First European Business Intelligence Summer School (eBISS 2011)

July 3 - 8, 2011 Paris, France

Graphs for Business Intelligence

Marie-Aude AUFAURE

Ecole Centrale Paris

SAP Academic Chair in Business Intelligence

Agenda

- Context/motivation for using graph structures
- Use cases
- Variety of graphs
- Application: extracting and aggregating graphs from relational databases

CONTEXT / MOTIVATION

Data Everywhere

Information overload

More data stored in the Web in 2010 than all data produced since the begining of humanity until 2003!

By 2013, the amount of trafic flowing over the internet annually will reach 667 Exabytes (CISCO)

Big Data

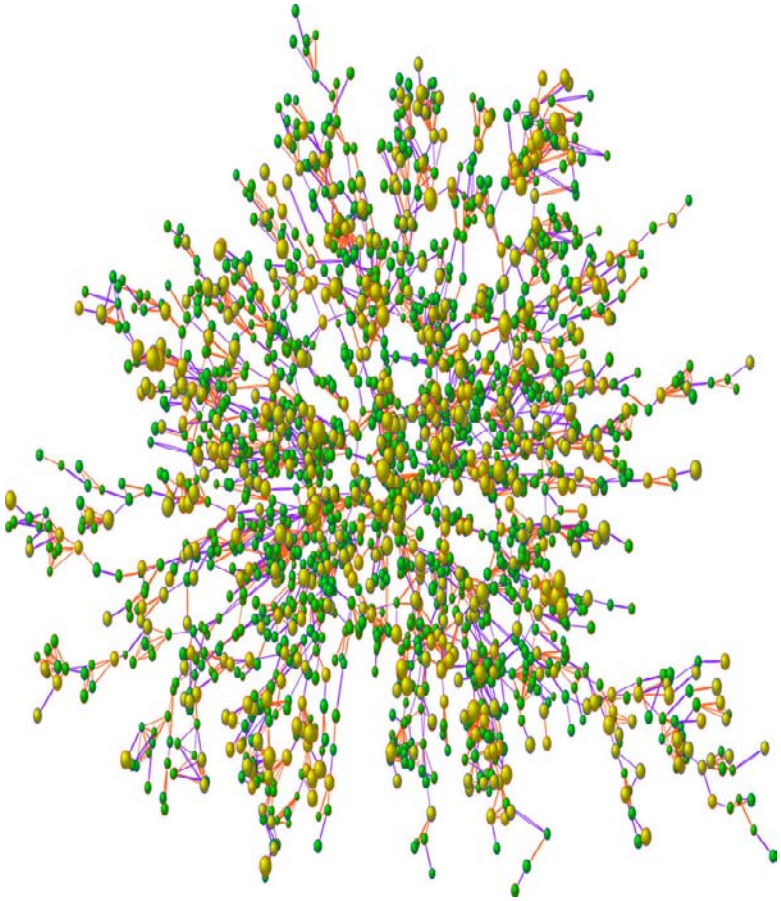
- Data are mainly **unstructured**
  - **80% of data manipulated in an enterprise are unstructured**
- Data are produced in **real time** and **distributed**
- Data come from **heterogeneous** sources in an unpredictable way
  - Mobile phone, sensors, computers, TV, etc.

⇒ The world contains an unimaginably vast amount of digital information which is getting even vaster even more rapidly

- Data-centered economy

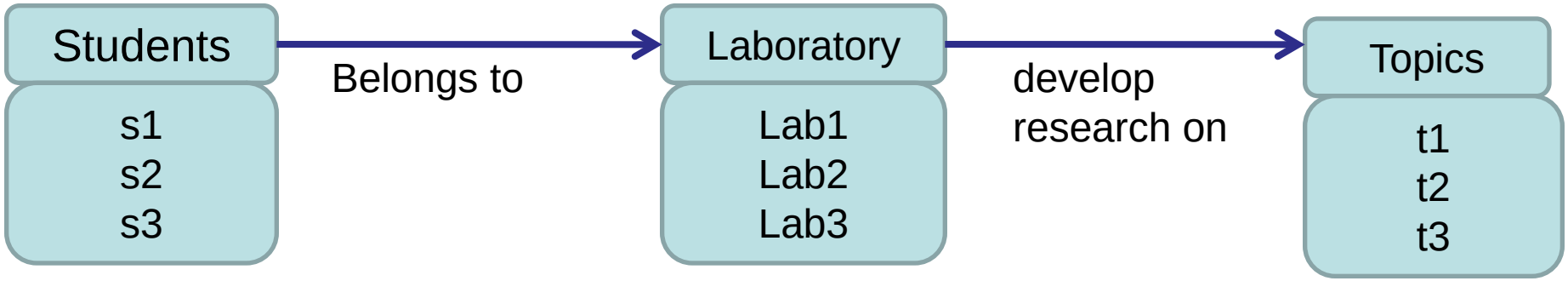
⇒ Big Data phenomena is considered as the **main computer science challenge** for the next decade

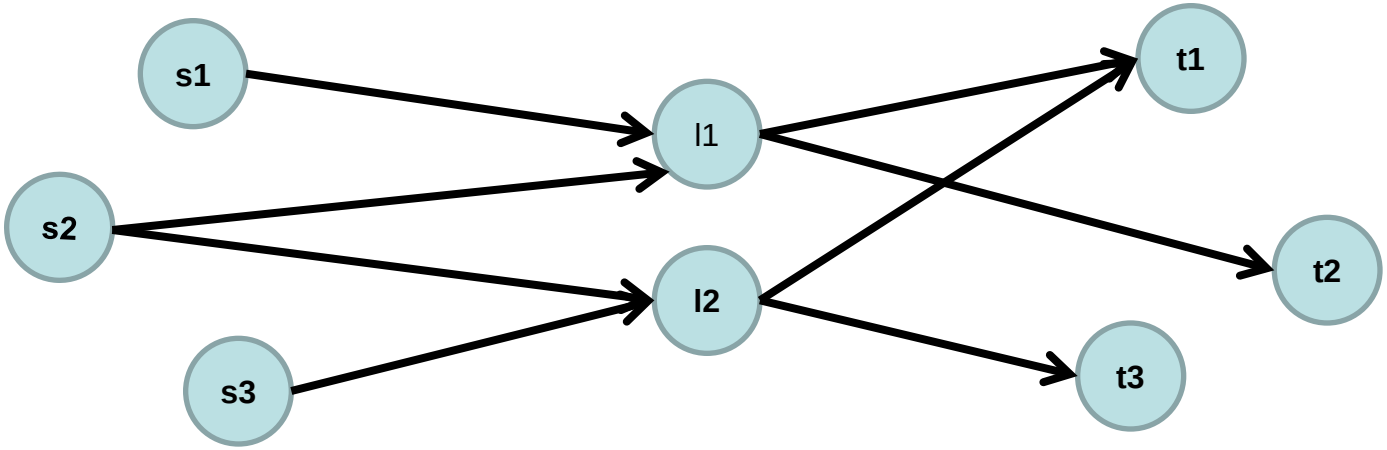
Graphs everywhere



-Social networks
-Web
-Enterprise databases
-Biology
-Etc.

Graphs everywhere





Graphs

Relational databases

Simple management of structured, semi-structured and unstructured information

XML

Web

From mathematical graph structure to graph databases

Graphs: what can we do with?

- Traversing linked information, finding shortest path, doing (semantic) partition
- Recommendation and discovery of potentially interesting linked information

Graphs: what can we do with?

- Exploit the graph structure of large repositories
  - Web environment
  - Digital documents repositories
  - Databases with metadata

USE CASES

Social networks, recommendation (enterprise and web contexts)

Social networks

- A social network is a social structure made up of individuals (or organizations) called "nodes", which are tied (connected) by one or more specific types of interdependency, such as friendship, kinship, co-authorship, common interest, etc.

- Social networks are mainly online ones
  - In 2010, Facebook: 400M users, Twitter: 25M, etc.

- Enterprise social networks:
  - 59% without any strategy for integrating a social network (Coleman Parkes Research - 2008)
  - But 60% consider that social networks should be very useful for collaboration

Why enterprise social networks?

- A technology for internal communication, information sharing and collaboration
- A technology for information communication towards clients
  - Vote for the best product,
  - Understand the clients needs
- A technology for watching the gossip
  - E-reputation, opinion mining
- A technology for creating collective intelligence
  - Collaborative common knowledge
  - Wikis and blogs associated to social networks

Social networks for enterprises, public administrations: scenarios

- Centrality, importance and influence of actors
  - Betweeness: mediators and isolated actors
  - Closeness : proximity
- Identification ok key actors
  - Entry points in the network
  - Sensible points in the network
  - Probability estimation of a relation
- Groups
  - Groups formation (from relations)
  - Cohesion, density, stability
  - Groups similarity

Centrality definition

- Being central is:
  - Be the source or destination of numerous relations: « degree centrality »
  - Be close to many actors: « closeness »
  - Be central for many connections: « betweeness »

Scenarios examples: (Human resources)

- Which persons have the biggest influence
  - Interests:
    - Transmit good practices
    - Improve social aspects
- Which groups are the most central ones?
  - Interest:
    - Find groups with a good communication, which are important cohesion elements
- How can I constitute and efficient team?
  - Interests:
    - Enabling innovation
    - Group cohesion
- Search for atypical elements:
  - Which persons are not very well integrated, and which group is the best one for them?

Scenarios examples: (Human resources)

- Communication between entities
  - Is it good?
  - How can it be improved?

- Which groups are similar?
  - Analyze groups with “good” properties
  - Apply observations on a group to another similar one
  - Explain a behavior with a structure

- Monitor and strengthen links with different partners

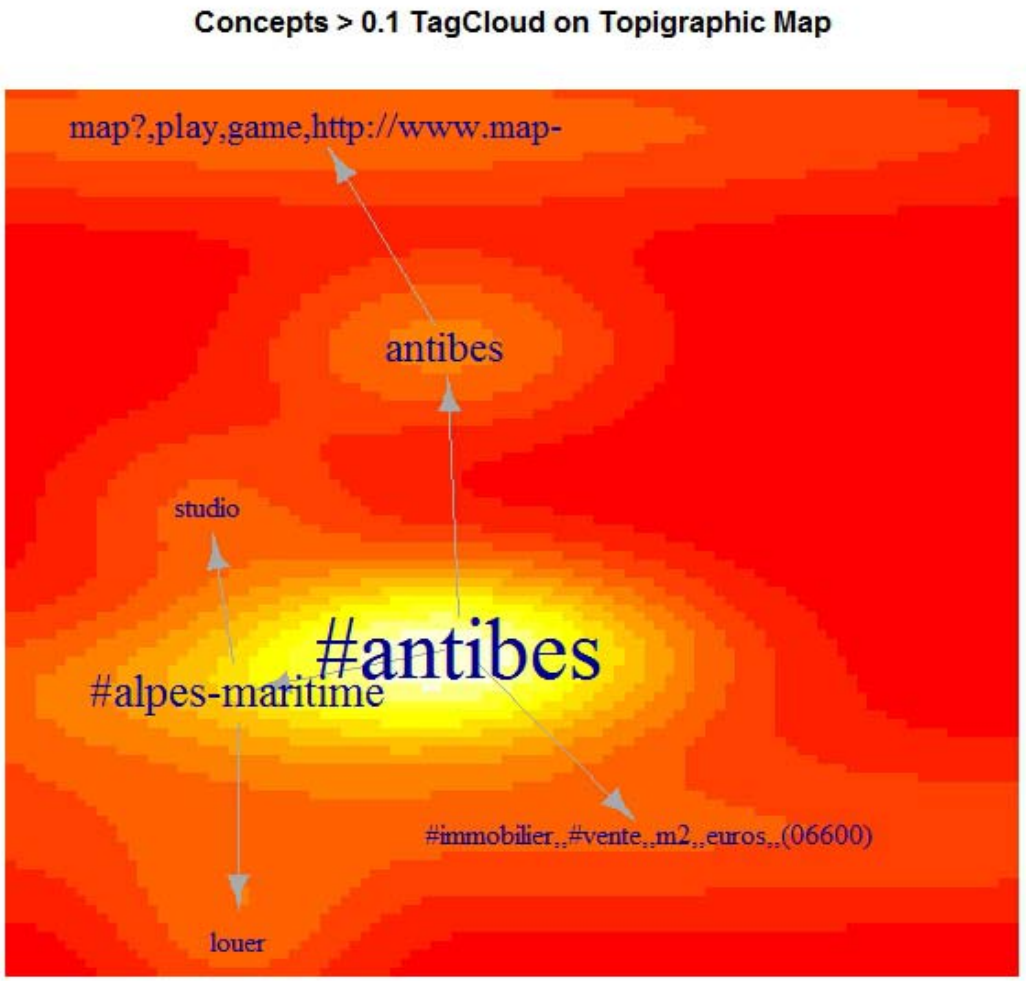
Scenarios examples: (Individuals)

- Networking
  - From whom should I be closer?
  - Identify important relation by studying relations of other persons
- Build a collaborative social network
  - Which relations could be interesting to set up?
- Learn from the experience of others
- Find an expert in the enterprise
- Develop my own profile for finding a job

Social networks for public administrations

- Public administrations need social networks:
  - As enterprises:
    - To analyze internal networks (projects, organization…)
    - To analyze external networks (suppliers, clients, partners…)
  - As an interface for citizens:
    - To be well-understood by citizens (who does what)
    - To understand citizens (who says what)
- Scenarios examples:
  - Need to look over the organizational structure (employees, departments, transversal projects) and identify costs
  - Need for citizens to understand the impact of public politics (offered services, available resources for each district of the city, which projects are the most relevant, citizens complains)
  - Opinion analysis from external social networks (Twitter for example)

Social networks for public administrations

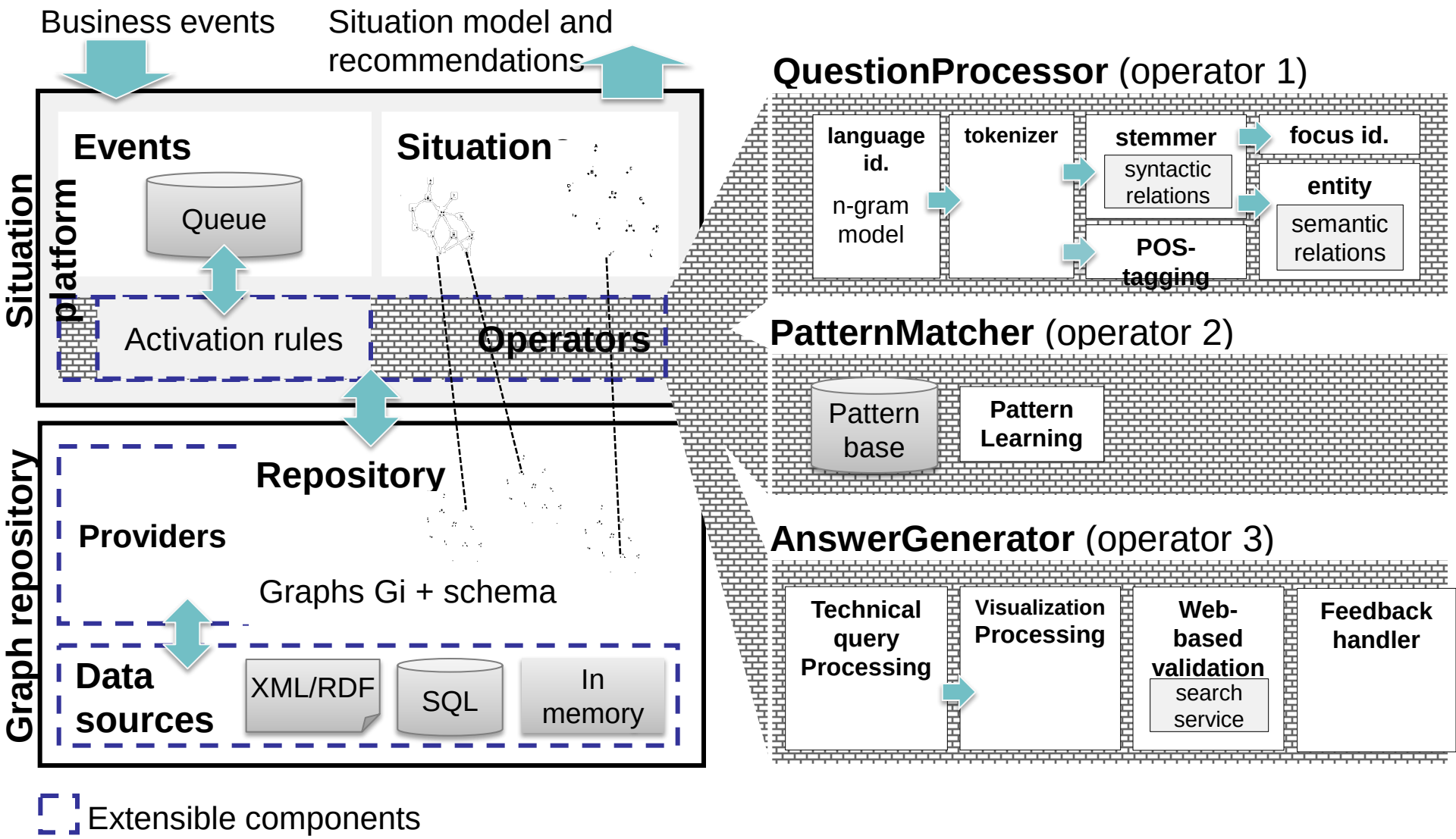


Graphs for Recommendation

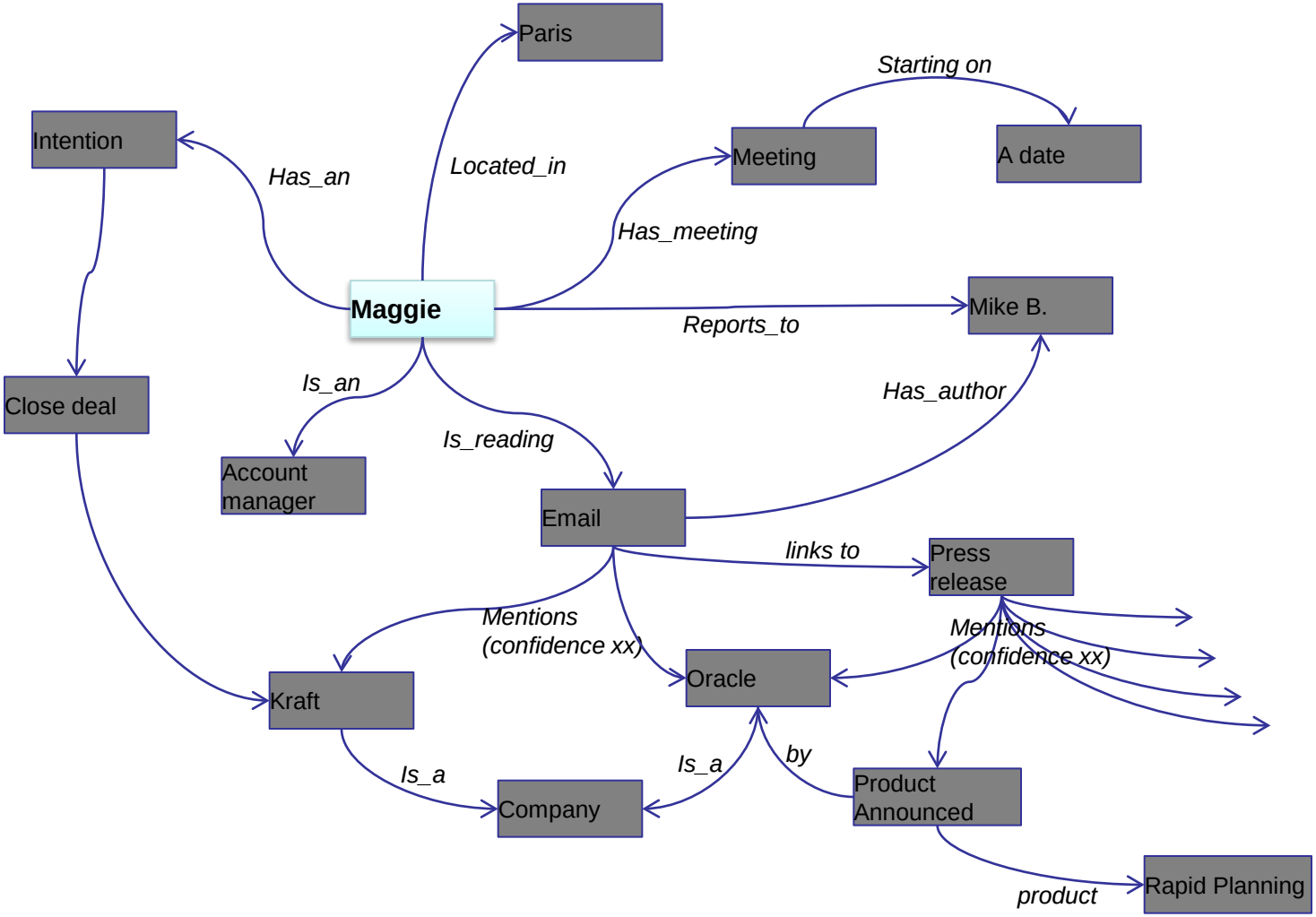
- In an enterprise context:
  - Recommendation of documents, experts
  - Help for decision-making
  - Situational applications
- In a web context:
  - Market-based applications
  - Social search
  - Web link analysis (based on the web graph structure)
    - PageRank, HITS

Recommendation in an enterprise context

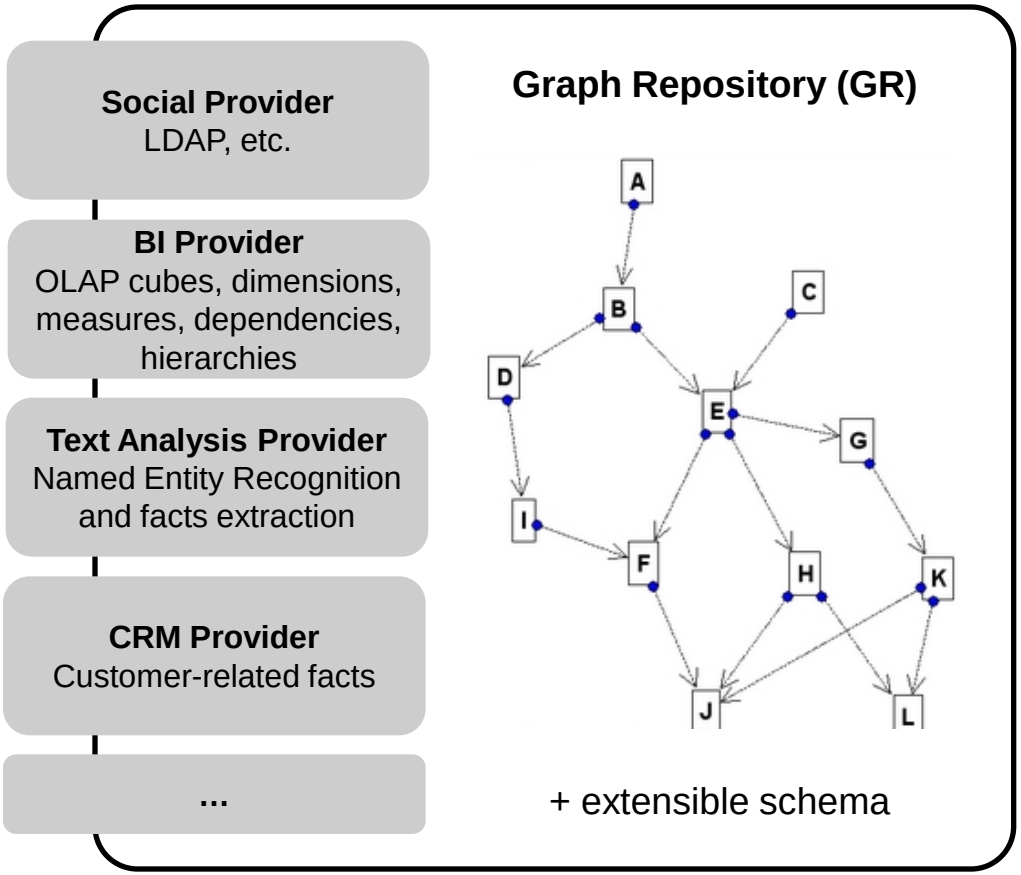
Thollot, Kuchmann & Aufaure, 2010



Modeling the situation as a user-centered graph



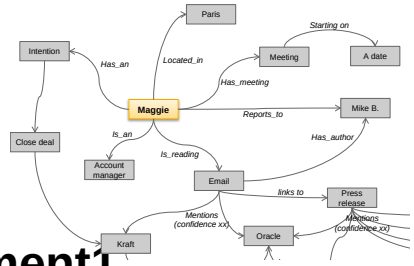
nodes and relations are described in a graph repository



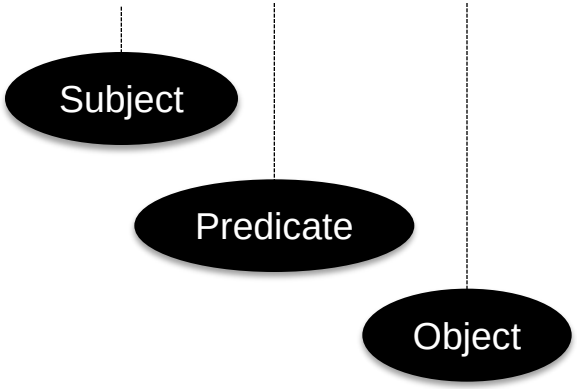
Providers feed the GR

Relations between nodes of these graphs represent static or time- dependent facts/assertions

- Homogeneous statements
  - Extended triples: (Subject Predicate Object + Metadata)
    - *situation statements*
    - Time dependency, uncertainty management, security



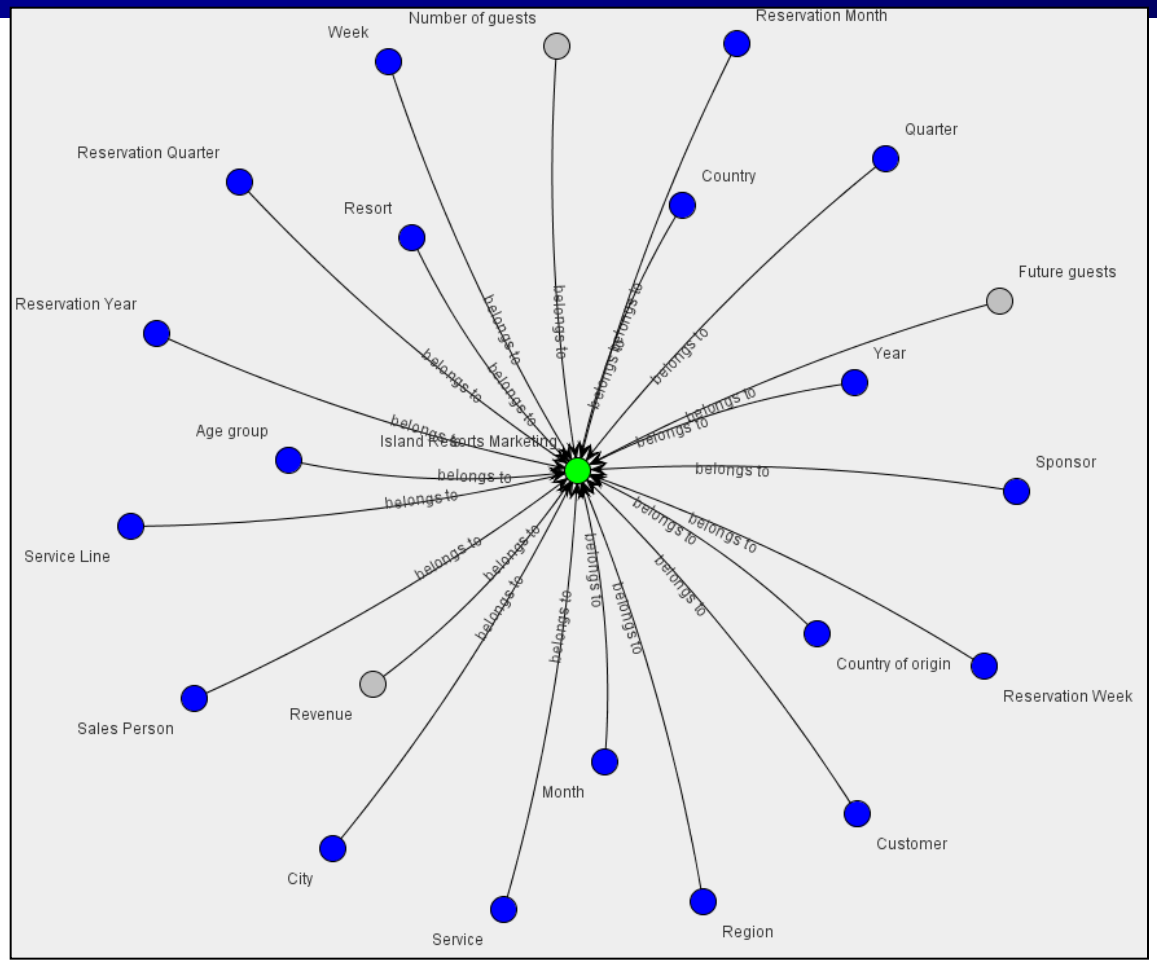
Maggie *read* document1



Maggie *read* document1

+ origin: email client (plugin)
+ time validity: timestamp + lifetime
+ security: private / public / ACL
+ confidence

Sample graphs – Universe/cube



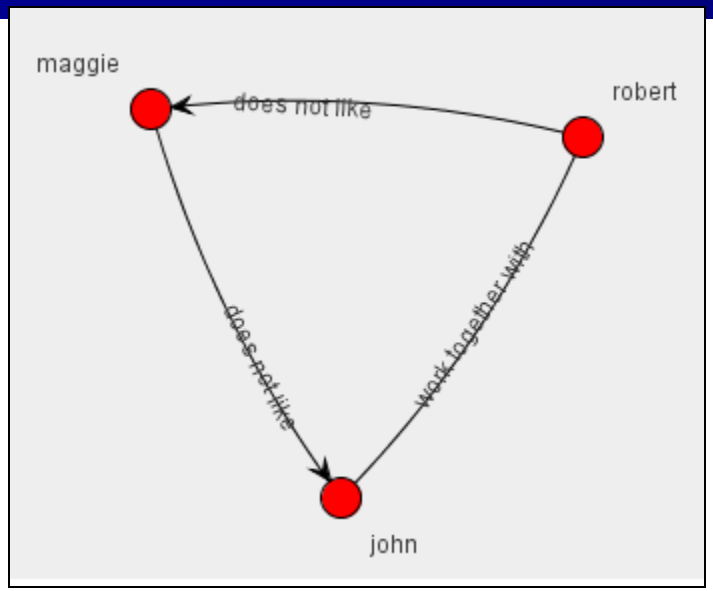
Legend
Green – universe/cube
Blue – dimension
Gray – measure

Information from BI provider.

Gives a graph view of a universe or a cube.

Depending on the underlying source, additional semantics can be pushed to the IM.

Sample graphs – Social network



Legend
Red – user/agent

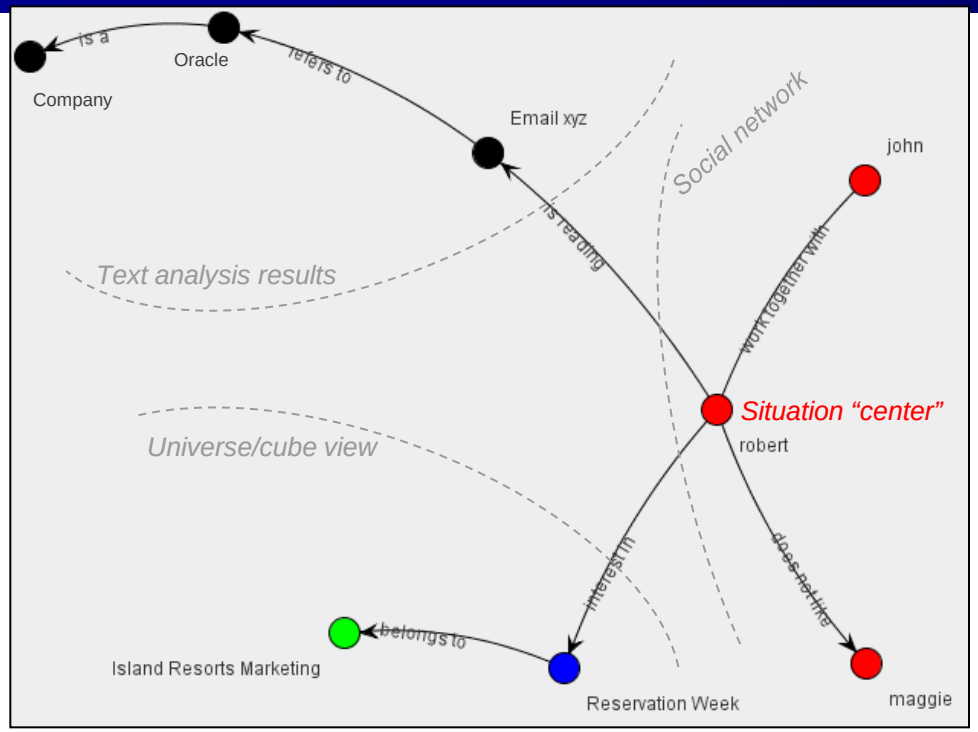
Information from Social provider.

Gives a graph view of users' social network.

The social network can be a very big graph, and returning the full view would be too costly.

Some Information Mesh providers need the ability to provide partial graphs. They should also be able to expand this view on-demand around a given node or relation.

Sample graphs – A user's situation



Legend
Green – universe/cube
Blue – dimension
Gray – measure
Red – user/agent
Black – text analysis-related entity

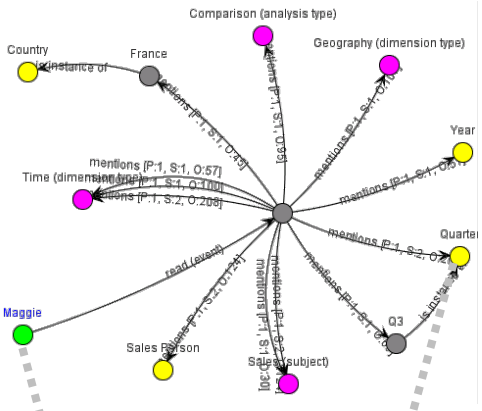
This graph represents Robert's situation. A situation is centered on the concerned user.

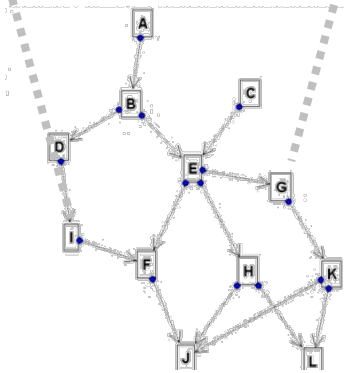
A situation establishes connections between different partial graphs of the Information Mesh (social network, universe view and text analysis results in this example).

In situation graphs, there is a combination of generally true and volatile nodes & relations.
E.g., relations (and nodes) from the universe view are static, while "is reading" is by nature very dynamic/volatile.

Representing a user's situation is key to serve relevant recommendations

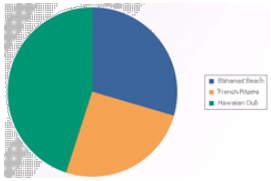
User's situation





Social network,BI,…

Recommendations



BI

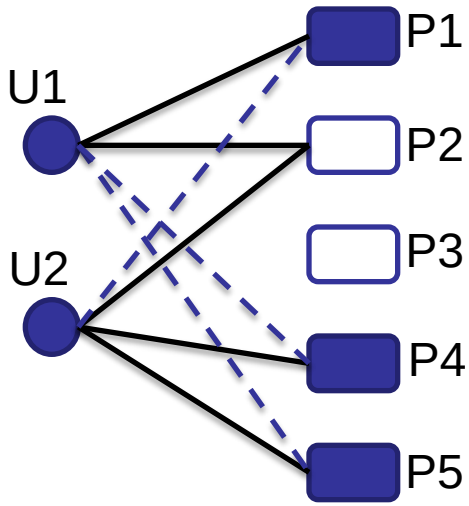
People

Tools & apps

Documents

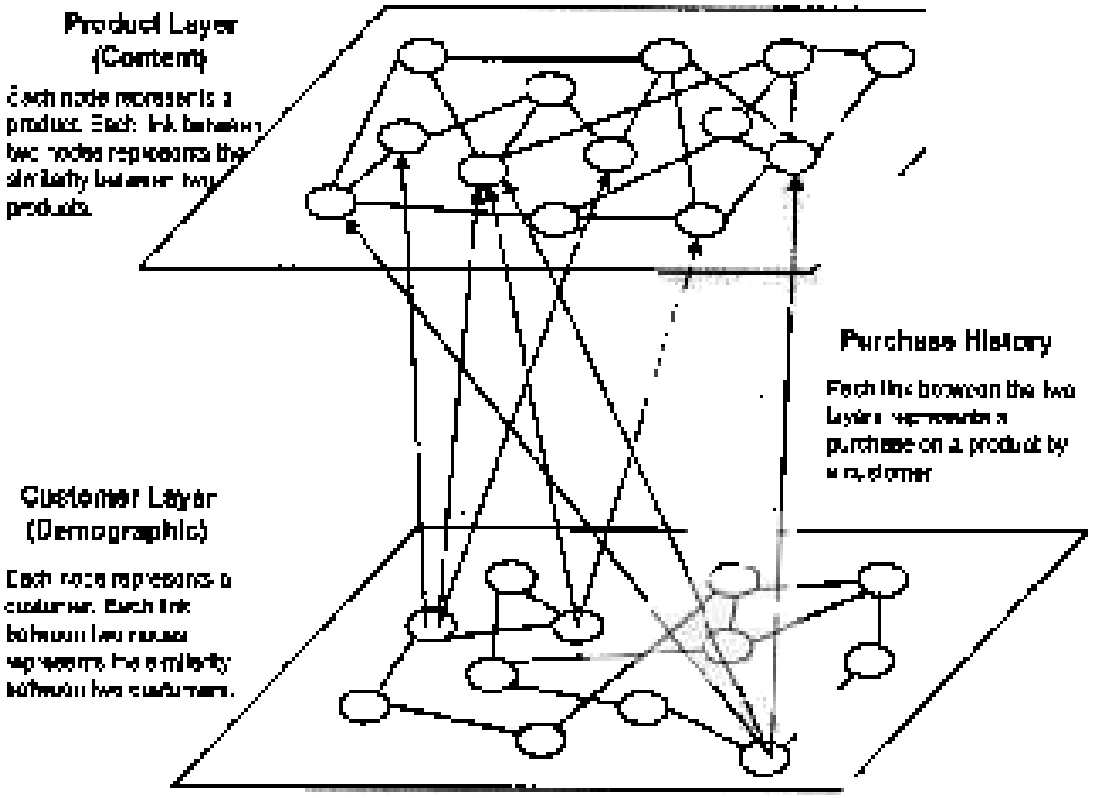
Recommendation in a web context

- Market-based applications
- Collaborative/content filtering
- Social recommendation



Possible recommendations (dashed edges)

Graphs used to model the recommendation problem

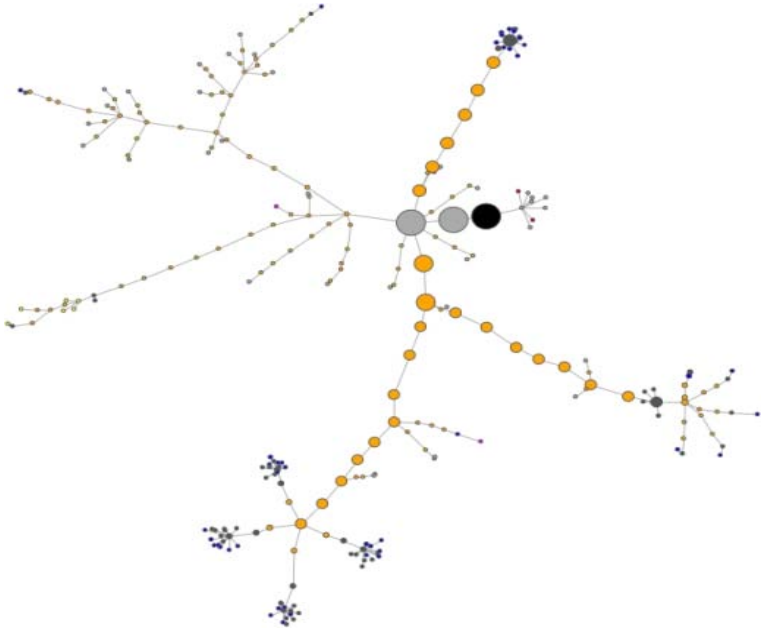


VARIETY OF GRAPHS

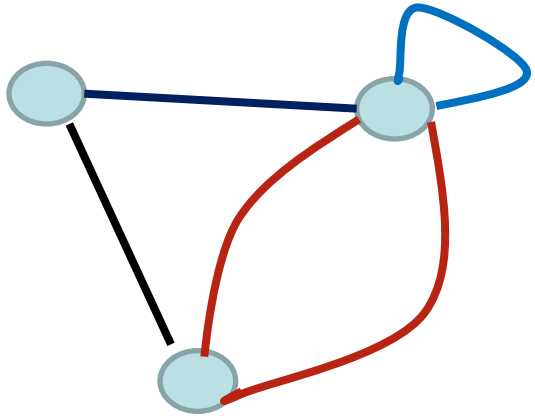
Graphs: Definition

A *graph G = (V, E)* consists of a set *V* of vertices (also called nodes a set *E* of edges where $E \subseteq V \times V$

This definition refers to ***simple*** and ***undirected*** graphs

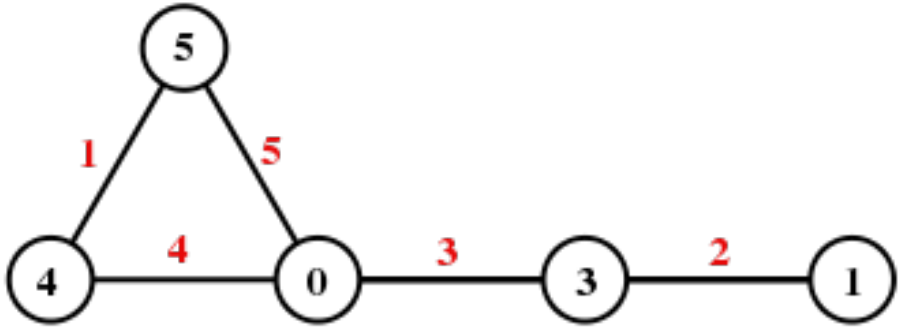


The term ***multigraph*** is generally understood to mean that multiple edges and loops are allowed



Labeled Graphs

Given two alphabets Σ_V and Σ_E a *labeled graph* is a triple $G=(V,E,l_V)$ where V is a finite set of nodes, $E \subseteq V \times \Sigma_E \times V$ is a ternary relation describing the edges (including the labeling of the edges) and $l_V : V \rightarrow \Sigma_V$ is a function describing the labeling of the nodes



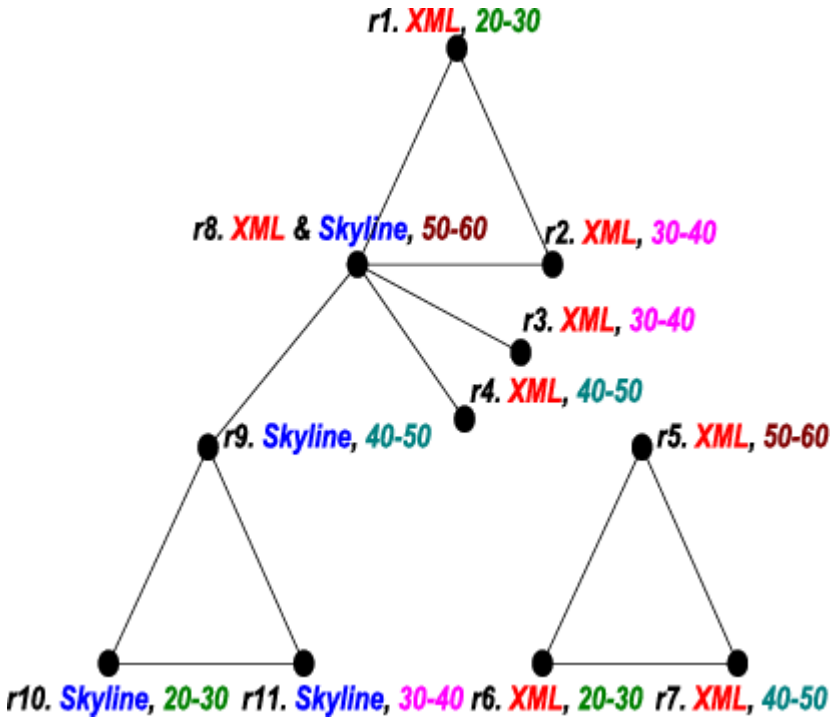
A **graph labeling** is the assignment of labels, traditionally represented by integers, to the edges or vertices, or both, of a graph

Applied to finite undirected simple graphs

Attributed graph

If nodes in a graph have attributes, the graph is an *attributed graph* denoted by *AG = (V, E, α)*, where *α* is a labeling function: $\alpha : V \rightarrow L_N$

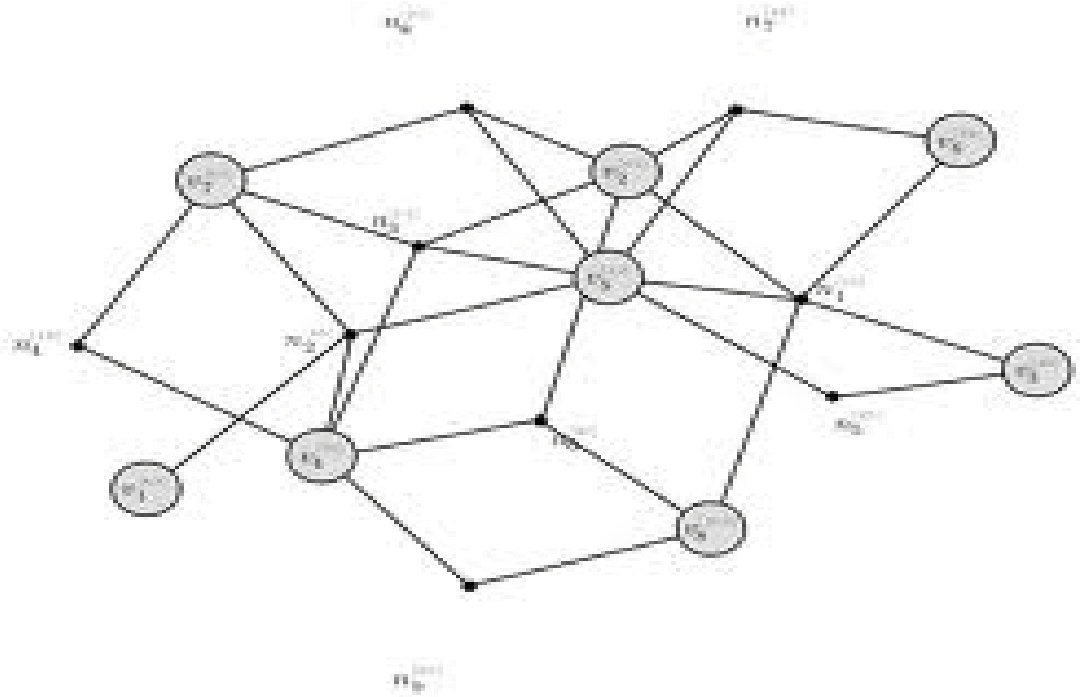
If both nodes and edges in a graph have attributes, the graph is an attributed graph denoted by *AG = (V, E, α, β)* where $\beta : E \rightarrow L$ is an edge labeling function.



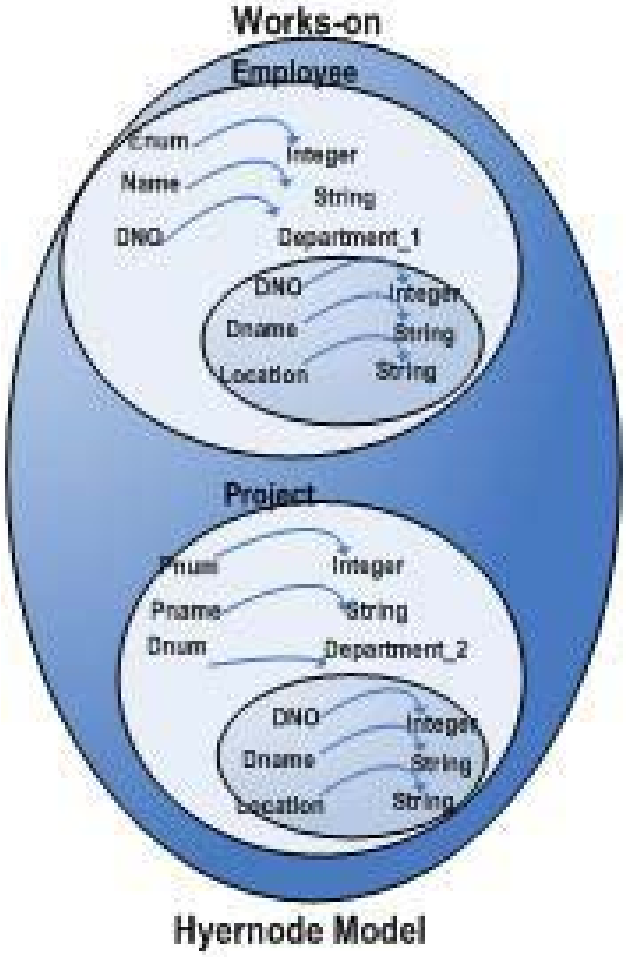
Coauthor network with 2 attributes: « Topic » and « Age »
Zhou & al, ICDM 2010

Hypergraphs

An *hypergraph G* is a tuple *(V, E, μ)*, where *V* is a finite set of nodes, *E* is a finite set of edges, $\mu : E \rightarrow V^*$ is a connection function where *V** means multiple nodes (an edge can connect any number of nodes).



Hypernodes

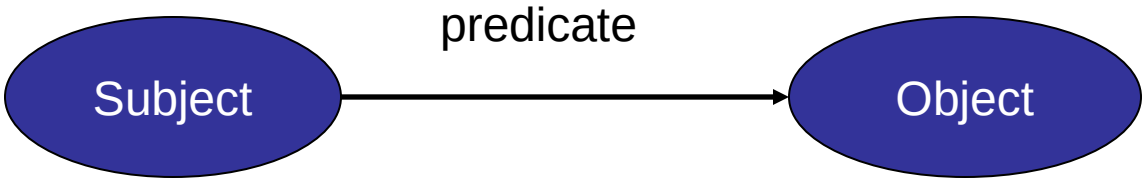


Hyernode Model

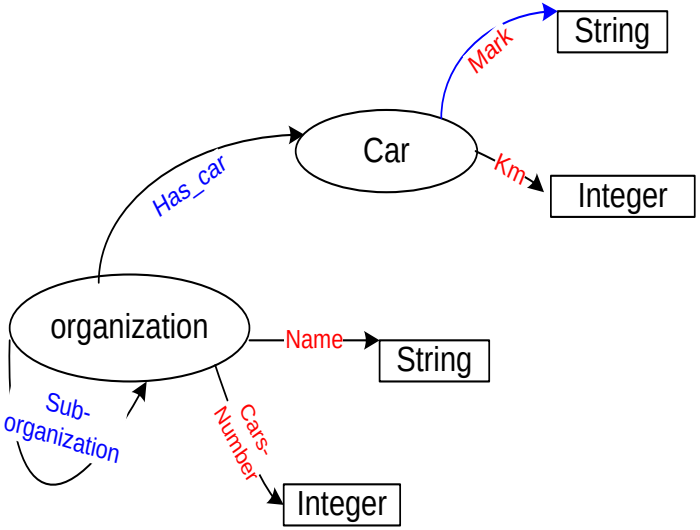
A *hypernode* has the form *H= (N, E)*
where
$H \in L$ is termed the label of the hypernode and
(N, E) is a digraph such that $N \subset (P \cup L)$ (*P* is the set of primitive nodes).

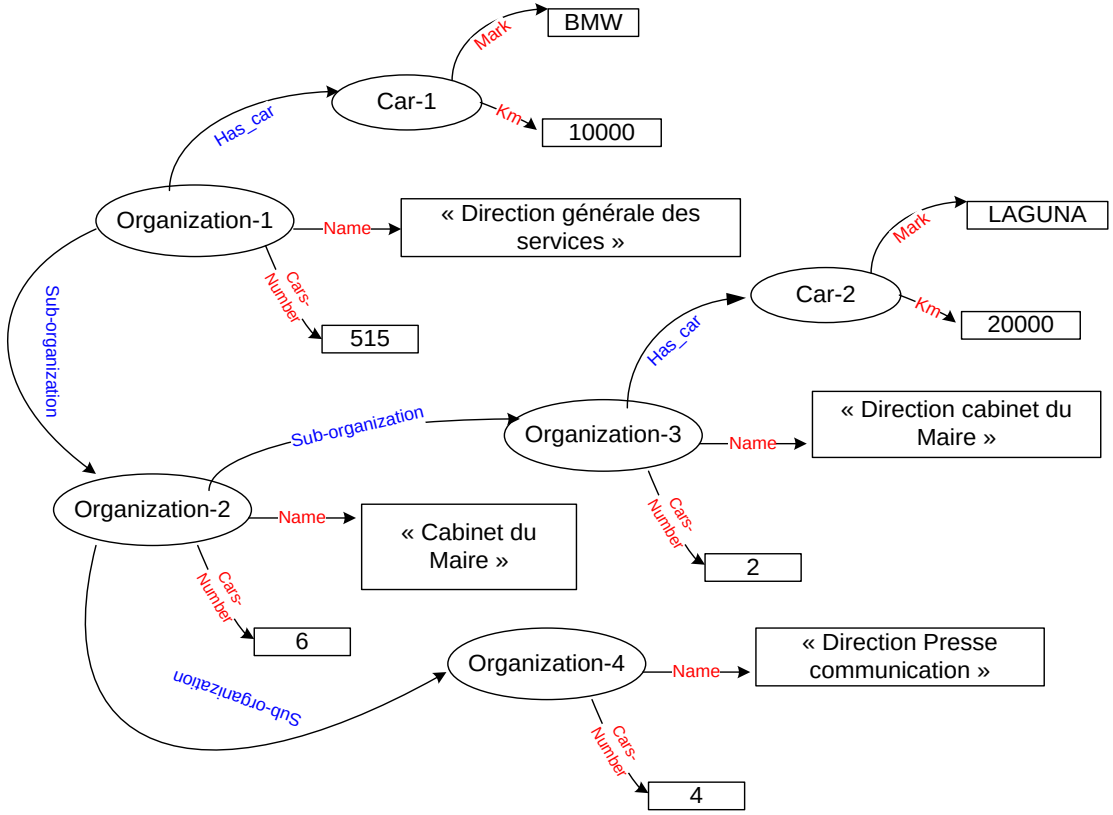
(N,E) is termed the digraph (directed graph) of the hypernode

Graphs for the Web: RDF



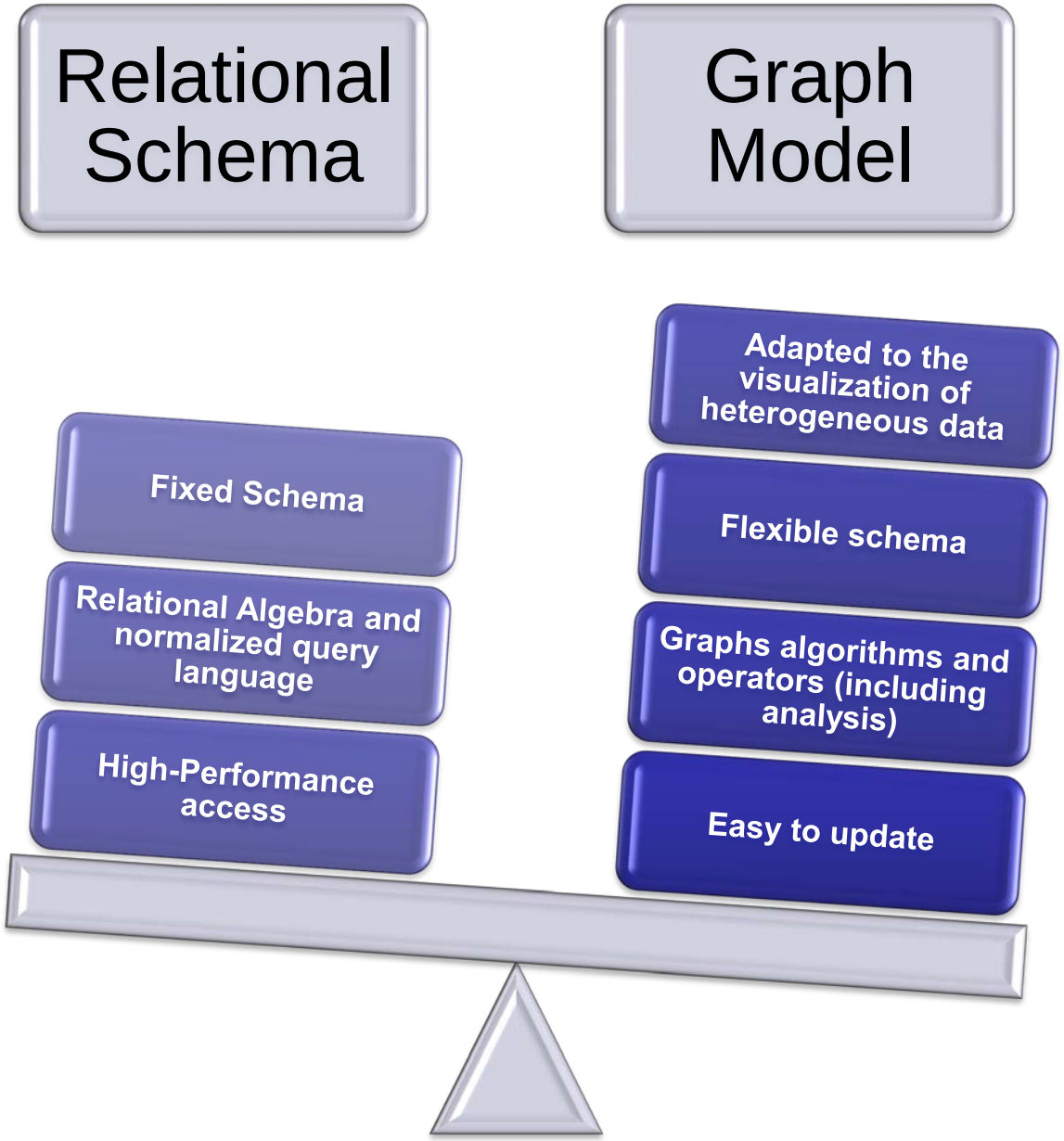
Exemple





Graph Databases and Models

- A **graph database** is a kind of NoSQL database (Not Only SQL) that uses graph structures with nodes, edges, and properties to represent and store information. General graph databases that can store any graph.
  - http://en.wikipedia.org/wiki/Graph_database
- Benefits:
  - No implicit model like within RDBMS
  - Explicit representation of the semantics
  - High-performance traversing (a node space)
- Existing graph databases:
  - Cassandra
  - Neo4j
  - Allegrograph
  - etc.


Relational Schema
Graph Model
Fixed Schema
Relational Algebra and normalized query language
High-Performance access
Adapted to the visualization of heterogeneous data
Flexible schema
Graphs algorithms and operators (including analysis)
Easy to update

Graph models

- Data and/or schema are represented by graphs (directed graphs in most cases)
  - Schema and Instances can be clearly distinguished
- Data manipulation:
  - Graphs transformation (graph rewriting)
  - Operations on graphs (paths, neighborhoods, subgraphs, graph patterns, connectivity, graphs statistics like centrality)
- Integrity constraints enforce data consistency
  - Schema-instance consistency, identity and referential integrity, functional and inclusion dependencies
  - Examples: labels with unique names, constraints on nodes, etc.

Graph Databases Models

- Model information whose logical structure is a graph
- Emphasize the interconnectivity between the data
- **two categories:**
  - Models based on simple nodes
    - directed or undirected graph with simple nodes and edges
    - Not adapted for the representation of complex entities
  - Models based on Hypernodes
    - Entities and relations based on hypernodes and hypergraphs
    - Attributes can be multivaluated
    - Appropriate to represent complex and dynamic objects

Data structures: representation of entities

- All graph db-models have their formal foundation, variations of the basic mathematic structure of a graph
  - Directed or undirected graphs
  - Labeled or unlabeled edges an nodes
  - Hypergraphs and hypernodes
- A (basic) schema graph defines:
  - *entity types* represented as nodes labeled with zero or one type name,
  - *primitive entities* represented as nodes labeled with basic types
  - *relations* represented as edges labeled with relation name
- Extension in other models to complex structures allowing the representation of complex objects

Data structures: representation of relations

- Types of relations supported by graph-db models:
  - *Attributes*: represent a property directly linked to an entity (represented by labeled edges)
  - *Neighborhood relations*: graph structure provides simple support and visualization of such relations
  - *Standard abstractions*: aggregation (part-of relation), composition (the opposite relation), association (n-ary relations), groupings or sets.
  - *Derivation and inheritance*: IS-A relations
  - *Nested relations*: recursively specified relations defined by nested objets. Supported by using hypernode or hypergraphs structures

Graph data models

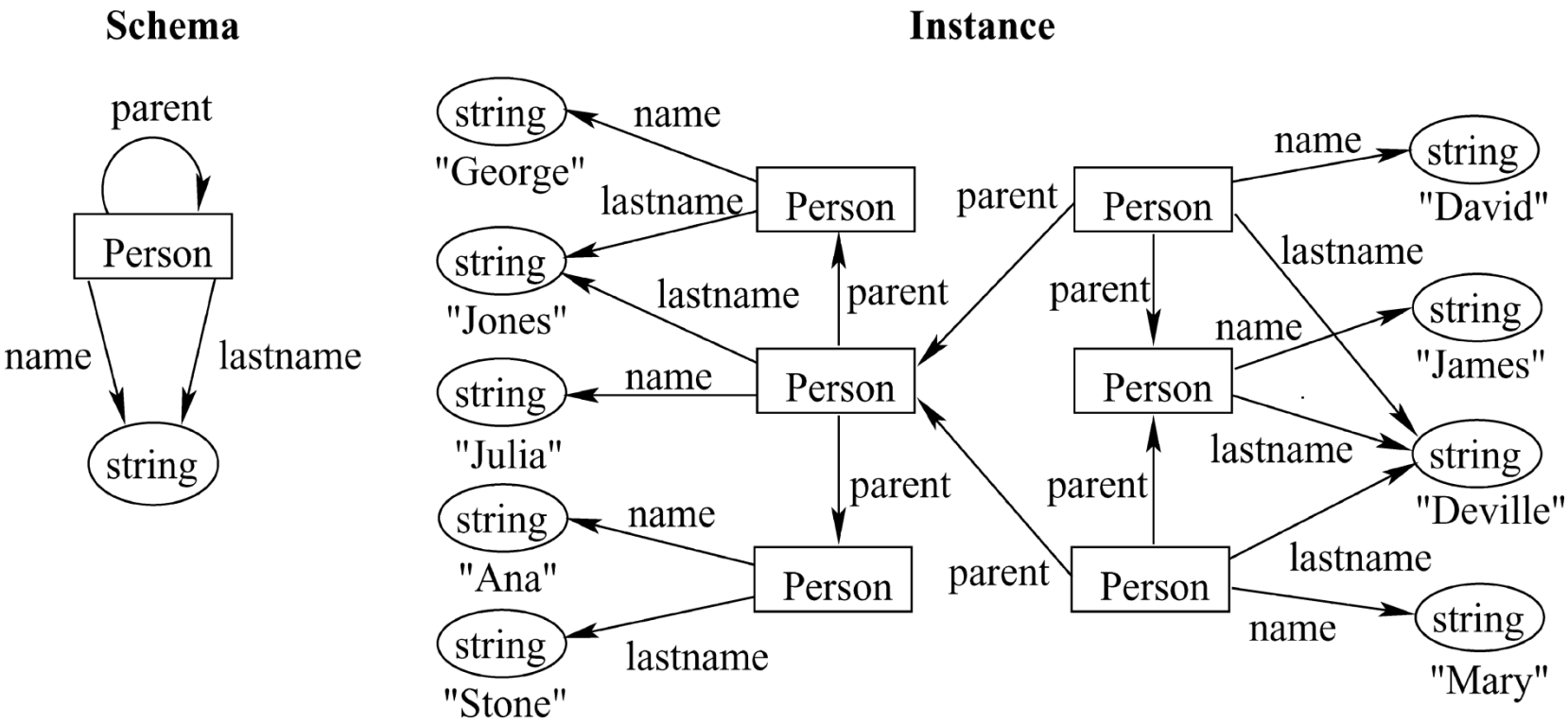
Model based on simple nodes

GOOD [Gyssens *et al*, 90],
GMOD [Andries *et al*, 93],
G-Log [Paredaens *et al*, 95],
PaMaL[Gemis, *et al*, 93]
GDM[Hidders,2002],
LDM [Kuper et Vardi,93],

Model based on hypernodes

Hypernode model [Levene *et al*,90],
GGL[Graves *et al*,90],
GROOVY [Levene *et al*,91]

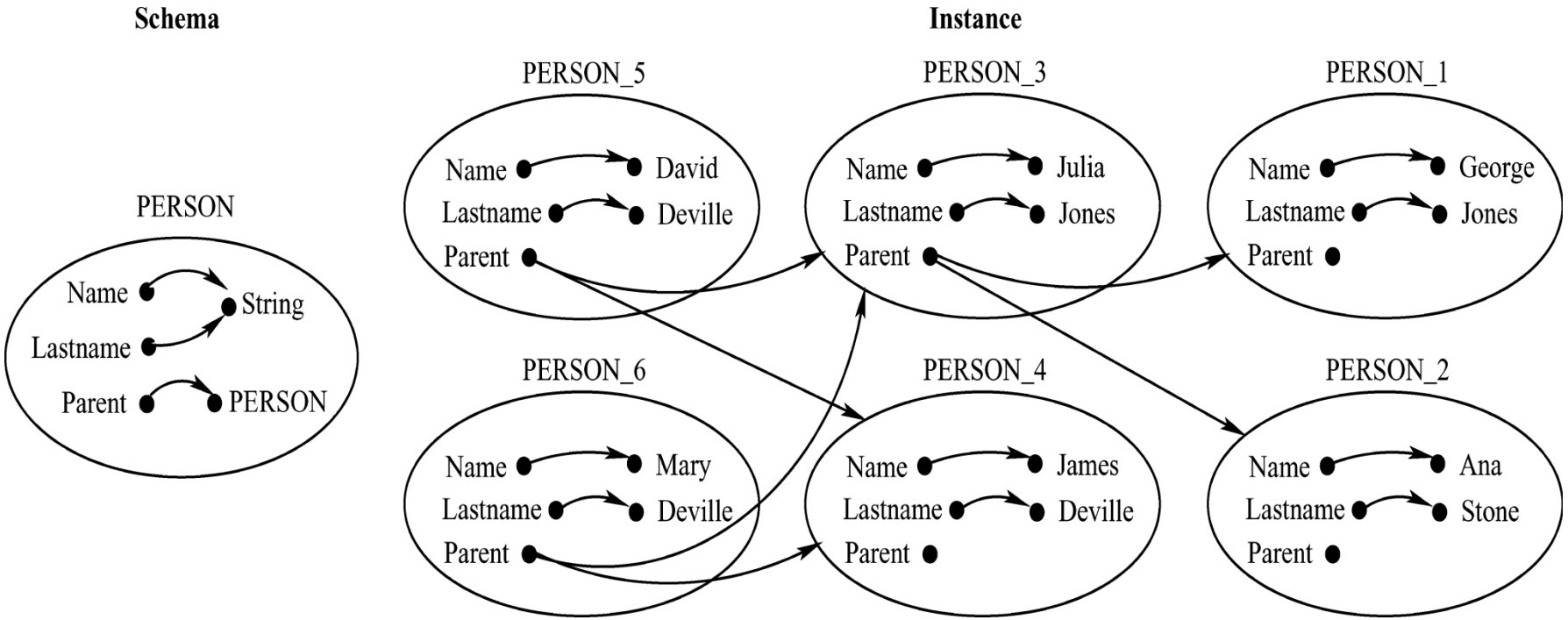
GMOD



In the schema, nodes represent abstract objects (Person) and labeled edges establish relations with primitive objects (properties name and lastname), and other abstract objects (parent relation).

For building an instance, we instantiate the schema for each person by assigning values to oval nodes

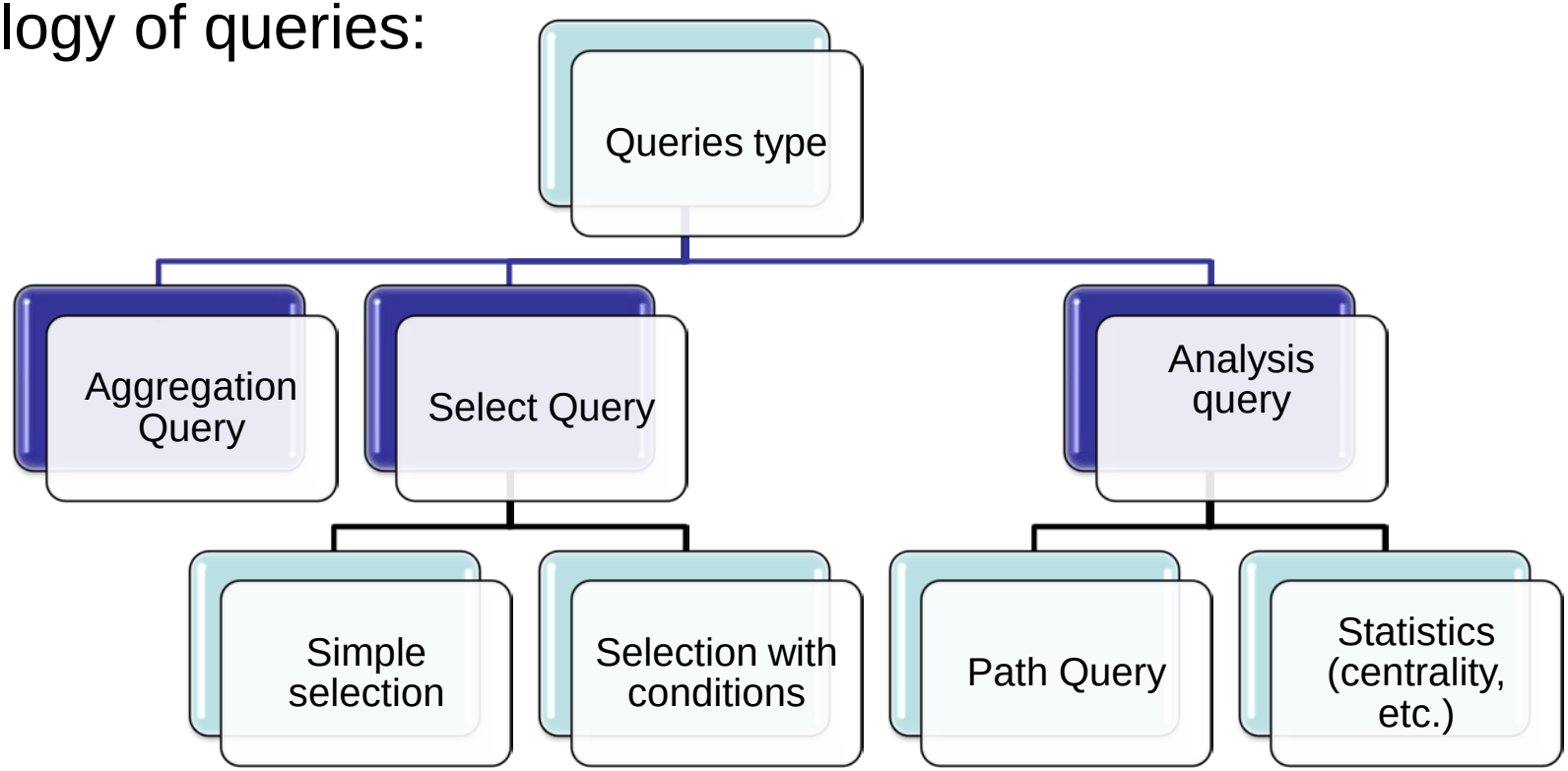
HYPERNODE



The schema (left) defines a *person as a complex object with the properties name* and lastname of type string, and parent of type person (recursively defined). The instance (on the right) shows the relations in the genealogy among different instances of person.

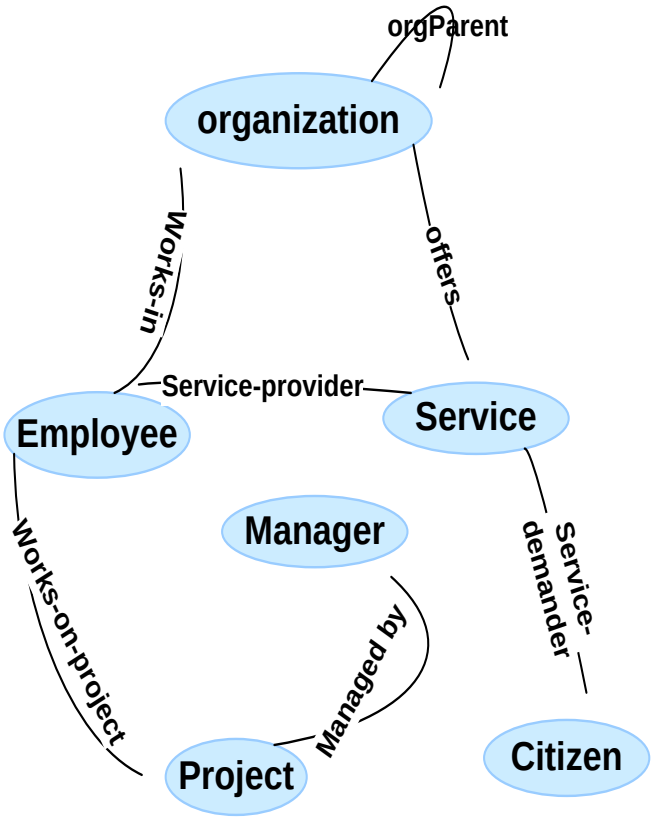
Query languages

- Graph models have generally an associated query language
  - Graphical/Visual, semantic, formal, SQL-like
  - RDF languages
  - XML languages
- Typology of queries:

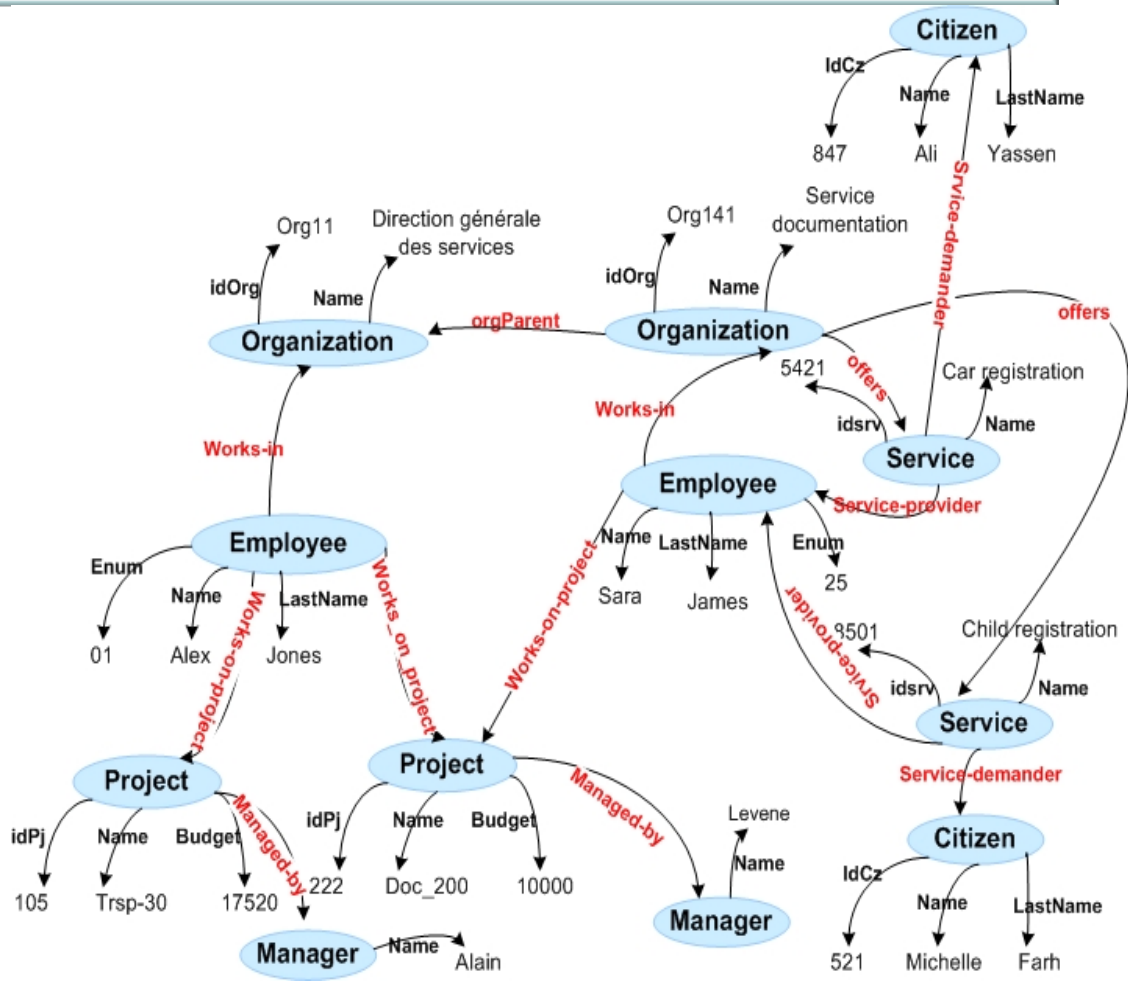


Example of visual query languages

These languages allow users to draw a query as a graph pattern. The result is the collection of all subgraphs matching the desired pattern.



Graph schema

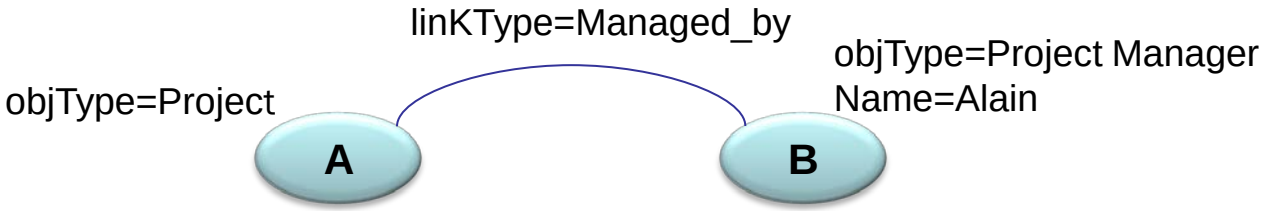


Graph instance

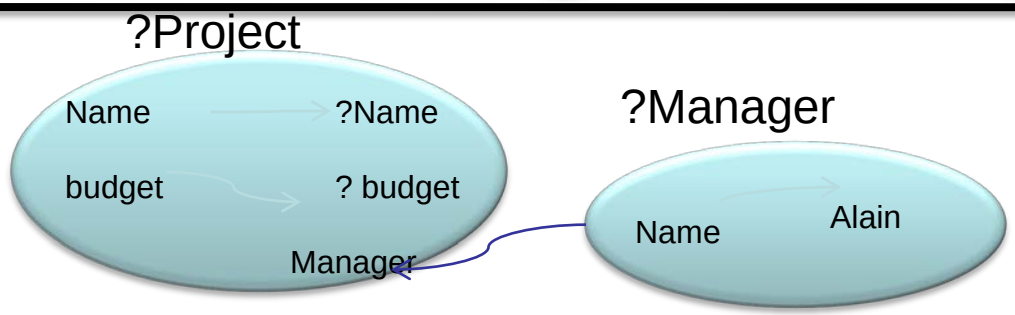
Example of visual query languages

Select projects where the project manager is « Alain »?

QGraph

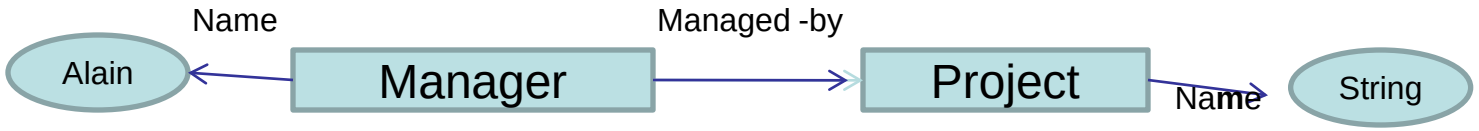


Hyperlog



G

GOOD



Example: Graph extraction and aggregation from relational databases

Soussi, Louati, Cuvelier, Aufaure, Lechevallier (2010, 2011)

Context/Motivation

- Motivation : be able to represent relations between any kind of object
  - Example : network of researchers and labs extracted from relational databases and enriched with web data

- Providing a unique representation for massive and heterogeneous data
  - Graphs : a natural representation
    - Adapted to both structured and unstructured data
    - Easy to manipulate
      - Data update without modifying the structure
      - Algorithms dedicated to graphs : shortest path, sub-graphs, etc.

Context/Motivation

- Social networks are playing an important role in information searching and sharing.
- Current methods for extracting/mining social networks are mainly based on web data
- Enterprise context: mainly relational databases (in the best case!)

→ **Social Networks Extraction from Relational Databases**

Use of graph models:

- More flexible and dynamic
- Reveal hidden relations
- Ease entities extraction

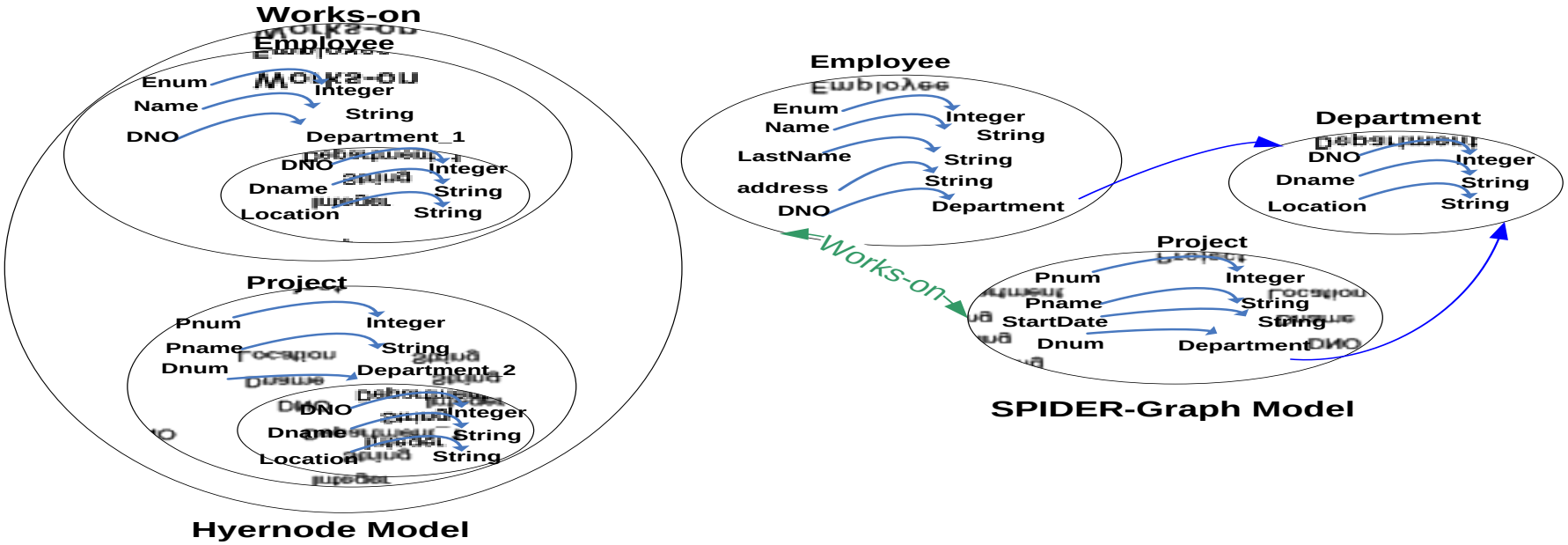
But: Extracted Social Networks can be very large

- Pertinent analysis and interpretation of data is difficult

→**Need of graph aggregation for the analysis**

→ **Use of the K-SNAP aggregation algorithm**

SPIDER-Graph Model



Hypernode Model vs Spider-Graph Model

Main differences between our model and the hypernode one:

1. Explicit representation of labeled edges to describe relations
2. Complex-node attributes types are references to other complex-nodes
3. Easier to visualize

SPIDER-Graph Model: Definitions

- Two levels: schema and instance (intent and extent)
- **Complex-node *CN***

CN is defined by *CN:= (c_n, A_{cn})* where: c_n denotes the name of *CN* and A_{cn} denotes a set of attributes A_{cn}:= $\{ a_{cn} | a_{cn} := \langle n, t \rangle \}$ where *n* is the attribute name, *t* is the type. *t* is a basic type (such as: Integer, String,...) or a reference to another *CN*.

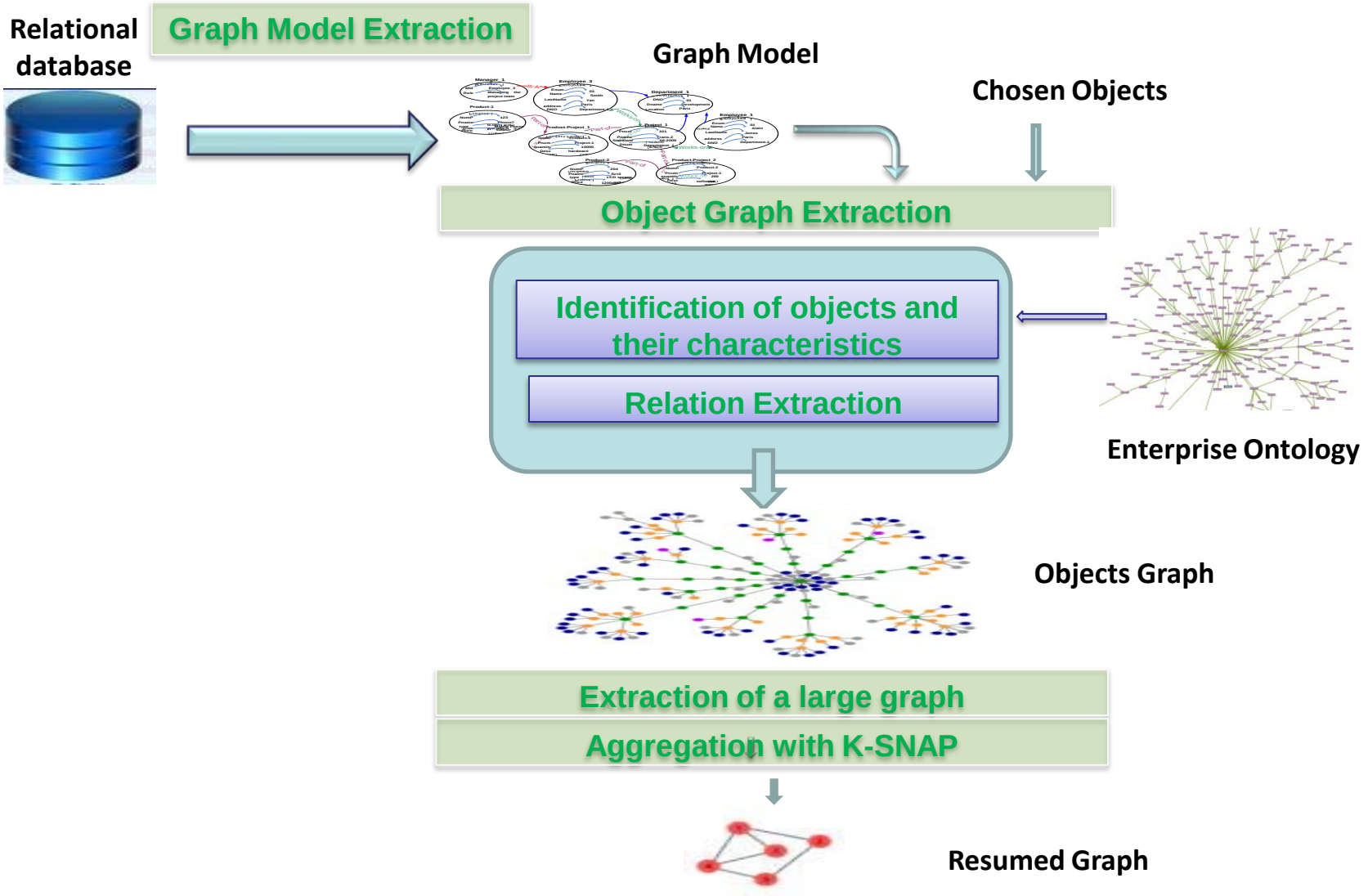
- **SPIDER-Graph Schema S**

S= $N_{cn} \cup R$ where: N_{CN} is a set of complex nodes and *R* is the set of relations between N_{cn} defined by

R:= $\{ \langle r, CN_s, CN_d \rangle, r \in R, CN_s, CN_d \in N_{CN} \}$

where *r* denotes the name of *R*, CN_s denotes the node source name and CN_d denotes the node destination name.

A general view of the approach



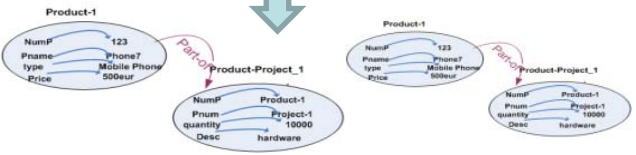
Selects nodes or concepts

Visual Query

Visual Query Language

Pattern Extraction

Graph Pattern Matching



Enterprise Data

Relational database

Enterprise Documents

Graph Model Extraction

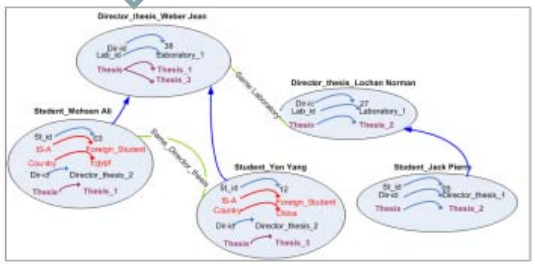
Enterprise Ontology building

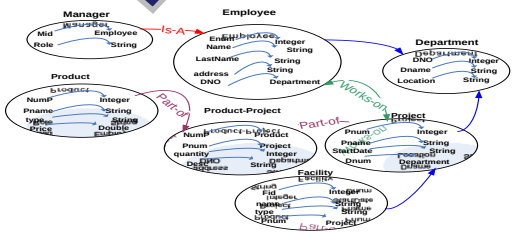
Object Graph Extraction

Object Identification

Relation Extraction from Ontology

Relation Extraction from Graph model





Transformation to a SPIDER-Graph Model from a RDBMS: Schema Translation

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

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

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

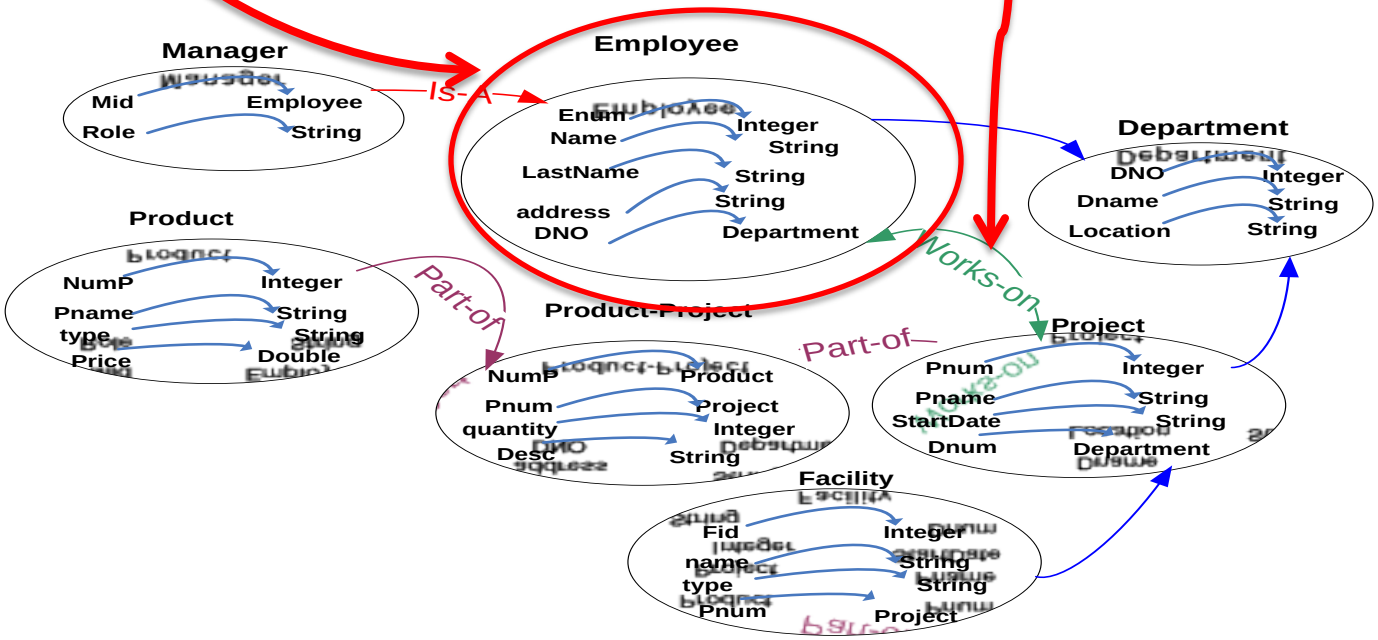
Project-Product			
Pnum#	**numP#**	desc	quantity
101	123	hardware	10000
102	123	Software	234
101	204	Software	200

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

Manager	
Mid#	Role
03	Managing the project team

(1) Complex-node creation

(2) Relations Identification



Transformation to a SPIDER-Graph Model: Data Migration

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

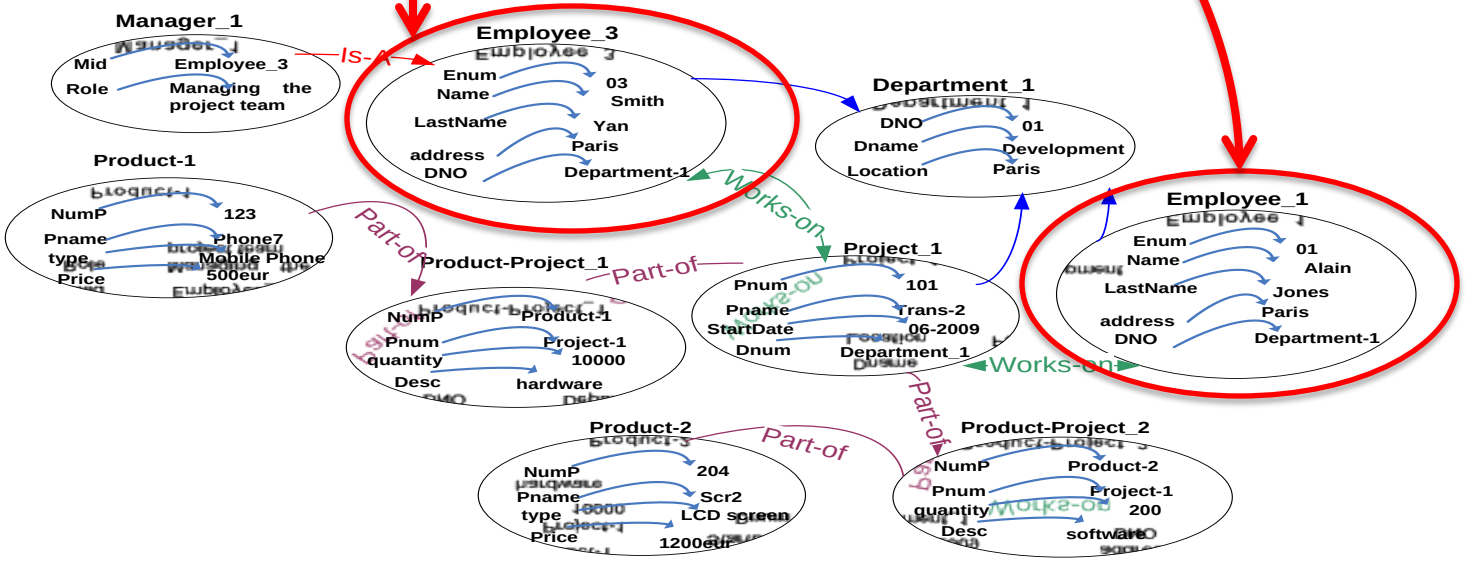
Project

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

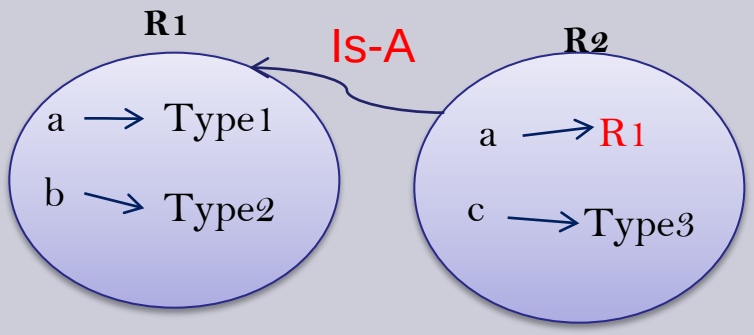
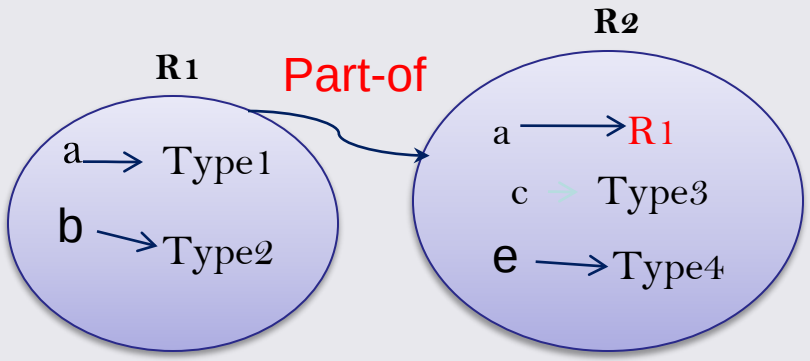
Manager

Mid#	Role
03	Managing the project team

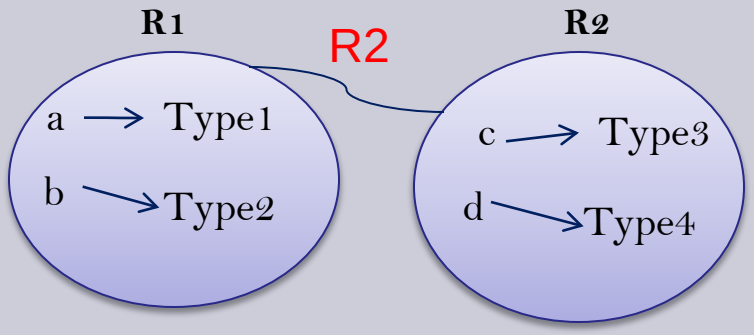
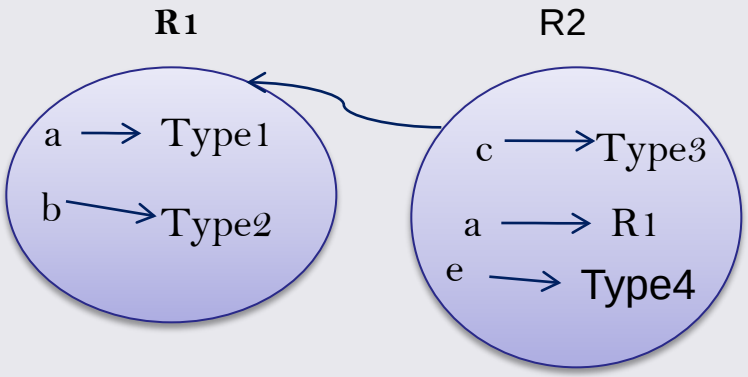
(3) Intance level (population)

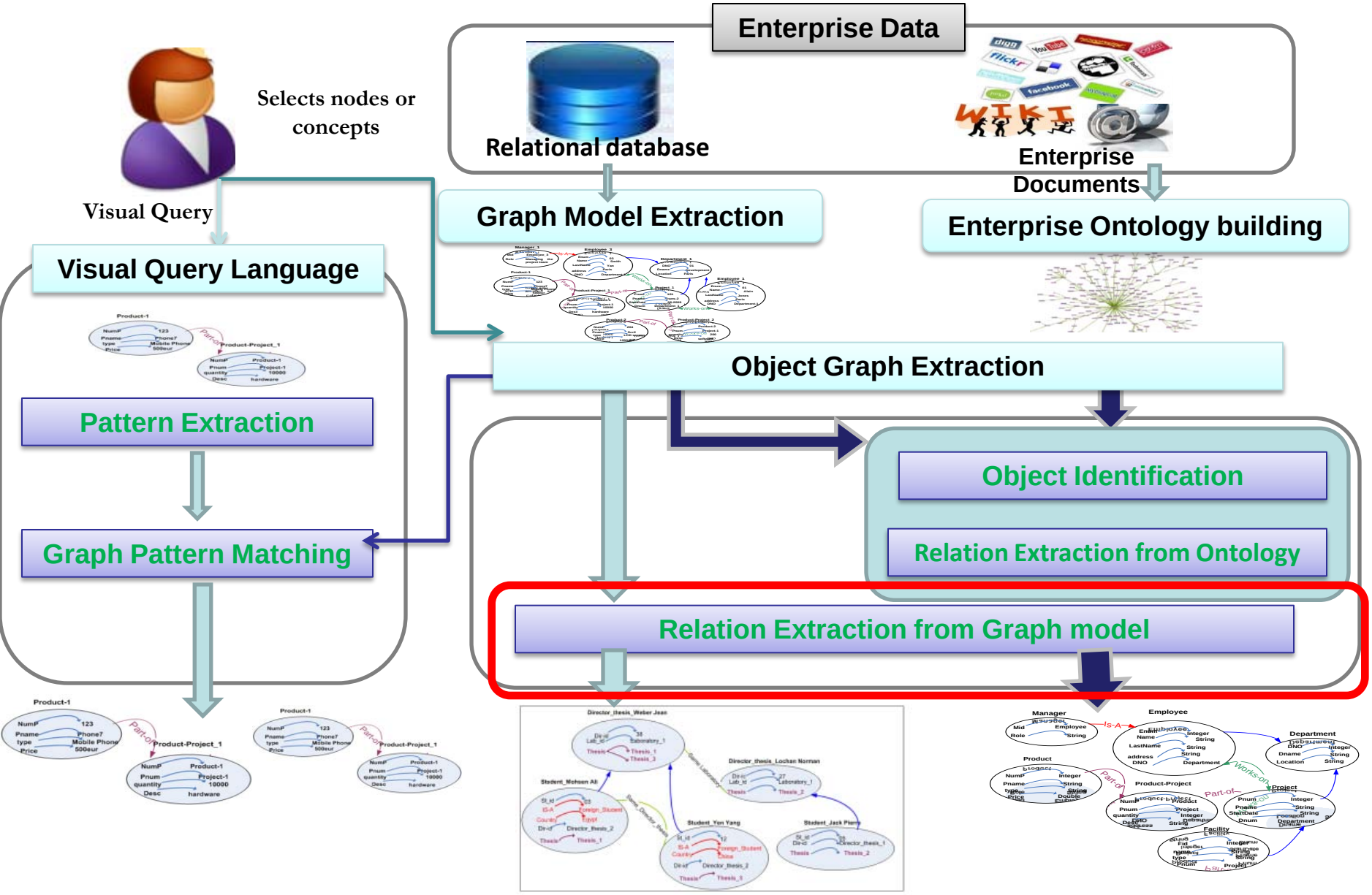


(Hidden) Relations identification

Relation	Transformation
R1 (a,b) R2(a #,c)	
R1 (a,b) R2(a#, c, e)	

(Hidden) Relations identification

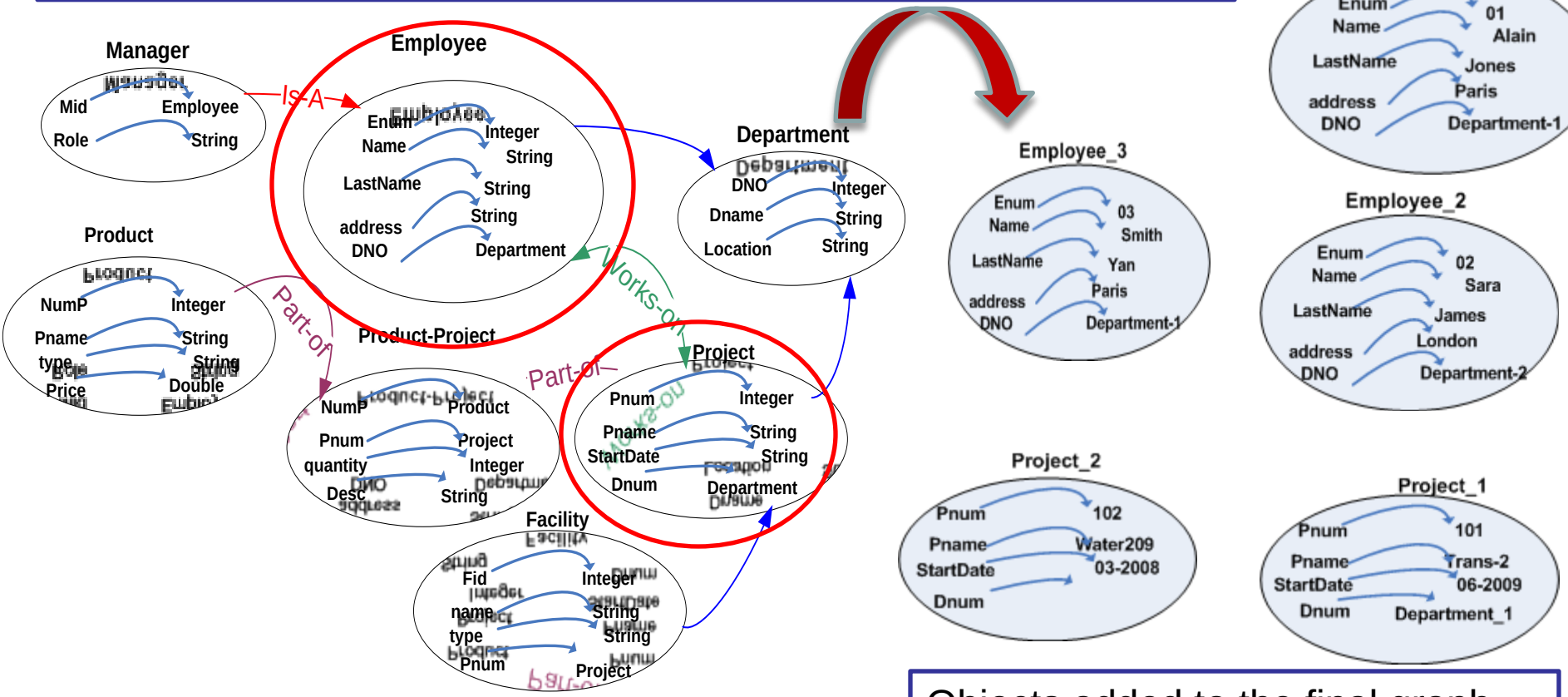
Relation	Transformation
R1 (a, b) R2(a#, c#) R3(c, d)	
R1 (a, b) R2(c, a#,e)	


Enterprise Data
Relational database
Enterprise Documents
Selects nodes or concepts
Visual Query
Graph Model Extraction
Enterprise Ontology building
Visual Query Language
Object Graph Extraction
Pattern Extraction
Object Identification
Graph Pattern Matching
Relation Extraction from Ontology
Relation Extraction from Graph model

Objects identification

Selected concepts: EMPLOYEE, PROJECT

Corresponding complex-Nodes:



Objects added to the final graph

Relation construction (definitions)

- **Objective: use existing relations and find hidden ones**
- **Input: SPIDER-Graph, set of relation patterns, Output: Interaction Graph**
- **Definition: Interaction Graph *GO***
  - ***GO* = (O_I, R_O) where:**

    O_I is a finite set of objects such as $O_I := \{o_I \mid o_I \in N_{CN}\}$

    R_O is a finite set of relations between objects such as

    $R_O := \{r \mid r := (l, o_{I1}, O_{I2})$ where l is the relation name and

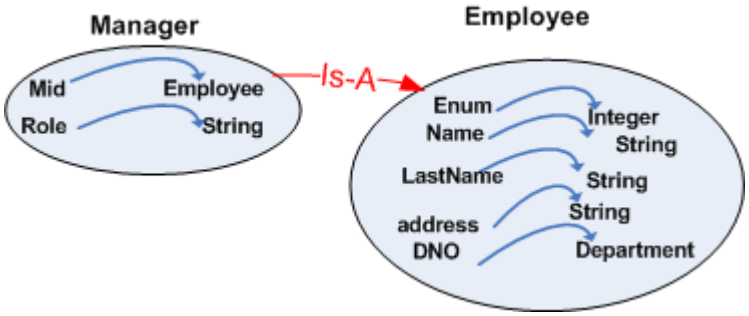
    $o_{I1}, O_{I2} \in O_I$
- **Relation pattern: $P_r = (n, o_{I1}, o_{I2}, o_m)$**
  - **n = name of the relation**
  - **o_{I1}, o_{I2} are entities sharing the relation**
  - **o_m is a mediator for this relation (complex object used to find a new relation**

Object Graph Extraction
Relation Construction-1-

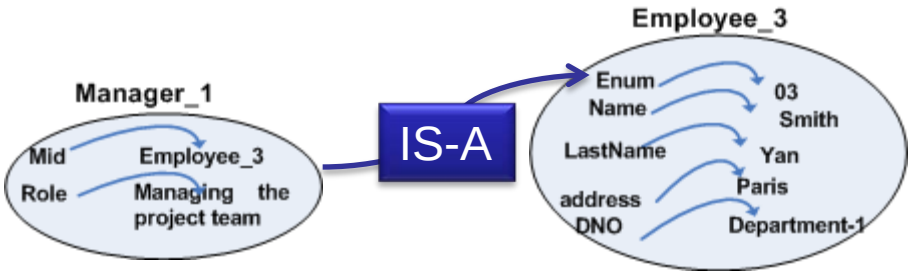
Patterns to identify new objects. **IS-A** Relation

	Initial Relation	Process
Definition	R1:=(« IS_A », CN_s, CN_d), CN_s or CN_d instances $\in O_l$	CN_s or CN_d instances are added to O_l
Example	<« IS_A », Employee, Manager>	Manager instances are added to O_l

Input Relation (Schema)



Detected Objects (instances)



Object Graph Extraction
Relation Construction-2-

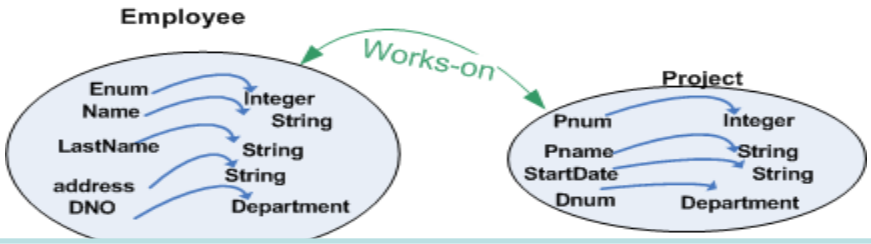
Patterns based on relation between chosen objects.

Initial Relation	Pattern	Process and description
$R_2 := \langle r, CN_s, CN_d \rangle$ where CN_s and CN_d instances $\in O_l$	$Pr_1 := \langle r, CN_{si}, CN_{di}, null \rangle$ where CN_{si} and CN_{di} are CN_S and CN_d instances.	Find all the existing relations between CN_s and CN_d instances then add them to R_O
	Pr_2:=<Same_(CN_d.name),CN_{si1}, CN_{si2} , CN_d > where *CN_{si1}* and *CN_{si2}* are *CN_s* instances and i!=j	Find all the CN_s instances which have relations with a same CN_d instances and link them with a new relation "Same_(CN_d.name)"
- CN_s and CN_d instances $\in O_l$ - No relation between CN_s and CN_d	*Pr_{path}:=<"",CN_{si}, CN_{di}, P>* where *P* is a semi-path or a path between *CN_s* and *CN_d*	-connected with a direct path -connected with indirect path -Not connected

Object Graph Extraction Relation Construction-3-

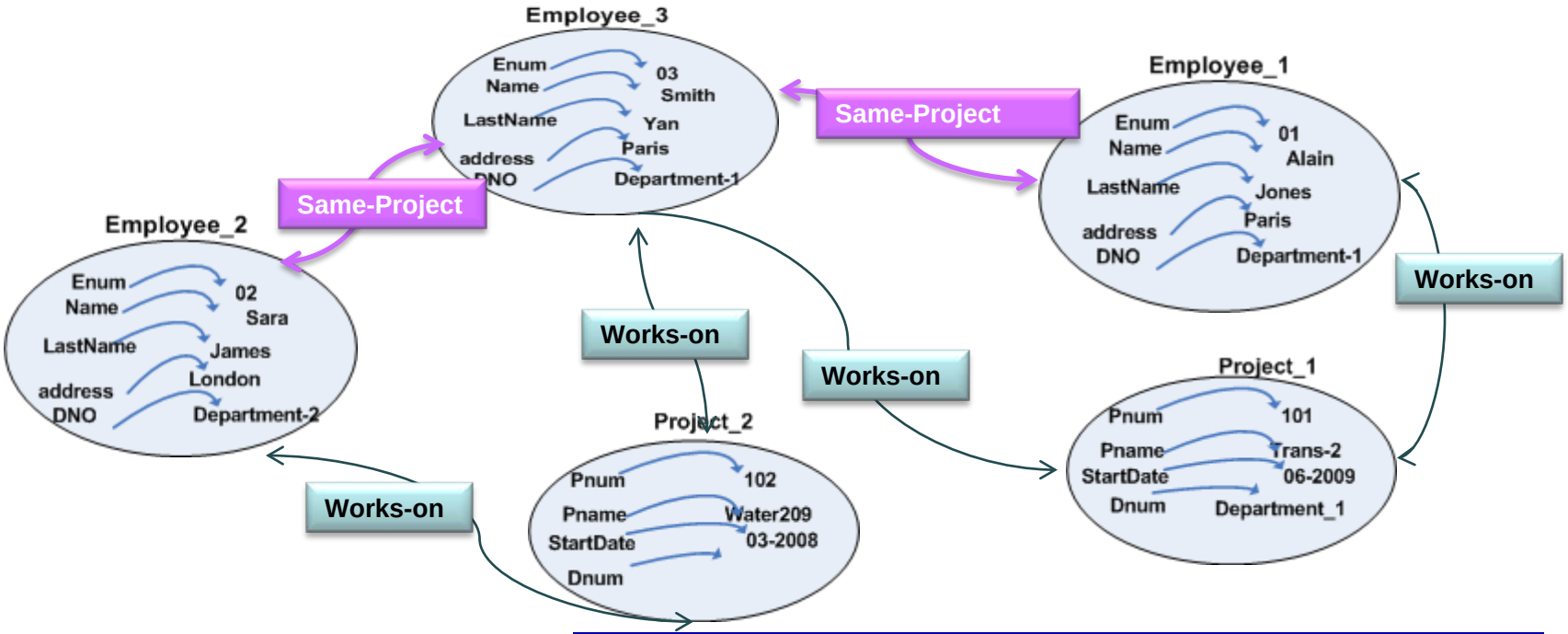
Patterns based on relation between chosen objects.

Input Relation



New Relations

Pr2:=<Same_(CNd.name),CNSIi, CNSIj , CNd > where CNSIi and CNSIj are CNs instances and i!=j



Object Graph Extraction
Relation Construction-4-

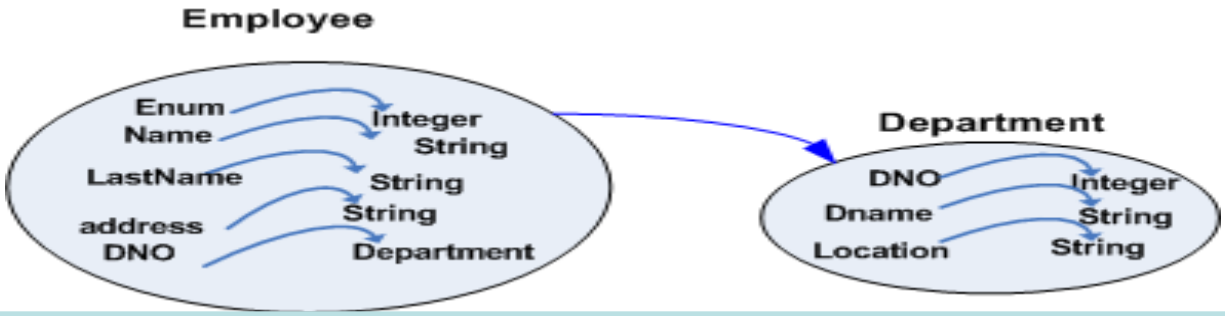
Patterns based on relation between a chosen object and another Complex node

Initial Relation	Pattern	Process and description
$R_3 := \langle r, CN_s, CN_d \rangle$ where $CN_s \in O_I$ and $CN_d \notin O_I$	Pr_3:=<Same_(CN_d.name), CN_{Sli}, CN_{Slj}, CN_d > where CN_{Sli}, CN_{Slj} are CN_S instances, i!=j and r != "Part-of"	Find all the CN_s instances which have relations with a same CN_d instances and link them with a new relation
	Pr_4:=<Same_CN_k.name, CN_{Sli}, CN_{Slj}, CN_k > where CN_{Sli}, CN_{Slj} are CN_S instances, i!=j, r="Part-of" and $CN_k \in$ {CN\CN has the relation R:=<"Part-of", CN, CN_d>}	1.Find all the complex-nodes CN_k having a "Part-of" relation with CN_d such as R:= <"Part-of", CN_k, CN_d > 2. add the new attribute to CN_s containing the name of CN_j 3.Then the pattern Pr_4 is applied : find all the CN_d instances which have relations with a same CN_j instances and link them with a new relation
$R_4 := \langle r, CN_s, CN_d \rangle$ where $CN_s \notin O_I$ and $CN_d \in O_I$	Pr_5:=<Same_CN_s.name, CN_{dI}, CN_{jI}, CN_s > where CN_{dI} is CN_d instance and CN_{jI} is instance of CN_j where $CN_j \in O_I \cap$ {CN\CN has the relation R:=<"", CN, CN_s>}.	1. add a new attributes in CN_d containing CN_s reference. 2. Then the pattern Pr_5 is applied : if CN_s has relations with other objects, link each object instance with a CN_d instance if they are in relation with the same CN_s instance.

Object Graph Extraction
Relation Construction-5-

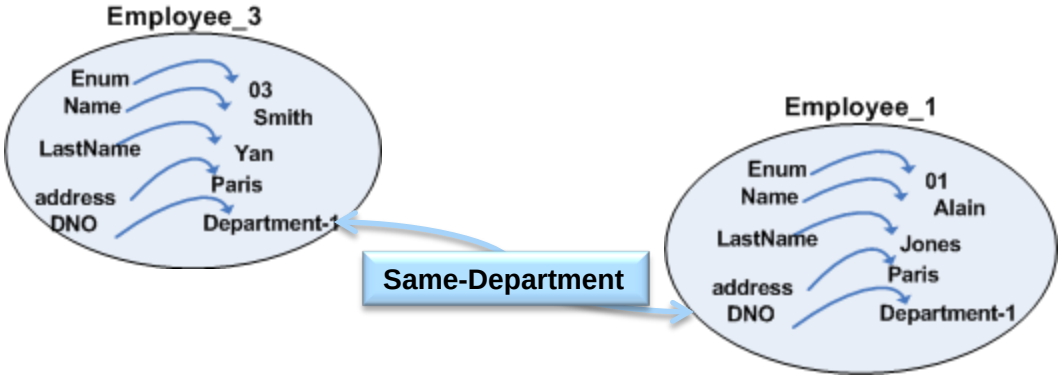
Patterns based on relation between a chosen object and another Complex node

Input Relation



New Relations

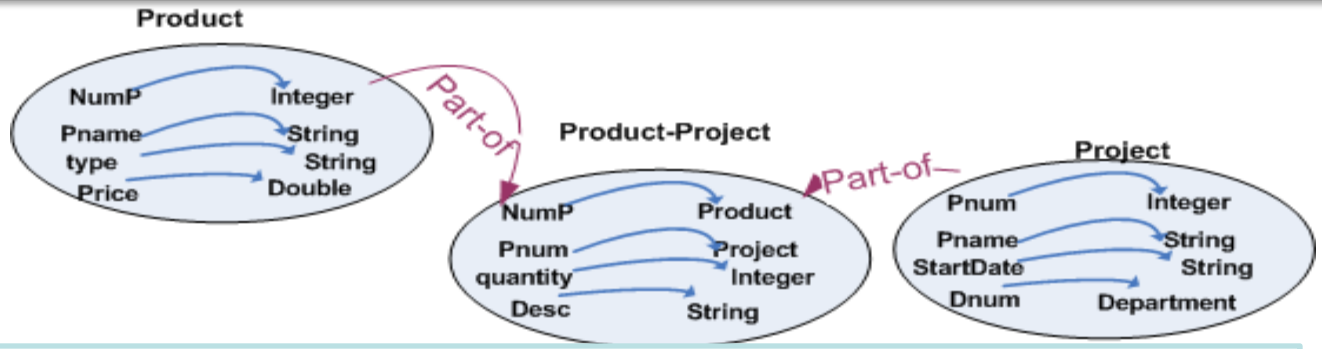
Pr_3:=<Same_(Department), $Employee_i$, $Employee_j$, $Department_d$ > where i!=j



Object Graph Extraction
Relation Construction-6-

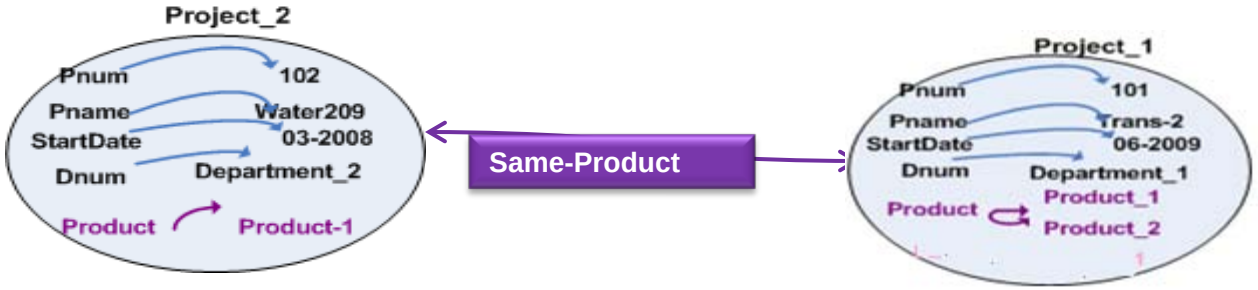
Patterns based on relation between a chosen object and another Complex node

Input Relation



New Relations

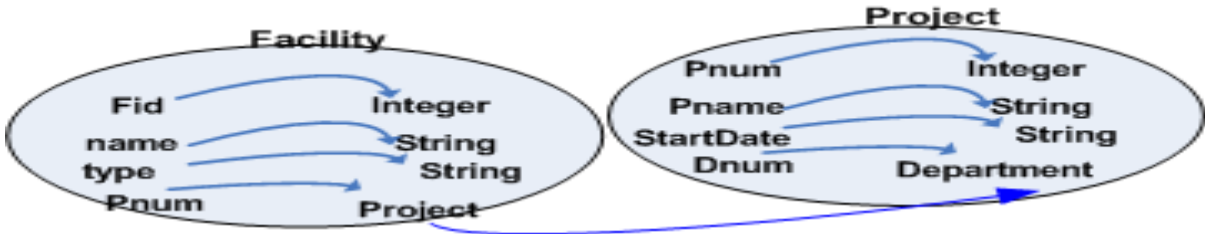
Pr_4:=<Same_Product, $Project_i$, $Project_j$, $Product\_Project_k$ > where" and i!=j



Object Graph Extraction
Relation Construction-7-

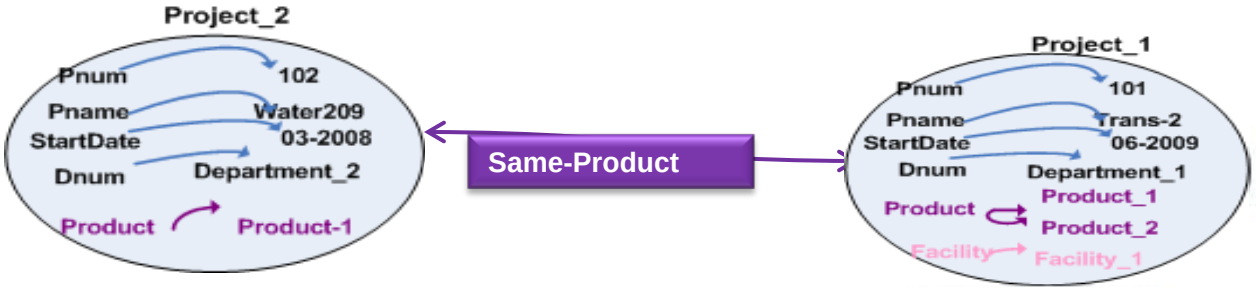
Patterns based on relation between a chosen object and another Complex node

Input Relation



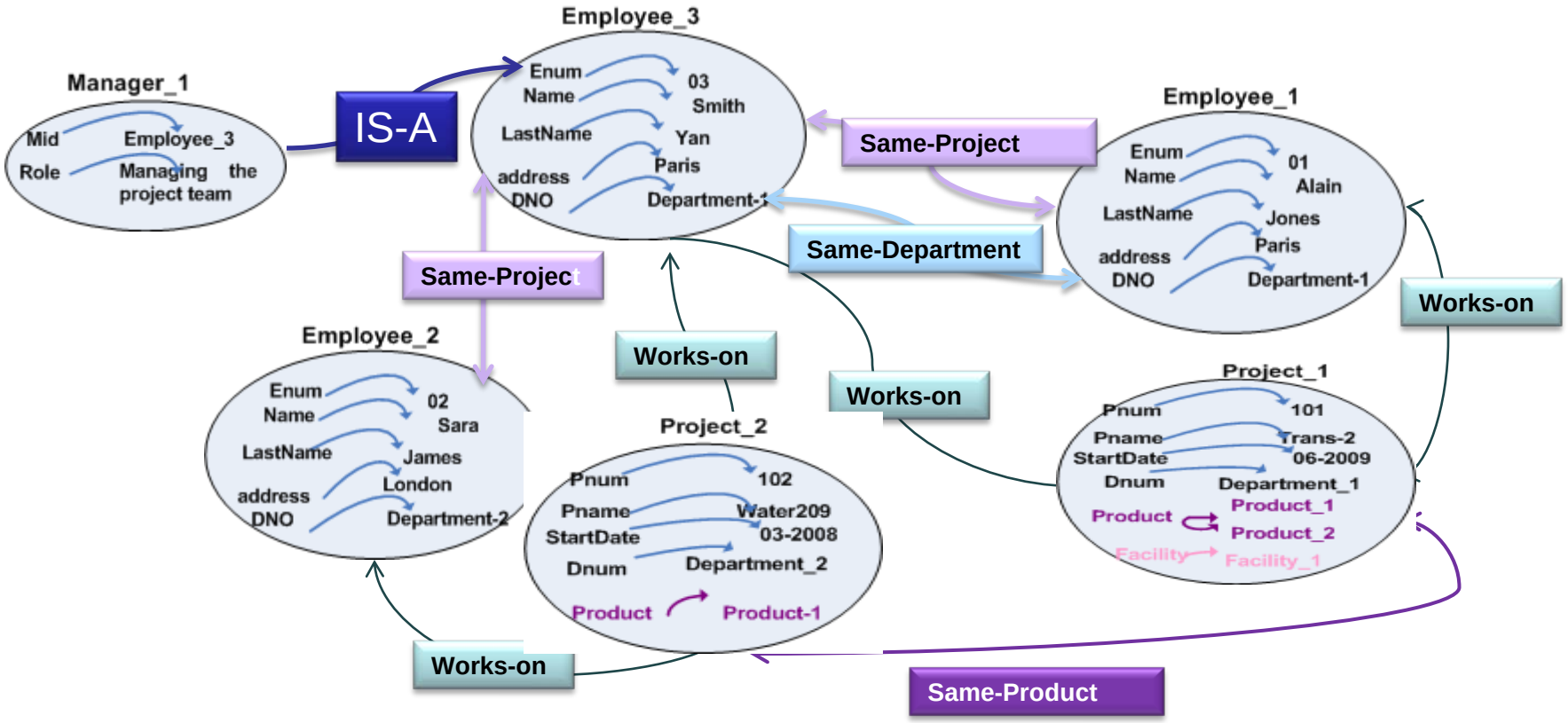
New Relations

Pr_5:=<Same_Facility, $Project_i$, $Project_j$, $Facility_s$ >

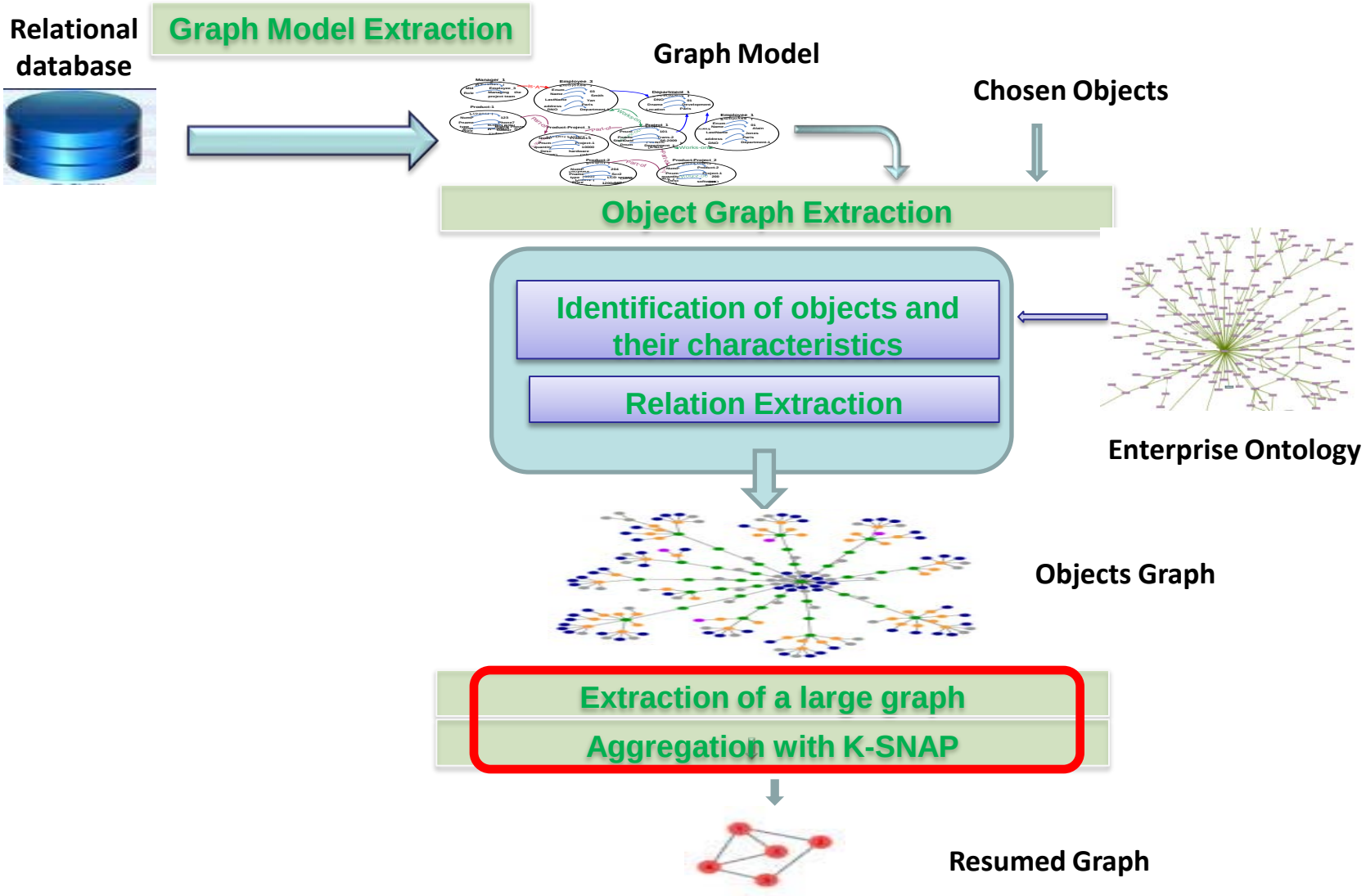


The Object Graph between Employee and Project

Objects: Employee and Project



Graph Aggregation

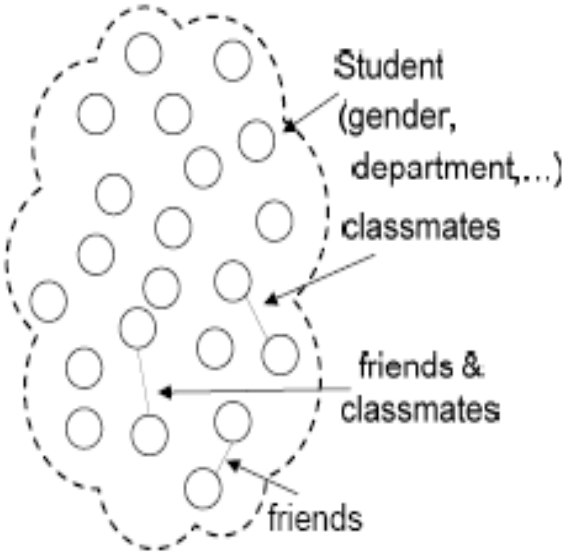


SNAP & k-SNAP Graph Aggregation
Tian, Hankins and Patel (SIGMOD 2008)

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

Working with...

- Nodes
- Edges (but with different types)
- Attributes (all identical for all nodes)
- Node Grouping = Partition of Nodes
- Neighbor Groups : groups in relationship with a node.



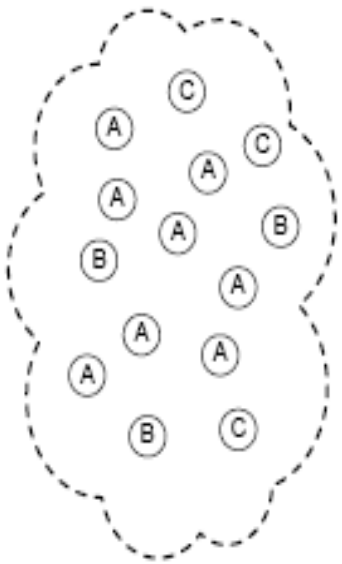
Graph: G

Compatible Groupings

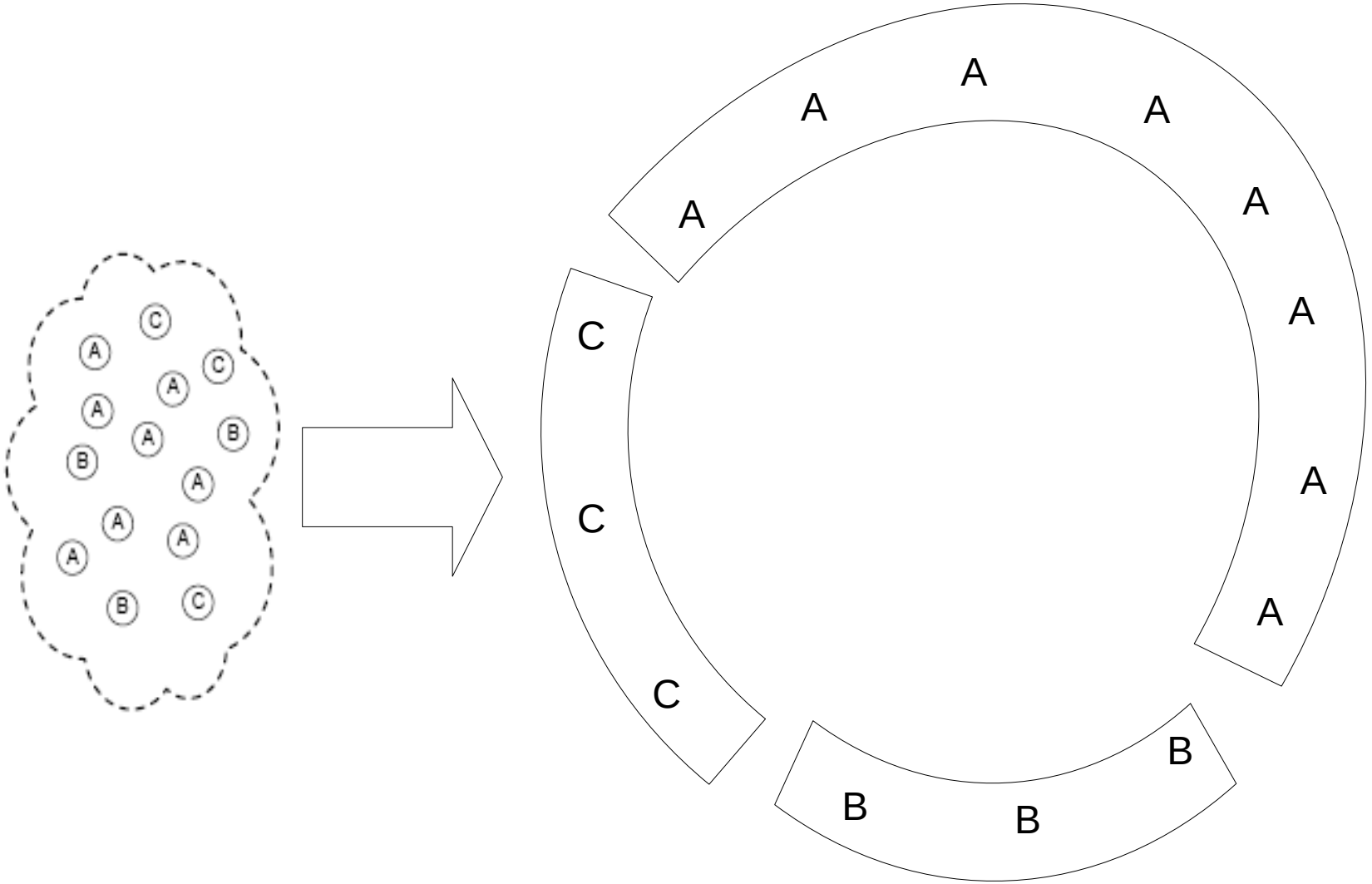
- A-compatible Grouping :
  - all nodes in the same group must have the same attributes.
- (A,R)-compatible Grouping:
  - A-compatible grouping,
  - all nodes in the same group must have the same neighbor groups.

(A,R)-compatible grouping performed by the SNAP :
Summarization by grouping **N**odes on **A**ttributes and **P**airwise relationships.

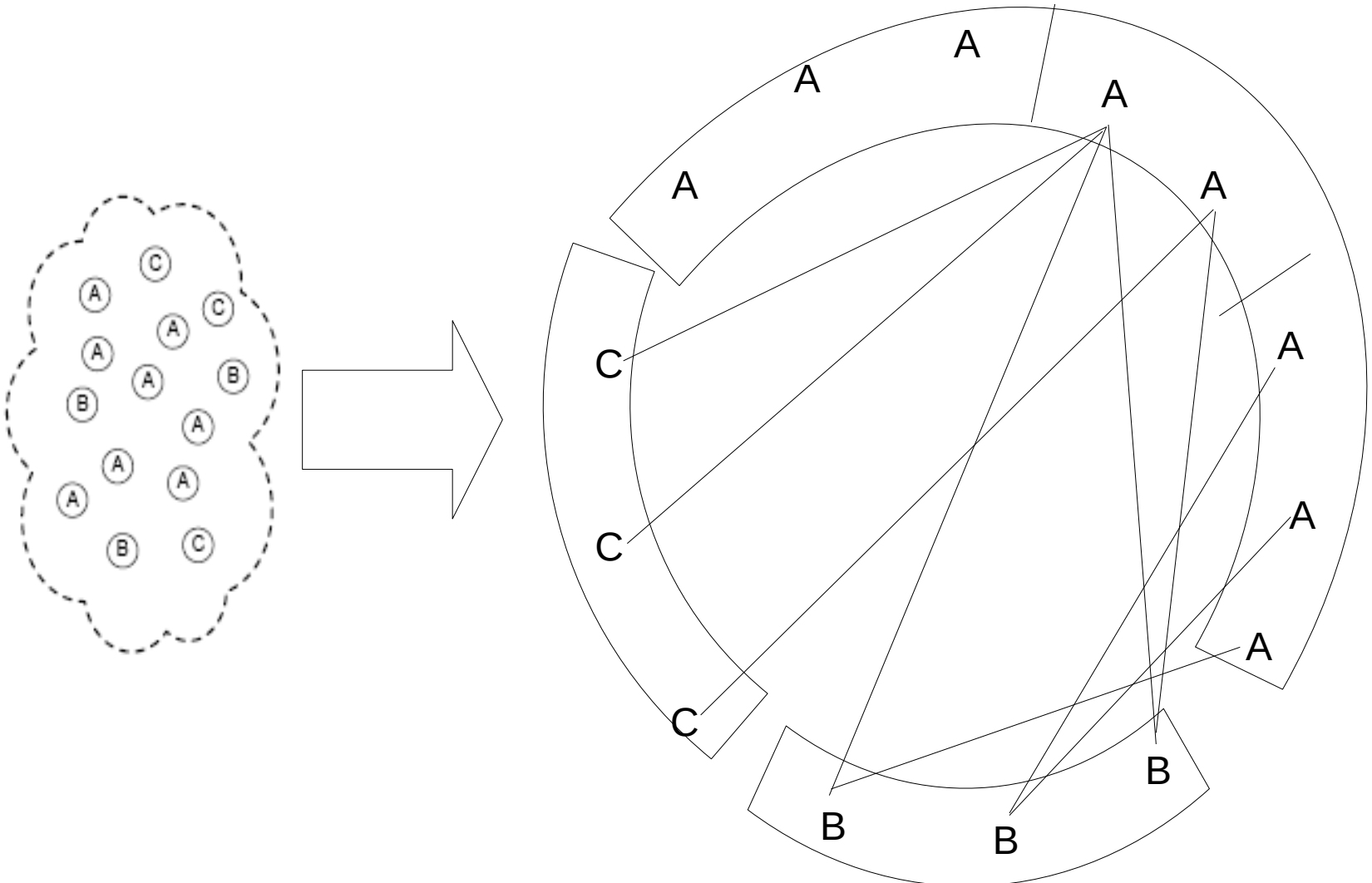
SNAP Operation



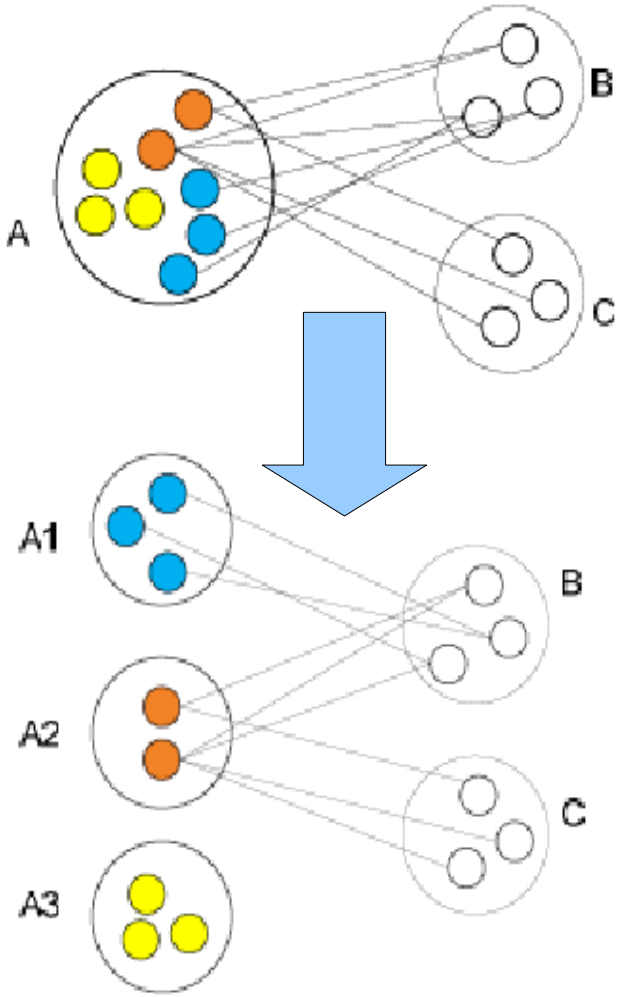
SNAP Operation

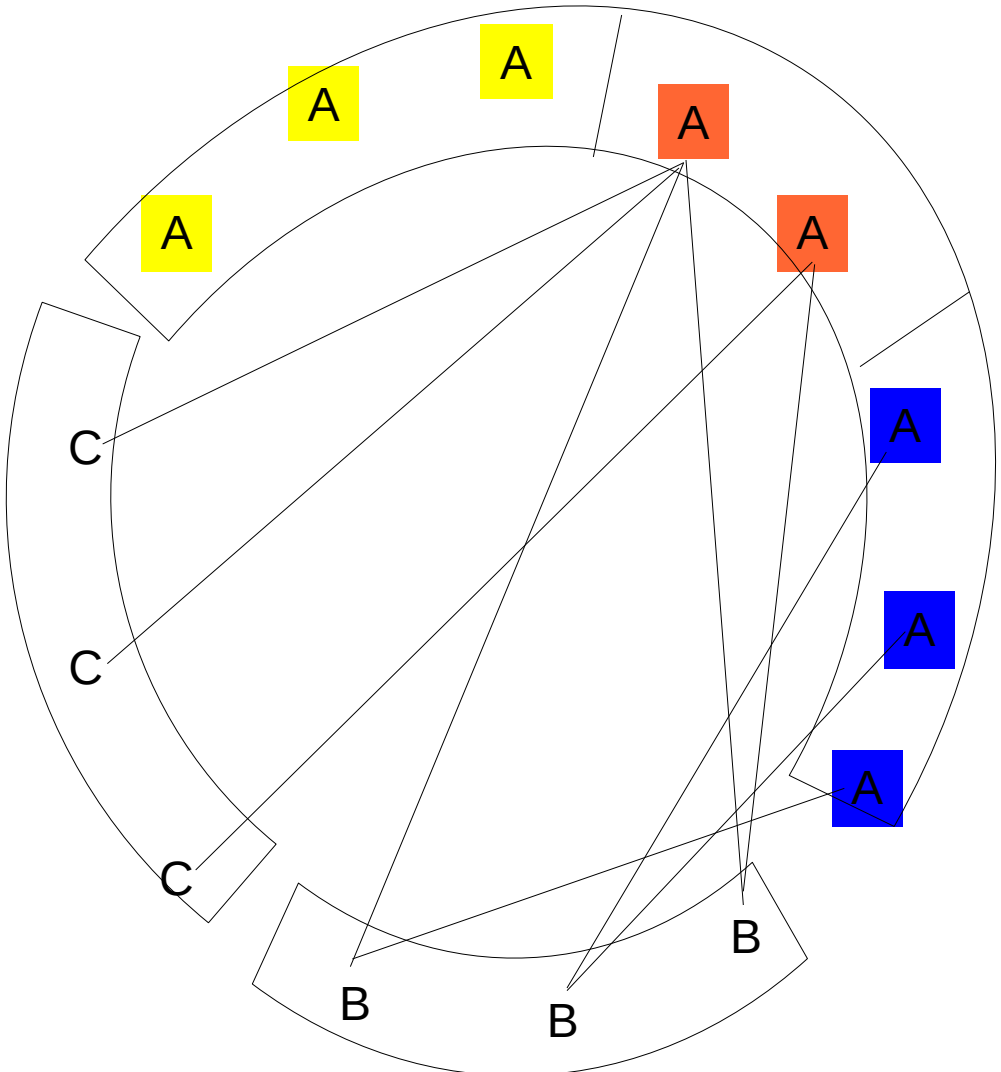


SNAP Operation

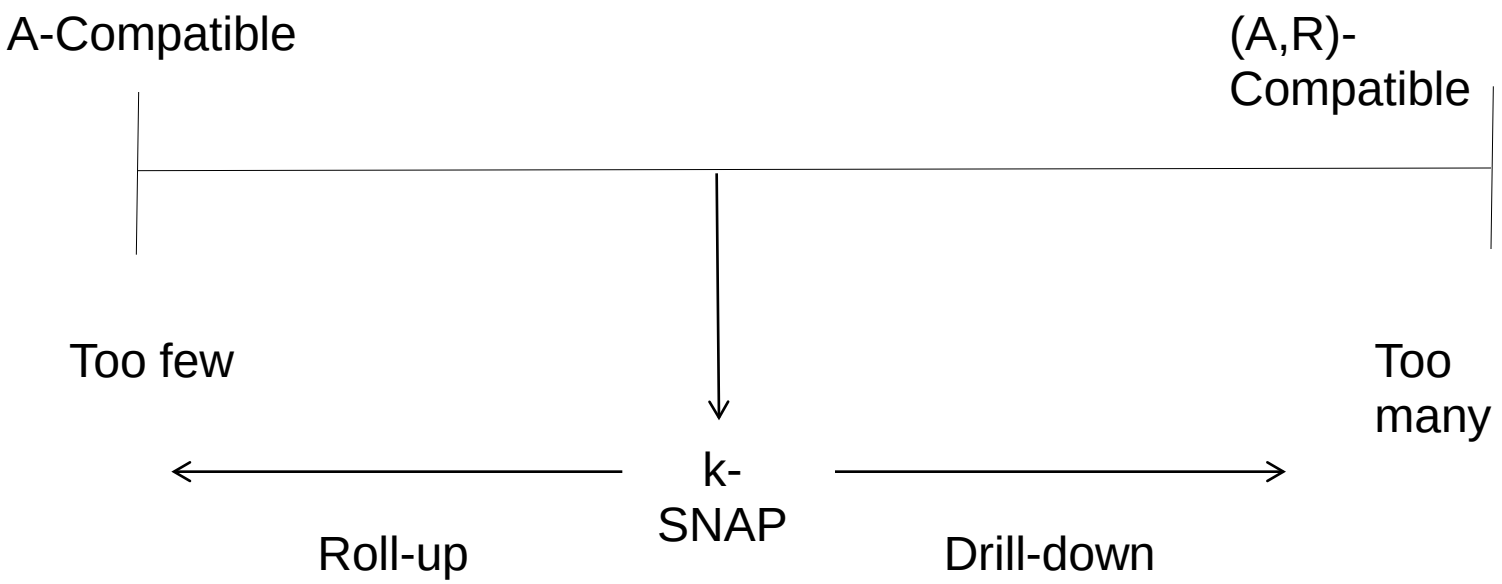


SNAP Operation





Why k-SNAP ?

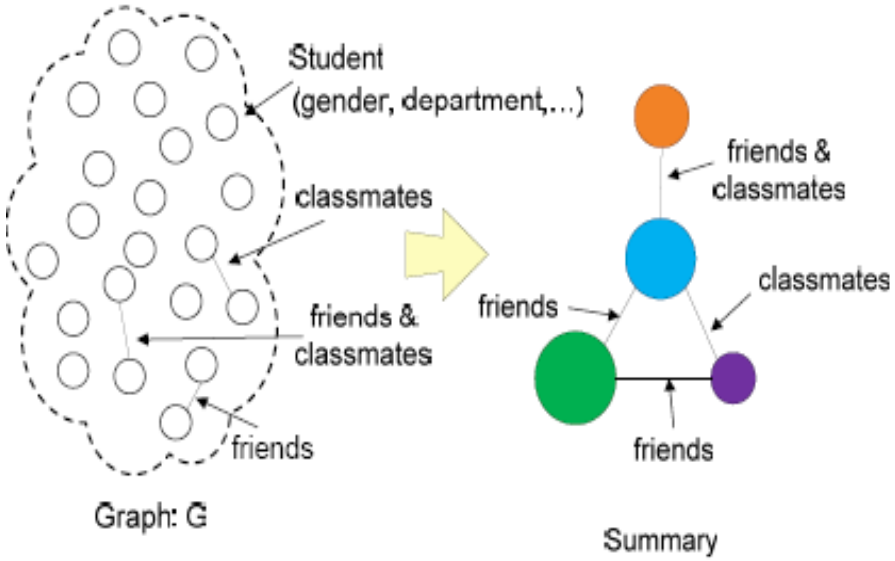

A-Compatible
(A,R)-Compatible
Too few
Too many
k-SNAP
Roll-up
Drill-down

Attributes and relationships

- Attributes and relationships together, but attributes first!

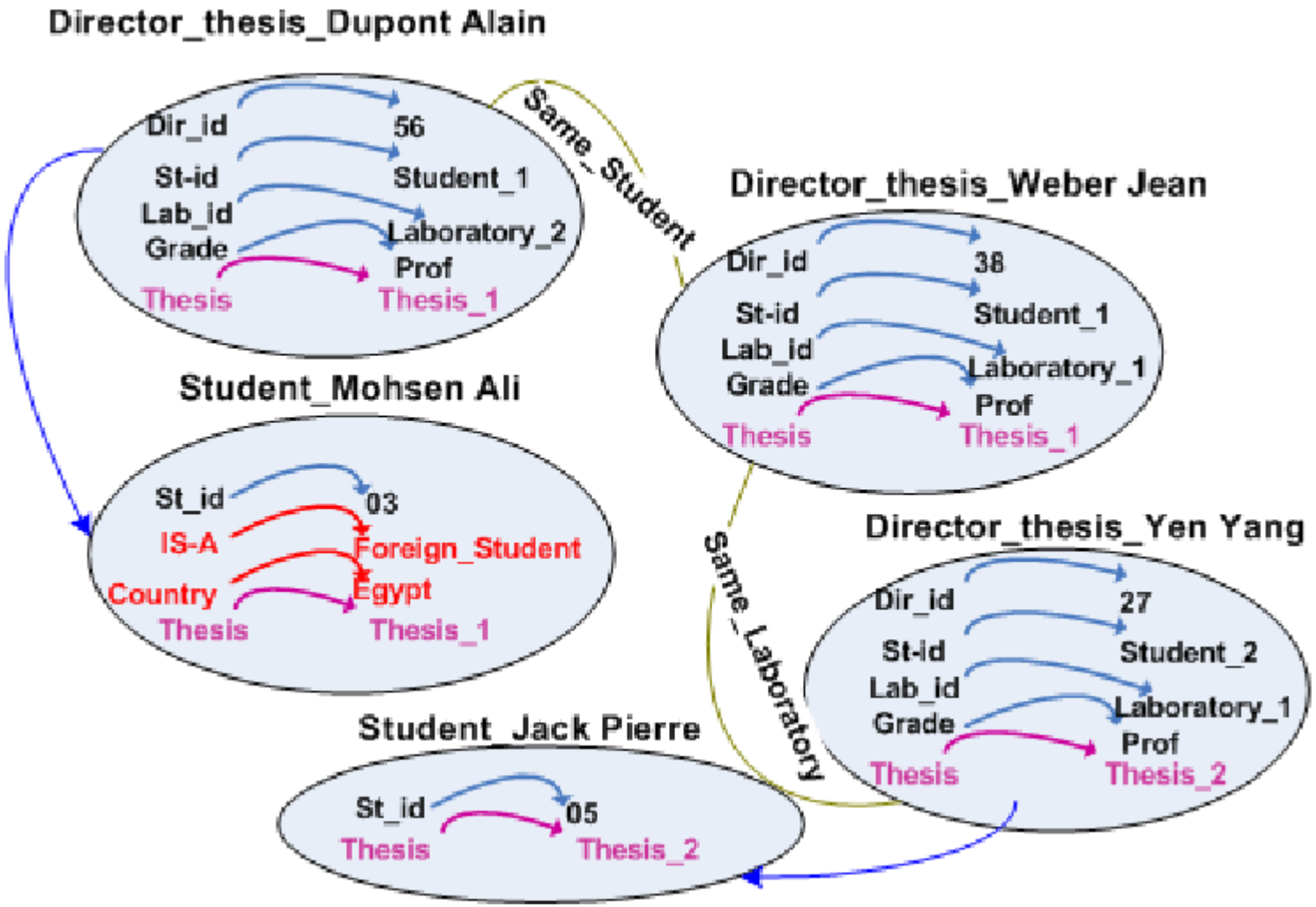
For example:

- All students in the blue group have the same gender and are in the same dept
- Every student in the blue group has:
- at least one “friend” in the green group
- at least one “classmate” in the purple group
- at least one “friend” in the orange group
- at least one “classmate” in the orange group

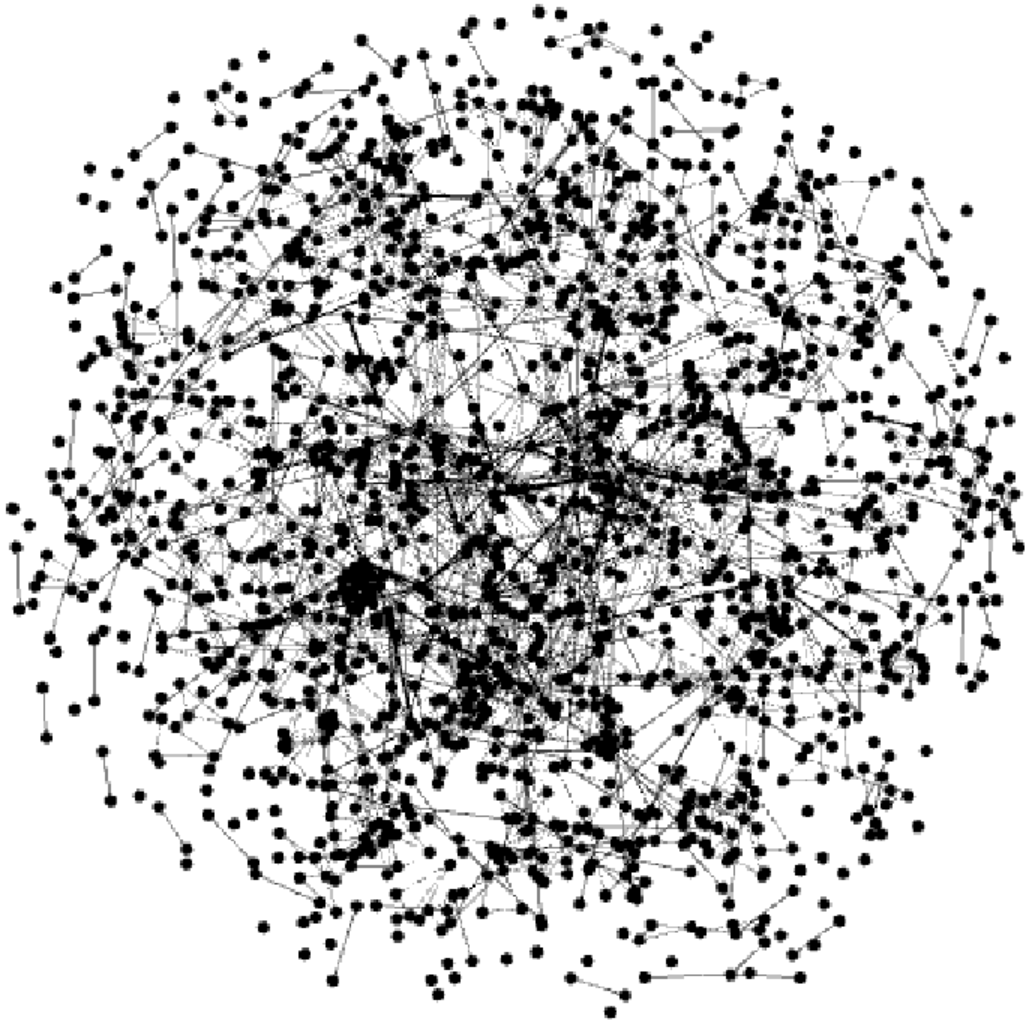


Graph Aggregation: example

Interaction Graph representing a social network (extracted from the ADEME Database)



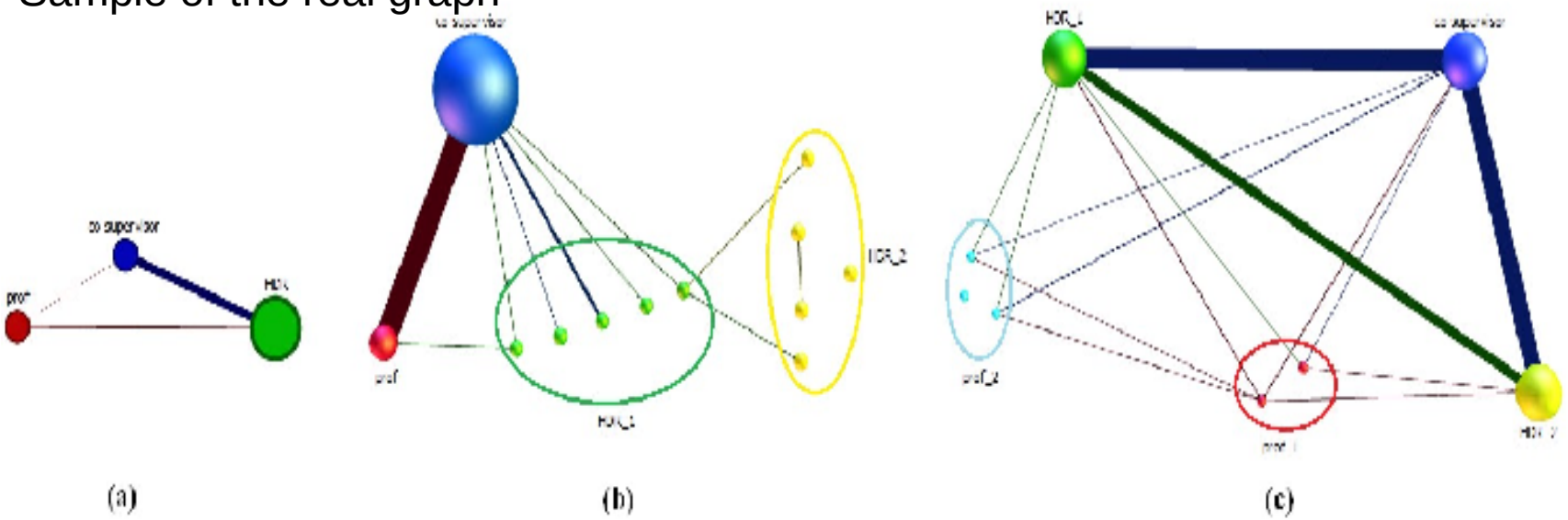
Graph Aggregation: example



Initial graph with selected nodes and relations:
Nodes: Thesis-Director
Attribute: grade
Relations: Same_Laboratory and Same_Student

Graph Aggregation

Sample of the real graph

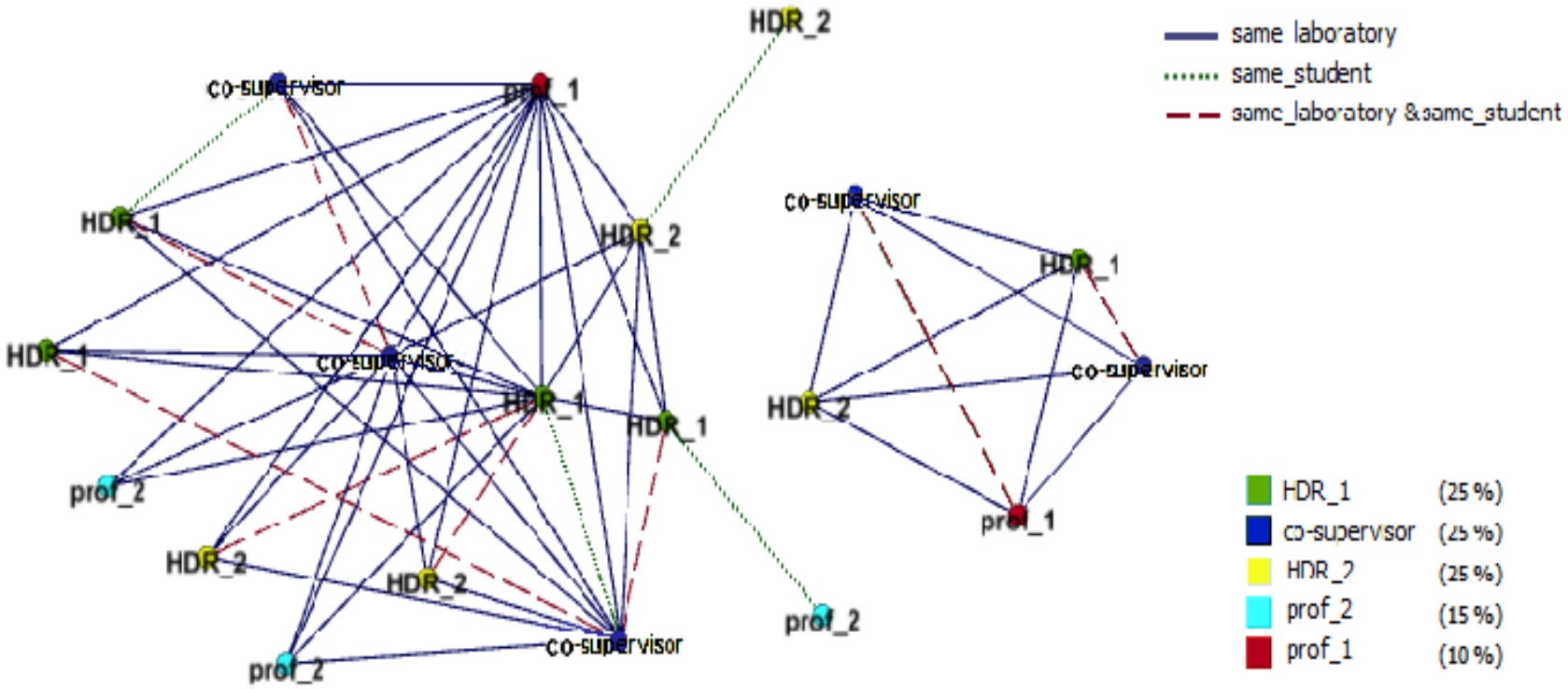


K-snap generates a summary formed by 3 groups (A-compatible grouping):
HDR, co-supervisor, prof (modalities of the attribute grade)

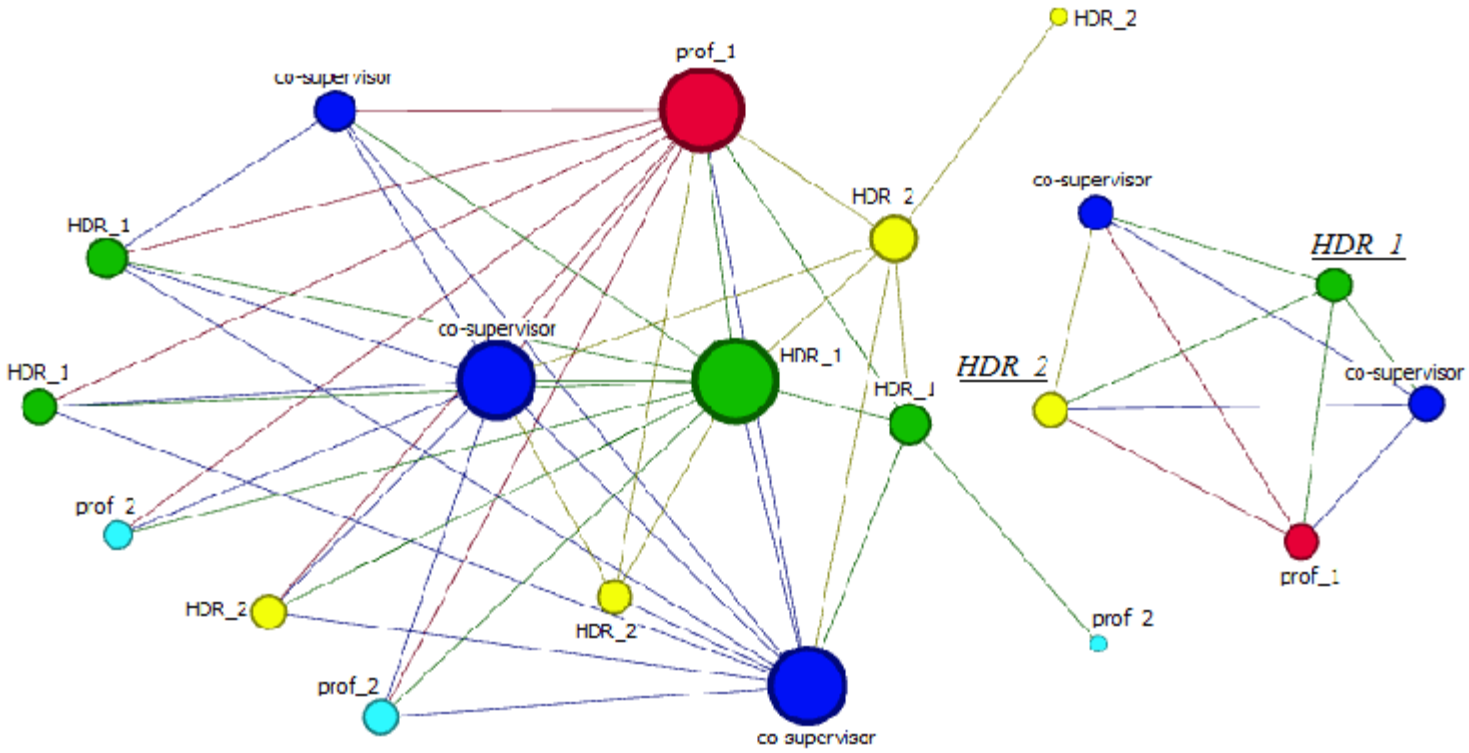
1st iteration: subdivision of the HDR group into 2 subgroups according to the relation Same_Student: HDR_1: HDRs supervising a student with at least one professor or co-supervisor, HDR_2: HDR supervising students having only as director HDRs

2nd iteration: subdivision of the Prof group into 2 subgroups according to the relation Same_Laboratory

Graph Aggregation



Graph Aggregation



Application of the centrality measure to compare the results

Limits of the K-SNAP algorithm

- Can only be applied to homogeneous graphs (same description for each node)
- Number of groups can be the Cartesian product of modalities of all attributes
- Inefficient in case of numerous attributes and/or numerical values

$\Rightarrow$Increase of small groups

Conclusion

- Graphs: towards a unified view of structured data and unstructured content?
- Models: many existing models
- Queries
  - Visual query language
  - Keywords search and Q&A
  - Paths queries – Aggregation queries
- Summarization – Aggregation (K-SNAP)
  - Customized summaries
  - Meaningful summaries for real applications,
  - Efficient and scalable for very large graphs
- Matching with a semantic layer (ontology)

Challenges/open problems

- Searching automatically the interaction between specific objects from RDB knowing that SQL queries cannot find this kind of information.
- Allowing users to analyze easily the resulting graph
- Add semantics
- find communities that not only takes into account links between individuals, but also their similarities based on their own attributes
- Manage the consistency of the graph

Questions/Comments?

Positions available in my group (research engineer/postdoc)

Contact: Marie-Aude.Aufaure@ecp.fr

http://perso.ecp.fr/~aufaurema/

http://www.mas.ecp.fr/BI/New/index.html