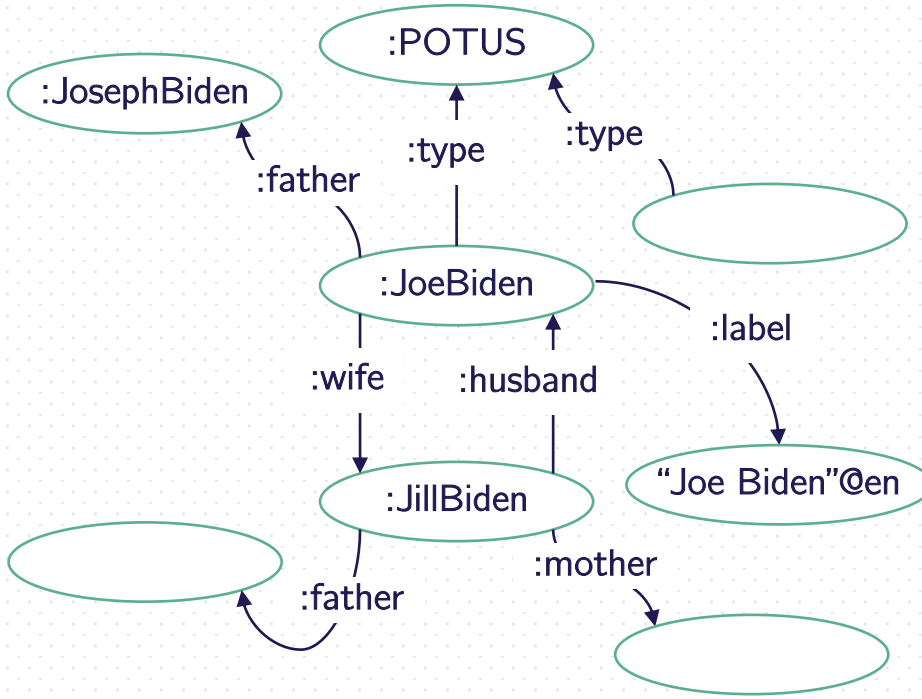
the RDF Model

- **Nodes** are either:
  - **Entities** (resources identified by IRI)
  - **Literals** (values as strings, integers, dates)
  - **Blank** Nodes (special kind of nodes without IRI)

- **Edges are statements**
  (Subject, Predicate, Object)

That's a triple!

- Edge types are resources




```
@prefix : <http://www.example.kg/> .
:JoeBiden :label "Joe Biden"@en .
:JoeBiden :type  :POTUS .
:JoeBiden :wife :JillBiden .
:JillBiden :husband :JoeBiden .
```

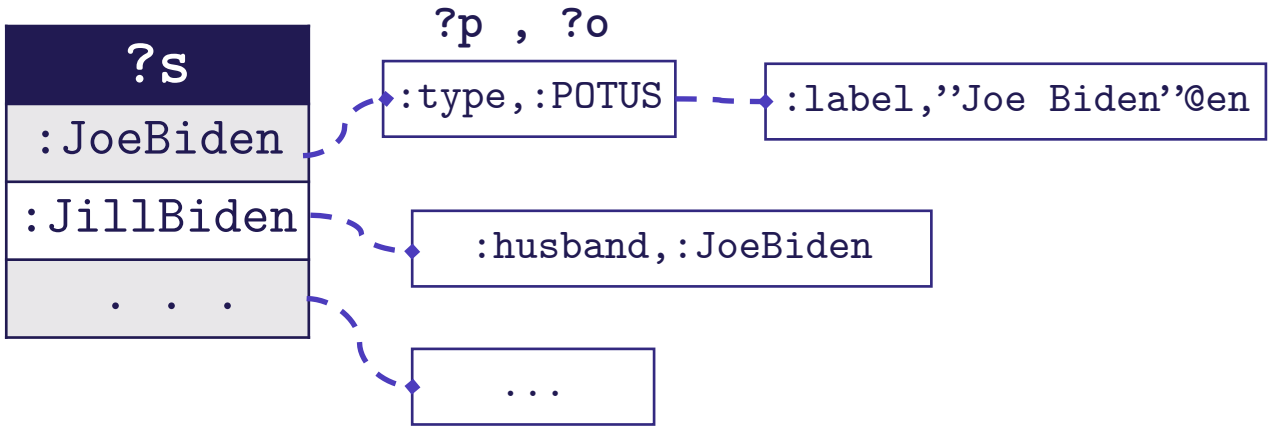
How to store a Triple?

2 random options

List of triples

?s	?p	?o
:JoeBiden	:label	“JoeBiden”@en
:JoeBiden	:type	:POTUS
:JillBiden	:husband	:JoeBiden
:JoeBiden	:wife	:JillBiden
. . .	. . .	. . .

Hash table



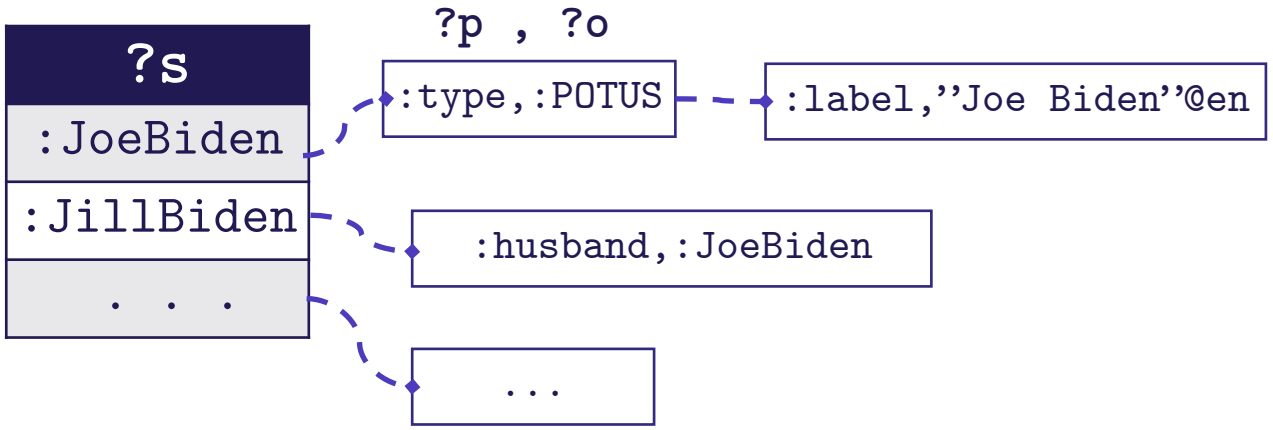
How to store a Triple?

a 3rd option

***SORTED* List of triples**

?s	?p	?o
:JillBiden	:husband	:JoeBiden
:JoeBiden	:label	‘‘JoeBiden’’@en
:JoeBiden	:type	:POTUS
:JoeBiden	:wife	:JillBiden
. . .	. . .	. . .

Hash table



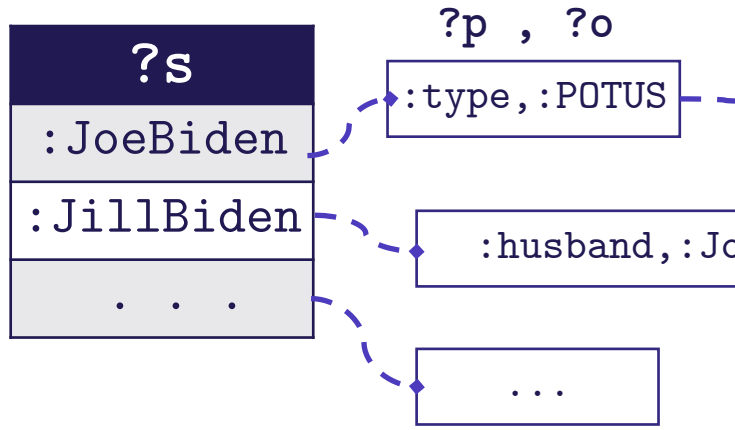
How to store a Triple?

a 4th & 5th option

Property Table

?s	:husband	:label	:type	:wife
:JillBiden	:JoeBiden	-	-	-
:JoeBiden	-	"JoeBiden"@en	:POTUS	:JillBiden
. . .	. . .	. . .	. . .	. . .

Hash table



Property Tables

:husband

?s	?o
:JillBiden	:JoeBiden
. . .	. . .

:label

?s	?o
:JoeBiden	"JoeBiden"@en
. . .	. . .

:type

?s	?o
:JoeBiden	:POTUS
. . .	. . .

:wife

?s	?o
:JillBiden	:JoeBiden
. . .	. . .

What's Best ?

It Depends!

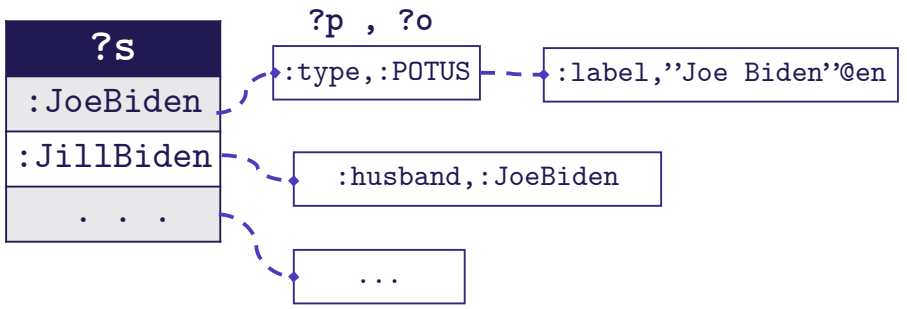
List of triples

?s	?p	?o
:JoeBiden	:label	‘‘JoeBiden’’@en
:JoeBiden	:type	:POTUS
:JillBiden	:husband	:JoeBiden
:JoeBiden	:wife	:JillBiden
. . .	. . .	. . .

Property Table

?s	:husband	:label	:type	:wife
:JillBiden	:JoeBiden	-	-	-
:JoeBiden	-	‘‘JoeBiden’’@en	:POTUS	:JillBiden
. . .	. . .	. . .	. . .	. . .

Hash table



***SORTED* List of triples**

?s	?p	?o
:JillBiden	:husband	:JoeBiden
:JoeBiden	:label	‘‘JoeBiden’’@en
:JoeBiden	:type	:POTUS
:JoeBiden	:wife	:JillBiden
. . .	. . .	. . .

Property Tables

:husband

?s	?o
:JillBiden	:JoeBiden
. . .	. . .

:label

?s	?o
:JoeBiden	‘‘JoeBiden’’@en
. . .	. . .

:type

?s	?o
:JoeBiden	:POTUS
. . .	. . .

:wife

?s	?o
:JoeBiden	:JillBiden
. . .	. . .

Querying a Triplestore

SPARQL & Graph patterns

- **Triple Pattern**

  A statement with variables

  (?f :label ?n) ⟶ ?f & ?n **are variables**

- **Basic Graph Pattern**

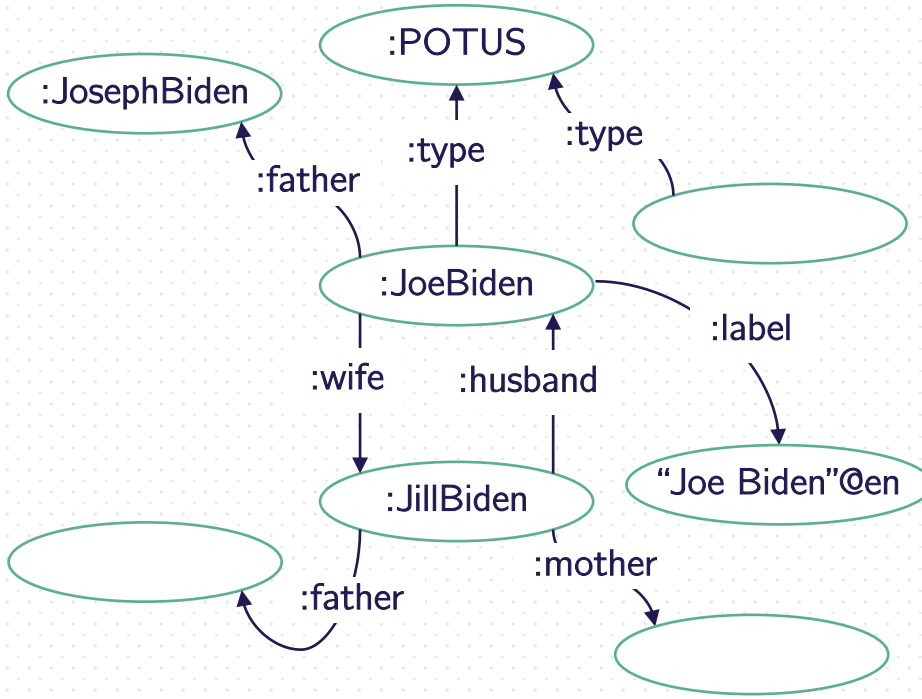
  A set of triple patterns that need to be satisfied

  (?f :label ?n) & (?f :type ?t)

- **Property Paths**

  RegEx like expression in place of triple patterns

  (?f :father+/:type/:subclass* :President)




```
SELECT ?f ?n
WHERE {
      ?f :father / :type :POTUS .
      ?f :label ?n .
}
```

Querying a Triplestore

The Space of Access Patterns

Presence of **CONSTANTS:**

(:JoeBiden :label ?n) **VS.** (?s :label ?n)

Nature of **TRAVERSAL:**

(?s :type / :subclass ?t) **VS.** ?s :type+ / :subclass* ?t

Nature of **FILTERS:**

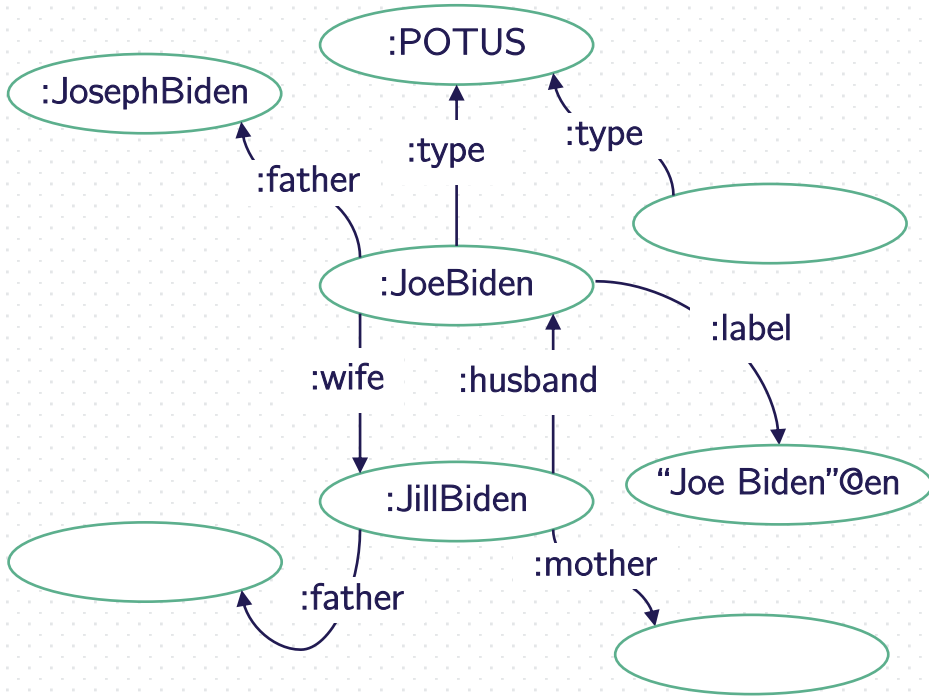
FILTER(?n >= “J”) **VS.** FILTER(isLiteral(?n))

Join on **PIVOT:**

Binary on Same Position **VS.** Arbitrary N-way

RETURN values**:**

All Values **VS.** Existence **VS.** Aggregates




```
SELECT ?f ?n
WHERE {
      ?f :father / :type :POTUS .
      ?f :label ?n .
}
```

What's Best ?

```
SELECT ?o
WHERE {
    :JoeBiden :type ?o .

}
```

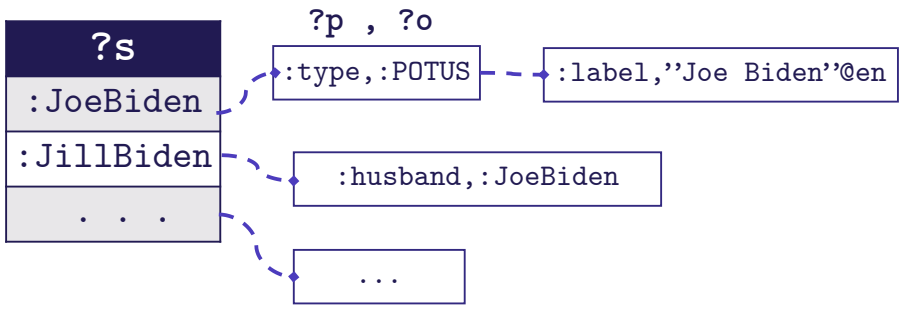
List of triples

?s	?p	?o
:JoeBiden	:label	''JoeBiden''@en
:JoeBiden	:type	:POTUS
:JillBiden	:husband	:JoeBiden
:JoeBiden	:wife	:JillBiden
. . .	. . .	. . .

Property Table

?s	:husband	:label	:type	:wife
:JillBiden	:JoeBiden	-	-	-
:JoeBiden	-	''JoeBiden''@en	:POTUS	:JillBiden
. . .	. . .	. . .	. . .	. . .

Hash table



***SORTED* List of triples**

?s	?p	?o
:JillBiden	:husband	:JoeBiden
:JoeBiden	:label	''JoeBiden''@en
:JoeBiden	:type	:POTUS
:JoeBiden	:wife	:JillBiden
. . .	. . .	. . .

Property Tables

:husband

?s	?o
:JillBiden	:JoeBiden
. . .	. . .

:label

?s	?o
:JoeBiden	''JoeBiden''@en
. . .	. . .

:type

?s	?o
:JoeBiden	:POTUS
. . .	. . .

:wife

?s	?o
:JoeBiden	:JillBiden
. . .	. . .

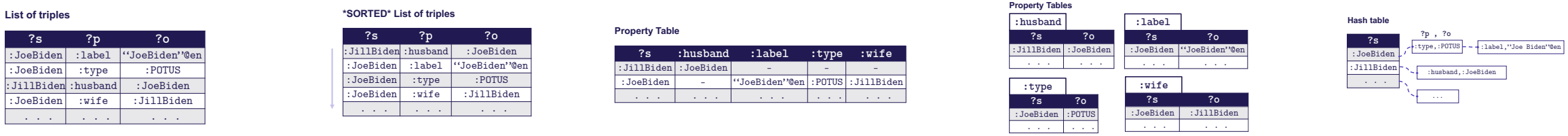
What's Best ?

- |**E**| total number of edges
- |**N**| total number of nodes
- |**L**| total number of edge types (labels)
- $|\mathbf{E}_{\ell}|$ **total number of edges for a label** ℓ
- $|\mathbf{E}_{\mathbf{n}}|$ total number of edges for node n

```
SELECT ?o
WHERE {
    :JoeBiden :type ?o .

}
```

O(|E|) O(log(|E|)) O(log(|N|) +|L|) $O(|E_{\ell}|)$ $O(|E_{n}|)$



Design Choice 1

Subdivision

reduce search space by fragmenting data

Compatibility

Representation can be "**compatible**" to an **access pattern in 3 ways:**

- **seek compatible:** the first result can be retrieved in a single random access
- **sequence compatible:** the remaining results can be computed without random access
- **selection compatible:** if no excess results are retrieved by the computation

```
SELECT ?o
WHERE {
    :JoeBiden :type ?o .

}
```

O(|E|)

List of triples

?s	?p	?o
:JoeBiden	:label	"JoeBiden"@en
:JoeBiden	:type	:POTUS
:JillBiden	:husband	:JoeBiden
:JoeBiden	:wife	:JillBiden
. . .	. . .	. . .

O(log(|E|))

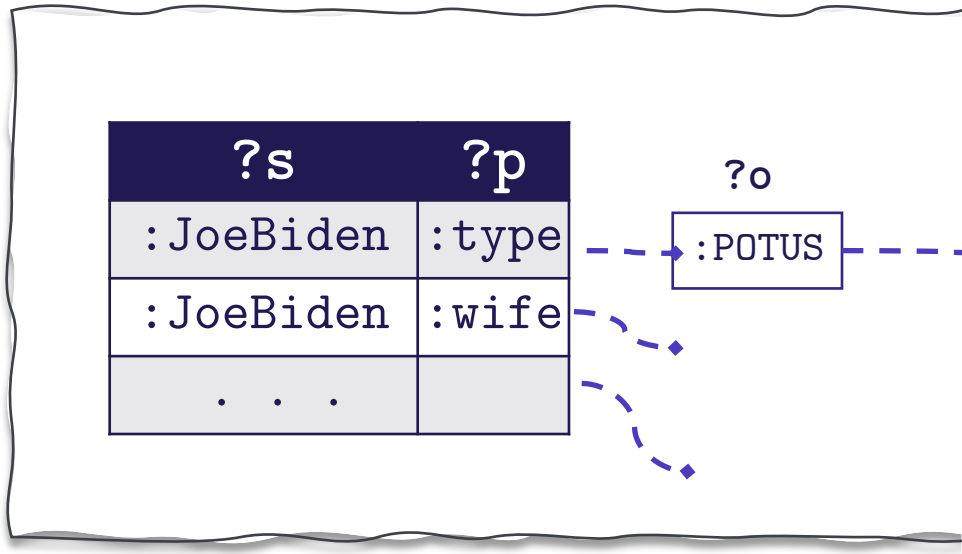
SORTED List of triples

?s	?p	?o
:JillBiden	:husband	:JoeBiden
:JoeBiden	:label	"JoeBiden"@en
:JoeBiden	:type	:POTUS
:JoeBiden	:wife	:JillBiden
. . .	. . .	. . .

O(log(|N|) +|L|)

Property Table

?s	:husband	:label	:type	:wife
:JillBiden	:JoeBiden	-	-	-
:JoeBiden	-	"JoeBiden"@en	:POTUS	:JillBiden
. . .	. . .	. . .	. . .	. . .



Design Choice 1

Subdivision

reduce search space by fragmenting data

Compatibility

Representation can be "**compatible**" to an **access pattern in 3 ways:**

- **seek compatible:** the first result can be retrieved in a single random access
- **sequence compatible:** the remaining results can be computed without random access
- **selection compatible:** if no excess results are retrieved by the computation

```
SELECT ?o
WHERE {
      :JoeBiden :type ?o .

}
```

$O(|E|)$ $O(\log(|E|))$ $O(\log(|N|) + |L|)$ $O(|E_\ell|)$ $O(|E_n|)$

List of triples

?s	?p	?o
:JoeBiden	:label	"JoeBiden"@en
:JoeBiden	:type	:POTUS
:JillBiden	:husband	:JoeBiden
:JoeBiden	:wife	:JillBiden
. . .	. . .	. . .

SORTED List of triples

?s	?p	?o
:JillBiden	:husband	:JoeBiden
:JoeBiden	:label	"JoeBiden"@en
:JoeBiden	:type	:POTUS
:JoeBiden	:wife	:JillBiden
. . .	. . .	. . .

Property Table

?s	:husband	:label	:type	:wife
:JillBiden	:JoeBiden	-	-	-
:JoeBiden	-	"JoeBiden"@en	:POTUS	:JillBiden
. . .	. . .	. . .	. . .	. . .

Property Tables

:husband

?s	?o
:JillBiden	:JoeBiden
. . .	. . .

:label

?s	?o
:JoeBiden	"JoeBiden"@en
. . .	. . .

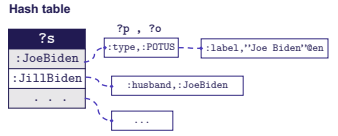
:type

?s	?o
:JoeBiden	:POTUS
. . .	. . .

:wife

?s	?o
:JoeBiden	:JillBiden
. . .	. . .

Hash table



Design Choice 1

Subdivision

reduce search space by fragmenting data

What's Best ?

Representation can be "**compatible**" to an **access pattern in 3 ways:**

- **seek compatible:** the first result can be retrieved in a single random access
- **sequence compatible:** the remaining results can be computed without random access
- **selection compatible:** if no excess results are retrieved by the computation

```
SELECT ?w
WHERE {
      ?w :label: ?l .

}
```

O(|E|)

O(|E|)

List of triples

?s	?p	?o
:JoeBiden	:label	"JoeBiden"@en
:JoeBiden	:type	:POTUS
:JillBiden	:husband	:JoeBiden
:JoeBiden	:wife	:JillBiden
. . .	. . .	. . .

O(|E|)

O(log(|E|))

SORTED List of triples

?s	?p	?o
:JillBiden	:husband	:JoeBiden
:JoeBiden	:label	"JoeBiden"@en
:JoeBiden	:type	:POTUS
:JoeBiden	:wife	:JillBiden
. . .	. . .	. . .

O(|N|)

O(log(|N|) +|L|)

Property Table

?s	:husband	:label	:type	:wife
:JillBiden	:JoeBiden	-	-	-
:JoeBiden	-	"JoeBiden"@en	:POTUS	:JillBiden
. . .	. . .	. . .	. . .	. . .

$O(|E_\ell|)$

$O(|E_\ell|)$

Property Tables

:husband	
?s	?o
:JillBiden	:JoeBiden
. . .	. . .

:label	
?s	?o
:JoeBiden	"JoeBiden"@en
. . .	. . .

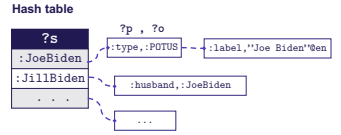
:type	
?s	?o
:JoeBiden	:POTUS
. . .	. . .

:wife	
?s	?o
:JoeBiden	:JillBiden
. . .	. . .

O(|E|)

$O(|E_n|)$

Hash table



Design Choice 1

Subdivision

reduce search space by fragmenting data

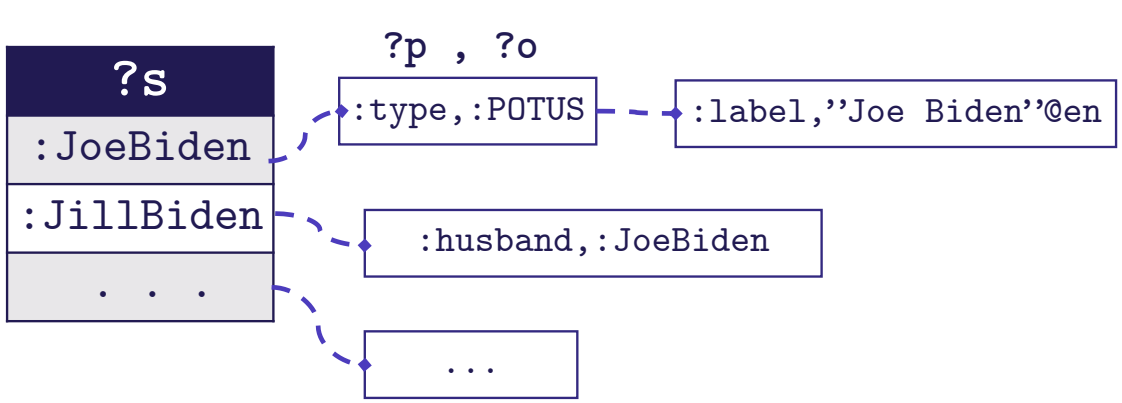
Multiple Representations

$O(|E|)$

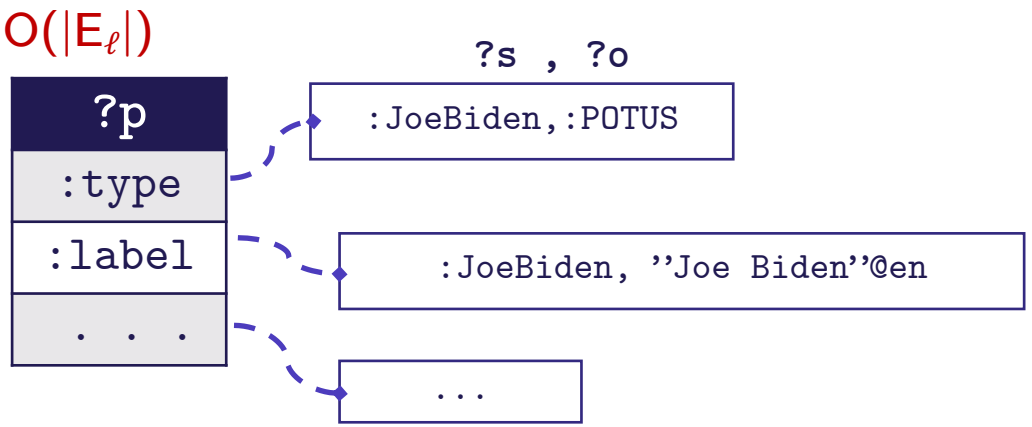
$O(|E_n|)$

```
SELECT ?w
WHERE {
      ?w :label: ?l .
      ?w :type ?t .
}
```

Hash table



$O(|E_\ell|)$



Design Choice 2

Redundancy

add copies of data with compatible data access methods

The Design Space

Design Choice 3

Compression

compress data representation to reduce bytes transferred

e.g., replace IRIs with IDs

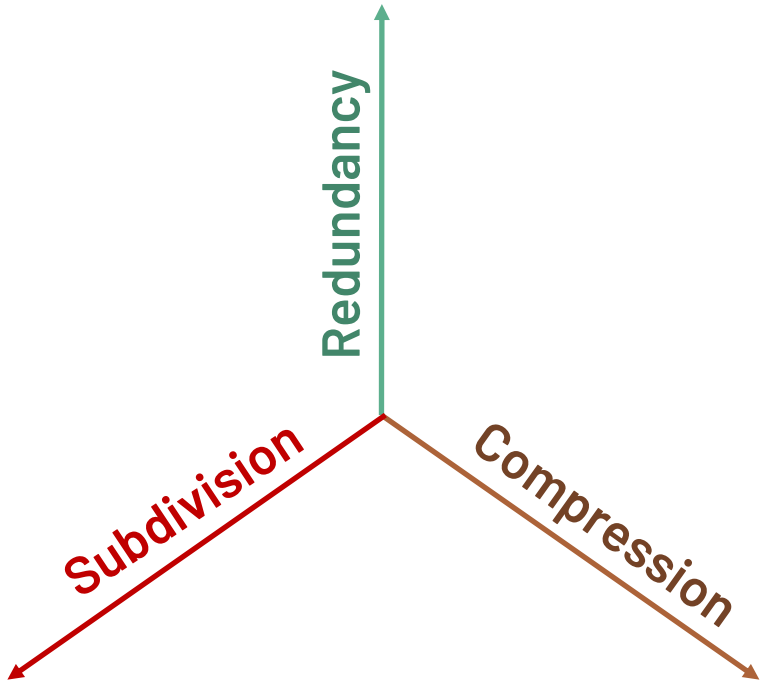


Table 2 Summary of data representation design space axes

Axis	Minimal Extreme	Maximal Extreme	Positive Effect	Negative Effect
Subdivision	Unsorted file	Pointers between every related data item	↓ # unneeded data items	↑ # random seeks
Compression	No compression	Compress all subdivisions in all representations	↓ # read bytes	↑ Decompression cost
Redundancy	Single representation	One representation for each possible BGP	↓ # random seeks	↑ maintenance cost, storage cost, query optimization time

The Design Space

Thanks
For your attention
Questions?

The **SCR** design space analysis provides:

- detailed analysis of ***access patterns***
- ***cost model*** *to evaluate the* ***compatibility*** *of different data representations*
- *Analysis of* ***prevalence of access patterns*** *in existing workloads*
- ***opportunity for identifying unexplored solutions*** *(automatically?)*

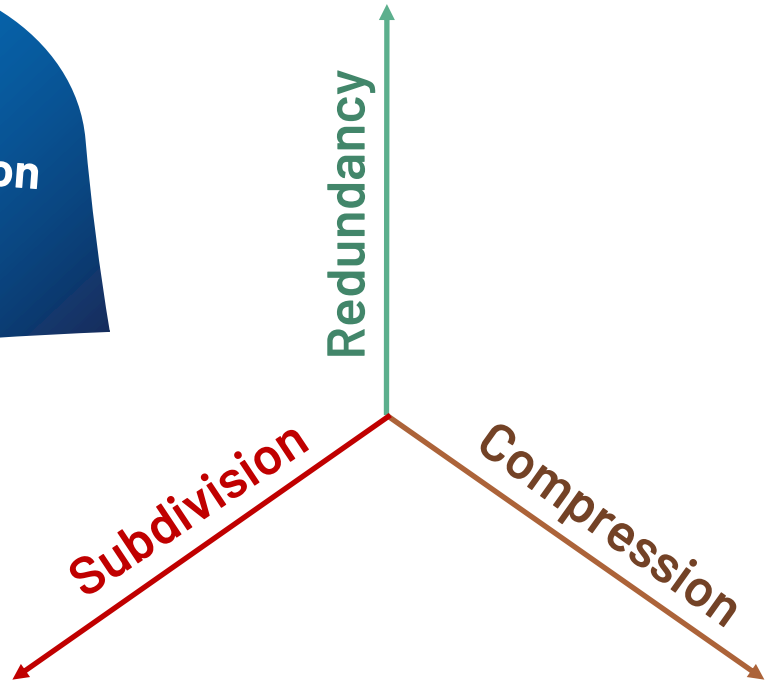


Table 16 Prevalence of access patterns in popular RDF benchmarks

		Constants							Filter			Traversal						
Set	#	S	P	O	SP	PO	SO	SPO	[ ]	[ )	Sp*	S·O	O·S	SkO	OkS	SP*O	OP*S	All
BioBench	22	0.00	0.77	0.00	0.14	0.86	0.00	0.00	0.00	0.00	0.14	0.77	0.86	0.64	0.36	0.00	0.00	0.86
WikiData	46	0.00	0.93	0.02	0.00	0.83	0.00	0.00	0.11	0.15	0.11	0.89	0.80	0.22	0.26	0.04	0.50	0.43
ComplexQ	3443	0.00	0.88	0.00	0.34	0.96	0.00	0.00	0.00	0.02	0.98	1.00	0.96	0.38	0.40	0.00	0.00	0.0*
SWDF	64030	0.24	0.23	0.01	0.31	0.03	0.0*	0.10	0.00	0.00	0.0*	0.56	0.05	0.21	0.0*	0.00	0.00	0.23
DBpedia	169721	0.05	0.69	0.03	0.19	0.71	0.0*	0.0*	0.0*	0.01	0.64	0.87	0.74	0.56	0.07	0.0*	0.0*	0.78