Advanced topics in Databases

Hellenic Mediterranean University

Prof. Demos Akoumianakis (da@hmu.gr)

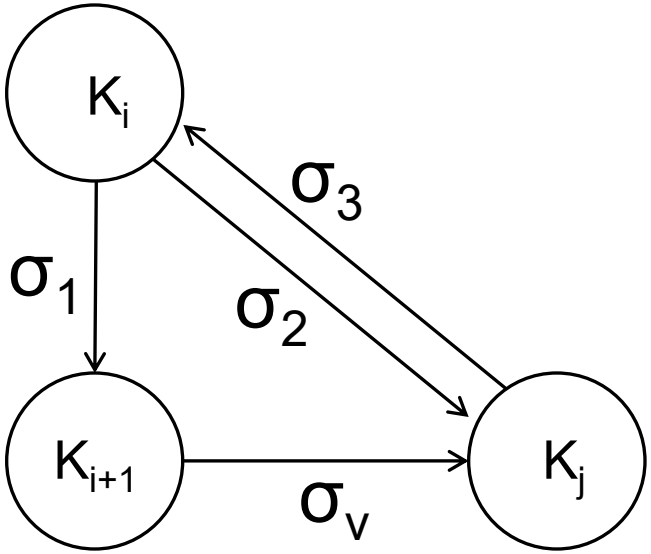
Progress so far

- ✓ *Object-relational extensions*
  - *New data types*
    - *Enum, ARRAYS, Composite Data Types, XMLTABLE, JSON*
  - *Inheritance in Postgres*
- ✓ *Hierarchical data, Nested sets, Recursion*
  - *Column data stores*
  - *MongoDB*

- Today
  - Graph data models
    - Principles
    - Label Property Graph
    - Representing graphs
    - Path queries

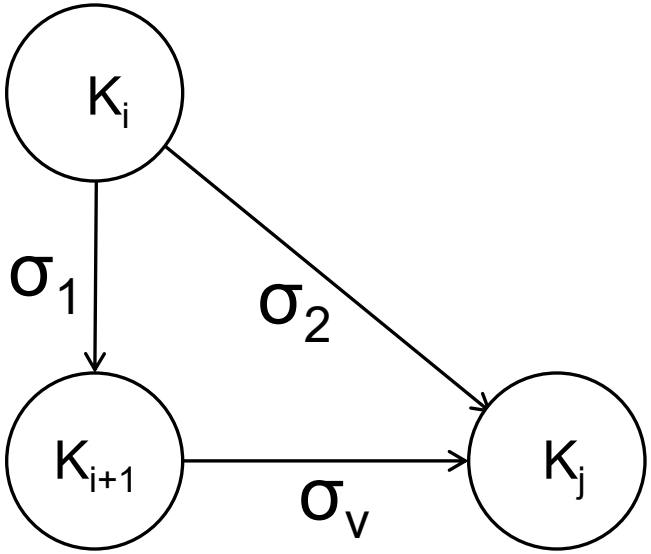
Definition

- A graph G is a set of nodes K which are connected with edges S
  - In other words G = (K, S) where for each K_i, $K_j \in K$ and $\sigma_k \in S$, it holds σ_k (K_i , K_j)



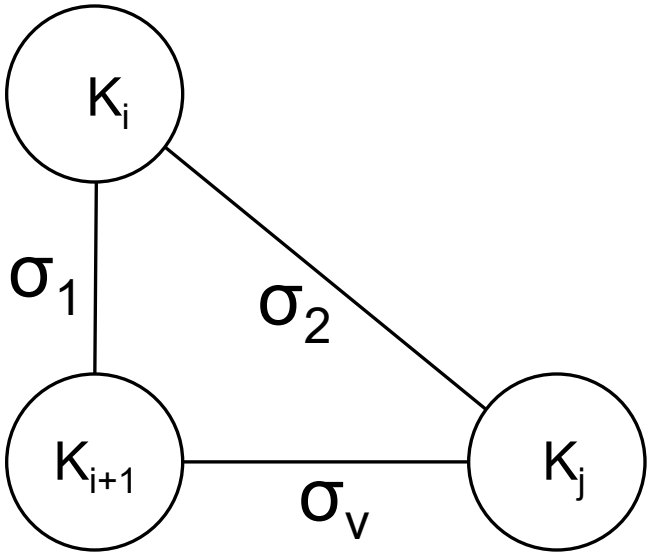
Types of graphs

- Several types, including
  - Directed (Κατευθυνόμενοι)
    - Edges specify direction (from one node to another)



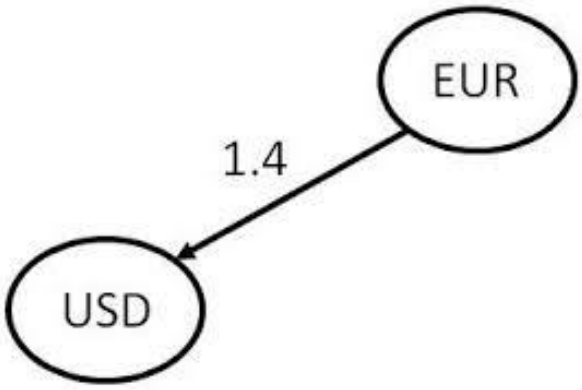
Types of graphs (cont.)

- Several types, including
  - Directed (Κατευθυνόμενοι)
    - Edges specify direction (from one node to another)
  - Undirected (Μη-κατευθυνόμενοι)
    - Edges do not specify direction



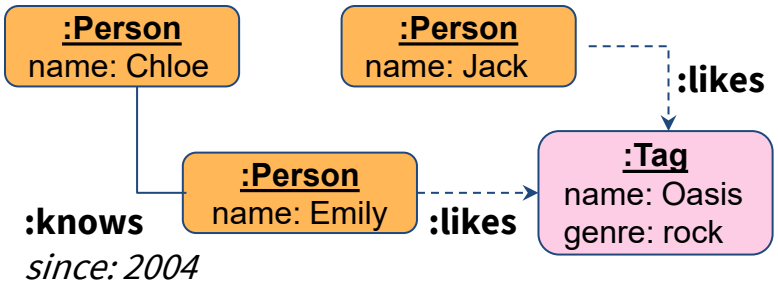
Types of graphs (cont.)

- Several types, including
  - Directed (Κατευθυνόμενοι)
    - Edges specify direction (from one node to another)
  - Undirected (Μη-κατευθυνόμενοι)
    - Edges do not specify direction
  - Weighted graphs (ανισοβαρή)
    - Edges may differ (in value, properties)

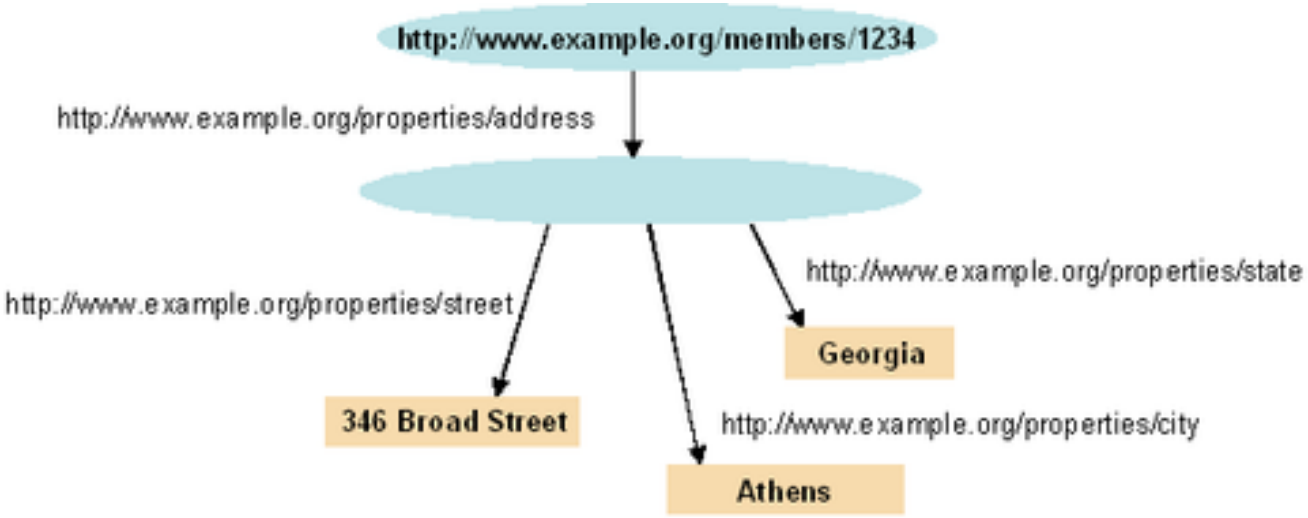


Types of graphs (cont.)

- Labelled property graph

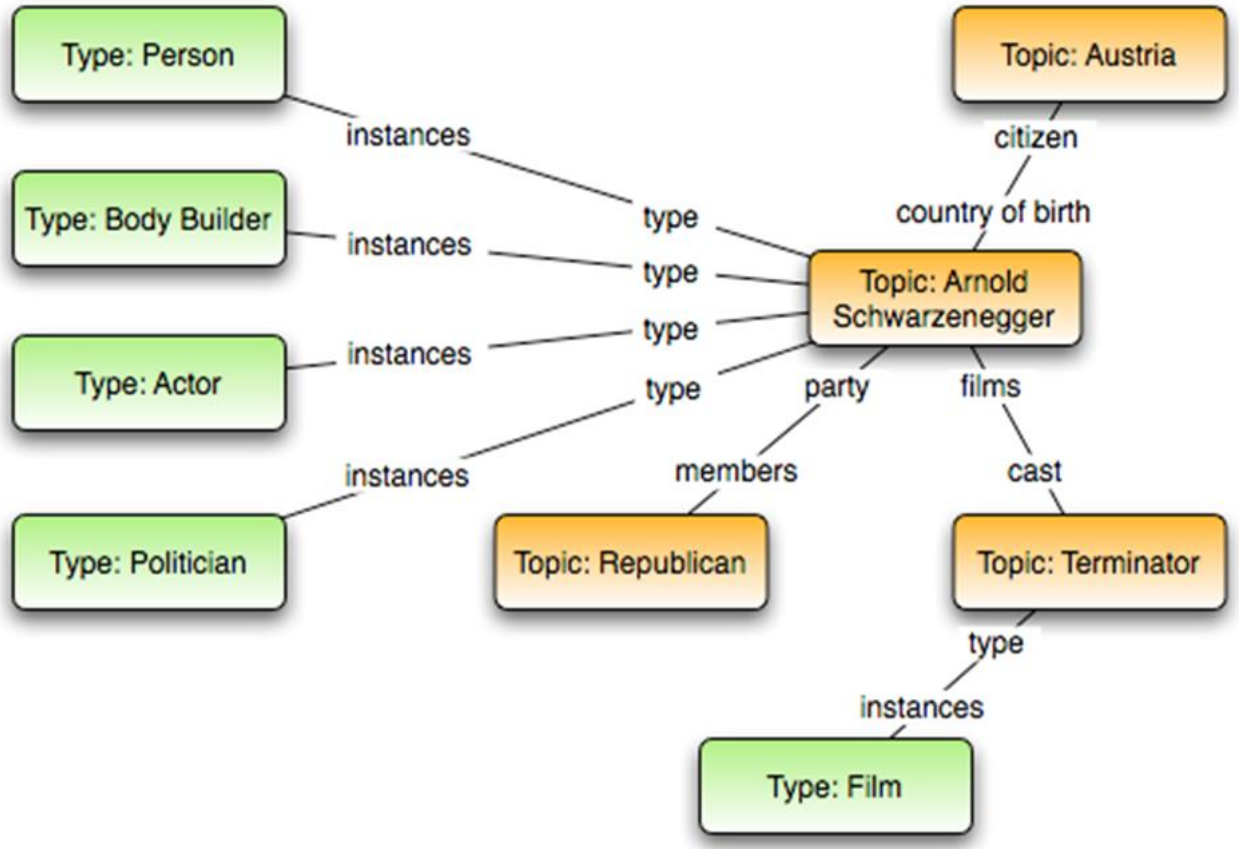


- Resource Description Framework (RDF)



Types of graphs (cont.)

- Other specialized types such as the Freebase graph type



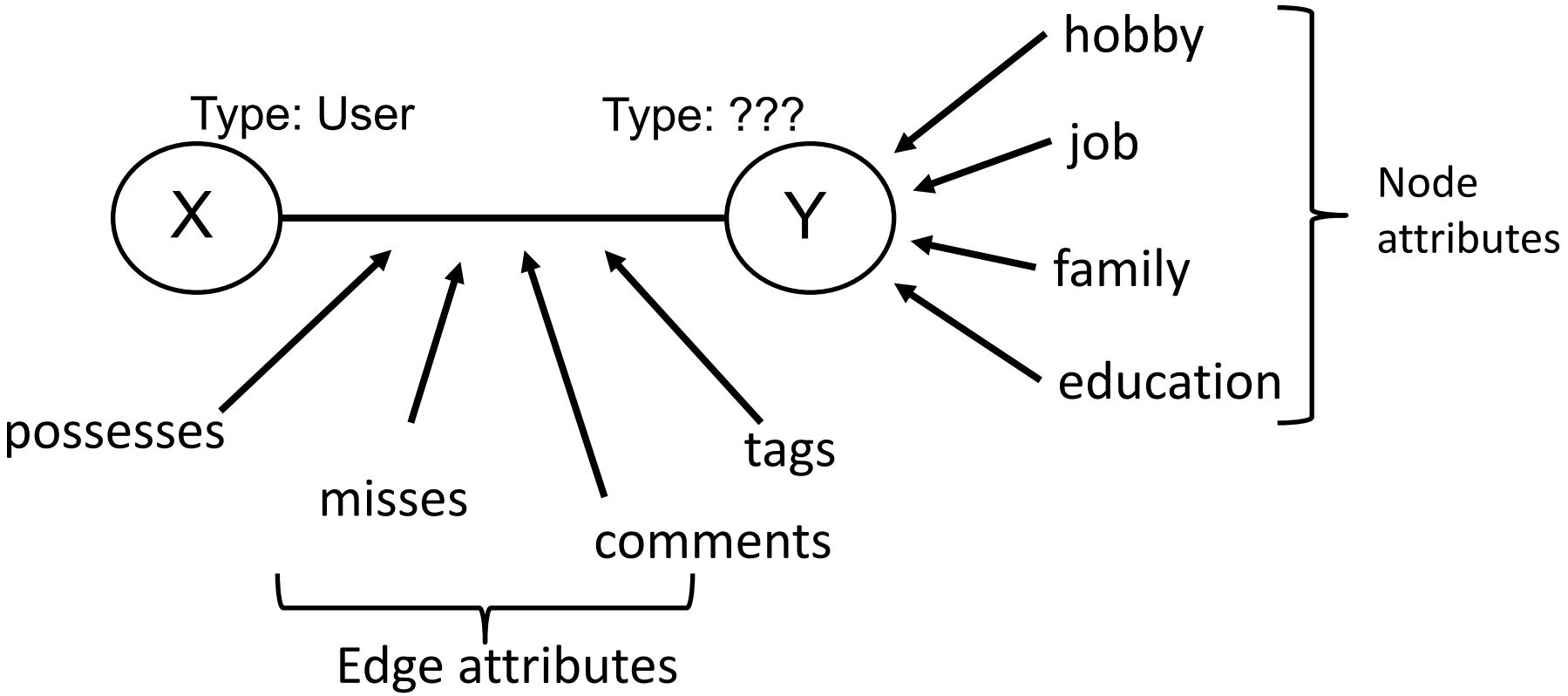
Property graphs

Property graphs

- A special type of graph with specific characteristics
  - Labelled graph which is made up from
    - nodes characterized by properties
      - key-value pairs
    - relationships which
      - may have properties and
      - specify direction (source/destination node)

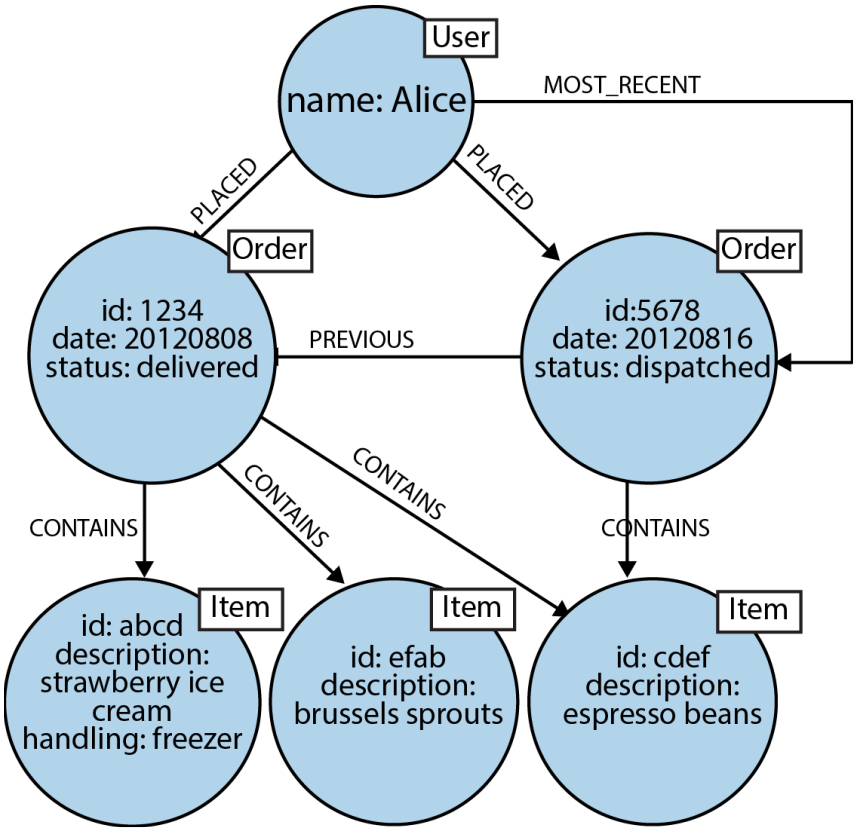
Property graphs (cont.)

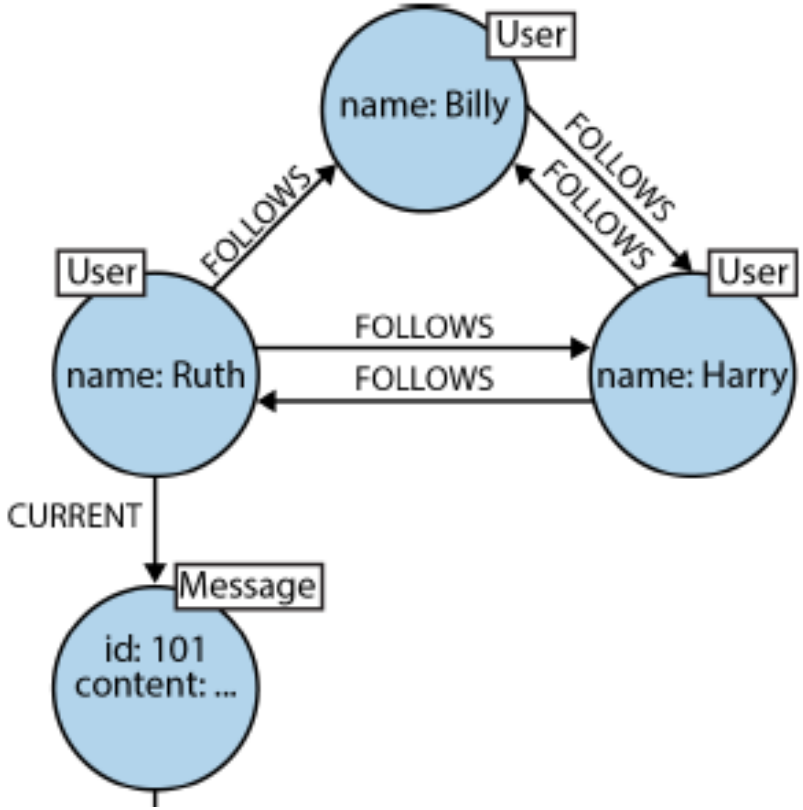
- Nodes of different types and edges/connections between them



Examples

- Labelled property graphs

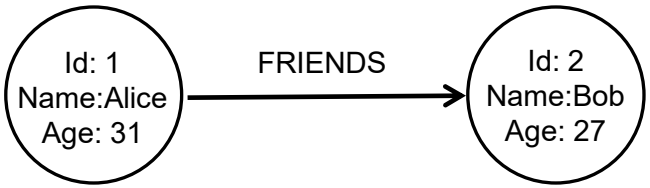




Representing graphs

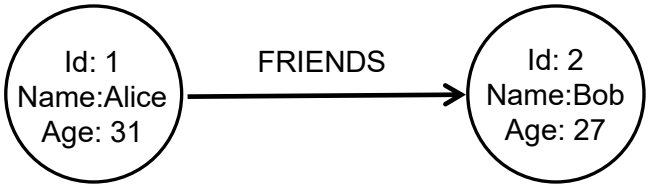
Techniques

- Alternative representations
  - JSON document
  - Adjacency list/matrix
  - Normalized schema / Unnormalized schema



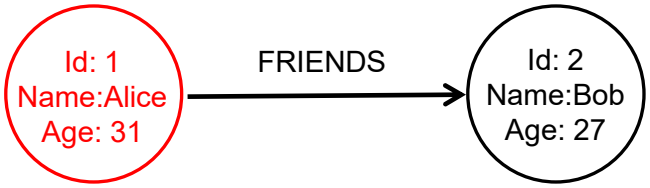
JSON format

- As JSON document



JSON format – vertices

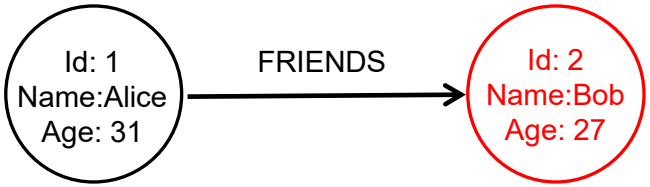
- JSON for nodes




```
{
   "graph": {
      "vertices": [
         {
            "name": "Alice",
            "age": 31,
            "_id": "1",
            "_type": "vertex"
         },
         {
            "name": "Bob",
            "age": 27,
            "_id": "2",
            "_type": "vertex"
         } …
}
}
```

JSON format – vertices (cont.)

- JSON for nodes




```
{
   "graph": {
      "vertices": [
         {
            "name": "Alice",
            "age": 31,
            "_id": "1",
            "_type": "vertex"
         },
         {
            "name": "Bob",
            "age": 27,
            "_id": "2",
            "_type": "vertex"
         } …
}
}
```

JSON format – edges

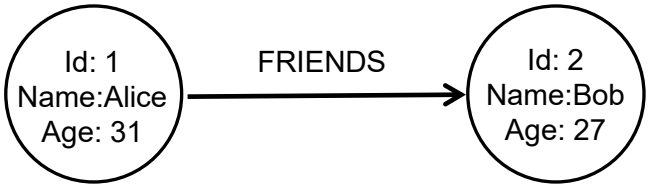
- JSON for edges


```
{
   "graph": {
      "vertices": [
         {
            "name": "Alice",
            "age": 31,
            "_id": "1",
            "_type": "vertex"
         },
         {
            "name": "Bob",
            "age": 27,
            "_id": "2",
            "_type": "vertex"
         } …
}
}
```

```
{
   "graph": {
      "edges": [
         {
            "type": "friends",
            "_id": "3",
            "_type": "edge",
            "_outV": "1",
            "_inV": "2",
            "_label": "knows"
         }
      ]
   }
}
```

An extract of a graph

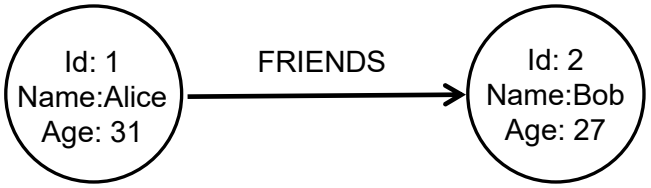
- The property graph as JSON




```
{
  "graph": {
    "vertices": [
      {
        "name": "Alice",
        "age": 31,
        "_id": "1",
        "_type": "vertex"
      },
      {
        "name": "Bob",
        "age": 27,
        "_id": "2",
        "_type": "vertex"
      }
    ],
    "edges": [
      {
        "type": "friends",
        "_id": "3",
        "_type": "edge",
        "_outV": "1",
        "_inV": "2",
        "_label": "knows"
      }
    ]
  }
}
```

GML format

- Graph Modeling Language (GML)

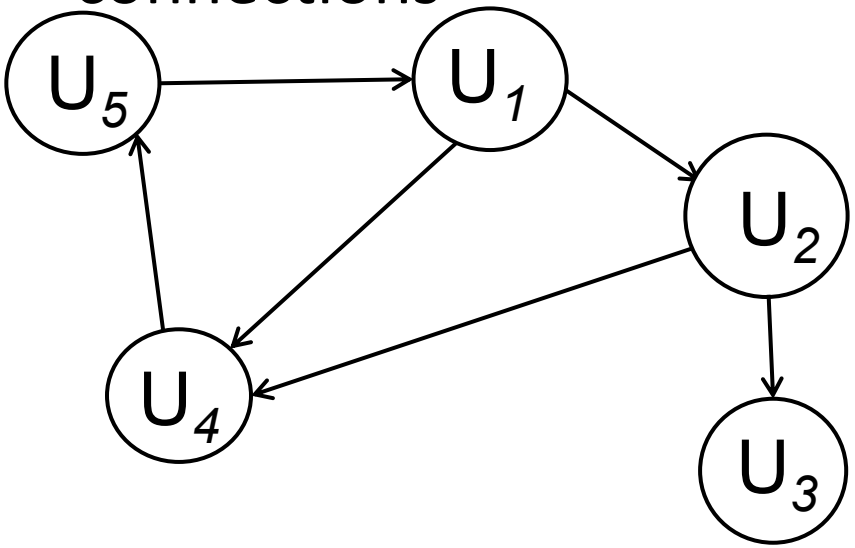



```
graph [
  comment "Simple property graph"
  directed 1
  IsPlanar 1
  node [
    id 1
    label "1"
    name "Alice"
    age 31
     ]
  node [
    id 2
    label "2"
    name "Bob"
    age 27
     ]
  edge [
    source 1
    target 2
    label "knows"
    type "friends"
     ]
  ]
```

Adjacency techniques

Adjacency matrix (Πίνακας γειτνίασης)

- MxM matrix
  - M are the modes and boolean value for representing connections



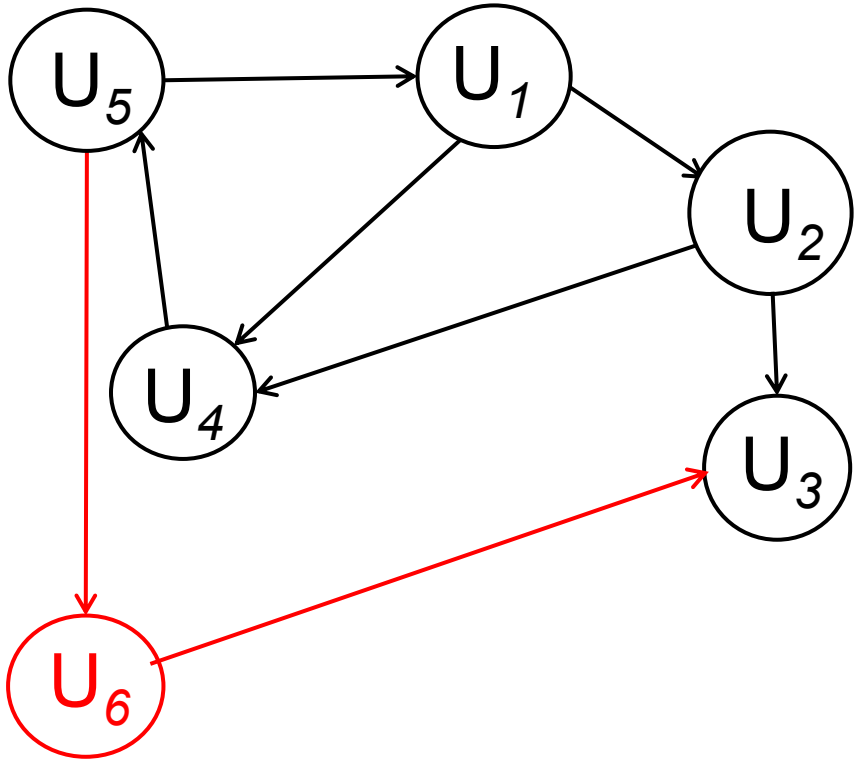
Destination

Source	1	2	3	4	5
1	0	1	0	1	0
2	0	0	1	1	0
3	0	0	0	0	0
4	0	0	0	0	1
5	1	0	0	0	0

- Matrix is altered as the graph is updated!!!

Adjacency matrix (cont.)

- Updating the graph with one node and a few edges

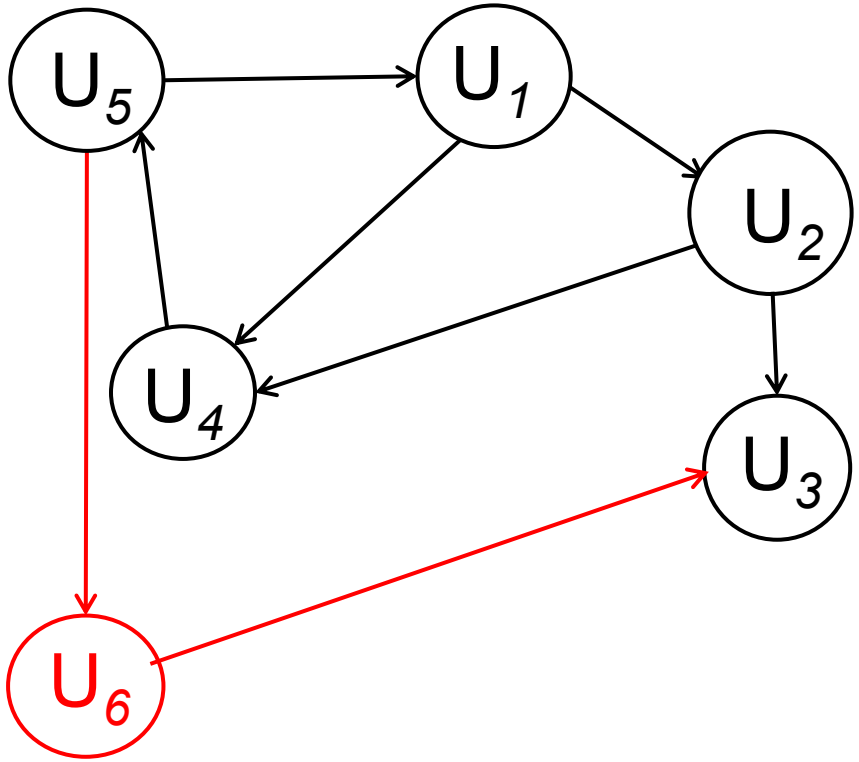


Destination

Source	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	1	1	0	0
3	0	0	0	0	0	0
4	0	0	0	0	1	0
5	1	0	0	0	0	1
6	0	0	1	0	0	0

Adjacency matrix (cont.)

- Not suitable for classical relational representation due to variable length



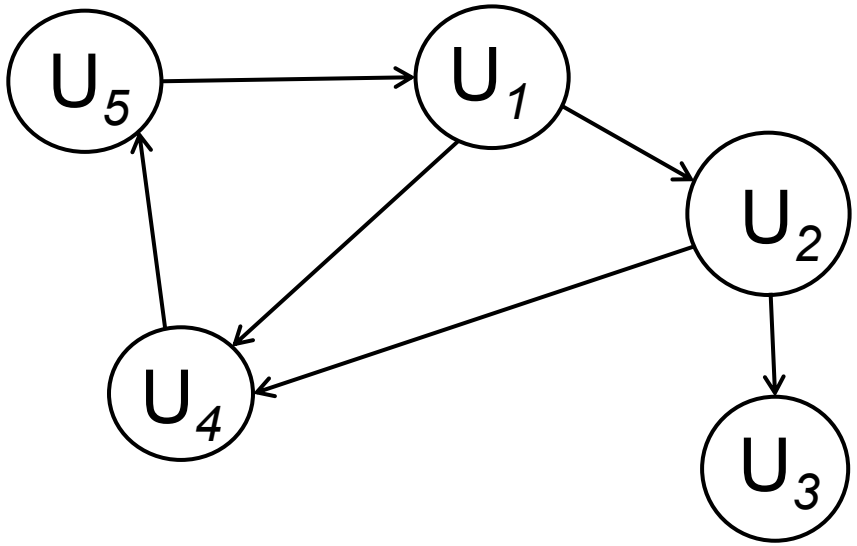
Destination

Source	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	1	1	0	0
3	0	0	0	0	0	0
4	0	0	0	0	1	0
5	1	0	0	0	0	1
6	0	0	1	0	0	0

Adjacency list

Principle

- For each node we list its the neighbors
  - In the example we have up to two neighbors



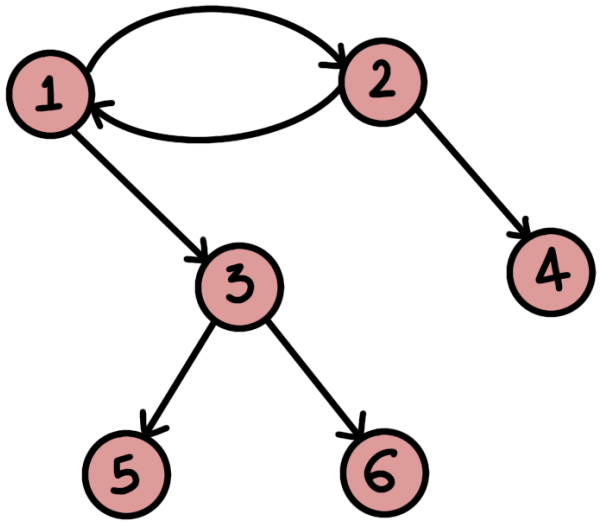
Κόμβος	1ος επόμενος	2ος επόμενος
1	2	4
2	3	4
3	NULL	
4	5	
5	1	

- It can be represented using array

Relational schema using ARRAY

- One-dimensional array

```
CREATE TABLE graph (
    vertex_id INT,
    vertices INT []
    PRIMARY KEY(vertex_id),
);
```



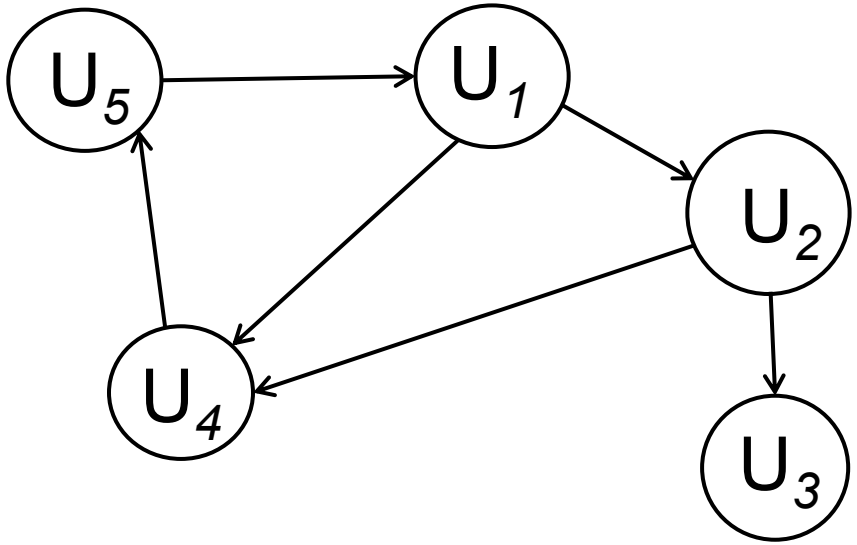
1, [2, 3]
2, [1, 4]
3, [5, 6]

Vertex	Vertices
1	[2,3]
2	[1,4]
3	[5,6]
4	NULL
5	NULL
6	NULL

Normalized relational schema

Normalized relational schema

- One table for nodes and another table for edges (with direction)

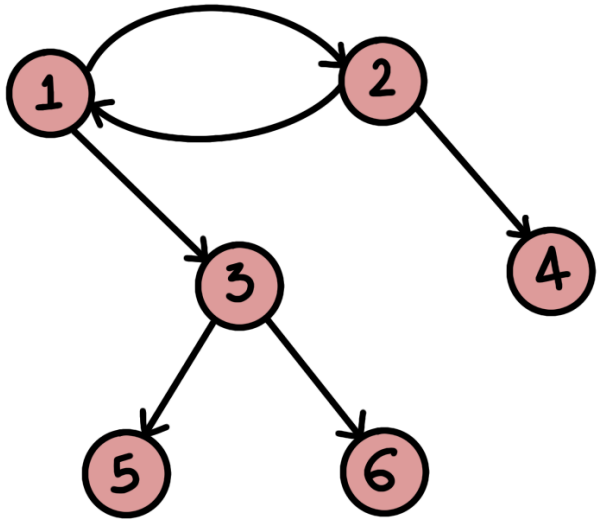


Κόμβος	Start	End
1	1	2
1	1	4
2	2	3
2	2	4
3	3	NULL
4	4	5
5	5	1

Normalized relational schema (cont.)

- Two-table schema

```
CREATE TABLE vertex (
    vertex_id INT,
    PRIMARY KEY(vertex_id)
    // Other columns
);
CREATE TABLE graph (
    vertex1_id INT,
    vertiex2_id INT,
    FOREIGN KEY(vertex1_id)
            REFERENCES vertex(vertex_id),
    FOREIGN KEY(vertex2_id)
            REFERENCES vertex(vertex_id),
    PRIMARY KEY(vertex1_id, vertex2_id)
);
```



[(1, 2),
(1, 3),
(2, 1),
(2, 4),
(3, 5),
(3, 6)]

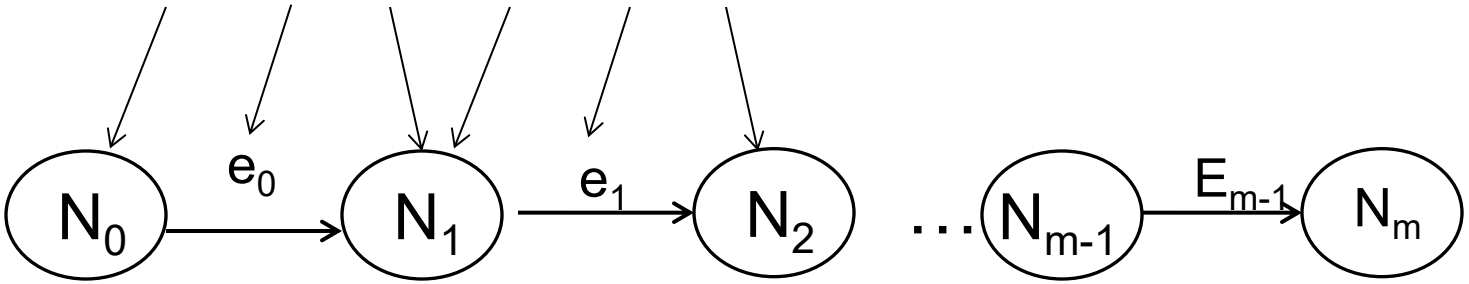
Vertexs1_id	Vertex2_id
1	2
1	3
2	1
2	4
3	5
3	6

Path queries

Definitions – Path

- A path $\pi_{Node_k \rightarrow Node_j}$ in graph G is a sequence such as

$$(N_0, e_0, N_1), (N_1, e_1, N_2), \cdots, (N_{m-1}, e_{m-1}, N_m)$$

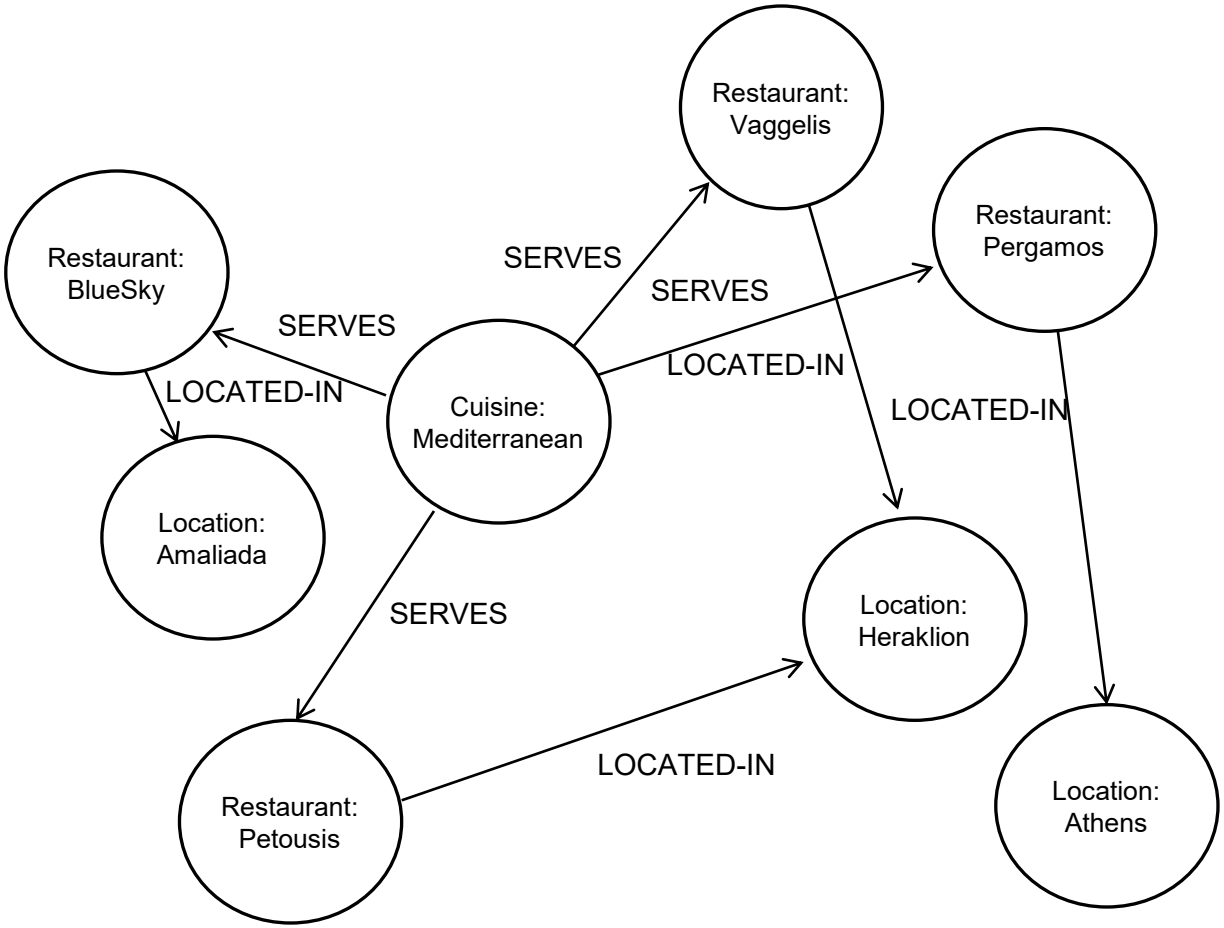


Path queries

- For a graph G = (N, E) with nodes N and edges E, a path query (*ερώτημα τροχιάς*) Q $[\alpha]_{Ni - Nj}$ is defined as the set of nodes and edges fro node N_i to node N_j while satisfying criterion [α]

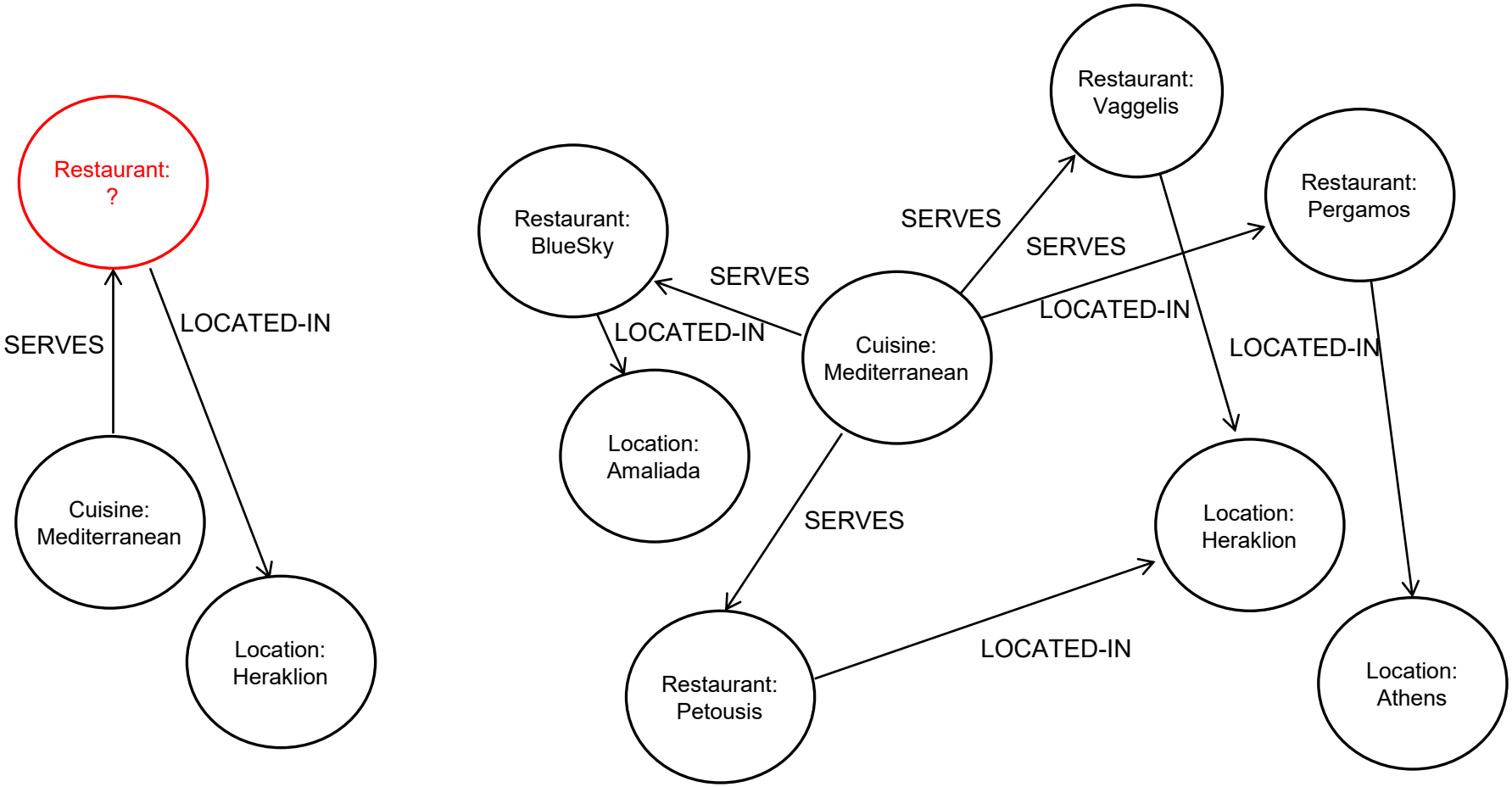
Example

- Consider the following property graph



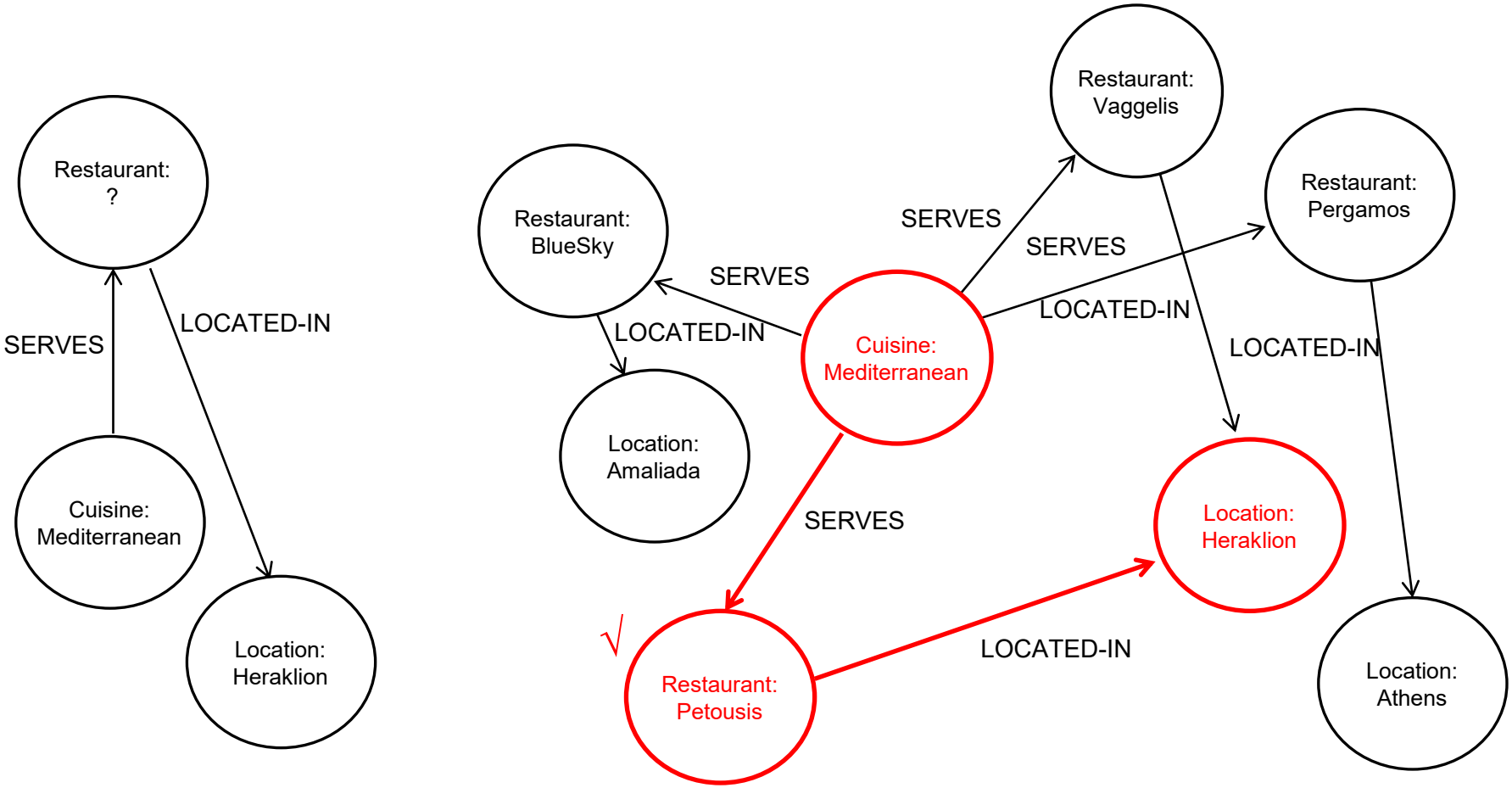
Example

- Query ‘Which restaurants in Heraklion serve Mediterranean cousin’



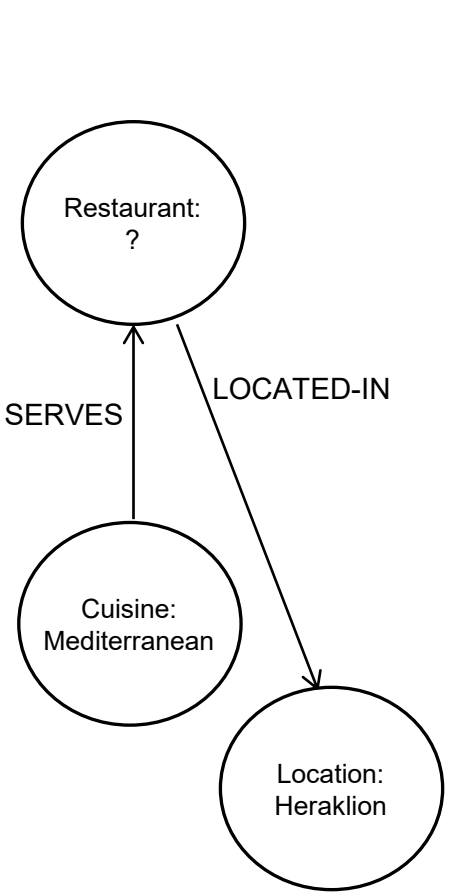
Example (cont.)

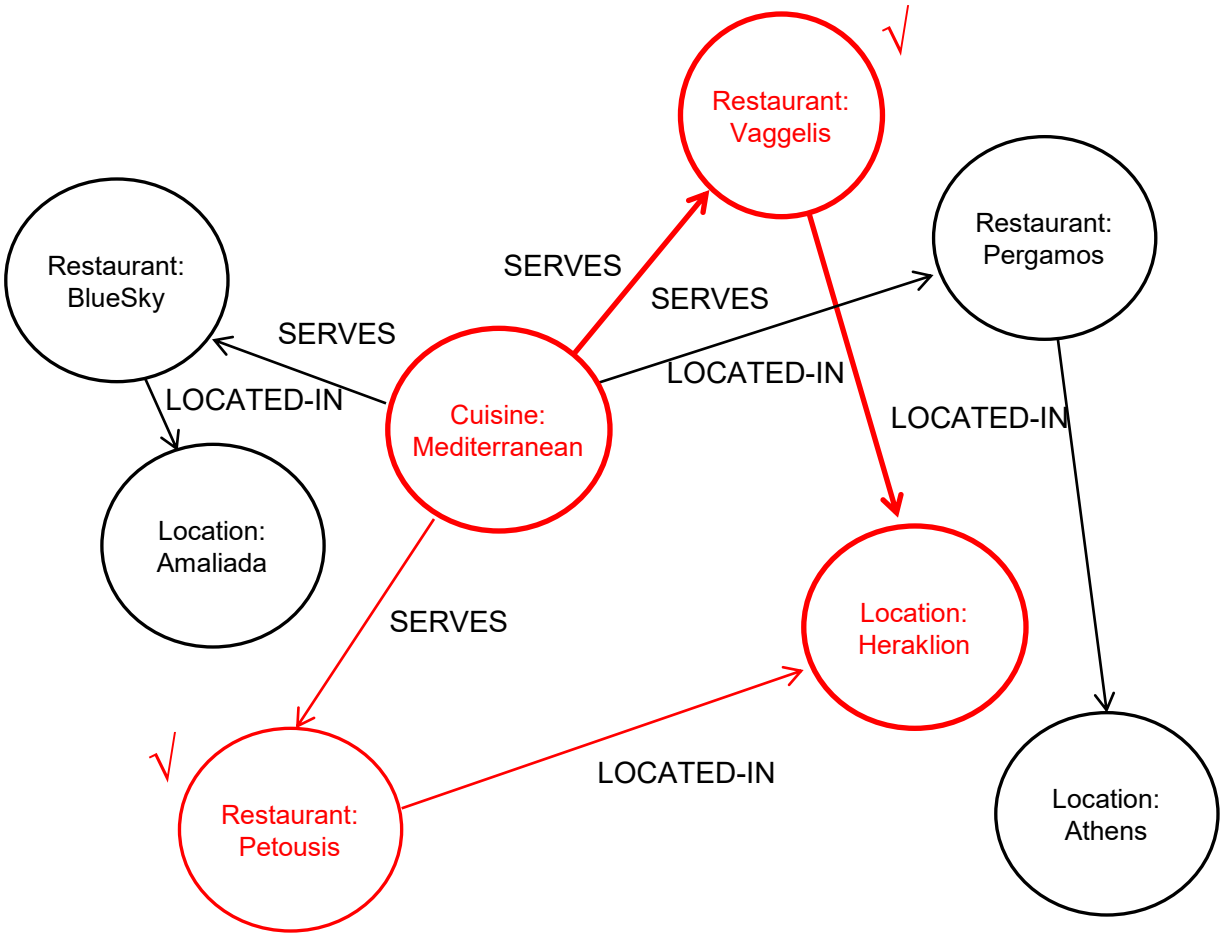
- Matching nodes & edges for ‘Which restaurants in Heraklion serve Mediterranean cousin’



Example (cont.)

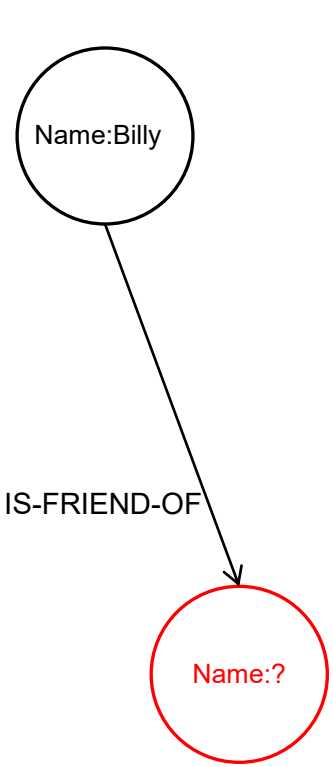
- Matching nodes & edges for ‘Which restaurants in Heraklion serve Mediterranean cousin’

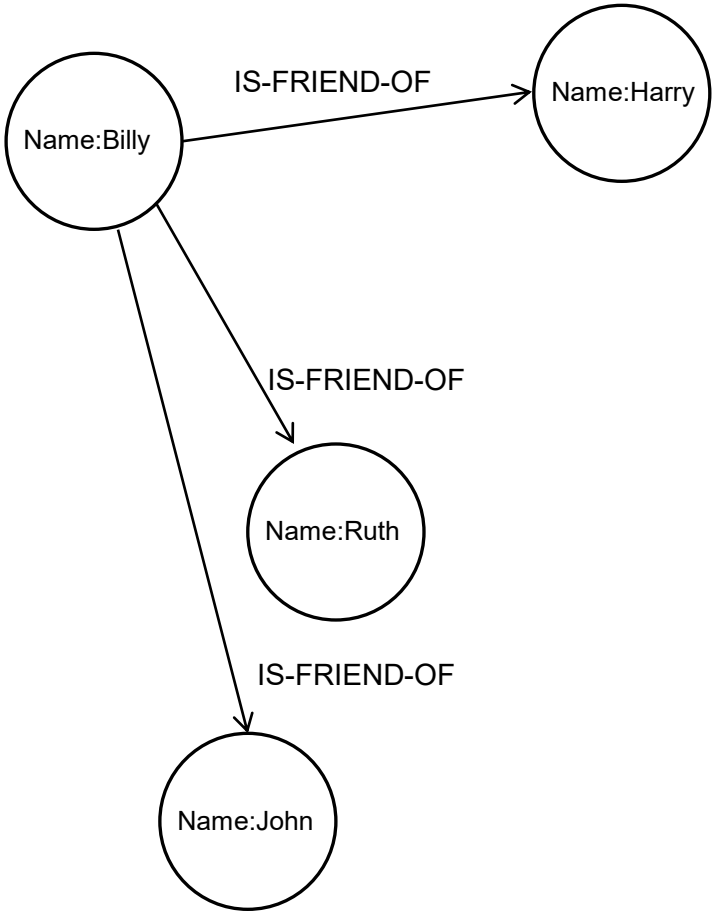




Example (cont.)

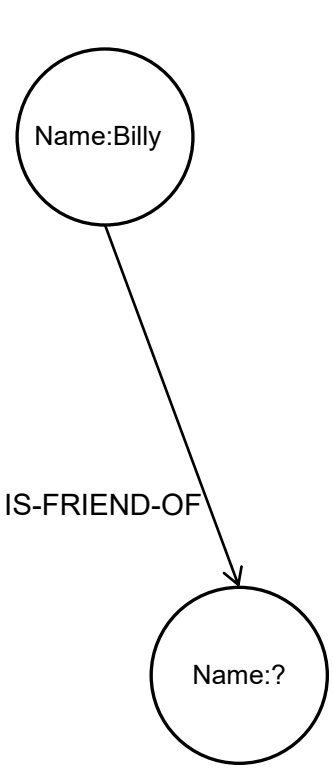
- Query ‘Find all friends of Billy’

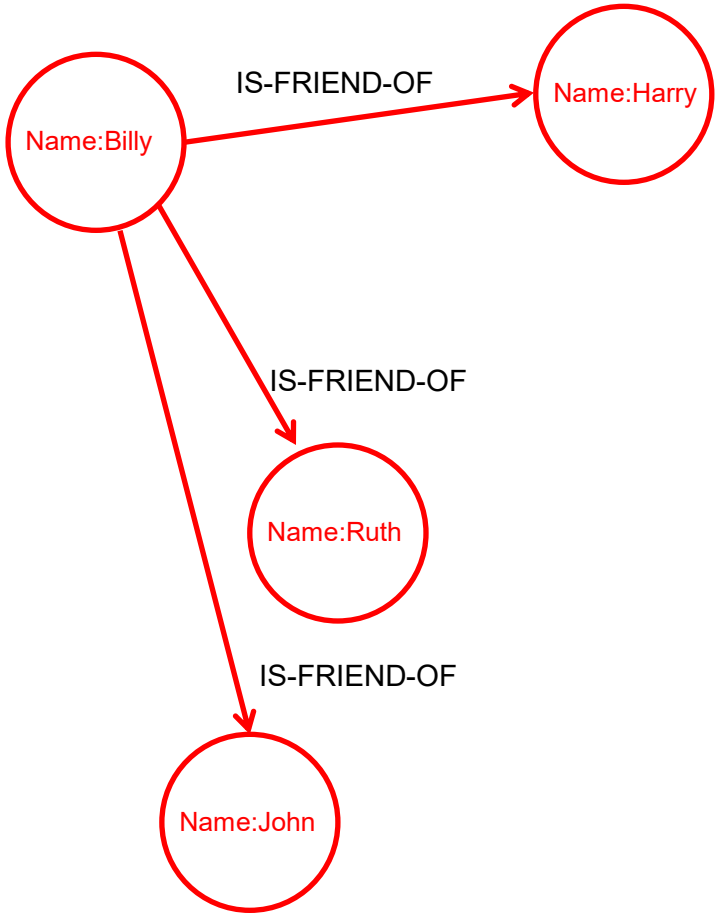




Example (cont.)

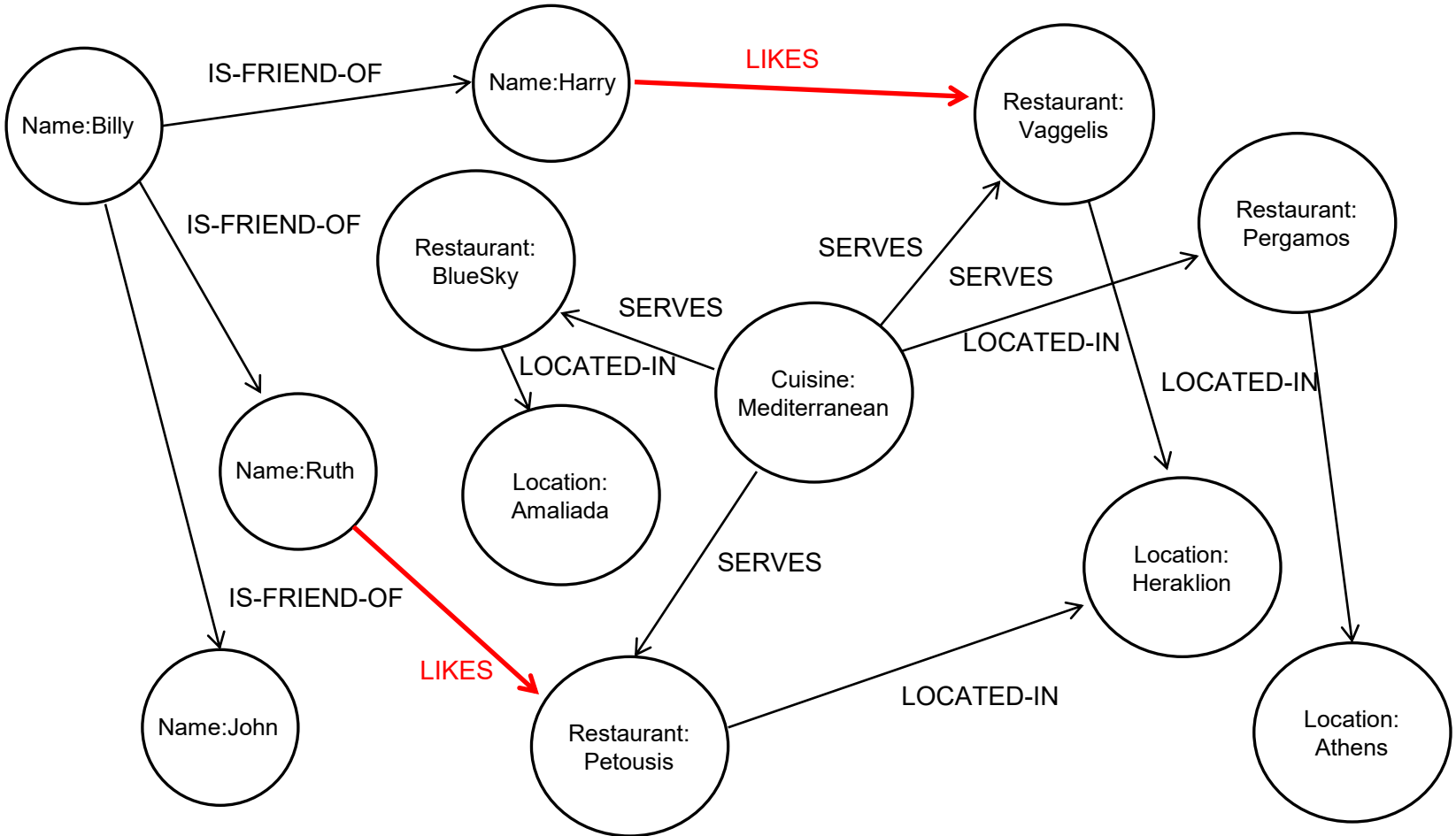
- Query ‘Find all friends of Billy’





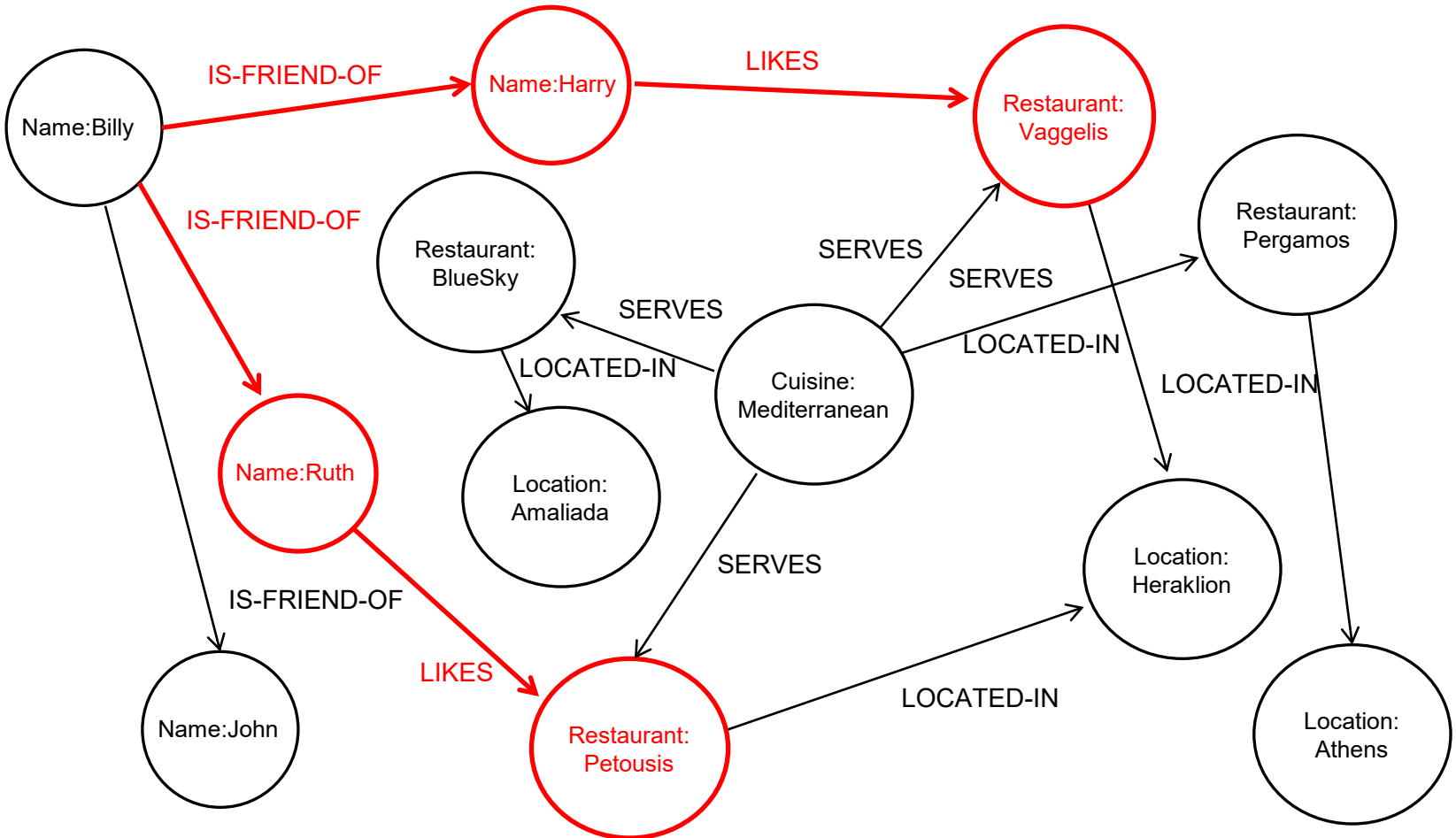
Example (cont.)

- If users declare preferences …



Example (cont.)

- If users declare preferences how would we find ‘What Billy’s friends like?’



From relations to graphs

Migration

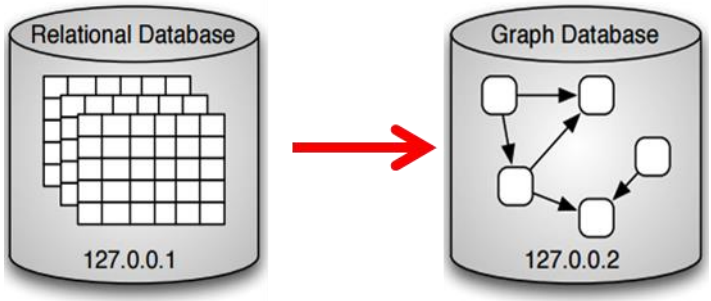
- Data *migration* is the structured process of permanently transferring data from one computer storage system, computing environment, or data format to another
  - Unlike continuous data *replication* or *integration*, data migration is typically a one-time project with a defined beginning and end, aimed at decommissioning the legacy source system once the data is safely validated in its new home

Why migration

- Several reasons
  - Cloud Adoption
    - Moving from legacy on-premises servers to cloud environments like Microsoft Azure or AWS.
  - System Upgrades
    - Replacing outdated infrastructure or upgrading to new versions of databases and applications
  - Mergers and Acquisitions
    - Combining disparate IT systems and databases from two separate companies into a single, unified environment

Issue of (primary) concern

- How can we map relational data to property graph, documents, or any other data model
  - No standard approach but important to be consistent
- We will concentrate on graphs and review two techniques
  - Orthodox
  - Schema graph



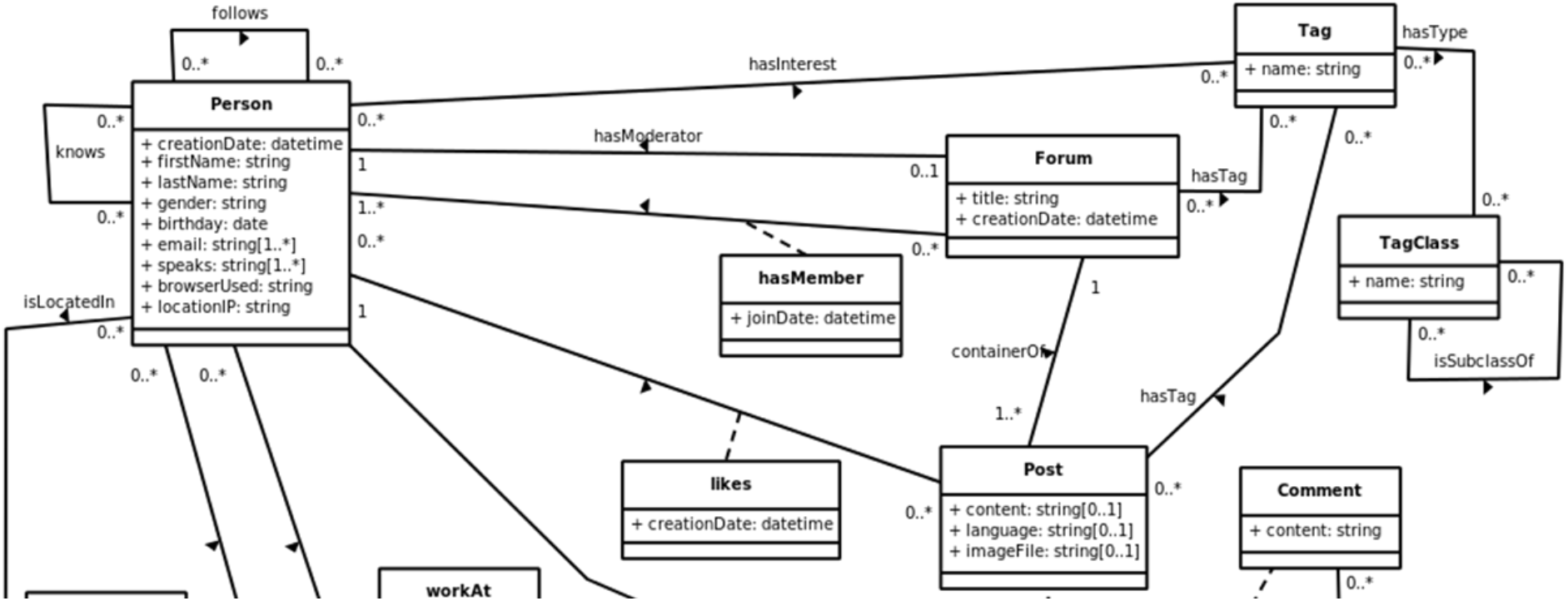
The two techniques (in summary)

- Orthodox
  - Focuses on data and ignores the structure of the schema
  - It is called orthodox because it leads to a labelled property graph
- Schema graph
  - Focuses on the schema and the integrity constraints

Orthodox leading to a canonical property graph

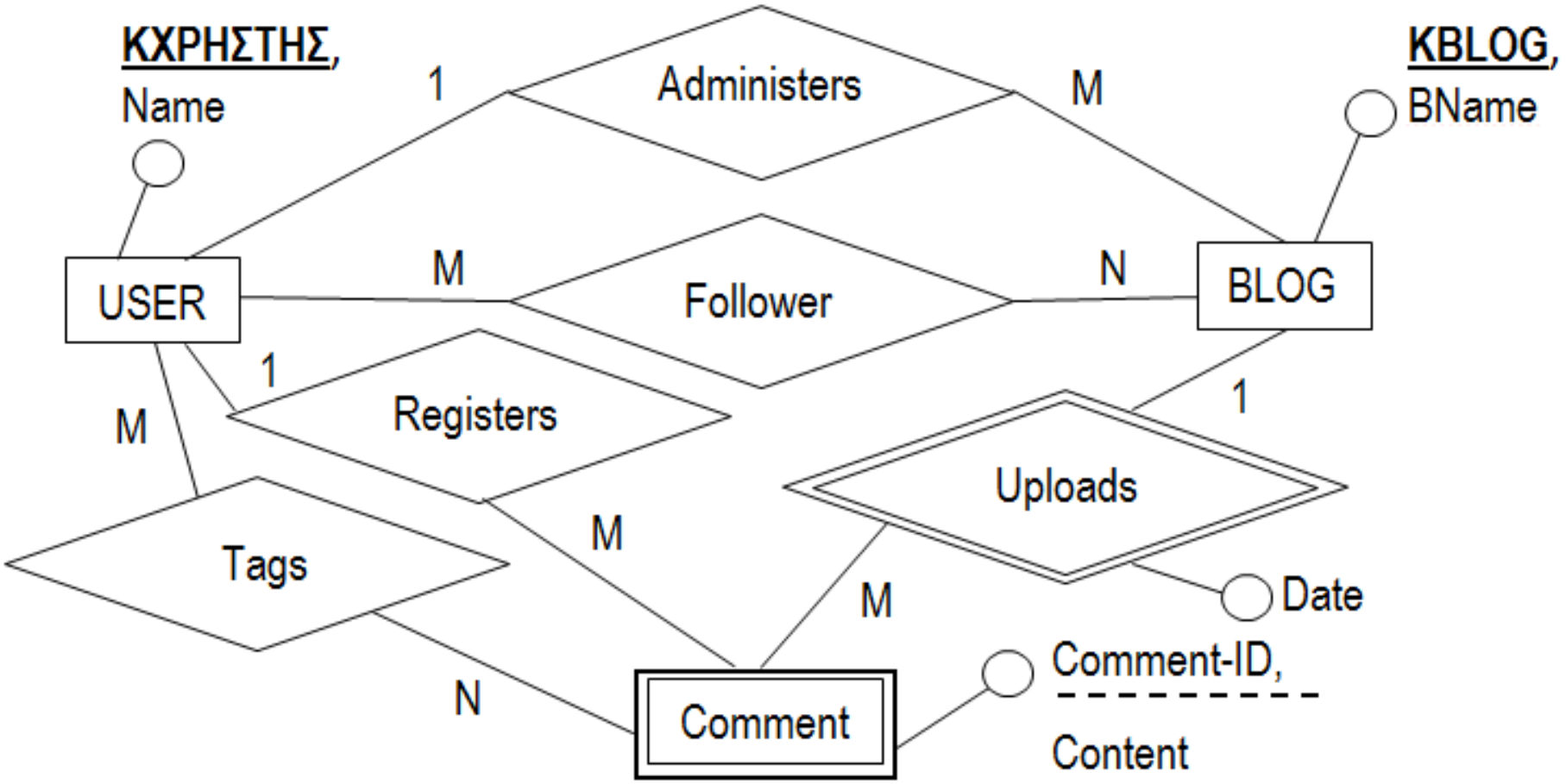
Reference case

- Consider the following



Reference case as E-R

- Equivalent ER representation



Reference case as relational tables

- The resulting relational schema

User (US)

	uid	uname
t_1	u01	Date
t_2	u02	Hunt

Follower (FR)

	fuser	fblog
t_3	u01	b01
t_4	u01	b02
t_5	u01	b03
t_6	u02	b01

Tag (TG)

	tuser	tcomment
t_7	u02	c01

Blog (BG)

	bid	bname	admin
t_8	b01	Information Systems	u02
t_9	b02	Database	u01
t_{10}	b03	Computer Science	u02

Comment (CT)

	cid	cblog	cuser	msg	date
t_{11}	c01	b01	u01	Exactly what I was looking for!	25/02/2013

Question

- How can we derive an equivalent graph (in our case a property graph);

User (US)

	uid	uname
t_1	u01	Date
t_2	u02	Hunt

Follower (FR)

	fuser	fblog
t_3	u01	b01
t_4	u01	b02
t_5	u01	b03
t_6	u02	b01

Tag (TG)

	tuser	tcomment
t_7	u02	c01

Blog (BG)

	bid	bname	admin
t_8	b01	Information Systems	u02
t_9	b02	Database	u01
t_{10}	b03	Computer Science	u02

Comment (CT)

	cid	cblog	cuser	msg	date
t_{11}	c01	b01	u01	Exactly what I was looking for!	25/02/2013

Orthodox approach

- Focusing on data – table USER

User (US)

	uid	uname
t_1	u01	Date
t_2	u02	Hunt

Orthodox approach (cont.)

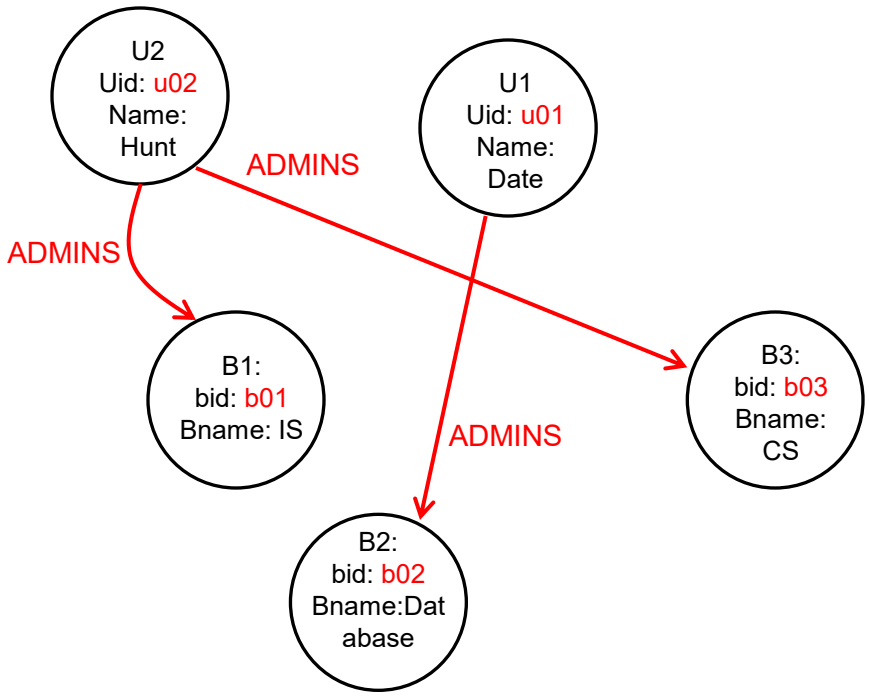
- Focusing on data – table BLOG

User (US)

	uid	uname
t_1	u01	Date
t_2	u02	Hunt

Blog (BG)

	bid	bname	admin
t_8	b01	Information Systems	u02
t_9	b02	Database	u01
t_{10}	b03	Computer Science	u02

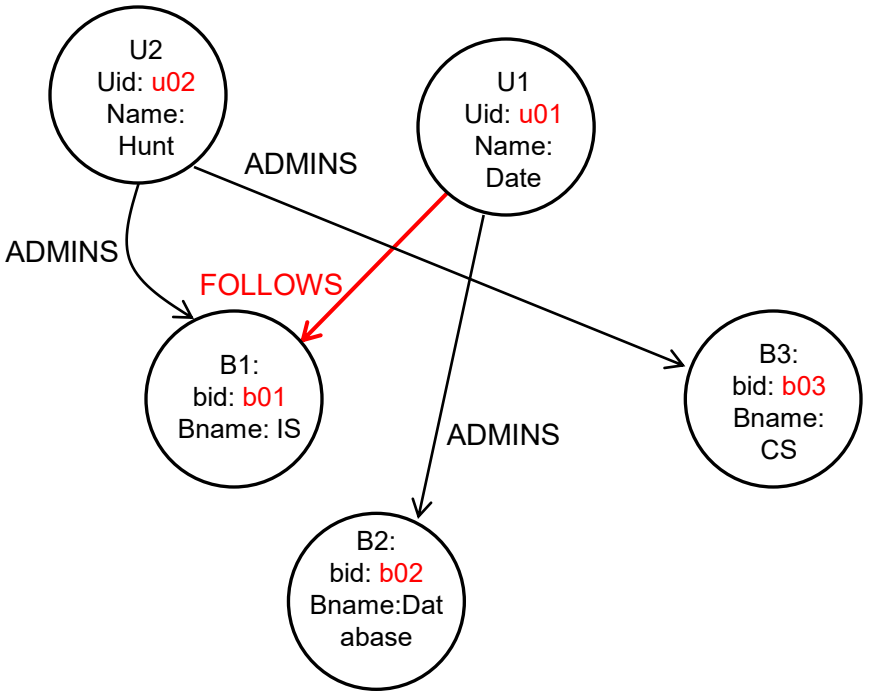


Orthodox approach (cont.)

- Focusing on data – table FOLLOWER

Follower (FR)

	fuser	fblog
t_3	u01	b01
t_4	u01	b02
t_5	u01	b03
t_6	u02	b01

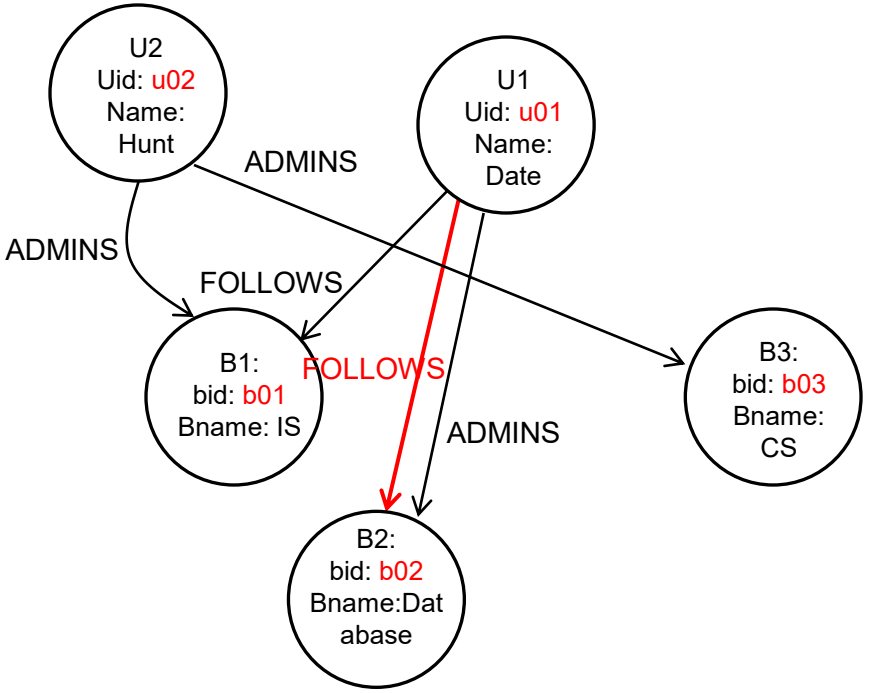


Orthodox approach (cont.)

- Focusing on data – table FOLLOWER

Follower (FR)

	fuser	fblog
t_3	u01	b01
t_4	u01	b02
t_5	u01	b03
t_6	u02	b01

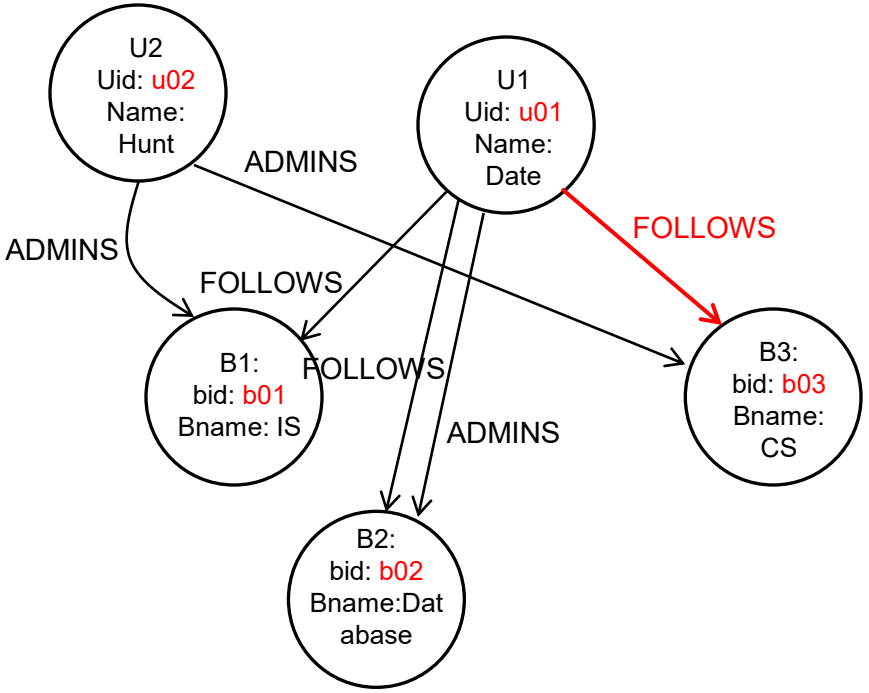


Orthodox approach (cont.)

- Focusing on data – table FOLLOWER

Follower (FR)

	fuser	fblog
t_3	u01	b01
t_4	u01	b02
t_5	u01	b03
t_6	u02	b01

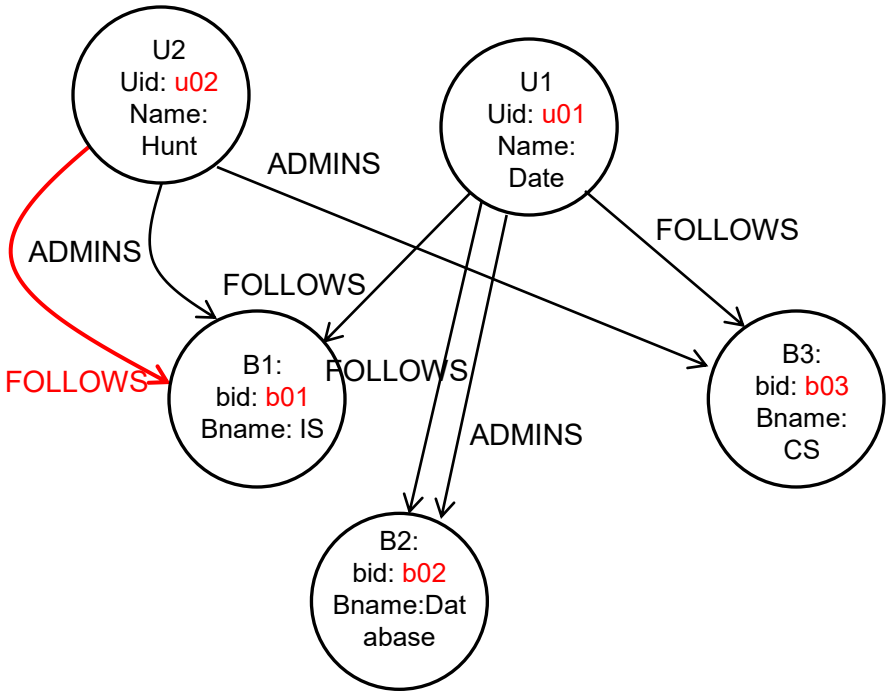


Orthodox approach (cont.)

- Focusing on data – table FOLLOWER

Follower (FR)

	fuser	fblog
t_3	u01	b01
t_4	u01	b02
t_5	u01	b03
t_6	u02	b01

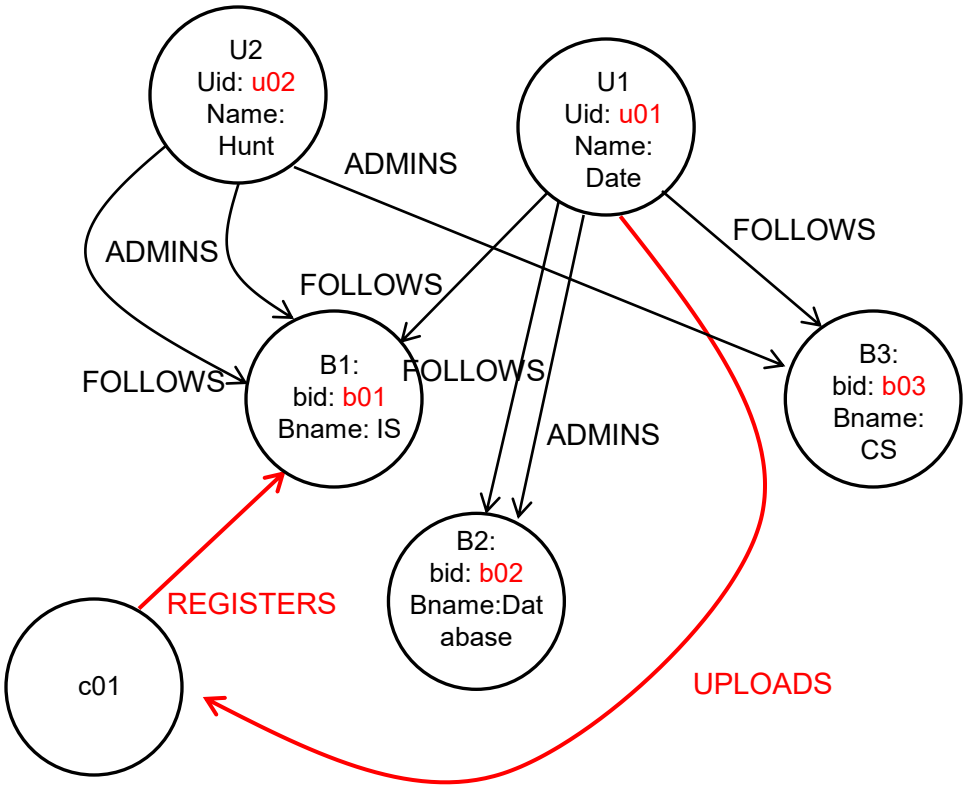


Orthodox approach (cont.)

- Focusing on data – table FOLLOWER

Comment (CT)

	cid	cblog	cuser	msg	date
t_{11}	c01	b01	u01	Exactly what I was looking for!	25/02/2013

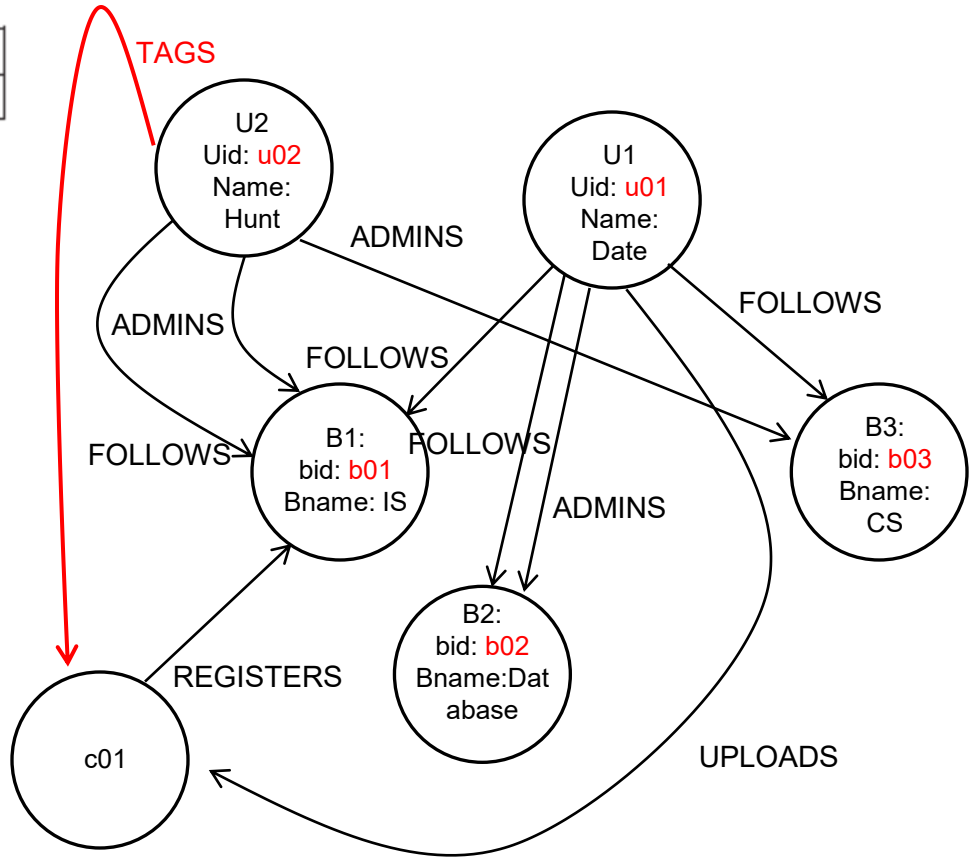


Orthodox approach (cont.)

- Focusing on data – table TAG

Tag (TG)

	tuser	tcomment
t_7	u02	c01



Consolidation

- From data to a canonical property graph

User (US)

	uid	uname
t_1	u01	Date
t_2	u02	Hunt

Follower (FR)

	fuser	fblog
t_3	u01	b01
t_4	u01	b02
t_5	u01	b03
t_6	u02	b01

Tag (TG)

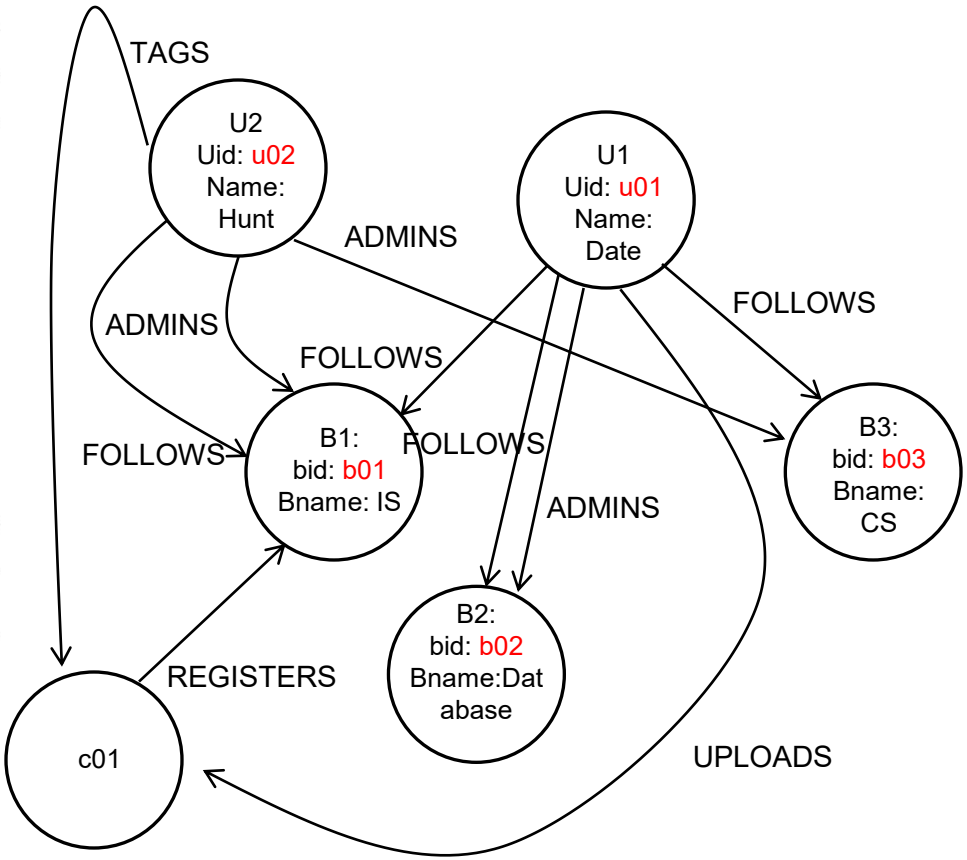
	tuser	tcomment
t_7	u02	c01

Blog (BG)

	bid	bname	admin
t_8	b01	Information Systems	u02
t_9	b02	Database	u01
t_{10}	b03	Computer Science	u02

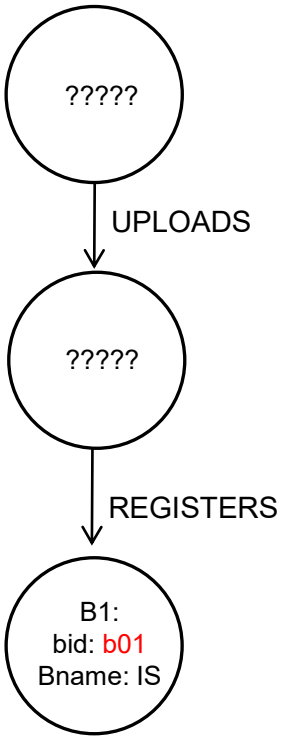
Comment (CT)

	cid	cblog	cuser	msg	date
t_{11}	c01	b01	u01	Exactly what I was looking for!	25/02/2013



Queries

- Find the users who post comments on blog ‘b01’;



User (US)

	uid	uname
t_1	u01	Date
t_2	u02	Hunt

Follower (FR)

	fuser	fblog
t_3	u01	b01
t_4	u01	b02
t_5	u01	b03
t_6	u02	b01

Tag (TG)

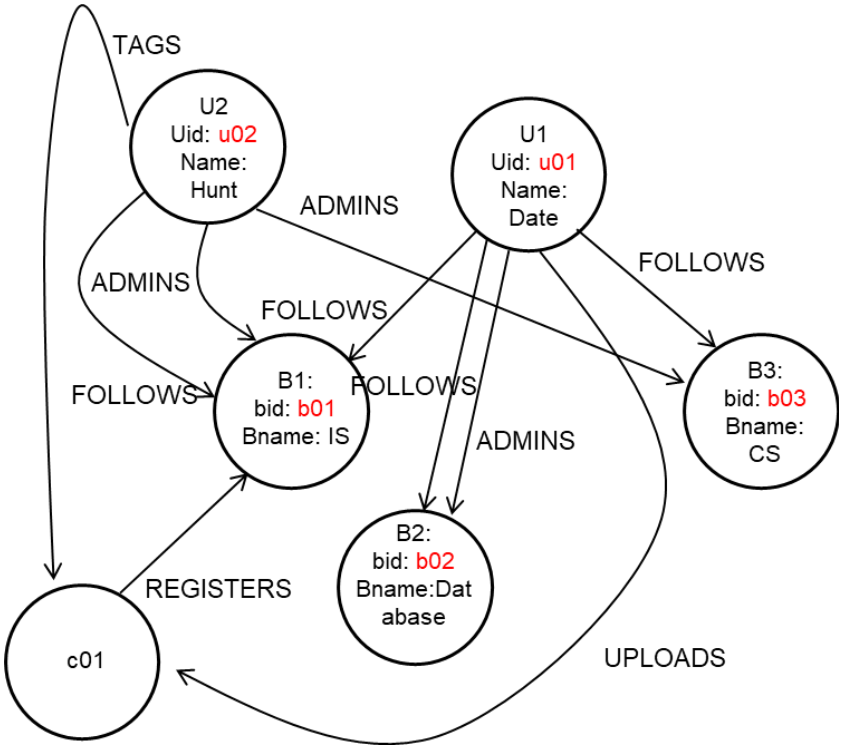
	tuser	tcomment
t_7	u02	c01

Blog (BG)

	bid	bname	admin
t_8	b01	Information Systems	u02
t_9	b02	Database	u01
t_{10}	b03	Computer Science	u02

Comment (CT)

	cid	cblog	cuser	msg	date
t_{11}	c01	b01	u01	Exactly what I was looking for!	25/02/2013



Queries (cont.)

- ✓ *Find the users who post comments on blog 'b01';*
- Find the users who tag comments posted on blog 'b01';

User (US)

	uid	uname
t_1	u01	Date
t_2	u02	Hunt

Follower (FR)

	fuser	fblog
t_3	u01	b01
t_4	u01	b02
t_5	u01	b03
t_6	u02	b01

Tag (TG)

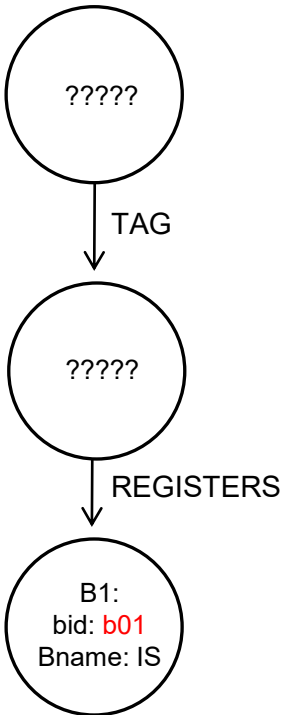
	tuser	tcomment
t_7	u02	c01

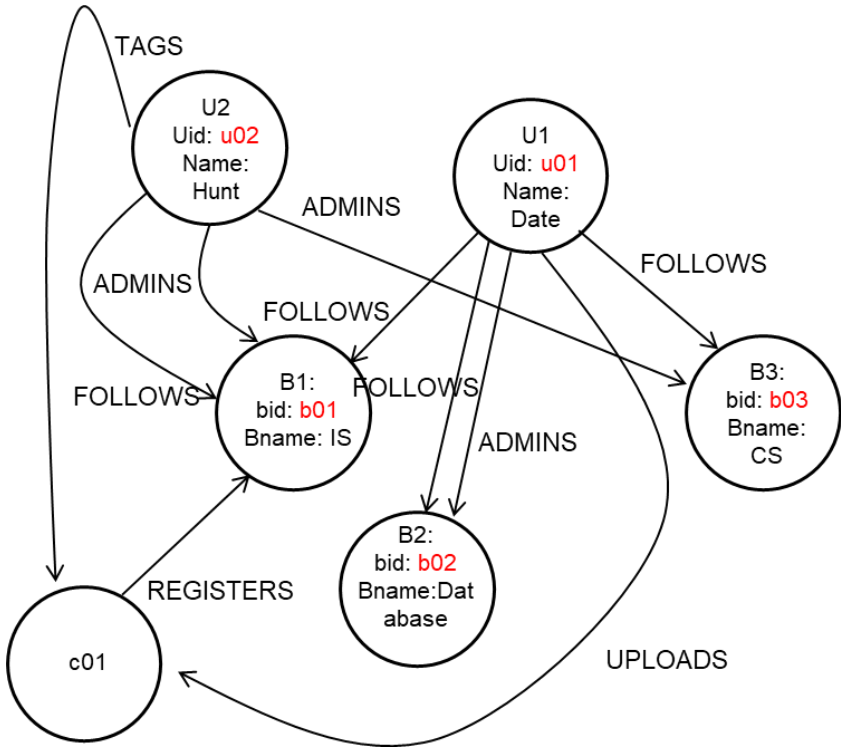
Blog (BG)

	bid	bname	admin
t_8	b01	Information Systems	u02
t_9	b02	Database	u01
t_{10}	b03	Computer Science	u02

Comment (CT)

	cid	cblog	cuser	msg	date
t_{11}	c01	b01	u01	Exactly what I was looking for!	25/02/2013





Queries (cont.)

- ✓ *Find the users who post comments on blog 'b01';*
- ✓ *Find the users who tag comments posted on blog 'b01';*
- Find users who either post comments on blog 'b01' or tag comments posted on 'b01';
  - Union (∪) of previous two sets of results

User (US)

	uid	uname
t_1	u01	Date
t_2	u02	Hunt

Follower (FR)

	fuser	fblog
t_3	u01	b01
t_4	u01	b02
t_5	u01	b03
t_6	u02	b01

Tag (TG)

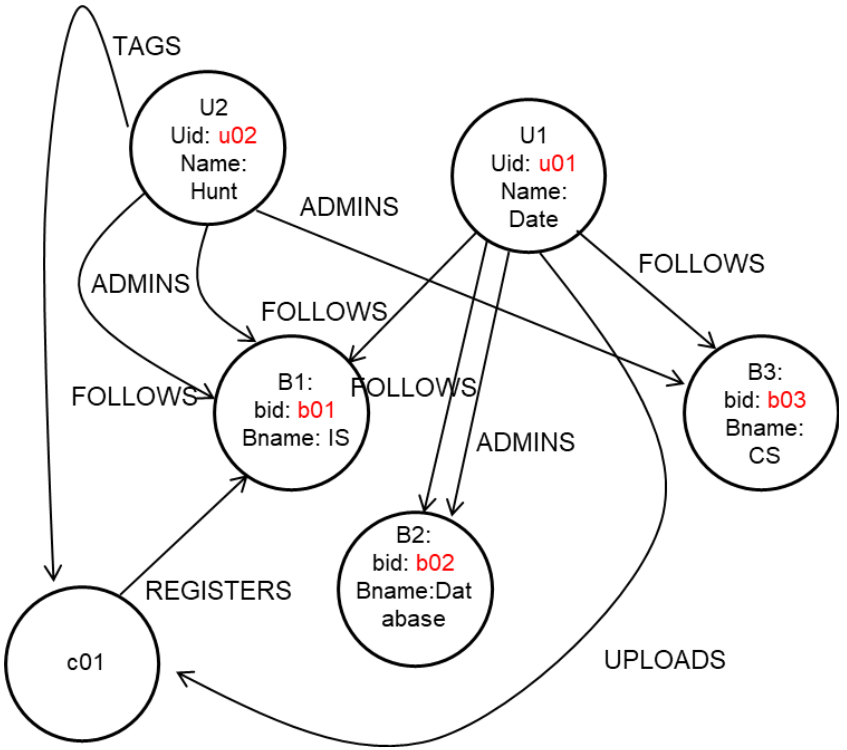
	tuser	tcomment
t_7	u02	c01

Blog (BG)

	bid	bname	admin
t_8	b01	Information Systems	u02
t_9	b02	Database	u01
t_{10}	b03	Computer Science	u02

Comment (CT)

	cid	cblog	cuser	msg	date
t_{11}	c01	b01	u01	Exactly what I was looking for!	25/02/2013



Queries (cont.)

- ✓ *Find the users who post comments on blog 'b01';*
- ✓ *Find the users who tag comments posted on blog 'b01';*
- ✓ *Find users who either post comments on blog 'b01' or tag comments posted on 'b01';*
- Find how 'u01' and 'u02' are related;
  - *U1 FOLLOWS blog administered by U2*

User (US)

	uid	uname
t_1	u01	Date
t_2	u02	Hunt

Follower (FR)

	fuser	fblog
t_3	u01	b01
t_4	u01	b02
t_5	u01	b03
t_6	u02	b01

Tag (TG)

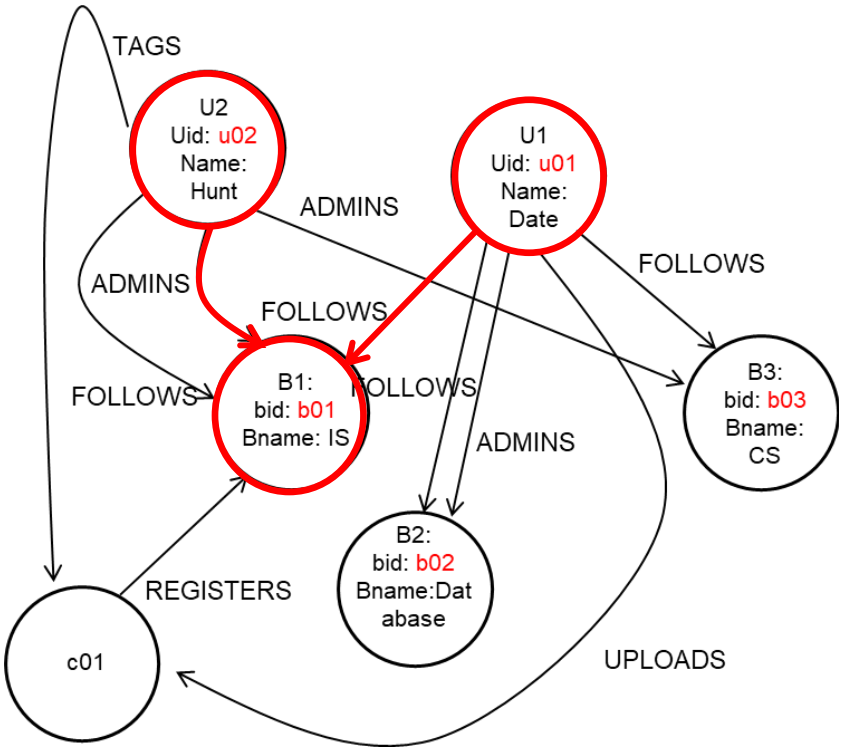
	tuser	tcomment
t_7	u02	c01

Blog (BG)

	bid	bname	admin
t_8	b01	Information Systems	u02
t_9	b02	Database	u01
t_{10}	b03	Computer Science	u02

Comment (CT)

	cid	cblog	cuser	msg	date
t_{11}	c01	b01	u01	Exactly what I was looking for!	25/02/2013



Queries (cont.)

- ✓ *Find the users who post comments on blog 'b01';*
- ✓ *Find the users who tag comments posted on blog 'b01';*
- ✓ *Find users who either post comments on blog 'b01' or tag comments posted on 'b01';*
- Find how 'u01' and 'u02' are related;
  - U1 FOLLOWS blog administered by U2
  - *U1 UPLOADS comments in blog administered by U2*

User (US)

	uid	uname
t_1	u01	Date
t_2	u02	Hunt

Follower (FR)

	fuser	fblog
t_3	u01	b01
t_4	u01	b02
t_5	u01	b03
t_6	u02	b01

Tag (TG)

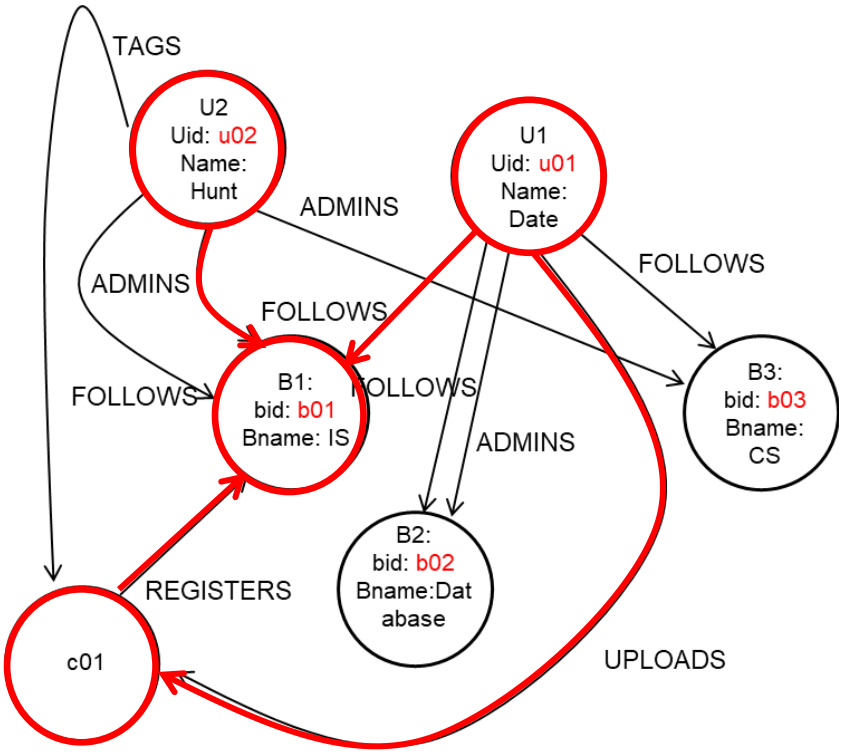
	tuser	tcomment
t_7	u02	c01

Blog (BG)

	bid	bname	admin
t_8	b01	Information Systems	u02
t_9	b02	Database	u01
t_{10}	b03	Computer Science	u02

Comment (CT)

	cid	cblog	cuser	msg	date
t_{11}	c01	b01	u01	Exactly what I was looking for!	25/02/2013



Queries (cont.)

- ✓ *Find the users who post comments on blog 'b01';*
- ✓ *Find the users who tag comments posted on blog 'b01';*
- ✓ *Find users who either post comments on blog 'b01' or tag comments posted on 'b01';*
- Find how 'u01' and 'u02' are related;
  - U1 FOLLOWS blog administered by U2
  - U1 UPLOADS comments in blog administered by U2
  - *U2 TAGS comment which is UPLOADED by U1 in blog administered by U2*

User (US)

	uid	uname
t_1	u01	Date
t_2	u02	Hunt

Follower (FR)

	fuser	fblog
t_3	u01	b01
t_4	u01	b02
t_5	u01	b03
t_6	u02	b01

Tag (TG)

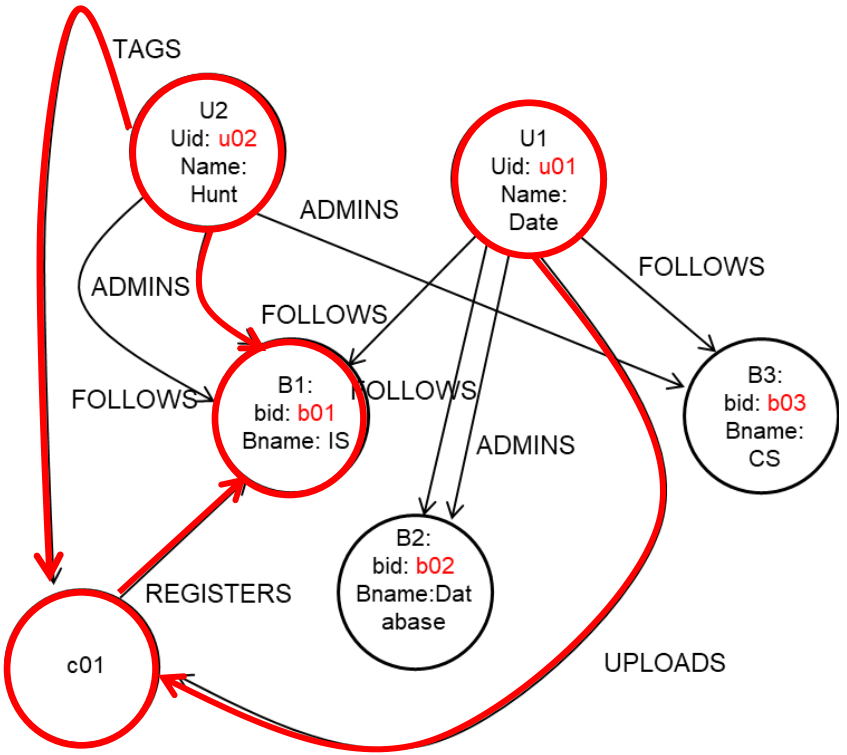
	tuser	tcomment
t_7	u02	c01

Blog (BG)

	bid	bname	admin
t_8	b01	Information Systems	u02
t_9	b02	Database	u01
t_{10}	b03	Computer Science	u02

Comment (CT)

	cid	cblog	cuser	msg	date
t_{11}	c01	b01	u01	Exactly what I was looking for!	25/02/2013



Ερωτήματα

- ✓ *Find the users who post comments on blog 'b01';*
- ✓ *Find the users who tag comments posted on blog 'b01';*
- ✓ *Find users who either post comments on blog 'b01' or tag comments posted on 'b01';*
- ✓ *Find how 'u01' and 'u02' are related;*
- How far are nodes 'u01' and 'b01' in all paths;
  - One step apart in path 'FOLLOWS'
  - Two steps apart in path UPLOADS - REGISTERS

User (US)

	uid	uname
t_1	u01	Date
t_2	u02	Hunt

Follower (FR)

	fuser	fblog
t_3	u01	b01
t_4	u01	b02
t_5	u01	b03
t_6	u02	b01

Tag (TG)

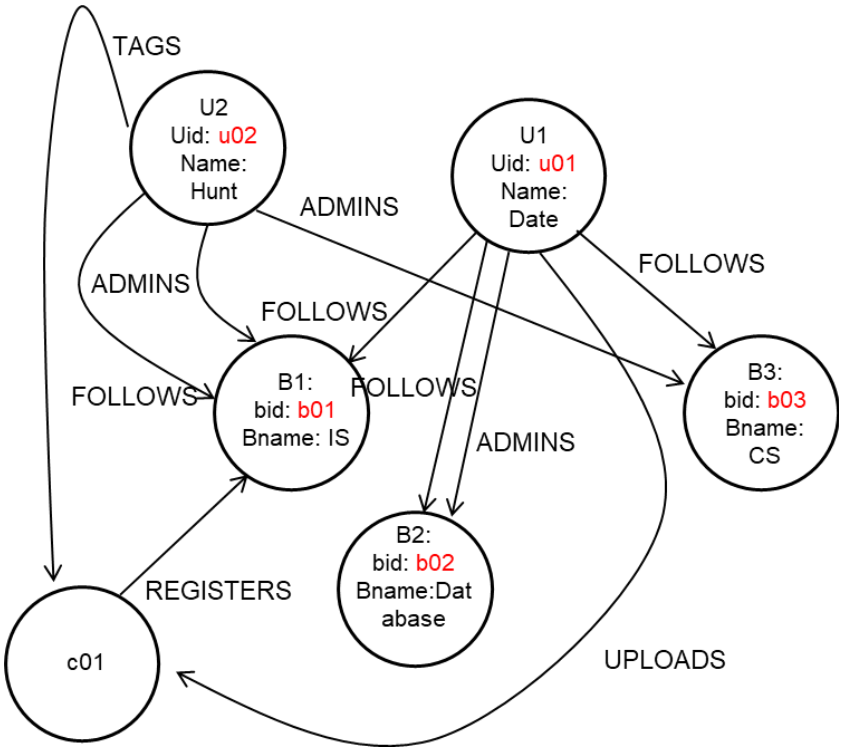
	tuser	tcomment
t_7	u02	c01

Blog (BG)

	bid	bname	admin
t_8	b01	Information Systems	u02
t_9	b02	Database	u01
t_{10}	b03	Computer Science	u02

Comment (CT)

	cid	cblog	cuser	msg	date
t_{11}	c01	b01	u01	Exactly what I was looking for!	25/02/2013



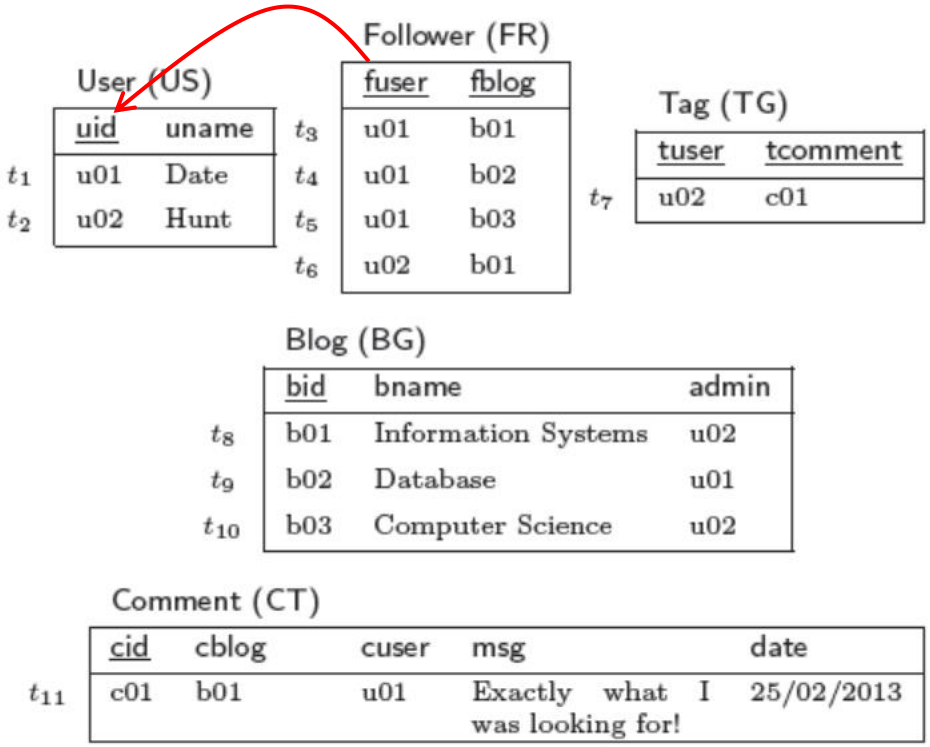
Ανάπτυξη schema graph

Approach

- Focus is on the relational schema (not the data)
- It does not result in a canonical property graph
- Issues about the type and range of queries

Constraints

- Focusing on foreign key constraints
  - FR.fuser REFERENCES US.uid



User (US)

	uid	uname
t_1	u01	Date
t_2	u02	Hunt

Follower (FR)

	fuser	fblog
t_3	u01	b01
t_4	u01	b02
t_5	u01	b03
t_6	u02	b01

Tag (TG)

	tuser	tcomment
t_7	u02	c01

Blog (BG)

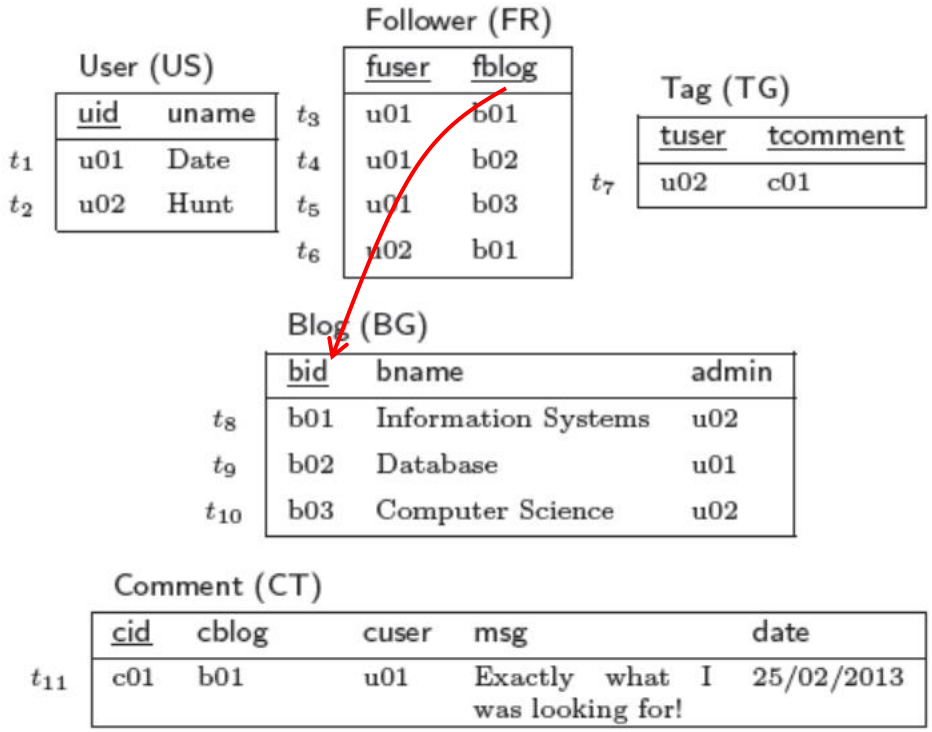
	bid	bname	admin
t_8	b01	Information Systems	u02
t_9	b02	Database	u01
t_{10}	b03	Computer Science	u02

Comment (CT)

	cid	cblog	cuser	msg	date
t_{11}	c01	b01	u01	Exactly what I was looking for!	25/02/2013

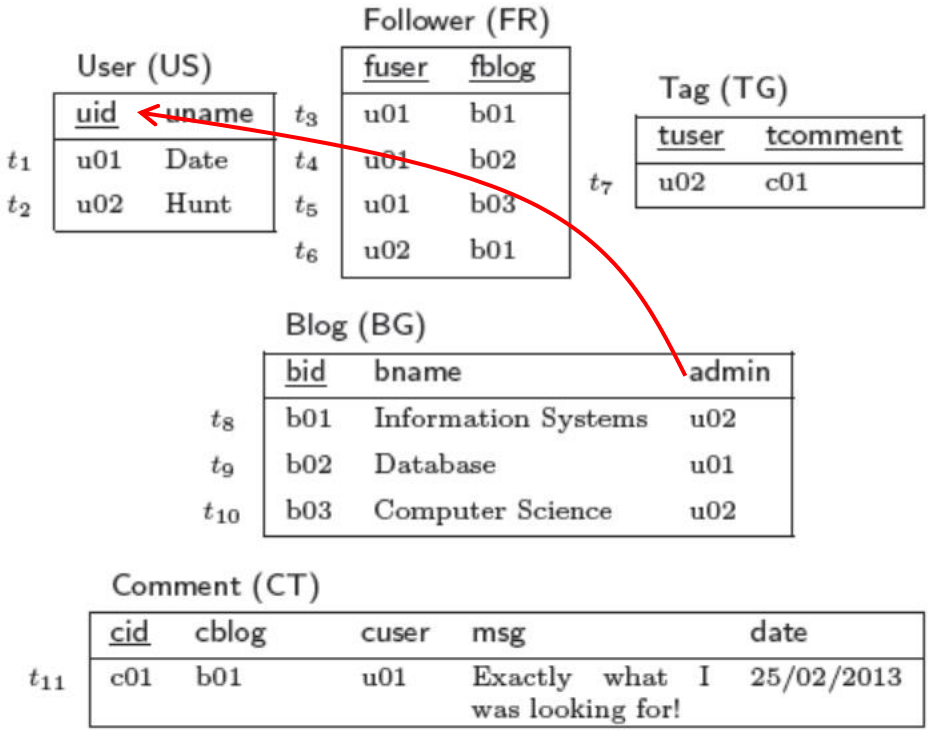
Constraints (cont.)

- Focusing on foreign key constraints
  - FR.fuser REFERENCES US.uid
  - FR.fblog REFERENES BG.bid



Constraints (cont.)

- Focusing on foreign key constraints
  - FR.fuser REFERENCES US.uid
  - FR.fblog REFERENES BG.bid
  - BG.admin REFERENCES US.uid



User (US)

	uid	uname
t_1	u01	Date
t_2	u02	Hunt

Follower (FR)

	fuser	fblog
t_3	u01	b01
t_4	u01	b02
t_5	u01	b03
t_6	u02	b01

Tag (TG)

	tuser	tcomment
t_7	u02	c01

Blog (BG)

	bid	bname	admin
t_8	b01	Information Systems	u02
t_9	b02	Database	u01
t_{10}	b03	Computer Science	u02

Comment (CT)

	cid	cblog	cuser	msg	date
t_{11}	c01	b01	u01	Exactly what I was looking for!	25/02/2013

Constraints (cont.)

- Focusing on foreign key constraints
  - FR.fuser REFERENCES US.uid
  - FR.fblog REFERENES BG.bid
  - BG.admin REFERENCES US.uid
  - CT.cblog REFERENCES BG.bid

User (US)

	uid	uname
t_1	u01	Date
t_2	u02	Hunt

Follower (FR)

	fuser	fblog
t_3	u01	b01
t_4	u01	b02
t_5	u01	b03
t_6	u02	b01

Tag (TG)

	tuser	tcomment
t_7	u02	c01

Blog (BG)

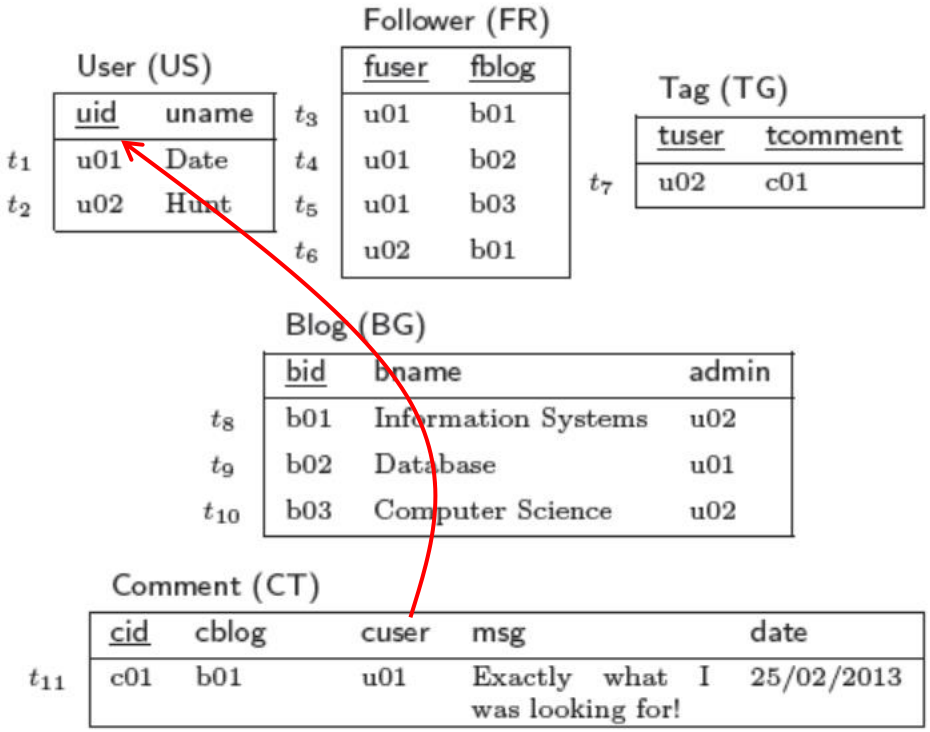
	bid	bname	admin
t_8	b01	Information Systems	u02
t_9	b02	Database	u01
t_{10}	b03	Computer Science	u02

Comment (CT)

	cid	cblog	cuser	msg	date
t_{11}	c01	b01	u01	Exactly what I was looking for!	25/02/2013

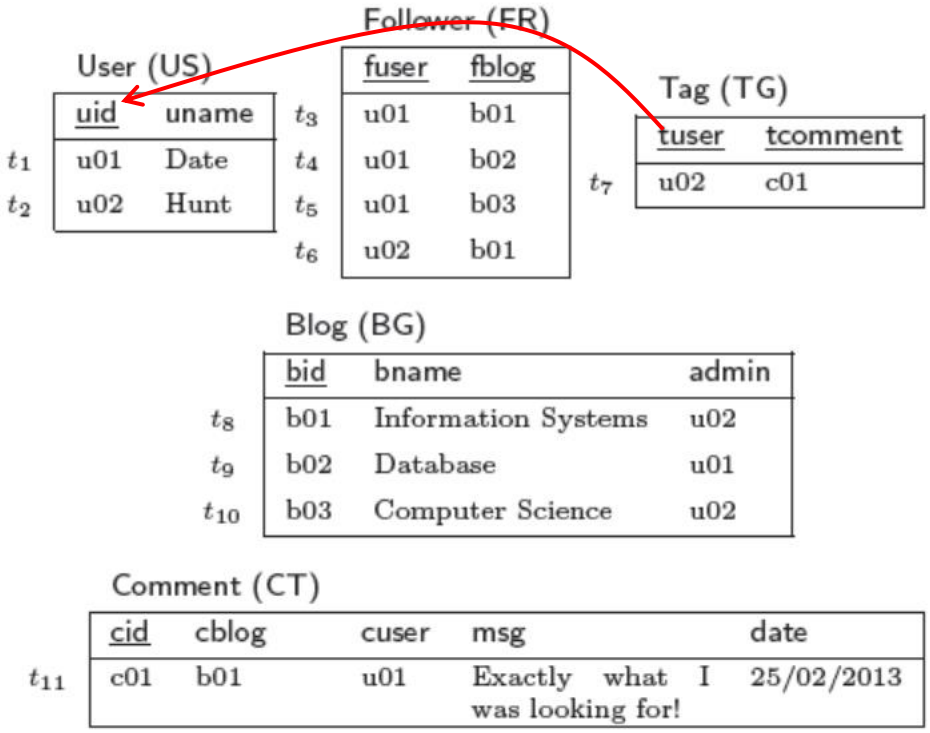
Constraints (cont.)

- Focusing on foreign key constraints
  - FR.fuser REFERENCES US.uid
  - FR.fblog REFERENES BG.bid
  - BG.admin REFERENCES US.uid
  - CT.cblog REFERENCES BG.bid
  - CT.cuser REFERENCES US.uid



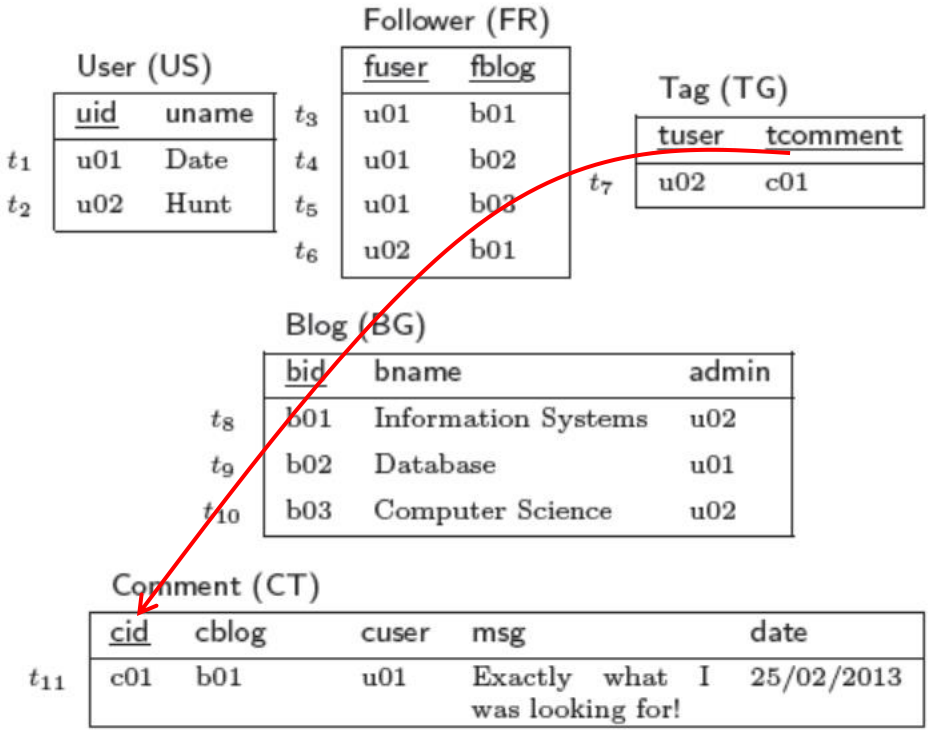
Constraints (cont.)

- Focusing on foreign key constraints
  - FR.fuser REFERENCES US.uid
  - FR.fblog REFERENES BG.bid
  - BG.admin REFERENCES US.uid
  - CT.cblog REFERENCES BG.bid
  - CT.cuser REFERENCES US.uid
  - TG.tuser REFERENCES US.uid



Constraints (cont.)

- Focusing on foreign key constraints
  - FR.fuser REFERENCES US.uid
  - FR.fblog REFERENES BG.bid
  - BG.admin REFERENCES US.uid
  - CT.cblog REFERENCES BG.bid
  - CT.cuser REFERENCES US.uid
  - TG.tuser REFERENCES US.uid
  - TG.tcomment REFERENCES CT.cid



Total set of constraints

- In total seven constraints
  - FR.fuser REFERENCES US.uid
  - FR.fblog REFERENES BG.bid
  - BG.admin REFERENCES US.uid
  - CT.cblog REFERENCES BG.bid
  - CT.cuser REFERENCES US.uid
  - TG.tuser REFERENCES US.uid
  - TG.tcomment REFERENCES CT.cid

User (US)

	uid	uname
t_1	u01	Date
t_2	u02	Hunt

Follower (FR)

	fuser	fblog
t_3	u01	b01
t_4	u01	b02
t_5	u01	b03
t_6	u02	b01

Tag (TG)

	tuser	tcomment
t_7	u02	c01

Blog (BG)

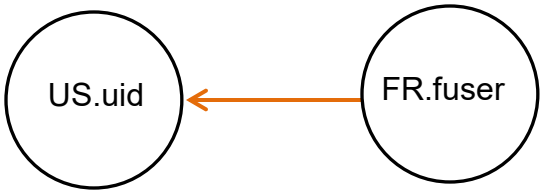
	bid	bname	admin
t_8	b01	Information Systems	u02
t_9	b02	Database	u01
t_{10}	b03	Computer Science	u02

Comment (CT)

	cid	cblog	cuser	msg	date
t_{11}	c01	b01	u01	Exactly what I was looking for!	25/02/2013

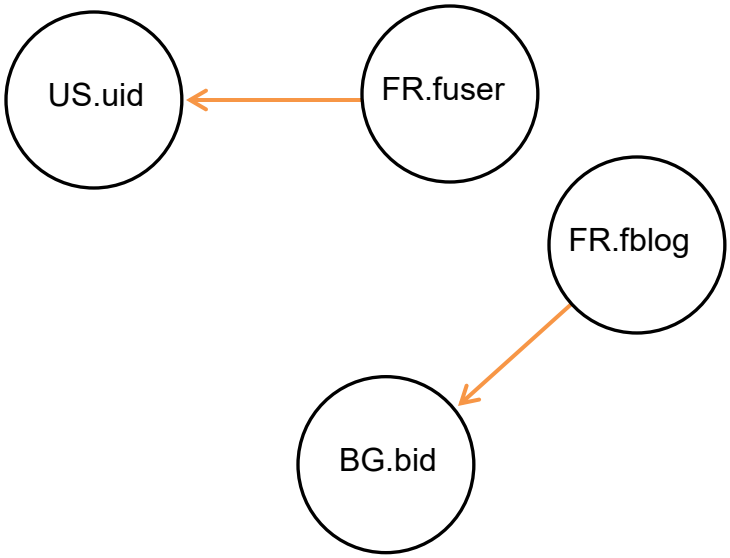
From constraints to graph schema

- From foreign key constraints to graph
  - FR.fuser REFERENCES US.uid



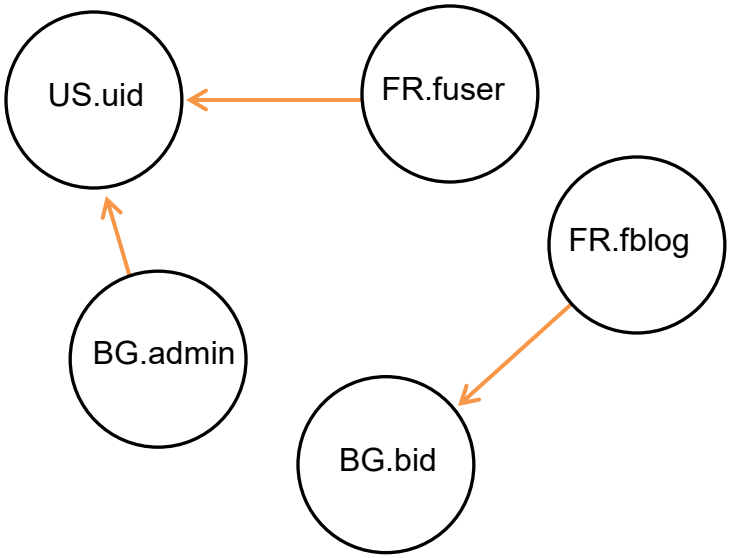
From constraints to graph schema (cont.)

- From foreign key constraints to graph
  - *FR.fuser REFERENCES US.uid*
  - FR.fblog REFERENES BG.bid



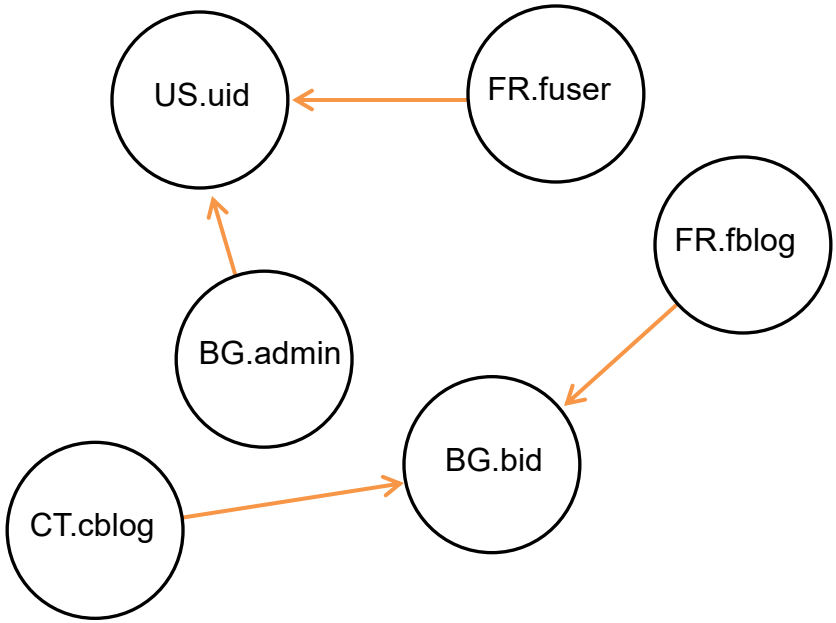
From constraints to graph schema (cont.)

- From foreign key constraints to graph
  - *FR.fuser REFERENCES US.uid*
  - *FR.fblog REFERENES BG.bid*
  - BG.admin REFERENCES US.uid



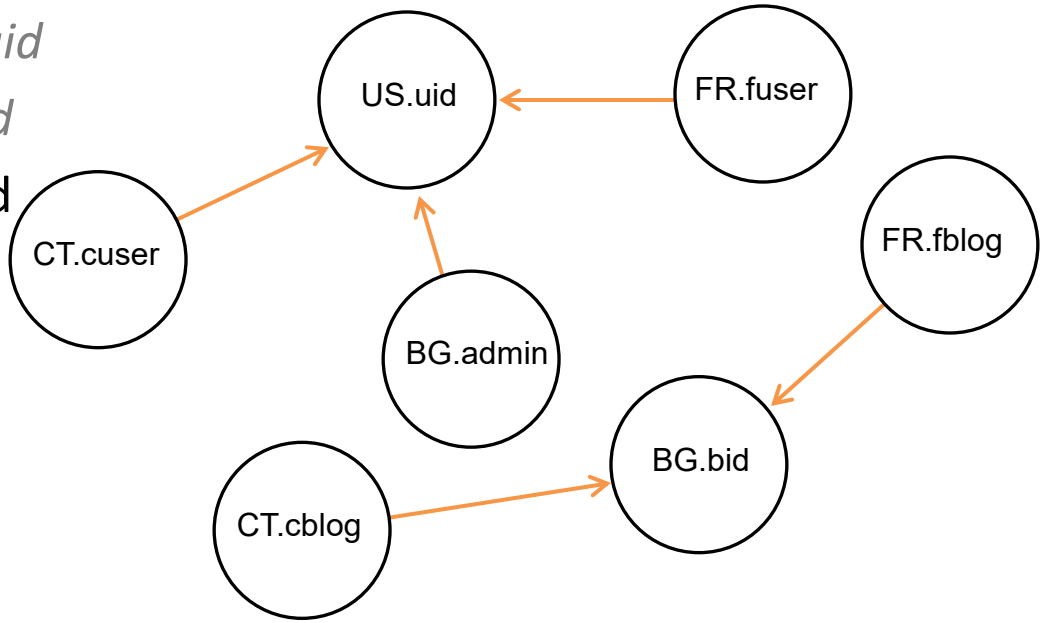
From constraints to graph schema (cont.)

- From foreign key constraints to graph
  - *FR.fuser REFERENCES US.uid*
  - *FR.fblog REFERENES BG.bid*
  - *BG.admin REFERENCES US.uid*
  - CT.cblog REFERENCES BG.bid



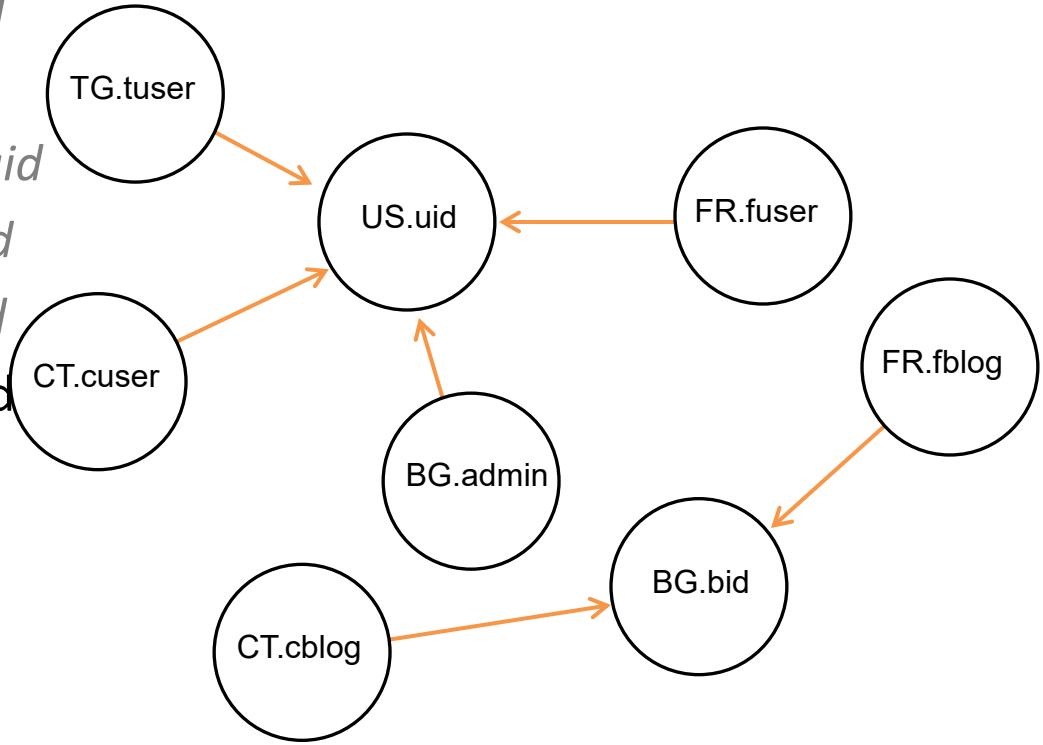
From constraints to graph schema (cont.)

- From foreign key constraints to graph
  - *FR.fuser REFERENCES US.uid*
  - *FR.fblog REFERENES BG.bid*
  - *BG.admin REFERENCES US.uid*
  - *CT.cblog REFERENCES BG.bid*
  - CT.cuser REFERENCES US.uid



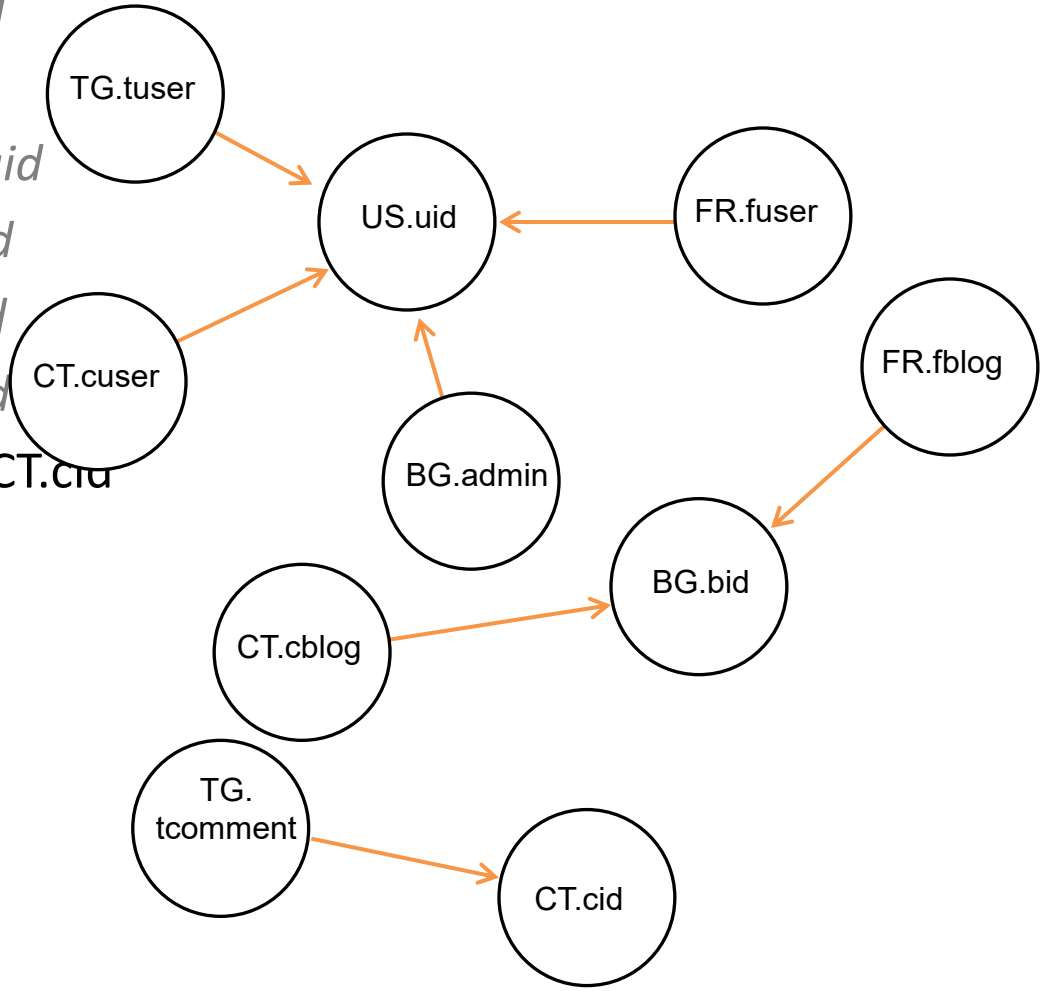
From constraints to graph schema (cont.)

- From foreign key constraints to graph
  - *FR.fuser REFERENCES US.uid*
  - *FR.fblog REFERENES BG.bid*
  - *BG.admin REFERENCES US.uid*
  - *CT.cblog REFERENCES BG.bid*
  - *CT.cuser REFERENCES US.uid*
  - TG.tuser REFERENCES US.uid



From constraints to graph schema (cont.)

- From foreign key constraints to graph
  - *FR.fuser REFERENCES US.uid*
  - *FR.fblog REFERENES BG.bid*
  - *BG.admin REFERENCES US.uid*
  - *CT.cblog REFERENCES BG.bid*
  - *CT.cuser REFERENCES US.uid*
  - *TG.tuser REFERENCES US.uid*
  - TG.tcomment REFERENCES CT.cid



From constraints to graph schema (cont.)

- Adding the remaining relationships (Tag(TG))

User (US)

	uid	uname
t_1	u01	Date
t_2	u02	Hunt

Follower (FR)

	fuser	fblog
t_3	u01	b01
t_4	u01	b02
t_5	u01	b03
t_6	u02	b01

Tag (TG)

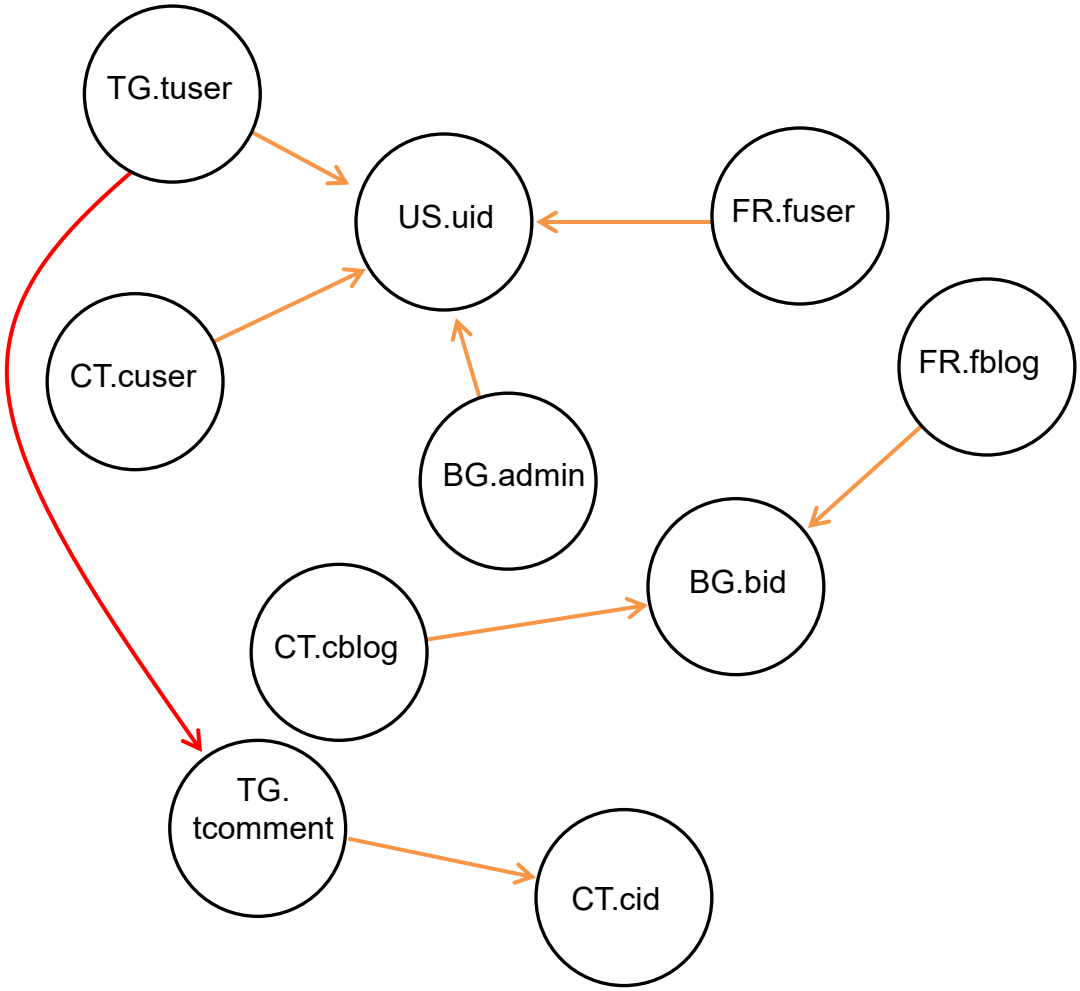
	tuser	tcomment
t_7	u02	c01

Blog (BG)

	bid	bname	admin
t_8	b01	Information Systems	u02
t_9	b02	Database	u01
t_{10}	b03	Computer Science	u02

Comment (CT)

	cid	cblog	cuser	msg	date
t_{11}	c01	b01	u01	Exactly what I was looking for!	25/02/2013



From constraints to graph schema (cont.)

- Adding the remaining relationships (Comment)

User (US)

	uid	uname
t_1	u01	Date
t_2	u02	Hunt

Follower (FR)

	fuser	fblog
t_3	u01	b01
t_4	u01	b02
t_5	u01	b03
t_6	u02	b01

Tag (TG)

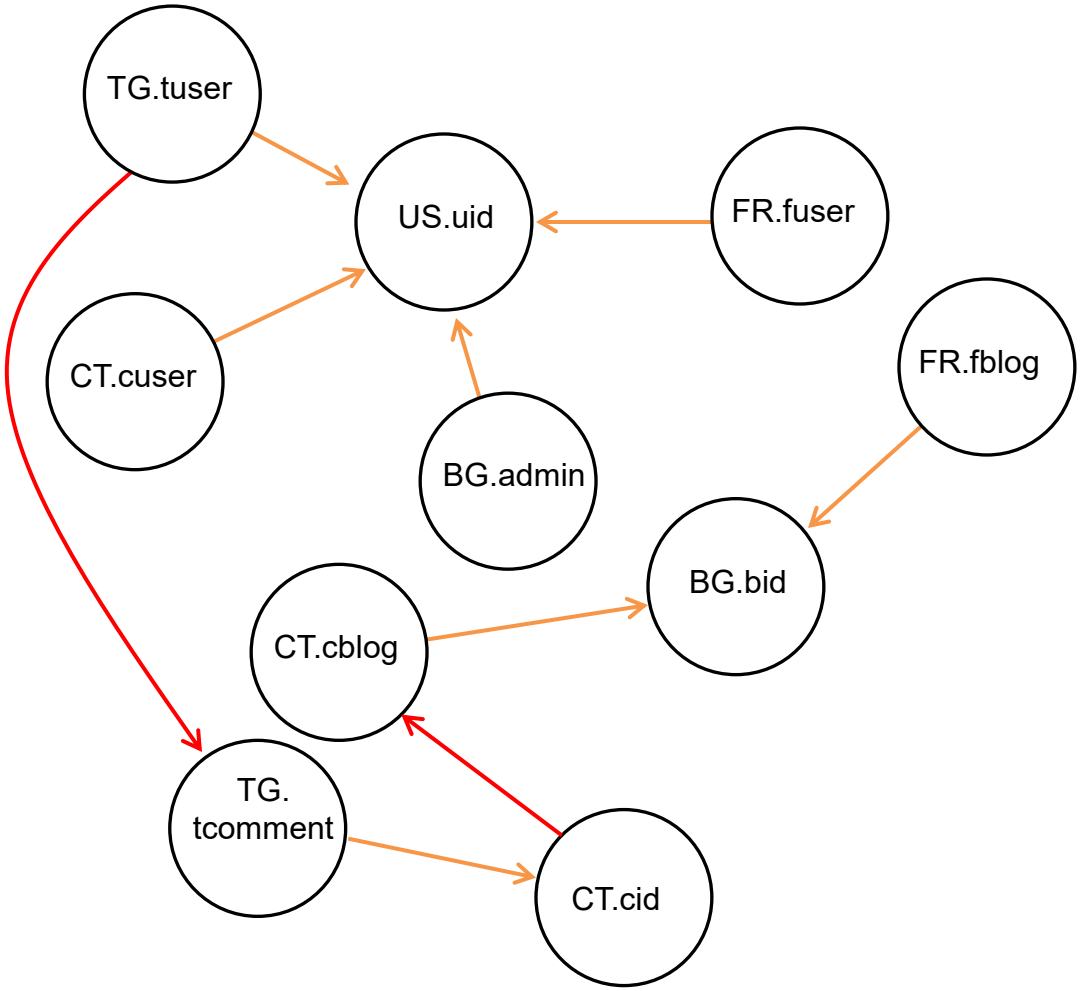
	tuser	tcomment
t_7	u02	c01

Blog (BG)

	bid	bname	admin
t_8	b01	Information Systems	u02
t_9	b02	Database	u01
t_{10}	b03	Computer Science	u02

Comment (CT)

	cid	cblog	cuser	msg	date
t_{11}	c01	b01	u01	Exactly what I was looking for!	25/02/2013



From constraints to graph schema (cont.)

- Adding the remaining relationships (Comment)

User (US)

	uid	uname
t_1	u01	Date
t_2	u02	Hunt

Follower (FR)

	fuser	fblog
t_3	u01	b01
t_4	u01	b02
t_5	u01	b03
t_6	u02	b01

Tag (TG)

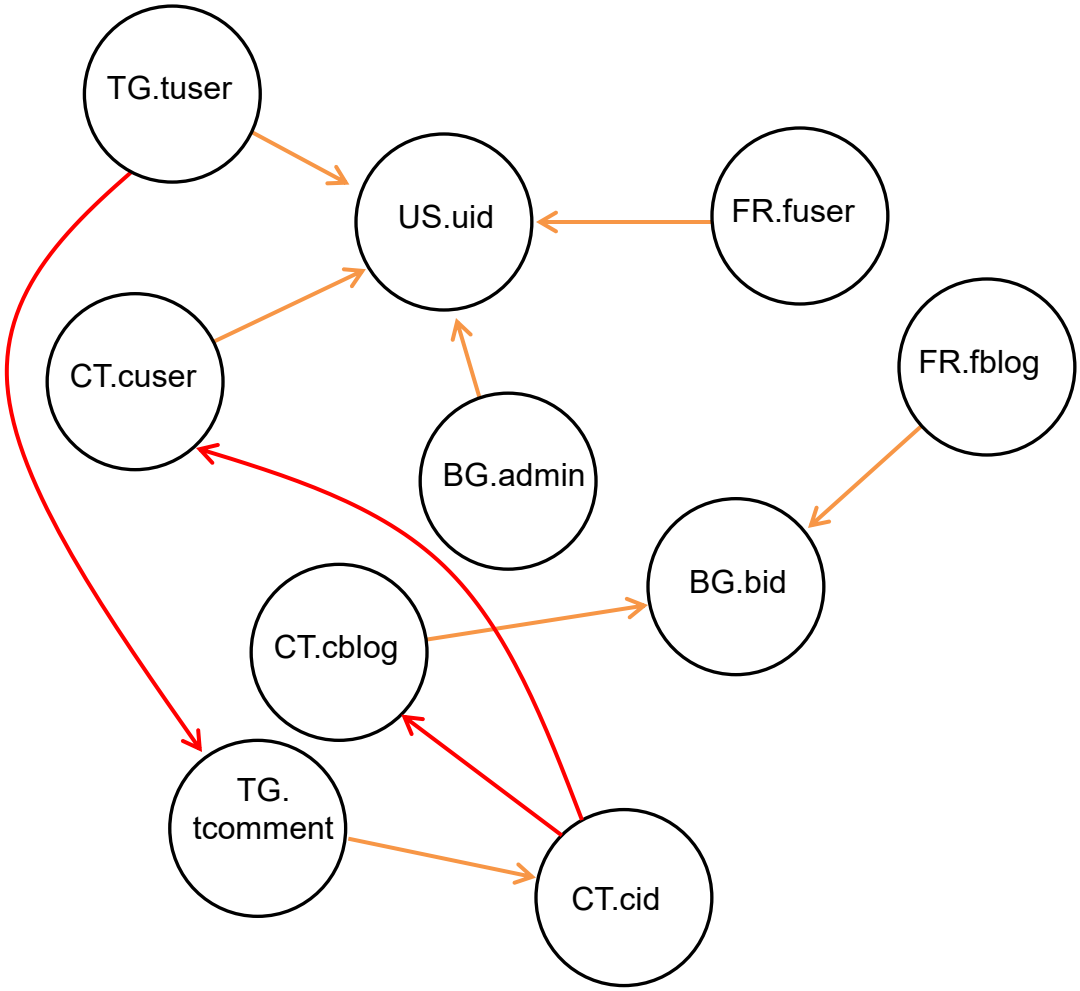
	tuser	tcomment
t_7	u02	c01

Blog (BG)

	bid	bname	admin
t_8	b01	Information Systems	u02
t_9	b02	Database	u01
t_{10}	b03	Computer Science	u02

Comment (CT)

	cid	cblog	cuser	msg	date
t_{11}	c01	b01	u01	Exactly what I was looking for!	25/02/2013



From constraints to graph schema (cont.)

- Adding the remaining relationships (Comment)

User (US)

	uid	uname
t_1	u01	Date
t_2	u02	Hunt

Follower (FR)

	fuser	fblog
t_3	u01	b01
t_4	u01	b02
t_5	u01	b03
t_6	u02	b01

Tag (TG)

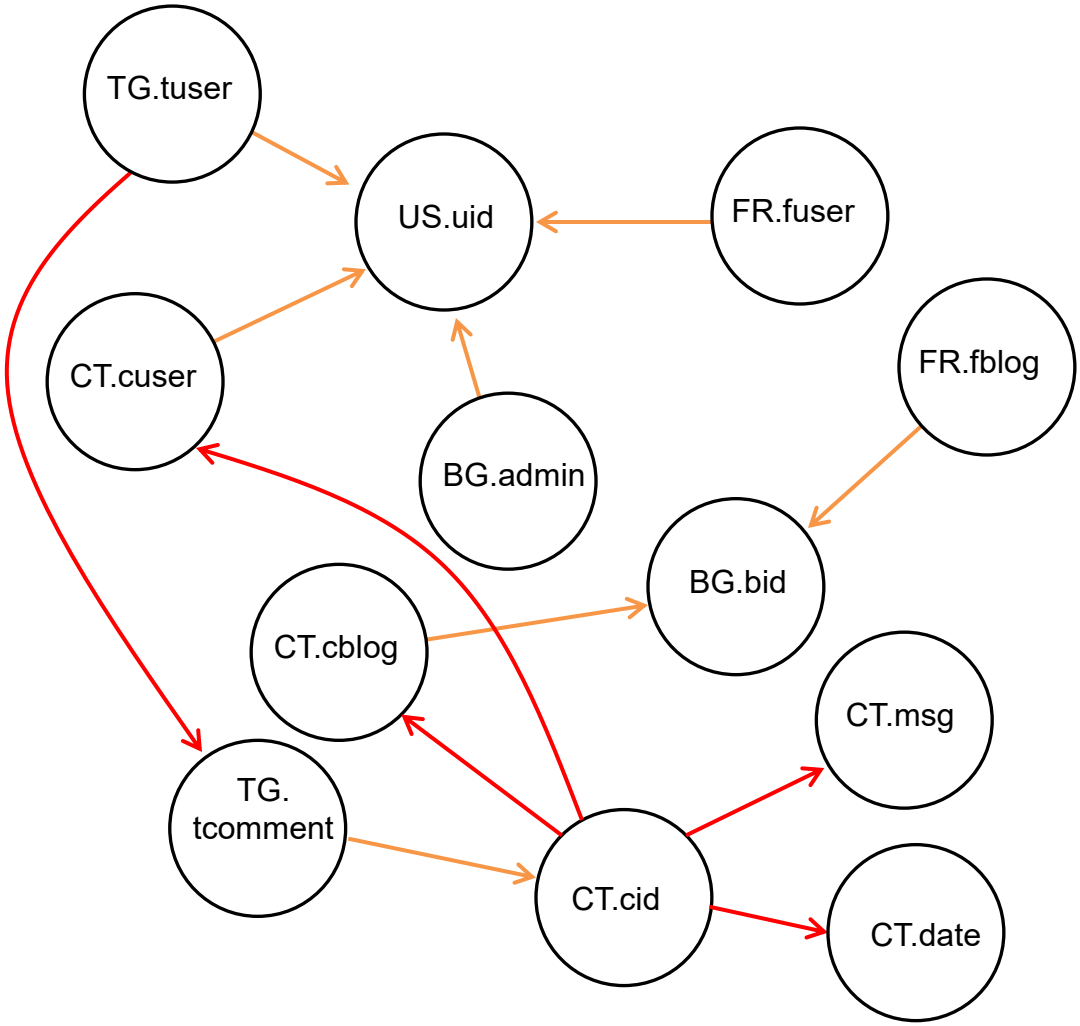
	tuser	tcomment
t_7	u02	c01

Blog (BG)

	bid	bname	admin
t_8	b01	Information Systems	u02
t_9	b02	Database	u01
t_{10}	b03	Computer Science	u02

Comment (CT)

	cid	cblog	cuser	msg	date
t_{11}	c01	b01	u01	Exactly what I was looking for!	25/02/2013



From constraints to graph schema (cont.)

- Adding the remaining relationships (Blog)

User (US)

	uid	uname
t_1	u01	Date
t_2	u02	Hunt

Follower (FR)

	fuser	fblog
t_3	u01	b01
t_4	u01	b02
t_5	u01	b03
t_6	u02	b01

Tag (TG)

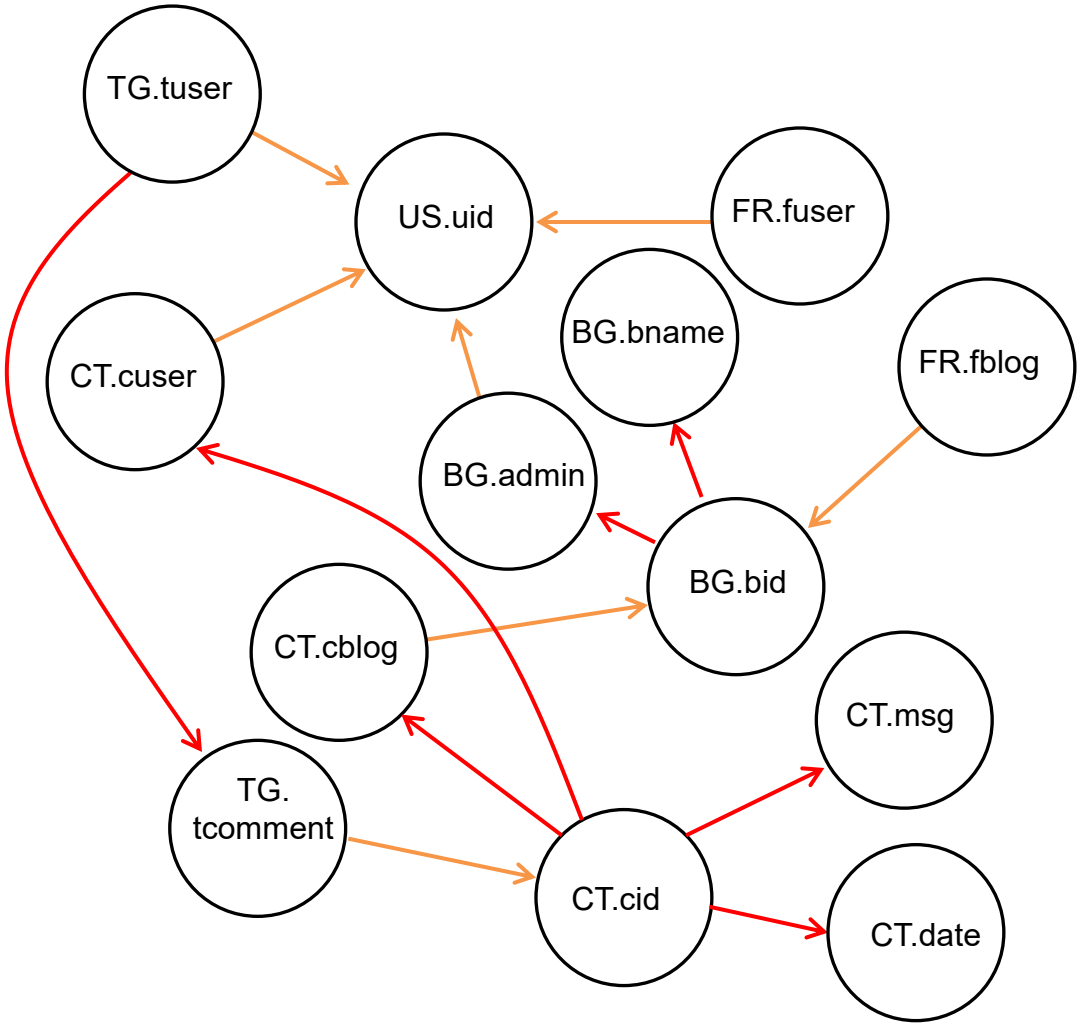
	tuser	tcomment
t_7	u02	c01

Blog (BG)

	bid	bname	admin
t_8	b01	Information Systems	u02
t_9	b02	Database	u01
t_{10}	b03	Computer Science	u02

Comment (CT)

	cid	cblog	cuser	msg	date
t_{11}	c01	b01	u01	Exactly what I was looking for!	25/02/2013



From constraints to graph schema (cont.)

- Adding the remaining relationships (Follower (FR))

User (US)

	uid	uname
t_1	u01	Date
t_2	u02	Hunt

Follower (FR)

	fuser	fblog
t_3	u01	b01
t_4	u01	b02
t_5	u01	b03
t_6	u02	b01

Tag (TG)

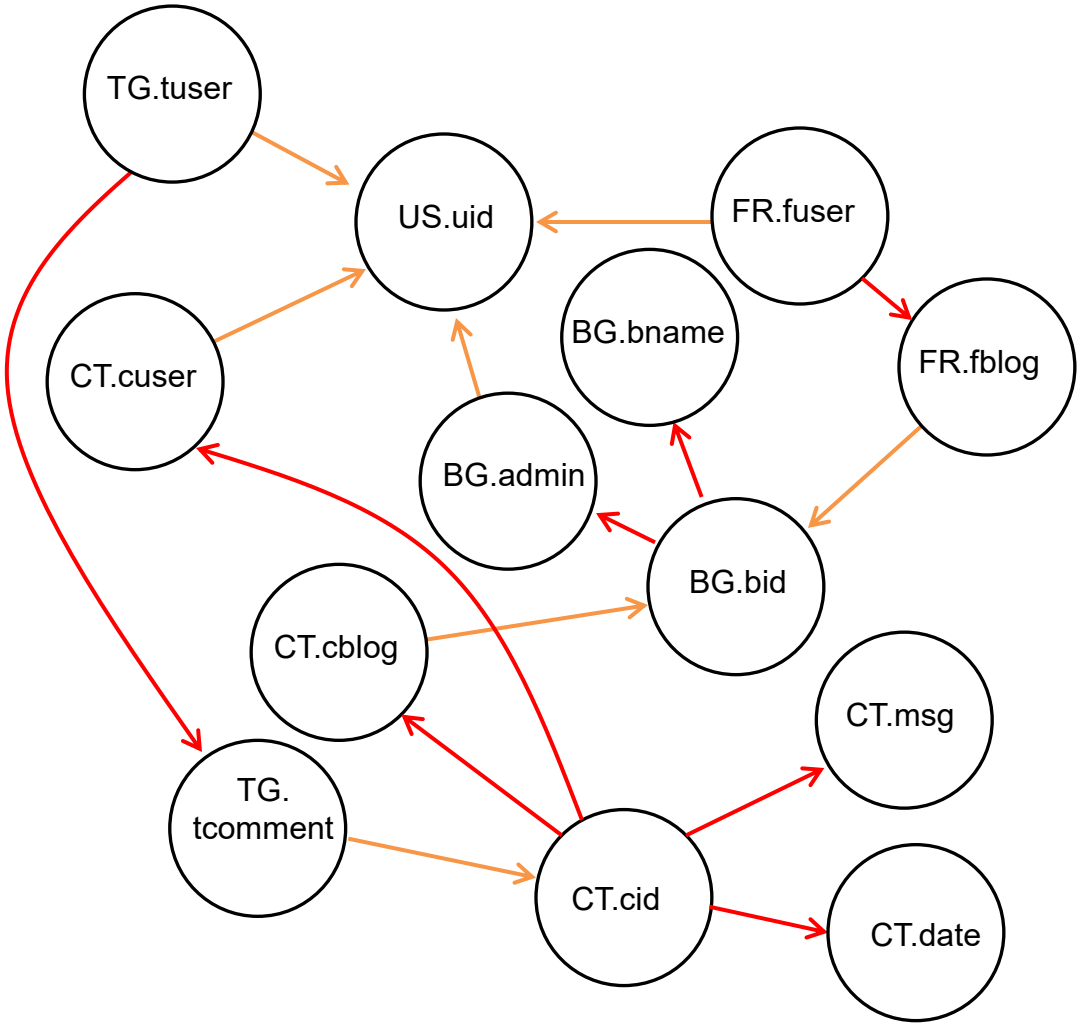
	tuser	tcomment
t_7	u02	c01

Blog (BG)

	bid	bname	admin
t_8	b01	Information Systems	u02
t_9	b02	Database	u01
t_{10}	b03	Computer Science	u02

Comment (CT)

	cid	cblog	cuser	msg	date
t_{11}	c01	b01	u01	Exactly what I was looking for!	25/02/2013



From constraints to graph schema (cont.)

- Adding the remaining relationships (User (US))

User (US)

	uid	uname
t_1	u01	Date
t_2	u02	Hunt

Follower (FR)

	fuser	fblog
t_3	u01	b01
t_4	u01	b02
t_5	u01	b03
t_6	u02	b01

Tag (TG)

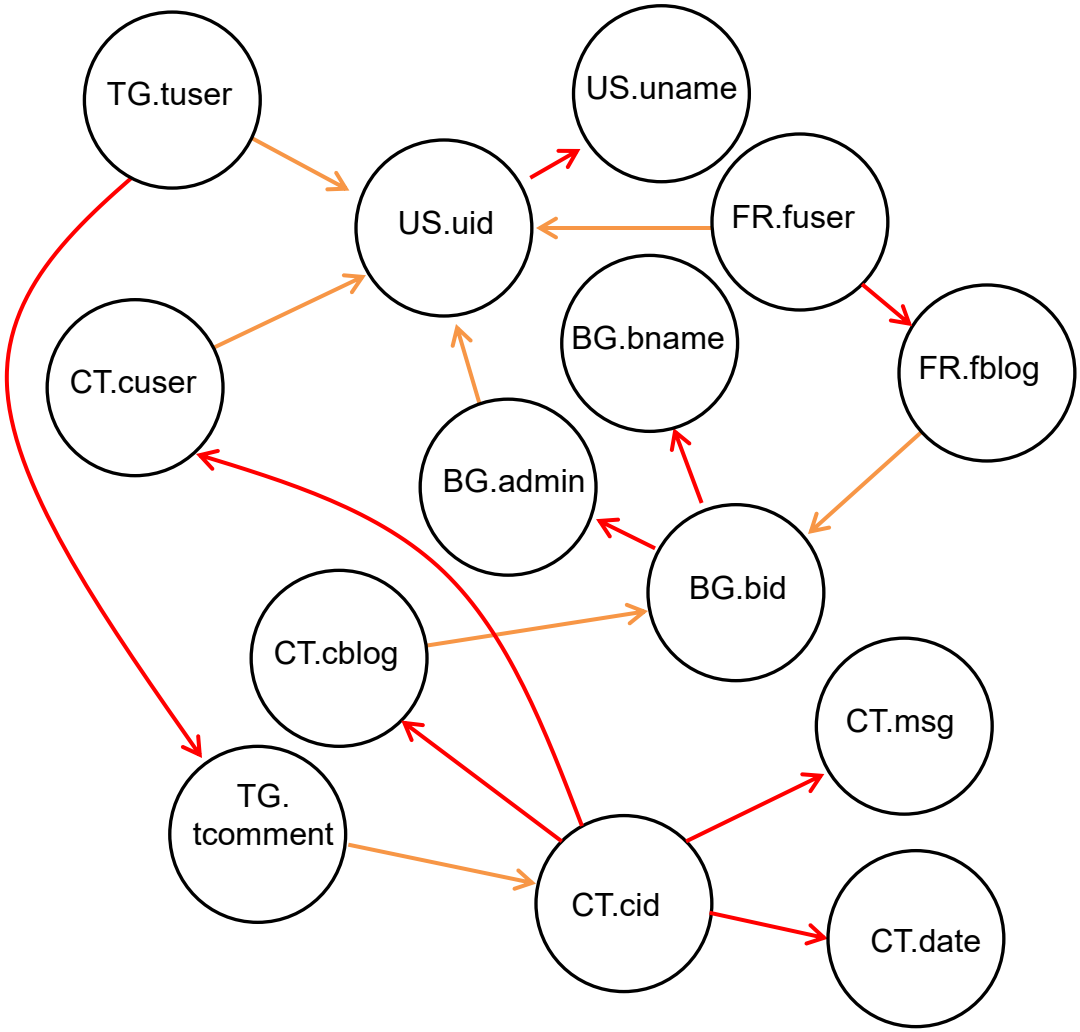
	tuser	tcomment
t_7	u02	c01

Blog (BG)

	bid	bname	admin
t_8	b01	Information Systems	u02
t_9	b02	Database	u01
t_{10}	b03	Computer Science	u02

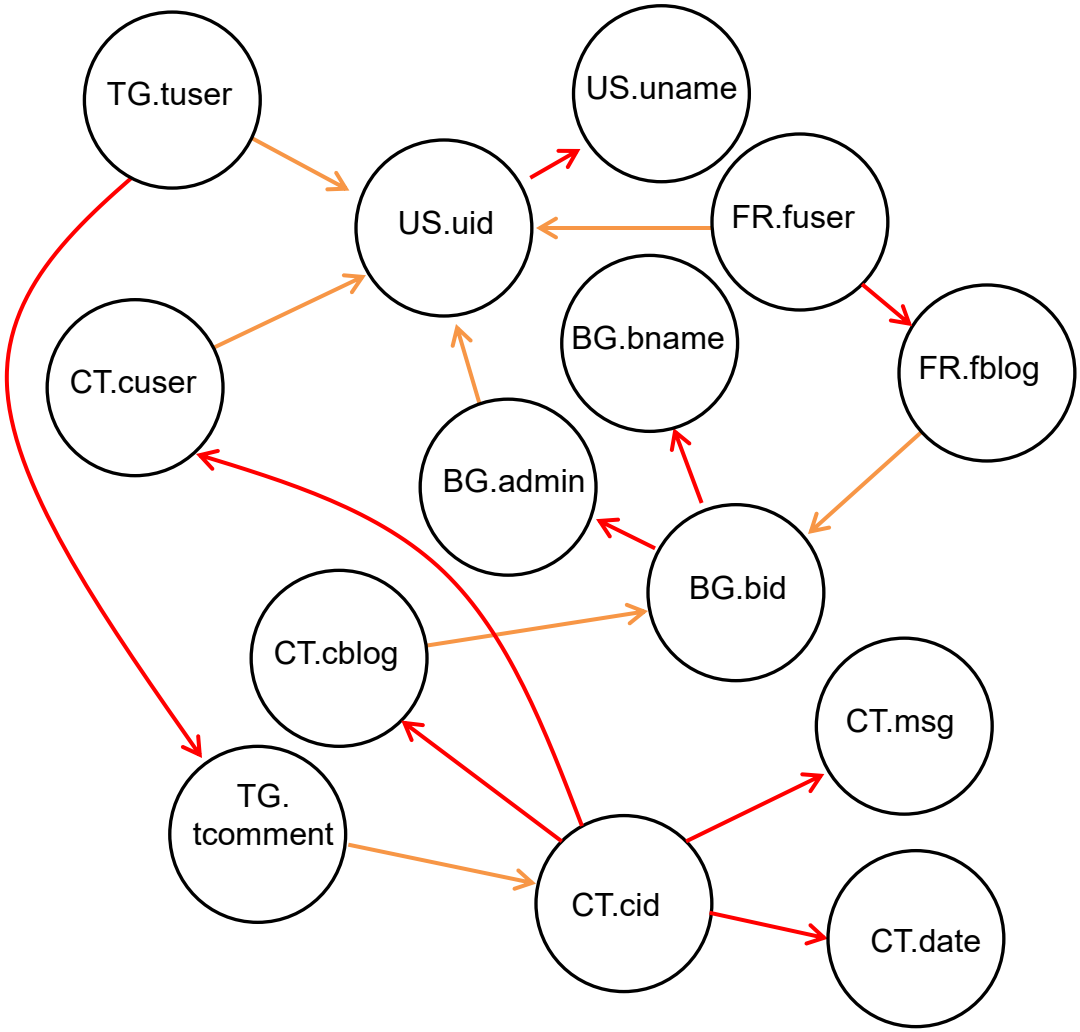
Comment (CT)

	cid	cblog	cuser	msg	date
t_{11}	c01	b01	u01	Exactly what I was looking for!	25/02/2013



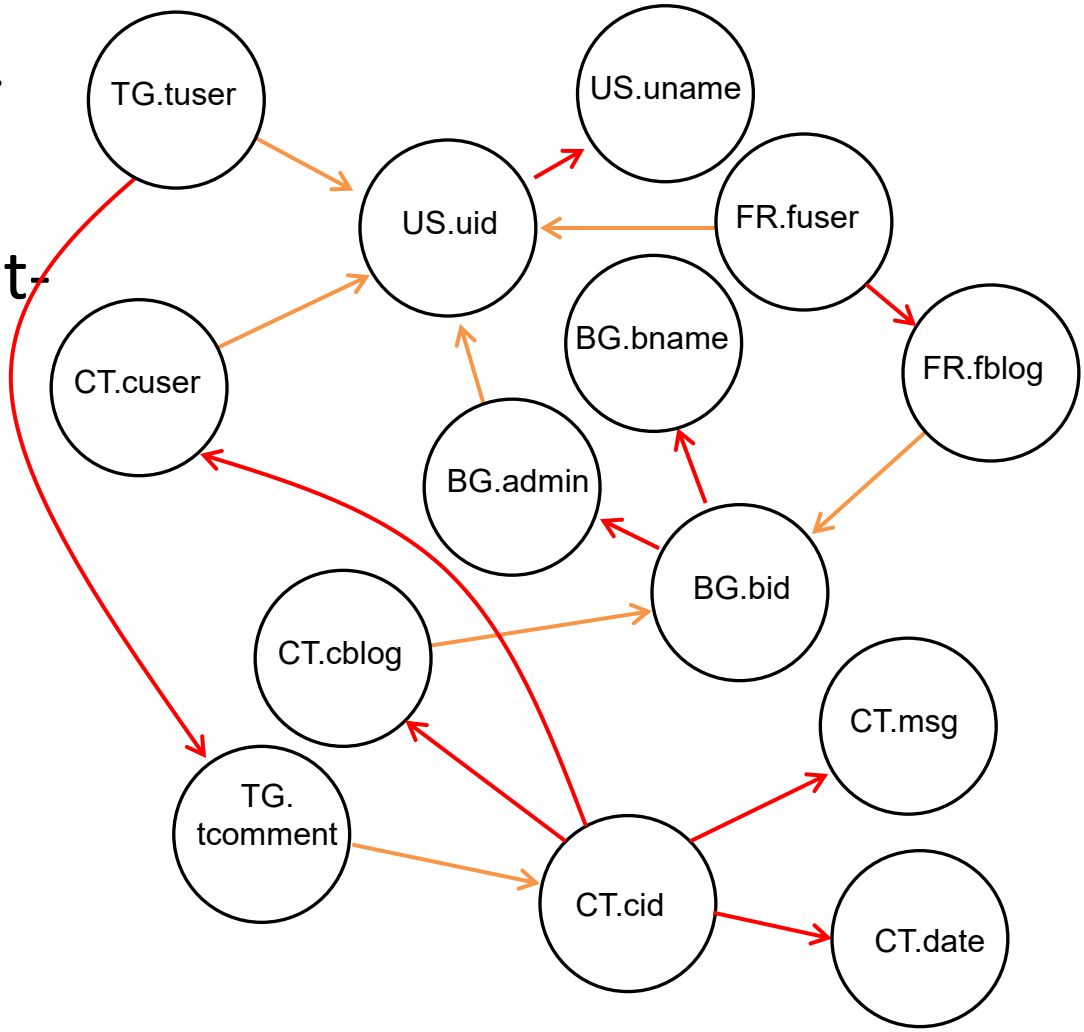
Notice

- This is not a property graph !



Notice (cont.)

- Differences across the nodes?
  - Nodes with no in-coming edges
  - Nodes with no out-goings edges



Ανάλυση κόμβων (συν.)

- Nodes with no in-coming edges
  - No destination in any path

User (US)

	uid	uname
t_1	u01	Date
t_2	u02	Hunt

Follower (FR)

	fuser	fblog
t_3	u01	b01
t_4	u01	b02
t_5	u01	b03
t_6	u02	b01

Tag (TG)

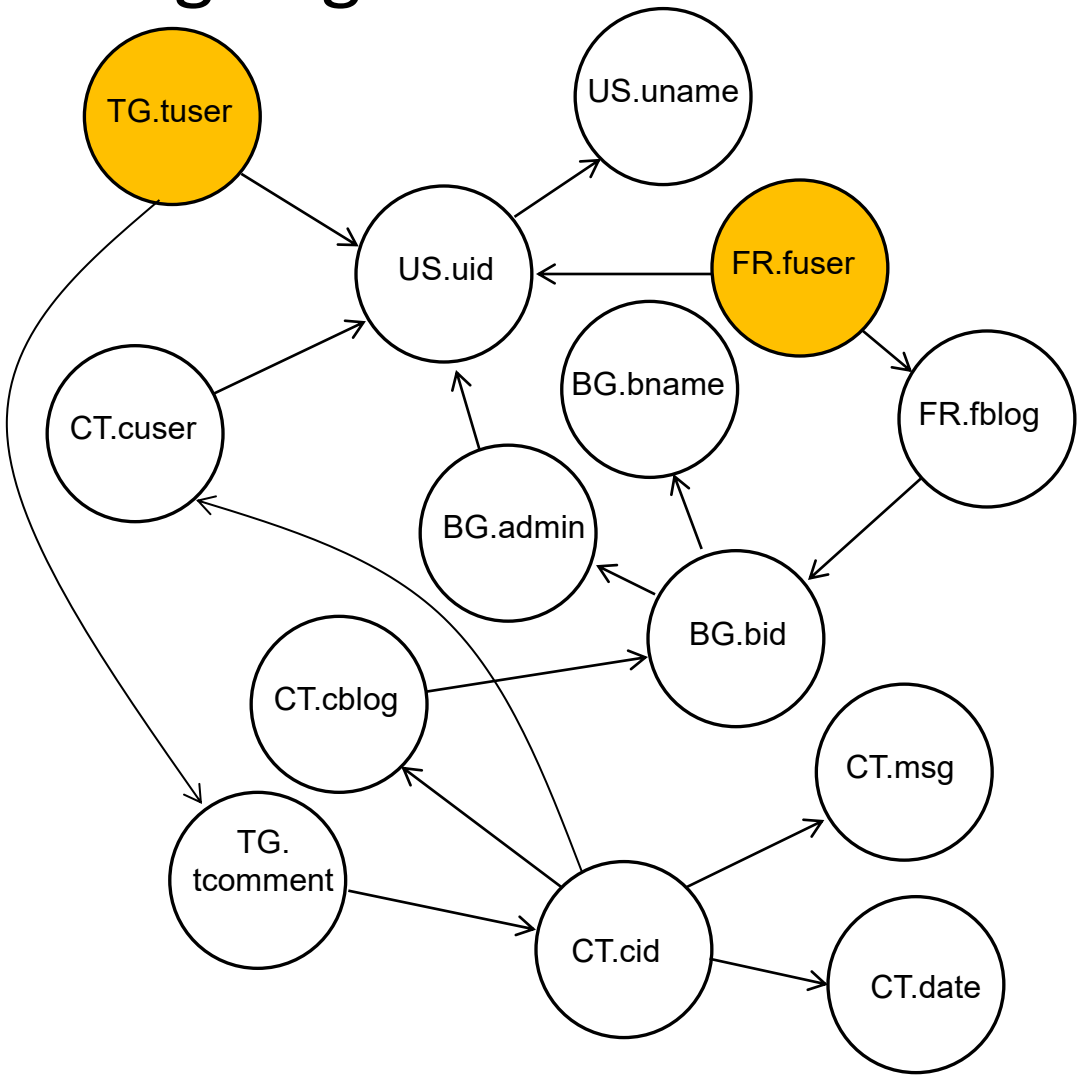
	tuser	tcomment
t_7	u02	c01

Blog (BG)

	bid	bname	admin
t_8	b01	Information Systems	u02
t_9	b02	Database	u01
t_{10}	b03	Computer Science	u02

Comment (CT)

	cid	cblog	cuser	msg	date
t_{11}	c01	b01	u01	Exactly what I was looking for!	25/02/2013



Ανάλυση κόμβων (συν.)

- Nodes with no out-going edges
  - No starting point in any path

User (US)

	uid	uname
t_1	u01	Date
t_2	u02	Hunt

Follower (FR)

	fuser	fblog
t_3	u01	b01
t_4	u01	b02
t_5	u01	b03
t_6	u02	b01

Tag (TG)

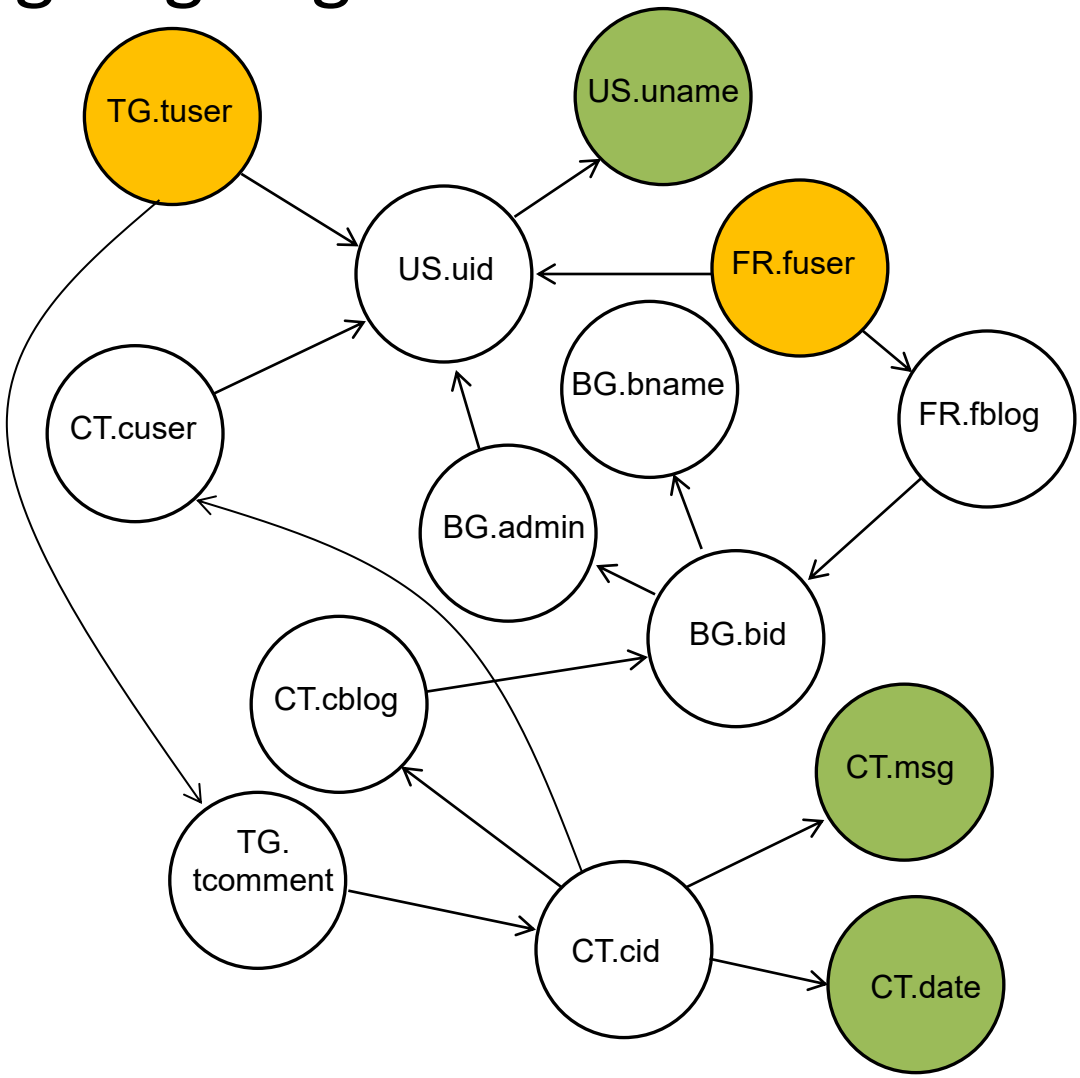
	tuser	tcomment
t_7	u02	c01

Blog (BG)

	bid	bname	admin
t_8	b01	Information Systems	u02
t_9	b02	Database	u01
t_{10}	b03	Computer Science	u02

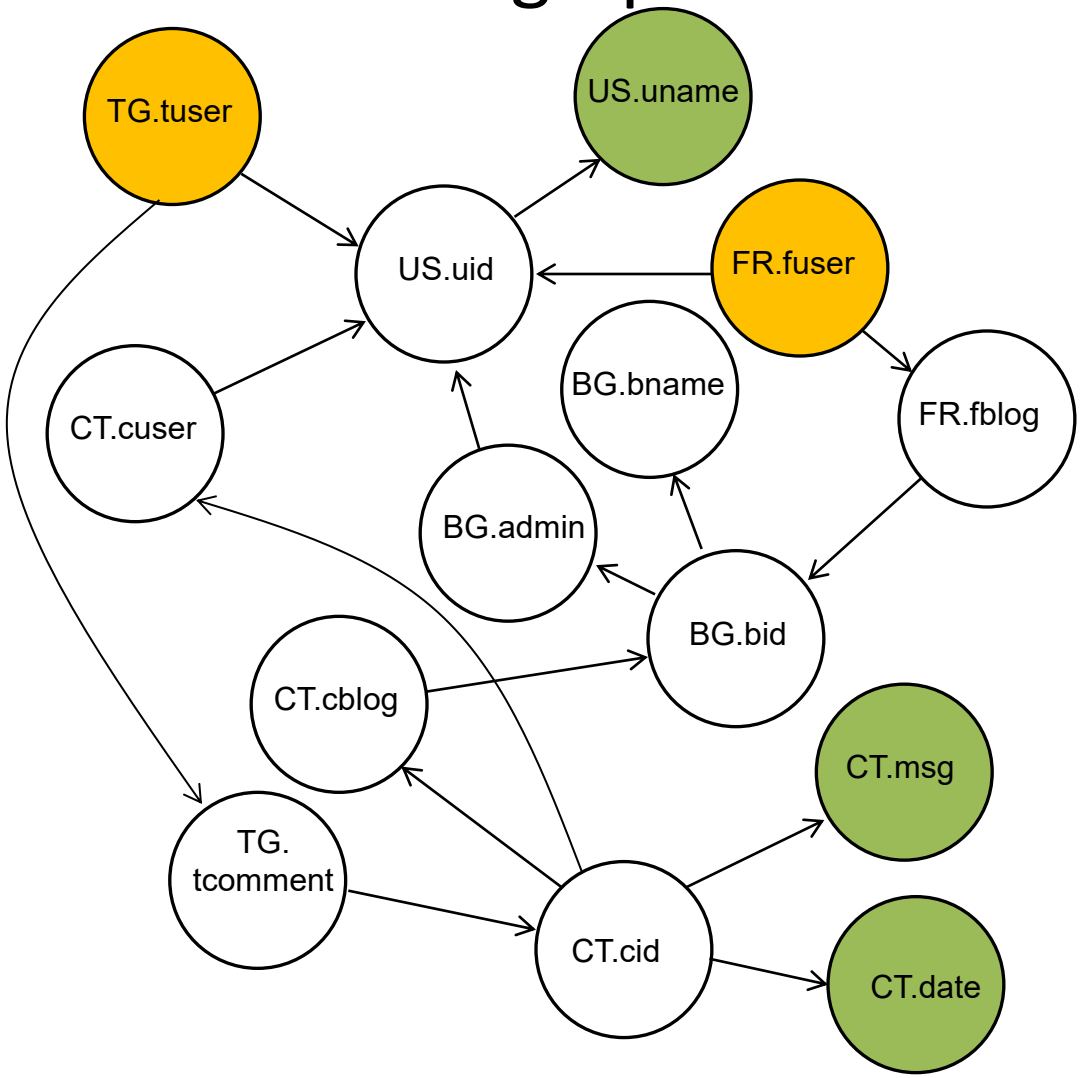
Comment (CT)

	cid	cblog	cuser	msg	date
t_{11}	c01	b01	u01	Exactly what I was looking for!	25/02/2013



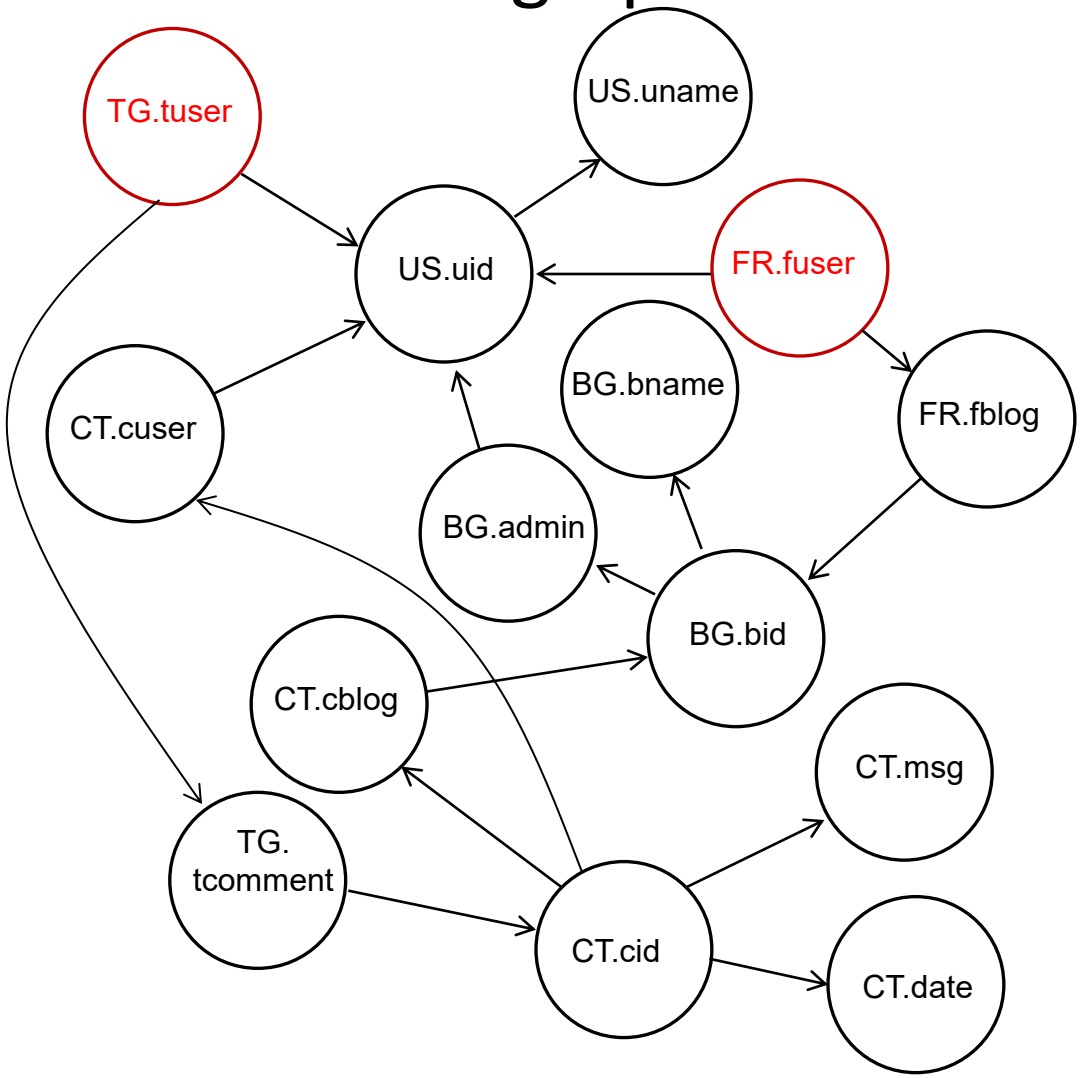
Path queries

- What can be derived from such a graph?



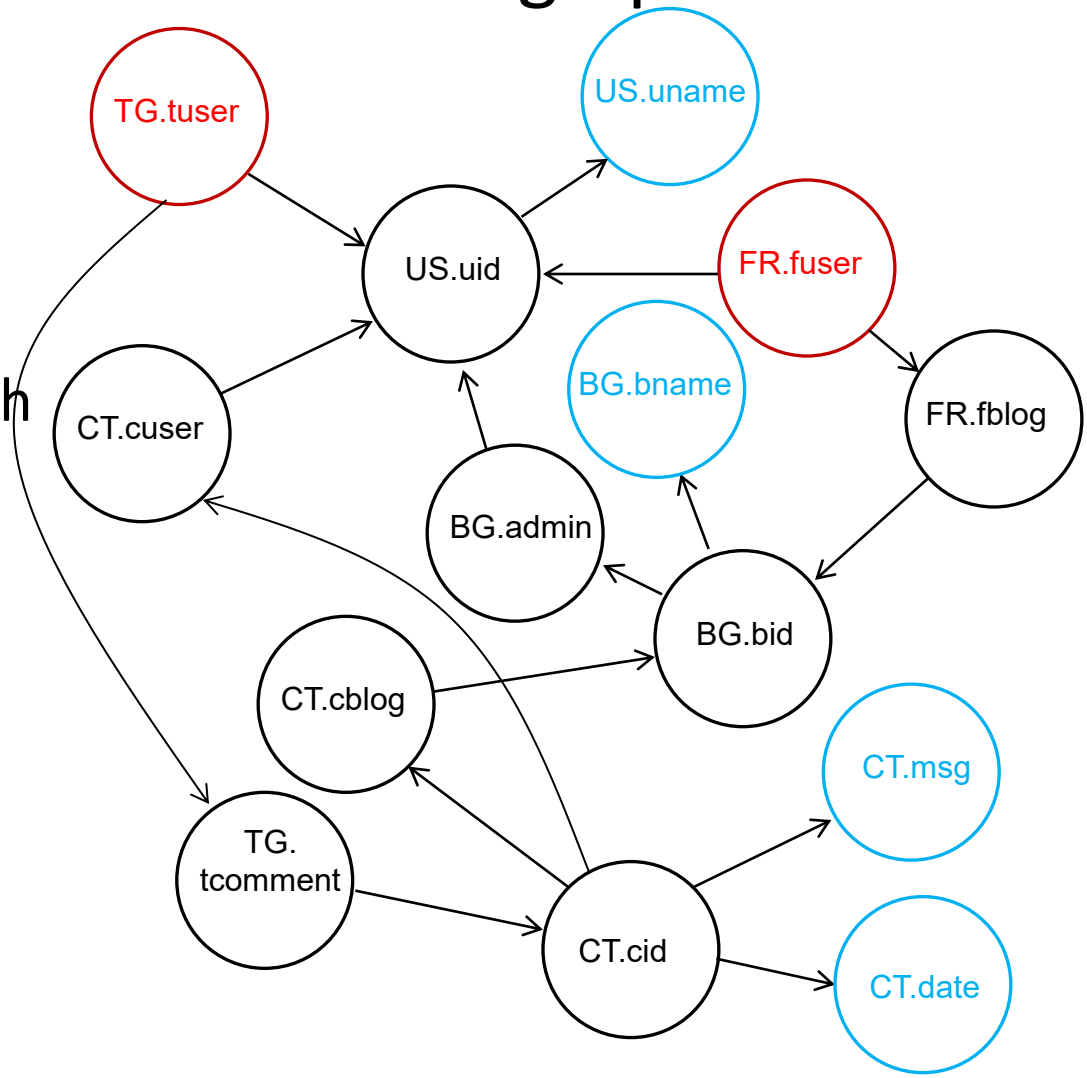
Path queries (cont.)

- What can be derived from such a graph?
  - Starting from a node with no in-coming edges (κόκκινο περίγραμμα)



Path queries (cont.)

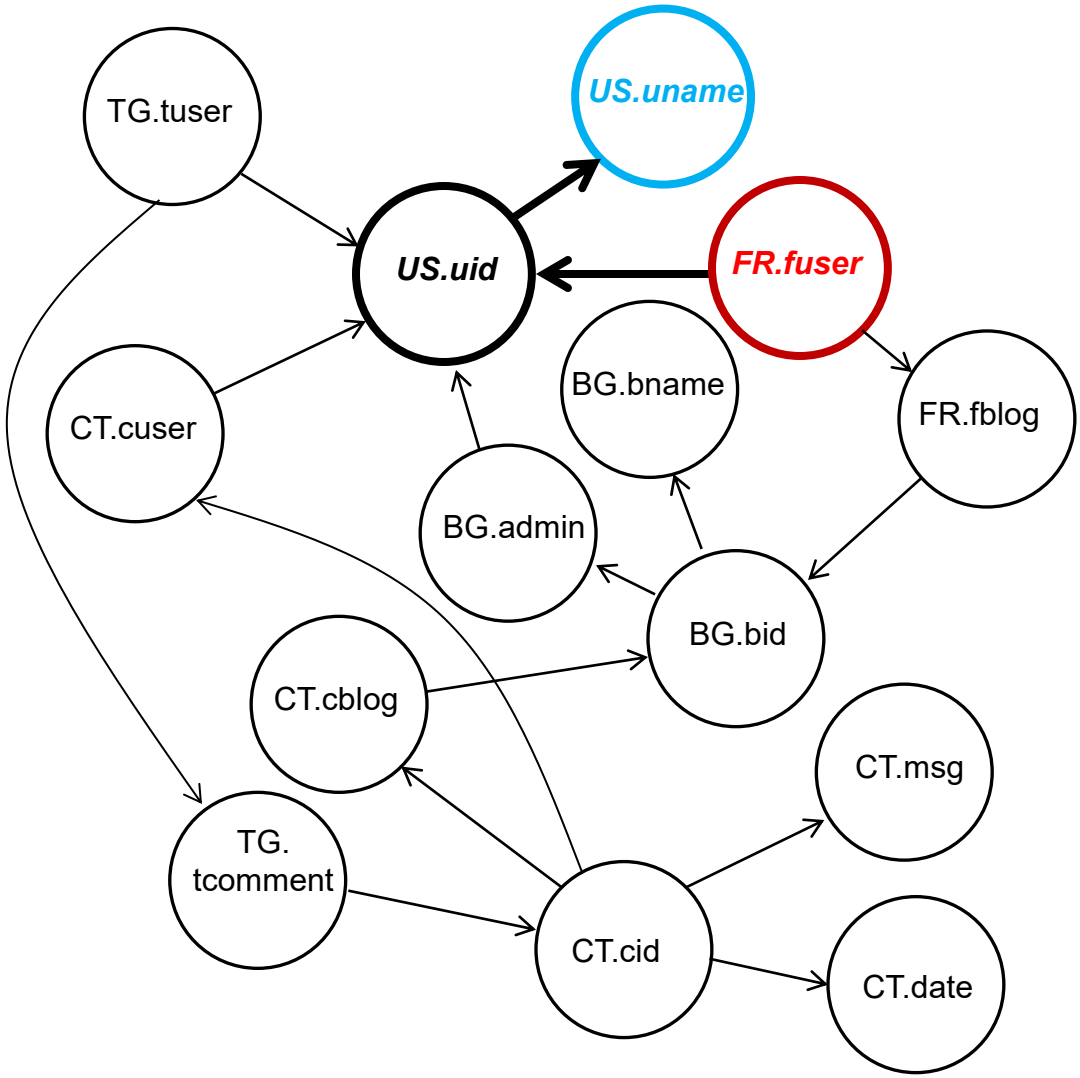
- What can be derived from such a graph?
  - Starting from a node with no in-coming edges (κόκκινο περίγραμμα)
  - Landing to a node with no out-going edges (μπλε περίγραμμα)



Path queries (cont.)

- Path query

sp1 : FR.fuser → US.uid → US.uname

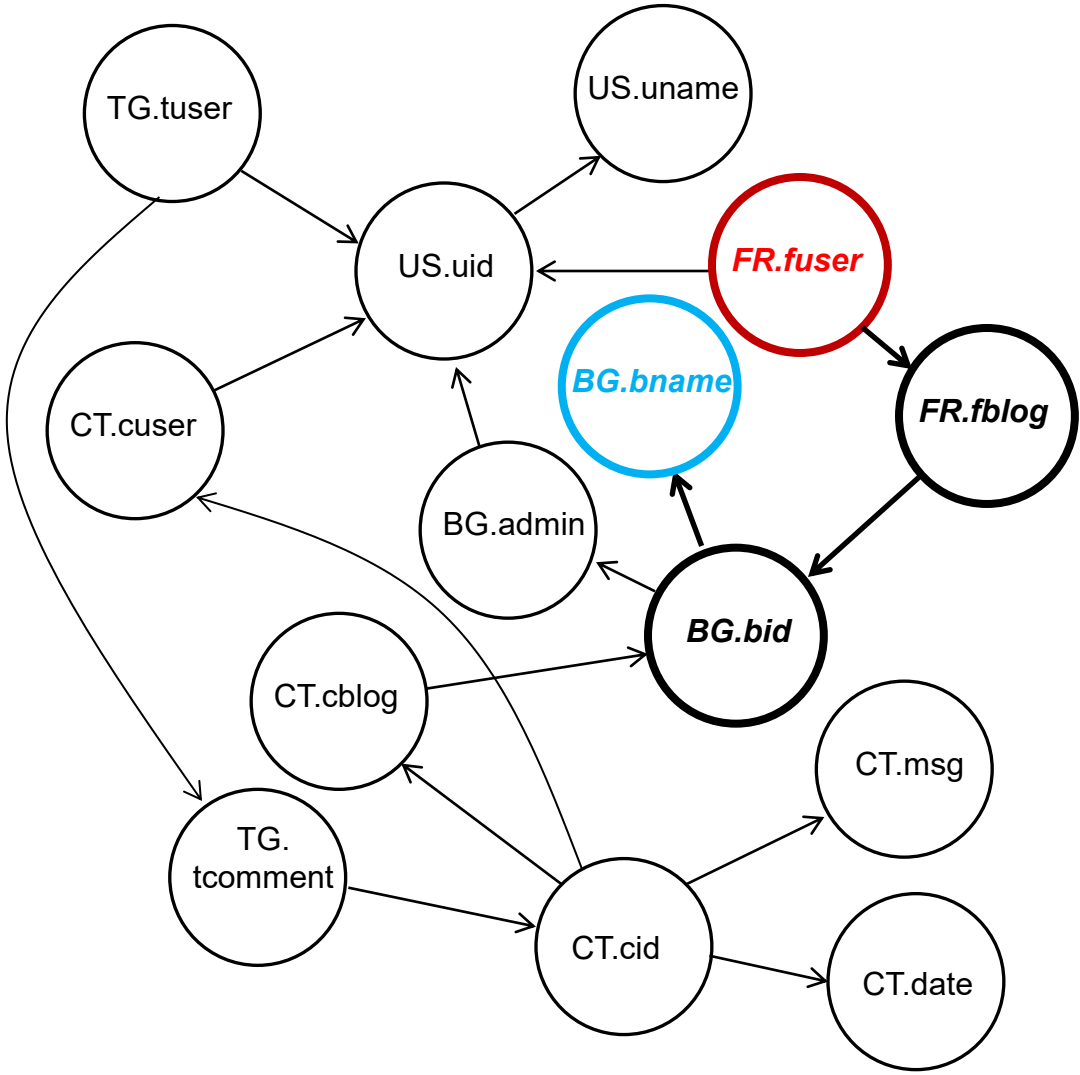


Path queries (cont.)

- Path query

sp1 : FR.fuser → US.uid → US.uname

sp2 : FR.fuser → FR.fblog → BG.bid → BG.bname



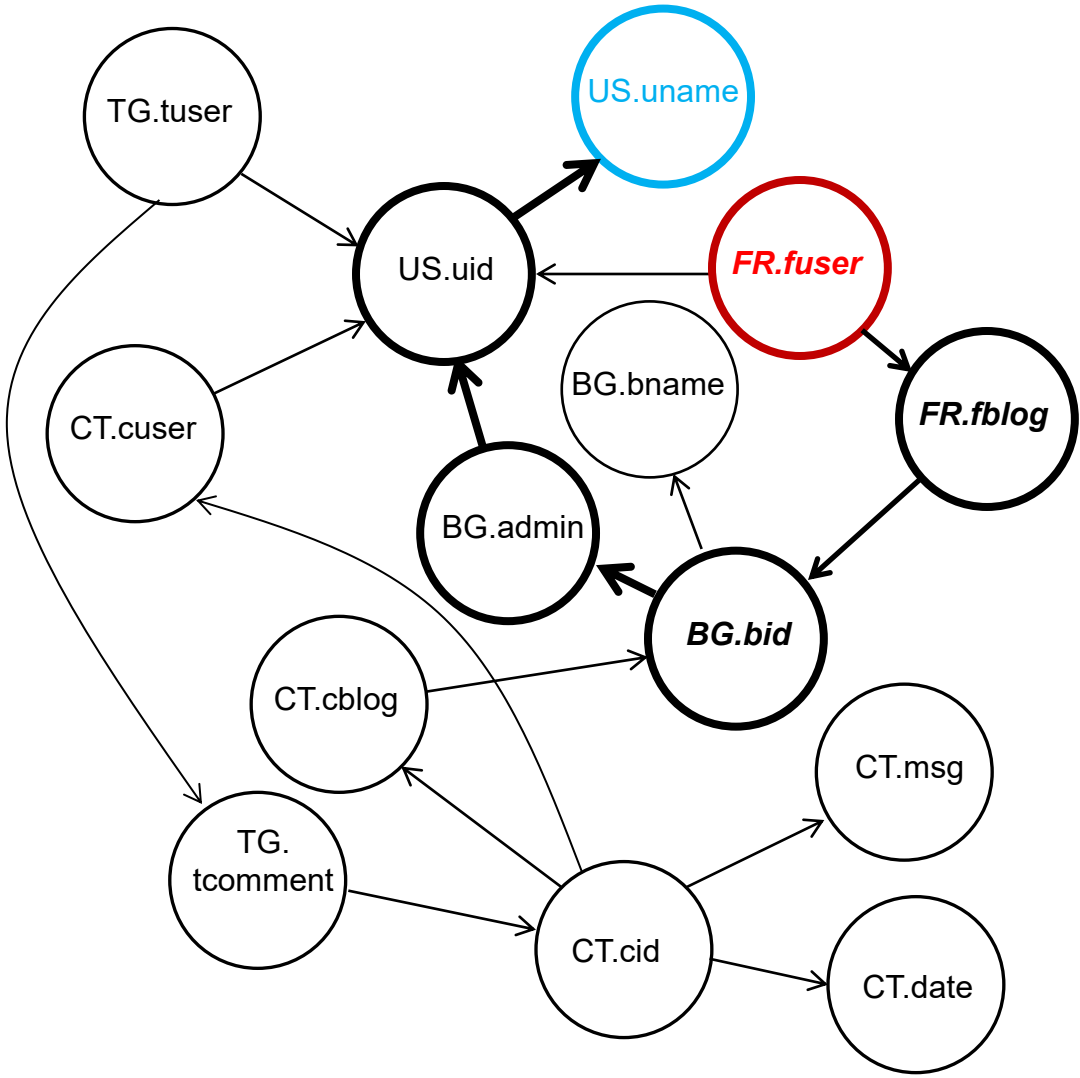
Path queries (cont.)

- Path query

sp1 : FR.fuser → US.uid → US.uname

sp2 : FR.fuser → FR.fblog → BG.bid → BG.bname

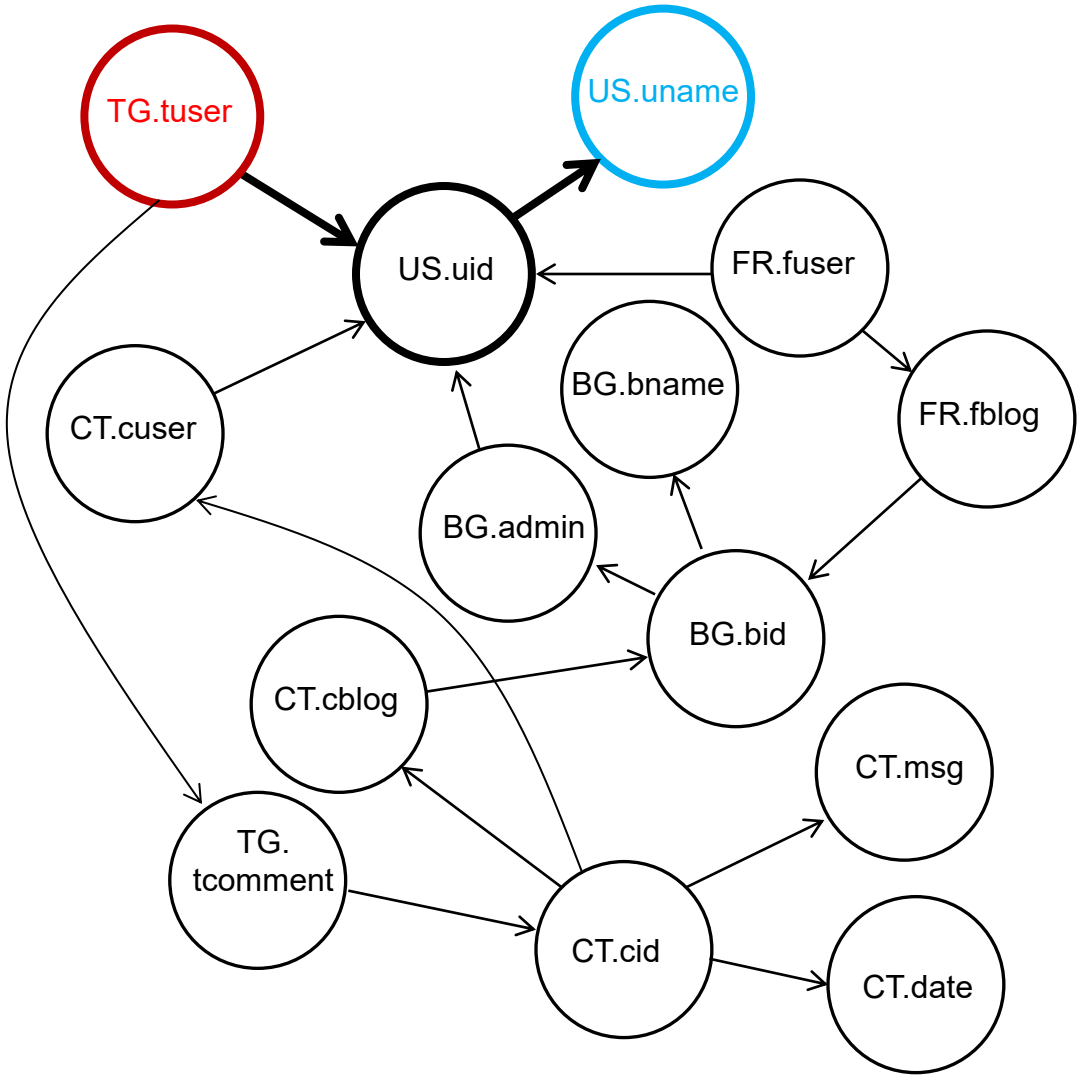
sp3 : FR.fuser → FR.fblog → BG.bid → BG.admin
→ US.uid → US.uname



Path queries (cont.)

- Path query

sp1 : FR.fuser → US.uid → US.uname
sp2 : FR.fuser → FR.fblog → BG.bid → BG.bname
sp3 : FR.fuser → FR.fblog → BG.bid → BG.admin
→ US.uid → US.uname
sp4 : TG.tuser → US.uid → US.uname



Path queries (cont.)

- Path query

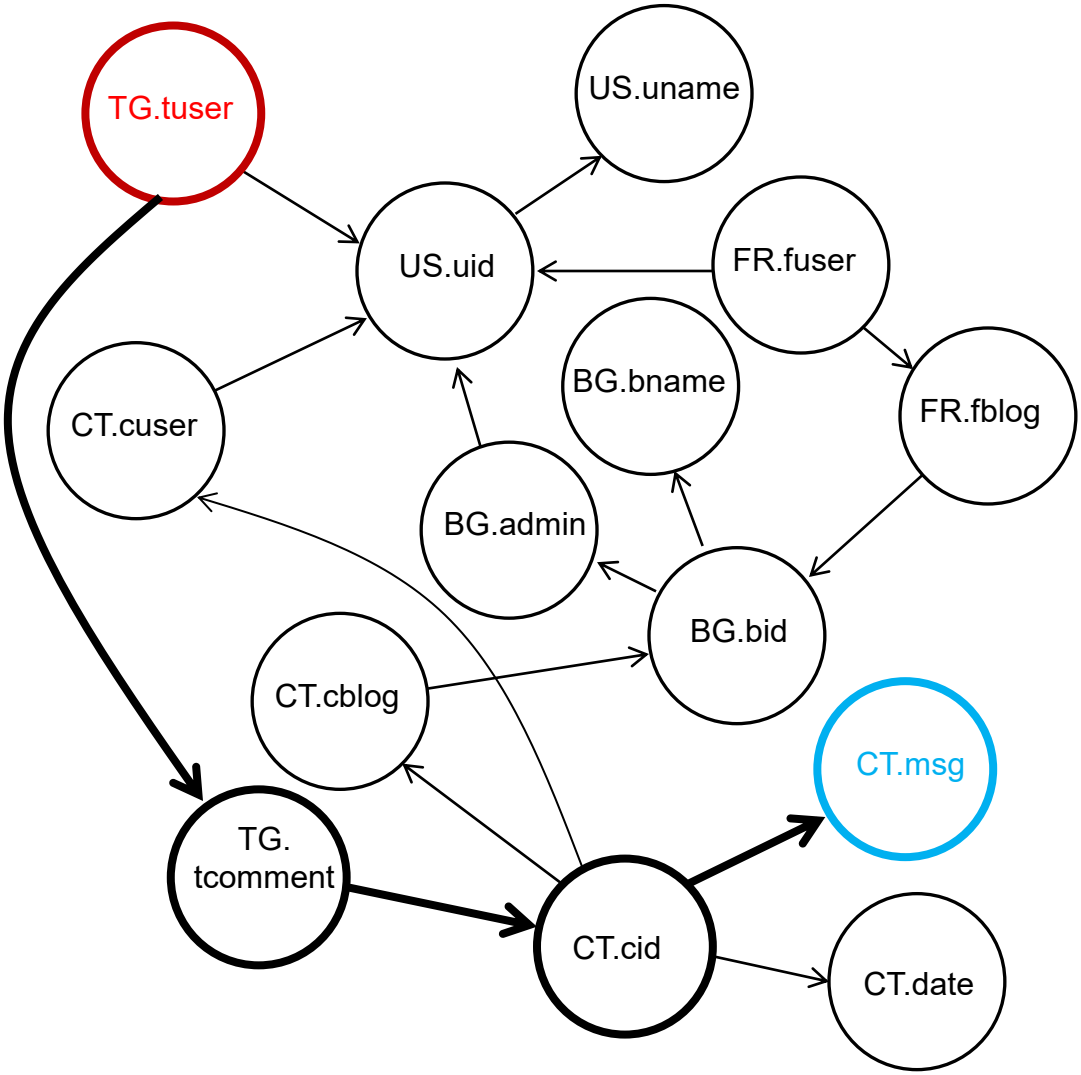
sp1 : FR.fuser → US.uid → US.uname

sp2 : FR.fuser → FR.fblog → BG.bid → BG.bname

sp3 : FR.fuser → FR.fblog → BG.bid → BG.admin
→ US.uid → US.uname

sp4 : TG.tuser → US.uid → US.uname

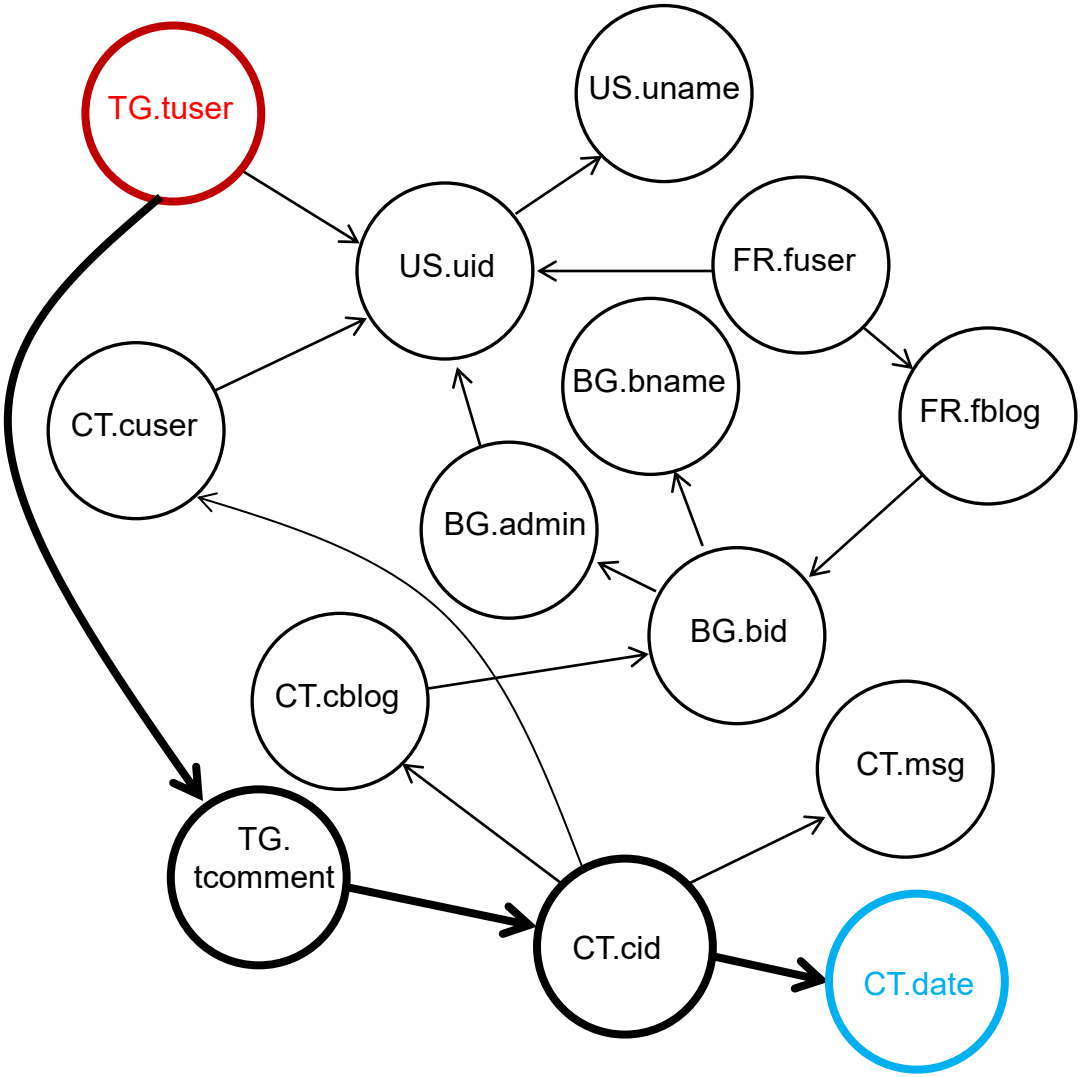
sp5 : TG.tuser → TG.tcomment → CT.cid → CT.msg



Path queries (cont.)

- Path query

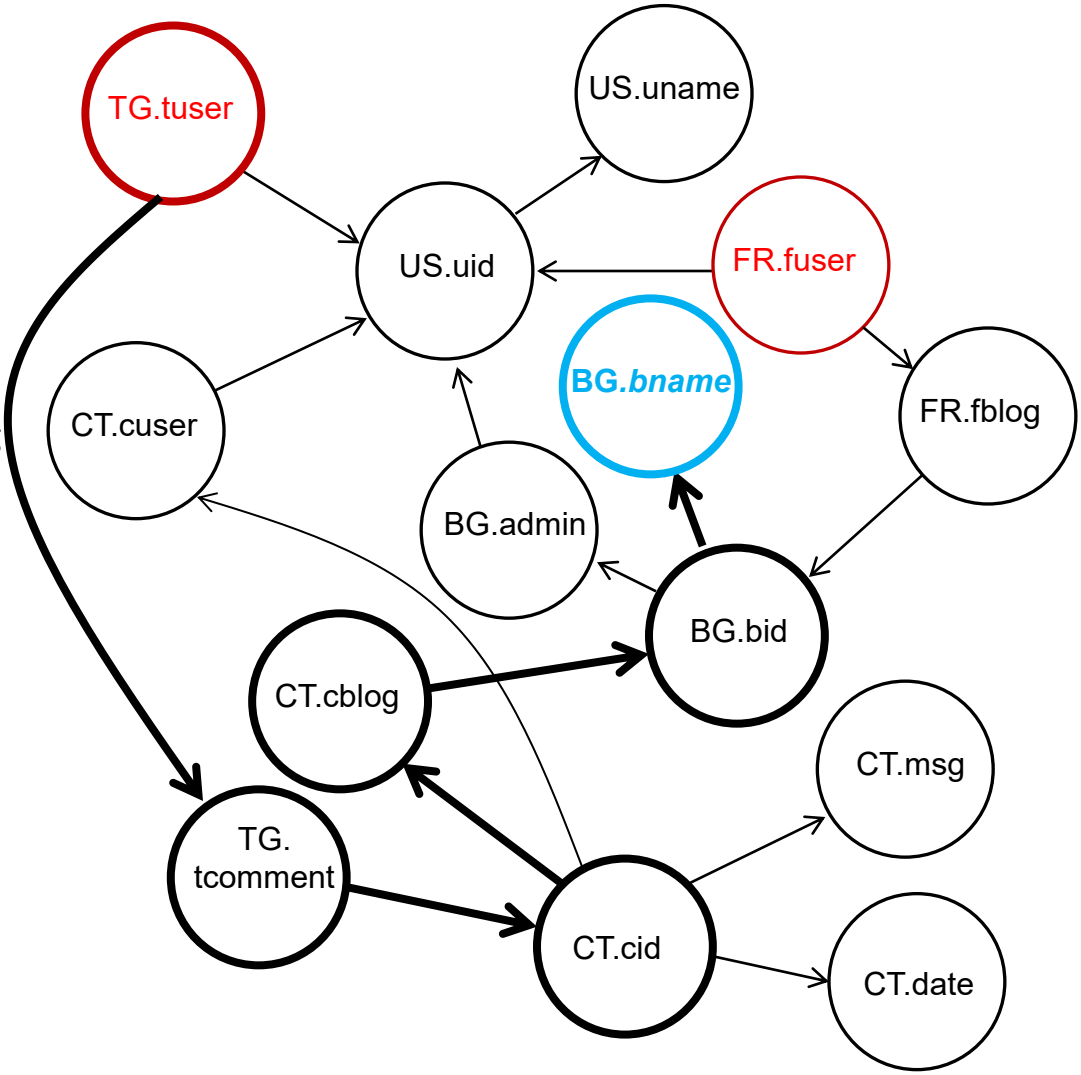
sp1 : FR.fuser → US.uid → US.uname
sp2 : FR.fuser → FR.fblog → BG.bid → BG.bname
sp3 : FR.fuser → FR.fblog → BG.bid → BG.admin
→ US.uid → US.uname
sp4 : TG.tuser → US.uid → US.uname
sp5 : TG.tuser → TG.tcomment → CT.cid → CT.msg
sp6 : TG.tuser → TG.tcomment → CT.cid → CT.date



Path queries (cont.)

- Path query

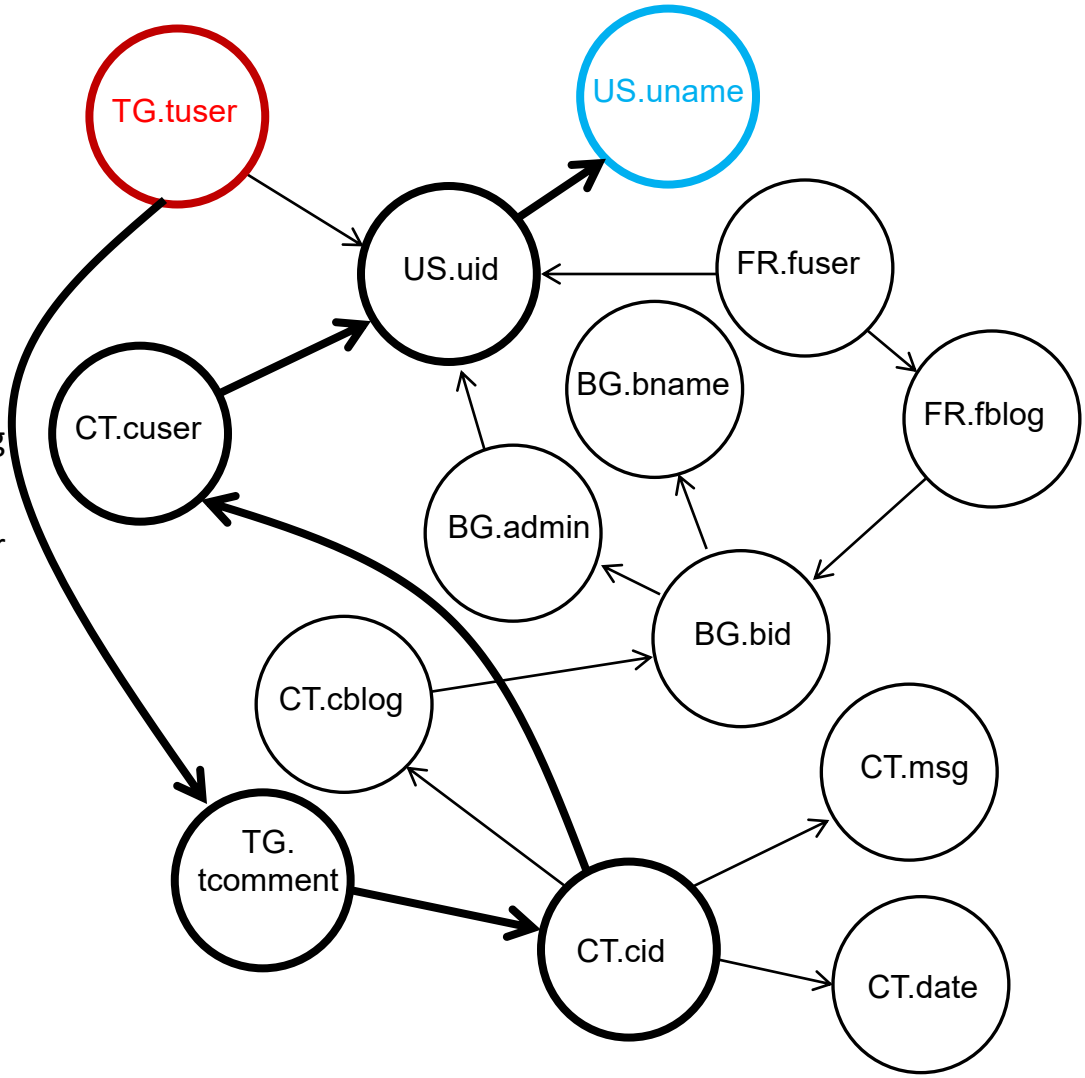
sp1 : FR.fuser → US.uid → US.uname
sp2 : FR.fuser → FR.fblog → BG.bid → BG.bname
sp3 : FR.fuser → FR.fblog → BG.bid → BG.admin
→ US.uid → US.uname
sp4 : TG.tuser → US.uid → US.uname
sp5 : TG.tuser → TG.tcomment → CT.cid → CT.msg
sp6 : TG.tuser → TG.tcomment → CT.cid → CT.date
sp7 : TG.tuser → TG.tcomment → CT.cid → CT.cblog
→ BG.bid → BG.bname



Path queries (cont.)

- Path query

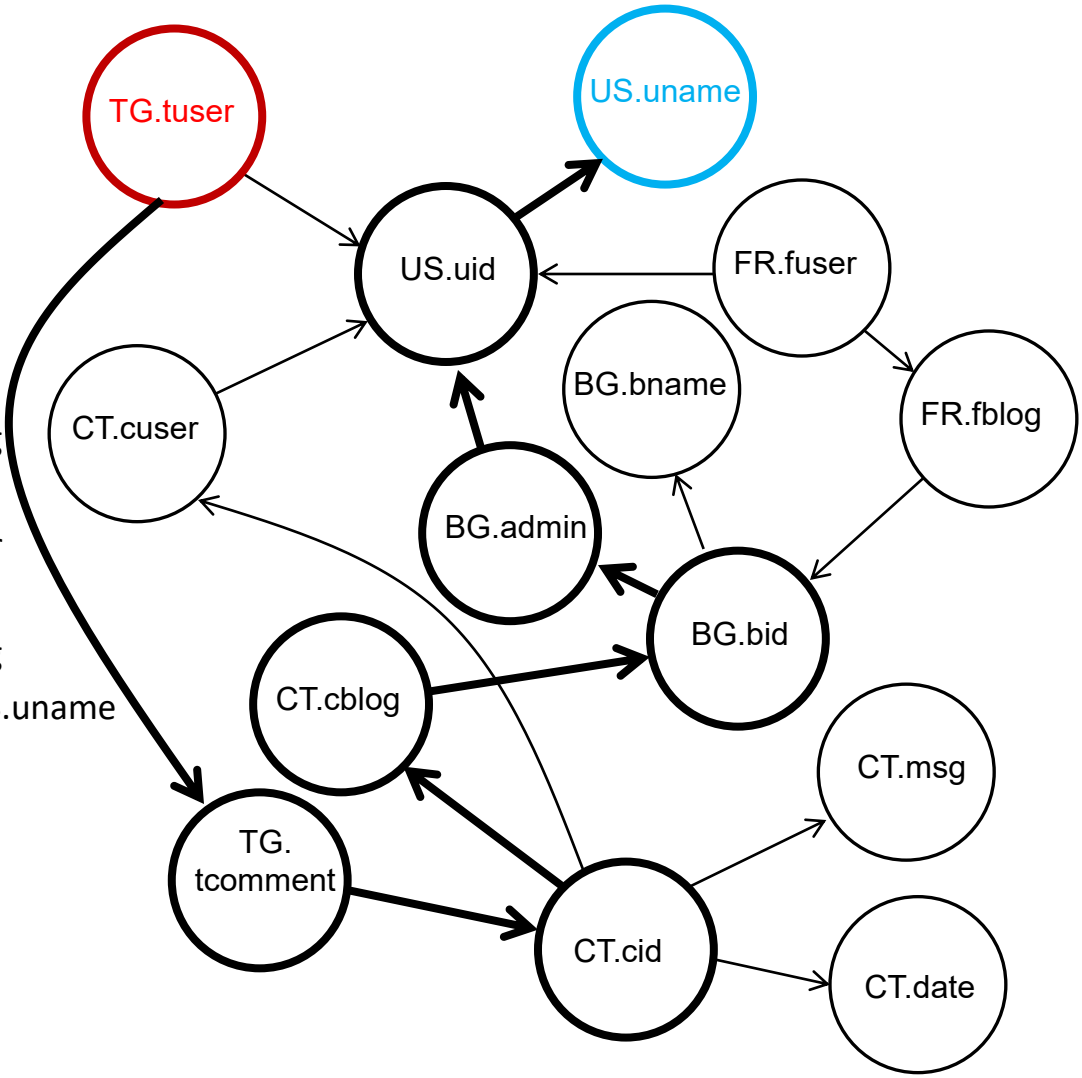
sp1 : FR.fuser → US.uid → US.uname
sp2 : FR.fuser → FR.fblog → BG.bid → BG.bname
sp3 : FR.fuser → FR.fblog → BG.bid → BG.admin
→ US.uid → US.uname
sp4 : TG.tuser → US.uid → US.uname
sp5 : TG.tuser → TG.tcomment → CT.cid → CT.msg
sp6 : TG.tuser → TG.tcomment → CT.cid → CT.date
sp7 : TG.tuser → TG.tcomment → CT.cid → CT.cblog
→ BG.bid → BG.bname:
sp8 : TG.tuser → TG.tcomment → CT.cid → CT.cuser
→ US.uid → US.uname



Path queries (cont.)

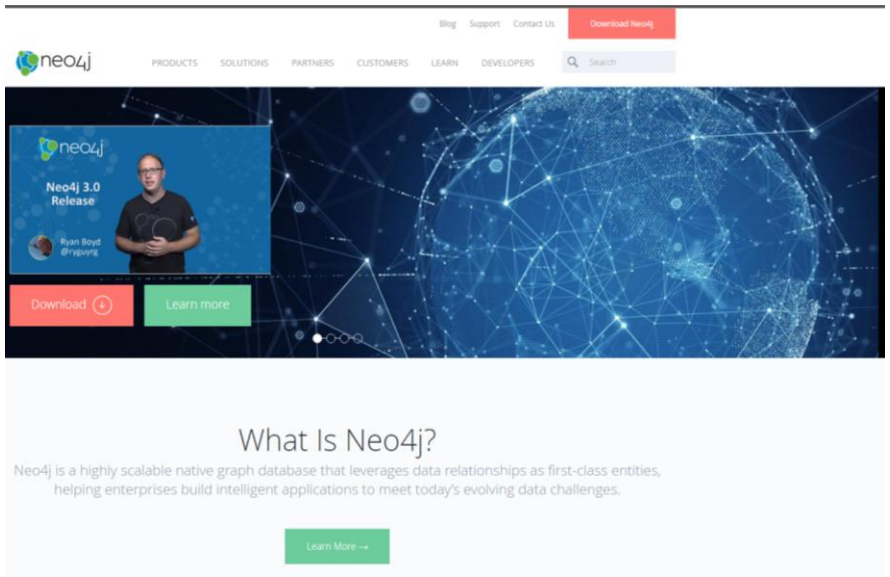
- Path query

sp1 : FR.fuser → US.uid → US.uname
sp2 : FR.fuser → FR.fblog → BG.bid → BG.bname
sp3 : FR.fuser → FR.fblog → BG.bid → BG.admin
→ US.uid → US.uname
sp4 : TG.tuser → US.uid → US.uname
sp5 : TG.tuser → TG.tcomment → CT.cid → CT.msg
sp6 : TG.tuser → TG.tcomment → CT.cid → CT.date
sp7 : TG.tuser → TG.tcomment → CT.cid → CT.cblog
→ BG.bid → BG.bname:
sp8 : TG.tuser → TG.tcomment → CT.cid → CT.cuser
→ US.uid → US.uname
sp9 : TG.tuser → TG.tcomment → CT.cid → CT.cblog
→ BG.bid → BG.admin → US.uid → US.uname



Implementing a property graph

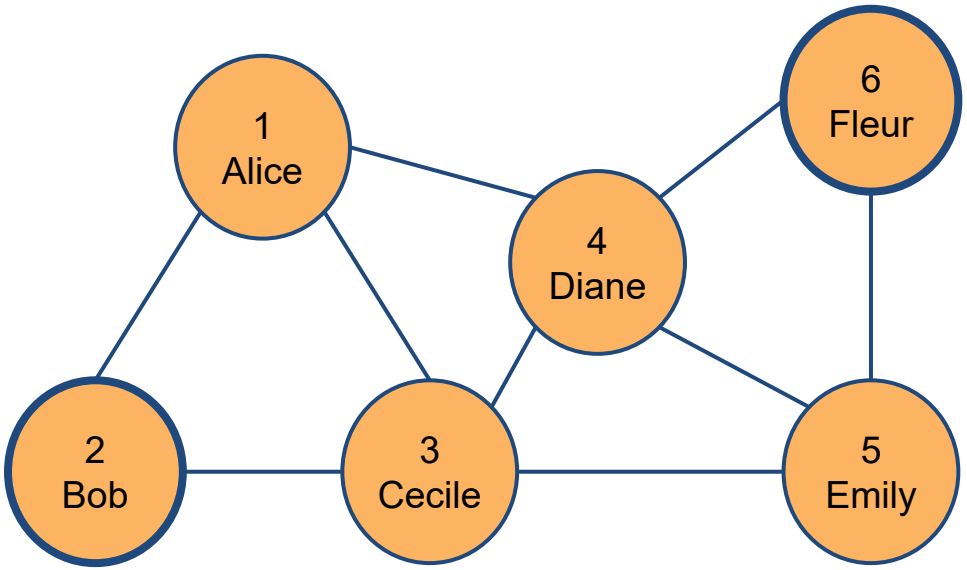
- Neo4j
  - Launched in 2010
  - Open source
  - Java-based
  - NoSQL Graph Database
- http://neo4j.com/



Άσκηση στη τάξη

Ερώτηση

- Τι είδους αναπαράσταση είναι η ακόλουθη;



Person

id [PK]	name
1	Alice
2	Bob
3	Cecile
4	Diane
5	Emily
6	Fleur

knows

person1 id [FK]	person2id [FK]
1	2
1	3
1	4
2	3
3	4
3	5
4	5
4	6
5	6
all edges backwards	

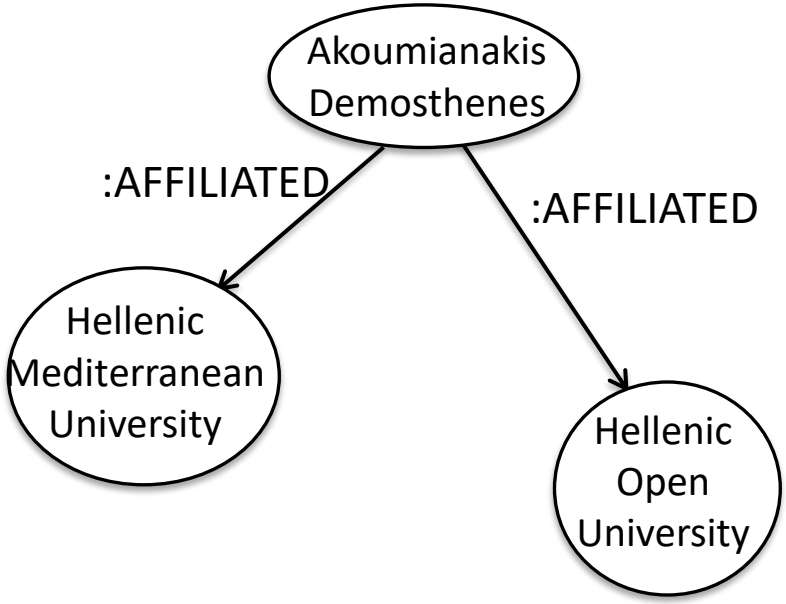
Άσκηση στην τάξη

Παράδειγμα

- Θεωρήστε καθηγητές και τα πανεπιστήμια που εργάζονται

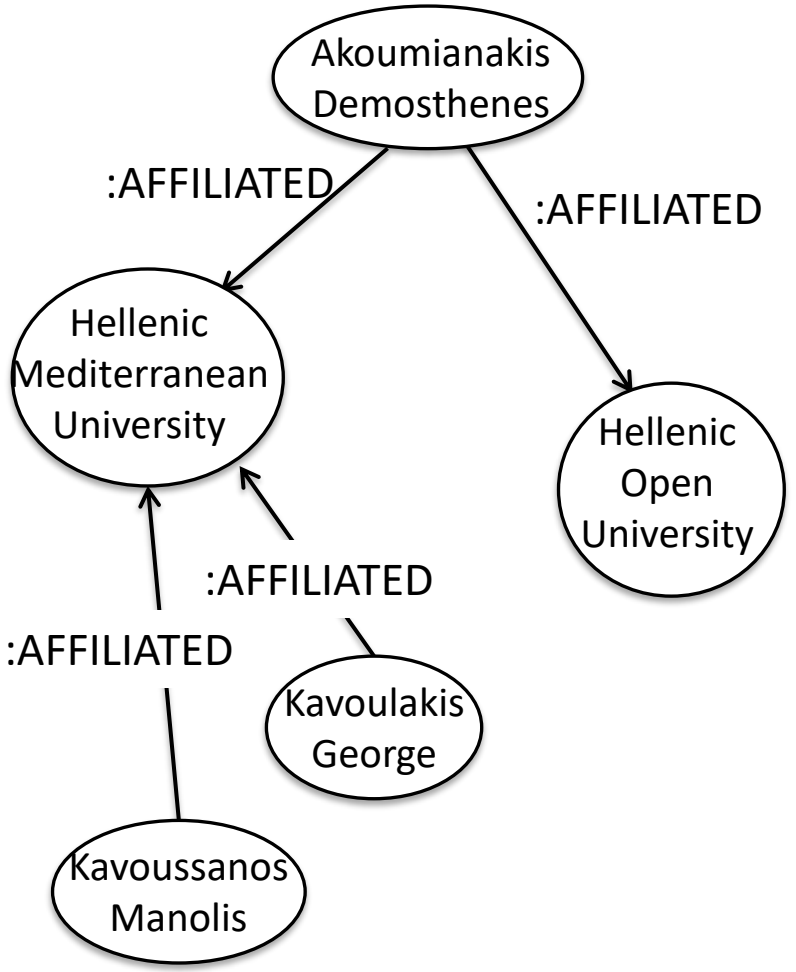
Παράδειγμα

- Θεωρήστε καθηγητές και τα πανεπιστήμια που εργάζονται
  - Prof. Akoumianakis is affiliated with HMU and HOU



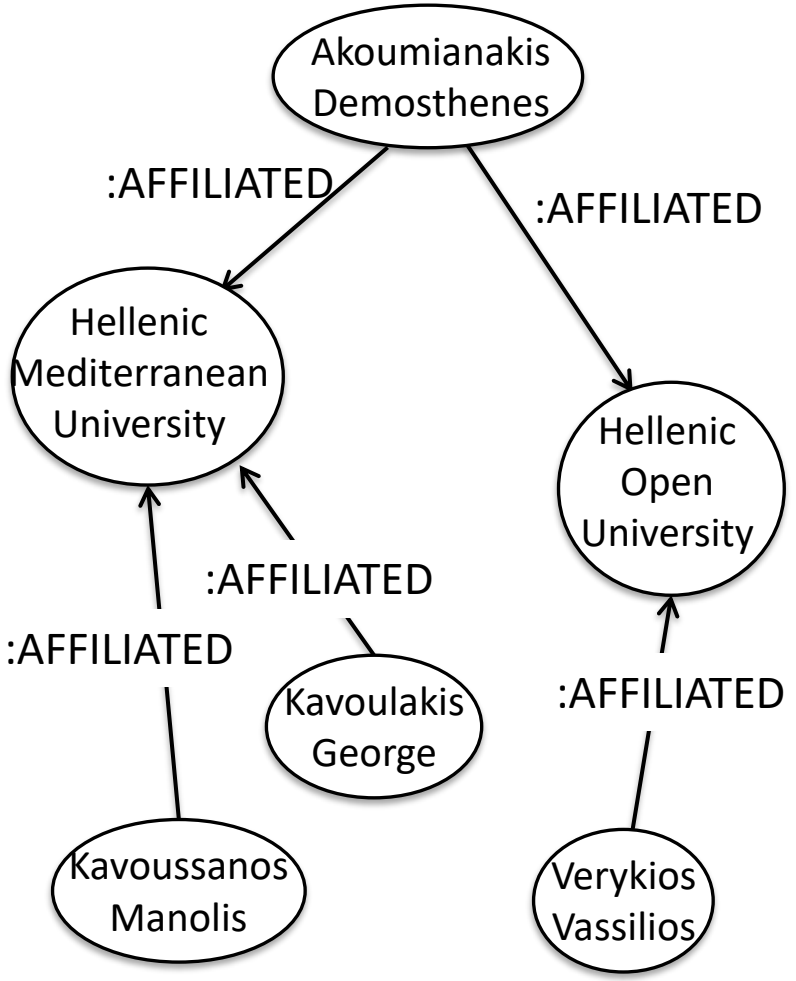
Παράδειγμα (συν.)

- Θεωρήστε καθηγητές και τα πανεπιστήμια που εργάζονται
  - Prof. Akoumianakis is affiliated with HMU and HOU
  - Prof. Kavoulakis and Prof. Kavoussanos are affiliated with HMU



Παράδειγμα (συν.)

- Θεωρήστε καθηγητές και τα πανεπιστήμια που εργάζονται
  - Prof. Akoumianakis is affiliated with HMU and HOU
  - Prof. Kavoulakis and Prof. Kavoussanos are affiliated with HMU
  - Prof. Verykios is affiliated with HOU



Σχεσιακή αναπαράσταση

- Τρεις πίνακες με κατάλληλους περιορισμούς

Prof_id	Firstname	Lastname	email
Demos	Demosthenes	Akoumianakis	da@hmu.gr
George	Georgios	Kavoulakis	gk@hmu.gr
Manos	Emmanouil	Kavoussanos	mk@hmu.gr
Billy	Vasilios	Verykios	vv@hou.gr

Uni_id	University
HMU	Hellenic Mediterranean University
HOU	Hellenic Open University

Prof_id	Uni_id
Demos	HMU
Demos	HOU
Manos	HMU
George	HMU
Billy	HOU

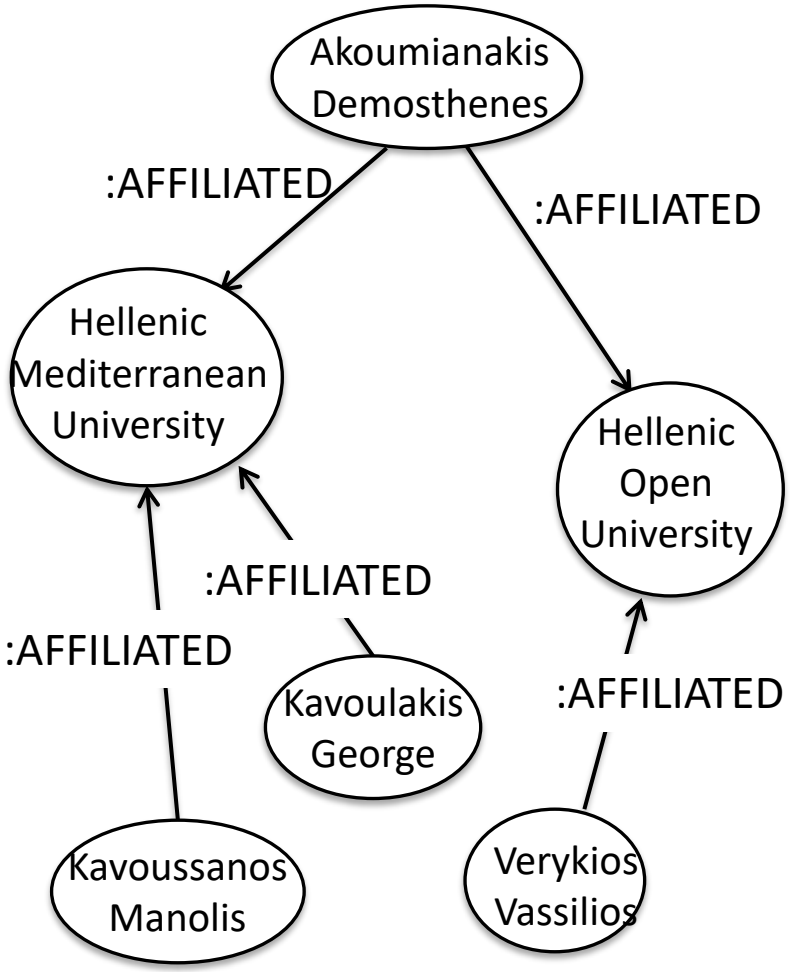
Σύγκριση

- Relational vs. property graph

Prof_id	Firstname	Lastname	email
Demos	Demosthenes	Akoumianakis	da@hmu.gr
George	Georgios	Kavoulakis	gk@hmu.gr
Manos	Emmanouil	Kavoussanos	mk@hmu.gr
Billy	Vasilios	Verykios	vv@hou.gr

Uni_id	University
HMU	Hellenic Mediterranean University
HOU	Hellenic Open University

Prof_id	Uni_id
Demos	HMU
Demos	HOU
Manos	HMU
George	HMU
Billy	HOU



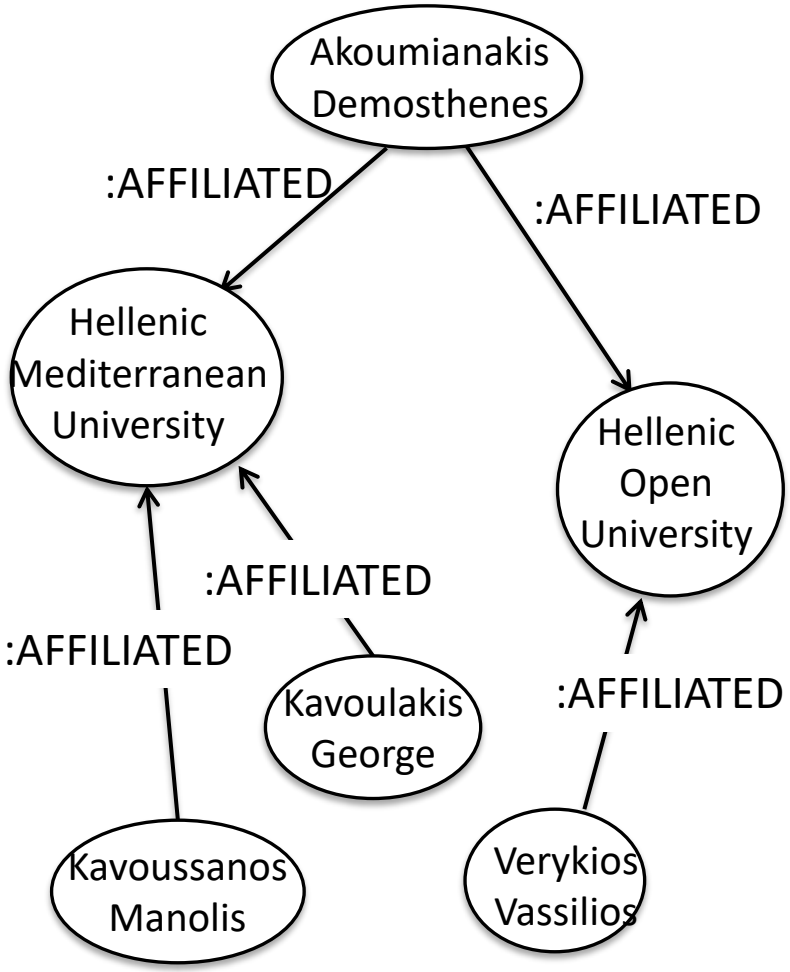
Ερώτηση – How do we represent

- … ότι ο Verykios endorses Akoumianakis

Prof_id	Firstname	Lastname	email
Demos	Demosthenes	Akoumianakis	da@hmu.gr
George	Georgios	Kavoulakis	gk@hmu.gr
Manos	Emmanouil	Kavoussanos	mk@hmu.gr
Billy	Vasilios	Verykios	vv@hou.gr

Uni_id	University
HMU	Hellenic Mediterranean University
HOU	Hellenic Open University

Prof_id	Uni_id
Demos	HMU
Demos	HOU
Manos	HMU
George	HMU
Billy	HOU



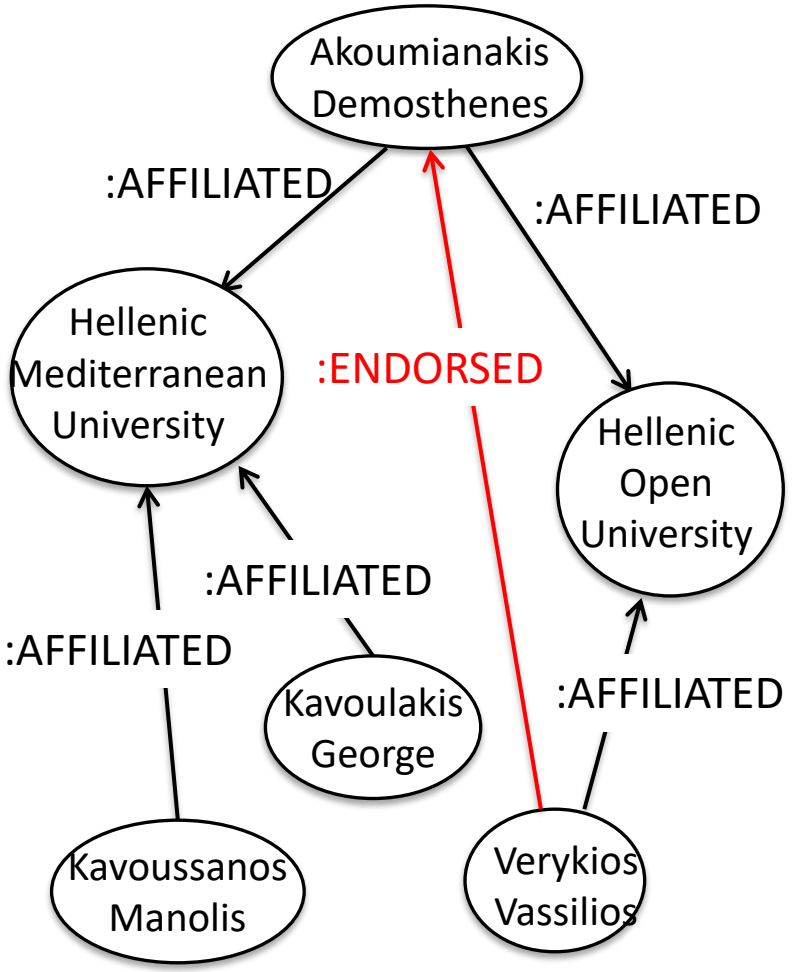
Representing endorsements

- … στο property graph

Prof_id	Firstname	Lastname	email
Demos	Demosthenes	Akoumianakis	da@hmu.gr
George	Georgios	Kavoulakis	gk@hmu.gr
Manos	Emmanouil	Kavoussanos	mk@hmu.gr
Billy	Vasilios	Verykios	vv@hou.gr

Uni_id	University
HMU	Hellenic Mediterranean University
HOU	Hellenic Open University

Prof_id	Uni_id
Demos	HMU
Demos	HOU
Manos	HMU
George	HMU
Billy	HOU



Representing endorsements

- … στο σχεσιακό σχήμα

Prof_id	Firstname	Lastname	email
Demos	Demosthenes	Akoumianakis	da@hmu.gr
George	Georgios	Kavoulakis	gk@hmu.gr
Manos	Emmanouil	Kavoussanos	mk@hmu.gr
Billy	Vasilios	Verykios	vv@hou.gr

Uni_id	University
HMU	Hellenic Mediterranean University
HOU	Hellenic Open University

Prof_id	Uni_id
Demos	HMU
Demos	HOU
Manos	HMU
George	HMU
Billy	HOU

Prof_id_1	Prof_id_2
Billy	Demos
…	…