Data Compression Conference (DCC 2020)
Snowbird, Utah
March 24-27

Revisiting compact RDF stores based on k²-trees

Nieves R. Brisaboa
Ana Cerdeira-Pena
Guillermo de Bernardo
Antonio Fariña

UNIVERSIDADE DA CORUÑA

citic

Laboratorio de Bases de Datos

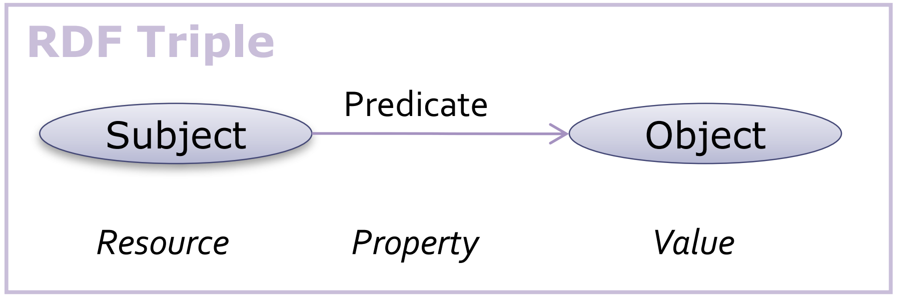
This work has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 690941

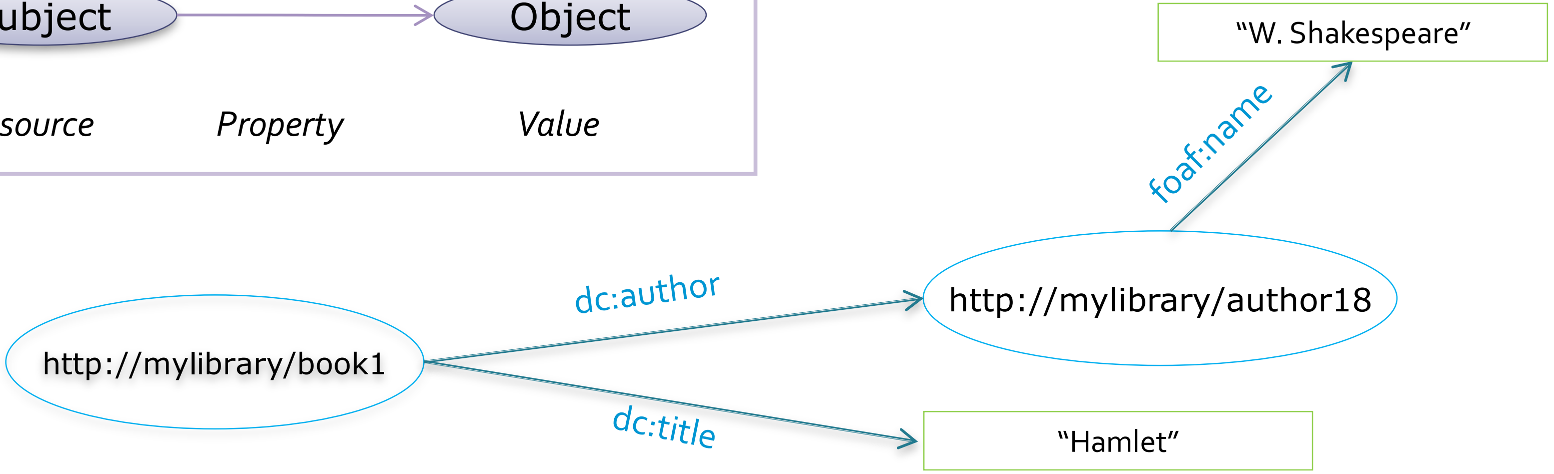
RDF and the Web of Data

- Web of Data
  - How to share structured knowledge on the Web?

- Resource Description Framework (RDF)
  - W3C Recommendation
  - Publish structured data
  - Graph-based data model for any kind of resource

RDF and SPARQL





http://mylibrary/book1	dc:author	http://mylibrary/author18
http://mylibrary/author18	foaf:name	"W. Shakespeare"
http://mylibrary/book1	dc:title	"Hamlet"

RDF and SPARQL

- Triple patterns
  - (s,?p,?o)

    (http://mylibrary/book1, ?p, ?o)
  - (s,p,o) (s,p,?o) (s,?p,o)
- SPARQL

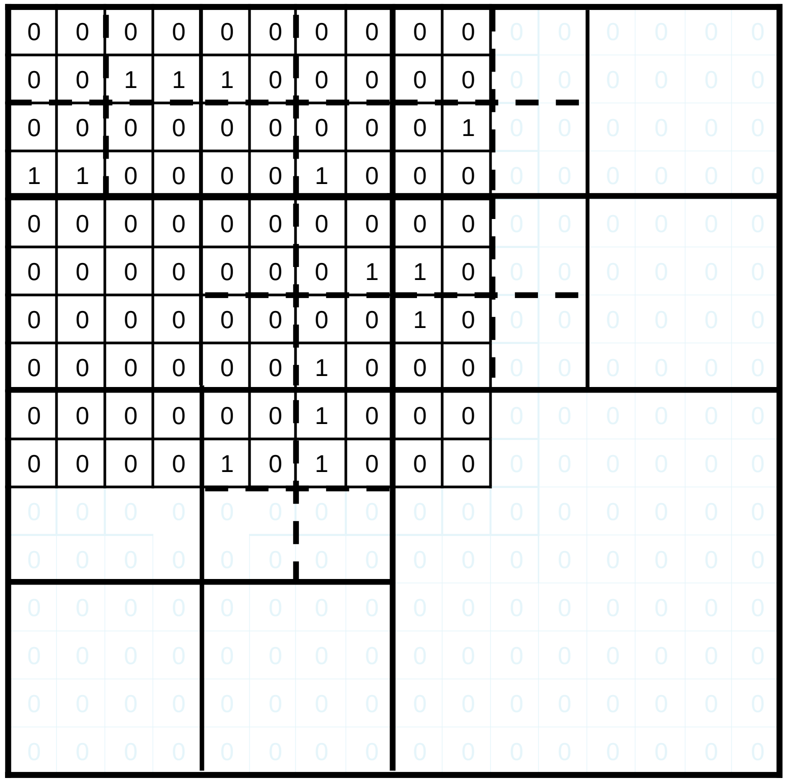
```
SELECT ?name
WHERE{
?book     dc:author     ?writer
?book     dc:title      "Hamlet" .
?writer   foaf:name     ?name .
}
```

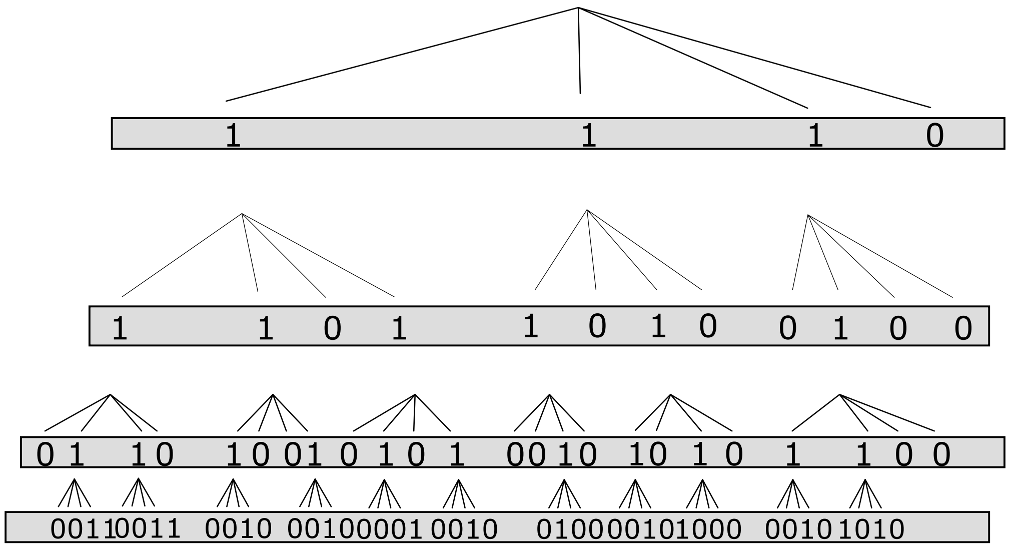
http://mylibrary/book1	dc:author	http://mylibrary/author18
http://mylibrary/author18	foaf:name	"W. Shakespeare"
http://mylibrary/book1	dc:title	"Hamlet"

RDF stores

- Compact representation of the RDF data
- Solutions
  - Relational databases
  - Custom solutions (multi-indexing structures)
  - Compact data structures: RDFCSA, k^2-triples
- Ability to answer SPARQL queries
  - Key element: triple patterns

The k²-tree

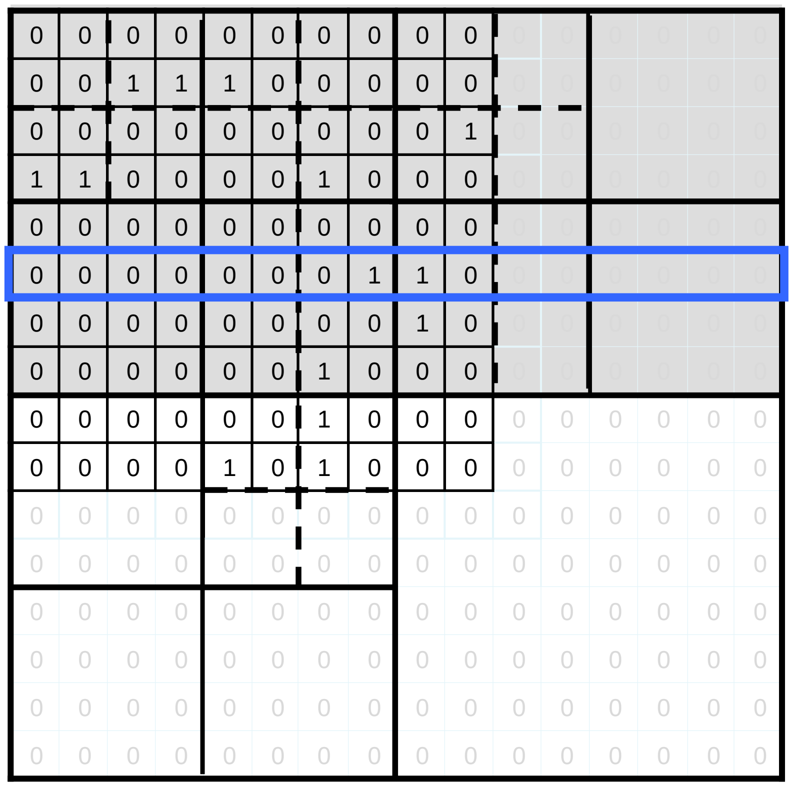


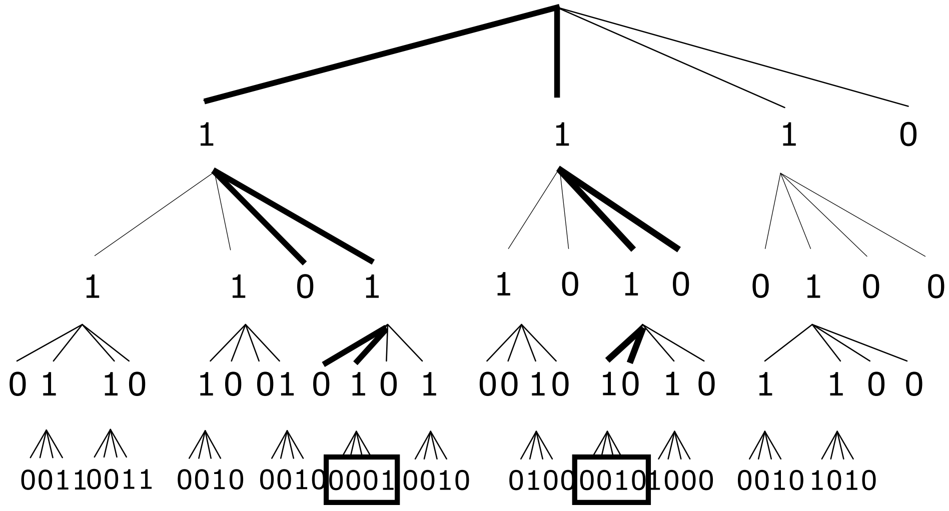


T = 1110110110100100011010010101001010101100

L = 0011001100100010000100100100001010000010101010

The k²-tree





```
     0  2   4  6   8  10  12  14  16  18 20  22  24  26 28  30
T = 1110 1101 1010010001101001 0101 001010101100
```

$$\text{Children}(p) = \text{rank}_1(T,p)\cdot k^2$$

```
     0  2   4   6   8  10  12 14  16  18  20  22 24  26  28  30
L = 0011001100100010 0001 001001000010100000101010
```

k²-triples

- Vertical partitioning
  - Generate a k^2-tree per predicate
  - Triple patterns are translated into cell/row/column/range queries
    - (s,p,?) → Find the ones in row s in the k^2-tree for predicate p

- k^2-triples compression overcomes other RDF stores

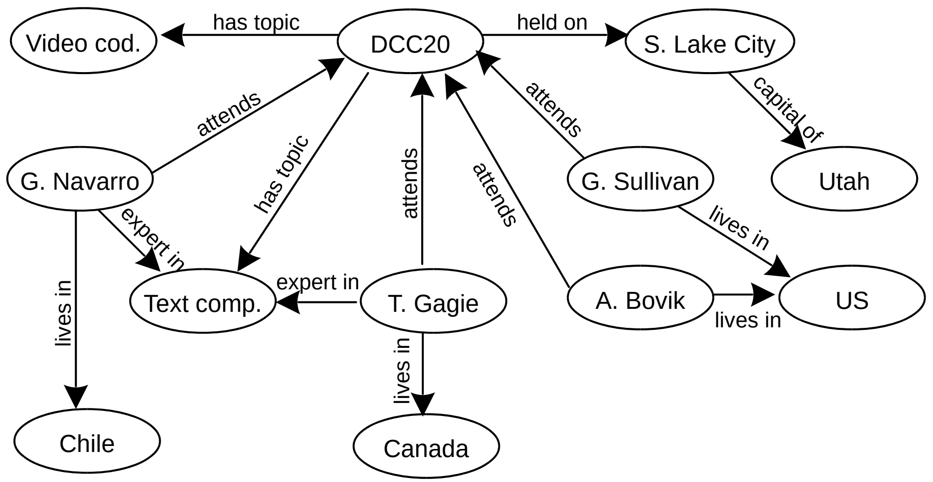
- Performance suffers in some queries

- k^2-triples+
  - Vertical partitioning inefficient with a large number of predicates
  - Additional S-P and O-P indexes to speed up queries

BMatrix

- Use k^2-trees to achieve good compression of the data
  - Simple and efficient compression
  - Queries should be translated into k^2-tree operations
- Abandon vertical partitioning idea
  - Look for other tradeoff oportunites
  - Make the structure independent of the number of predicates
  - Improve performance in queries with unbound predicate

BMatrix: dictionary encoding



(DCC20, held on , S. Lake City)
(S. Lake City, capital of, Utah)
(DCC20, has topic, Text comp.)
(DCC20, has topic, Video cod.)
(G. Navarro, attends, DCC20)
(G. Navrro, lives in, Chile)
(G. Navarro, expert in, Text comp.)
(T.Gagie, attends, DCC20)
(T.Gagie, lives in, Canada)
(T. Gagie, expert in, Text comp.)
(A. Bovik, attends, DCC20)
(A. Bovik, lives in, US)
(G. Sullivan, attends, DCC20)
(G. Sullivan, lives in, US)

SO	1	DCC20
	2	S. Lake City
S	3	A. Bovik
	4	G. Navarro
	5	G. Sullivan
	6	T. Gagie
O	3	Canada
	4	Chile
	5	Text comp.
	6	US
	7	Utah
	8	Video cod.

P	1	attends
	2	capital of
	3	expert in
	4	has topic
	5	held on
	6	lives in

(1,5,2)
(2,2,7)
(1,4,5)
(1,4,8)
(4,1,1)
(4,6,4)
(4,3,5)
(6,1,1)
(6,6,3)
(6,3,5)
(3,1,1)
(3,6,6)
(5,1,1)
(5,6,6)

BMatrix: data structure

- Sort triples in P-O-S order
  - Any P-based order can be applied

(1,5,2)	(3,1,1)
(2,2,7)	(4,1,1)
(1,4,5)	(5,1,1)
(1,4,8)	(6,1,1)
(4,1,1)	(2,2,7)
(4,6,4)	(4,3,5)
(4,3,5)	(6,3,5)
(6,1,1)	(1,4,5)
(6,6,3)	(1,4,8)
(6,3,5)	(1,5,2)
(3,1,1)	(6,6,3)
(3,6,6)	(4,6,4)
(5,1,1)	(3,6,6)
(5,6,6)	(5,6,6)

- Data structures:
  - ST: subject-triple matrix
  - OT: object-triple matrix
  - BP: bitmap marking predicate changes

BMatrix: data structure

ST

	(**3**,1,1)	(**4**,1,1)	(**5**,1,1)	(**6**,1,1)	(**2**,2,7)	(**4**,3,5)	(**6**,3,5)	(**1**,4,5)	(**1**,4,8)	(**1**,5,2)	(**6**,6,3)	(**4**,6,4)	(**3**,6,6)	(**5**,6,6)
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1								1	1	1				
2					1									
3	1												1	
4		1				1						1		
5			1											1
6				1			1				1			

OT

	(3,1,**1**)	(4,1,**1**)	(5,1,**1**)	(6,1,**1**)	(2,2,**7**)	(4,3,**5**)	(6,3,**5**)	(1,4,**5**)	(1,4,**8**)	(1,5,**2**)	(6,6,**3**)	(4,6,**4**)	(3,6,**6**)	(5,6,**6**)
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	1	1	1										
2										1				
3											1			
4												1		
5						1	1	1						
6													1	1
7					1									
8									1					

K²-trees

	(3,**1**,1)	(4,**1**,1)	(5,**1**,1)	(6,**1**,1)	(2,**2**,7)	(4,**3**,5)	(6,**3**,5)	(1,**4**,5)	(1,**4**,8)	(1,**5**,2)	(6,**6**,3)	(4,**6**,4)	(3,**6**,6)	(5,**6**,6)
BP	**1**	0	0	0	**1**	**1**	0	**1**	0	**1**	**1**	0	0	0

Bitmap (rank/select)

BMatrix: triple patterns

- (4,3,5)
  - Find range of triples for p=3
  - Row query (row 4) in ST
    - Limited to range of triples

      T6= (4,3,?)
  - Check cell in (5,6) OT

    T6 = (4,3,5)

ST

	(3,1,1)	(4,1,1)	(5,1,1)	(6,1,1)	(2,2,7)	(4,3,5)	(6,3,5)	(1,4,5)	(1,4,8)	(1,5,2)	(6,6,3)	(4,6,4)	(3,6,6)	(5,6,6)
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1								1	1	1				
2					1									
3	1												1	
4		1				1						1		
5			1											1
6				1			1				1			

OT

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	1	1	1										
2										1				
3											1			
4												1		
5						1	1	1						
6													1	1
7					1									
8									1					

BP

1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0	0	0	1	1	0	1	0	1	1	0	0	0

BMatrix: triple patterns

- (4,?p,?o)
  - Row query in ST
    - T2, T6, T12
  - For each result, column query in OT
    - T2: (4,?,1)
    - T6: (4,?,5)
    - T12: (4,?,4)
  - Find predicates in BP
    - $\text{rank}_1(BP, p)$

	(3,1,1)	(4,1,1)	(5,1,1)	(6,1,1)	(2,2,7)	(4,3,5)	(6,3,5)	(1,4,5)	(1,4,8)	(1,5,2)	(6,6,3)	(4,6,4)	(3,6,6)	(5,6,6)
ST	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1								1	1	1				
2					1									
3	1												1	
4		1				1						1		
5			1											1
6				1			1				1			

OT	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	1	1	1										
2										1				
3											1			
4												1		
5						1	1	1						
6													1	1
7					1									
8									1					

BP	1	0	0	0	1	1	0	1	0	1	1	0	0	0

BMatrix: optimizations

- Space improvements:
  - Matrices have a single 1 per column
  - Custom compression of k^2-tree lower levels
- Time improvements
  - Optimized column queries
  - Custom solution for BP
- Several alternatives for some queries
  - Chained evaluation vs Query-and-merge

    (s,?p,o)

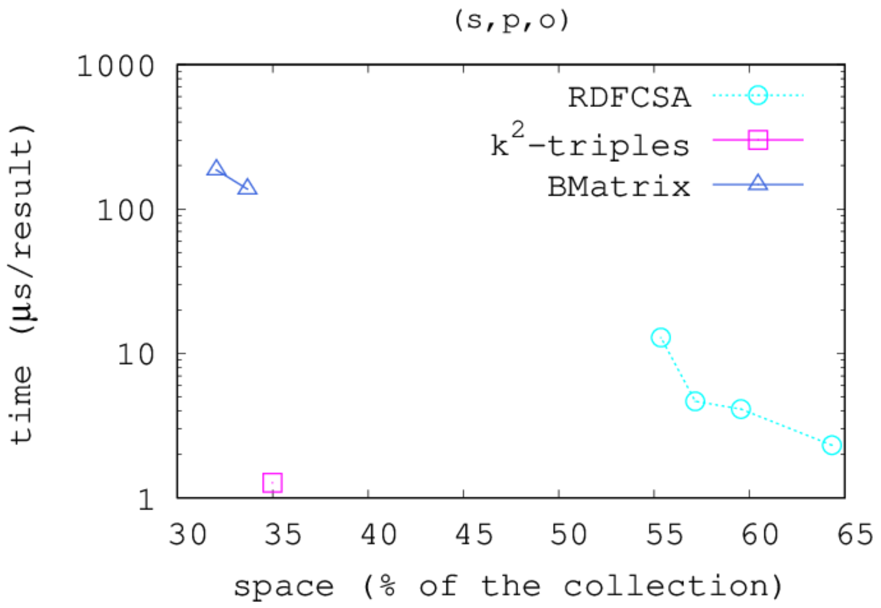
Experimental evaluation

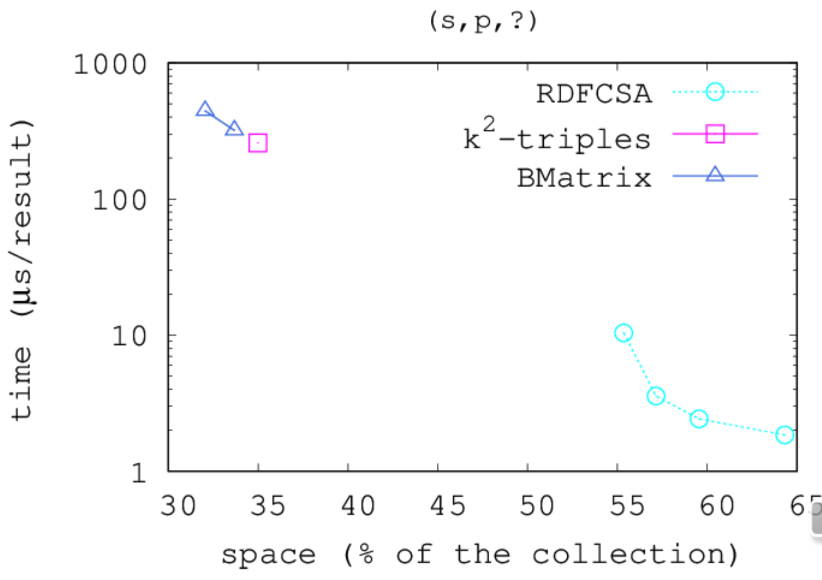
- ❑ DBPedia dataset (v3.5.1)
  - ➢ 232 million triples
  - ➢ 40k predicates

- ❑ Baselines:
  - ➢ k^2-triples and k^2-triples+
    - ❑ Same configuration in K2-trees in both approaches
    - ❑ Custom compression of lower levels in our solution
  - ➢ RDFCSA

Experimental evaluation

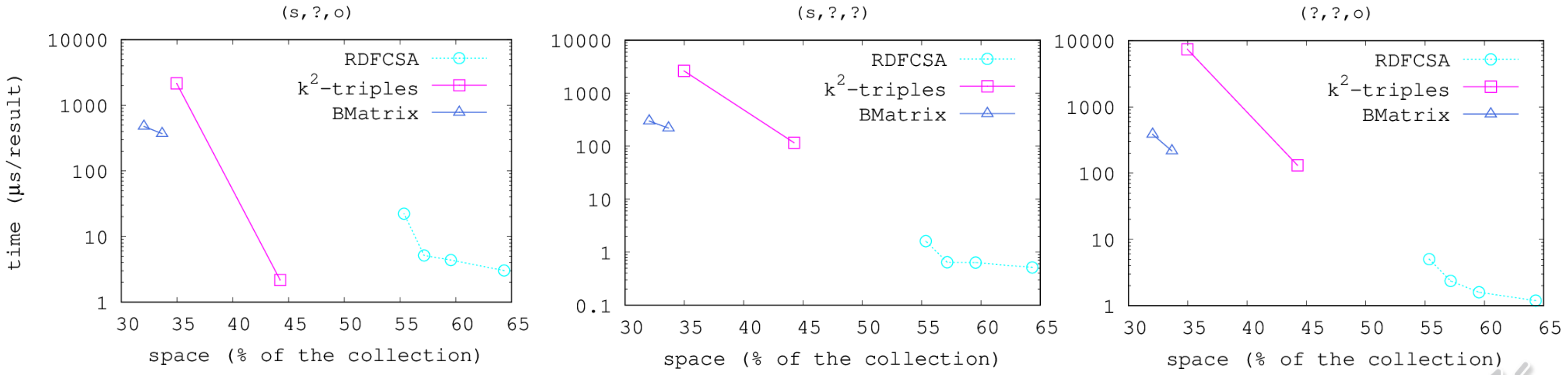
- Bound predicate queries
  - BMatrix smallest solution
  - Performance varies:
    - Comparable in more complex patterns
    - Slower in simpler patterns





Experimental evaluation

- Unbound predicate queries
  - BMatrix comparable to k²-triples+ using 40% less space
  - k²-triples very inefficient in most patterns



Conclusions

- New representation for RDF data
  - Designed for complex datasets → many predicates
  - Improves compression results of existing solutions
- Key features
  - Query times are relatively stable for all triple pattern queries
  - Space/time trade-off. Results depend on type of queries
- Future work
  - Support for joins and performance
  - Search for application-specific arrangements and tradeoffs

Data Compression Conference (DCC 2020)
Snowbird, Utah
March 24-27

Revisiting compact RDF stores based on k²-trees

Nieves R. Brisaboa
Ana Cerdeira-Pena
Guillermo de Bernardo
Antonio Fariña

gdebernardo@udc.es

This work has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 690941