# SPARQL query performance

But first, SHACL demo

# BGP (Basic Graph Pattern)

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE {
  ?x foaf:name ?name .
  ?x foaf:mbox ?mbox .
}
```

Get all triples from the database that match

$?x^1$ foaf:name ?name .

And all triples that match

$?x^2$ foaf:mbox ?mbox .

Then join them together where $?x^1 == ?x^2$

# Filters

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE  {
  ?x foaf:name ?name .
  ?x foaf:mbox ?mbox .
  FILTER(?name == "Håvard")
}
```

Get all triples from the database that match

?$x^1$ foaf:name ?name .

And all triples that match

?$x^2$ foaf:mbox ?mbox .

Then join them together where ?$x^1$ == ?$x^2$

**Then filter that result so that ?name == “Håvard”**

# Filter rewrite

- Move filter as close to the part of the BGP where it applies.
- Reduce the amount of data early
- Common rewrite
- Haven’t found any triples stores with function indexes

# Filter rewrite

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE  {
  ?x foaf:name ?name .
  FILTER(?name == "Håvard")
  ?x foaf:mbox ?mbox .
}
```

Get all triples from the database that match

$?x^1$ foaf:name ?name .

**Filter those triples so that ?name == “Håvard”**

And all triples that match

$?x^2$ foaf:mbox ?mbox .

Then join them together where $?x^1 == ?x^2$

# Indexes

- Triple: S (subject) P (predicate) O (object)
- `?x foaf:name ?name .`
  - We only know P
- We need a PSO index

# RDF

ex:Håvard foaf:name “Håvard”

ex:Håvard foaf:mbox “haavard.ottestad@acando.no”

ex:Veronika foaf:name “Veronika”

ex:Veronika foaf:mbox “veronika.heimsbakk@acando.no”

# PSO Index

| P | S | O |
|---|---|---|
| foaf:mbox | ex:Håvard | “haavard.ottestad@acando.no” |
| foaf:mbox | ex:Veronika | “veronika.heimsbakk@acando.no” |
| foaf:name | ex:Håvard | “Håvard” |
| foaf:name | ex:Veronika | “Veronika” |

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
SELECT ?x ?name ?mbox
WHERE  {
  ?x foaf:name ?name .
  ?x foaf:mbox ?mbox .
}
```

# Join on ?x

| P | S | O |
|---|---|---|
| foaf:mbox | ex:Håvard | “haavard...” |
| foaf:mbox | ex:Veronika | “veronika...” |
| foaf:name | ex:Håvard | “Håvard” |
| foaf:name | ex:Veronika | “Veronika” |

| P | S | O |
|---|---|---|
| foaf:mbox | ex:Håvard | “haavard...” |
| foaf:mbox | ex:Veronika | “veronika...” |
| foaf:name | ex:Håvard | “Håvard” |
| foaf:name | ex:Veronika | “Veronika” |

| ?x | ?name | ?mbox |
|---|---|---|
|  |  |  |
|  |  |  |

# Join on ?x

| P | S | O |
|---|---|---|
| foaf:mbox | ex:Håvard | “haavard...” |
| foaf:mbox | ex:Veronika | “veronika...” |
| foaf:name | ex:Håvard | “Håvard” |
| foaf:name | ex:Veronika | “Veronika” |

| P | S | O |
|---|---|---|
| foaf:mbox | ex:Håvard | “haavard...” |
| foaf:mbox | ex:Veronika | “veronika...” |
| foaf:name | ex:Håvard | “Håvard” |
| foaf:name | ex:Veronika | “Veronika” |

| ?x | ?name | ?mbox |
|---|---|---|
| ex:Håvard | “Håvard” | |
| | | |

# Join on ?x

| P | S | O |
|---|---|---|
| foaf:mbox | ex:Håvard | “haavard...” |
| foaf:mbox | ex:Veronika | “veronika...” |
| foaf:name | ex:Håvard | “Håvard” |
| foaf:name | ex:Veronika | “Veronika” |

| P | S | O |
|---|---|---|
| foaf:mbox | ex:Håvard | “haavard...” |
| foaf:mbox | ex:Veronika | “veronika...” |
| foaf:name | ex:Håvard | “Håvard” |
| foaf:name | ex:Veronika | “Veronika” |

| ?x | ?name | ?mbox |
|---|---|---|
| ex:Håvard | “Håvard” | “haavard...” |
| | | |

# Join on ?x

| P | S | O |
|---|---|---|
| foaf:mbox | ex:Håvard | “haavard...” |
| foaf:mbox | ex:Veronika | “veronika...” |
| foaf:name | ex:Håvard | “Håvard” |
| foaf:name | ex:Veronika | “Veronika” |

| P | S | O |
|---|---|---|
| foaf:mbox | ex:Håvard | “haavard...” |
| foaf:mbox | ex:Veronika | “veronika...” |
| foaf:name | ex:Håvard | “Håvard” |
| foaf:name | ex:Veronika | “Veronika” |

| ?x | ?name | ?mbox |
|---|---|---|
| ex:Håvard | “Håvard” | “haavard...” |
| | | |

# Join on ?x

| P | S | O |
|---|---|---|
| foaf:mbox | ex:Håvard | “haavard...” |
| foaf:mbox | ex:Veronika | “veronika...” |
| foaf:name | ex:Håvard | “Håvard” |
| foaf:name | ex:Veronika | “Veronika” |

| P | S | O |
|---|---|---|
| foaf:mbox | ex:Håvard | “haavard...” |
| foaf:mbox | ex:Veronika | “veronika...” |
| foaf:name | ex:Håvard | “Håvard” |
| foaf:name | ex:Veronika | “Veronika” |

| ?x | ?name | ?mbox |
|---|---|---|
| ex:Håvard | “Håvard” | “haavard...” |
| ex:Veronika | “Veronika” | |

# Join on ?x

| P | S | O |
|---|---|---|
| foaf:mbox | ex:Håvard | “haavard...” |
| foaf:mbox | ex:Veronika | “veronika...” |
| foaf:name | ex:Håvard | “Håvard” |
| foaf:name | ex:Veronika | “Veronika” |

| P | S | O |
|---|---|---|
| foaf:mbox | ex:Håvard | “haavard...” |
| foaf:mbox | ex:Veronika | “veronika...” |
| foaf:name | ex:Håvard | “Håvard” |
| foaf:name | ex:Veronika | “Veronika” |

| ?x | ?name | ?mbox |
|---|---|---|
| ex:Håvard | “Håvard” | “haavard...” |
| ex:Veronika | “Veronika” | “veronika...” |

# Join on ?x

| P | S | O |
|---|---|---|
| foaf:mbox | ex:Håvard | “haavard...” |
| foaf:mbox | ex:Veronika | “veronika...” |
| foaf:name | ex:Håvard | “Håvard” |
| foaf:name | ex:Veronika | “Veronika” |

| P | S | O |
|---|---|---|
| foaf:mbox | ex:Håvard | “haavard...” |
| foaf:mbox | ex:Veronika | “veronika...” |
| foaf:name | ex:Håvard | “Håvard” |
| foaf:name | ex:Veronika | “Veronika” |

| ?x | ?name | ?mbox |
|---|---|---|
| ex:Håvard | “Håvard” | “haavard...” |
| ex:Veronika | “Veronika” | “veronika...” |

# Join on ?x

| P | S | O |
|---|---|---|
| foaf:mbox | ex:Håvard | “haavard...” |
| foaf:mbox | ex:Veronika | “veronika...” |
| foaf:name | ex:Håvard | “Håvard” |
| foaf:name | ex:Veronika | “Veronika” |

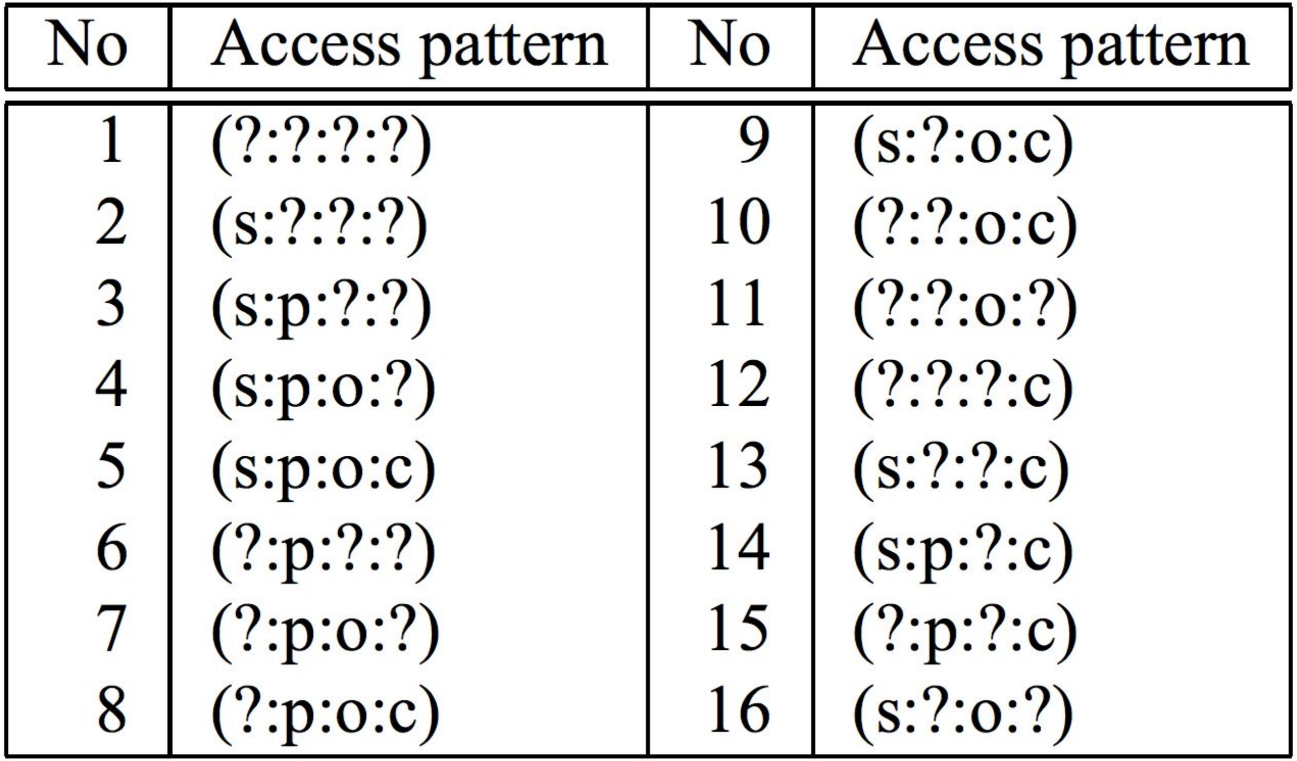
| P | S | O |
|---|---|---|
| foaf:mbox | ex:Håvard | “haavard...” |
| foaf:mbox | ex:Veronika | “veronika...” |
| foaf:name | ex:Håvard | “Håvard” |
| foaf:name | ex:Veronika | “Veronika” |

| ?x | ?name | ?mbox |
|---|---|---|
| ex:Håvard | “Håvard” | “haavard...” |
| ex:Veronika | “Veronika” | “veronika...” |

# Indexes

- B+ tree
  - Common for disk based indexes
  - Sorted
  - Supports range queries
- Indexes are usually for quads
  - PSOC (context/graph)
- 24 possible indexes (4!)

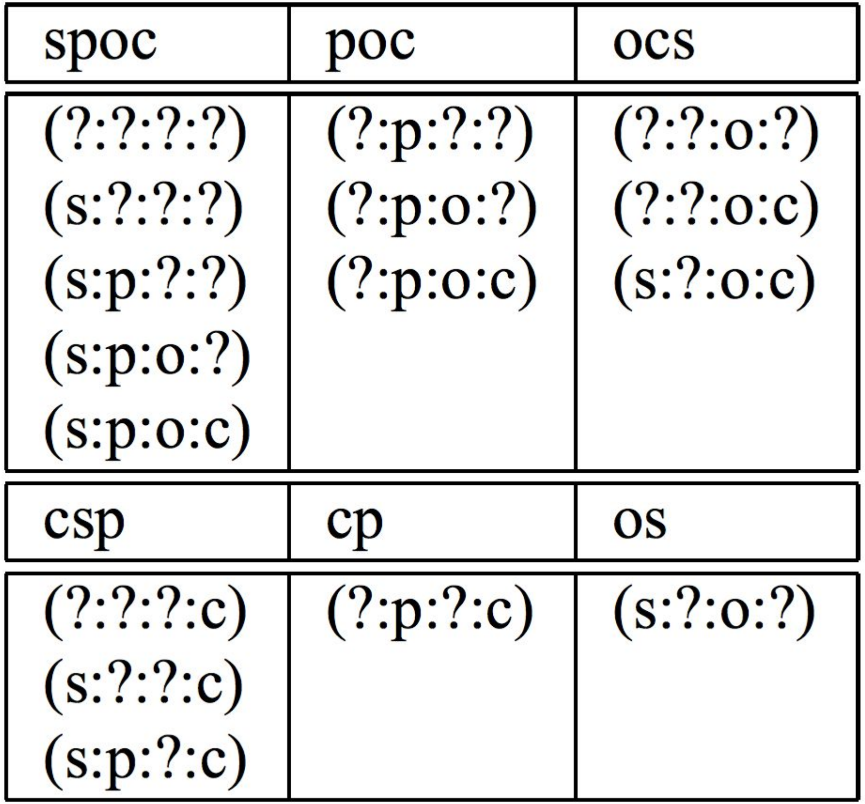
# Access patterns



| No | Access pattern | No | Access pattern |
|---|---|---|---|
| 1 | (?:?:?:?) | 9 | (s:?:o:c) |
| 2 | (s:?:?:?) | 10 | (?:?:o:c) |
| 3 | (s:p:?:?) | 11 | (?:?:o:?) |
| 4 | (s:p:o:?) | 12 | (?:?:?:c) |
| 5 | (s:p:o:c) | 13 | (s:?:?:c) |
| 6 | (?:p:?:?) | 14 | (s:p:?:c) |
| 7 | (?:p:o:?) | 15 | (?:p:?:c) |
| 8 | (?:p:o:c) | 16 | (s:?:o:?) |

Harth, Andreas, and Stefan Decker. "Optimized index structures for querying rdf from the web." *Web Congress, 2005. LA-WEB 2005. Third Latin American*. IEEE, 2005.

# 6 required indexes

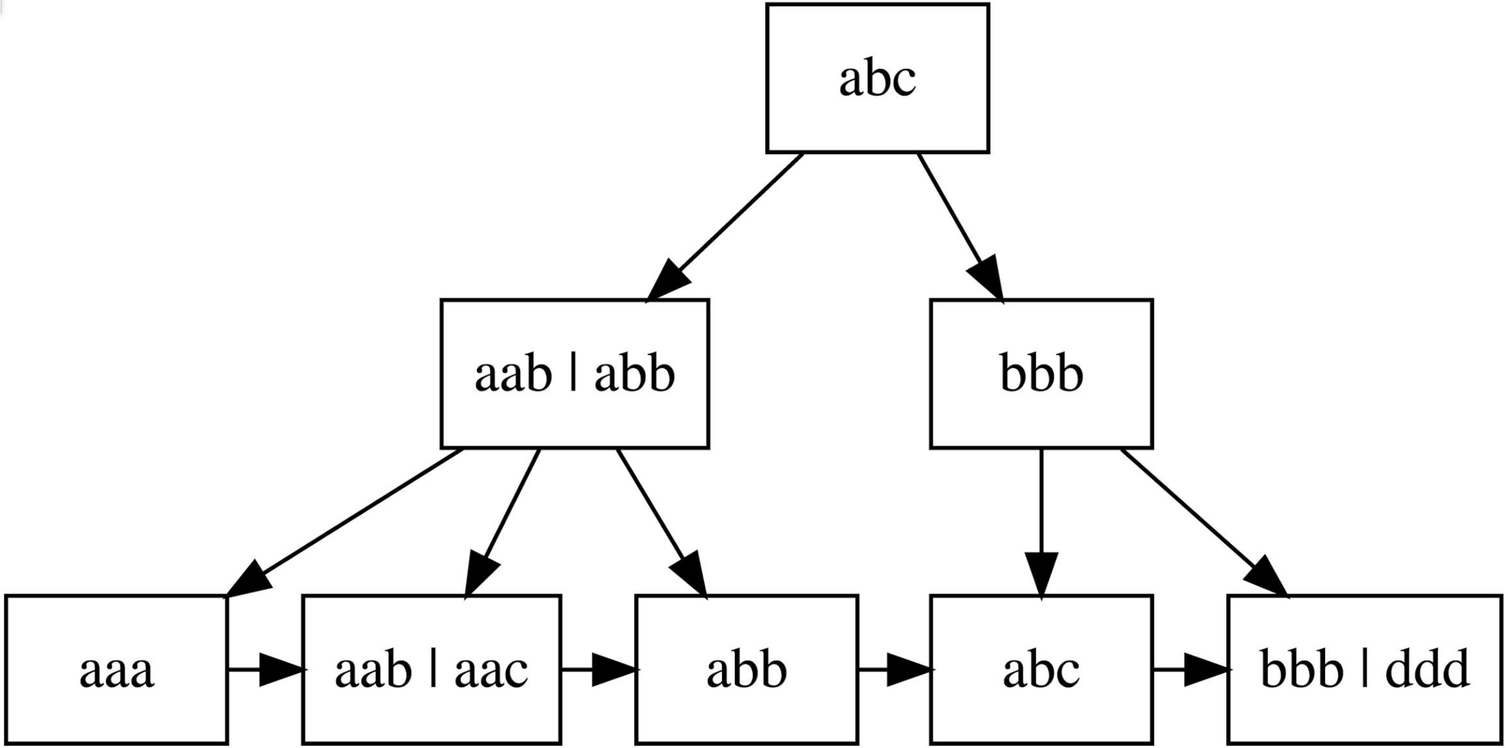


| spoc | poc | ocs |
|---|---|---|
| (?:?:?:?)<br>(s:?:?:?)<br>(s:p:?:?)<br>(s:p:o:?)<br>(s:p:o:c) | (?:p:?:?)<br>(?:p:o:?)<br>(?:p:o:c) | (?:?:o:?)<br>(?:?:o:c)<br>(s:?:o:c) |

| csp | cp | os |
|---|---|---|
| (?:?:?:c)<br>(s:?:?:c)<br>(s:p:?:c) | (?:p:?:c) | (s:?:o:?) |

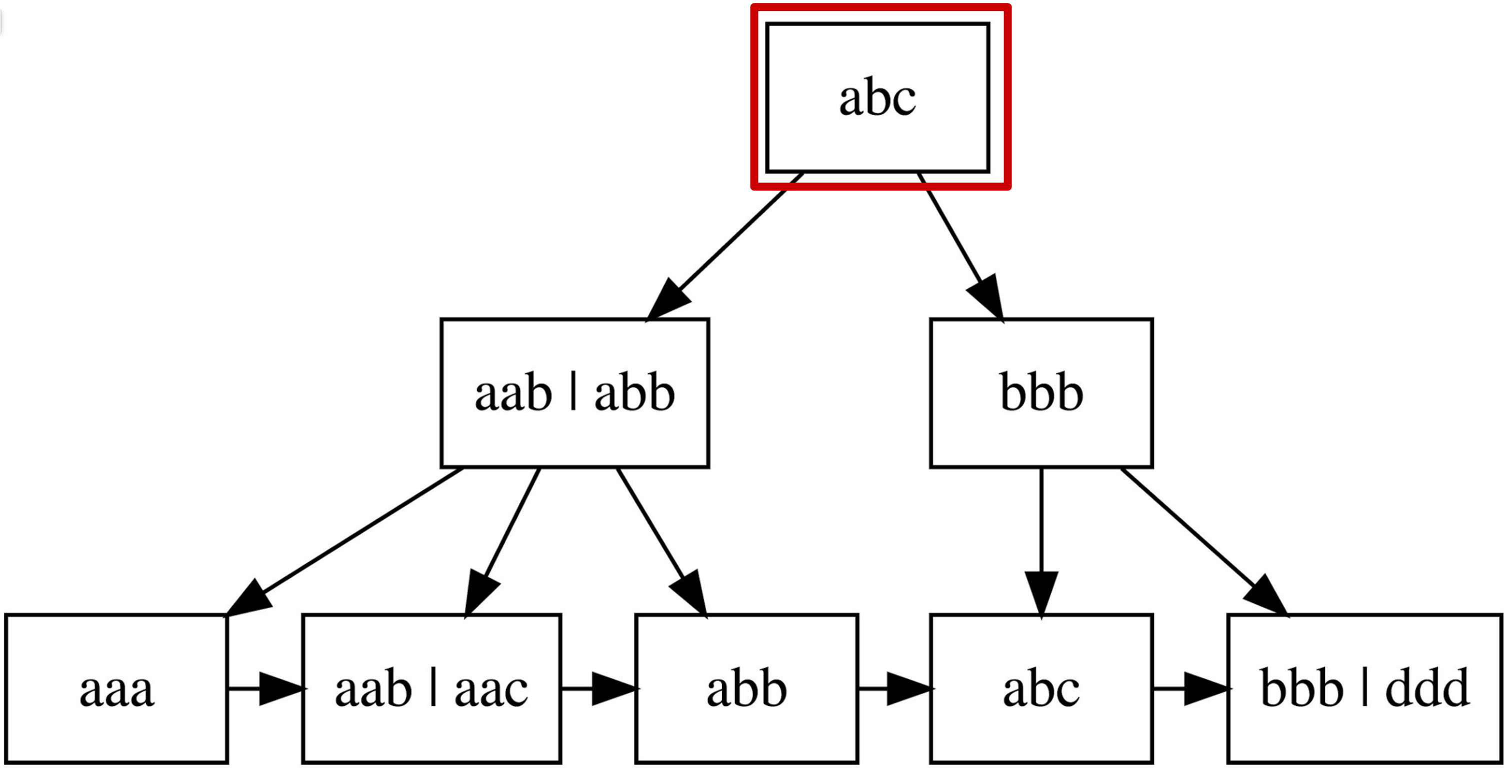
Harth, Andreas, and Stefan Decker. "Optimized index structures for querying rdf from the web." *Web Congress, 2005. LA-WEB 2005. Third Latin American*. IEEE, 2005.

# Range queries in B+ trees - **aac → abc**

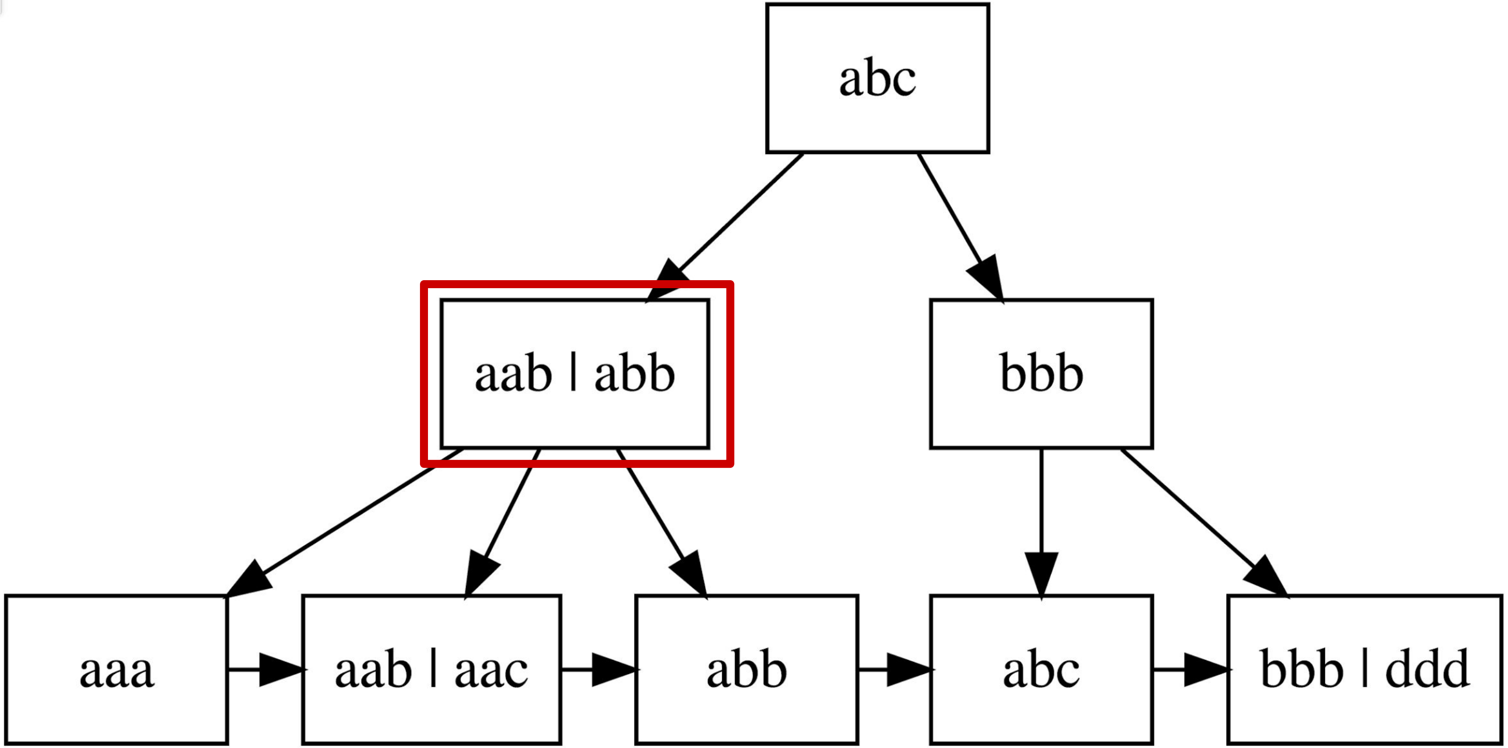


# Range queries in B+ trees - **aac → abc**

g

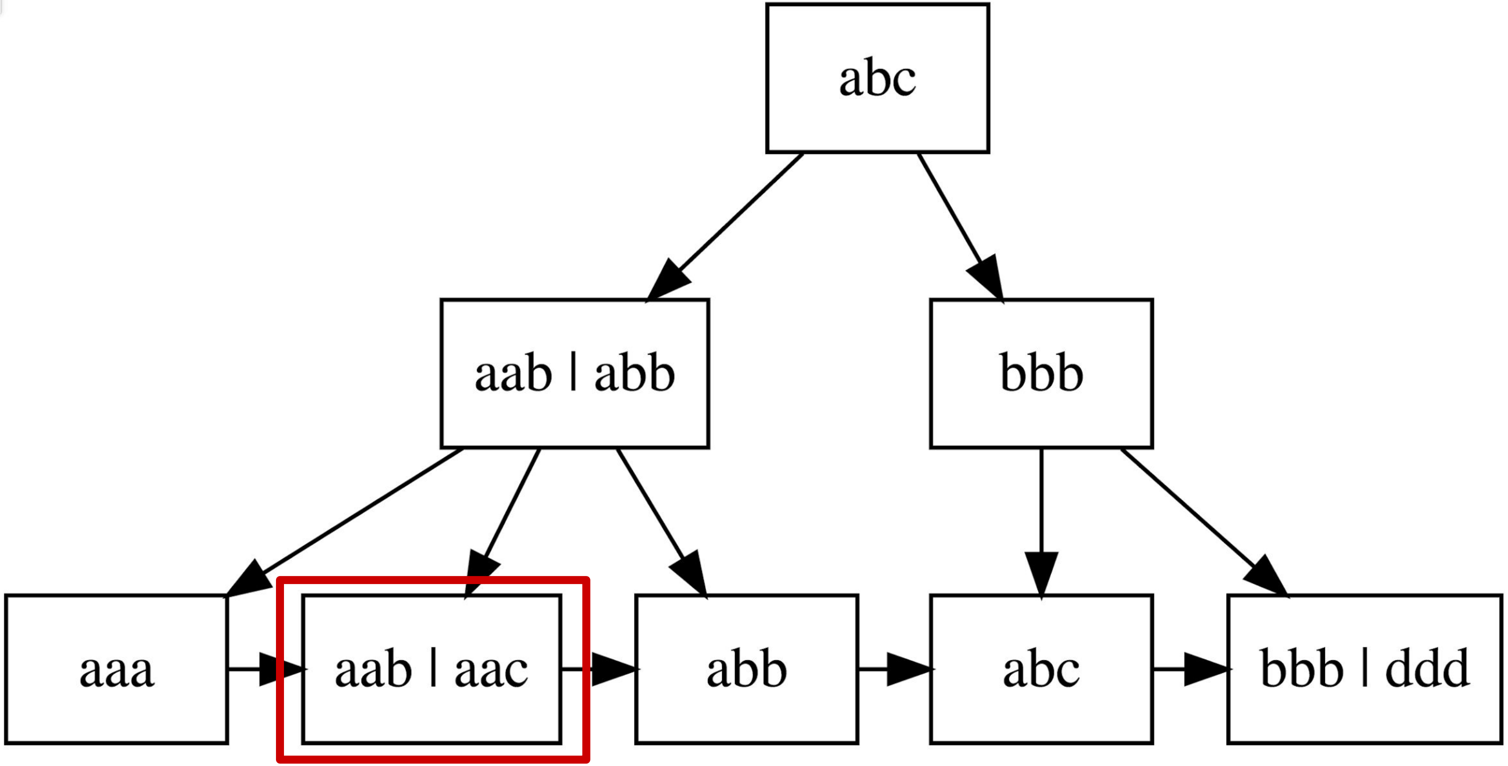


# Range queries in B+ trees - **aac → abc**

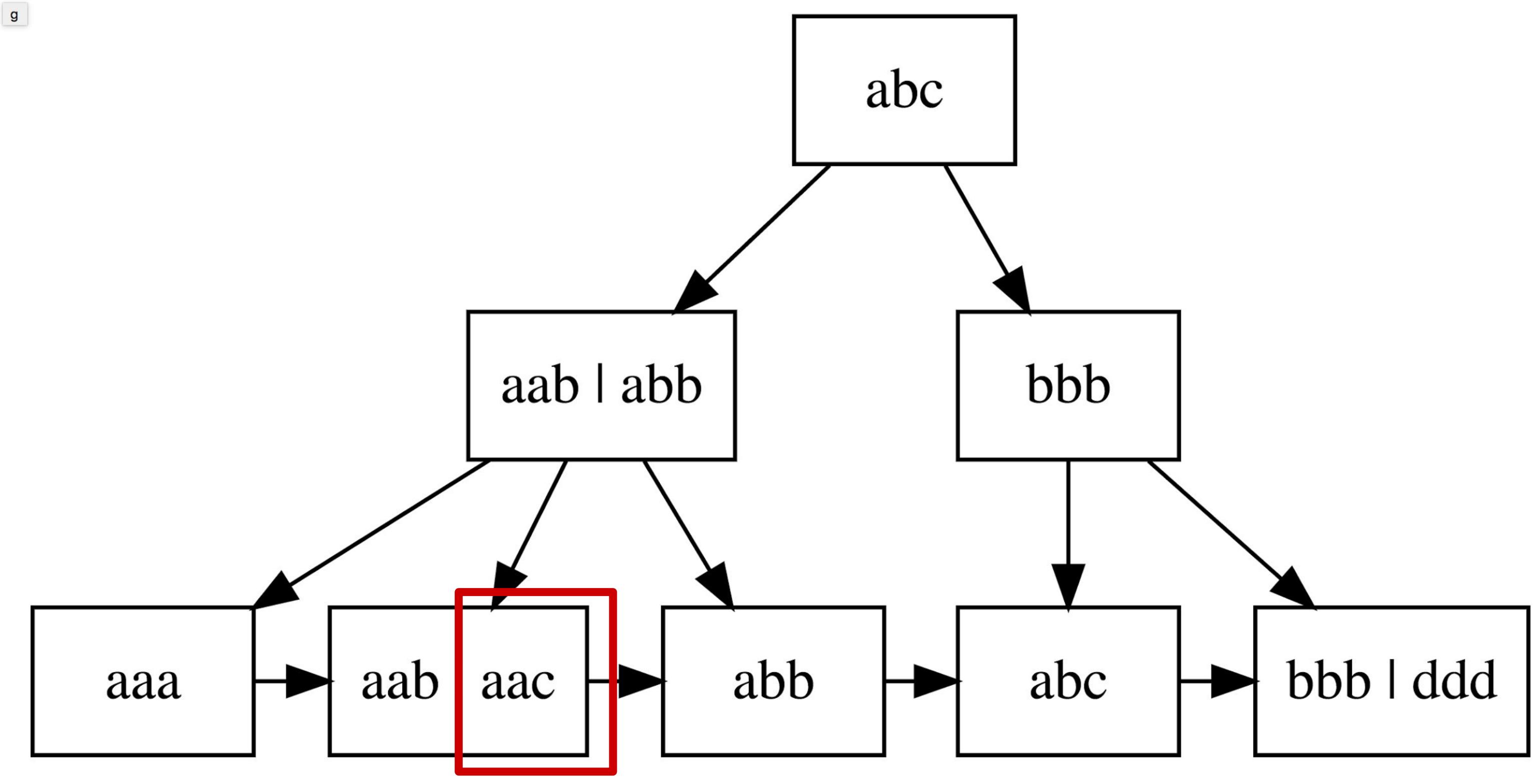


# Range queries in B+ trees - **aac** → **abc**

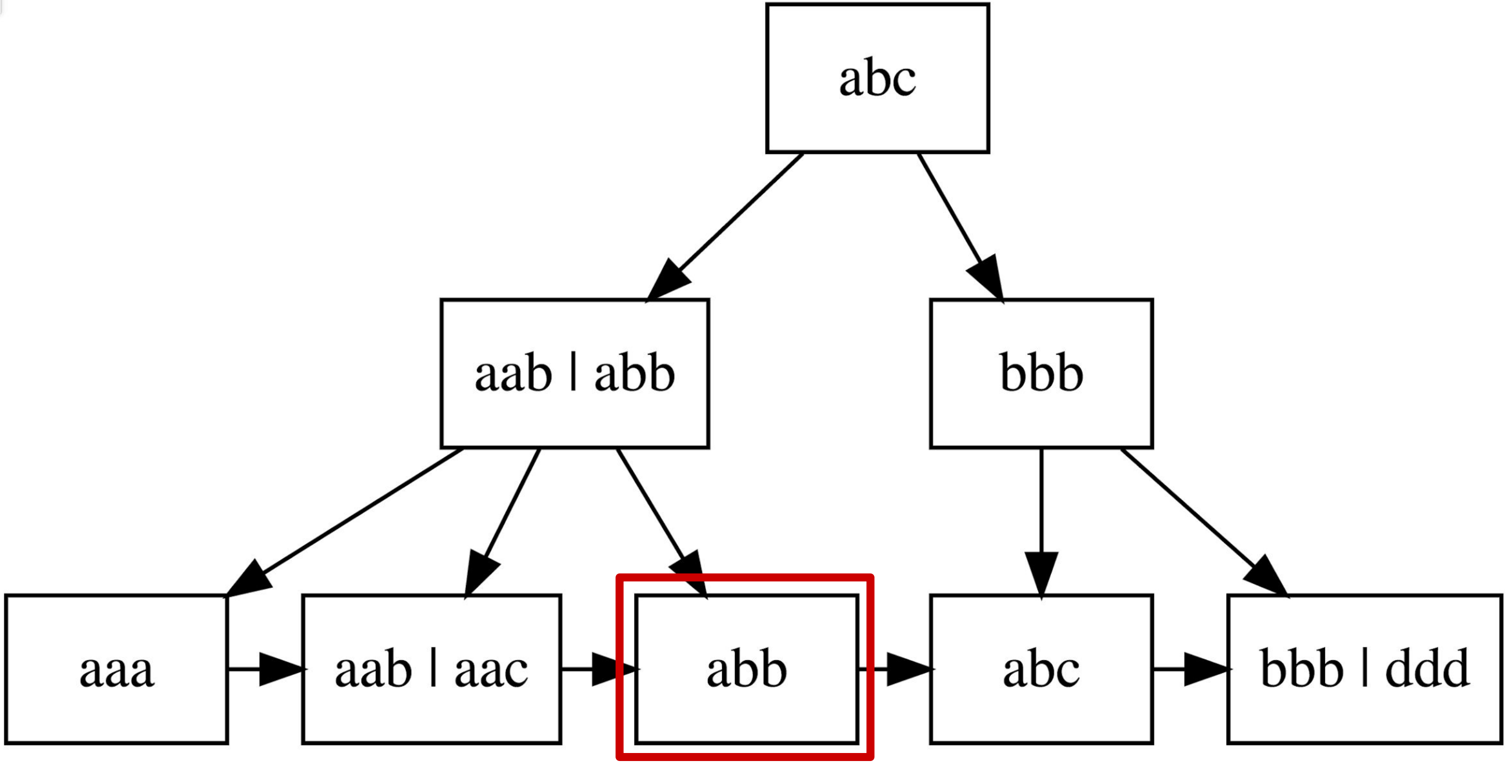
g



# Range queries in B+ trees - **aac → abc**

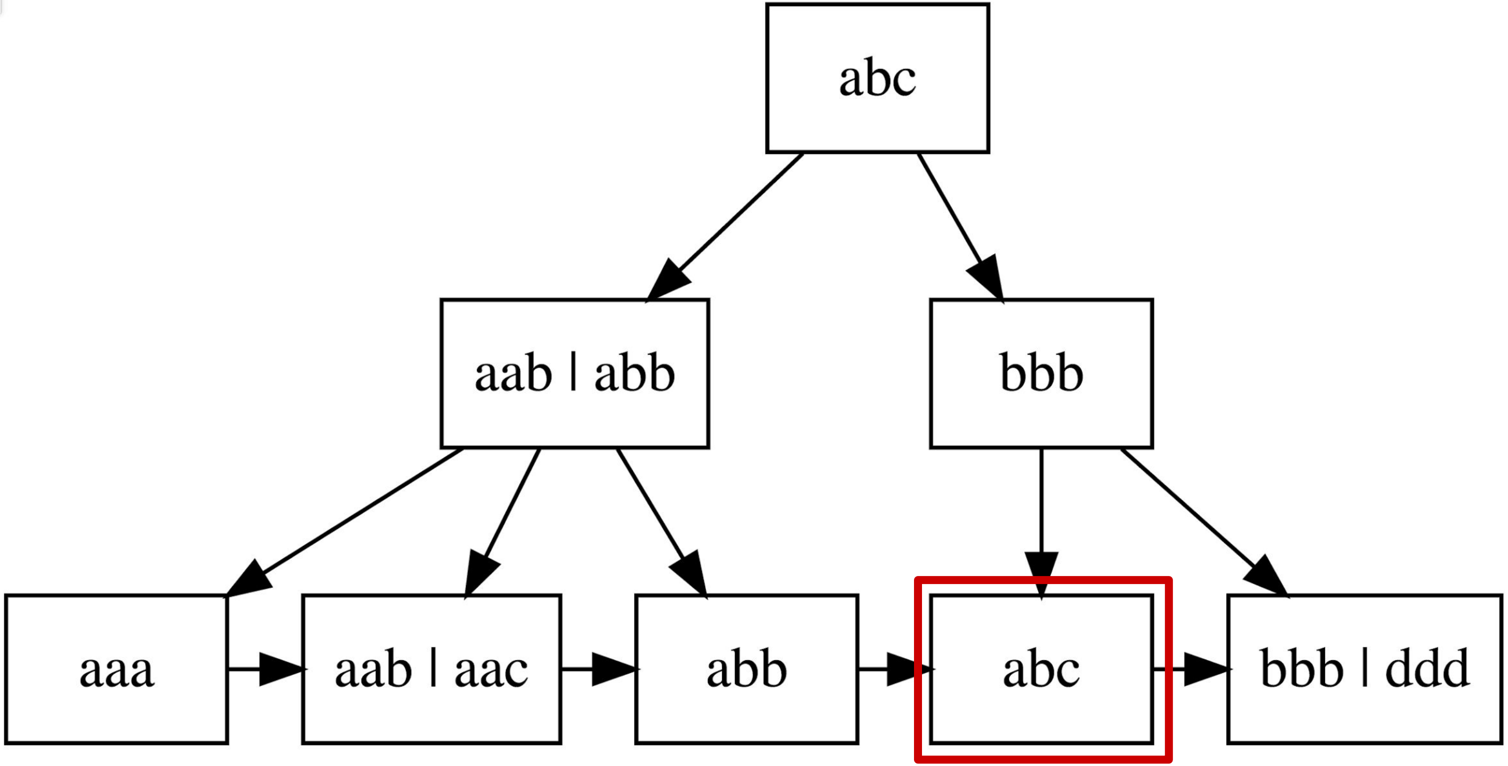


# Range queries in B+ trees - **aac** → **abc**



# Range queries in B+ trees - **aac** → **abc**

g



# Stats

- Statistics about the data
- How many triples with foaf:name?
- How many with foaf:mbox?
- How many instances of a class?

# How does it look

```
(prefix ((: <http://example/))
  (stats
    (meta
     … metadata here …
    )
    (foaf:name 2)
    (foaf:mbox 2)
  ))
```

Two triples with this predicate

**Predicate:** foaf:mbox

# Our query

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE  {
  ?x foaf:name ?name .
  ?x foaf:mbox ?mbox .
}
```

# Stats based rewrite

```
(prefix ((: <http://example/))
  (stats
    (meta
     … metadata here …
    )
    (foaf:name 999999)
    (foaf:mbox 2)
  ))
```

# Join on ?x

| P | S | O |
|---|---|---|
| foaf:mbox | ex:Håvard | “haavard...” |
| foaf:mbox | ex:Veronika | “veronika...” |
| foaf:name | ex:Håvard | “Håvard” |
| foaf:name | ex:Veronika | “Veronika” |

| P | S | O |
|---|---|---|
| foaf:mbox | ex:Håvard | “haavard...” |
| foaf:mbox | ex:Veronika | “veronika...” |
| foaf:name | ex:Håvard | “Håvard” |
| foaf:name | ex:Veronika | “Veronika” |

| ?x | ?name | ?mbox |
|---|---|---|
| | | |
| | | |

# Join on ?x

| P | S | O |
|---|---|---|
| foaf:mbox | ex:Håvard | “haavard...” |
| foaf:mbox | ex:Veronika | “veronika...” |
| foaf:name | ex:Håvard | “Håvard” |
| foaf:name | ex:Veronika | “Veronika” |

| P | S | O |
|---|---|---|
| foaf:mbox | ex:Håvard | “haavard...” |
| foaf:mbox | ex:Veronika | “veronika...” |
| foaf:name | ex:Håvard | “Håvard” |
| foaf:name | ex:Veronika | “Veronika” |

| ?x | ?name | ?mbox |
|---|---|---|
| ex:Håvard | “Håvard” | |
| | | |

# Join on ?x

| P | S | O |
|---|---|---|
| foaf:mbox | ex:Håvard | “haavard...” |
| foaf:mbox | ex:Veronika | “veronika...” |
| foaf:name | ex:Håvard | “Håvard” |
| foaf:name | ex:Veronika | “Veronika” |

| P | S | O |
|---|---|---|
| foaf:mbox | ex:Håvard | “haavard...” |
| foaf:mbox | ex:Veronika | “veronika...” |
| foaf:name | ex:Håvard | “Håvard” |
| foaf:name | ex:Veronika | “Veronika” |

| ?x | ?name | ?mbox |
|---|---|---|
| ex:Håvard | “Håvard” | “haavard...” |
| | | |

# Join on ?x

| P | S | O |
|---|---|---|
| foaf:mbox | ex:Håvard | “haavard...” |
| foaf:mbox | ex:Veronika | “veronika...” |
| foaf:name | ex:Håvard | “Håvard” |
| foaf:name | ex:Veronika | “Veronika” |

| P | S | O |
|---|---|---|
| foaf:mbox | ex:Håvard | “haavard...” |
| foaf:mbox | ex:Veronika | “veronika...” |
| foaf:name | ex:Håvard | “Håvard” |
| foaf:name | ex:Veronika | “Veronika” |

| ?x | ?name | ?mbox |
|---|---|---|
| ex:Håvard | “Håvard” | “haavard...” |
| | | |

# Join on ?x

| P | S | O |
|---|---|---|
| foaf:mbox | ex:Håvard | “haavard...” |
| foaf:mbox | ex:Veronika | “veronika...” |
| foaf:name | ex:Håvard | “Håvard” |
| foaf:name | ex:Veronika | “Veronika” |

| P | S | O |
|---|---|---|
| foaf:mbox | ex:Håvard | “haavard...” |
| foaf:mbox | ex:Veronika | “veronika...” |
| foaf:name | ex:Håvard | “Håvard” |
| foaf:name | ex:Veronika | “Veronika” |

| ?x | ?name | ?mbox |
|---|---|---|
| ex:Håvard | “Håvard” | “haavard...” |
| ex:Veronika | “Veronika” | |

# Join on ?x

| P | S | O |
|---|---|---|
| foaf:mbox | ex:Håvard | “haavard...” |
| foaf:mbox | ex:Veronika | “veronika...” |
| foaf:name | ex:Håvard | “Håvard” |
| foaf:name | ex:Veronika | “Veronika” |

| P | S | O |
|---|---|---|
| foaf:mbox | ex:Håvard | “haavard...” |
| foaf:mbox | ex:Veronika | “veronika...” |
| foaf:name | ex:Håvard | “Håvard” |
| foaf:name | ex:Veronika | “Veronika” |

| ?x | ?name | ?mbox |
|---|---|---|
| ex:Håvard | “Håvard” | “haavard...” |
| ex:Veronika | “Veronika” | “veronika...” |

# Join on ?x

| P | S | O |
|---|---|---|
| foaf:mbox | ex:Håvard | “haavard...” |
| foaf:mbox | ex:Veronika | “veronika...” |
| foaf:name | ex:Håvard | “Håvard” |
| foaf:name | ex:Veronika | “Veronika” |

| P | S | O |
|---|---|---|
| foaf:mbox | ex:Håvard | “haavard...” |
| foaf:mbox | ex:Veronika | “veronika...” |
| foaf:name | ex:Håvard | “Håvard” |
| foaf:name | ex:Veronika | “Veronika” |

| ?x | ?name | ?mbox |
|---|---|---|
| ex:Håvard | “Håvard” | “haavard...” |
| ex:Veronika | “Veronika” | “veronika...” |

# Join on ?x

| P | S | O |
|---|---|---|
| foaf:mbox | ex:Håvard | “haavard...” |
| foaf:mbox | ex:Veronika | “veronika...” |
| foaf:name | ex:Håvard | “Håvard” |
| foaf:name | ex:Veronika | “Veronika” |

| P | S | O |
|---|---|---|
| foaf:mbox | ex:Håvard | “haavard...” |
| foaf:mbox | ex:Veronika | “veronika...” |
| foaf:name | ex:Håvard | “Håvard” |
| foaf:name | ex:Veronika | “Veronika” |

| ?x | ?name | ?mbox |
|---|---|---|
| ex:Håvard | “Håvard” | “haavard...” |
| ex:Veronika | “Veronika” | “veronika...” |

# Stats based rewrite

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE  {
    ?x foaf:name ?name .
    ?x foaf:mbox ?mbox .
}
```

# Stats based rewrite

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE  {
    ?x foaf:mbox ?mbox . # moved up
    ?x foaf:name ?name .
}
```

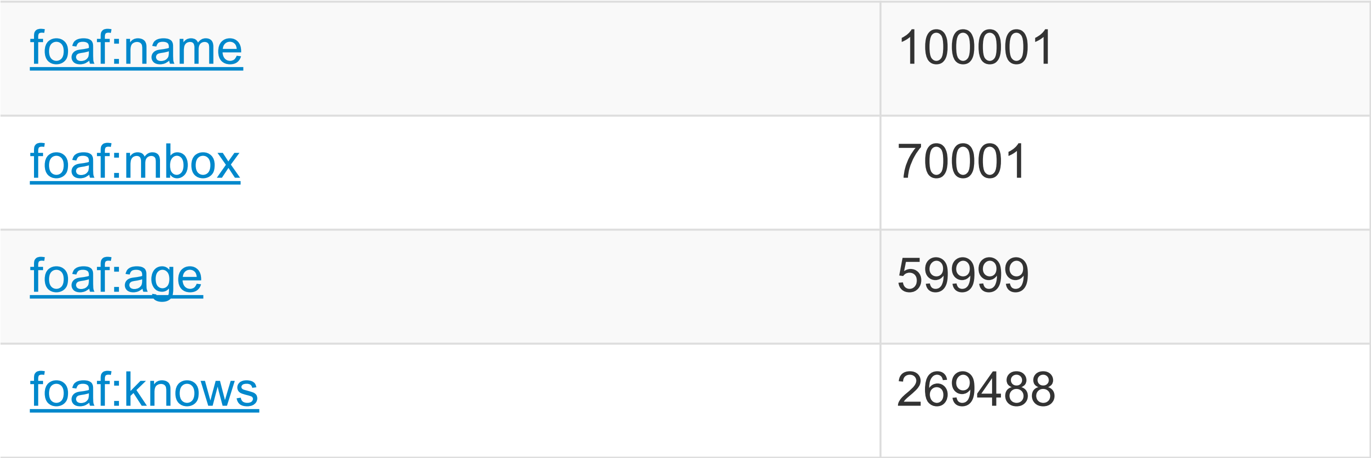
# Stats based rewrite

- Reorder BGP
- Find the most selective patterns
- Move the most selective to top of query
- Join on those first
- Reduces number of triples joined
- Downside:
  - Stats could be wrong
  - Stats need to be maintained

# Jena and stats based rewrite

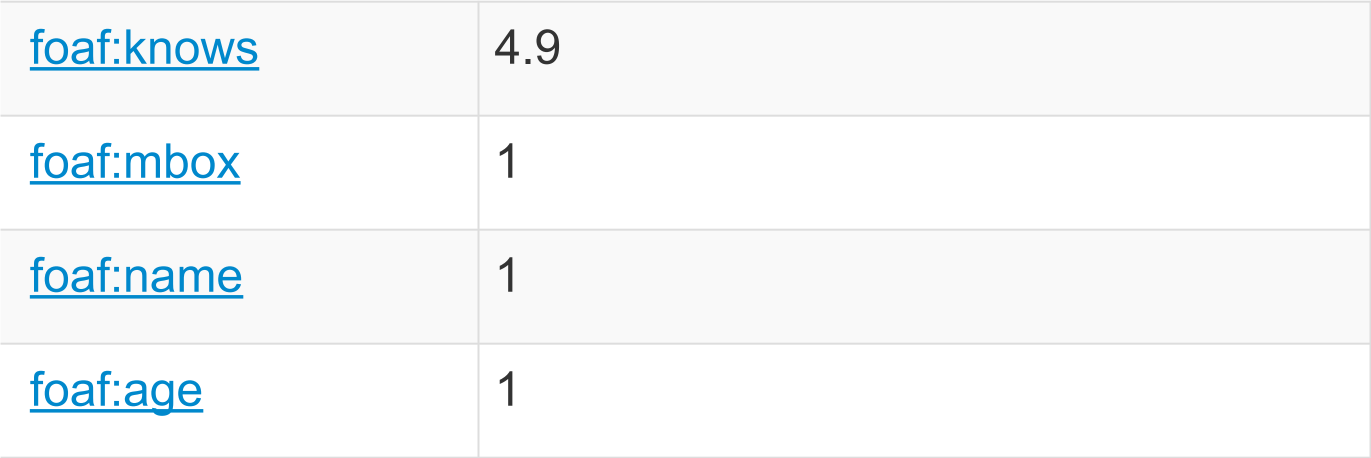
- Jena TDB
  - Simple stats and rewrite
  - Manual maintenance of stats
- Jena in-memory
  - No stats
  - Manual rewrites are useful
  - Jena in-memory databases are used frequently
    - Even in production
  - Our query from before
    - foaf:name, then foaf:mbox: 8310 ms
    - foaf:mbox, then foaf:name: 7 ms

# Query plan



| | |
|---|---|
| foaf:name | 100001 |
| foaf:mbox | 70001 |
| foaf:age | 59999 |
| foaf:knows | 269488 |

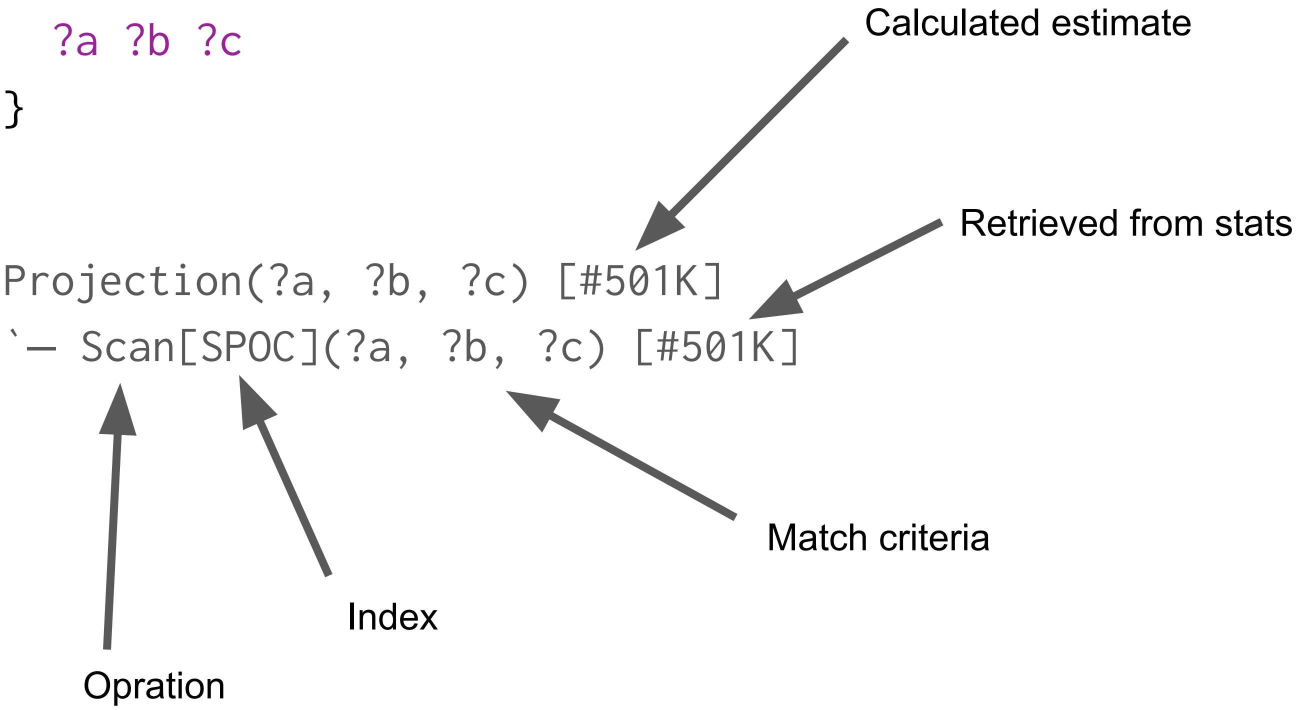
# Query plan



| | |
|---|---|
| foaf:knows | 4.9 |
| foaf:mbox | 1 |
| foaf:name | 1 |
| foaf:age | 1 |

# Query plan

```
SELECT * WHERE {
  ?a ?b ?c
}
```



```
Projection(?a, ?b, ?c) [#501K]
`— Scan[SPOC](?a, ?b, ?c) [#501K]
```

Calculated estimate

Retrieved from stats

Match criteria

Index

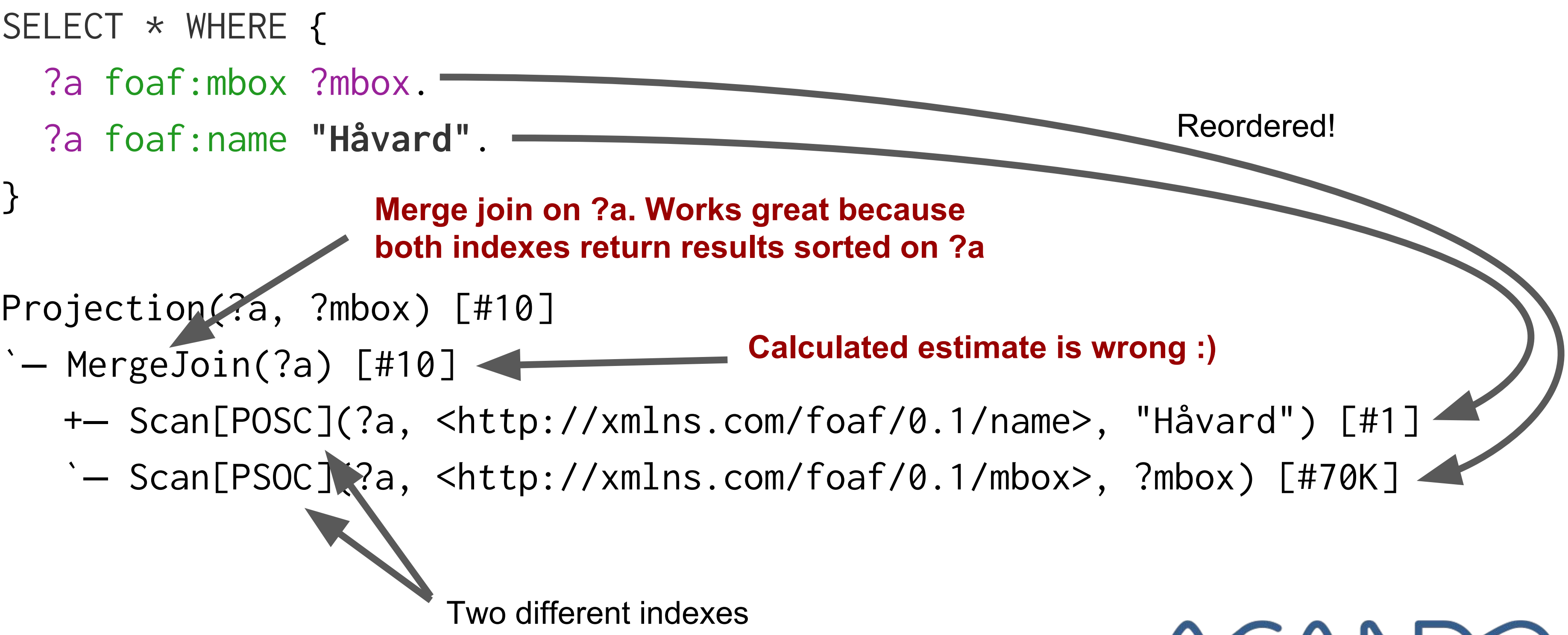
Opration

# Query plan

```
SELECT * WHERE {
  ?a ?b ?c
}
```

```
Projection(?a, ?b, ?c) [#501K]
`- Scan[SPOC](?a, ?b, ?c) [#501K]
```

# Query plan



```
SELECT * WHERE {
   ?a foaf:mbox ?mbox.
   ?a foaf:name "Håvard".
}
```

Reordered!

**Merge join on ?a. Works great because both indexes return results sorted on ?a**

```
Projection(?a, ?mbox) [#10]
`— MergeJoin(?a) [#10]
    +— Scan[POSC](?a, <http://xmlns.com/foaf/0.1/name>, "Håvard") [#1]
    `— Scan[PSOC](?a, <http://xmlns.com/foaf/0.1/mbox>, ?mbox) [#70K]
```

**Calculated estimate is wrong :)**

Two different indexes

# Laziness

- All operations are lazy (if possible)
- Keep as little data in memory as possible
- SCAN operators return iterators
  - Call .next() to get next element
- MergeJoin also returns iterator
  - Calling .next() will return the next tuple
  - .next() may need to call .next() on the inner SCANs
    - Maybe multiple times
- Other
  - DirectHashJoin
  - Filter
  - Union

```
SELECT * WHERE {
  ?a foaf:mbox ?mbox.
  ?a foaf:name "Håvard".
  ?a foaf:knows ?b.
  ?b foaf:name ?b_name.
}
```

```
Projection(?a, ?mbox, ?b, ?b_name) [#25]
`— MergeJoin(?b) [#25]
   +— Sort(?b) [#25]
   |  `— NaryJoin(?a) [#25]
   |     +— Scan[POSC](?a, <http://xmlns.com/foaf/0.1/name>, "Håvard") [#1]
   |     +— Scan[PSOC](?a, <http://xmlns.com/foaf/0.1/mbox>, ?mbox) [#70K]
   |     `— Scan[PSOC](?a, <http://xmlns.com/foaf/0.1/knows>, ?b) [#271K]
   `— Scan[PSOC](?b, <http://xmlns.com/foaf/0.1/name>, ?b_name) [#100K]
```

**Need to sort before merge join.**

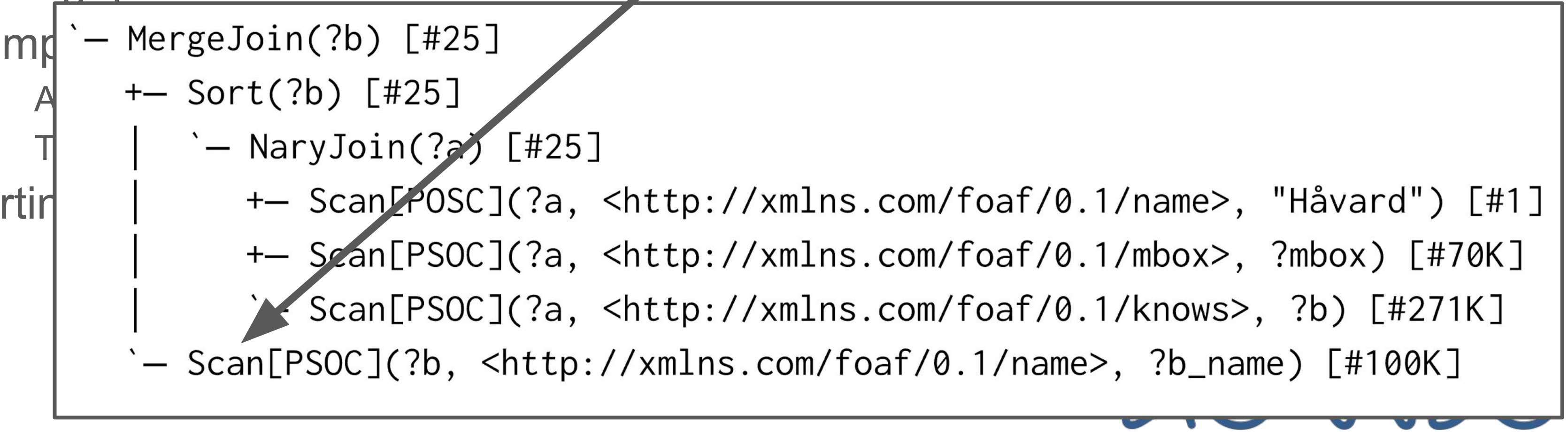
**Natural join on ?a. Could use multiple join algorithms including hash join and parallel join.**

# Accumulators

- Not all operations be be streamed
- Some need to keep all results in memory
- Eg.
    - Sorting
    - Group by
    - HashJoin
        - Because it needs to build a hash table
- Sometimes called pipeline breakers

# IO bound

- Reading from a spinning disk is very slow
- Moving parts
  - Rotate disk
  - Swing arm
- Seek time is: 4 ms
- Throughput: 250 MB/s
- Comp
  - A
  - T
- Sortir



```
`— MergeJoin(?b) [#25]
   +— Sort(?b) [#25]
   |   `— NaryJoin(?a) [#25]
   |       +— Scan[POSC](?a, <http://xmlns.com/foaf/0.1/name>, "Håvard") [#1]
   |       +— Scan[PSOC](?a, <http://xmlns.com/foaf/0.1/mbox>, ?mbox) [#70K]
   |       `— Scan[PSOC](?a, <http://xmlns.com/foaf/0.1/knows>, ?b) [#271K]
   `— Scan[PSOC](?b, <http://xmlns.com/foaf/0.1/name>, ?b_name) [#100K]
```

**Could be 25 random reads instead!**

# Cross product (cartesian)

- If you data has 3 people
- Cross product is 3 x 3 = 9
- Grows very quickly
- 1 000 000 things is easy to keep in memory
  - As java integers: 4 MB
  - Cross product 1 000 000 000 000
    - As Java integers: 4 TB

# Nested optionals

- Start with a person
- Find any other people they might know
  - But maybe they don’t know anyone
- Find out if those people know anyone else
  - Maybe they also don’t know anyone

# Nested optionals

```
SELECT * WHERE {
   <http://example.org/18948> foaf:mbox ?mbox;

   OPTIONAL {
     <http://example.org/18948> foaf:knows   ?knows.
     ?knows foaf:name ?knows_name.

     OPTIONAL{
       ?knows foaf:knows   ?knows_knows.
       ?knows_knows foaf:name ?knows_knows_name.
     }
   }
}
```

# Nested optionals

And keep them all in memory

Essentially cross product

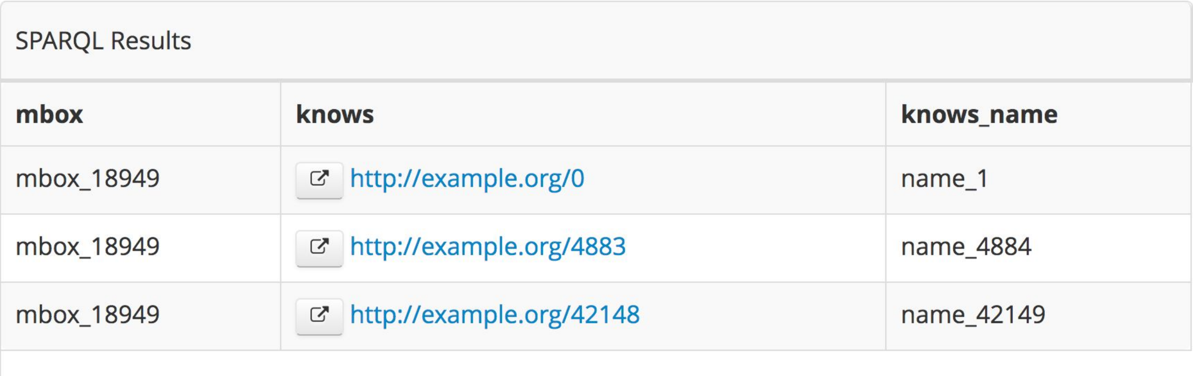
```
Projection(?mbox, ?knows, ?knows_name, ?knows_knows, ?knows_knows_name) [#3]
`— LoopJoinOuter(_) [#3]
   +— Scan[SPOC](<http://example.org/18948>, <http://xmlns.com/foaf/0.1/mbox>, ?mbox) [#1]
   `— MergeJoinOuter(?knows) [#3]
      +— MergeJoin(?knows) [#3]
      |  +— Scan[SPOC](<http://example.org/18948>, <http://xmlns.com/foaf/0.1/knows>, ?knows) [#3]
      |  `— Scan[PSOC](?knows, <http://xmlns.com/foaf/0.1/name>, ?knows_name) [#100K]
      `— Sort(?knows) [#271K]
         `— MergeJoin(?knows_knows) [#271K]
            +— Scan[POSC](?knows, <http://xmlns.com/foaf/0.1/knows>, ?knows_knows) [#271K]
            `— Scan[PSOC](?knows_knows, <http://xmlns.com/foaf/0.1/name>, ?knows_knows_name) [#100K]
```

# Nested optionals

- Difficult to implement efficiently
- Some databases can handle this
- Others can´t
- Two approaches to optimize by hand
    - Multiple queries
    - Union queries

# Nested optionals - multiple queries

```
SELECT * WHERE {
  <http://example.org/18948> foaf:mbox ?mbox;
  OPTIONAL {
    <http://example.org/18948> foaf:knows   ?knows.
    ?knows foaf:name ?knows_name.
  }
}
```



SPARQL Results

| mbox | knows | knows_name |
|---|---|---|
| mbox_18949 | http://example.org/0 | name_1 |
| mbox_18949 | http://example.org/4883 | name_4884 |
| mbox_18949 | http://example.org/42148 | name_42149 |

# Nested optionals - multiple queries

```
SELECT * WHERE {

  VALUES (?knows ){
      (<http://example.org/0>)
      (<http://example.org/4883>)
      (<http://example.org/42148>)
    }

  ?knows foaf:knows   ?knows_knows.
  ?knows_knows foaf:name ?knows_knows_name.
}
```

# Nested optionals - Union

- Joins are slow because of potentially unbound variables
- Force all variables to be bound
- Duplicate up query until all possible optional patterns are hardcoded

# Nested optionals - union

```
SELECT * WHERE {
  {
    <http://example.org/18948> foaf:mbox ?mbox. # knows no one
  } UNION {
    <http://example.org/18948> foaf:mbox ?mbox. # knows someone but they don’t know anyone
    <http://example.org/18948> foaf:knows ?knows.
    ?knows foaf:name ?knows_name.
  } UNION {
    <http://example.org/18948> foaf:mbox ?mbox. # knows someone who knows someone else
    <http://example.org/18948> foaf:knows   ?knows.
    ?knows foaf:name ?knows_name.
    ?knows foaf:knows ?knows_knows.
    ?knows_knows foaf:name ?knows_knows_name.
  }
}
```

```
Projection(?mbox, ?knows, ?knows_name, ?knows_knows, ?knows_knows_name) [#7]
`— Union [#7]
   +— Scan[SPOC](<http://example.org/18948>, <http://xmlns.com/foaf/0.1/mbox>, ?mbox) [#1]
   `— MergeJoin(?knows) [#6]
      +— Sort(?knows) [#6]
      |  `— Union [#6]
      |     +— LoopJoin(_) [#3]
      |     |  +— Scan[SPOC](<http://example.org/18948>, <http://xmlns.com/foaf/0.1/mbox>, ?mbox) [#1
      |     |  `— Scan[SPOC](<http://example.org/18948>, <http://xmlns.com/foaf/0.1/knows>, ?knows) [
      |     `— LoopJoin(_) [#3]
      |        +— Scan[SPOC](<http://example.org/18948>, <http://xmlns.com/foaf/0.1/mbox>, ?mbox) [#1]
      |        `— MergeJoin(?knows_knows) [#3]
      |           +— Sort(?knows_knows) [#3]
      |           |  `— MergeJoin(?knows) [#3]
      |           |     +— Scan[SPOC](<http://example.org/18948>, <http://xmlns.com/foaf/0.1/knows>,
      |           |     `— Scan[PSOC](?knows, <http://xmlns.com/foaf/0.1/knows>, ?knows_knows) [#271K
      |           `— Scan[PSOC](?knows_knows, <http://xmlns.com/foaf/0.1/name>, ?knows_knows_name) [#
      `— Scan[PSOC](?knows, <http://xmlns.com/foaf/0.1/name>, ?knows_name) [#100K]
```

# Summary

- Basic Graph Pattern
- Indexes
- Statistics
  - Optimisation
- Query plan
  - Selection, join, sort
- Nested optionals

# Thank you