European Union
European Social Fund

OPERATIONAL PROGRAMME
EDUCATION AND LIFELONG LEARNING
investing in knowledge society
MINISTRY OF EDUCATION & RELIGIOUS AFFAIRS
MANAGING AUTHORITY

NSRF 2007-2013
programme for development
EUROPEAN SOCIAL FUND

Co-financed by Greece and the European Union

A Map-Reduce algorithm for querying linked data based on query decomposition into stars

Christos Nomikos[1], Manolis Gergatsoulis[2], Eleftherios Kalogeros[2], Matthew Damigos[2]

[1]Department of Computer Science and Engineering, University of Ioannina, Greece
[2]Database & Information Systems Group (DBIS), Department of Archives, Library Science and Museology, School of Information Science and Informatics, Ionian University, Greece

March 28, 2014

DBIS database & information systems group ionian university

Structure of the Presentation

Data and Query Graphs

Data Graphs Partitioning

Query Decomposition

Embeddings

Query Evaluation Strategy

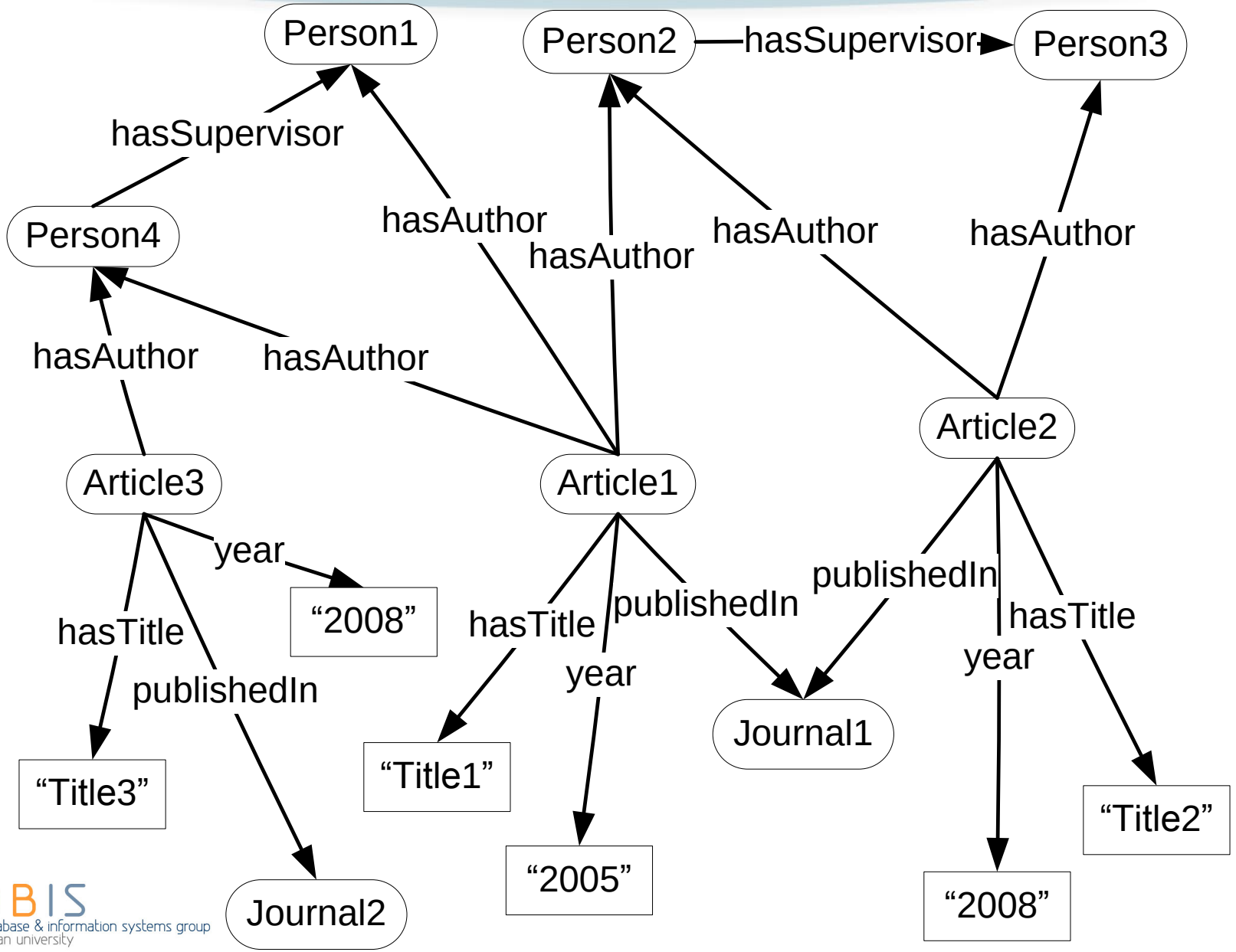
Map-Reduce

A Query Evaluation Algorithm Based on Map-Reduce

Conclusions & Future Work

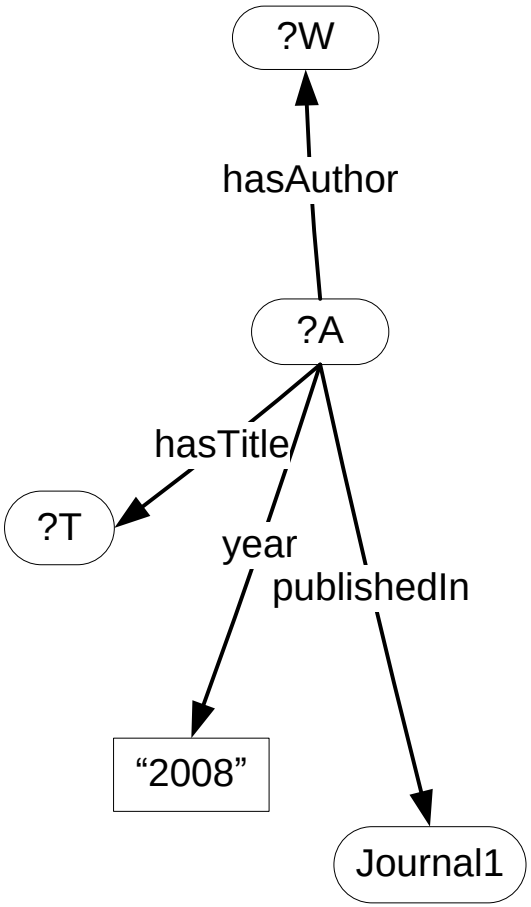
DBIS
database & information systems group
ionian university

Data Graph



DBIS database & information systems group ionian university

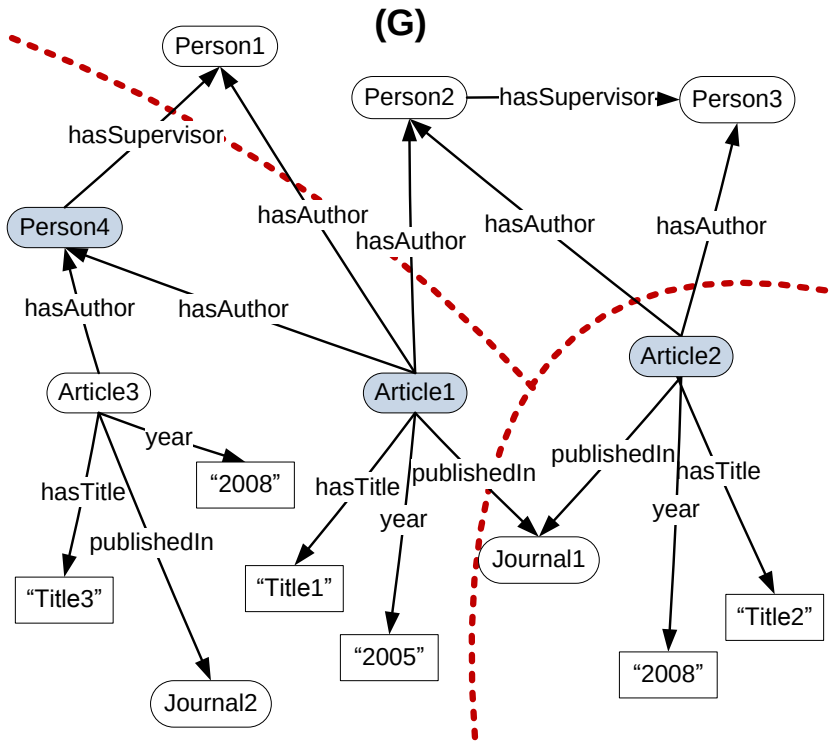
Star Query Graph

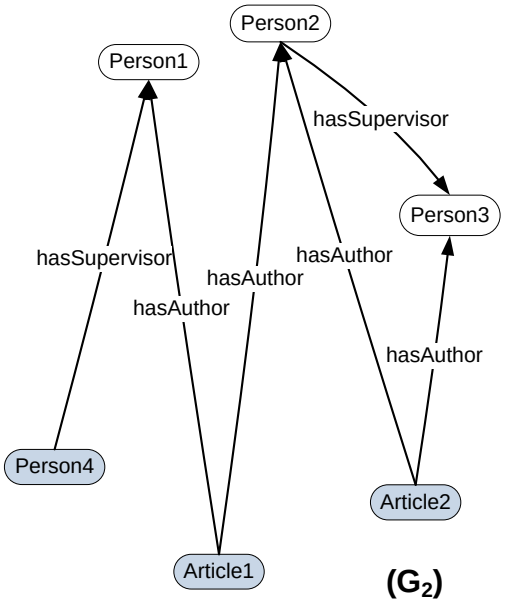


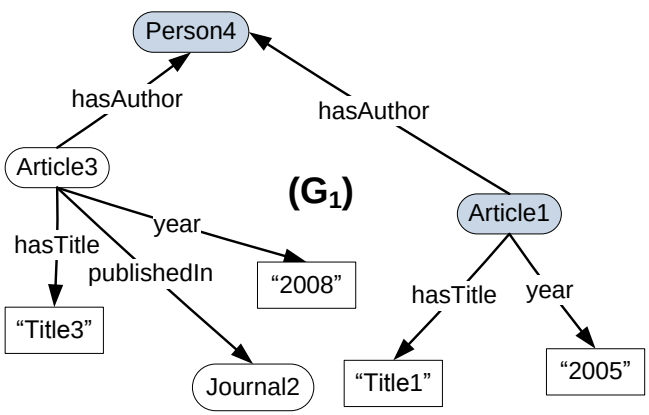
Asks for the Titles (variable $?T$) and Authors (variable $?W$) of the articles (variable $?A$) published in the journal *Journal*1 in year 2008.

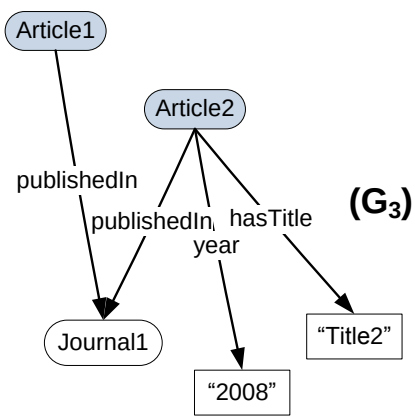
- Node $?A$ is the central node of the star query because is the common node in all the query triples

Data Graph Partitioning



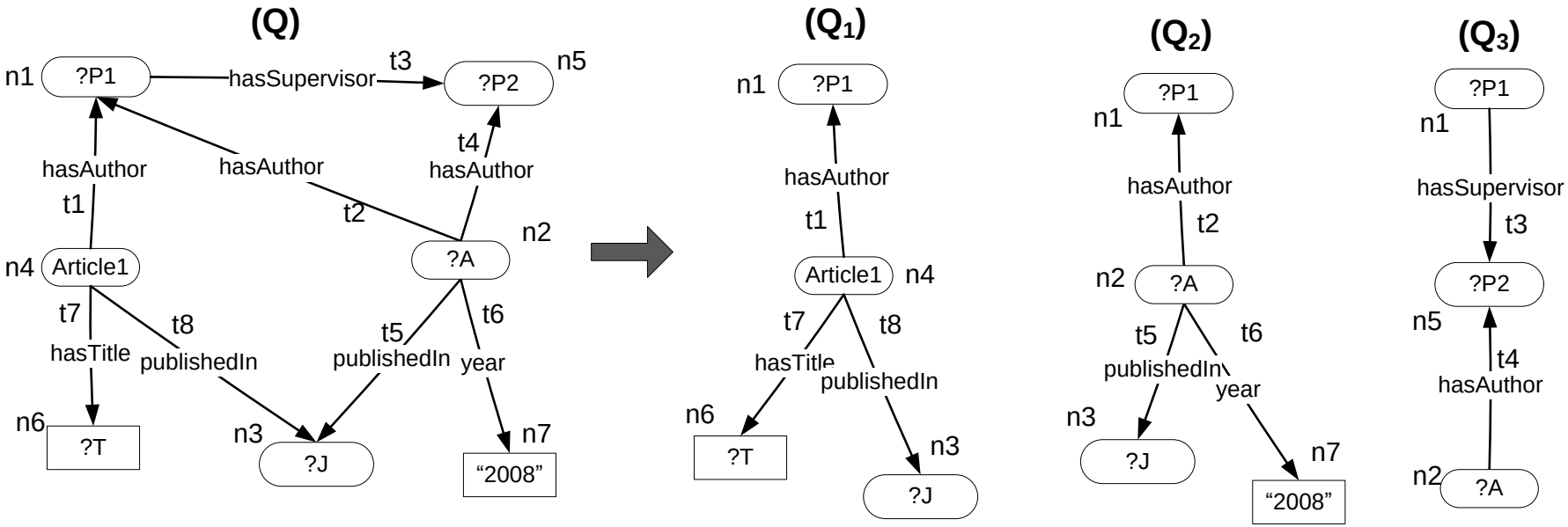






DBIS
database & information systems group
ionian university

Query Decomposition into Star Subqueries



- Branching Nodes (BN)
  - $n_1 \longrightarrow (Q_1, Q_2, Q_3)$
  - $n_2 \longrightarrow (Q_2, Q_3)$
  - $n_3 \longrightarrow (Q_1, Q_2)$
- Missing Branching Nodes (NBL)
  - $[(n_2, Q_1), (n_3, Q_3)]$

DBIS database & information systems group ionian university

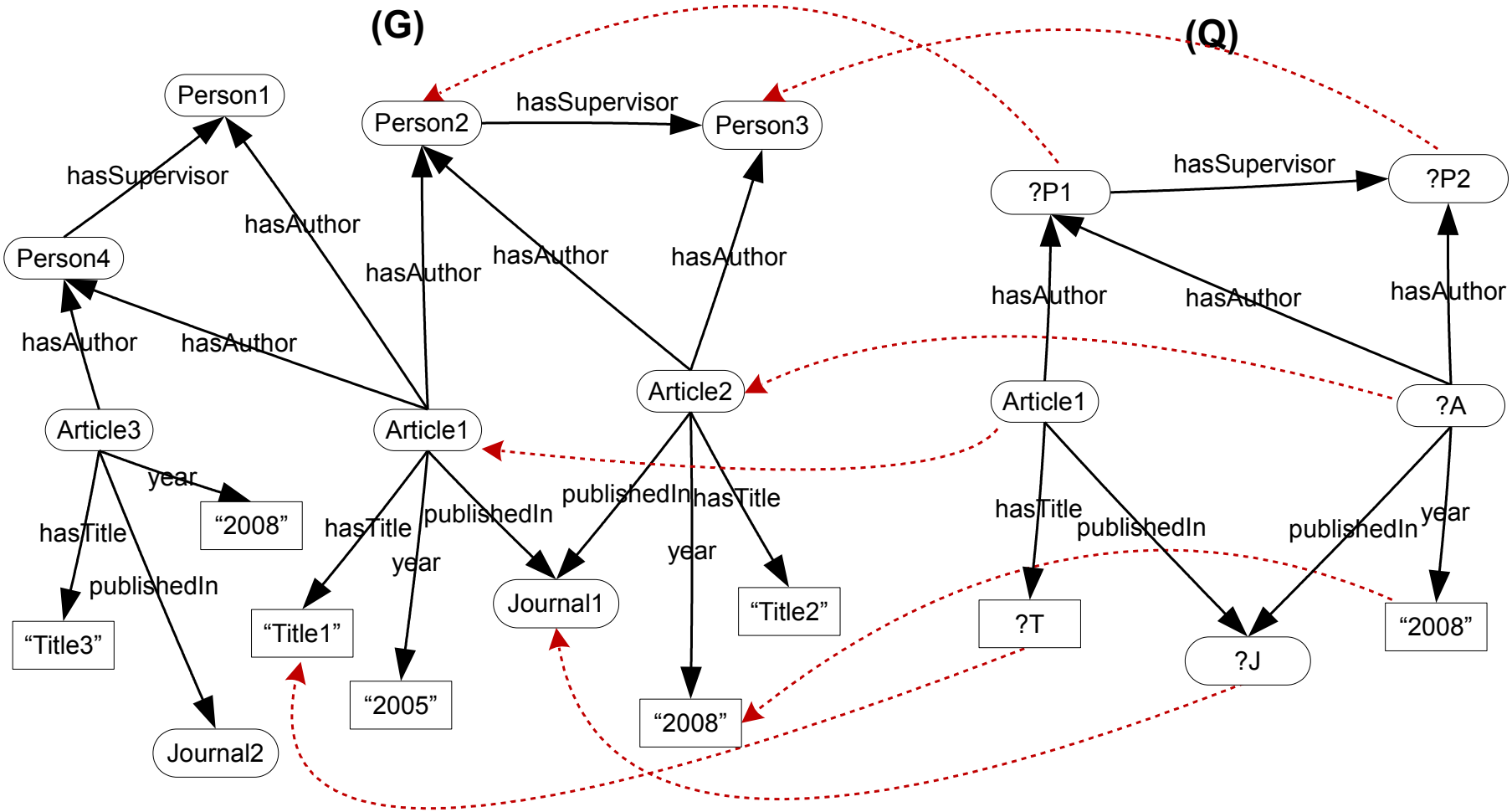
Embedding (1/2)

- Embedding e is a total mapping $e : nd(Q) \rightarrow nd(G)$ with the following properties:
  1. For each non-variable node $v \in nd(Q)$, it is $e(v) = v$.
  2. For each triple $(s, p, o) \in Q$, $(e(s), p, e(o))$ is in G.
- Two embeddings e_1 and e_2 are compatible if they have the same values in their common nodes.
- The join of e_1 and e_2 is the embedding e of $Q_1 \cup Q_2$ in G defined as follows:

$$e(v) = \begin{cases} e_1(v) & \text{if } v \in nd(Q_1) \\ e_2(v) & \text{otherwise} \end{cases}$$

- Theorem: e is an embedding of Q in G if and only if there exist mutually compatible embeddings $e_1, \ldots, e_n$ of $Q_1, \ldots, Q_n$ in G such that the join of $e_1, \ldots, e_n$ is e
- An embedding e of Q to G is an answer

Embedding (2/2)



The answer obtained is $(?P1, ?A, ?J, ?P2, ?T) =$ $(Person2, Article2, Journal1, Person3, \text{“}Title1\text{”})$.

DBIS database & information systems group ionian university

Query Evaluation Strategy

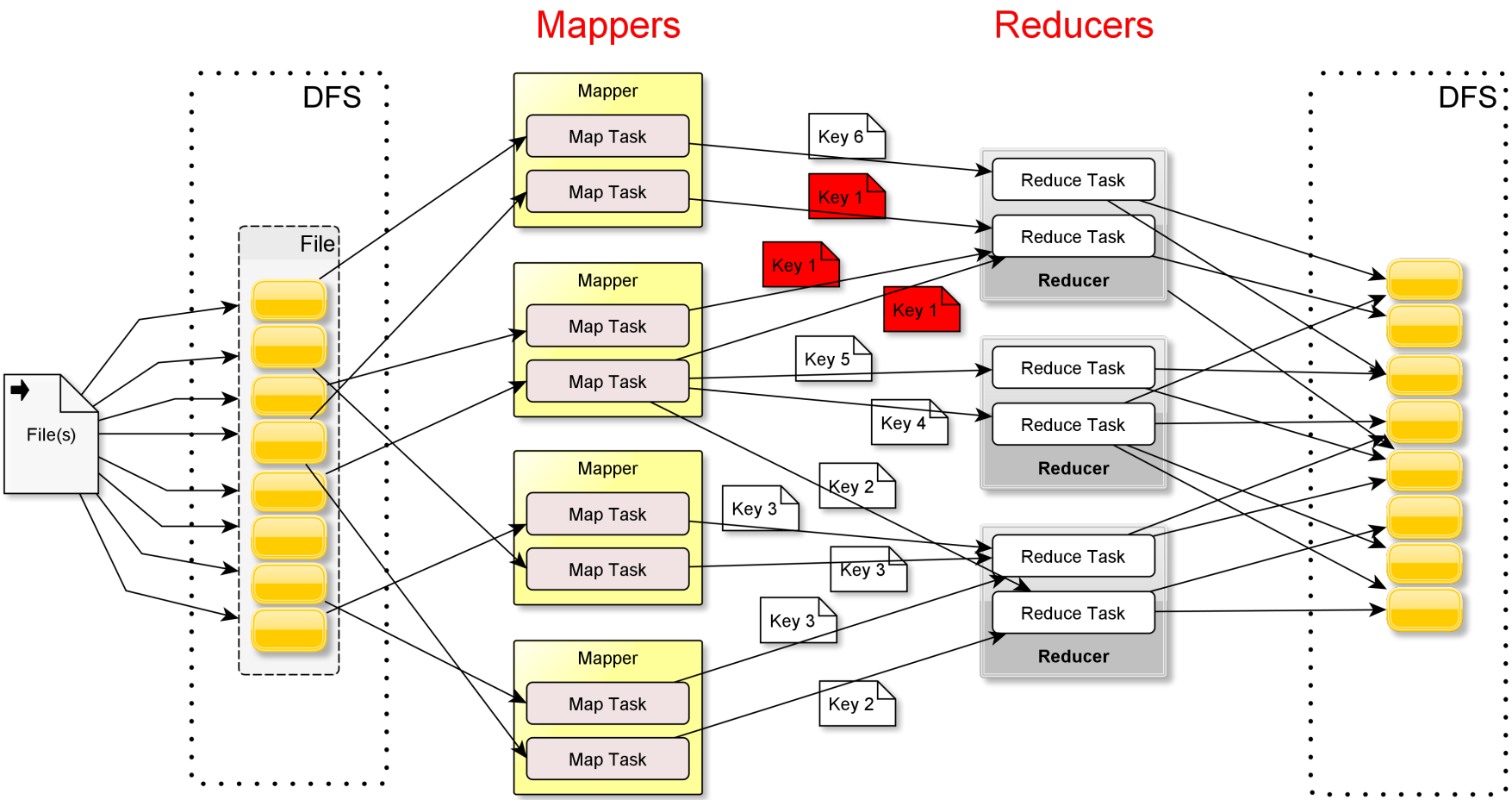
The data graph G is partitioned into m segments $G_1, \ldots, G_m$, each of them is stored in different computer nodes.

- **Step 1:** Decompose query Q into star subqueries $Q_1, \ldots, Q_n$.
- **Step 2:** Compute all possible embeddings of each triple in Q in every segment G_i of G.
- **Step 3:** For each subquery Q_j, collect the embeddings of all triples in Q_j and join compatible embeddings in all possible ways to compute the embeddings of Q_j in G.
- **Step 4:** Join compatible embeddings $Q_1, \ldots, Q_n$ in all possible ways to compute the embeddings of Q in G.

Map-Reduce (1/2)

- MapReduce: a programming model based on the user-defined functions Map and Reduce which run in isolation in cluster nodes.
- Map applies on one or more files in DFS and sends [key,value] pairs to the reducers. This process, called Map task, runs on a node called Mapper. A mapper may run multiple map tasks over different input files.
- All pairs with the same key initialize a single reduce process, called Reduce task. A reduce task also results [key,value] pairs stored in the DFS. This comprises one MapReduce step.
- The output of the reducer can be set as the input of a map function, creating in this way procedures of multiple MR steps.

Map-Reduce (2/2)



A Map Reduce Algorithm to Implement the Query Evaluation Strategy

The proposed strategy can be implemented though a Map-Reduce algorithm which consists of:

- A preprocessing phase
- Two Map-Reduce phases

The Preprocessing Phase (1/2)

Preprocessing phase does the following:

- Q is decomposed into a set of star subqueries $Q_1, \ldots, Q_n$.
- Numbering functions I_t, I_b and I_{nb} assign unique integer IDs to triples (1 to $|Q|$), to branching nodes (1 to $|B(Q)|$), and to non-branching nodes ($|B(Q)| + 1$ to $|nd(Q)|$), of Q.
- The lists: *BN* of the branching nodes, *NBN* of non-branching nodes, and *tripleList* of the triples of Q, are constructed.
- A query prototype (*bnFlags*, *nbnFlags*, *tFlags*) relates to each Q_i, where *bnFlags*, *nbnFlags* and *tFlags* have one place for each branching node, non-branching node, and triple in Q, respectively, to denote its presence (denoted by "+") or absence (denoted by "-") in the subquery Q_i.
- The set $NBL = \{(b_i, Q_j) |\ b_i \in BN \text{ and } b_i \notin nd(Q_j)\}$, is constructed.

The Preprocessing Phase (2/2)

- The embeddings of a (sub)query are represented as triples of tuples $(BNt, NBNt, tF)$, where:
  - BNt (resp. $NBNt$) stores the images of branching (resp. non-branching) nodes.
  - Asterisks ('*') are used to represent missing values.
  - tF keeps tracks for the triples participating in the embedding ('+'/'-' sign in the corresponding place of tF).

Preprocessing emits the above to the mappers of Phase 1 with key the pair $(subqueryID, SegmentID)$.

Mapper of Phase 1

The operation of the mapper is divided into two parts

- Part1: Computes all the embeddings of each triple of Q_i in G_j that map the central node c_i to a border node, and emits the results to the reducers of Phase1 with key $(Q_i, e(c_i))$.
- Part2: Computes all the embeddings of Q_i in G_j, which map c_i to a non-border node of G_j and emits the results to the mappers of Phase 2 with key Q_i. The values of missing braching nodes are also emitted.

Reducer of Phase 1

- Input key: (Q_i, v).
- Input values: a list of pairs of the form (n_k, u) where n_k is a node of Q_i different from c_i and u is a possible value for n_k in an embedding of Q_i in G.
- The Reducer computes all the embeddings of Q_i that map central node c_i of Q_i to v and emits the results with key Q_i. Reducer emits also the values of missing branching nodes.

Mapper of Phase 2

- Input key: the ID of a subquery Q_i
- Input Values: a set E of the parts (bn, nbn) of the embeddings of Q_i and a set V of pairs (i, v), where v is a candidate value for $bn[i]$
- The Mapper gets the embeddings of a subquery and fills in its missing branching node values using values from the embeddings of other subqueries.
- Emits the resulted embeddings to the reducers of Phase 2 with key the tuple of the branching node values *bn*

Reducer of Phase 2

- Input key: a tuple of branching node values
- Input Values: pairs of the form (Q_i, a tuple of non-branching nodes)
- A reducer task selects one embedding for each subquery in $(Q_1, \ldots, Q_n)$ and joins them to construct an answer of Q (Note: the joined embeddings are, by construction, compatible)

Experimental Results (1/4)

- Experimental implementation using Hadoop 1.0.4. on a cluster of 14 nodes of the following characteristics: Intel Pentium(R) Dual-Core CPU E5700 3.00GHz with 4GB RAM.
- Datasets used: obtained and adapted from the Lehigh University Benchmark (LUBM).
- Graph segments are stored in different nodes, in relational MySQL databases whose schema consists of two tables, one containing the triples and the other the border nodes.
- Embeddings are obtained by transforming (sub)query graphs to SQL queries and applying them to databases.
- Five different datasets randomly split into 14 segments of equal size, stored in cluster nodes, used in our experiments.

Experimental Results (2/4)

- The algorithm is scalable in terms of the dataset size
- The degree of the star (i.e. the number of the triples in the star) also affects the execution time
  the larger the degree the smaller the execution time.

Dataset size	stars of degree 6	Stars of degree 3	Stars of degree 2	Num of Answers
113.6MB	190	219	254	93
231.6MB	212	258	280	189
491.5MB	310	369	459	402
1.2GB	650	689	727	999
2.5GB	1149	1200	1261	2007

Table: Execution times (in seconds) for three different star-decompositions of a query in five different datasets, using a cluster of 14 nodes.

Experimental Results (3/4)

- The algorithm scales well by increasing the number of nodes in the cluster
  The same queries have been computed (using the same star-decompositions) in 4, 9 and 14 computer nodes, for a fixed data set of size 231.6MB

# nodes	stars of degree 6	stars of degree 3	stars of degree 2
4	278	322	343
9	271	309	316
14	212	258	280

Table: Execution times (in seconds) for three different star-decompositions of a query in a dataset of size 231.6MB, using clusters of 4, 9, and 14 nodes.

Experimental Results (4/4)

- The algorithm of this paper performs better in most cases (depending on the form of the given query) than the algorithm proposed in *[Globe2013] M. Gergatsoulis,C. Nomikos,E. Kalogeros, and M. Damigos. An Algorithm for Querying Linked Data Using Map-Reduce. In Globe, pp. 51-62, 2013.*.

Alg.	**query1**	**query2**	**query3**	**query4**	**query5**
[Globe2013]	8,24	9,44	9,03	10,37	5,06
this	4,27	5,45	5,25	11,53	5,59

Table: Execution times (in minutes) of two algorithms for five different queries, in a dataset of size 491,5MB, using a cluster of 14 nodes.

Conclusions (1/2)

- We propose a two-phase MapReduce algorithm for querying large ammount of distributed linked data, that extends our approach *M. Gergatsoulis,C. Nomikos,E. Kalogeros, and M. Damigos. An Algorithm for Querying Linked Data Using Map-Reduce. In Globe, pp. 51-62, 2013.*
- The algorithm is independent of the data partitioning and storing
- The input query is decomposed into star subqueries and the answers to these subqueries are computed and joined to obtain the answers to the given query

Conclusions (2/2)

- Experimental evaluation shows that the algorithm is scalable in terms of a) the size data graph b) number of nodes in the cluster
- We do not allow queries with variables in the place of predicates. However, it is easy to extend the algorithm to allow such variables.
- We assumed that a data triple cannot belong to two different graph segments. However, the proposed algorithm can also be applied to segments that violate this restriction.

Interesting Problems for Future Work

- How to partition the data triples so as to minimize the number of border nodes and/or the cost of evaluating queries.
- How to decompose the queries to achieve more efficient evaluation.

Thank you!