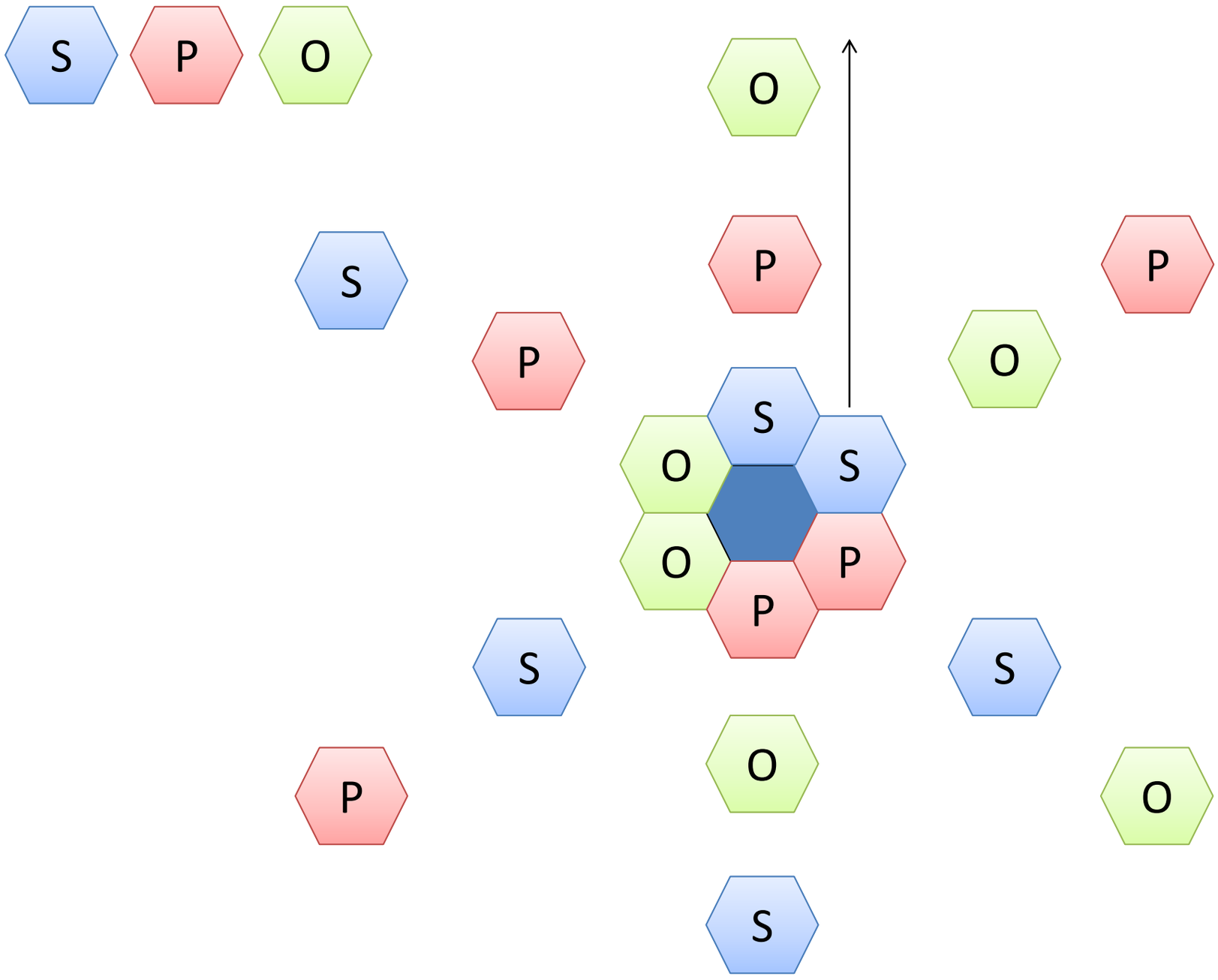
# REMINDER: IMPORTANT NATIVE-INDEXING DATASTORES

# Native RDF indexing - Hexastore

- Hexastore [Weiss et al. 2008]
  - Κρατάμε όλα τα sorted permutations των triples
  - SPO, SOP, PSO, POS, OSP, OPS
  - Όλα τα query patterns μπορούν να απαντηθούν με ένα Index scan
  - Όλα τα αρχικά join μπορούν να γίνουν με αποδοτικά merge-joins
  - Πιο ακριβά hash ή sort-merge joins χρειάζονται μόνο όταν κάνουμε join non-ordered intermediate results

# Hexastore: Sextuple Indexing



# Five-fold Increase in Index Space

- Sharing The Same Terminal Lists
  - SPO-PSO, SOP-OSP, POS-OPS

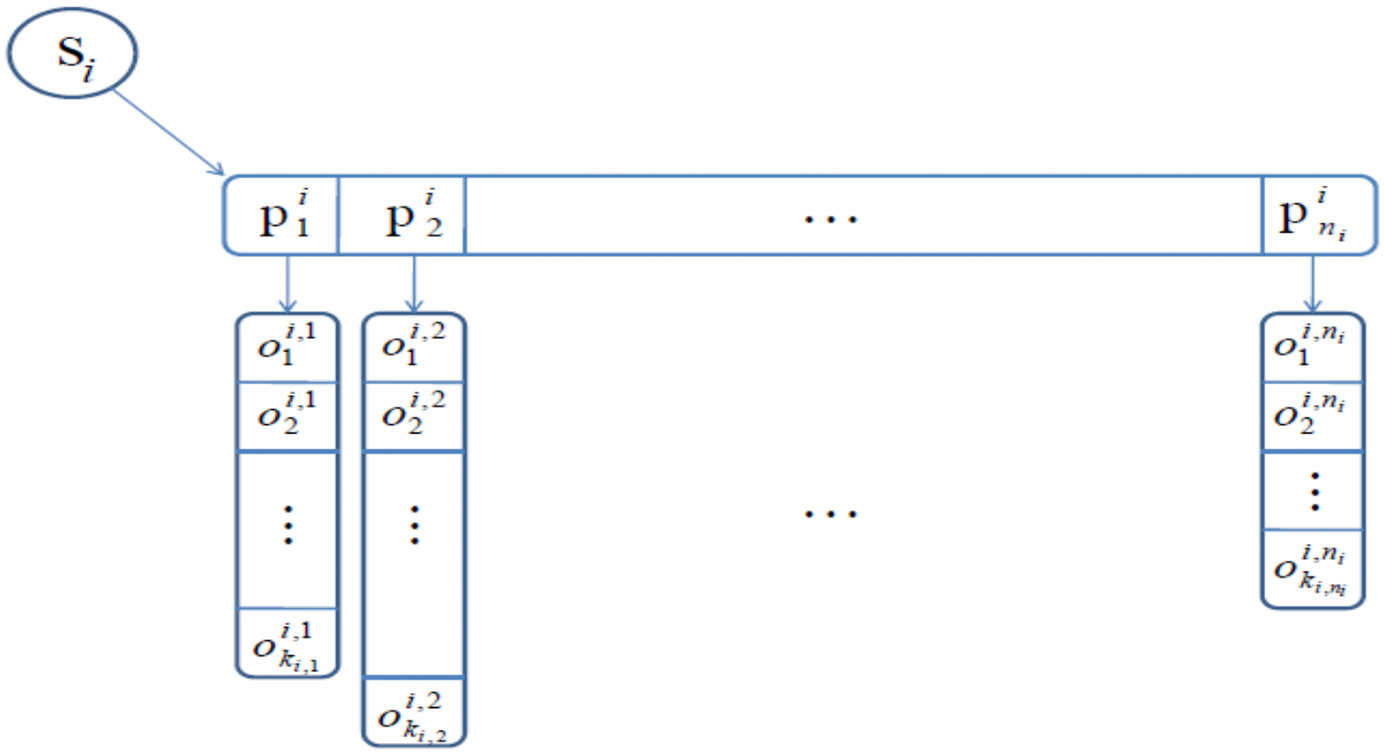



**Figure 2:** spo indexing in a Hexastore

  - The key of each of the three resources in a triple appears in two headers and two vectors, but only in one list

# Mapping Dictionary

- Replacing all literals by unique IDs using a mapping dictionary

| S | P | O |
|---|---|---|
| object214 | hasColor | blue |
| object214 | belongsTo | object352 |
| … | … | … |

| S | P | O |
|---|---|---|
| 0 | 1 | 2 |
| 0 | 3 | 4 |
| … | … | … |

| ID | Value |
|---|---|
| 0 | object214 |
| 1 | hasColor |
| … | … |

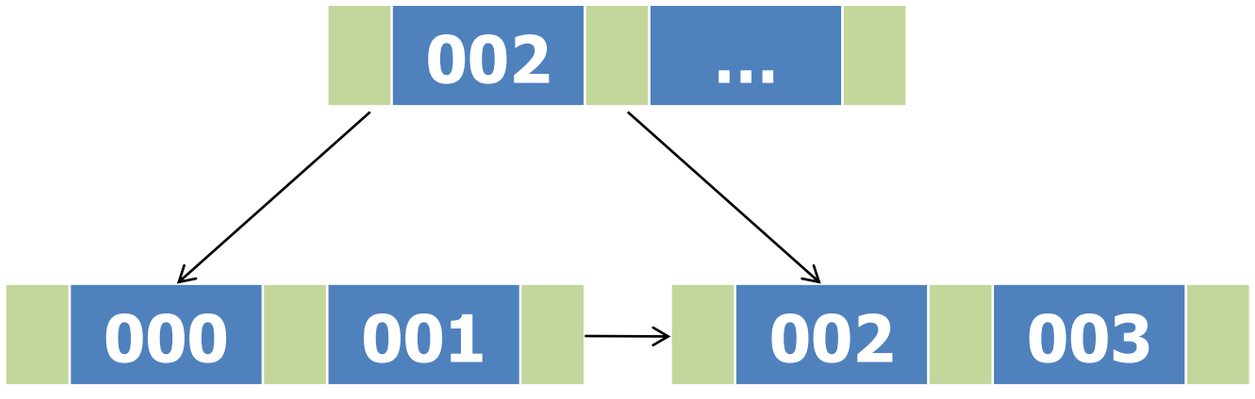
– Mapping dictionary compresses the triple store
  - Reduced redundancy, Saving a lot of physical space
– We can concentrate on a logical index structure rather than the physical storage design

# Clustered B+-Tree (RDF-3X, VLDB 2008)

| S | P | O |
|---|---|---|
| 0 | 1 | 2 |
| 0 | 3 | 4 |
| … | … | … |

Actually, we don't need this table!

- Store everything in a clustered B+-Tree
  - Triples are sorted in lexicographical order
    - Allowing the conversion of SPARQL patterns into range scan
  - We don't have to do entire table scan




<Mapping Dictionary>

| ID | Value |
|---|---|
| 0 | object214 |
| 1 | hasColor |
| … | … |

# Native RDF indexing – RDF3X

| Query pattern | | | Index |
|---|---|---|---|
| Subject | Predicate | Object | |
| _ | _ | _ | όλα |
| ? | _ | _ | pos, ops |
| _ | ? | _ | osp, sop |
| _ | _ | ? | spo, pso |
| ? | ? | _ | osp, ops |
| _ | ? | ? | spo, sop |
| ? | _ | ? | pos, pso |
| ? | ? | ? | όλα |

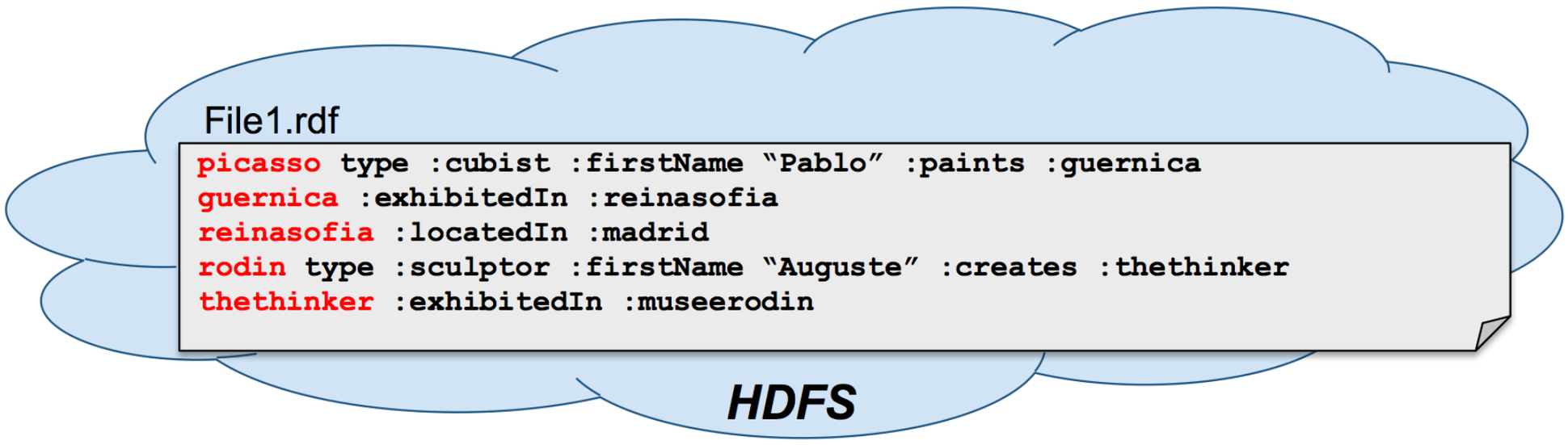
- RDF-3X [Neumann et al. 2008]
  - 6 indexes, aggregated indexes για στατιστικά και εύρεση του βέλτιστου πλάνου εκτέλεσης
  - Aggressive compression

# MapReduce-based engines

- Αποθήκευση RDF δεδομένων σε αρχεία HDFS
- Υλοποίηση join με χρήση MapReduce
- Αντιπροσωπευτικά συστήματα
  - SHARD [Rohloff 2010]
  - HadoopRDF [Husain 2011]
  - PigSPARQL [Schätzle 2012]

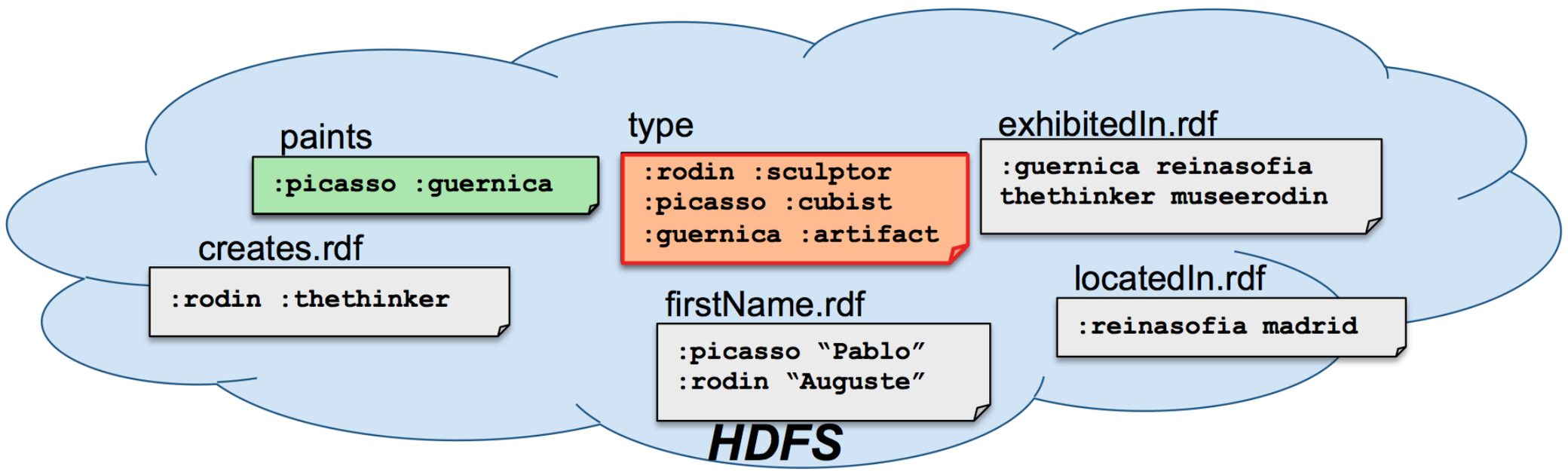
# SHARD

- Αρχεία HDFS:
  - Μια γραμμή περιέχει όλες τις τριάδες που έχουν ένα συγκεκριμένο subject
- Query processing
  - Left deep join plans
  - Ένα MapReduce job για κάθε BGP του ερωτήματος
  - Μικρές δυνατότητες φιλτραρίσματος των RDF δεδομένων



# HadoopRDF

- Αρχεία HDFS:
  - RDF τριάδες ομαδοποιημένες ανά predicate
  - Εσωτερική ομαδοποίηση των αρχείων με βάση το type του κάθε object
- Query processing
  - Επιλογή αρχείων που ταιριάζουν σε κάθε BGP
  - Πολλαπλά join ανά MapReduce job
  - Heuristic join planner



# Pig SPARQL

- Αρχεία HDFS:
  - Ένα ενιαίο αρχείο που περιέχει όλα τα RDF δεδομένα
  - Χωρίς δυνατότητες ευρετηρίασης
- Query processing
  - Μετάφραση SPARQL σε Pig scripts
  - Εκτέλεση των PIG scripts με MapReduce jobs

# NoSQL indexing

- Χρήση key-value store για δημιουργία ευρετηρίων
  - Αριθμός ευρετηρίων 1-6
  - Ανάκτηση δεδομένων για BGP:
    - Με lookups ή range scans
- Επεξεργασία ερωτημάτων
  - Με τοπική επεξεργασία
  - Με χρήση MapReduce
- Αντιπροσωπευτικά συστήματα
  - Stratustore [Stein 2010]
  - Rya [Punnoose 2012]
  - H2RDF [Papailiou 2012]
  - H2RDF+ [Papailiou 2013]
  - MAPSIN [Schätzle 2012]

# NoSQL indexing

**SPO**

| Key | (attribute, value) |
|---|---|
| :picasso,:firstName,"Pablo" | - |
| :picasso,:p… | |
| :picasso,ty… | |
| :guernica,:… | |
| … | |

**POS**

| Key | (attribute, value) |
|---|---|
| :exhibitedIn,:reinasofia,:guernica | - |
| :firstName,"Pablo",:picasso, | - |
| :paints,:gu… | |
| type,:cubis… | |
| … | |

**OSP**

| Key | (attribute, value) |
|---|---|
| :cubist,:picasso,type, | - |
| :guernica,:picasso,:paints | - |
| "Pablo",:picasso,:firstName | - |
| :reinasofia,:guernica,:exhibitedIn | - |
| … | - |

# NoSQL indexing

- Stratustore (Amazon SimpleDB)
  - S|P|O — 1 hash index
- Rya (Apache Accumulo)
  - SPO|–, POS|–, OSP|– — 3 sorted indexes
- H2RDF (Apache HBase)
  - SPO|–, POS|–, OSP|– — 3 sorted indexes
- H2RDF+ (Apache HBase)
  - SPO|–, SOP|–, POS|–, PSO|–, OPS|–, OSP|– — 6 sorted indexes
- MAPSIN (Apache HBase)
  - SPO|, OPS|– — 2 sorted indexes

# Rya

- Αποθήκευση δεδομένων
  - Apache Accumulo
  - 3 sorted indexes
  - Ανάκτηση δεδομένων με lookup για όλα τα BGP patterns
- Επεξεργασία ερωτημάτων
  - Index nested loops
  - Αναζήτηση του accumulo index για κάθε κλειδί του join
  - Αποδοτικό για μικρού μεγέθους join
  - Απαιτεί πολλά index lookups για non selective ερωτήματα

# H2RDF

- Αποθήκευση δεδομένων
  - Apache HBase
  - 3 sorted indexes
  - Ανάκτηση δεδομένων με lookup για όλα τα BGP patterns
- Επεξεργασία ερωτημάτων
  - Partial Input Hash joins
  - Διαλέγει τον κατάλληλο join αλγόριθμο ανάλογα με το μέγεθος δεδομένων των BGP
    - Αν υπάρχει μικρό pattern μόνο αυτό χρησιμοποιείται ως είσοδος
  - Κατανεμημένη (MapReduce) ή κεντρική εκτέλεση των join

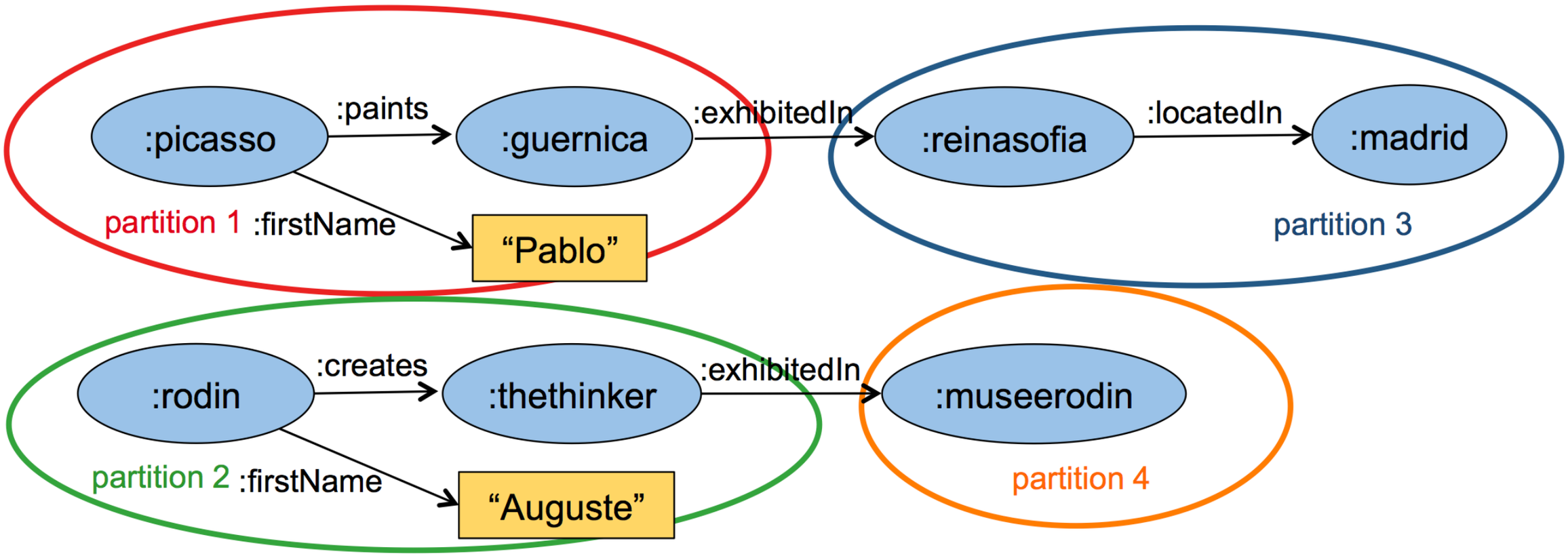
# H2RDF+

- Αποθήκευση δεδομένων
  - Apache HBase
  - 6 sorted indexes και aggregated satatistics
  - Συμπιέση των ευρετηρίων
  - Ανάκτηση ταξινομημένων δεδομένων για όλα τα BGP patterns
- Επεξεργασία ερωτημάτων
  - Αξιοποιεί τα 6 index για την εκτέλεση Merge join
  - Multi-way Merge και Sort-Merge joins
  - Κατανεμημένη (MapReduce) ή κεντρική εκτέλεση των join
  - Μοντέλο κόστους για τα joins
  - Ελαστική επιλογή των resources

# MAPSIN

- Αποθήκευση δεδομένων
  - Apache HBase
  - 2 sorted indexes
  - Ανάκτηση δεδομένων για τα BGP patterns με γνωστό predicate
- Επεξεργασία ερωτημάτων
  - MapReduce map phase join algorithm
  - Lookup operations για κάθε κλειδί του join

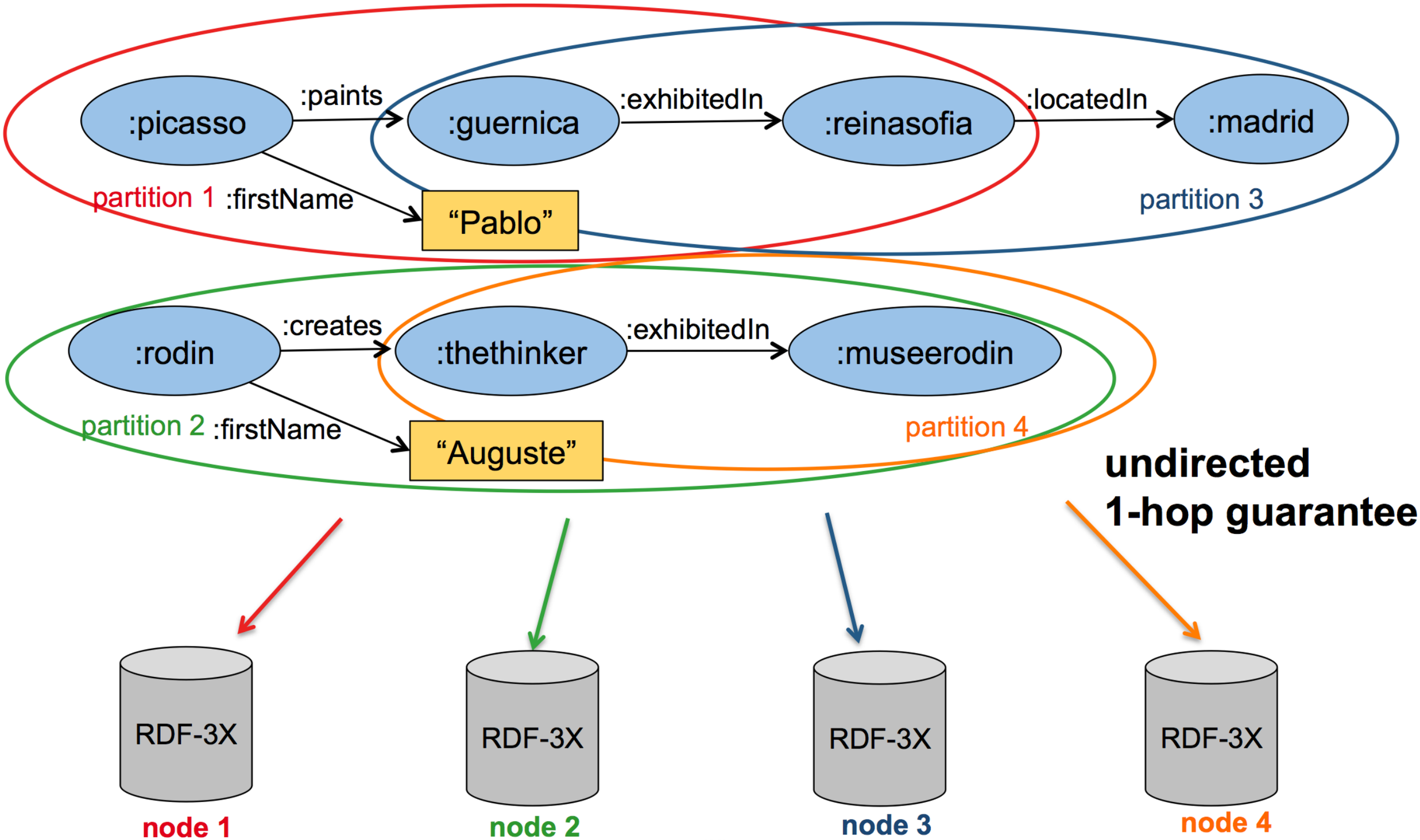
# Graph partitioning




# Graph partitioning

- Graph partitioning [Huang et al. 11]
  - Graph partitioning για διαμειρασμό των RDF δεδομένων
  - Κάθε κόμβος χρησιμοποιεί ένα RDF-3X για τα δεδομένα του δικού του graph partition
  - n-hop replication scheme
    - Εκτός από τα δεδομένα του partition του ο κάθε κόμβος κάνει replicate και δεδομένα που είναι έως n βήματα μακριά
    - Παράλληλη εκτέλεση για ερωτήματα με διάμετρο μικρότερη του n
    - Τα μεγαλύτερα ερωτήματα χωρίζονται σε μικρότερα, διαμέτρου n
    - Τα αποτελέσματα των υποερωτημάτων συνδυάζονται με χρήση MapReduce

# 1-hop guarantee

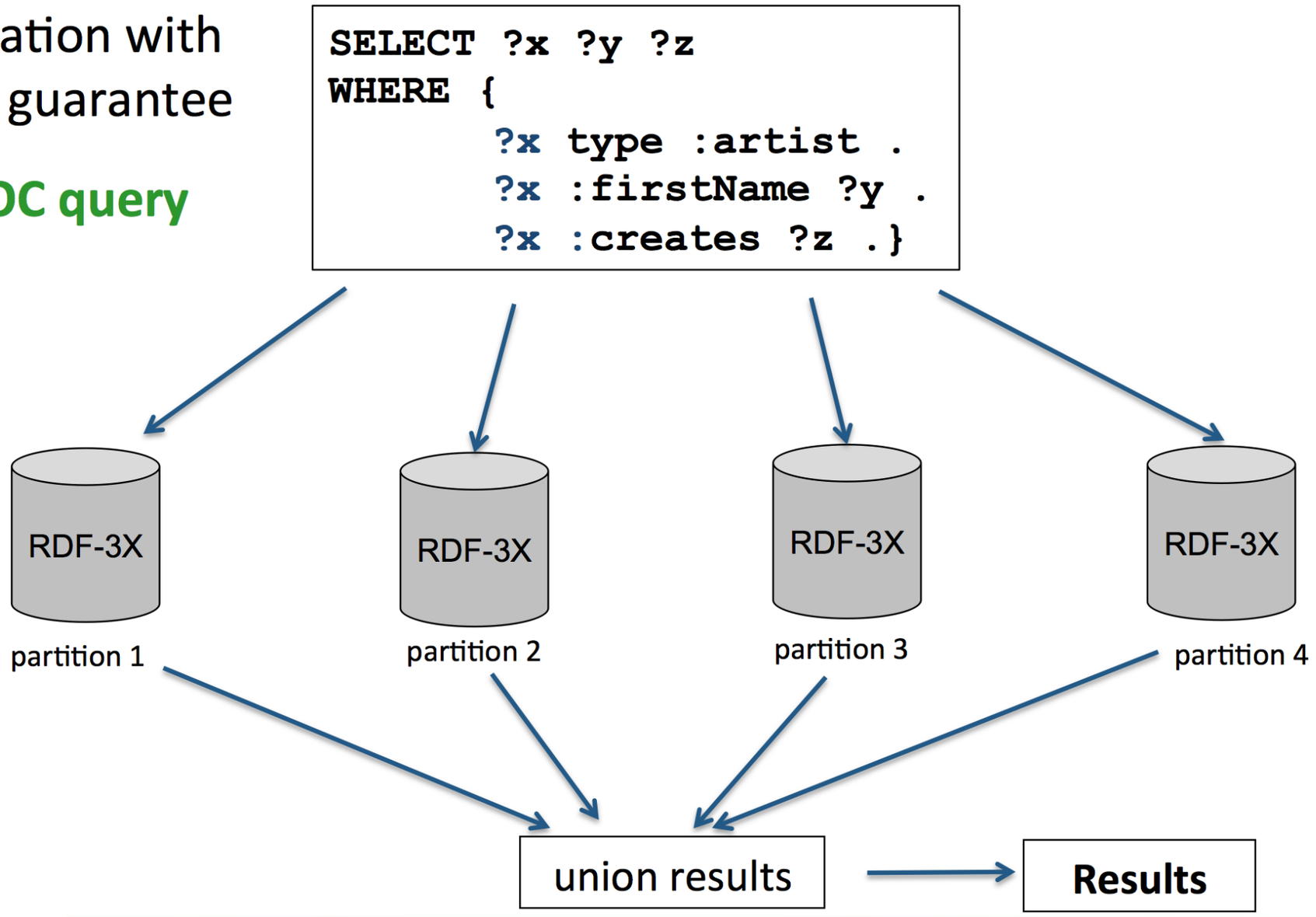


# Graph partitioning

replication with
1-hop guarantee

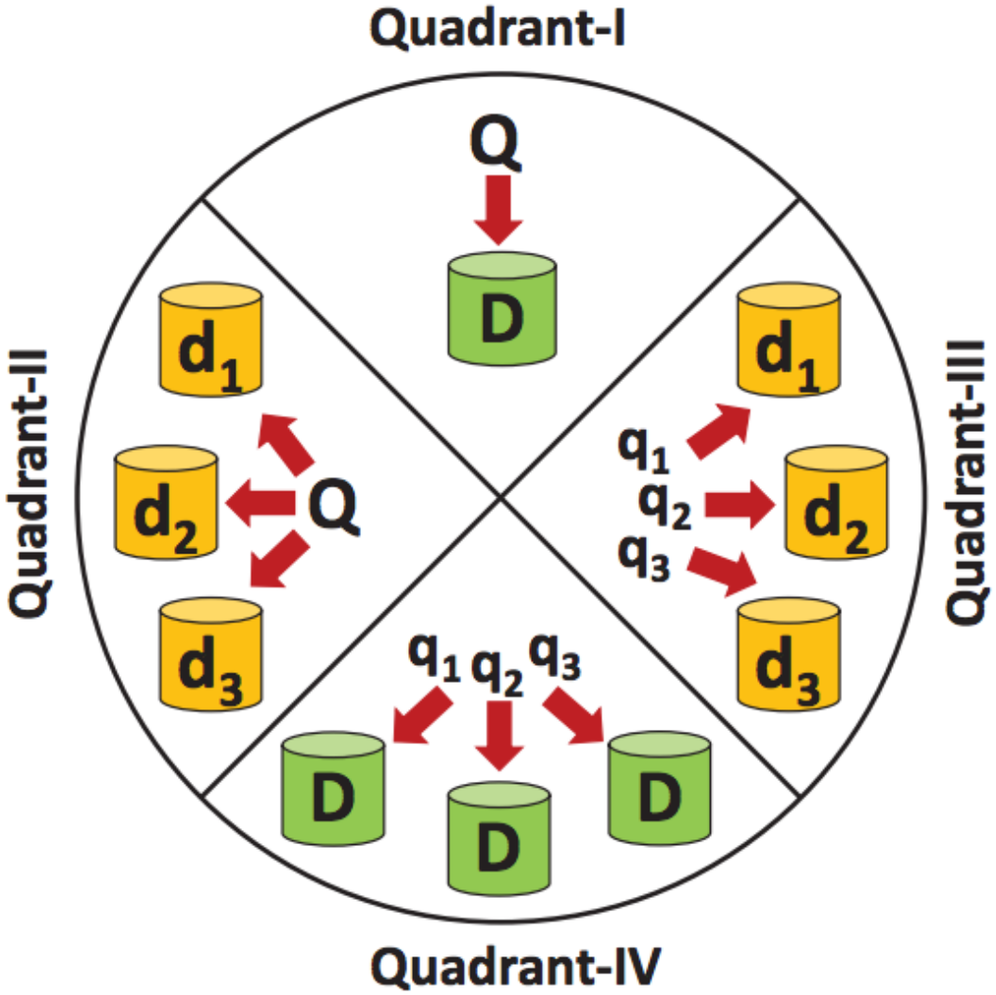
**PWOC query**

```
SELECT ?x ?y ?z
WHERE {
        ?x type :artist .
        ?x :firstName ?y .
        ?x :creates ?z .}
```



# DREAM

- DREAM [Hammoud 2015]
  - Διαχωρισμός συστημάτων ανάλογα με την κατανομή των δεδομένων και της επεξεργασίας
  - Υλοποίηση συστήματος που ανήκει στο Quadrant-IV



# Distributed Main Memory

- Memory cloud: TrinityRDF [Zeng et al. 2013]
  - Αποθηκεύει τα RDF δεδομένα στην κύρια μνήμη όλων των κόμβων του cluster
  - Κάθε κόμβος κρατάει ένα memory hash-map των RDF δεδομένων που του αντιστοιχούν
  - Query execution
    - Graph exploration με μηνύματα μεταξύ των κόμβων
    - Ουσιαστικά κάνει ένα semi-join processing για το query
    - Δεν κάνει πλήρες reduction για ερωτήματα με κύκλους
    - Στο τέλος τα reduced αποτελέσματα μαζεύονται σε ένα κεντρικό server που κάνει το τελικό processing

# Distributed Main Memory

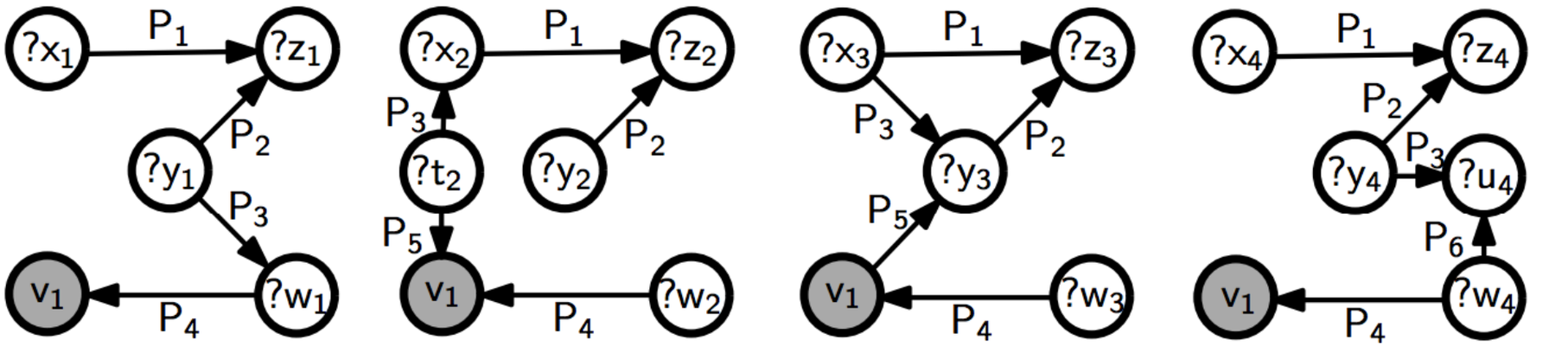
- TriAD [Gurajada 2014]
  - Hash partitioning
  - Αποθηκεύει τα RDF δεδομένα στην κύρια μνήμη όλων των κόμβων
  - 6 indexes + aggregated statistics
  - MPI-based asynchronous join execution
  - Βασισμένο στο RDF-3X για join planning

# Workload-adaptive engines

- Προσαρμογή στο workload
  - Adaptive data partitioning
  - Result caching
  - Multi-query optimization
- Αντιπροσωπευτικά συστήματα
  - SPARQL Multi-query optimization [Le 2012]
  - Partout [Galarraga 2014]
  - Warp [Hose 2014]
  - Chameleon-db [Aluc 2015]
  - H2RDF+ caching [Papailiou 2015]
  - AdPart [Harbi 2016]

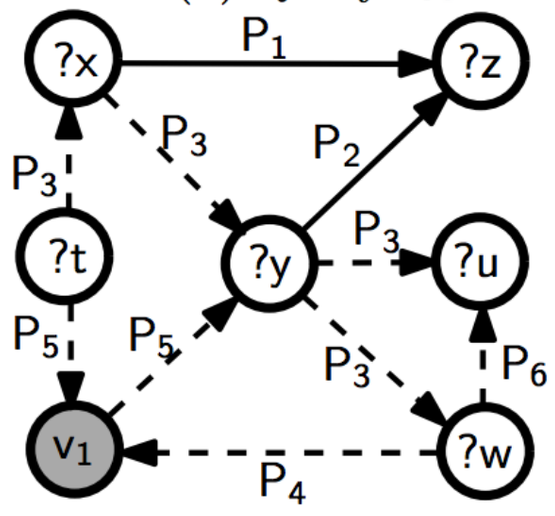
# Multi-query optimization

- Συνένωση SPARQL ερωτημάτων που έχουν overlap
  - Χρησιμοποιεί OPTIONAL patterns




(a) Query $Q_1$ (b) Query $Q_2$ (c) Query $Q_3$ (d) Query $Q_4$

```
SELECT *
WHERE { ?x P1 ?z, ?y P2 ?z,
    OPTIONAL {?y P3 ?w, ?w P4 v1 }
    OPTIONAL {?t P3 ?x, ?t P5 v1, ?w P4 v1 }
    OPTIONAL {?x P3 ?y, v1 P5 ?y, ?w P4 v1 }
    OPTIONAL {?y P3 ?u, ?w P6 ?u, ?w P4 v1 }
}
```



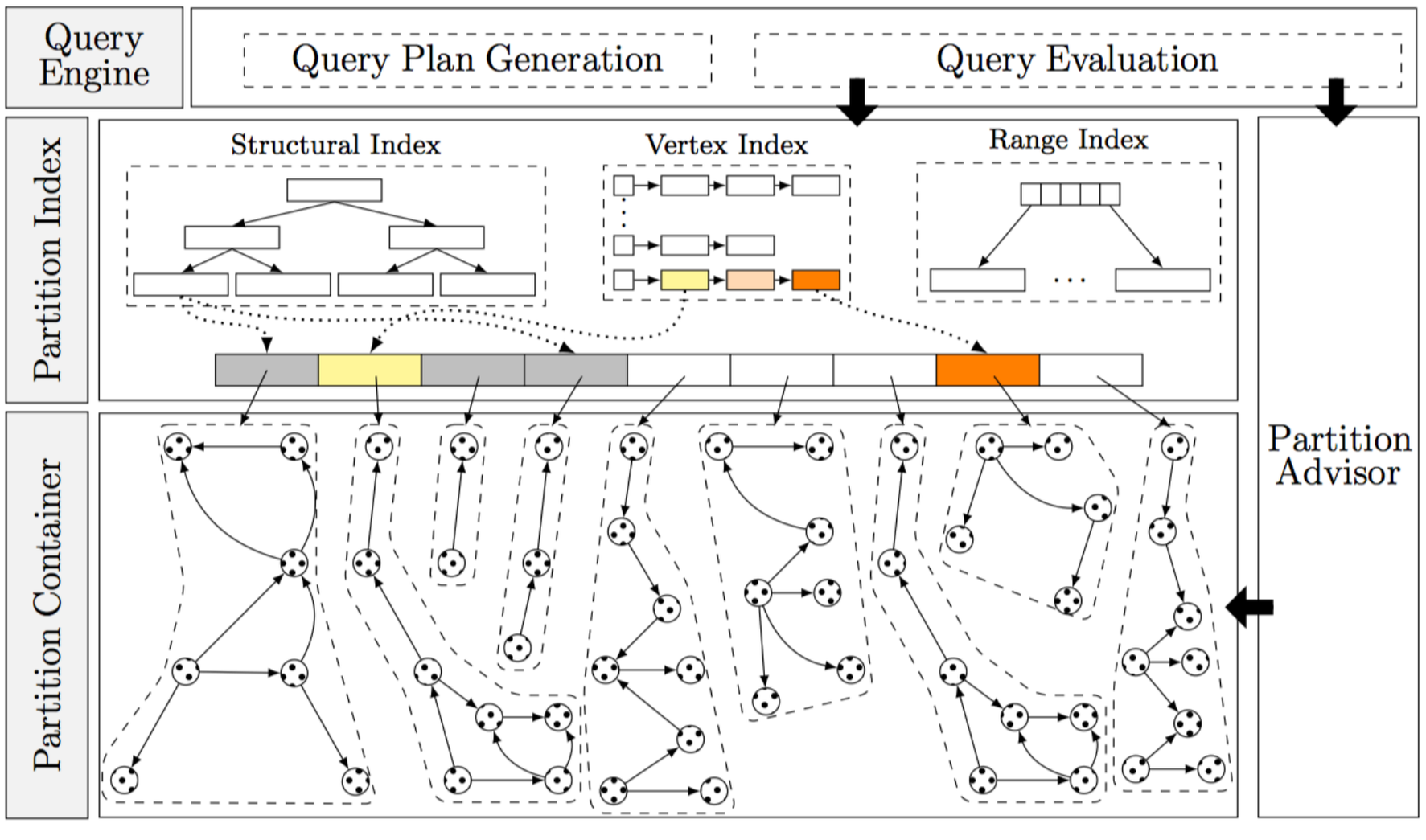

(e) Example query $Q_{OPT}$

# Partout - Warp

- Επεξεργασία του workload των ερωτημάτων
  - Επεκτείνουν graph partitioning τεχνικές
  - Εύρεση ενός καλού graph partitioning με βάση τα ερωτήματα
  - Adaptive replication των RDF τριάδων
  - Ελαχιστοποίηση της επικοινωνίας κατά την εκτέλεση
- Εκτέλεση ερωτημάτων
  - Επικοινωνία μόνο για τα κομμάτια του ερωτήματος που χρειάζονται απομακρυσμένα δεδομένα

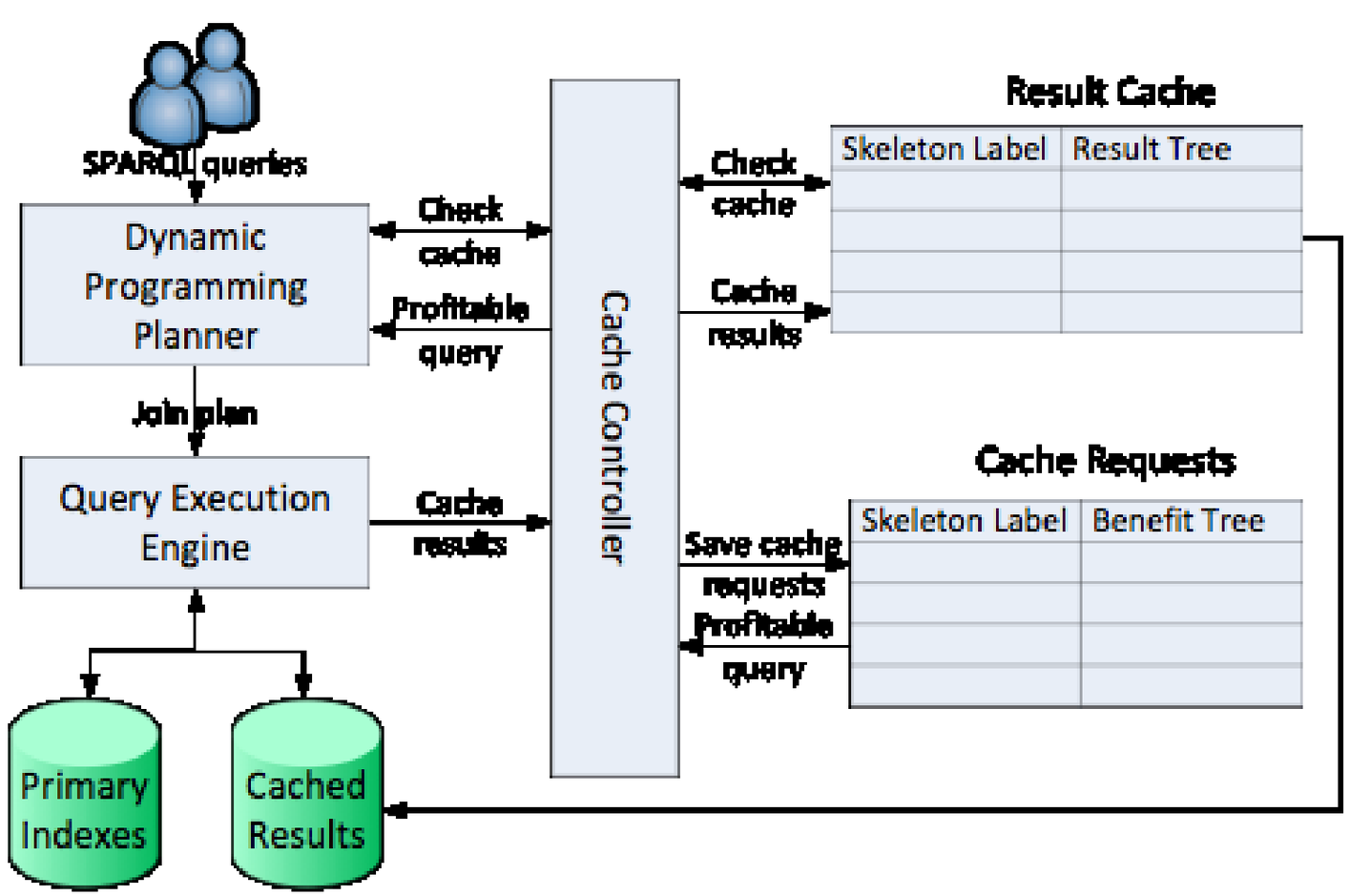
# Chameleon-db

- Χρήση ευρετηρίων που περιέχουν αποτελέσματα για graph query patterns
  - Partition με βάση graph patterns
  - Εύρεση του καλύτερου partitioning με βάση τα ερωτήματα του workload



# H2RDF+ SPARQL caching

- Canonical labelling για SPARQL ερωτήματα
- Χρήση του label για δημιουργία κρυφής μνήμης
- Επέκτεση ενός DP planner για cache requests
  - Ελέγχει αν η κρυφή μνήμη περιέχει χρήσιμα αποτελέσματα για κάθε subgraph του ερωτήματος
- Cache controller παρακολουθεί τα cache requests
  - Βρίσκει και εκτελεί “profitable” ερωτήματα



# AdPart

- Αρχικό Hash based partitioning
  - Ομαδοποίηση των δεδομένων με βάση το subject τους
  - Star ερωτήματα μπορούν να εκτελεστούν με τοπικά δεδομένα
- Παρακολούθηση του workload
  - Εύρεση των hot query pattern
  - Συνδυασμός επιμέρους ερωτημάτων με χρήση OPTIONAL
  - Repartition ή replication αυτών των pattern
- Εκτέλεση ερωτημάτων
  - MPI asynchronous joins
  - Αν τα δεδομένα μπορούν να βρεθούν τοπικά εκτελείται το ερώτημα χωρίς επικοινωνία

# Ερωτήσεις