Agenda

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

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

By 2013, the amount of traffic 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

Students
- s1
- s2
- s3

Belongs to

Laboratory
- Lab1
- Lab2
- Lab3

develop research on

Topics
- t1
- t2
- t3
Graphs

Simple management of structured, semi-structured and unstructured information

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
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
  - Betweenness: mediators and isolated actors
  - Closeness: proximity

- Identification of 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: « betweenness »
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 an 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

Concepts > 0.1 TagCloud on Topographic Map

map, play, game, http://www.map-

antibes

#antibes

#alpes-maritime

#immobilier, #vente, m2, euros, (06600)

loner
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

Business events

Situation model and recommendations

Events

Queue

Situation

Operators

Activation rules

QuestionProcessor (operator 1)

language id.

tokenizer

stemmer

POS-tagging

focus id.

entity

semantic relations

PatternMatcher (operator 2)

Pattern base

Pattern Learning

AnswerGenerator (operator 3)

Technical query Processing

Visualization Processing

Web-based validation search service

Feedback handler

Graph repository

Repository

Providers

Graphs Gi + schema

Data sources

XML/RDF

SQL

In memory

Extensible components

1st European Business Intelligence Summer School
Modeling the situation as a user-centered graph

- Intention: Close deal
- Has_an: Maggie
- Located_in: Email
- Has_meeting: Meeting
- Starting on: A date
- Reports_to: Mike B.
- Has_author: Press release
- links to: Rapid Planning
- Mentions (confidence xx): Product Announced, Oracle
- Is_a: Company, Kraft, Mike B., Maggie
- Is reading: Email
- Is_an: Account manager
- product: Rapid Planning
- Has_an: Mike B.
Graphs for Business Intelligence – AUFAURE

Graph Repository (GR)

- extensible schema

Providers feed the GR

Social Provider
LDAP, etc.

BI Provider
OLAP cubes, dimensions, measures, dependencies, hierarchies

Text Analysis Provider
Named Entity Recognition and facts extraction

CRM Provider
Customer-related facts

...
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

Subject
Predicate
Object

+ origin: email client (plugin)
+ time validity: timestamp + lifetime
+ security: private / public / ACL
+ confidence
Sample graphs – Universe/cube

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.

Legend
Green – universe/cube
Blue – dimension
Gray – measure
Sample graphs – Social network

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.

Legend
Red – user/agent
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

Recommendations

Social network, BI, ...

People

Tools & apps

Documents

1st European Business Intelligence Summer School
Recommendation in a web context

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

Graphs used to model the recommendation problem

Possible recommendations (dashed edges)
VARIETY OF GRAPHS
Graphs: Definition

A graph $G = (V, E)$ consists of a set $V$ of vertices (also called nodes) and 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 $\Sigma_V$ and $\Sigma_E$ 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 $AG = (V, E, \alpha)$, where $\alpha$ 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 $AG = (V, E, \alpha, \beta)$ where $\beta : E \rightarrow L$ is an edge labeling function.

Coauthor network with 2 attributes: « Topic » and « Age »
Zhou & al, ICDM 2010
An hypergraph $G$ is a tuple $(V, E, \mu)$, 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).
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

1st European Business Intelligence Summer School
A **graph database** is a kind of [NoSQL database](http://en.wikipedia.org/wiki/NoSQL) (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](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

- Fixed Schema
- Relational Algebra and normalized query language
- High-Performance access

Graph Model

- Adapted to the visualization of heterogeneous data
- Flexible schema
- Graphs algorithms and operators (including analysis)
- Easy to update

1st European Business Intelligence Summer School
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 objects. 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]
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.
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:

  - Aggregation Query
    - Simple selection
  - Select Query
    - Selection with conditions
  - Analysis query
    - Path Query
    - Statistics (centrality, etc.)
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.
Example of visual query languages

Select projects where the project manager is « Alain »?

QGraph

Hyperlog

GOOD

1st European Business Intelligence Summer School
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.
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
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

Graph Model Extraction

Identification of objects and their characteristics

Relation Extraction

Object Graph Extraction

Graph Model

Chosen Objects

Enterprise Ontology

Objects Graph

Extraction of a large graph

Aggregation with K-SNAP

Resumed Graph

Relational database
Graphs for Business Intelligence – MA AUFAURE

Relational database

Enterprise Ontology building

Enterprise Documents

Object Identification

Relation Extraction from Ontology

Relation Extraction from Graph model

Object Graph Extraction

Graph Model Extraction

Visual Query Language

Visual Query

Selects nodes or concepts

Pattern Extraction

Graph Pattern Matching

1st European Business Intelligence Summer School
Transformation to a SPIDER-Graph Model from a RDBMS: Schema Translation

<table>
<thead>
<tr>
<th>Employee</th>
<th>Department</th>
<th>Facility</th>
<th>Works On</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Dname</td>
<td>Fid#</td>
<td>Enum#</td>
</tr>
<tr>
<td>LastName</td>
<td>DNO</td>
<td>name</td>
<td>Pnum#</td>
</tr>
<tr>
<td>address</td>
<td>Location</td>
<td>type</td>
<td>Pnum</td>
</tr>
<tr>
<td>DNO#</td>
<td></td>
<td>Price</td>
<td></td>
</tr>
</tbody>
</table>

1. Complex-node creation
2. Relations Identification

1st European Business Intelligence Summer School
Transformation to a SPIDER-Graph Model: Data Migration

(3) Instance level (population)
## (Hidden) Relations Identification

<table>
<thead>
<tr>
<th>Relation</th>
<th>Transformation</th>
</tr>
</thead>
</table>
| R1 \((a, b)\)  
R2\((a #, c)\) | ![Diagram](https://via.placeholder.com/150)

- **Relation**: R1 \((a, b)\)  
  - **Transformation**: a → Type1  
  - **Type**: Type1  
  - **Relation**: R1
- **Relation**: R2\((a #, c)\)
  - **Transformation**: a → R1  
  - **Type**: Type3  
  - **Relation**: R2

<table>
<thead>
<tr>
<th>Relation</th>
<th>Transformation</th>
</tr>
</thead>
</table>
| R1 \((a, b)\)  
R2\((a #, c, e)\) | ![Diagram](https://via.placeholder.com/150)

- **Relation**: R1 \((a, b)\)  
  - **Transformation**: a → Type1  
  - **Type**: Type1  
  - **Relation**: R1
- **Relation**: R2\((a #, c, e)\)
  - **Transformation**: a → R1  
  - **Type**: Type3  
  - **Relation**: R2
  - **Transformation**: e → Type4

---

1st European Business Intelligence Summer School
(Hidden) Relations identification

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

![Diagram showing relations and transformations](image)
Objects identification

Selected concepts: EMPLOYEE, PROJECT

Corresponding complex-Nodes:

Objects added to the final graph

1st European Business Intelligence Summer School
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 $O_I := \{o_l \mid o_l \in N_{CN}\}$
  
  $R_O$ is a finite set of relations between objects such as
  
  $R_O := \{r \mid r := (l, o_{l1}, O_{l2})$ where $l$ is the relation name and
  
  $o_{l1}, O_{l2} \in O_I$

- **Relation pattern**: $P_r = (n, o_{l1}, o_{l2}, o_m)$
  
  - $n$ = name of the relation
  - $o_{l1}, o_{l2}$ are entities sharing the relation
  - $o_m$ is a mediator for this relation (complex object used to find a new relation)
### Patterns to identify new objects

<table>
<thead>
<tr>
<th>Initial Relation</th>
<th>Process</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Definition</strong></td>
<td>CN₀ or CN_d instances are added to Oᵢ</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>Manager instances are added to Oᵢ</td>
</tr>
</tbody>
</table>

**IS-A Relation**

**Input Relation (Schema)**

**Detected Objects (instances)**
## Patterns based on relation between chosen objects.

<table>
<thead>
<tr>
<th>Initial Relation</th>
<th>Pattern</th>
<th>Process and description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_2 := \langle r, CN_s, CN_d \rangle$</td>
<td>$Pr_1 := \langle r, CN_{si}, CN_{di}, null \rangle$ where $CN_{si}$ and $CN_{di}$ are $CN_s$ and $CN_d$ instances.</td>
<td>Find all the existing relations between $CN_s$ and $CN_d$ instances then add them to $R_O$.</td>
</tr>
<tr>
<td>where $CN_s$ and $CN_d$ instances $\in O_I$</td>
<td>$Pr_2 := \langle \text{Same}<em>\text{(CN_d.name)}, CN</em>{si1}, CN_{si2}, CN_d \rangle$ where $CN_{si1}$ and $CN_{si2}$ are $CN_s$ instances and $i!=j$</td>
<td>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)”</td>
</tr>
</tbody>
</table>
| $- \text{ CN_s and CN_d instances } \in O_I$ | $Pr_{path} := \langle '', CN_{si}, CN_{di}, P \rangle$ 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 |
Patterns based on relation between chosen objects.

Pr2:=<Same_(CNd.name),CNSIi, CNSIj, CNd >  where CNSIi and CNSIj are CNs instances and i!=j
## Graphs for Business Intelligence – MA AUFAURE

**Initial Relation**

\[ R_3 := \langle r, CN_s, CN_d \rangle \]
where \( CN_s \in O_l \) and \( CN_d \notin O_l \)

**Pattern**

\[ Pr_3 := \langle \text{Same}_{(CN_d}\text{.name)}, CN_{SI_i}, CN_{SI_j}, CN_d \rangle \]
where \( CN_{SI_i}, CN_{SI_j} \) are \( CN_S \) instances, \( i \neq j \) and \( r \neq \text{“Part-of”} \)

**Process and description**
Find all the \( CN_s \) instances which have relations with a same \( CN_d \) instances and link them with a new relation

**Initial Relation**

\[ R_4 := \langle r, CN_s, CN_d \rangle \]
where \( CN_s \notin O_l \) and \( CN_d \in O_l \)

**Pattern**

\[ Pr_4 := \langle \text{Same}_{CN_k}\text{.name}, CN_{SI_i}, CN_{SI_j}, CN_k \rangle \]
where \( CN_{SI_i}, CN_{SI_j} \) are \( CN_S \) instances, \( i \neq j \), \( r = \text{“Part-of”} \) and \( CN_k \in \{CN\mid CN \text{ has the relation } R := \langle \text{“Part-of”}, CN, CN_d \rangle \} \)

**Process and description**
1. Find all the complex-nodes \( CN_k \) having a “Part-of” relation with \( CN_d \) such as \( R := \langle \text{“Part-of”}, CN_k, CN_d \rangle \)
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

**Initial Relation**

\[ R_5 := \langle r, CN_s, CN_{d_l}, CN_{j_l}, CN_s \rangle \]
where \( CN_{d_l} \) is \( CN_d \) instance and \( CN_{j_l} \) is instance of \( CN_j \) where \( CN_j \in O_l \cap \{CN\mid CN \text{ has the relation } R := \langle \text{“”, CN, CN_s} \rangle \} \)

**Pattern**

\[ Pr_5 := \langle \text{Same}_{CN_s}\text{.name}, CN_{dl}, CN_{jl}, CN_s \rangle \]

**Process and description**
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.
Patterns based on relation between a chosen object and another Complex node

Input Relation

New Relations

\[ Pr_3 := \langle \text{Same}_{\text{(Department)}}, \text{Employee}_i, \text{Employee}_j, \text{Department}_d \rangle \]

where \( i \neq j \)
Object Graph Extraction
Relation Construction-6-

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

Input Relation

New Relations

Pr₄:=<Same_Product, Projectᵢ, Projectⱼ, Product_Projectₖ >
where” and i!=j
Patterns based on relation between a chosen object and another Complex node

Input Relation

New Relations

Pr₅ :=<Same_Facility, Projectᵢ, Projectᵢ, Facilityₛ >
The Object Graph between Employee and Project

**Objects:** Employee and Project
Graph Aggregation

Relational database

Graph Model Extraction

Graph Model

Chosen Objects

Object Graph Extraction

Identification of objects and their characteristics

Relation Extraction

Enterprises Ontology

Objects Graph

Extraction of a large graph

Aggregation with K-SNAP

Resumed Graph

1st European Business Intelligence Summer School
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.
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 Nodes on Attributes and Pairwise relationships.
SNAP Operation
SNAP Operation

1st European Business Intelligence Summer School
SNAP Operation
Why k-SNAP?

A-Compatible

Too few

k-SNAP

Roll-up

Too many

k-SNAP

(A,R)-Compatible

Drill-down
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)

1\textsuperscript{st} 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

2\textsuperscript{nd} iteration: subdivision of the Prof group into 2 subgroups according to the relation
  - Same\_Laboratory
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

⇒ 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