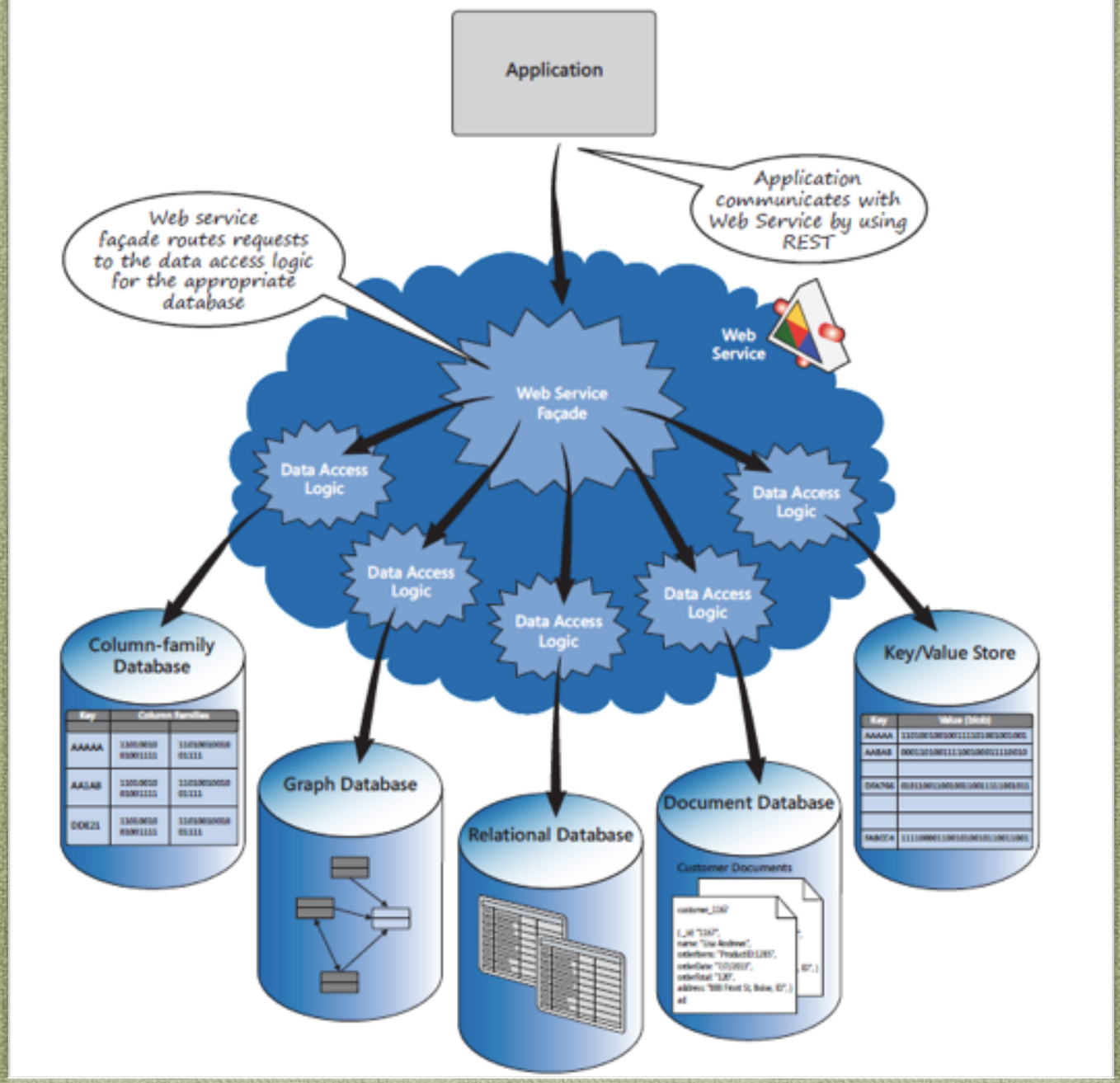
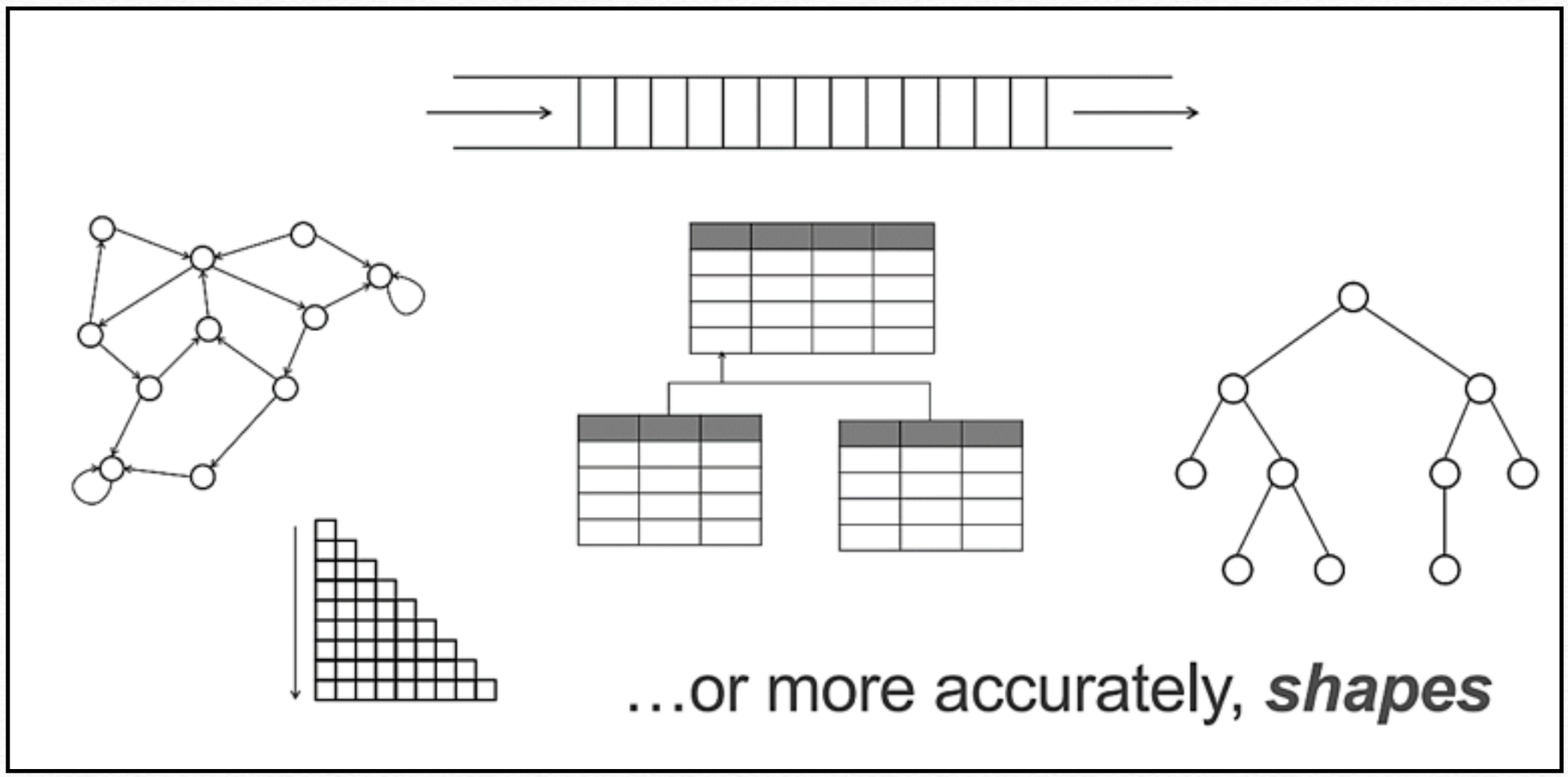


Polyglot Data Management

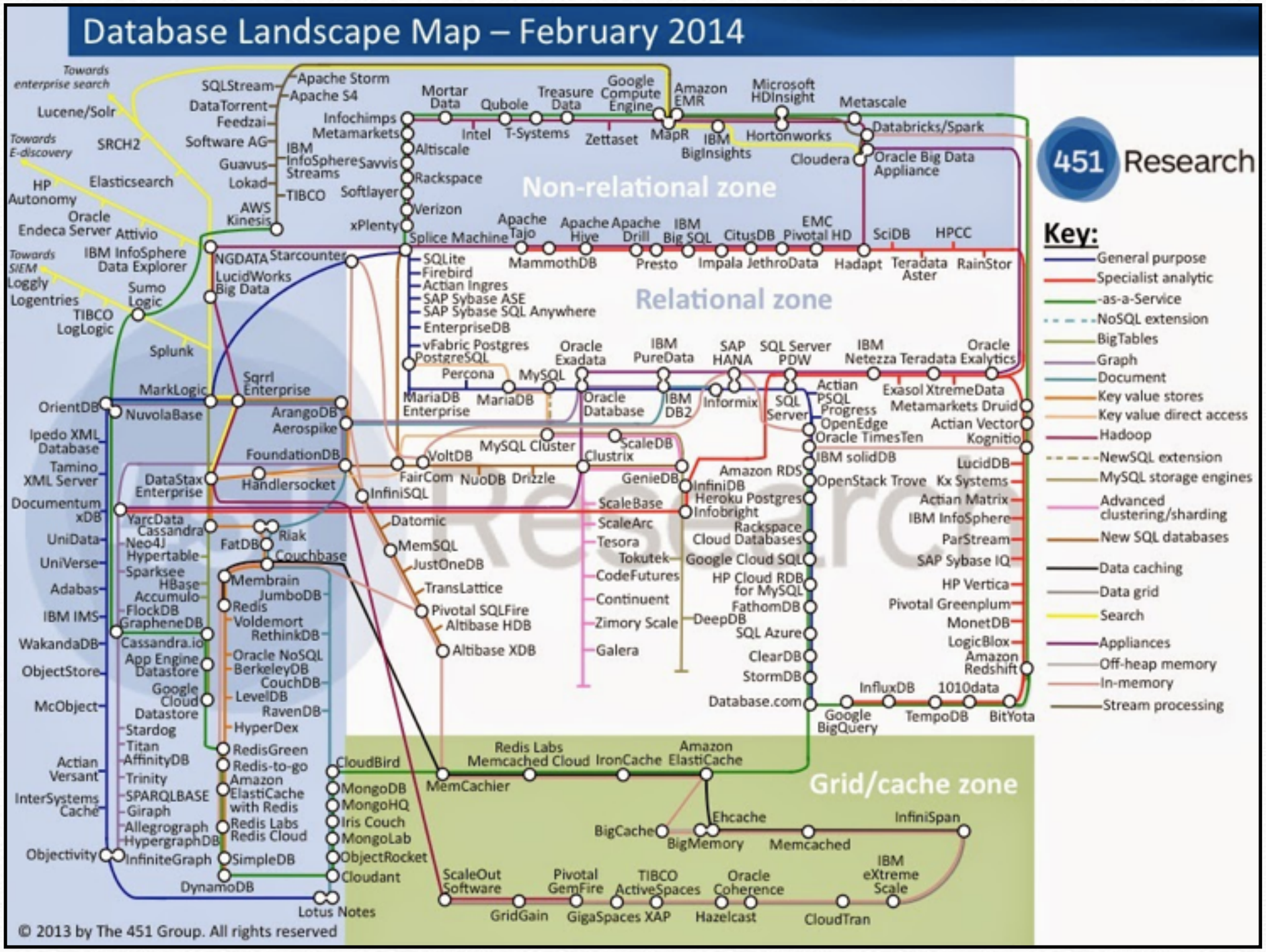
17/05/2017 - Big Data 2017

Different types of data shapes

- Data can have a variety of **shapes**. String cubes, graphs, relational tables, and trees are all examples of different data shapes



Different types of databases

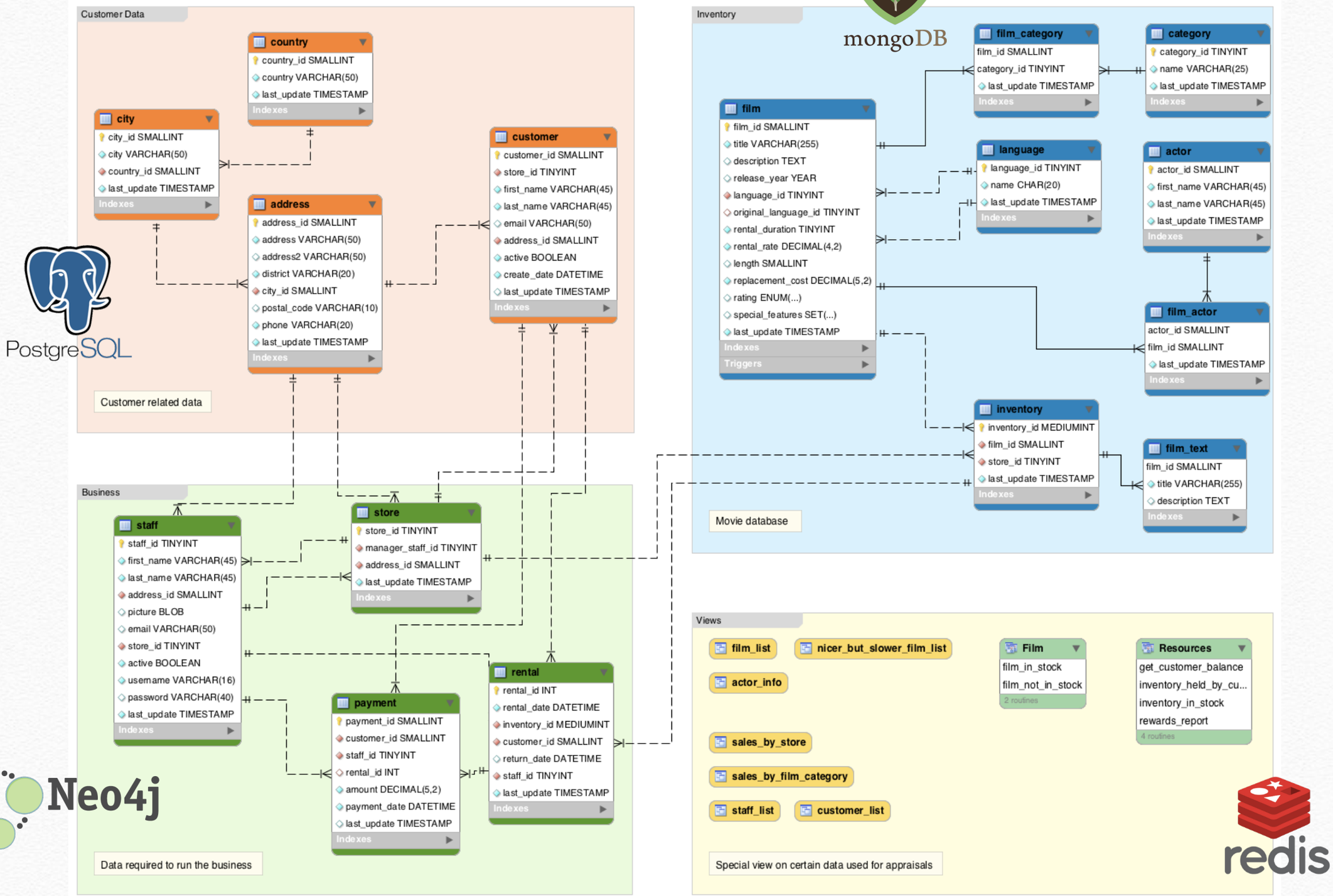


Polyglot Data Management

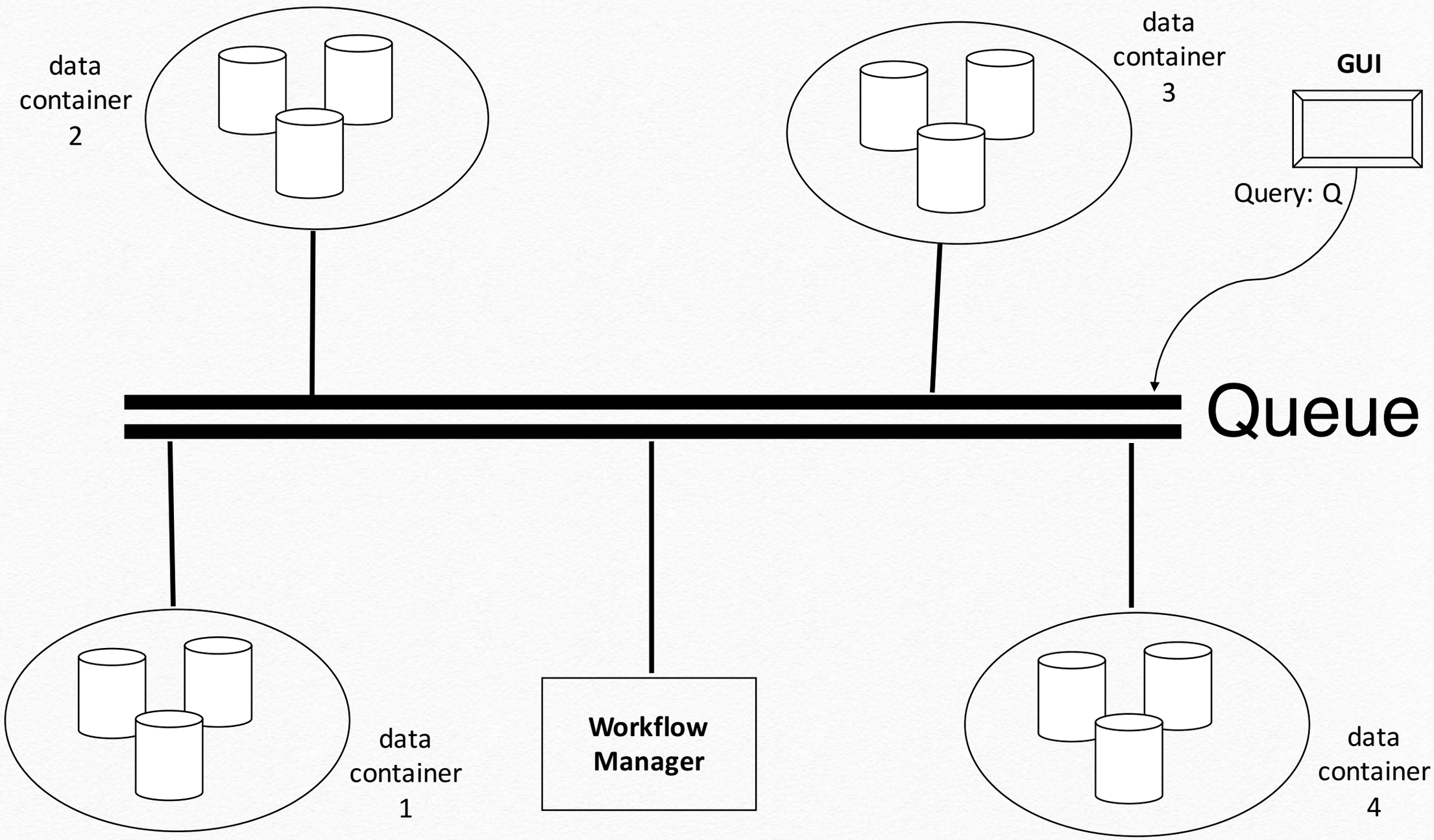
- A *polyglot data management platform* gives users the flexibility to choose the right tool for the job, instead of being forced into a solution that might not be optimal.

- The polyglot approach to data management must consider the capabilities associated with data as it flows through the information architecture from Acquisition to the General Core to Access.

Sample Scenario: a set of data containers



Communication between data containers



Sample Scenario: technology plethora

- RDBMS (PostgreSql)

- Document Store (MongoDB)

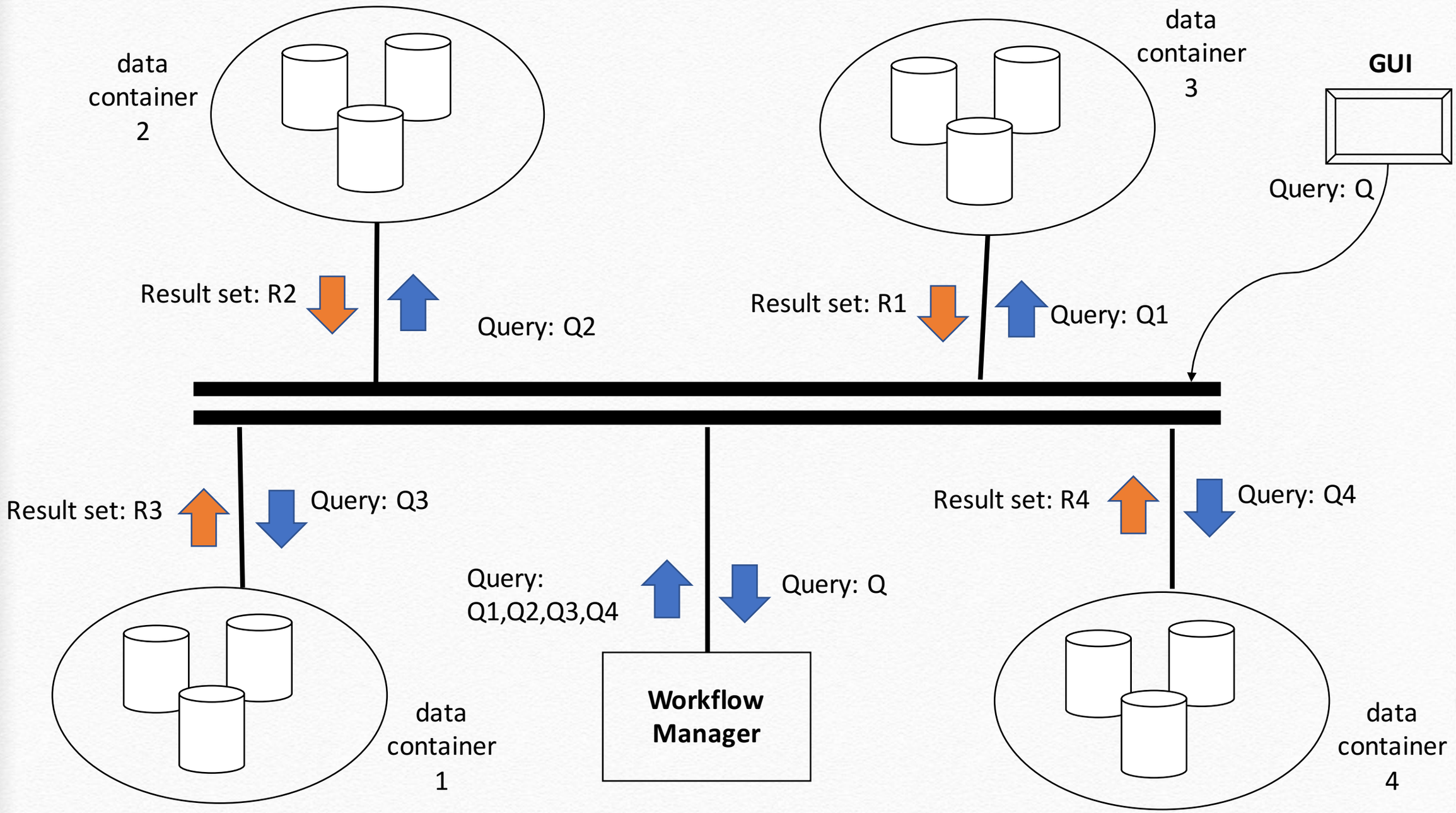
- Key-Value Store (Redis)

- GraphDB (Neo4J)

Sample Scenario: query language plethora

- RDBMS (SQL) PostgreSQL
- Document Store (SQL binding, Java Object) mongoDB
- Key-Value Store (Java Object) redis
- GraphDB (Cypher) Neo4j

USE CASE

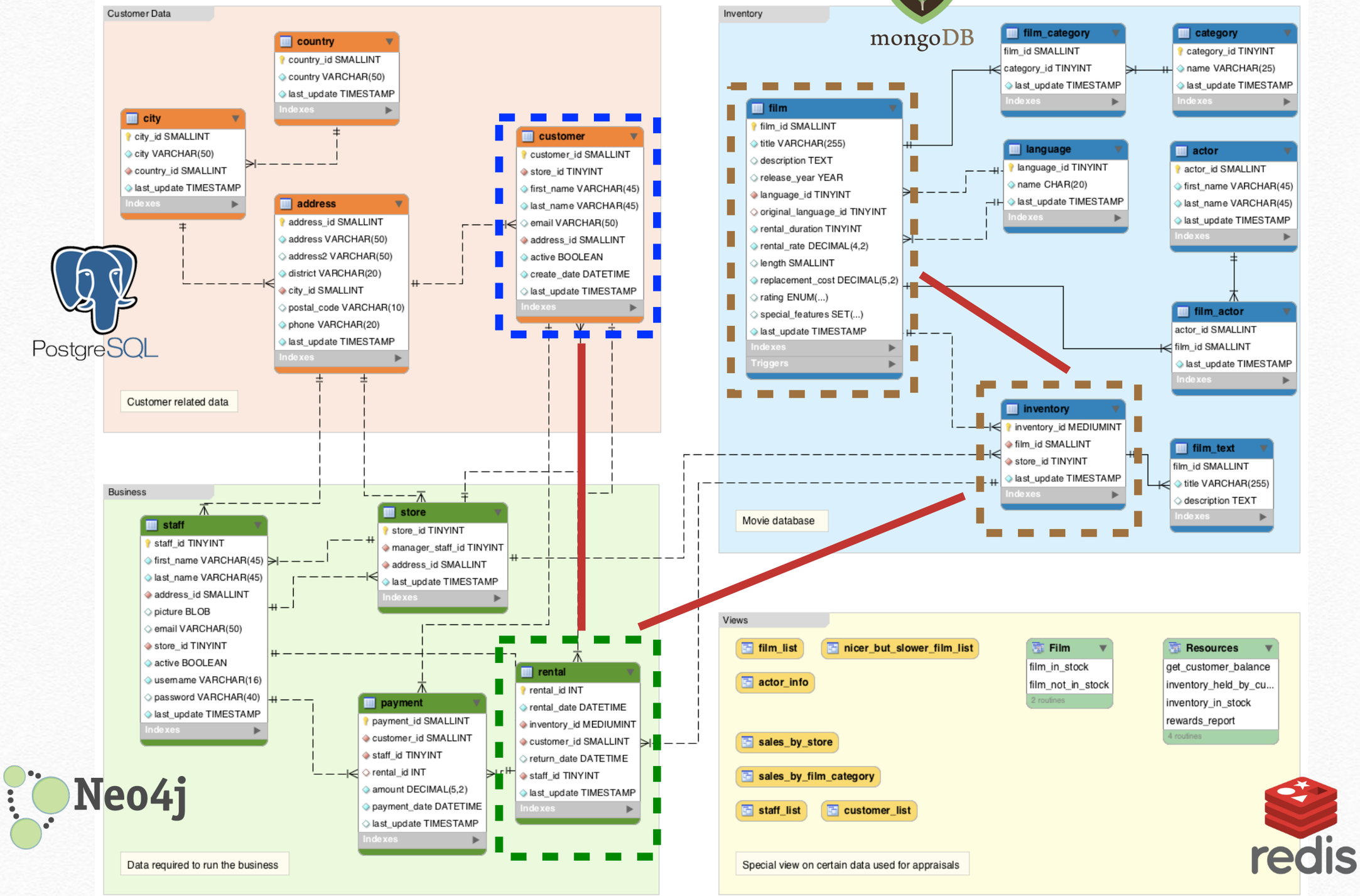

data container 2
data container 3
GUI
Query: Q
Result set: R2
Query: Q2
Result set: R1
Query: Q1
Result set: R3
Query: Q3
Query: Q1,Q2,Q3,Q4
Query: Q
Result set: R4
Query: Q4
Workflow Manager
data container 1
data container 4

Sample USE CASE

- Query Q: select last name of customers having rented the film with title “Titanic”
- Language: SQL

```
SELECT C.Last_Name
FROM Customer C, Rental R, Inventory I, Film F
WHERE C.customer_ID = R.customer_ID AND
      R.inventory_ID = I.inventory_id AND
      I.film_id = F.film_id AND F.title = “Titanic”
```

Sample USE CASE



Sample USE CASE

Q1

```
SELECT Last_Name FROM Customer WHERE customer_id = ?
```

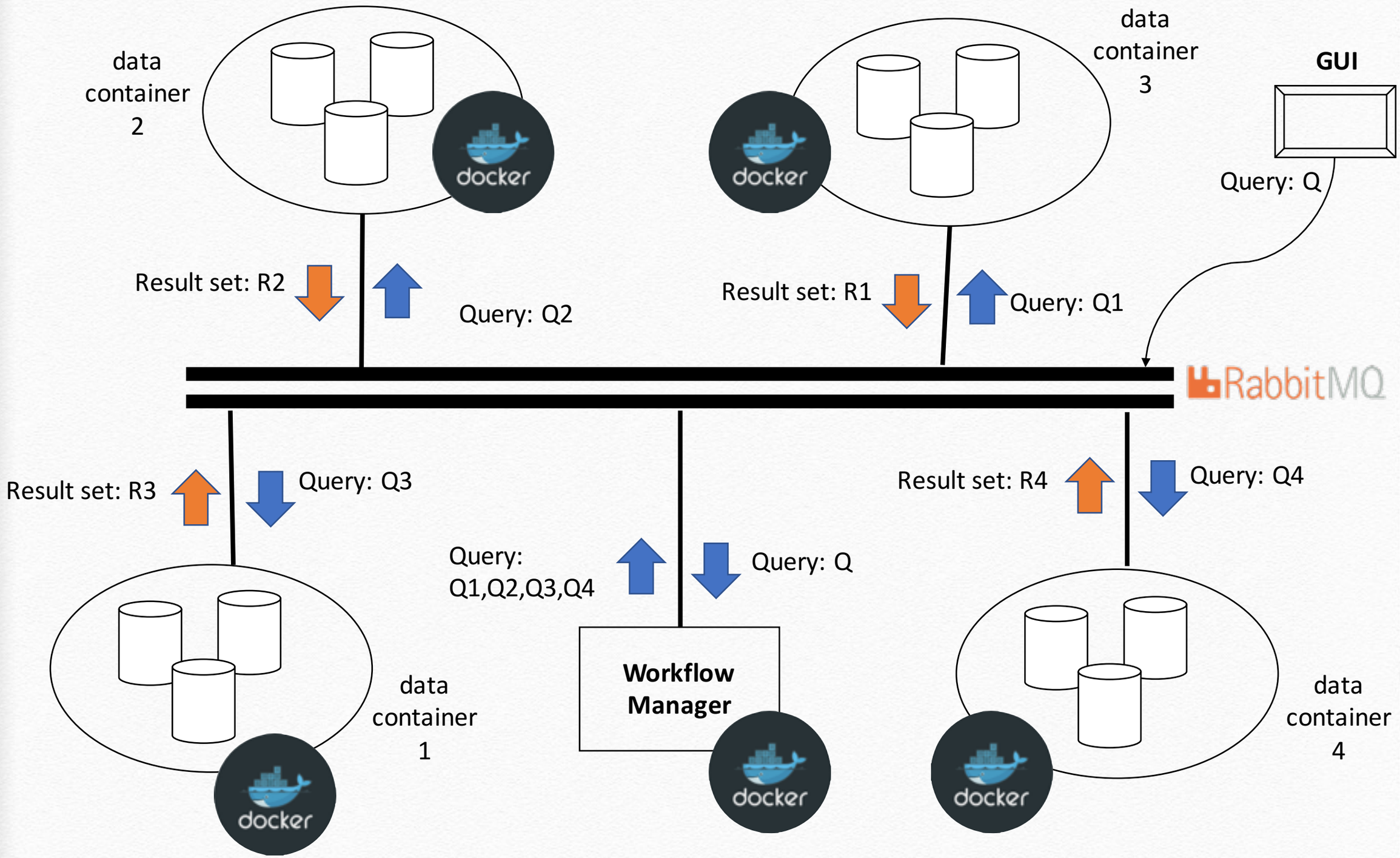
Q2

```
MATCH (node1)
WHERE node1.inventory_id = ?
RETURN node1.customer_id
```

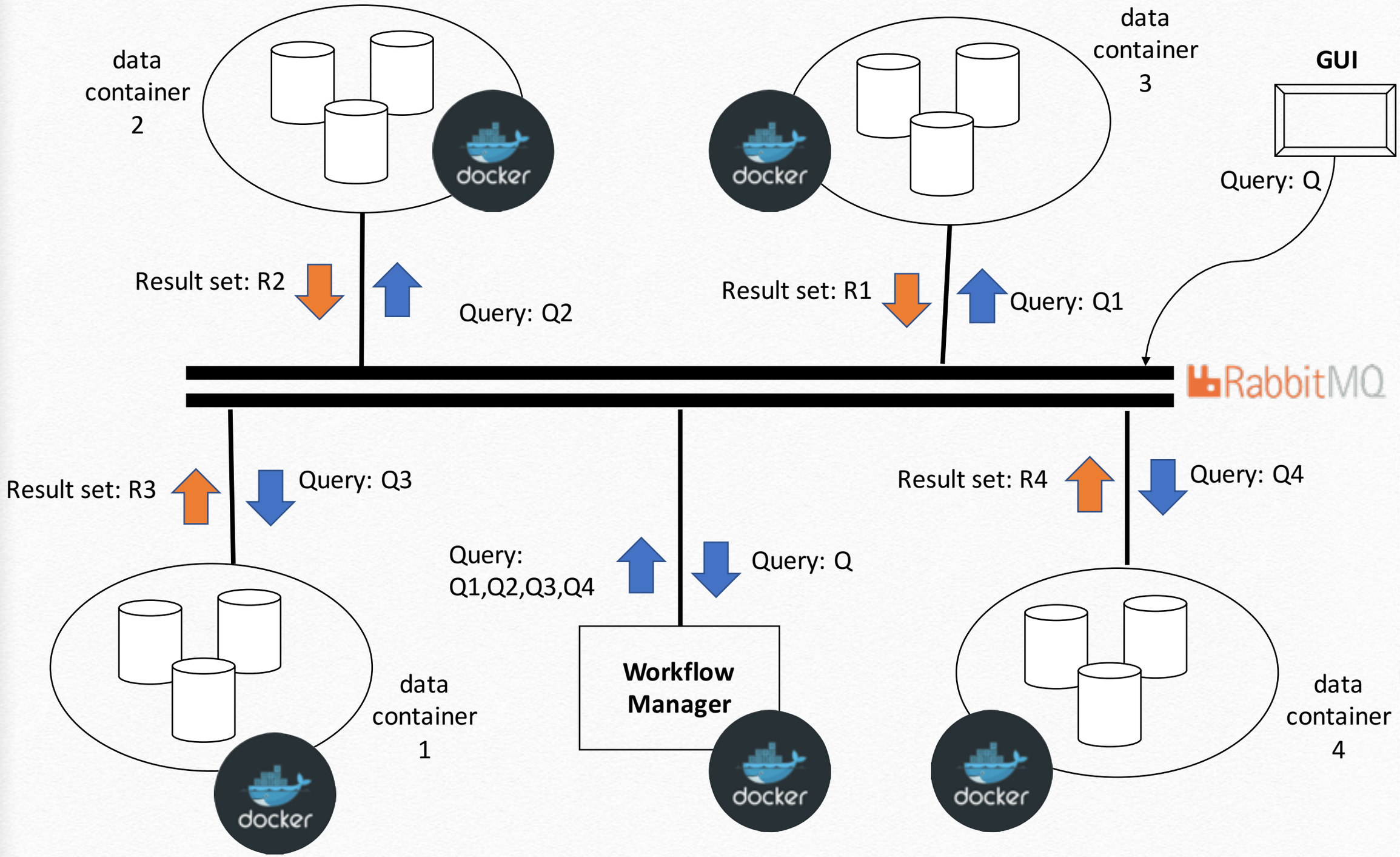
Q3

```
db.inventory.find(
        { film: { title: “Titanic”} },
        {inventory_id: 1}
        )
```

Micro-Services via Docker Containers

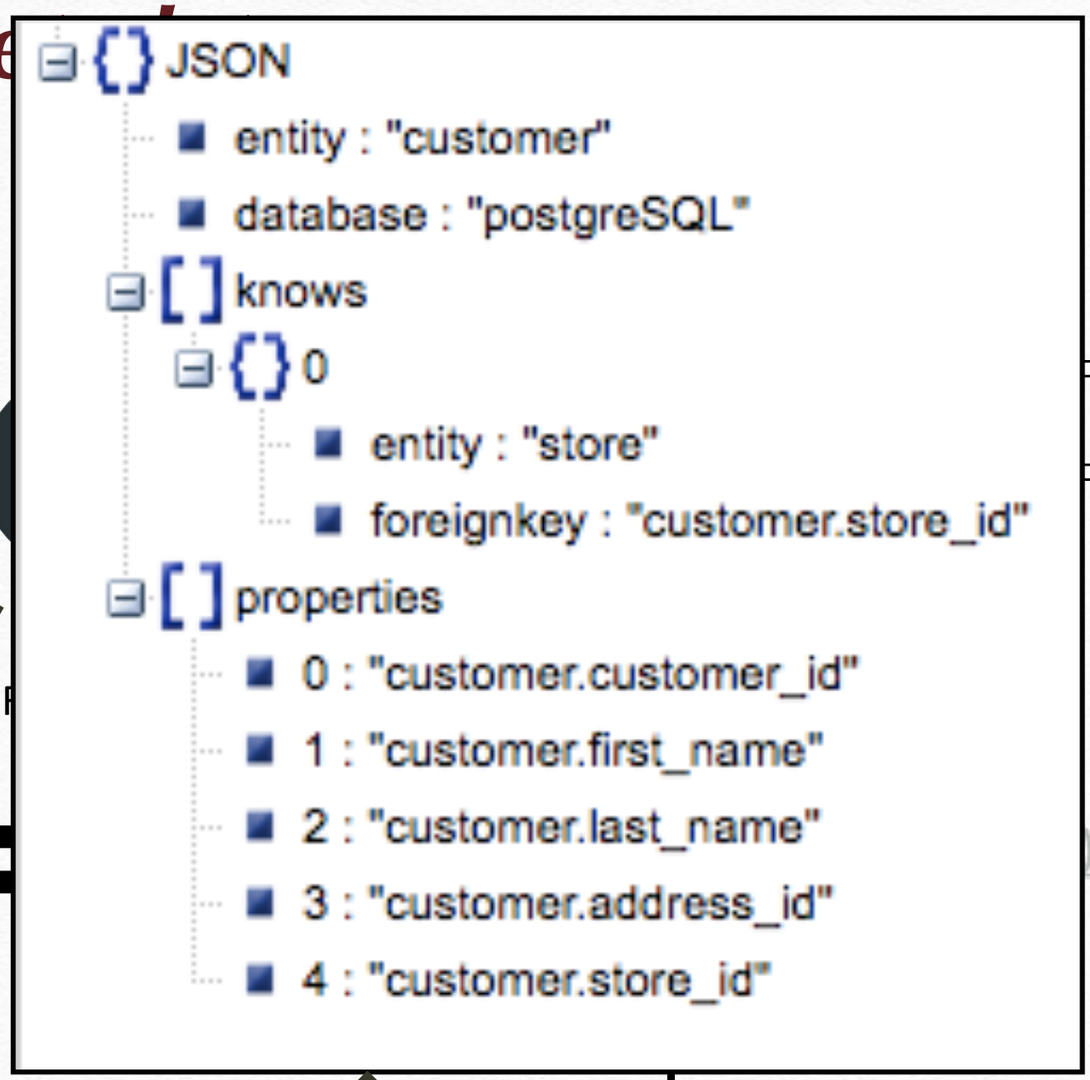


JSON Metadata



JSON Me

```
SELECT A.phone
FROM Customer C, Address A, Store R
WHERE C.customer_ID = R.customer_ID AND
      A.address_ID = R.address_ID AND
      C.first_name = “Roberto”
```



Result set: R3

Query: Q3

Query: Q1,Q2,Q3,Q4

Query: Q

Workflow Manager

data container 1

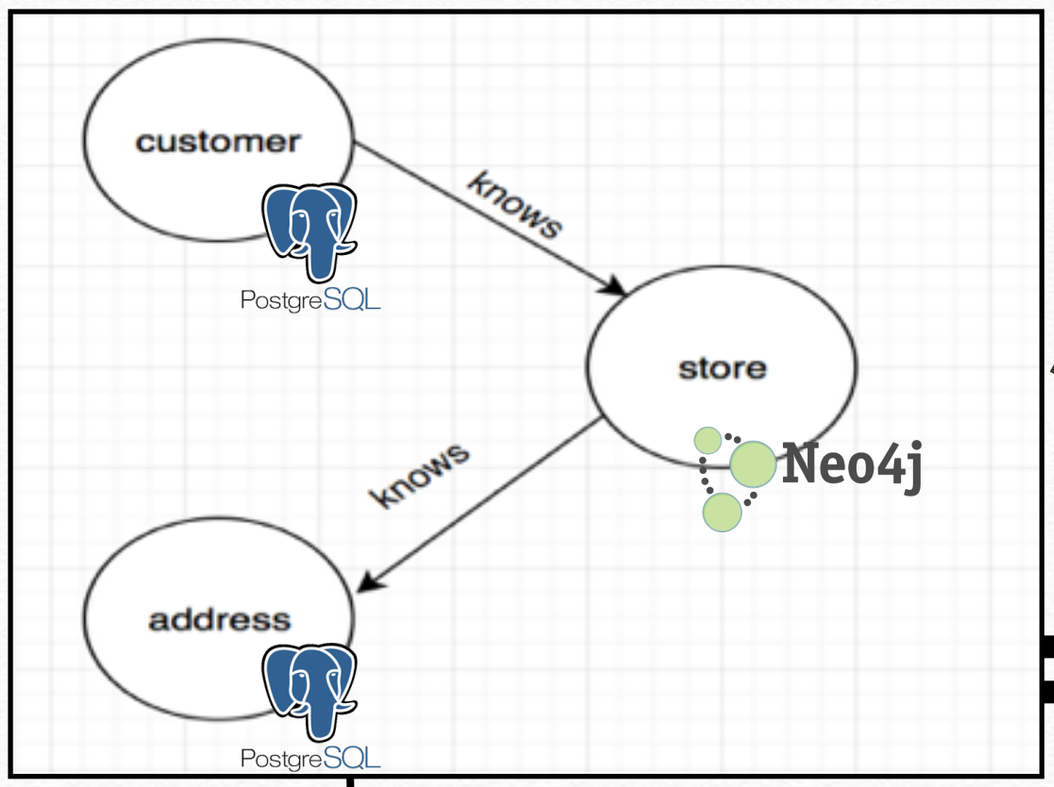
data container 4

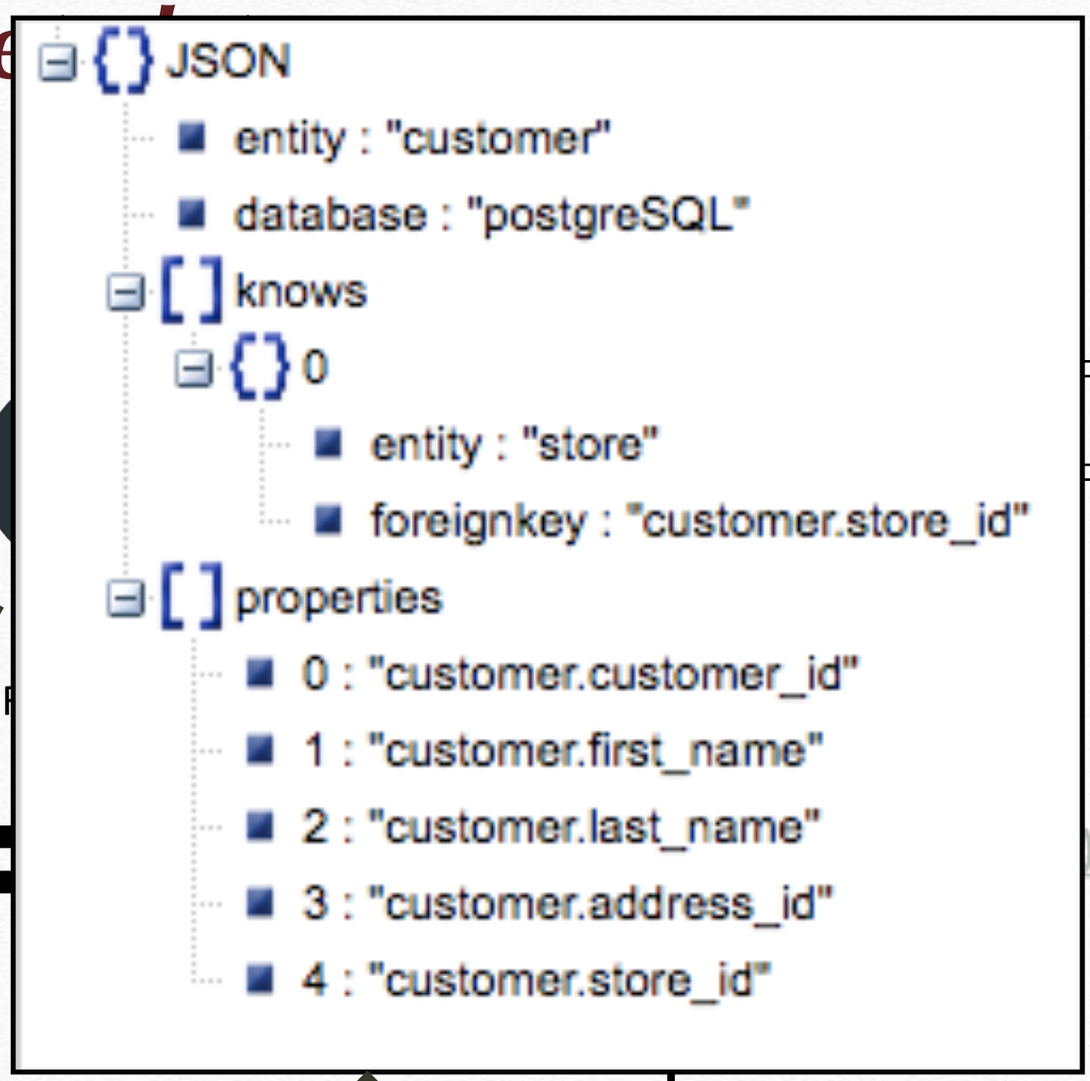
docker

docker

docker

JSON Me



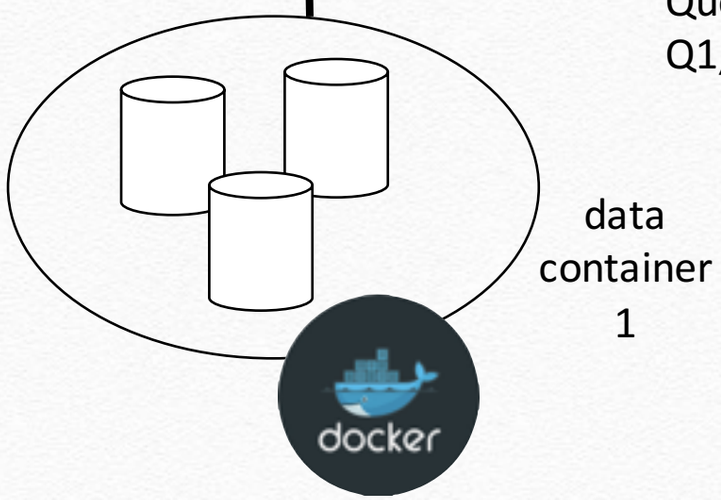


Result set: R3

Query: Q3

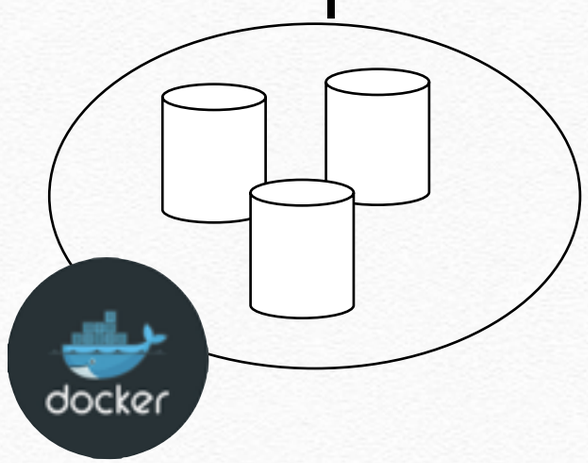
Query: Q1,Q2,Q3,Q4

Query: Q



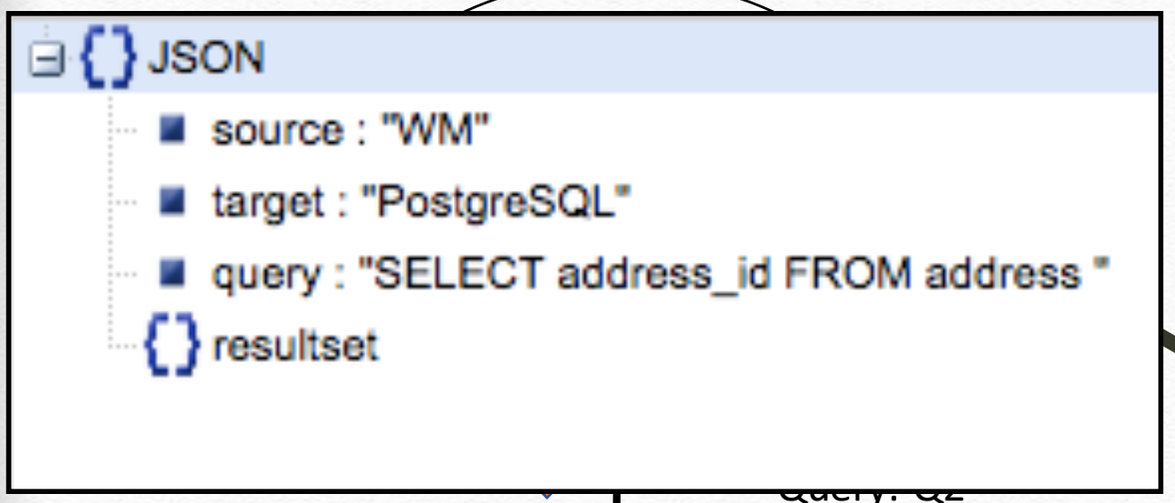
data container 1

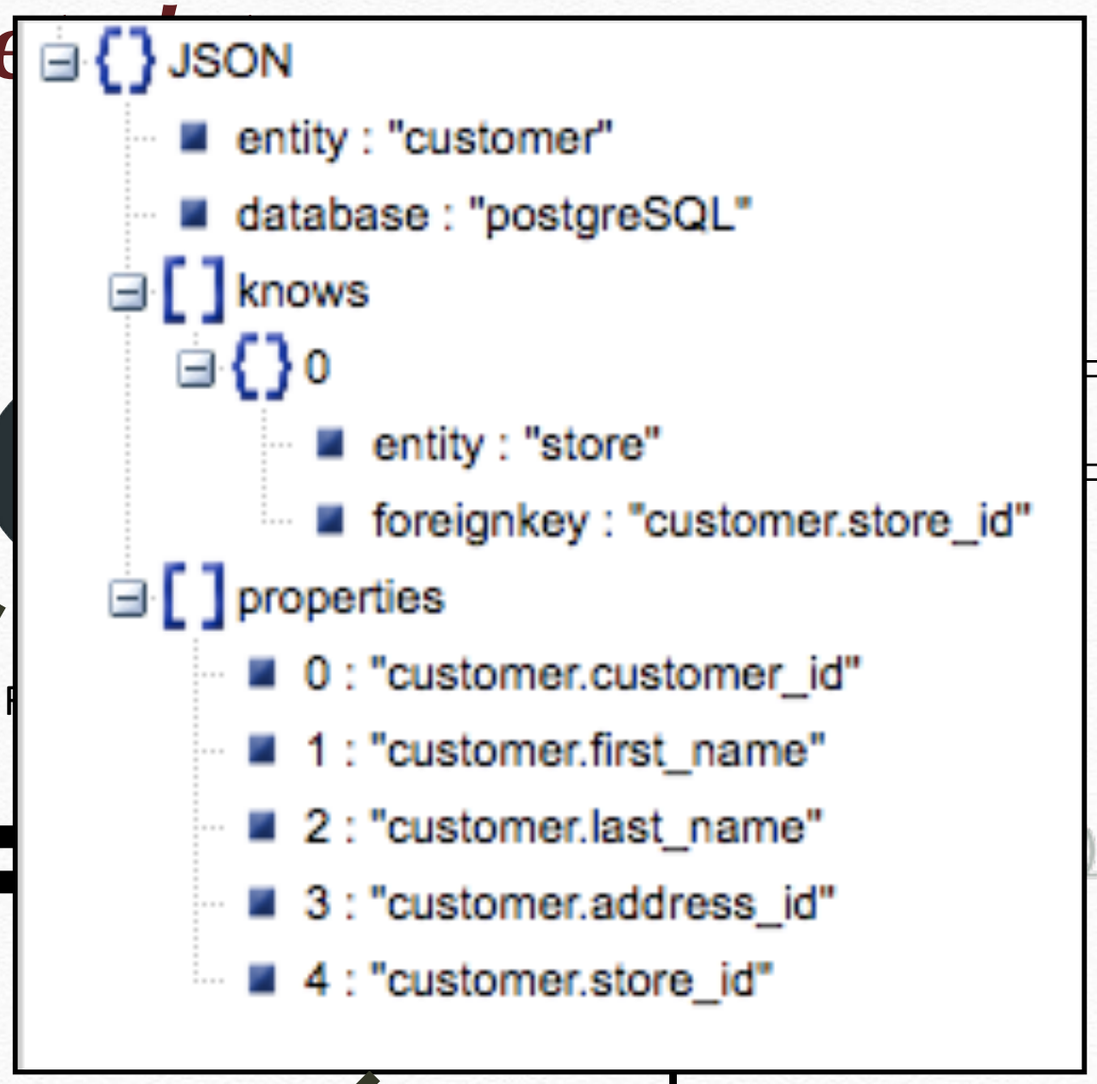
Workflow Manager



data container 4

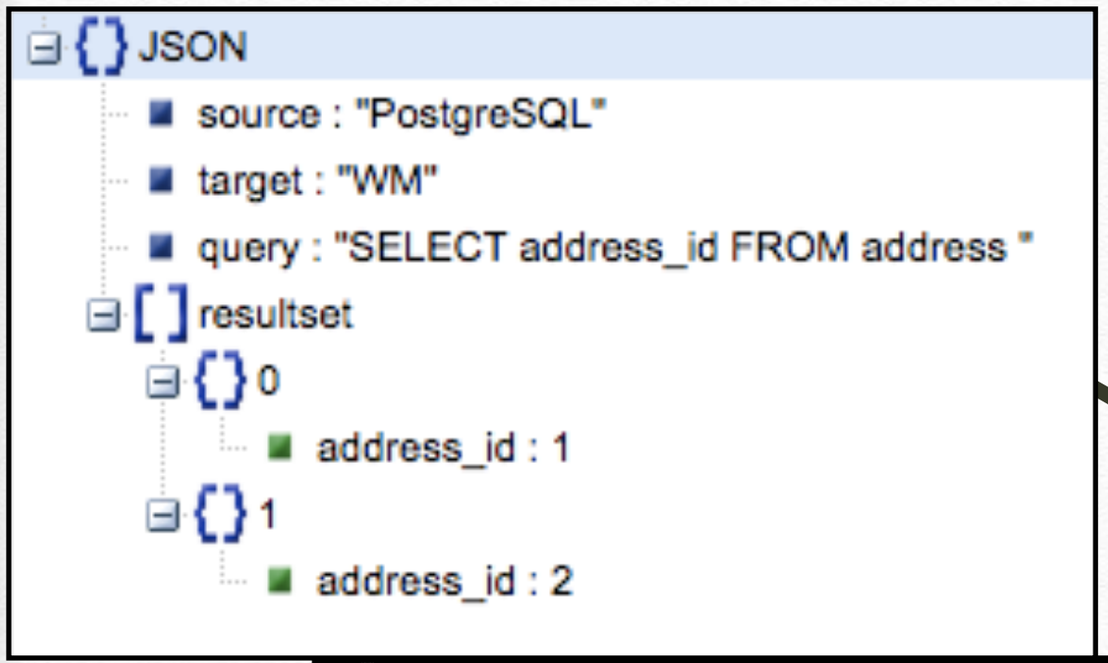
JSON Me

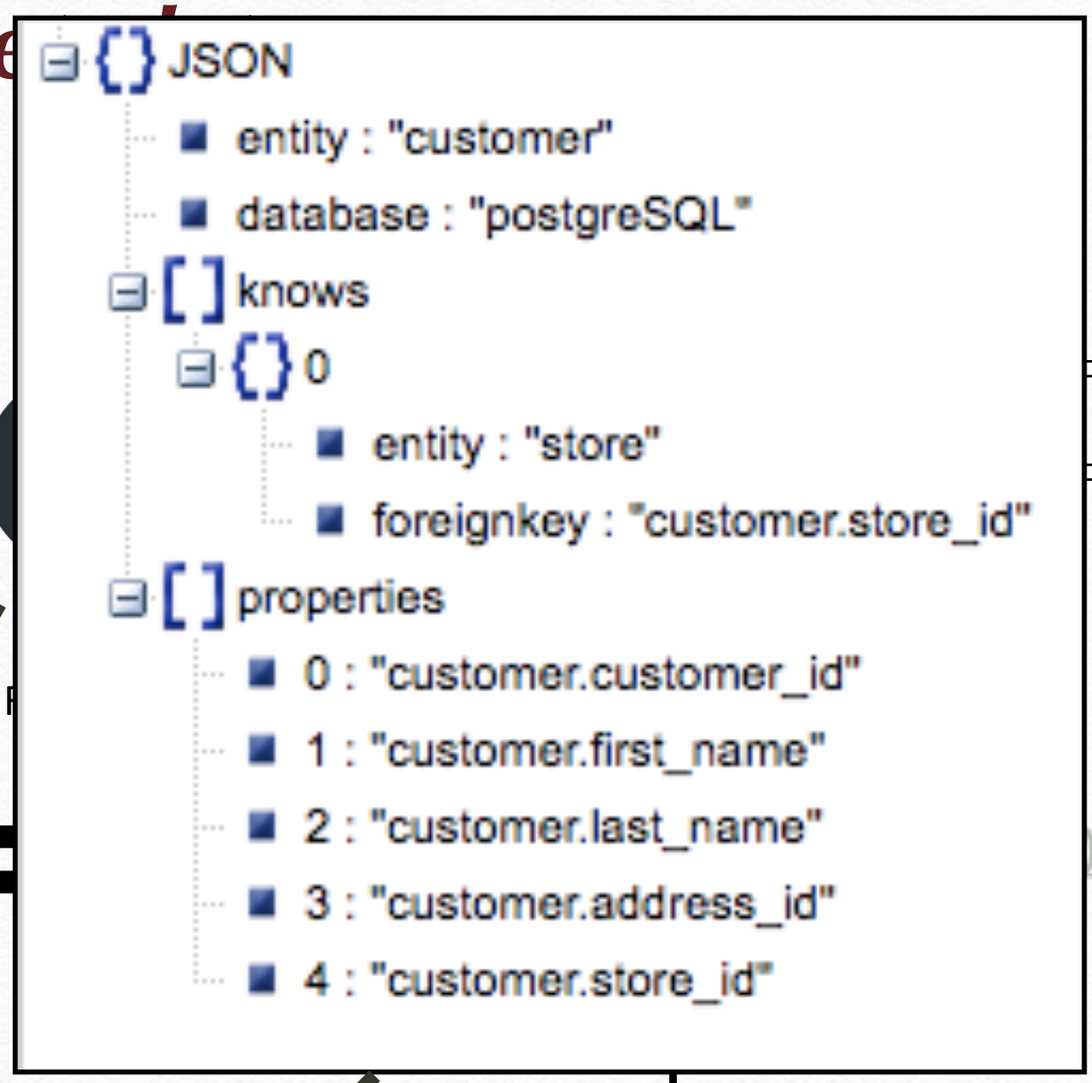

JSON
source : "WM"
target : "PostgreSQL"
query : "SELECT address_id FROM address "
resultset


JSON
entity : "customer"
database : "postgreSQL"
knows
0
entity : "store"
foreignkey : "customer.store_id"
properties
0 : "customer.customer_id"
1 : "customer.first_name"
2 : "customer.last_name"
3 : "customer.address_id"
4 : "customer.store_id"

PostgreSQL
Result set: R3
Query: Q3
Query:
Q1,Q2,Q3,Q4
Query: Q
data
container
1
Workflow
Manager
data
container
4
docker
docker
docker

JSON Me





PostgreSQL

Result set: R3

Query: Q3

Query: Q1,Q2,Q3,Q4

Query: Q

data container 1

Workflow Manager

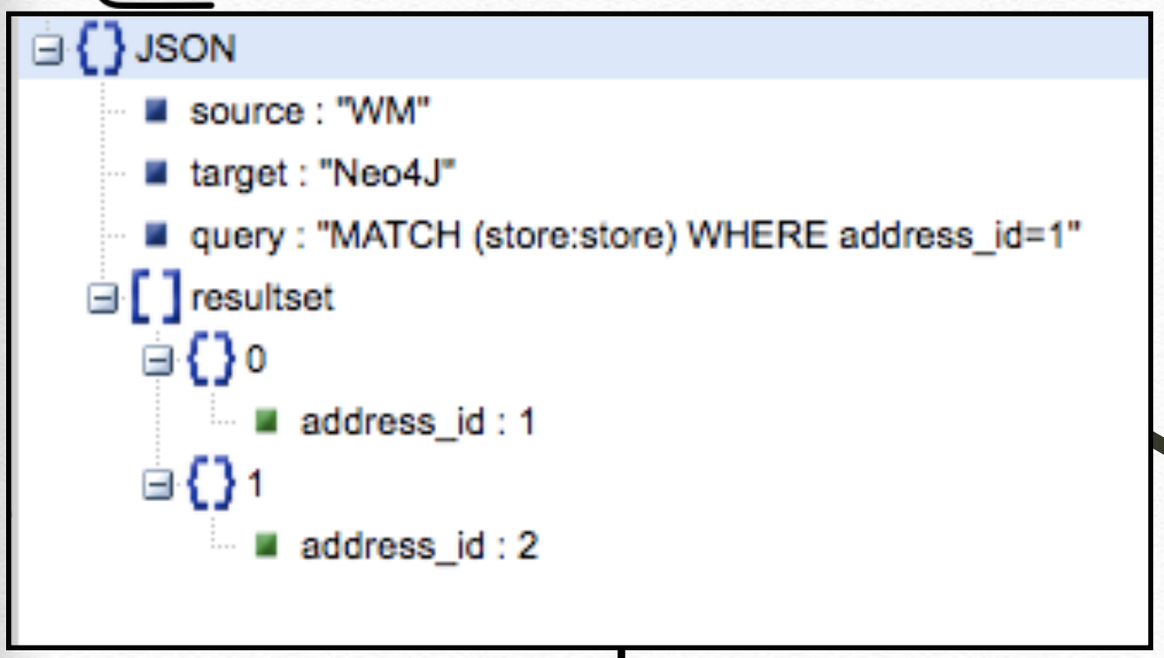
data container 4

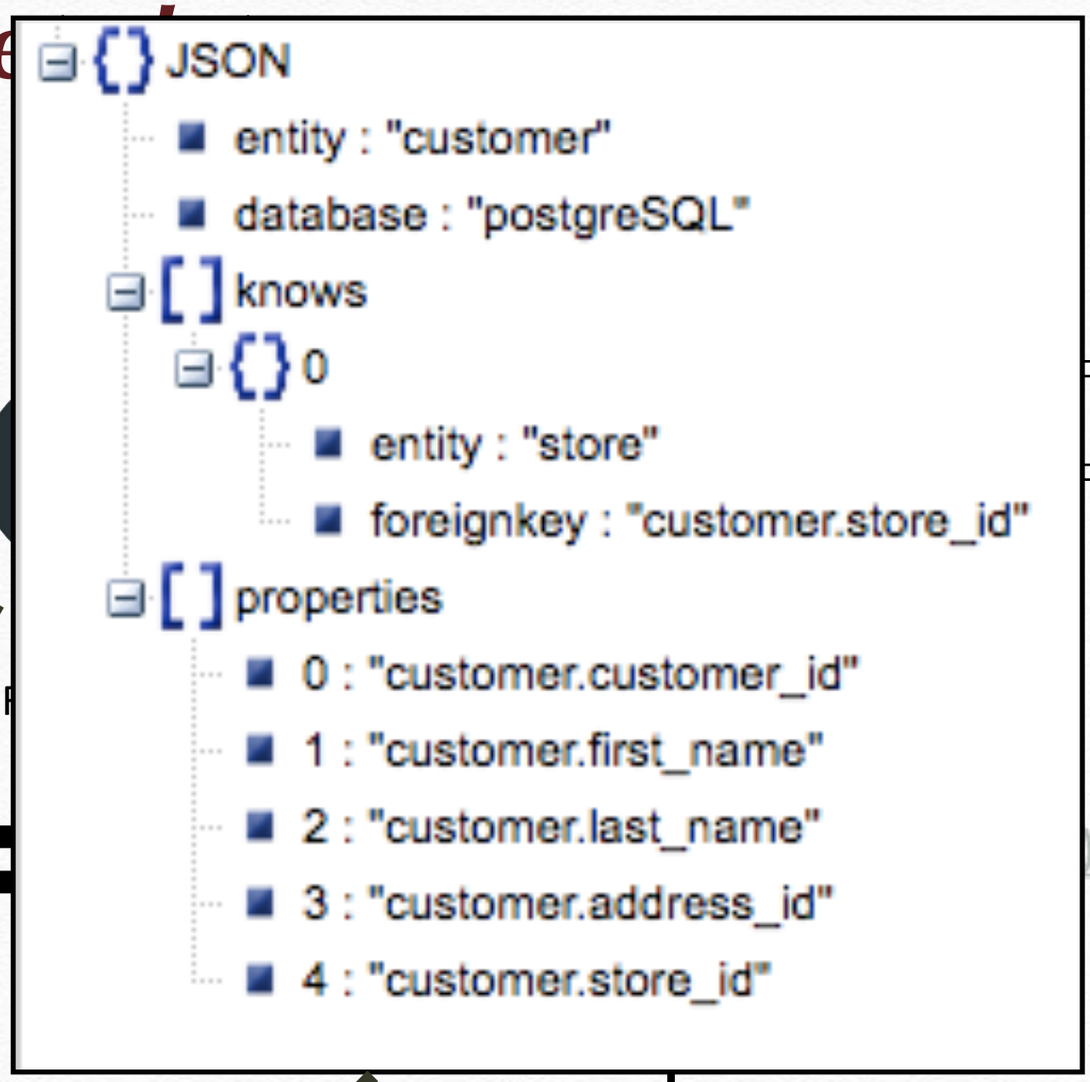
docker

docker

docker

JSON Me



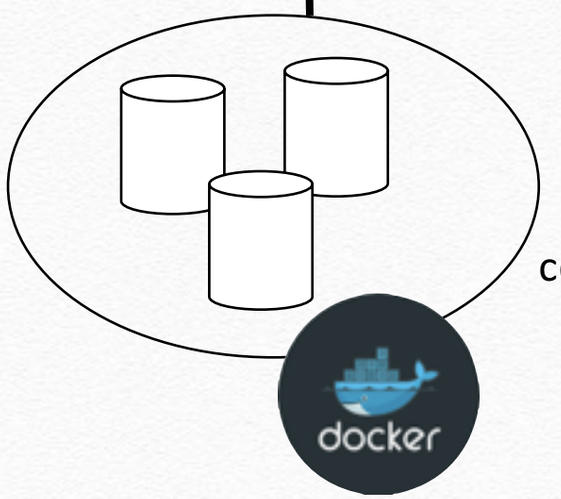


Result set: R3

Query: Q3

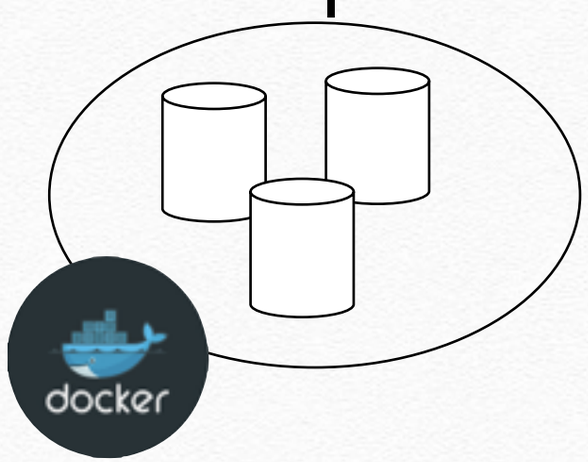
Query: Q1,Q2,Q3,Q4

Query: Q



data container 1

Workflow Manager



data container 4

Prototype: http://pamir.dia.uniroma3.it:8080/0Polystore1/

Proposals: #1

- **Persist**: given input sources *s1, s2, … sn*, link and persist them in the system

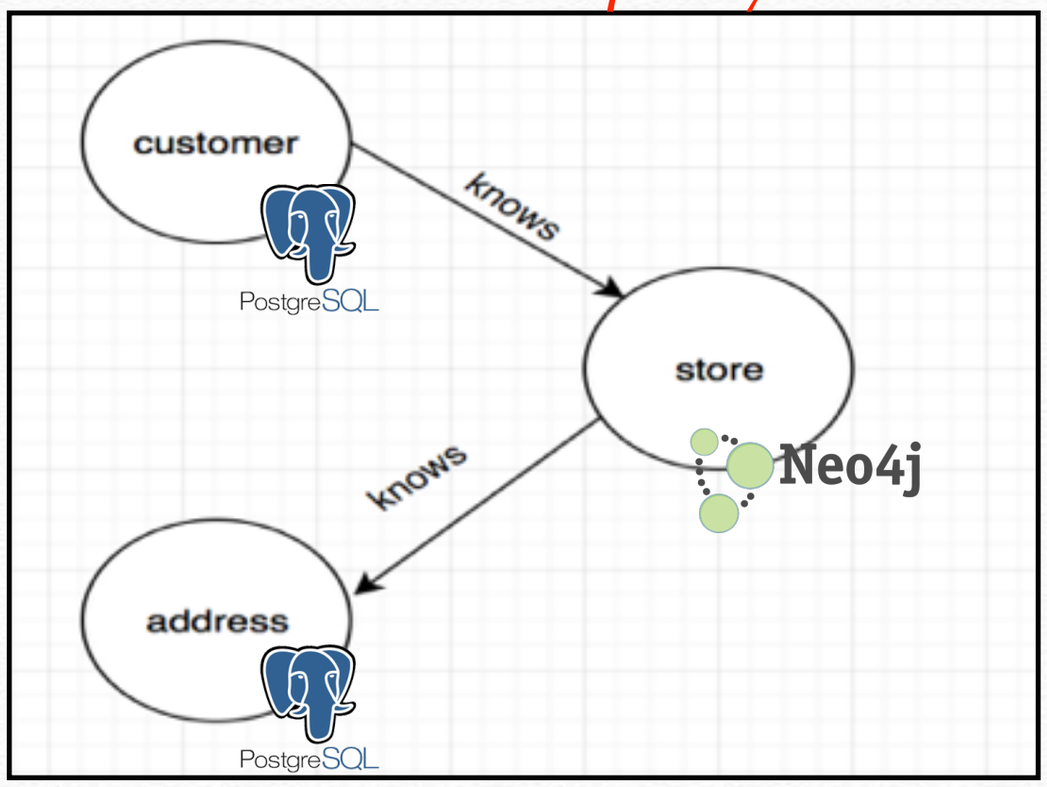
- *case study*: Films and reviews

  - ☑ iMDB -> film details (MongoDB)
  - ☑ Wikipedia -> exploration of film details, e.g. plot, cast and so on, (PostgreSQL)
  - ☑ Youtube -> film trailer (Redis and/or Oracle NoSql)
  - ☑ Movielens -> film ratings (Neo4J)
  - ☑ Amazon -> film commerce (Cassandra)

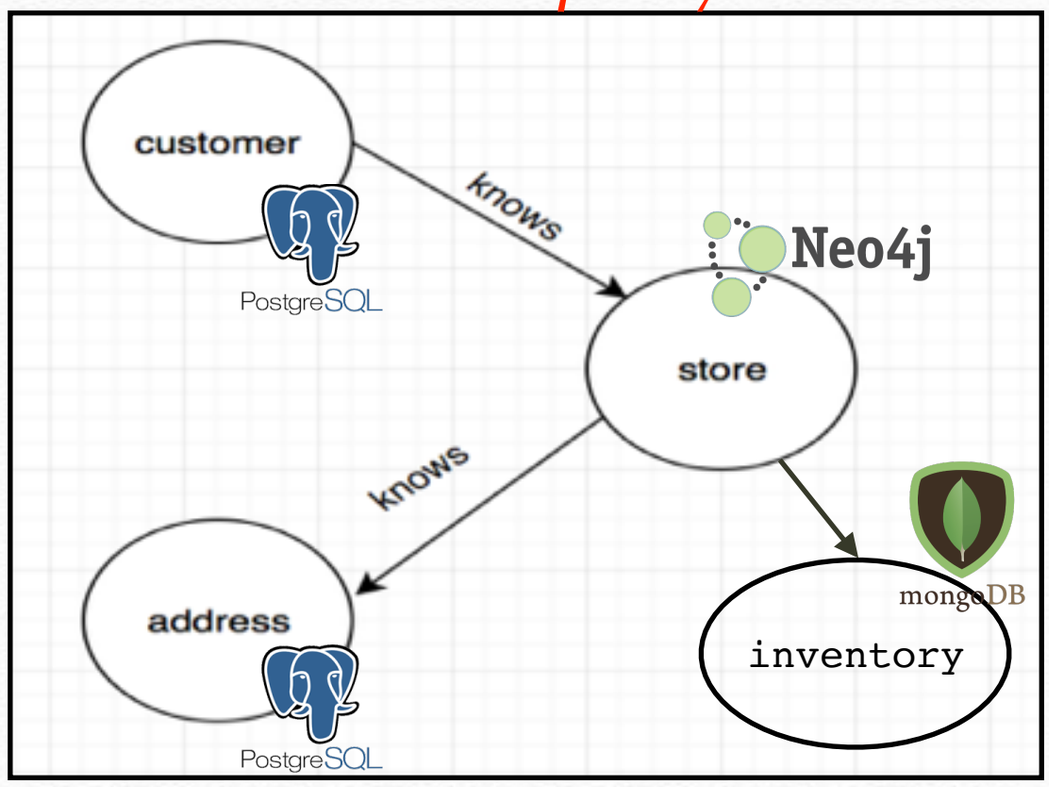
Proposals: #2

- **Flow**: execution plan improvement algorithm
- *case study*

chain query



star query



Proposals: #3

- **Query Analysis**: extended analysis of query language constructs used in the system and implementation in the query plan

- *case study*:

  - ☑ SQL -> OUTER JOINS, UNION, DIFFERENCE, NESTED QUERY

  - ☑ Cypher -> SUB-QUERY

  - ☑ Document Store -> NESTED COLLECTIONS

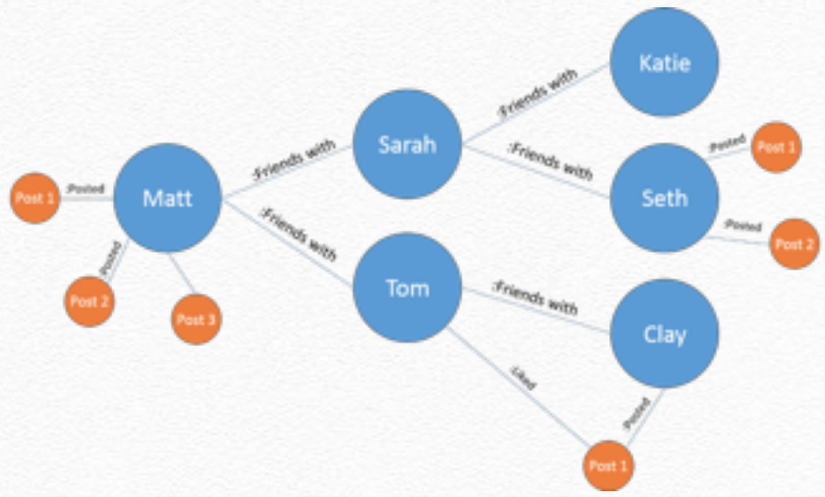
Proposals: #4

- **Distributed Visual Exploration (data and schema)**:
- exporting of the system in a distributed environment
- front-end for exploring and querying data containers (i.e the exploration is depending on the selected query language)

Results | Messages

	LocationID	Name	CostRate	Availability	ModifiedDate
1	1	Tool Crib	0.00	0.00	2002-06-01 00:00:00.000
2	2	Sheet Metal Racks	0.00	0.00	2002-06-01 00:00:00.000
3	3	Paint Shop	0.00	0.00	2002-06-01 00:00:00.000
4	4	Paint Storage	0.00	0.00	2002-06-01 00:00:00.000
5	5	Metal Storage	0.00	0.00	2002-06-01 00:00:00.000
6	6	Miscellaneous Storage	0.00	0.00	2002-06-01 00:00:00.000
7	7	Finished Goods Storage	0.00	0.00	2002-06-01 00:00:00.000
8	10	Frame Forming	22.50	96.00	2002-06-01 00:00:00.000
9	20	Frame Welding	25.00	108.00	2002-06-01 00:00:00.000
10	30	Debur and Polish	14.50	120.00	2002-06-01 00:00:00.000
11	40	Paint	15.75	120.00	2002-06-01 00:00:00.000
12	45	Specialized Paint	18.00	80.00	2002-06-01 00:00:00.000
13	50	Subassembly	12.25	120.00	2002-06-01 00:00:00.000
14	60	Final Assembly	12.25	120.00	2002-06-01 00:00:00.000

```
{id: 1,
 name: Joe,
 url: '...',
 stream:
 [{
   user: 2,
   title: 'today',
   body: 'go fly a kite',
   likes: [
    {user: 3},
    {user: 1},
   ],
 }]
}
```



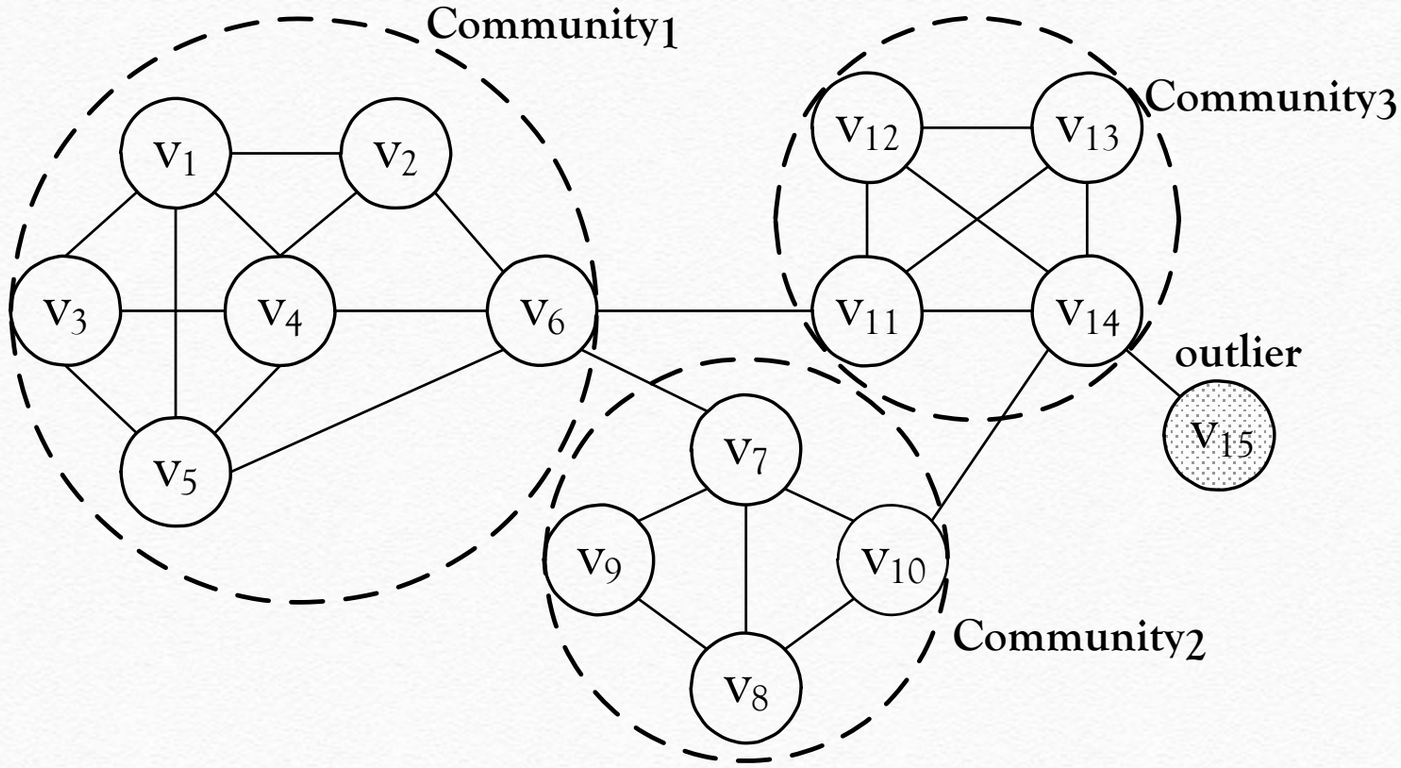
Community Profiling in Social Networks

17/05/2017 - Big Data 2017

Community in social media

- ***Community***: It is formed by individuals such that those within a group interact with each other more frequently than with those outside the group

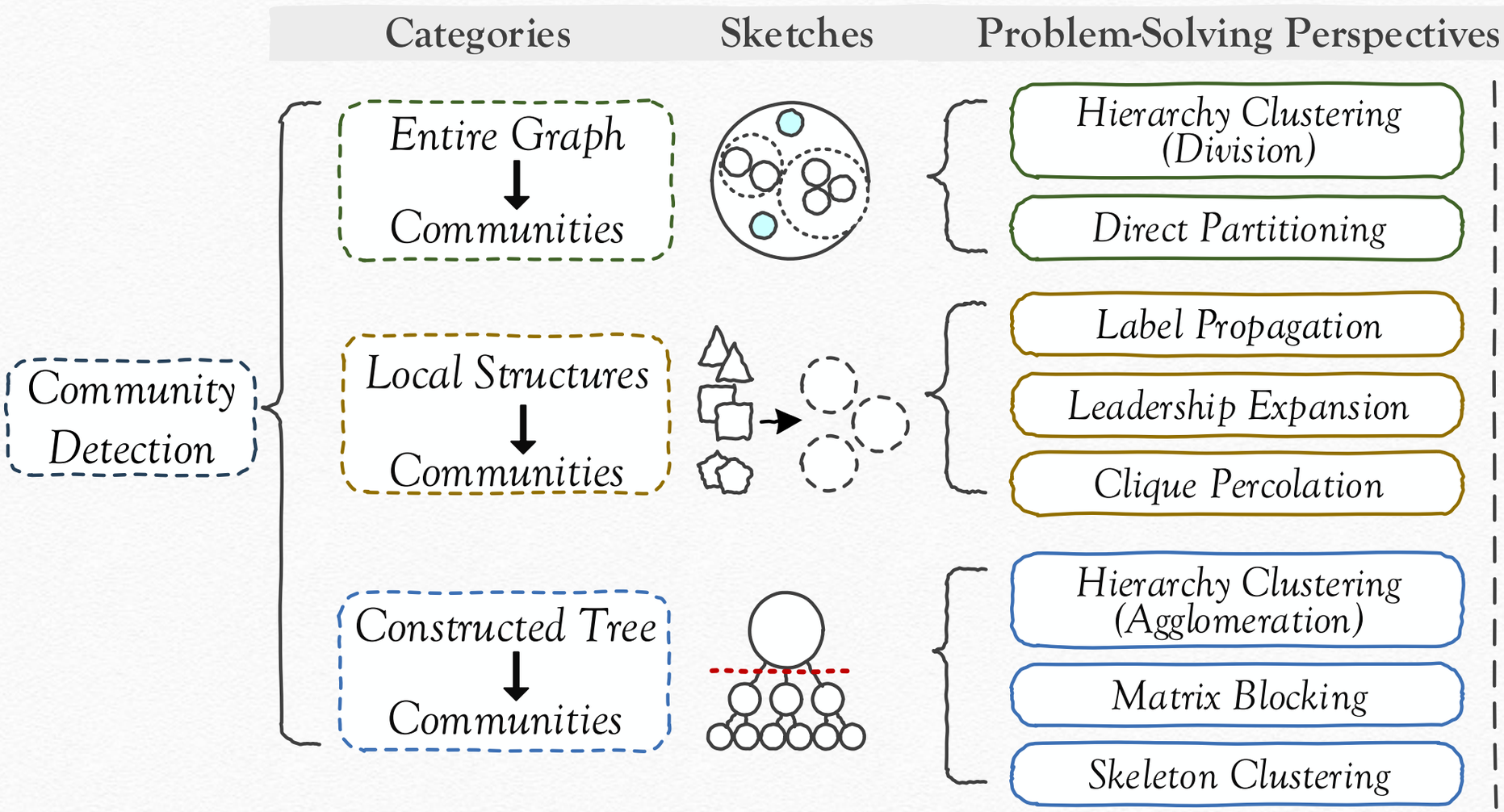
- ***Community detection***: discovering groups in a network where individuals' group memberships are not explicitly given



Community in social media

- ***Community***: It is formed by individuals such that those within a group interact with each other more frequently than with those outside the group

- ***Community detection***: discovering groups in a network where individuals' group memberships are not explicitly given



Community Profiling

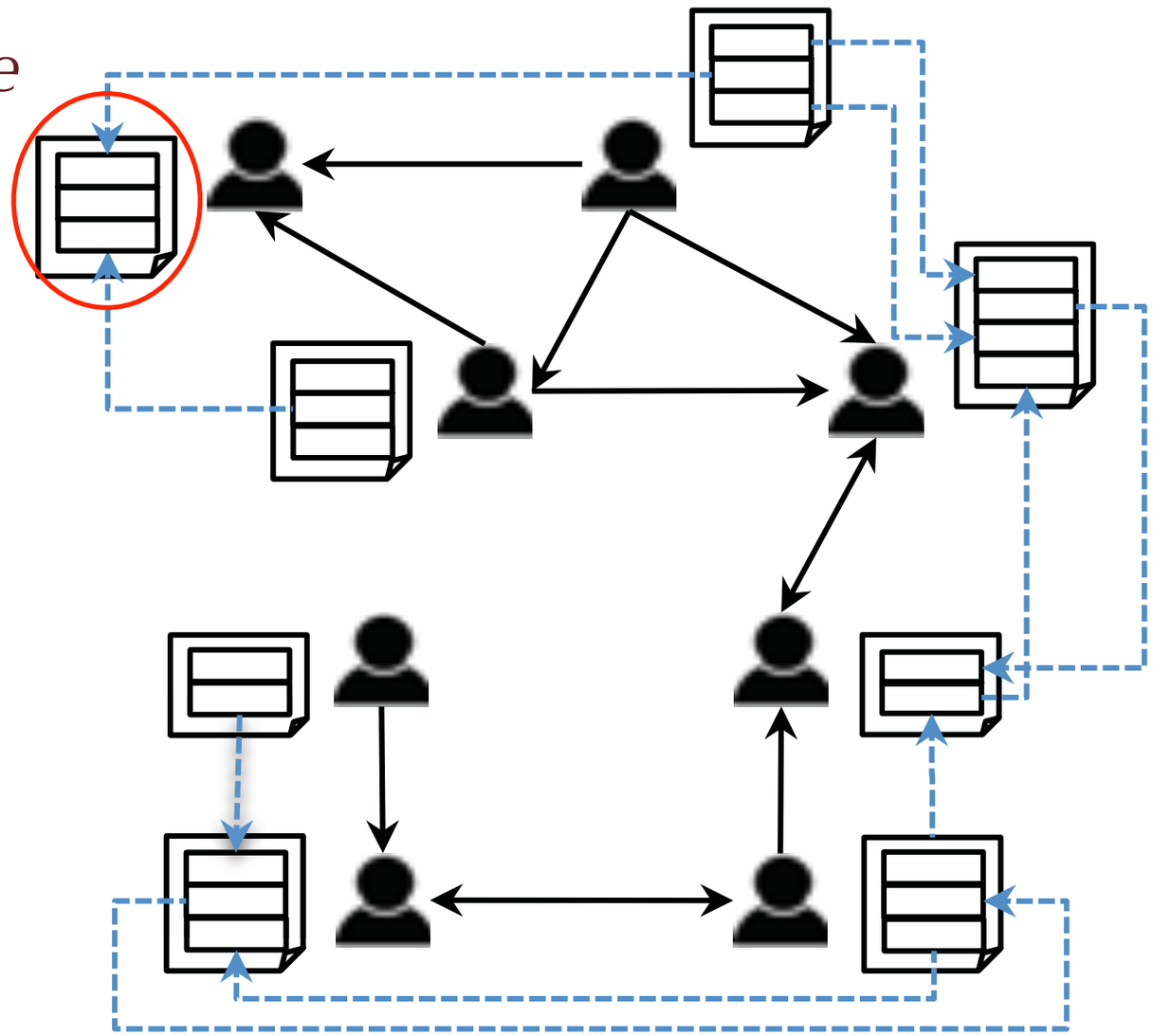
- ***Positive Aspect***: community membership assists us to better understand the network *structure*.

- ***Negative Aspect***: membership alone, without knowing ***what a community is*** and ***how it interacts with others***, has only limited applications

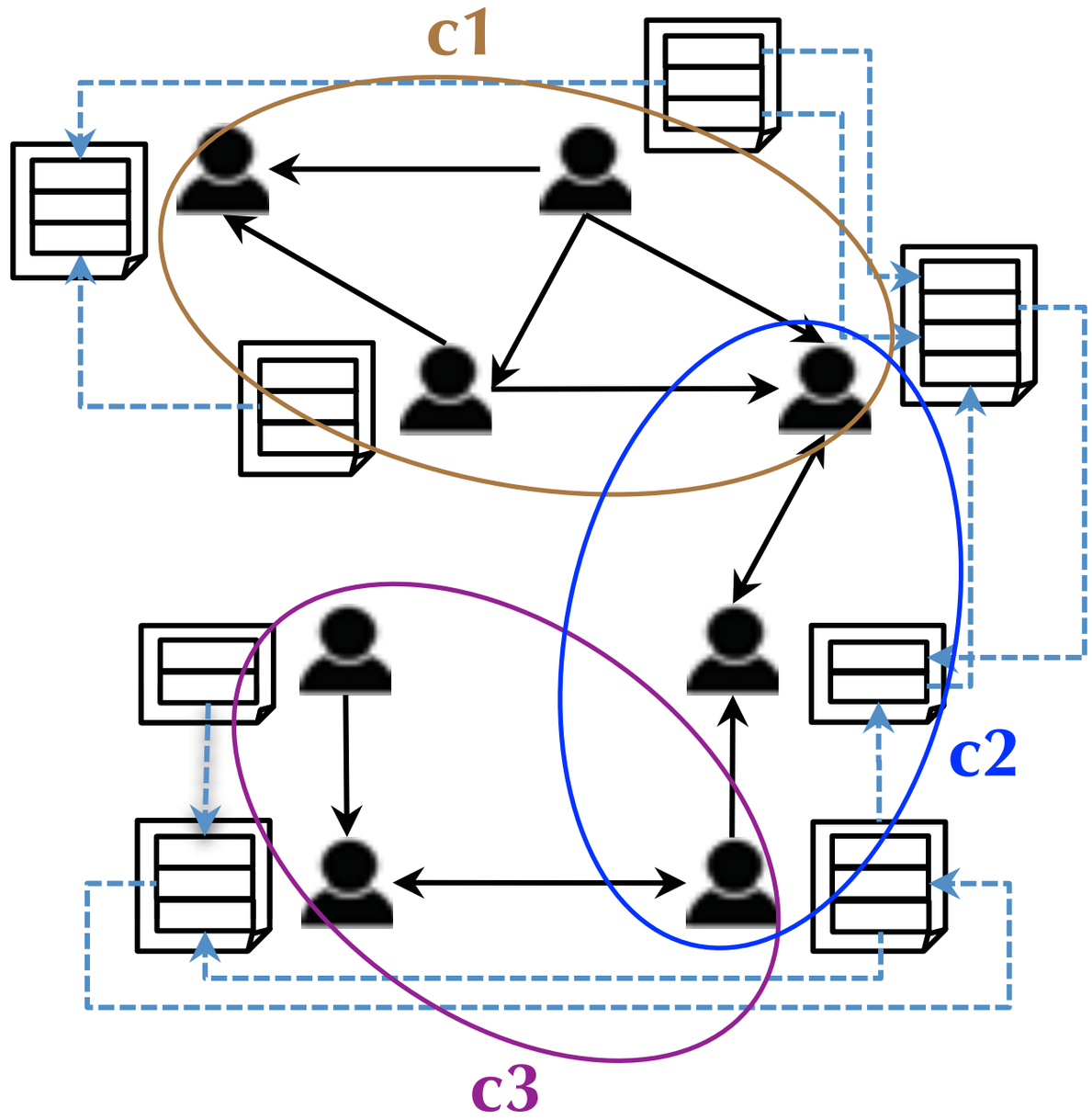
- ***Community profiling***: to characterize the **intrinsic** nature and **extrinsic** behavior of a community – thereby enabling useful community-level applications.

Community Profiling

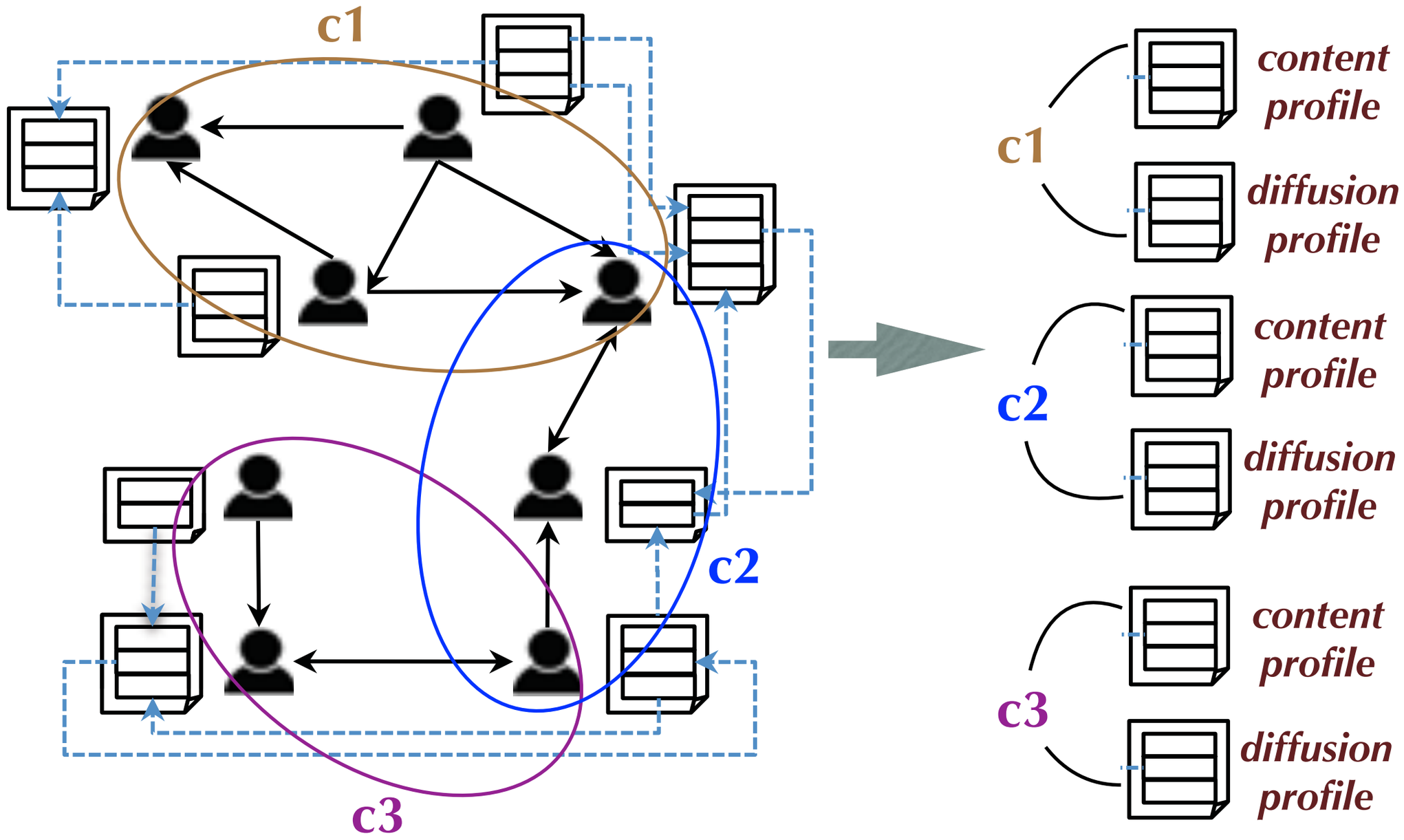
user profile

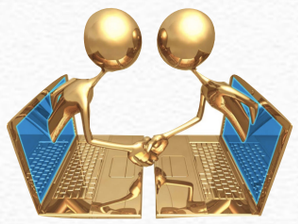


Community Profiling

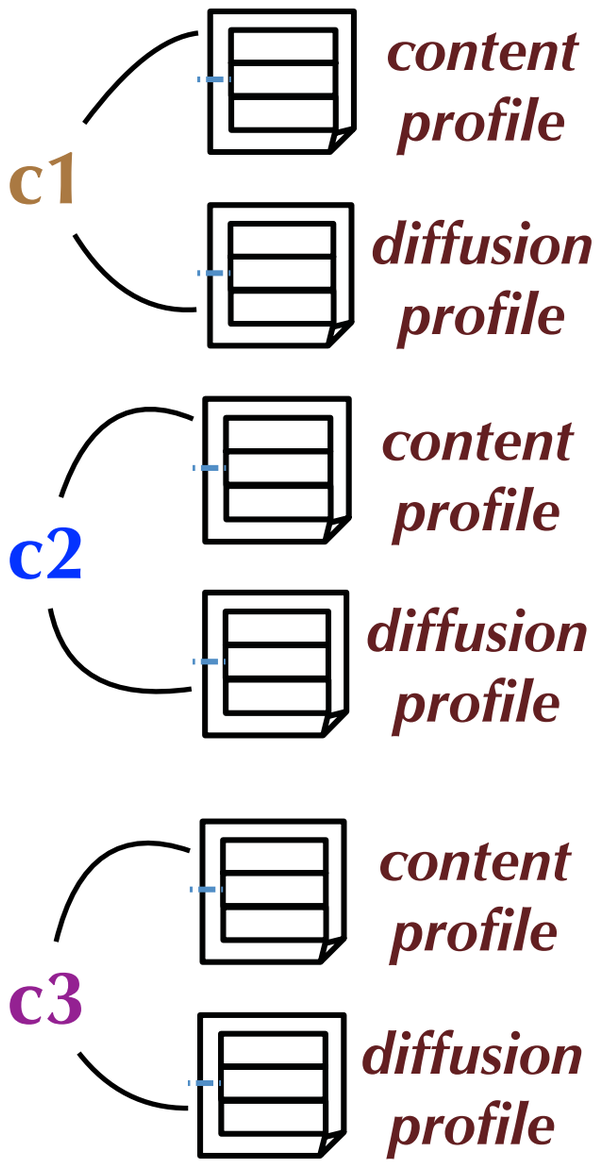


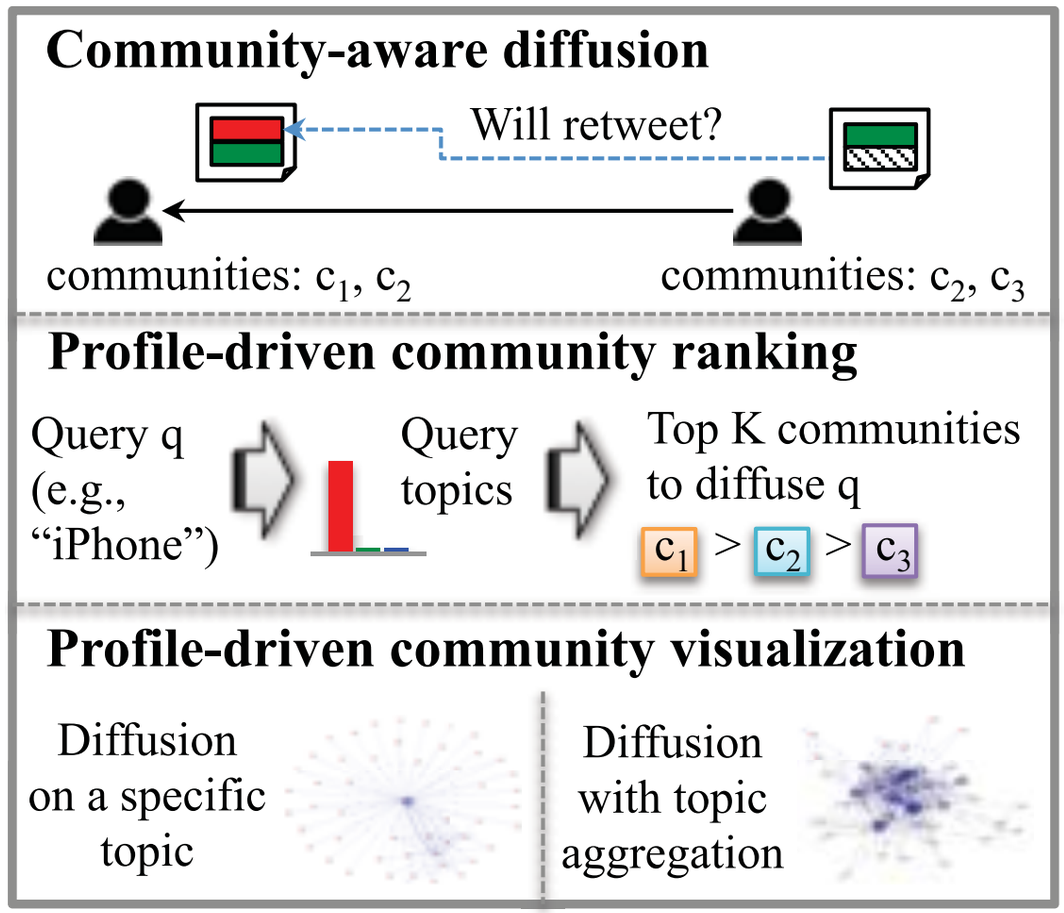
Community Profiling



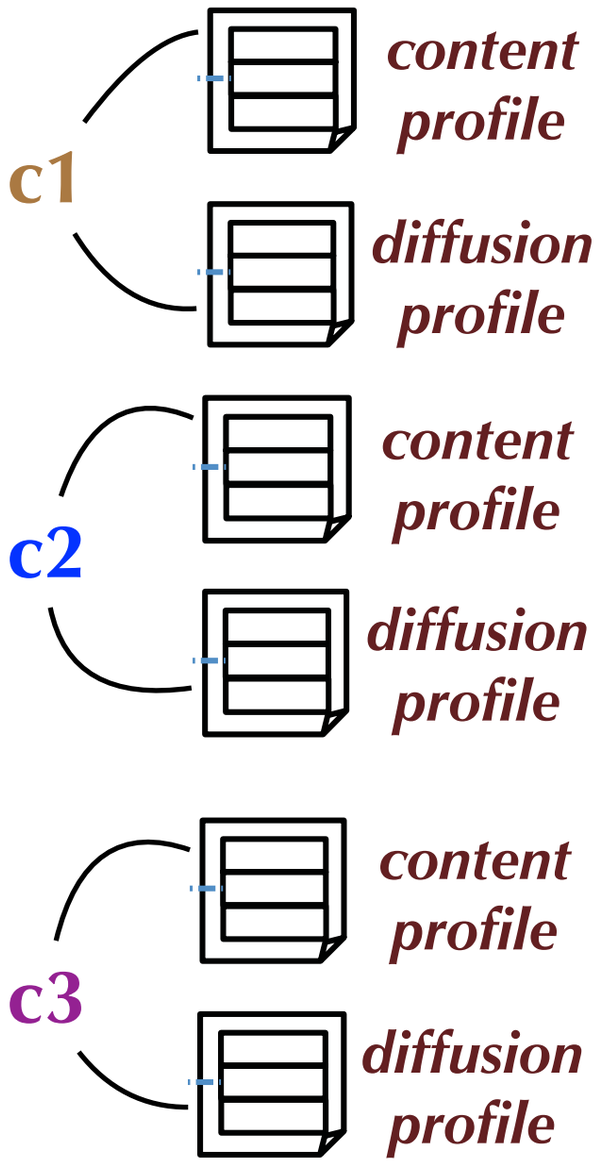


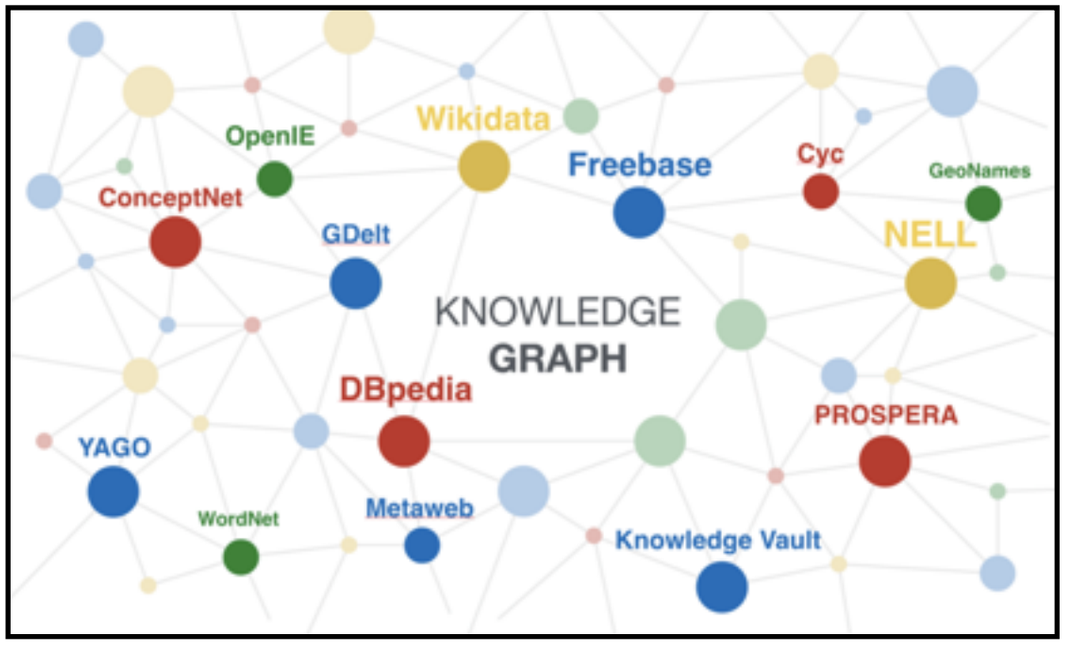
Community Profiling



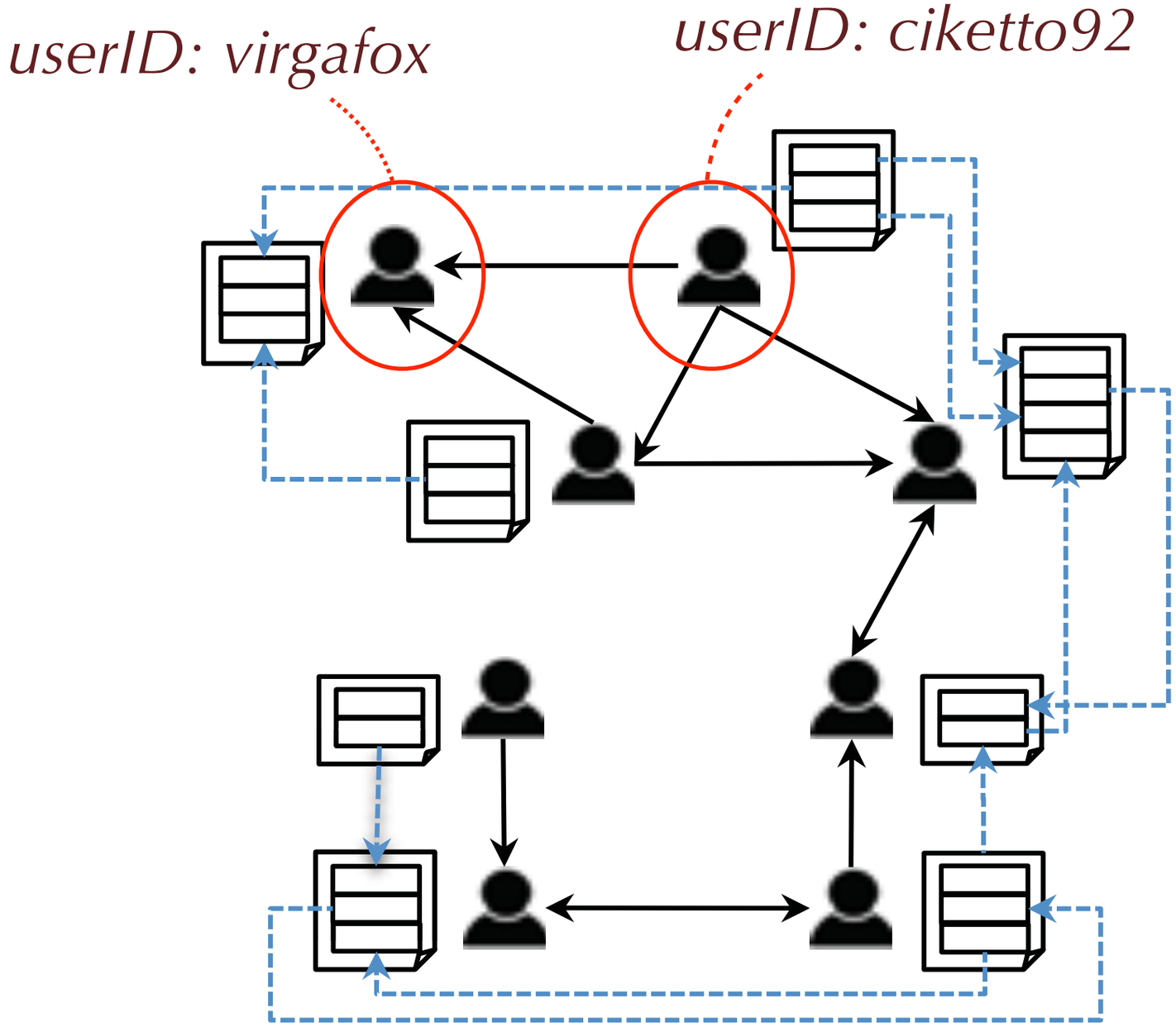


Community Profile Modeling





Case study: TWITTER (15.000 users)

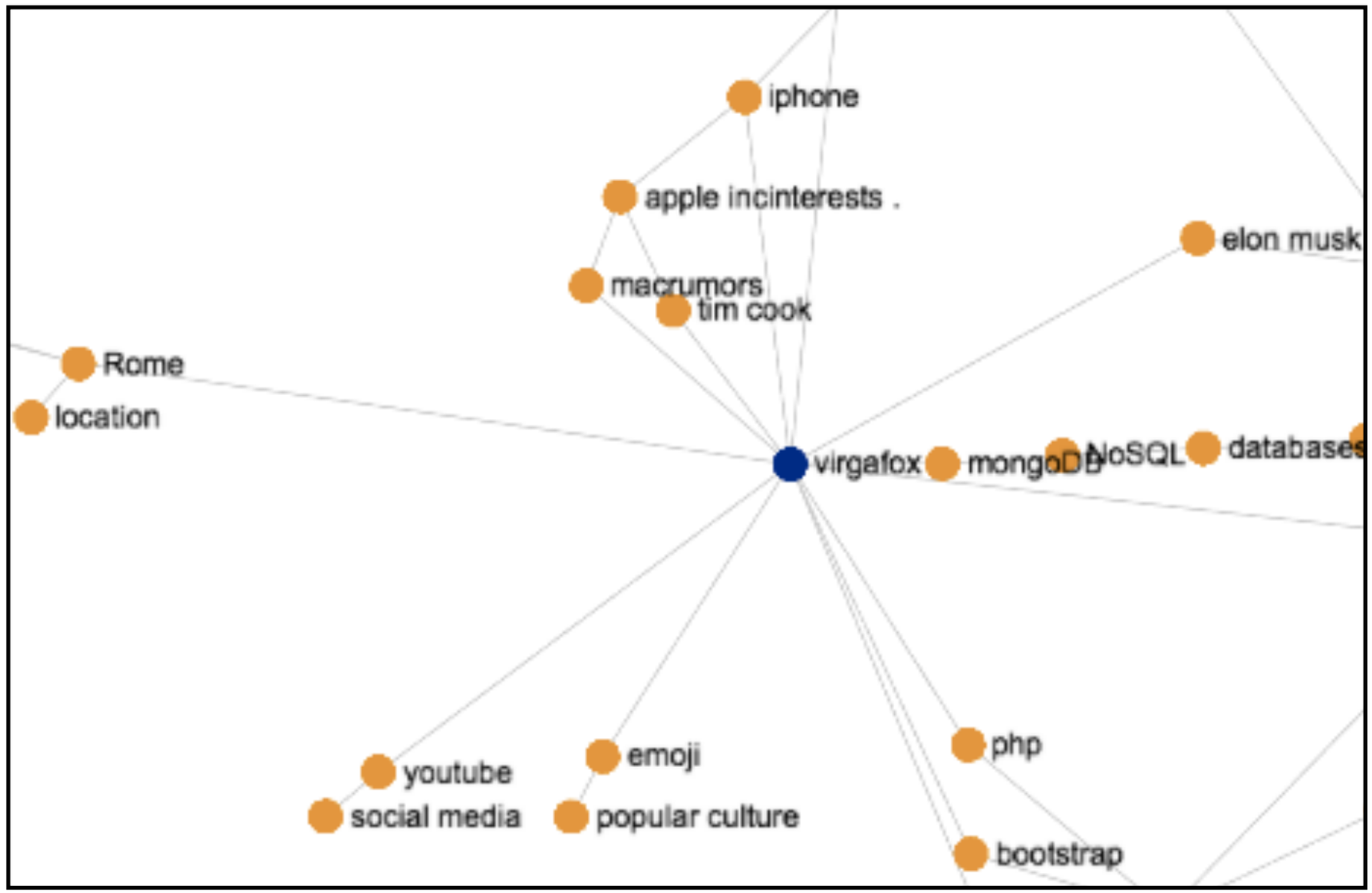


Case study: TWITTER (15.000 users)

userID: virgafox

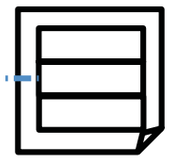
macromeasures

Cayley



Case study: TWITTER (15.000 users)

userID: virgafox



macromeasures

JSON
interests
all
54493b73d4ac1418c2030519
category : "show"
display : "Silicon Valley (show)"
level : "Medium"
name : "Silicon Valley"
parents
0 : "55afc38392cffb786d83f7a6"
1 : "55afc38592cffb786d83f82e"
score : 2
5457c85ad4ac147c59c0ca4f
category : "default"
display : "Business news"
level : "Low"
name : "Business news"
parents
0 : "55afc38392cffb786d83f797"
1 : "55afc38492cffb786d83f7d0"
score : 1

Proposals: #1

- **Community Profile Algebra**: define and implement operators for profile management

- *case study*:

  - ☑ profile intersection
  - ☑ profile union
  - ☑ profile alignment
  - ☑ profile difference

Proposals: #2

- **Profile Modeling**: represent a user profile in terms of a knowledge graph referring to a different ontology

- *case study*:

  - ☑ ontology: Schema.org

  - ☑ social network: Twitter (15000 userIDs)

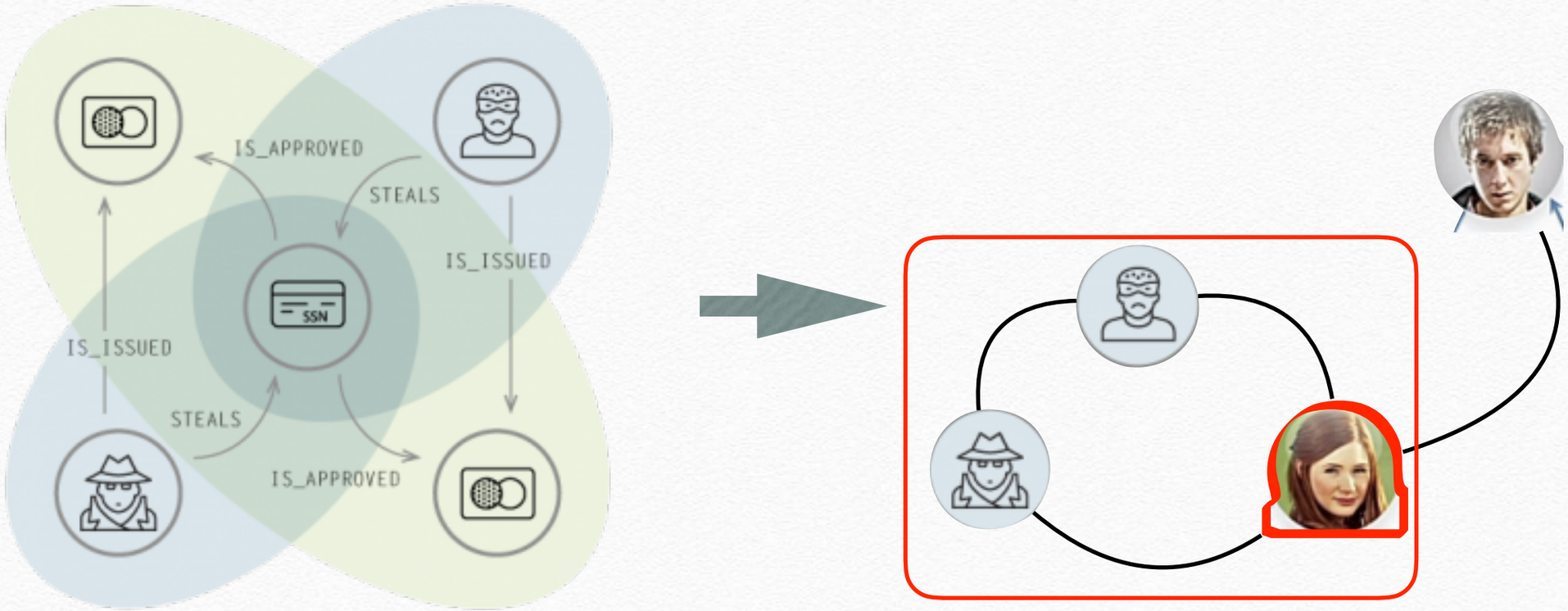
Fraud Detection

17/05/2017 - Big Data 2017

Fraud Detection via Community Detection

- ***Detecting Fraud*** : Fraud is an adaptive crime, so it needs special methods of intelligent data analysis to detect and prevent it. These methods exist in the areas of Knowledge Discovery in Databases (KDD), Data Mining, Machine Learning and Statistics

- ***Community detection***: discovering groups in a network having relationships with defrauders

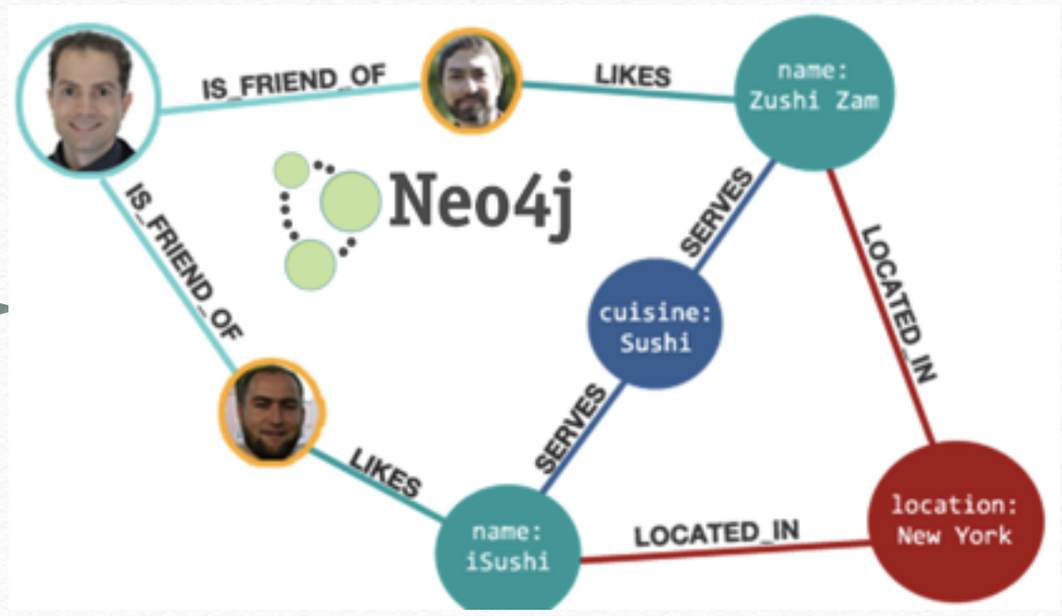


Fraud Detection via Community Detection

- ***CASE STUDY***: Orders coming from an important phone company and requirements about fraud warnings

- ***From Tables to Graph***: orders are structured in a relation DB;

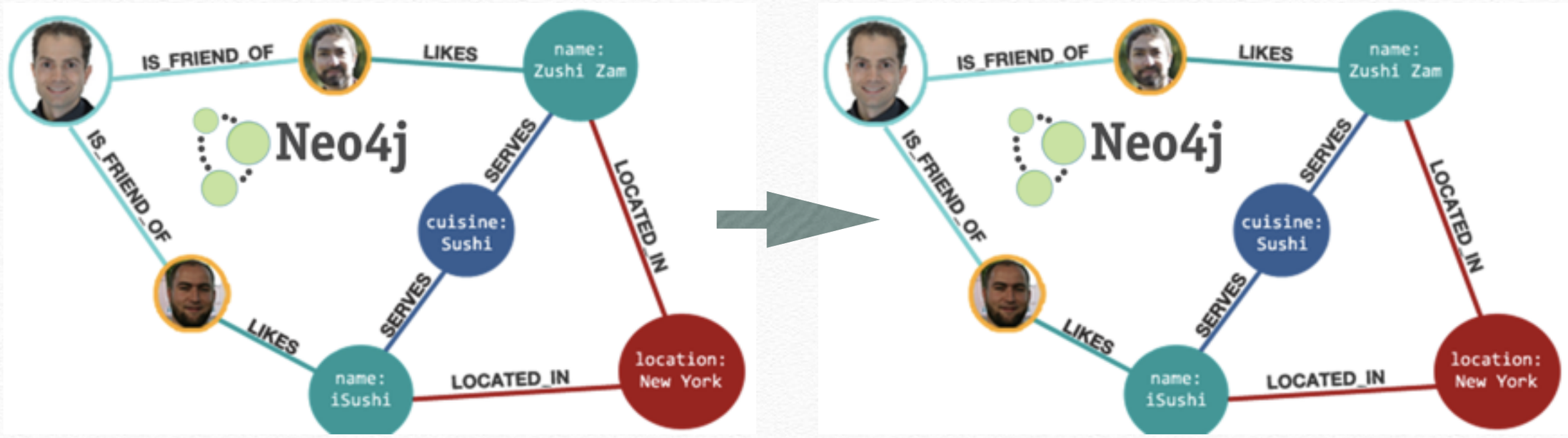
Name	FName	City	Age	Salary
Smith	John	3	35	$280
Doe	Jane	1	28	$325
Brown	Scott	3	41	$265
Howard	Shemp	4	48	$359
Taylor	Tom	2	22	$250



Fraud Detection via Community Detection

- ***CASE STUDY***: Orders coming from an important phone company and requirements about fraud warnings

- ***From Graph to Graph***: requirements produce patterns (graph query) retrieving users with abnormal behavior



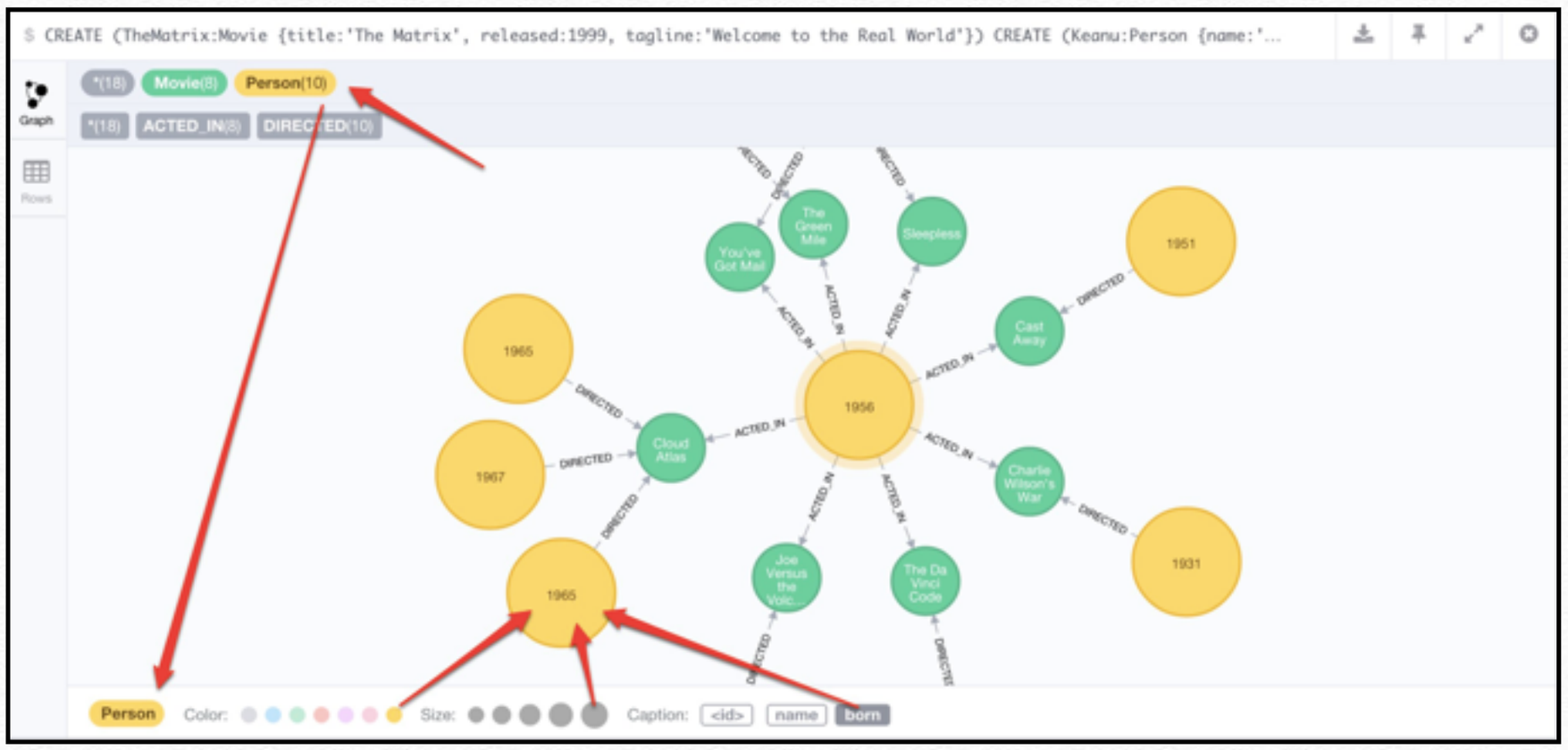
Proposals: #1

- **Visual Pattern computation**: define an on-click front-end to produce and to compute patterns on the graphed

- *case study*:

  - ☑ graph DBMS: Neo4J

  - ☑ multiple databases on Neo4J



Proposals: #2

- **Geo-referential Exploration**: usage of geo-referential front-end for pattern exploration

- *case study*:

  - ☑ geo-referential system: NanoCubes

  - ☑ language: Javascript and D3.js