Crunch and Manage Graph data: the survival kit

Marie-Aude AUFAURE
Business Intelligence Team
Ecole Centrale Paris

Context

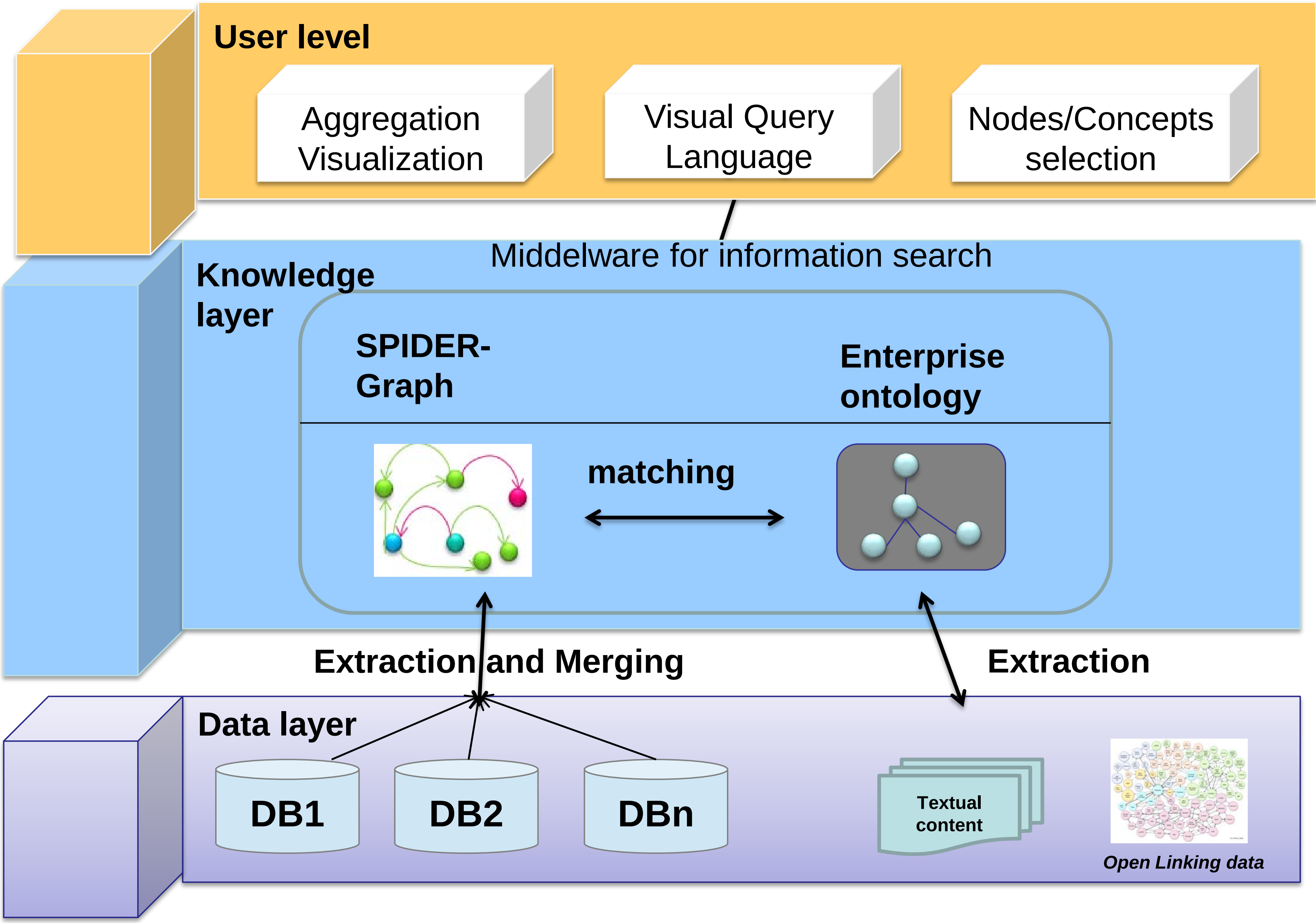
- Data everywhere – Big Data phenomenon
- Data are mainly **unstructured**
  - **80% of data manipulated in an enterprise are unstructured**
- Data are produced in **real time** and **distributed**
- Data come from **heterogeneous** sources in an unpredictable way
  - Mobile phone, sensors, computers, TV, etc.

⇒ Our objective: bridge the gap between structured data and unstructured content in a Business Intelligence perspective

What happens in an enterprise context?

- Unstructured information can be modeled by graphs (RDF, GraphML, etc.)
- But, **key information** is still stored in **relational databases**
  - Heterogeneous data
  - Non explicit representation of relations between objects
  - No link between all databases of an enterprise
- Objective: provide a **unique representation** for massive and heterogeneous data using graphs
- Problems:
  - Extracting these graphs
  - Processing and visualizing big graphs

$\Rightarrow$ We developed a set of tools for managing complex heterogeneous graphs



A set of tools for managing large heterogeneous graphs

DATA LAYER

Graph extraction from relational databases

NoSQL databases

- Relational databases are widely used for any kind of application because of:
  - Transaction management (ACID properties)
  - Advanced functionalities
  - Query language
- This richness is also a drawback
  - Distributed databases are complex
    - Transactions are difficult to manage (and consistency is not always necessary)
    - Cost of join operation is too high
- The NoSQL (“Not Only SQL”) movement appeared in 2009
  - Data scalability
  - Big Data Management
  - Compromise on ACID properties, transaction management

Distributed Systems

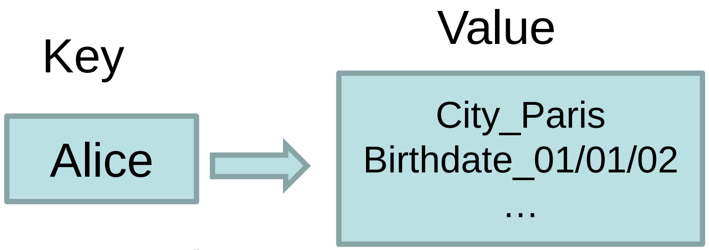
- Issues for data replication:
  - Performance (writing several copies of an item)
  - Consistency: is the ability of a system to guarantee that the transaction of each user run in isolation from other transactions, and never fails $\Rightarrow$ difficult to manage with data replication
- consistency levels:
  - Strong consistency (ACID properties): synchronous replication and possibly heavy locking mechanisms
  - Eventual consistency: the system guarantees the convergence towards a consistent step
  - Weak consistency: fully favors efficiency, never wait for write and read operations $\Rightarrow$ some requests may be processed on outdated data

The “CAP” theorem

- CAP:
  - Consistency (all nodes see the same data at the same time)
  - Availability (node failures do not prevent survivors from continuing to operate)
  - Partition tolerance (system continues to operate despite arbitrary message loss)
- A distributed system cannot simultaneously satisfy consistency and availability while being tolerant to failures

NoSQL models

- Key/Value (e.g. Dynamo):
  - One key, one value
  - Hashtable



- Document databases (e.g. MongoDB)
  - Key store, but the value is a (structured) document (XML or JSON)
  - Querying data is possible (by other means than just a key)
- Column databases (e.g. Cassandra)
  - Each key is associated with multiple attributes (number of columns dynamic)
  - Inspired by Google BigTable
- Graphs databases (e.g. Neo4j)
  - Inspired by Euler Graph Theory
  - Focused on modeling the structure of the data

Modelling with Cassandra

Key	Value
City	Paris
ZIP	75000

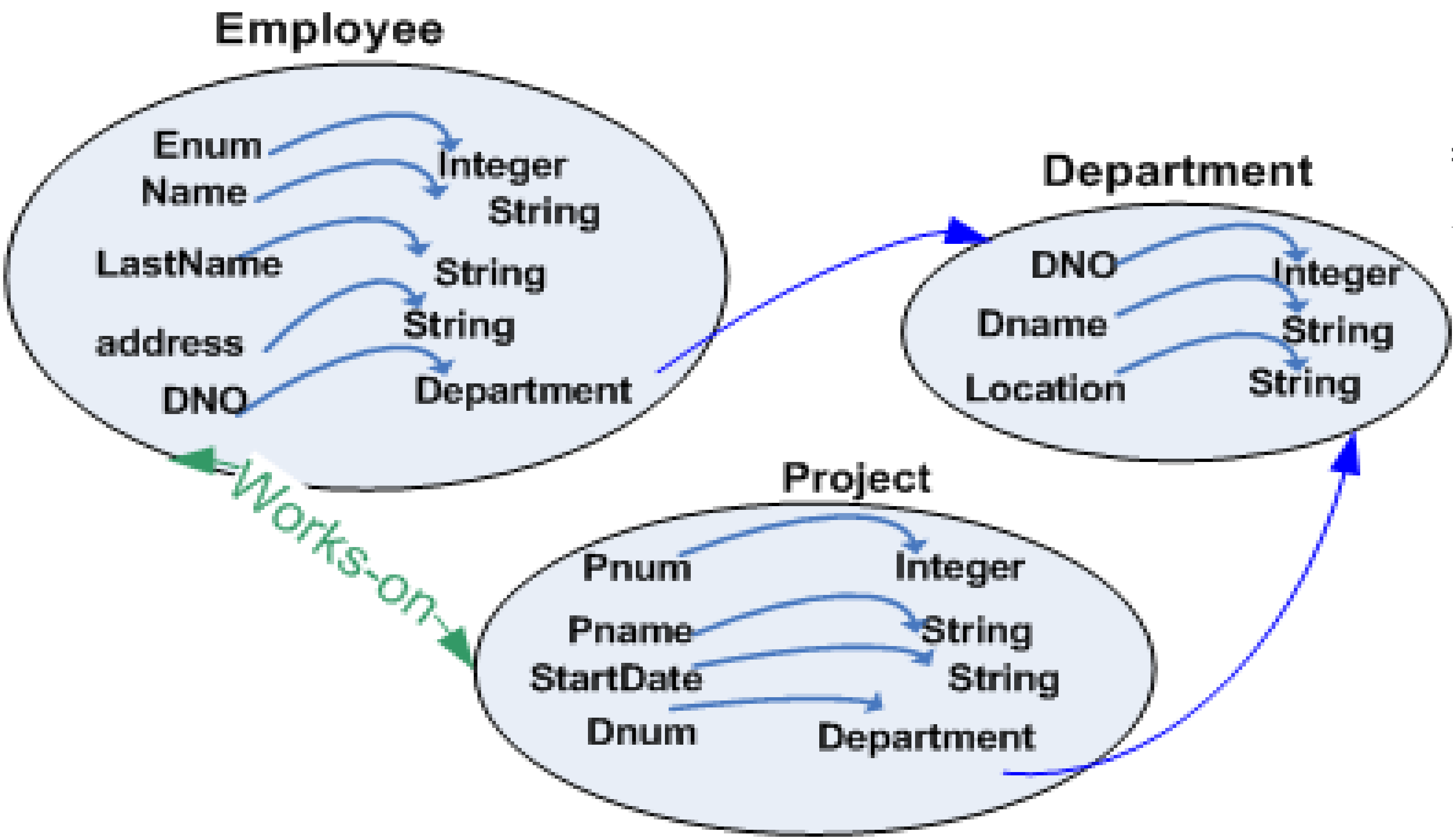
Key	Value	
Home	Key	Value
	City	Paris
	ZIP	75000
Work	Key	Value
	City	Levallois
	ZIP	92000

Key	Value		
Alice	Key	Value	
	Home	Key	Value
		City	Paris
		ZIP	75000
	Work	Key	Value
		City	Levallois
		ZIP	92000
Bob	Key	Value	
	Home	Key	Value
		City	Cannes
		ZIP	06400
	Work	Key	Value
		City	Antibes
		ZIP	06600

Variety of graphs

- From simple graphs (basic mathematic definition):
  - No information about nodes (all nodes have the same semantics, no attributes) $E \subseteq V \times V$
  - Mainly focus on the relations between objects
- To labeled and attributed graphs
  - Add semantic information to nodes
- And more complex structures like Hypergraphs and Hypernodes allowing nested structures (complex attributes and/or relations)
- Our need: a model able to represent heterogeneous graphs based on complex objects and a clear separation between the schema and the instance level

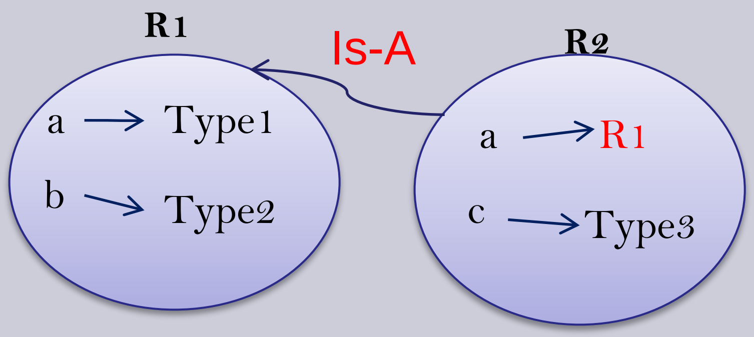
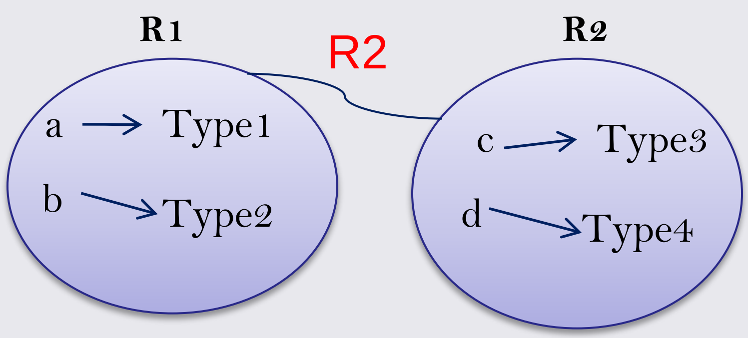
SPIDER-Graph Model



Data level: extraction of graphs from relational databases

- Definition of a set of transformation rules using metadata information (primary and foreign keys in the relational schema) for building a SPIDER graph (schema and instance levels)
- Each relation is transformed into a complex node
- Implicit relations are transformed into edges
- The resulting graph is populated
- Experimented with a real database
  - Containing 30 tables about 1788 students
  - SPIDER-Graph schema: 30 nodes and 48 relations
  - SPIDER-Graph instance: 12213 nodes and 13282 relations

Patterns examples

<table>
<tr><th>Relation</th><th>Transformation</th></tr>
<tr><td>R1 (a,b)
R2(a #,c)</td><td>
</td></tr>
<tr><td>R1 (a, b)
R2(a#, c#)
R3(c, d)</td><td>
</td></tr>
</table>

Transformation to a SPIDER-Graph Model: Schema Translation

Employee

<u>Enum</u>	Name	LastName	address	DNO#
01	Alain	Jones	Paris	1
02	Sara	James	London	2
03	Smith	Yan	Paris	1

Department

Dname	<u>DNO</u>	Location
Development	1	Paris
Modeling	2	London

Works_On

<u>Enum#</u>	<u>Pnum#</u>
01	101
02	102
03	101
03	102

Product

<u>numP#</u>	name	type	Price
123	Phone7	Mobile phone	500eur
204	Scr2	LCD screen	1200eur

Facility

<u>Fid#</u>	name	type	Pnum#
03	Ld 2	office	101

Project-Product

<u>Pnum#</u>	<u>numP#</u>	desc	quantity
101	123	hardware	10000
102	123	Software	234
101	204	Software	200

Project

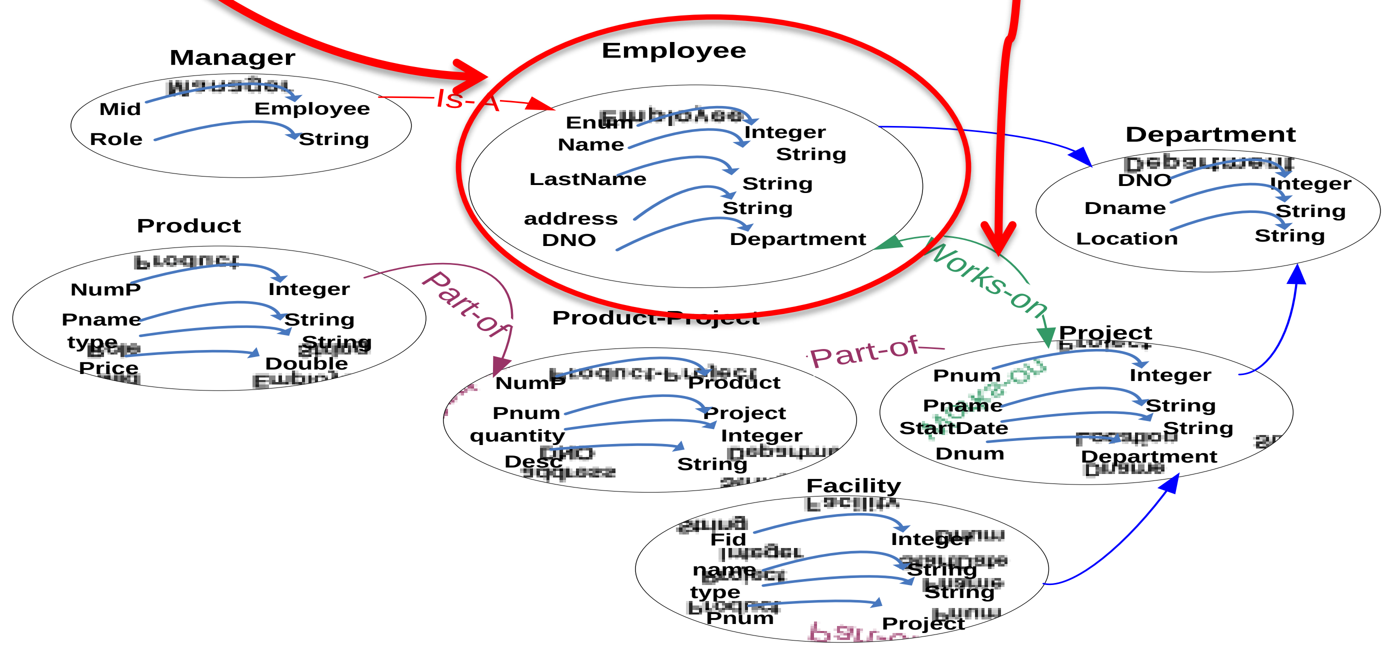
Pname	<u>Pnum</u>	Startdate	Dnum#
Trans_2	101	06-2009	1
Water209	102	03-2008	2

Manager

<u>Mid#</u>	Role
03	Managing the project team

(1) Complex-node creation

(2) Relations Identification



Transformation to a SPIDER-Graph Model: Data Migration

(3) Intance level (population)

Employee

Enum	Name	LastName	address	DNO#
01	Alain	Jones	Paris	1
02	Sara	James	London	2
03	Smith	Yan	Paris	1

Department

Dname	DNO	Location
Development	1	Paris
Modeling	2	London

Works_On

Enum#	Pnum#
01	101
02	102
03	101
03	102

Product

numP#	name	type	Price
123	Phone7	Mobile phone	500eur
204	Scr2	LCD screen	1200eur

Facility

Fid#	name	type	Pnum#
03	Ld 2	office	101

Project-Product

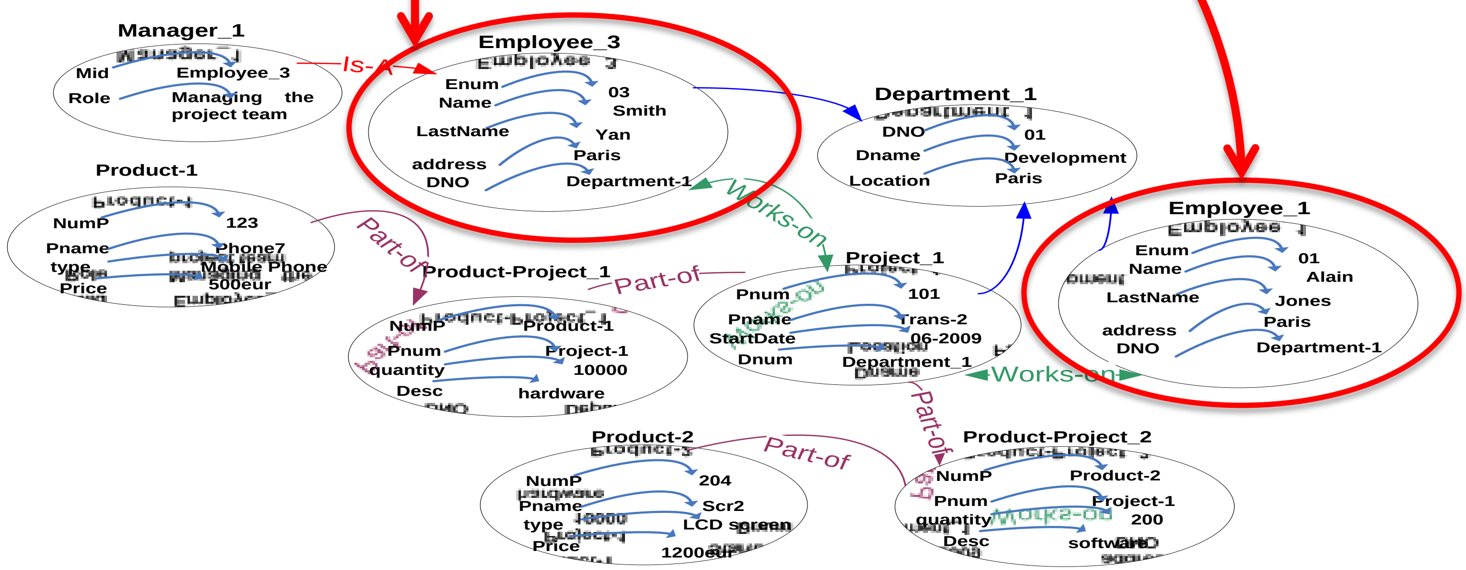
Pnum#	numP#	desc	quantity
101	123	hardware	10000
102	123	Software	234
101	204	Software	200

Project

Pname	Pnum	Startdate	Dnum#
Trans_2	101	06-2009	1
Water209	102	03-2008	2

Manager

Mid#	Role
03	Managing the project team



DATA LAYER

Graphs merging

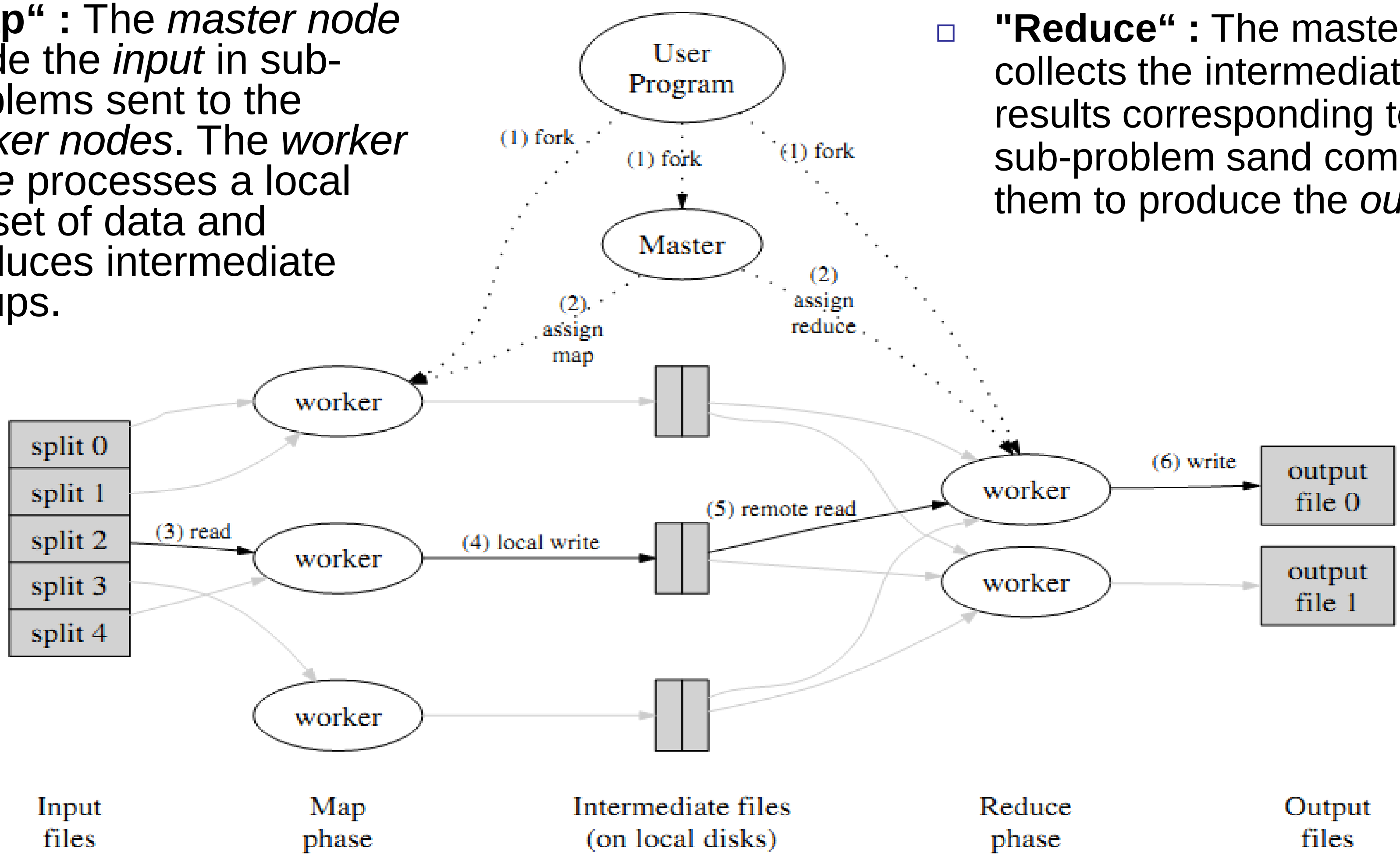
Motivation

- Being able to visualize information from different data sources into one graph
- Accelerate merging process through distributed computing
- Being able to treat very large graphs

- Our case study : extracting graphs from databases before merging them

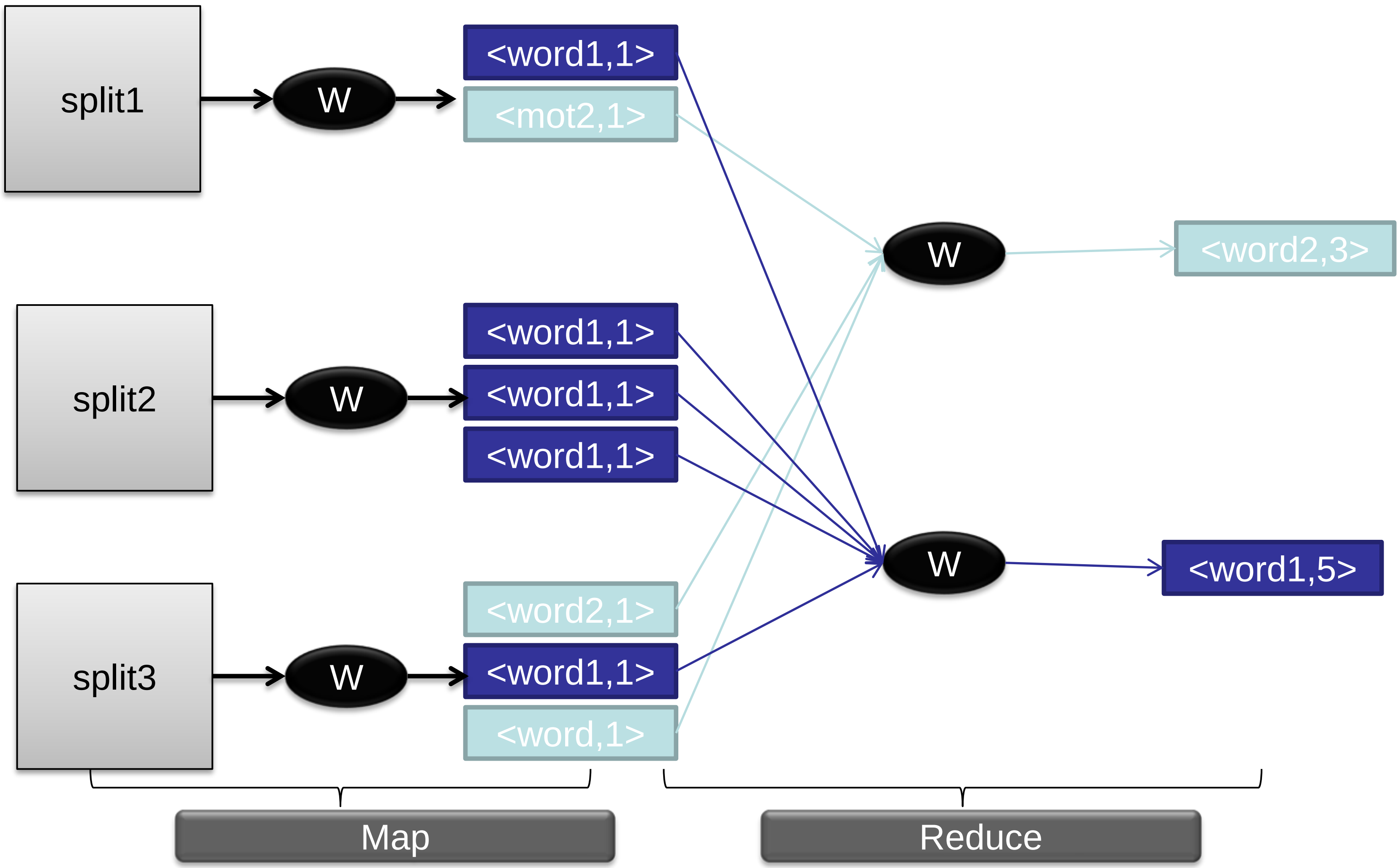
Distributed computing: Map/Reduce

- **"Map" :** The *master node* divide the *input* in sub-problems sent to the *worker nodes*. The *worker node* processes a local subset of data and produces intermediate groups.
- **"Reduce" :** The master node collects the intermediate results corresponding to a sub-problem sand combines them to produce the *output*

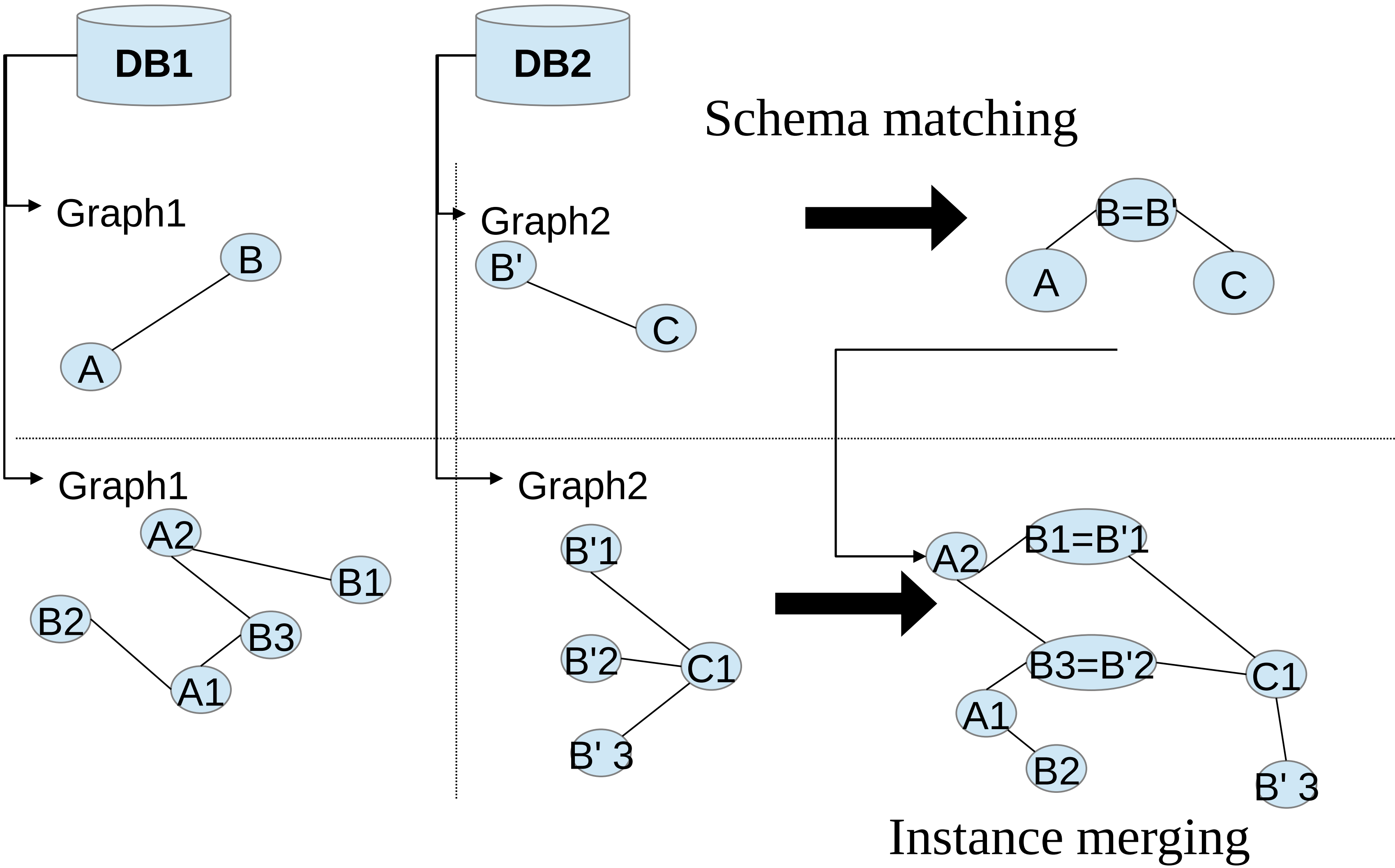


[J Dean & S Ghemawat]

Map/Reduce: a simple example



Algorithm

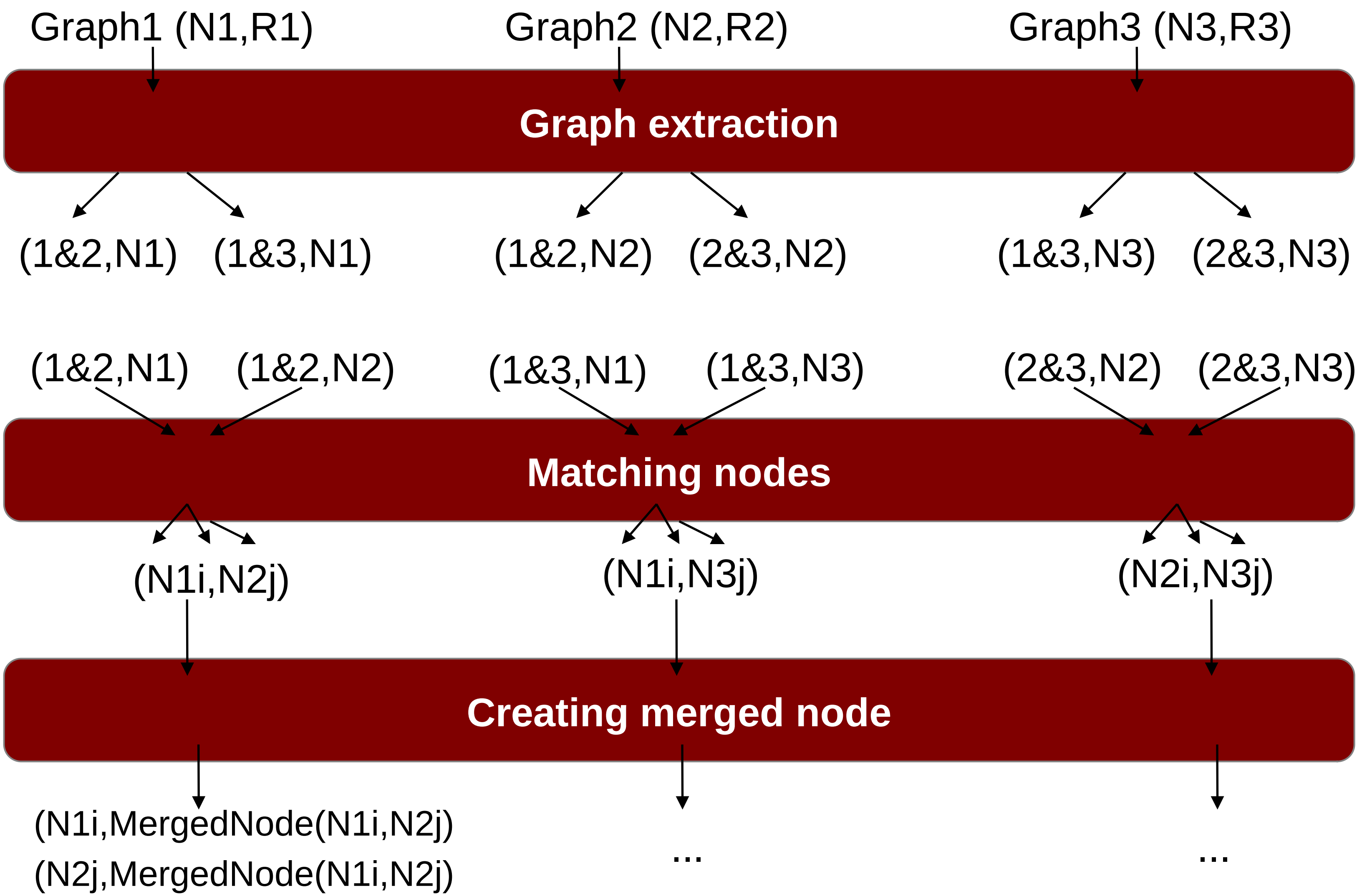

DB1
DB2
Schema matching
Graph1
Graph2
B
A
B'
C
B=B'
A
C
Graph1
Graph2
A2
B1
B2
B3
A1
B'1
B'2
C1
B' 3
A2
B1=B'1
B3=B'2
C1
A1
B2
B' 3
Instance merging

Schema matching algorithm

- Objective : return a mapping to prepare the merging
- 3 super steps :
  - Graph extraction
  - Matching nodes between two graphs
  - Creating the merged node

- Input : N schema graphs
- Output : List of nodes to be modified ($node_{i,fromGraphj}$; mergednode)

Schema matching algorithm



Matching two schema nodes

- For a given node N1j
  - We select the nodes whose name is quite similar using string similarity
    sim (N1j,N2i) > 0.4
  - In those nodes, we compute the similarity between attributes, and the match exists if simAttributes > treshold (0.9 for testing)
- Matching rules are produced:

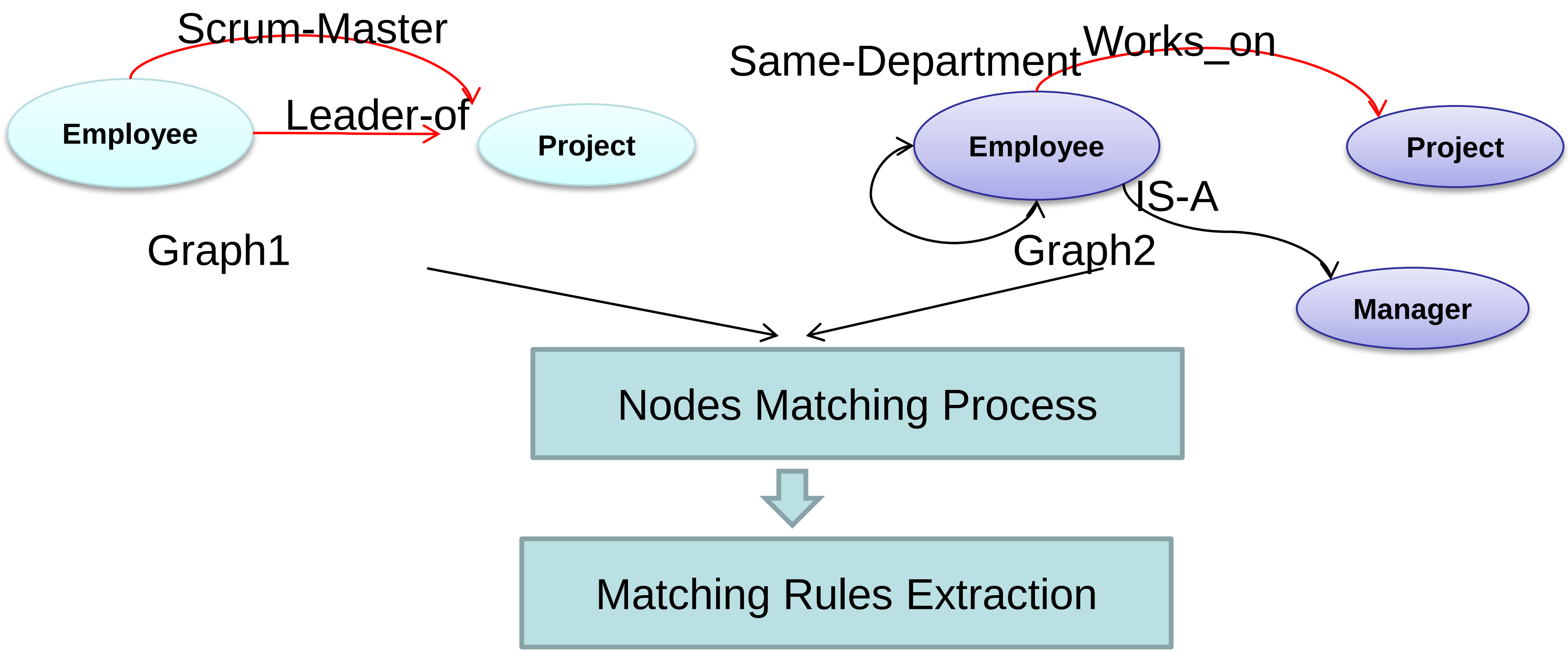
N1(name,type,attributes)

N2(name,type,attributes)

Merged node N:

- name(N1),
- type(N1),
- attr(N1) U (attr(N2)-attr(N1)∩attr(N2))

Schema Matching



Merge Graph1.Employee and Graph2.Employee

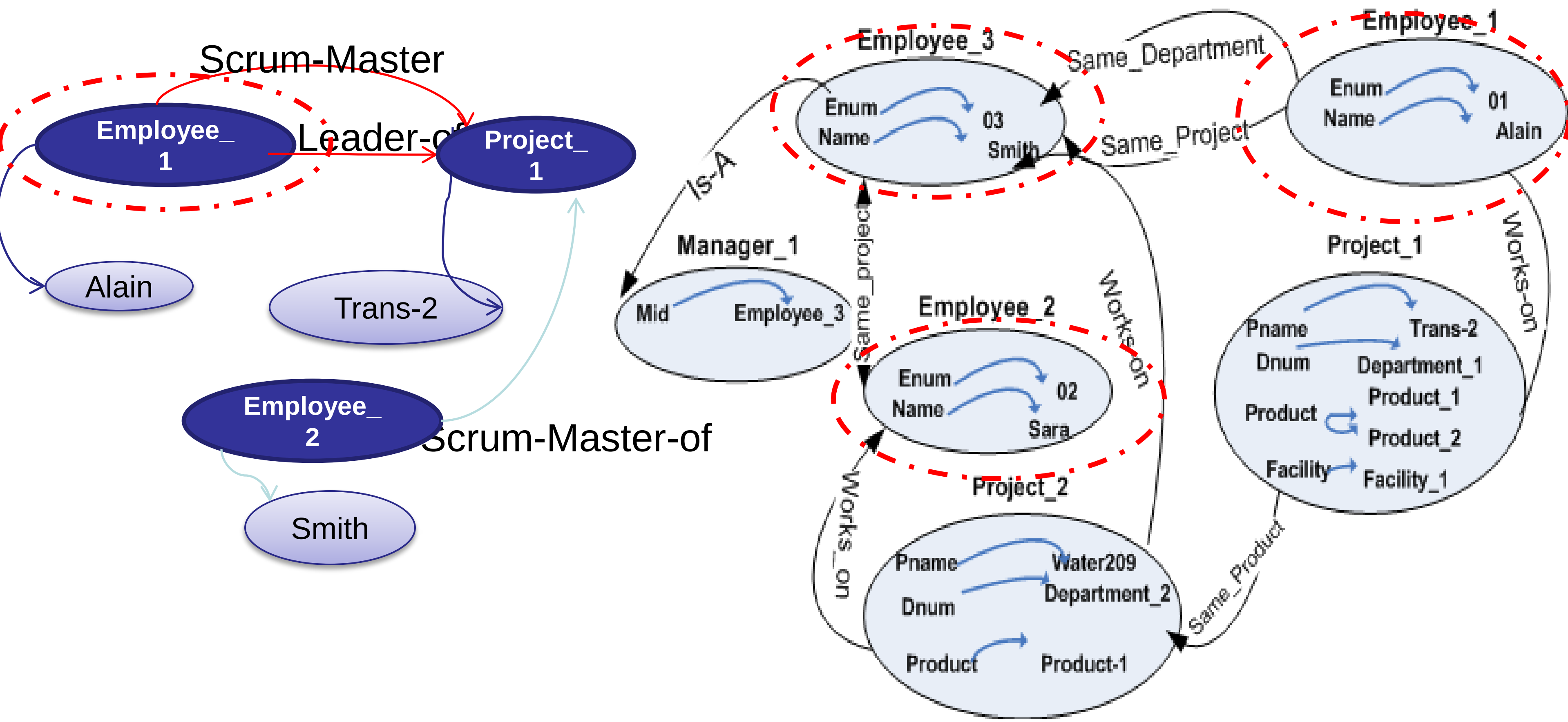
Merge Graph1.Project and Graph2.Project

Instance merging algorithm

- This algorithm can be divided in two parts:
  - Extracting candidates for matching
  - Matching instance nodes between two graphs
  - Creating the merged node
  - Creating the merged graph

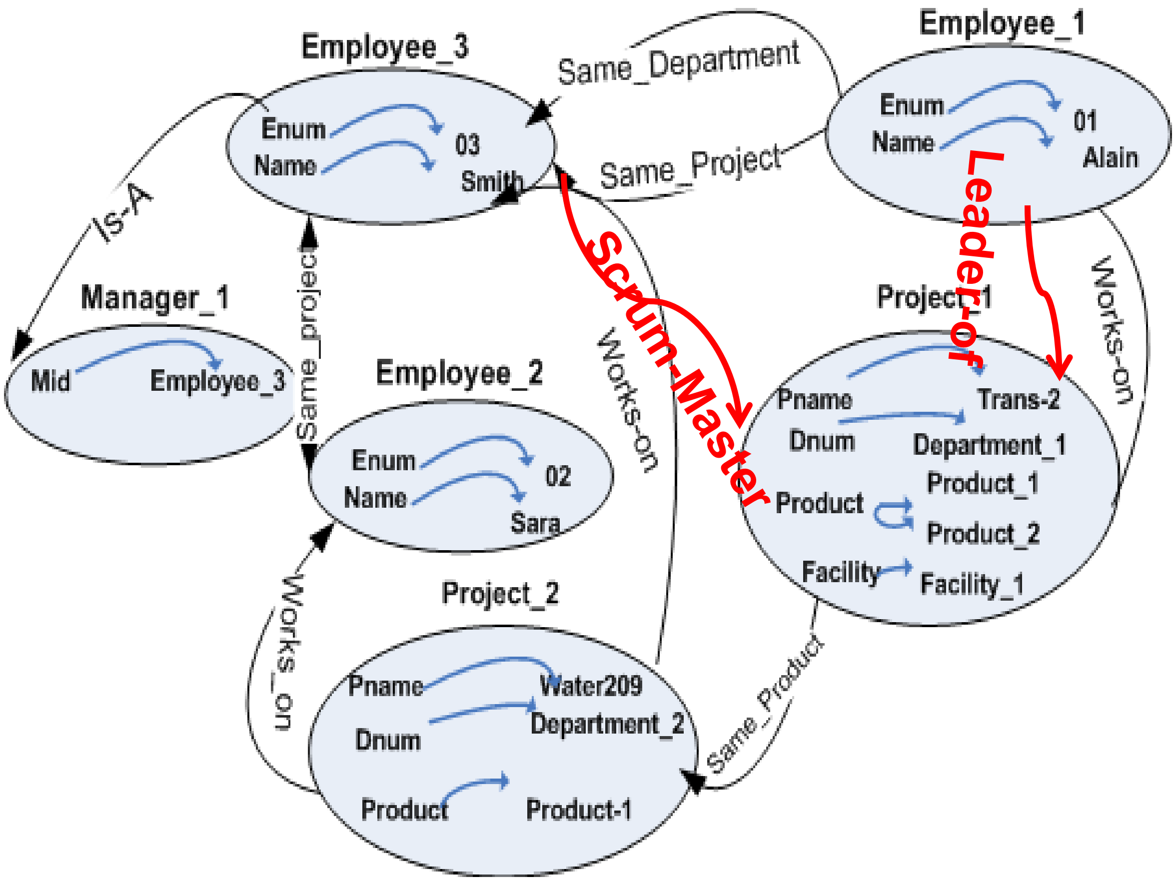
- For a given node N1j :
  - We determine the attributes matched
  - For those attributes, if the average of the values similarity is higher than a certain threshold β (around 0.8), it is a match

Instance Matching

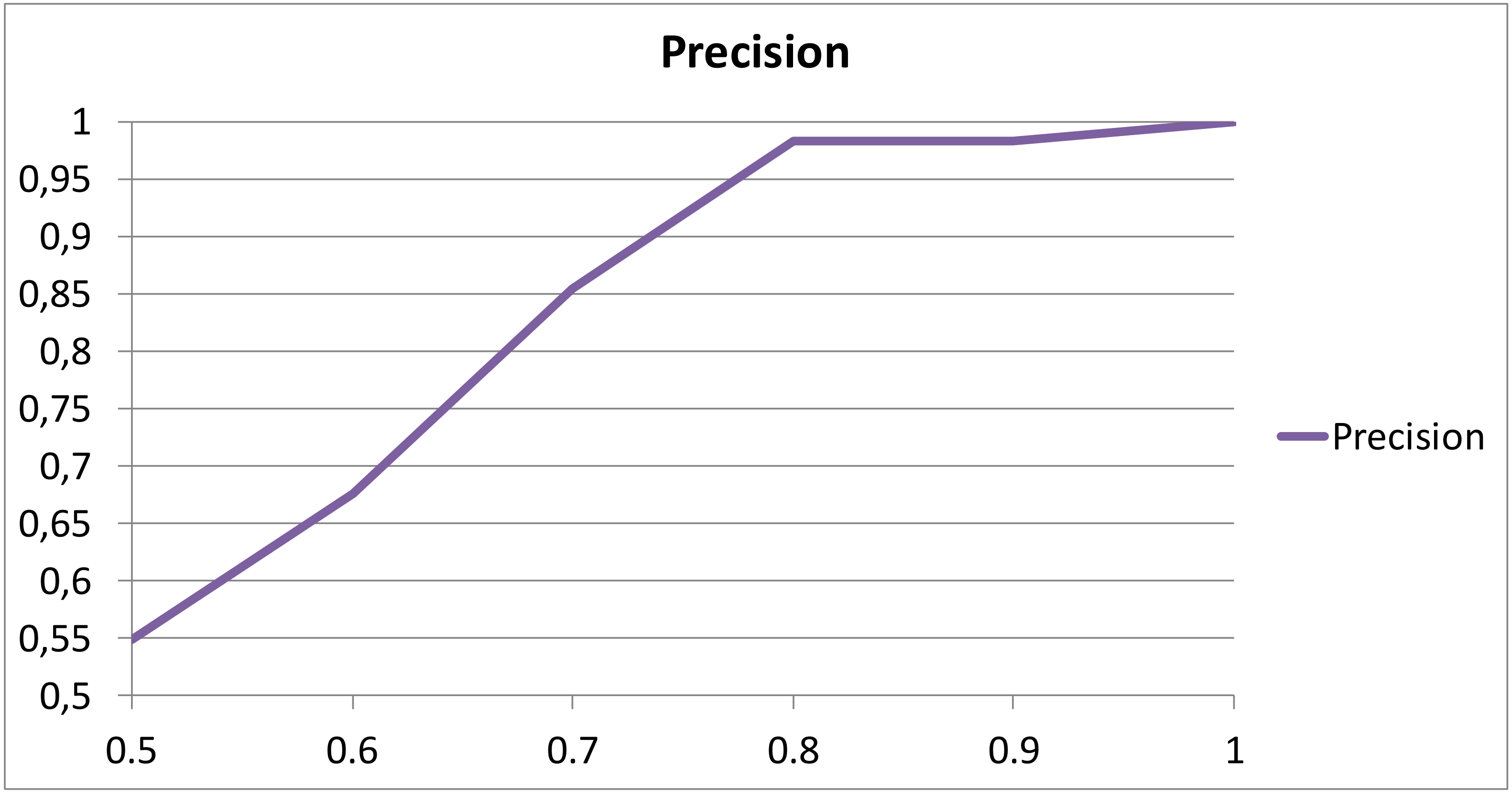


Graph Merging

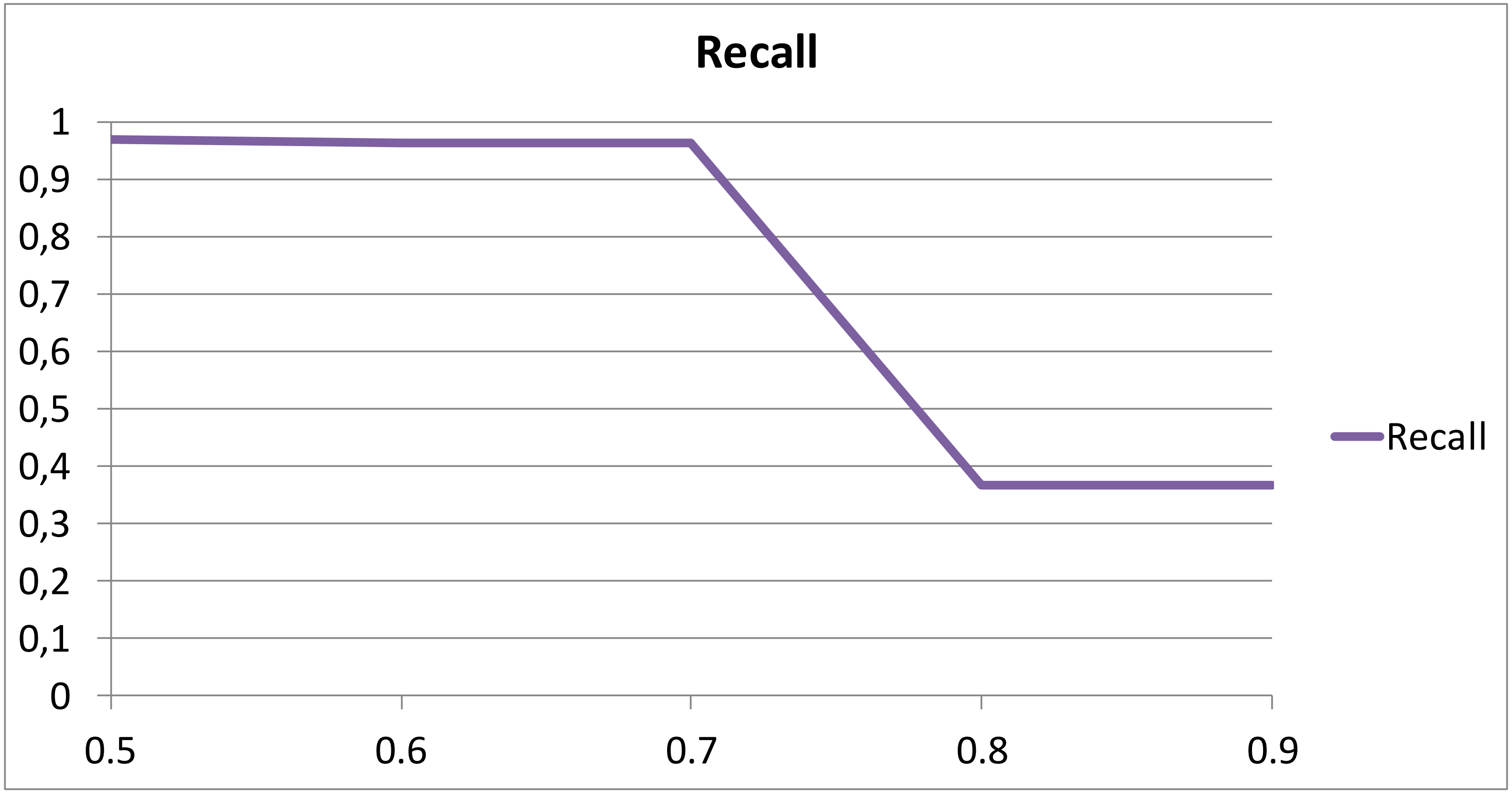
Apply the matching rules



Precision over the similarity threshold β



Recall over the similarity threshold β



Issues

- Performance evaluation:
  - Better with the increase of the number of graphs (due to the matching process)
- Precision and Recall:
  - similarity threshold between 0.7 and 0.8
- Limits of the string similarity measure
  - Synonyms are not taken into account (e.g. city and town)
- Precision
  - Jaro-Winkler precision is around 90% for normal text
- Test over a large cluster will have to be done
- The identification of matched nodes could be improved
- Hadoop is a batch process, not suited for real-time
- But, the algorithm is working on large complex graphs using Hadoop

USER LAYER

Graph aggregation and visualization

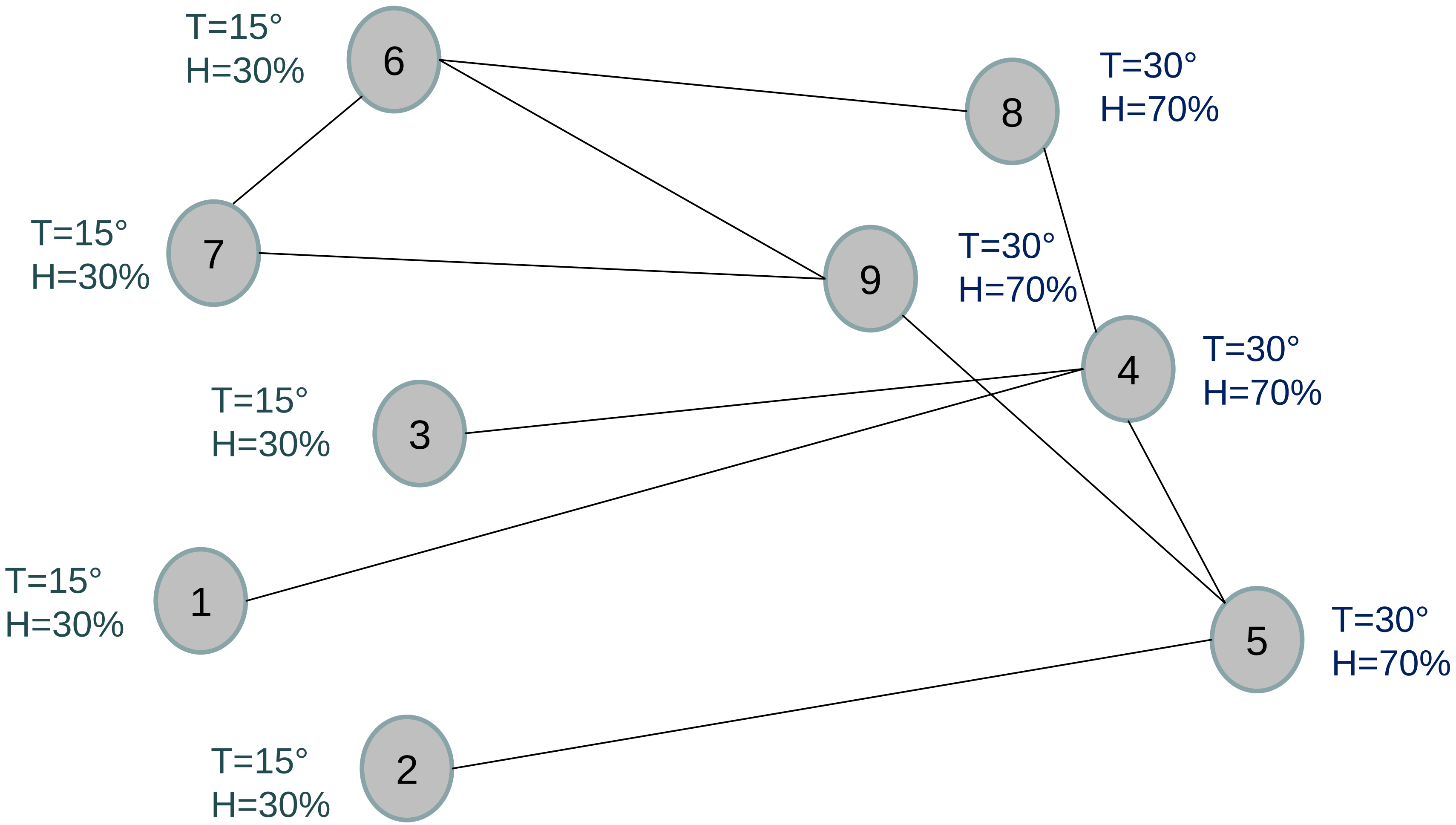
Graph Aggregation: SNAP & k-SNAP
Tian, Hankins and Patel (SIGMOD 2008)

- Summarization based on user-selected node attributes and relationships.
- Provide "drill-down" and "roll-up" abilities to navigate multi-resolution summaries.
- Produce meaningful summaries for real applications (and multiple points of view)
- Efficient and scalable for very large graphs

Main principle: compatible groupings

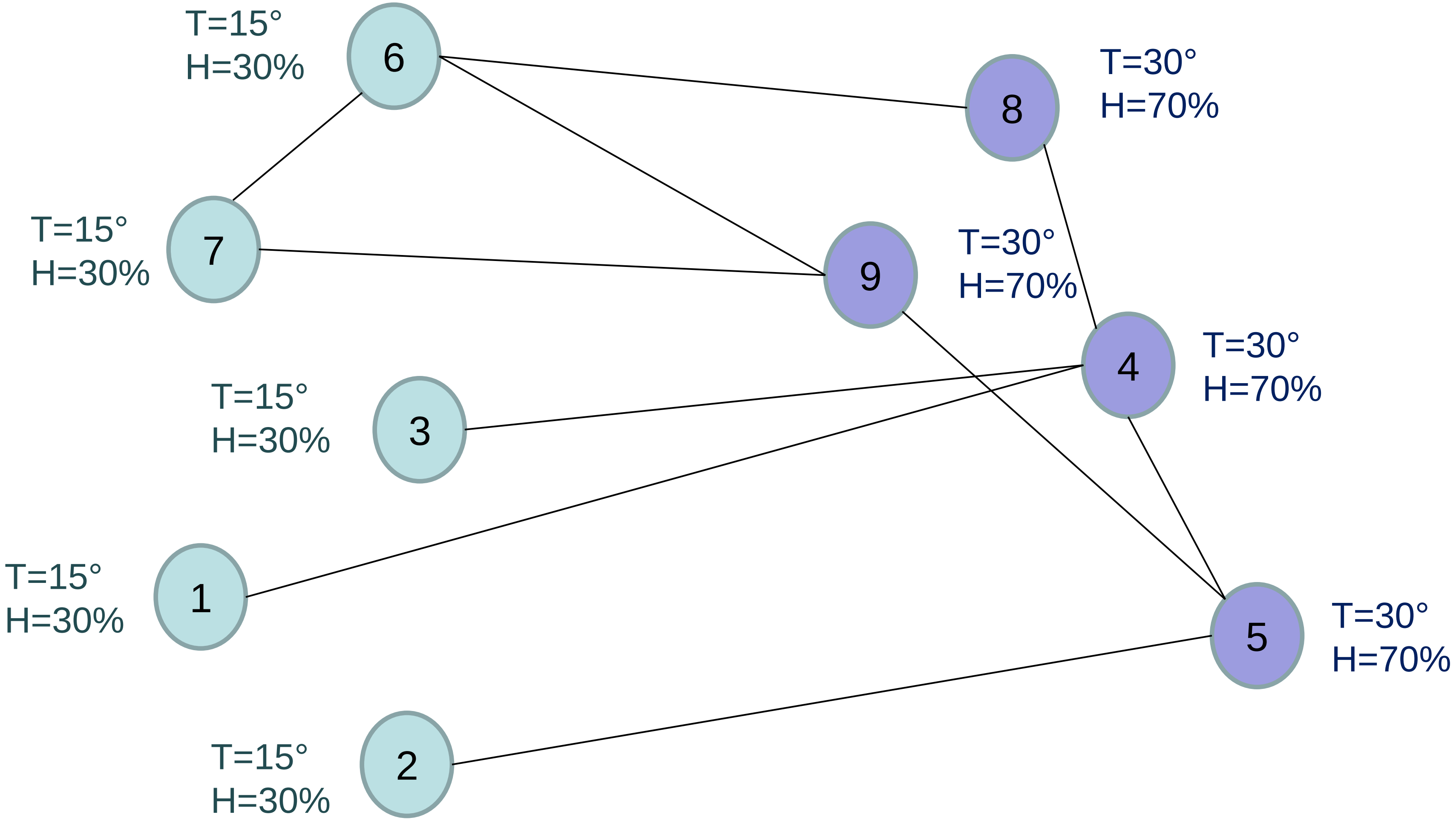
- A-compatible Grouping :
  - All nodes in the same group must have the same attributes.
- (A,R)-compatible Grouping:
  - A-compatible grouping,
  - all nodes in the same group must have the same neighbor groups.
- 2 steps: attributes selection and group division

Example



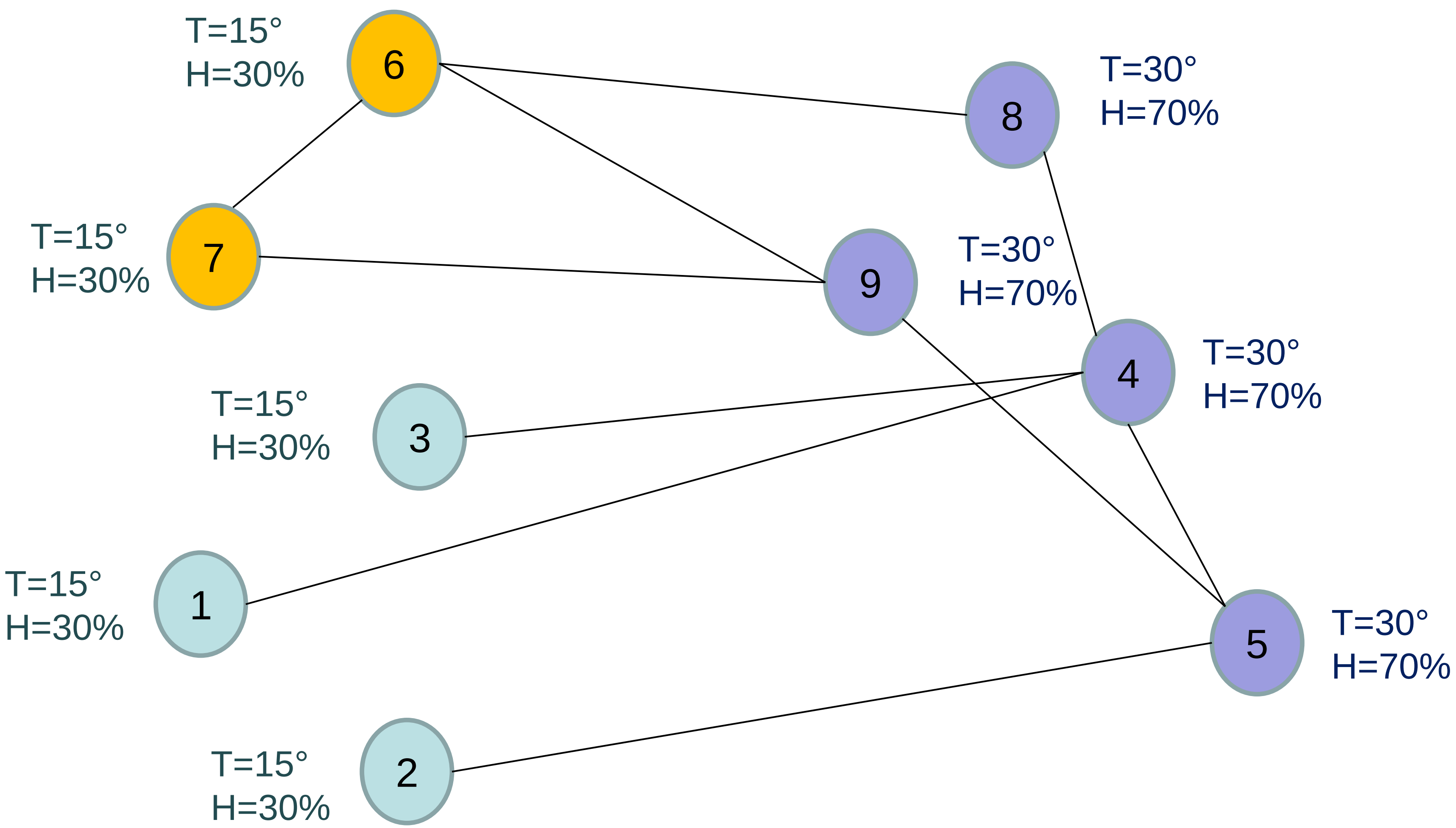
T=15°
H=30%
6
8
T=30°
H=70%
T=15°
H=30%
7
9
T=30°
H=70%
4
T=30°
H=70%
T=15°
H=30%
3
T=15°
H=30%
1
5
T=30°
H=70%
T=15°
H=30%
2

Example

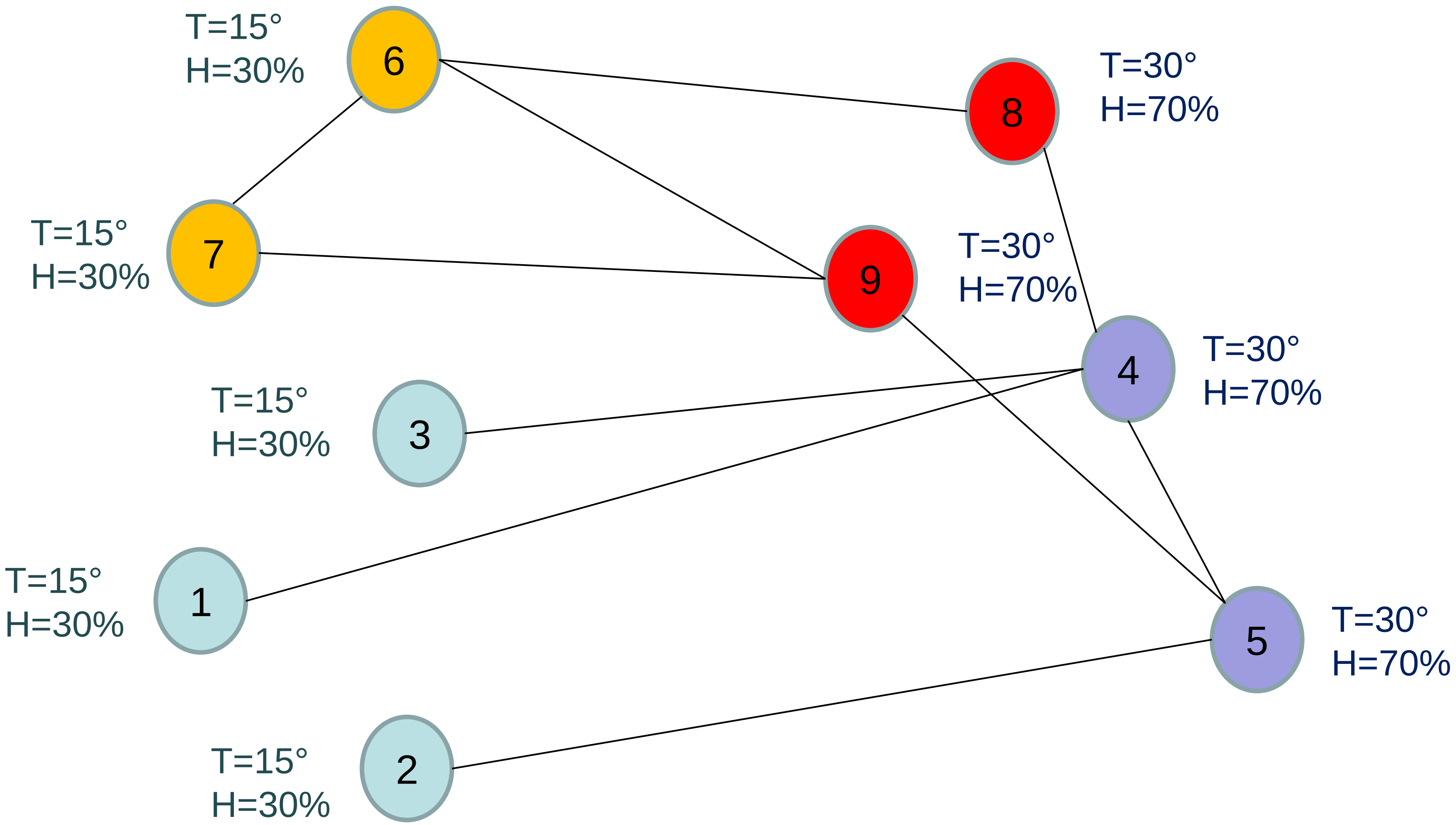


T=15°
H=30%
6
T=30°
H=70%
8
T=15°
H=30%
7
9
T=30°
H=70%
4
T=30°
H=70%
T=15°
H=30%
3
T=15°
H=30%
1
5
T=30°
H=70%
T=15°
H=30%
2

Example


T=15°
H=30%
6
T=30°
H=70%
8
T=15°
H=30%
7
9
T=30°
H=70%
4
T=30°
H=70%
T=15°
H=30%
3
T=15°
H=30%
1
5
T=30°
H=70%
T=15°
H=30%
2

Example



T=15°
H=30%
6
T=30°
H=70%
8
T=15°
H=30%
7
9
T=30°
H=70%
4
T=30°
H=70%
T=15°
H=30%
3
T=15°
H=30%
1
5
T=30°
H=70%
T=15°
H=30%
2

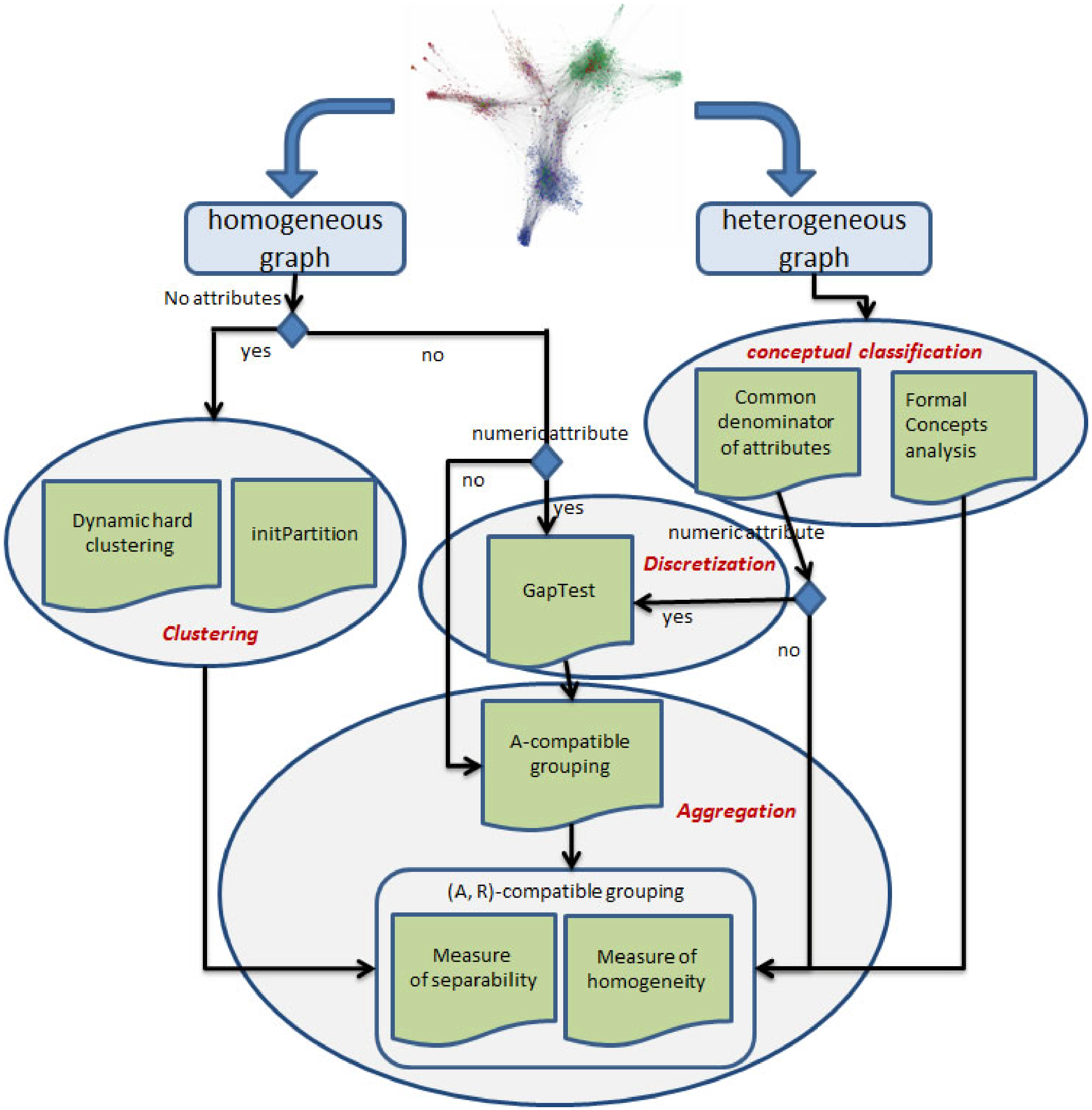
Limits of SNAP

- Aggregation is very rigid in terms of attributes : Cartesian product of all modalities.
- Ineffective with the presence of a large number of attributes or / and numerical attributes.

  ⇨ increases the number of groups with small size
- K-SNAP relaxes the homogeneity requirement and allows users to control the size of the summaries;
- In many applications, the graphs can be heterogeneous, i.e., objects are not represented by the same list of attributes; K-SNAP cannot be applied on heterogeneous graphs

- We proposed:
  - A measure of homogeneity and separability
  - Extensions of the algorithm for heterogeneous graphs

Architecture of our proposal



Proposition of a new evaluation measure: homogeneity measure using a distance

- For a given partition $P = (C_1, C_2 .., C_n)$, this measure Δ locally evaluates the heterogeneity of each class C_i for a relation R_j and determines the less dense to be spitted.

- We use the contingency table (or association table) defined for each pair (m, n) of nodes for a given relationship defined as follows:

	Object n	
Object m	a	b
	c	d

where $a = |N_{R_t}(m) \cap N_{R_t}(n)|, b = |N_{R_t}(m)| - a, c = |N_{R_t}(n)| - a,$

$$N_{R_t}(v) = \{w \in V \mid (u, w) \in E_i\} \cup \{v\}$$

Example

	A2	
A1	1	3
	1	

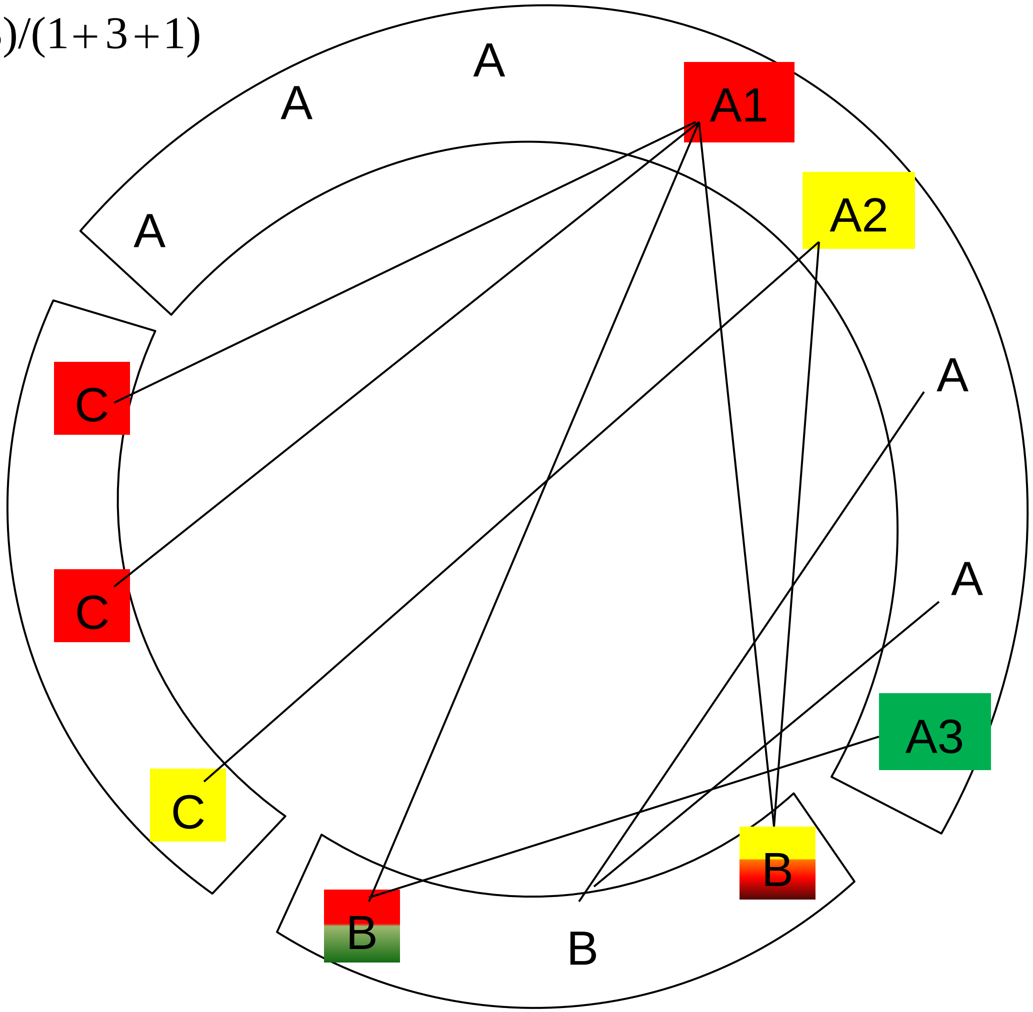
$$d^{j}(A1, A2) = (1+3)/(1+3+1)$$

	A3	
A1	1	3
	0	

$$d^{j}(A1, A3) = 3/4$$

	A3	
A2	0	1
	1	

$$d^{j}(A1, A3) = 1$$



Selection Step

- Starting from a A-compatible grouping, our procedure is to look at each iteration the relationship R_{j^*} and the group C_{i^*} to divide that maximize the evaluation measure δ_i^j up that the cardinal of the partition is equal to K.

- The measure is defined as follows, using the Jaccard distance:

$$\Delta(P) = \sum_{R_j \in R} \Delta_j(P) = \sum_{R_j \in R} \sum_{1 \le i \le |P|} \delta_i^j$$

$$avec \qquad \delta_i^j = \sum_{m \in C_i} \sum_{n \in C_i |} d^j(m,n)$$

$$with \qquad d^j(m,n) = (\mathrm{b}+\mathrm{c})/(\mathrm{a}+\mathrm{b}+\mathrm{c}) \quad \text{is the Jaccard distance for a relation } R_j$$

Division step

- Division step is performed as follows:

1. Determine the node v_d called « central node » of the class to split verifying:

$$d = \arg\max_{v \in C_{i^*}} Deg_{R_{j^*}}(v)$$

2. Divide C_{i^*} into two sub-classes according to the following strategy : one contains the neighbour of the central node, and the other one all nodes not connected to the central node.

Measure of homogeneity

- For a given partition $P = (C_1, C_2 .., C_n)$,this measure denoted Δ evaluates the homogeneity of each class C_i and determines the less homogeneous to be splitted .

Principle:

- For each relation R_j, we denote:

$$IA^j(c_i) = \frac{1}{|c_i|^\alpha} \sum_{v \in c_i} Deg_j(v)$$

Intra-group criterion

Density of group C_i / relation $_j$

$$IE^j(c_i) = \frac{1}{|E_j|} \sum_{v \in V - c_i} Deg_j(v)$$

Inter-group criterion

Density and connectivity

$$\triangle = \sum_{i=1}^{|p|} \sum_{E_j \in R} \delta_i^j = \sum_{i=1}^{|p|} \sum_{E_j \in R} \frac{IA^j(c_i)}{IE^j(c_i)}$$

Homogeneity measure

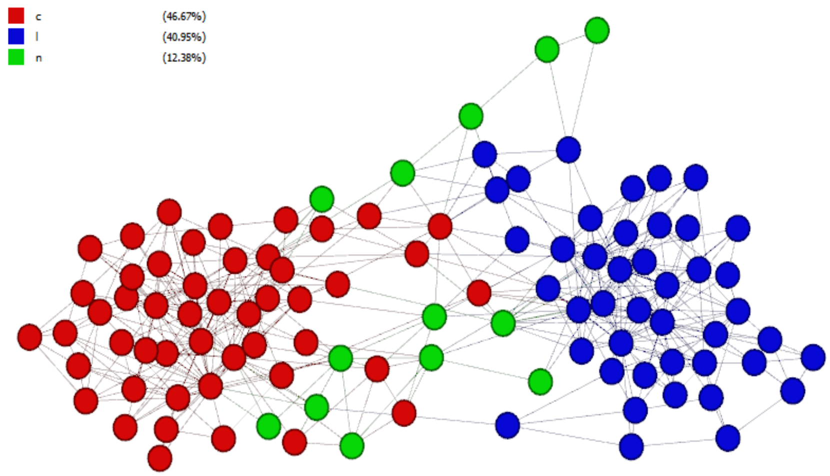
with $Deg_j(v)$: is the degree of vertex v according to the relationship R_j et α ∊ [1, 2]

The most homogeneous group with few relations outside is not divided

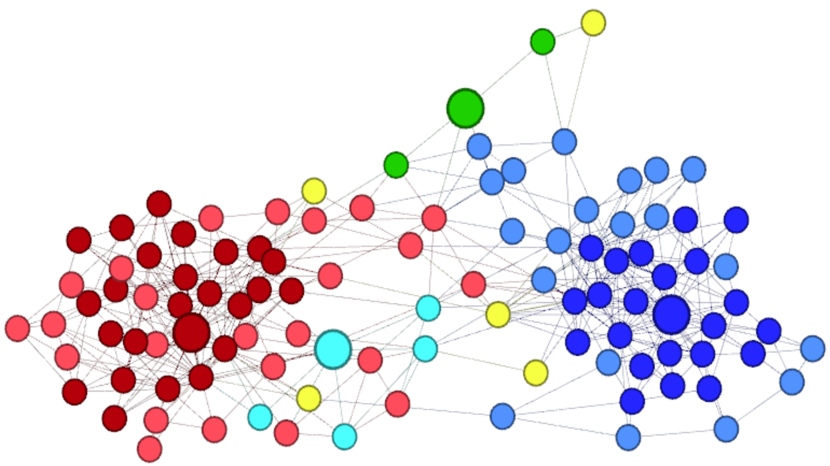
Application: network of books about American politics

Elaborated by Mark Newman [Newman, 2003].
Contains105 nodes and 441 vertex.

Initial partition
A-compatible

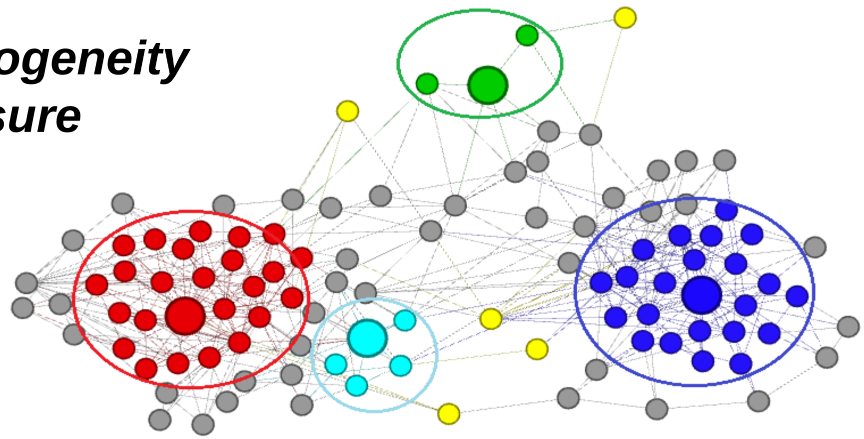


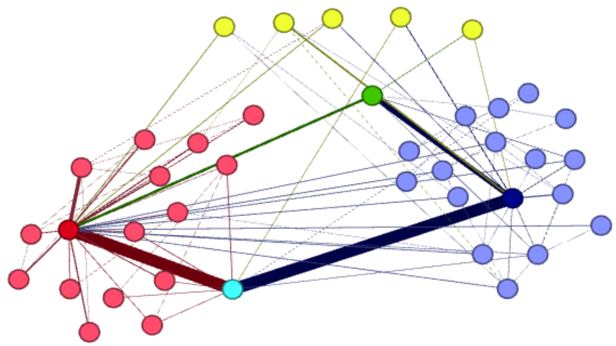
Final partition
After 4 iterations (7 groups)



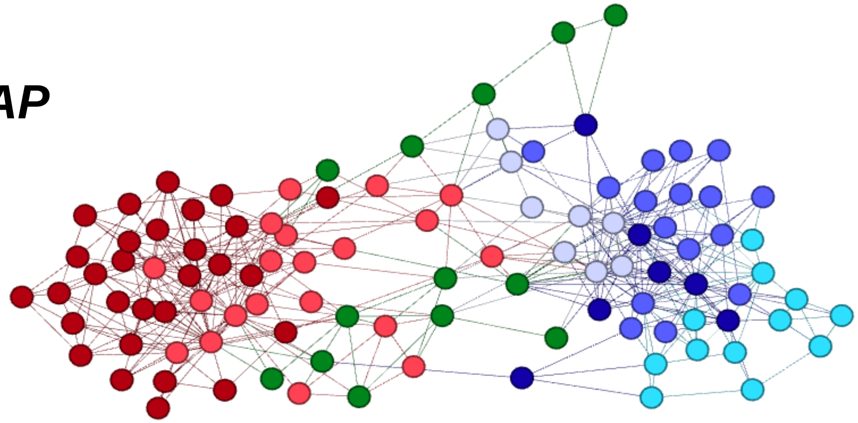
Application: network of books about American politics

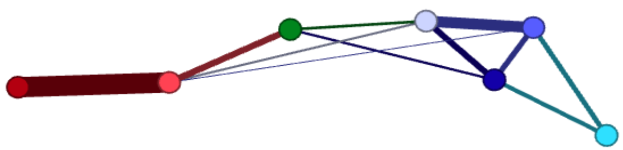
Homogeneity measure





K-SNAP (K=7)





Conclusion: tools developed

- SPIDER-Graph:
  - a model for representing heterogeneous graphs containing complex nodes
  - A set of transformation patterns – matching with a semantic layer
- Merging of graphs extracted from RDBMS using Hadoop/HDFS
- Queries
  - Visual query language
  - Keywords search (by selecting nodes or ontology concepts)
- Summarization – Aggregation (K-SNAP)
  - Customized summaries
  - Meaningful summaries for real applications,
  - Efficient and scalable for very large graphs

Readings

1. Tian, Y., R. A. Hankins, et J. M. Patel (2008). Efficient aggregation for graph summarization. In SIGMOD ’08
2. Rania Soussi, Etienne Cuvelier, Marie-Aude Aufaure, Amine Louati and Yves Lechevallier (2012) DB2SNA: an All-in-one Tool for Extraction and Aggregation of underlying Social Networks from Relational Databases. In: The influence of technology on social network analysis and Mining, Tansel Ozyer *et al.* (eds.), Springer, Nov. 2012.
3. Rania Soussi and Marie-Aude Aufaure (2012) Enterprise Ontology Learning for Heterogeneous Graphs Extraction, 2nd International Conference on Model & Data Engineering (MEDI'2012) October 3 - 5 2012. Poitiers, France.
4. J Dean & S Ghemawat : *MapReduce: Simplified Data Processing on Large Clusters*, 2004
5. Serge Abiteboul, Ioana Manolescu, Philippe Rigaux, Marie-Christine Rousset, Pierre Senellart: Web Data Management, Cambridge University Press, 2011
6. Roberto Zicari: Big Data: Challenges and Opportunities, http://www.odbms.org/downloads.aspx#adm_ar

Team : Etienne Cuvelier (postdoc, ECP), Rania Soussi (PhD, ECP), Amine Louati (intern, ECP&INRIA), Bruno Almeida Pimentel (intern INRIA), Alessandra Anyzewski (intern INRIA), kieu van cuong (intern ECP), Baptiste Thillay (intern, ECP).

In collaboration with Yves Lechevallier (INRIA Axis)