Instituto Tecnológico de Buenos Aires

Introduction to Graph Databases

NoSQL and Graph Databases

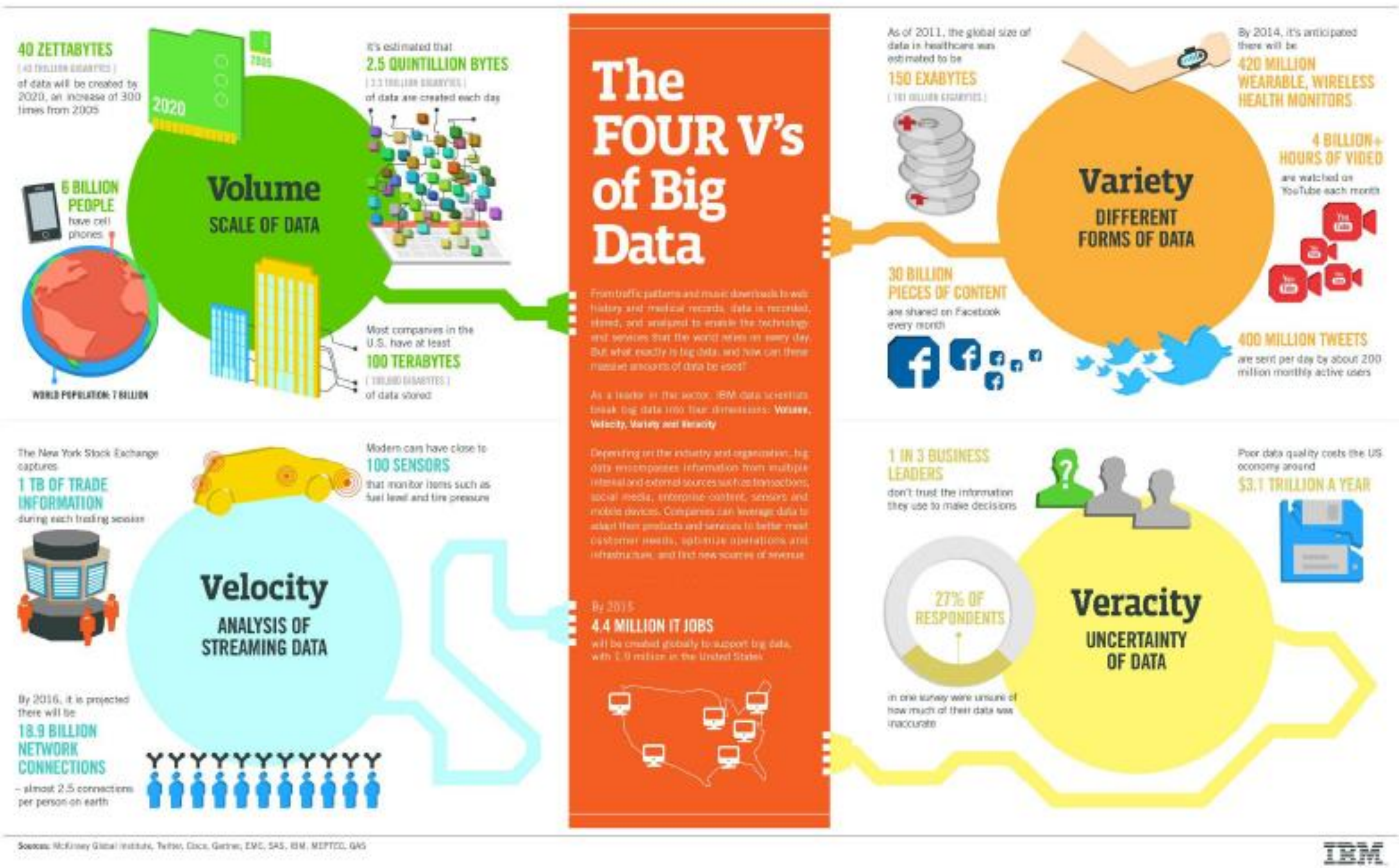
Dr. Alejandro Vaisman
avaisman@itba.edu.ar

Motivation

Typical BI scenario years ago …

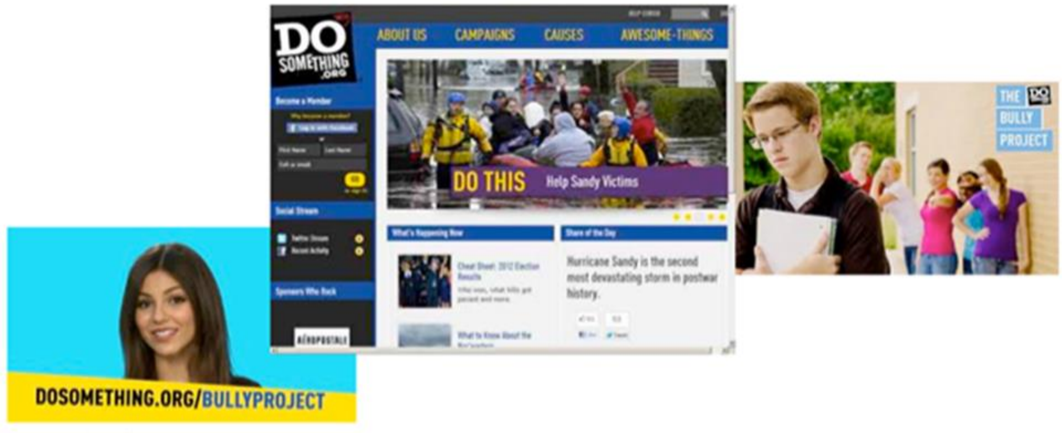
- **Early 90's**: Data Warehousing + Data Mining
- Big data: GB…TB!!
- Structured data
  - Mostly relational
  - Spreadsheets
  - (Some) Text
  - Web still in its infancy
- Problem: Data integration
- **Today**: Data deluge on the Web
- Daily TB of data of different kinds
  - Geographic
  - Text
  - Video, image
  - Audio

Volume, Velocity, Variety, Veracity



Social media monitoring

Social media monitoring



Bully Watch
Monitoring Bullying Together 24/7

Social media monitoring

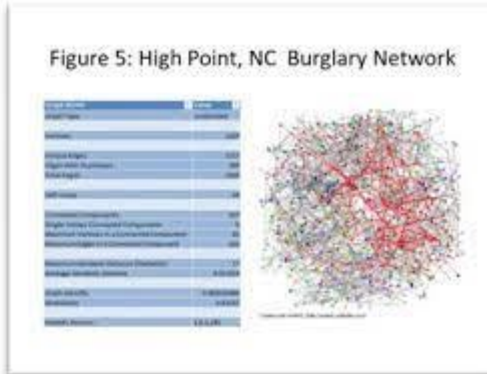


Figure 5: High Point, NC Burglary Network

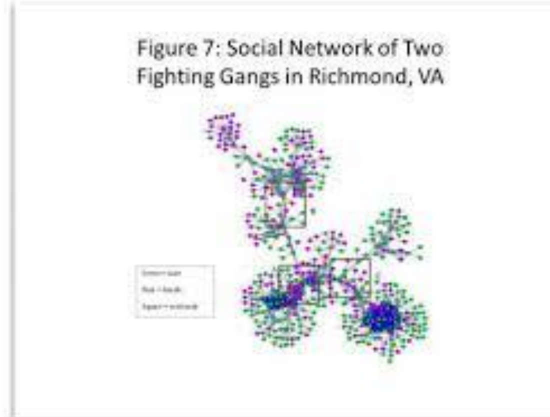
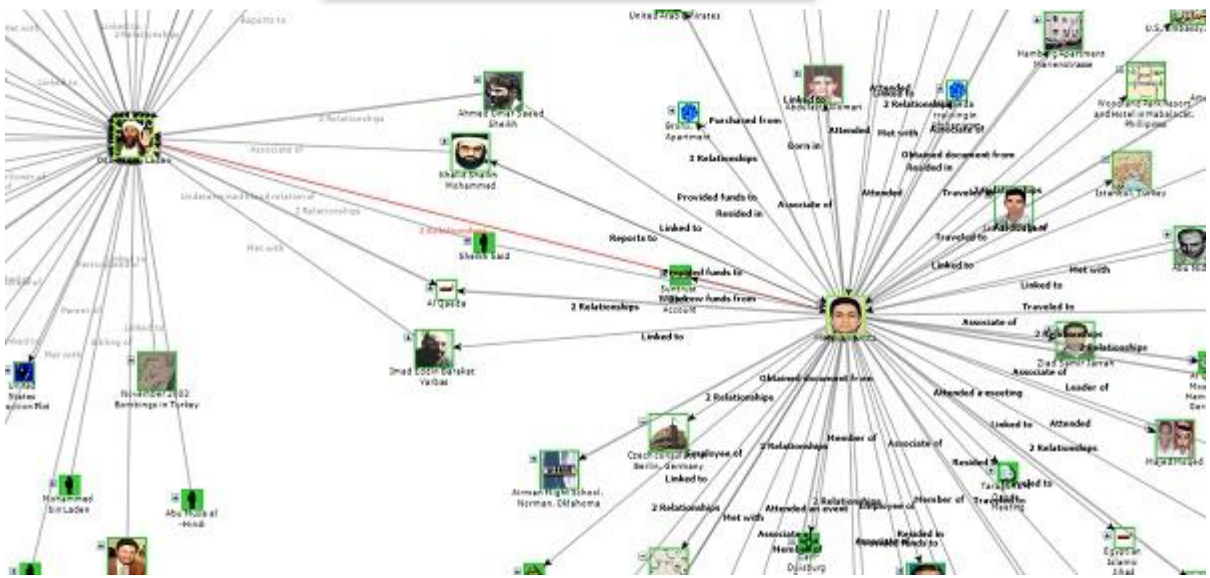


Figure 7: Social Network of Two Fighting Gangs in Richmond, VA

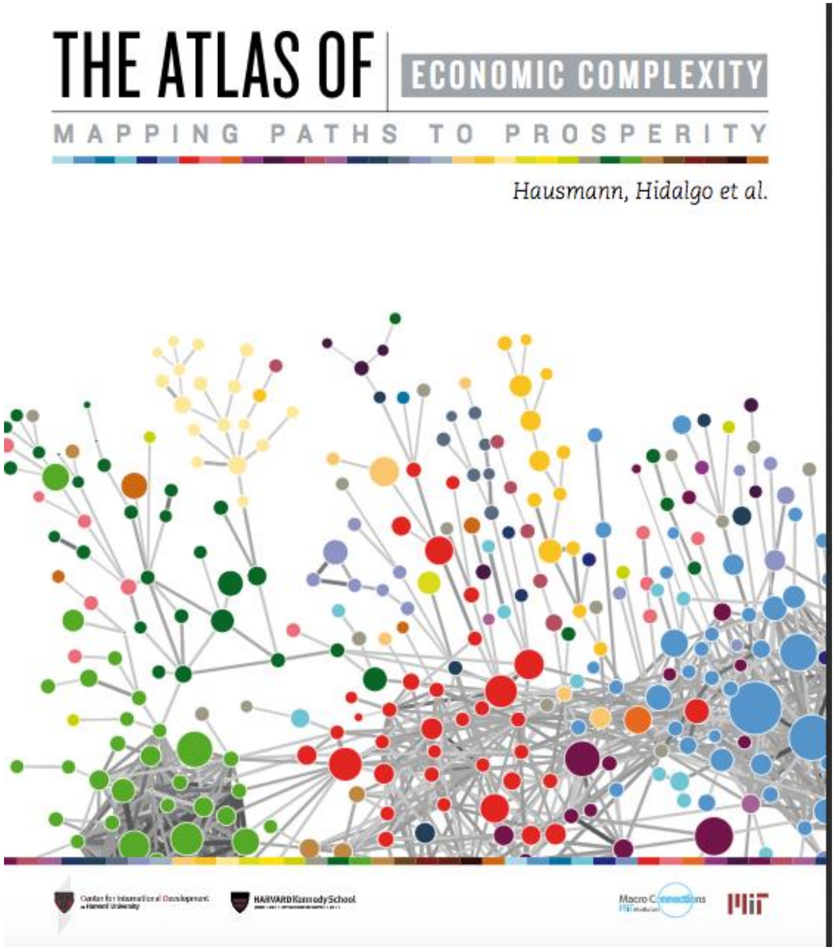


A world of interrelated information

Data of 200 countries, 750 products, 50 years.
Visualizations produced by MIT media lab

http://atlas.cid.harvard.edu/

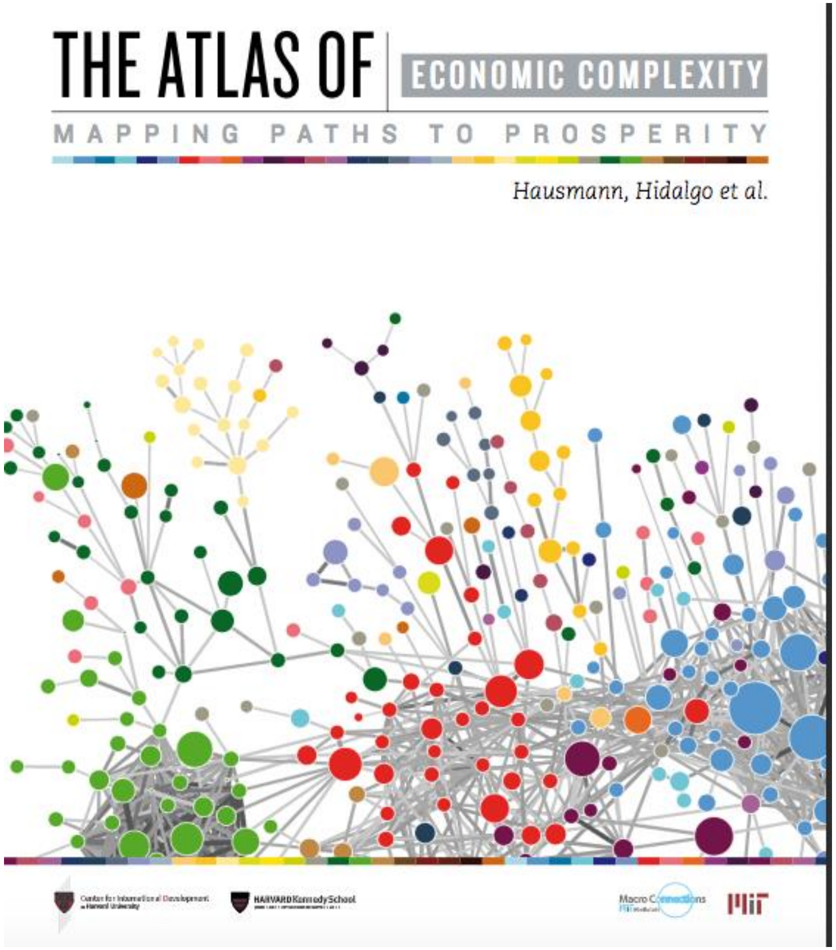
http://globe.cid.harvard.edu/



A world of interrelated information

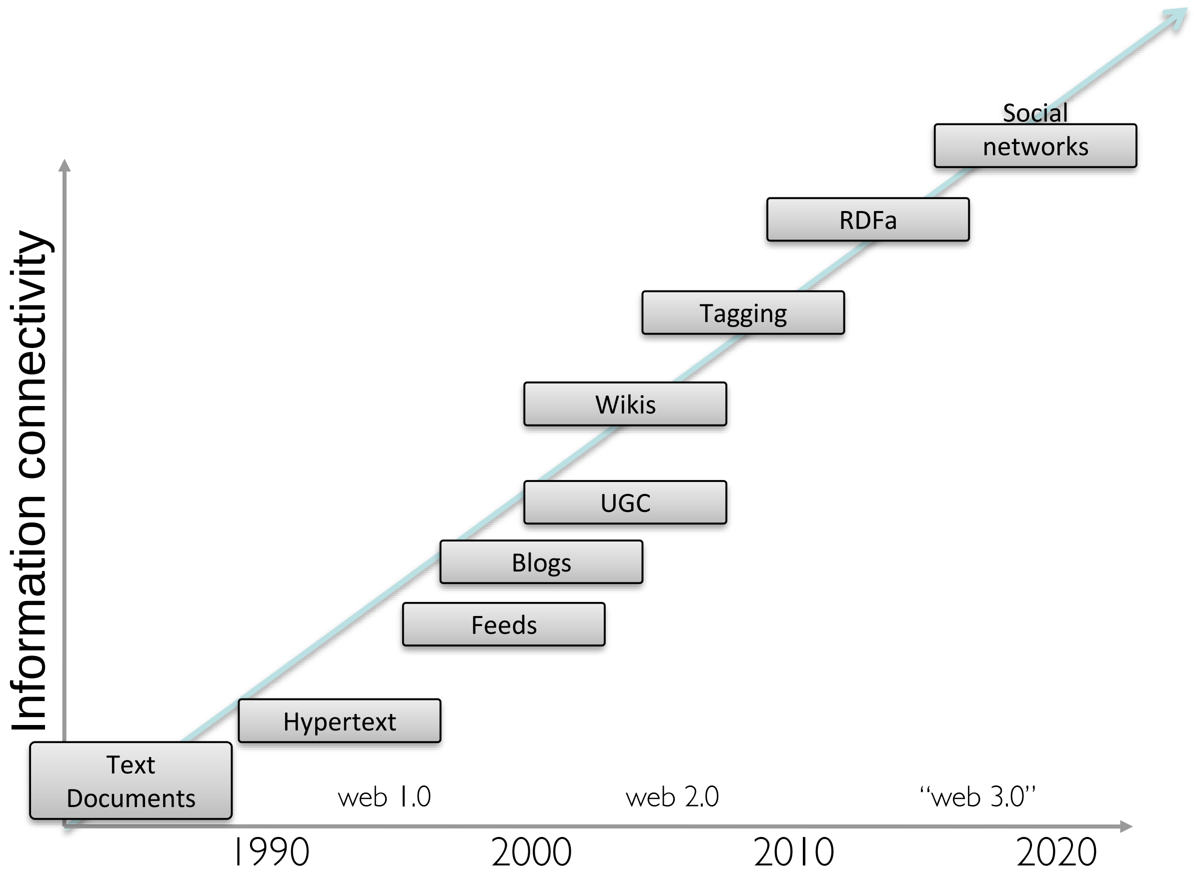
Indicators:

- Capability Distance (countries start making products “close” to the ones they already make, the “chicken and egg” problem)
- Complexity outlook index (how many new products could a country produce)
- Diversity (diferent products produced by a country)
- Economic complexity (measures how much of the knowledge of the society is transferred to the products it produces)



Main characteristic of these data

Trend : Connectedness



How do we deal with this?

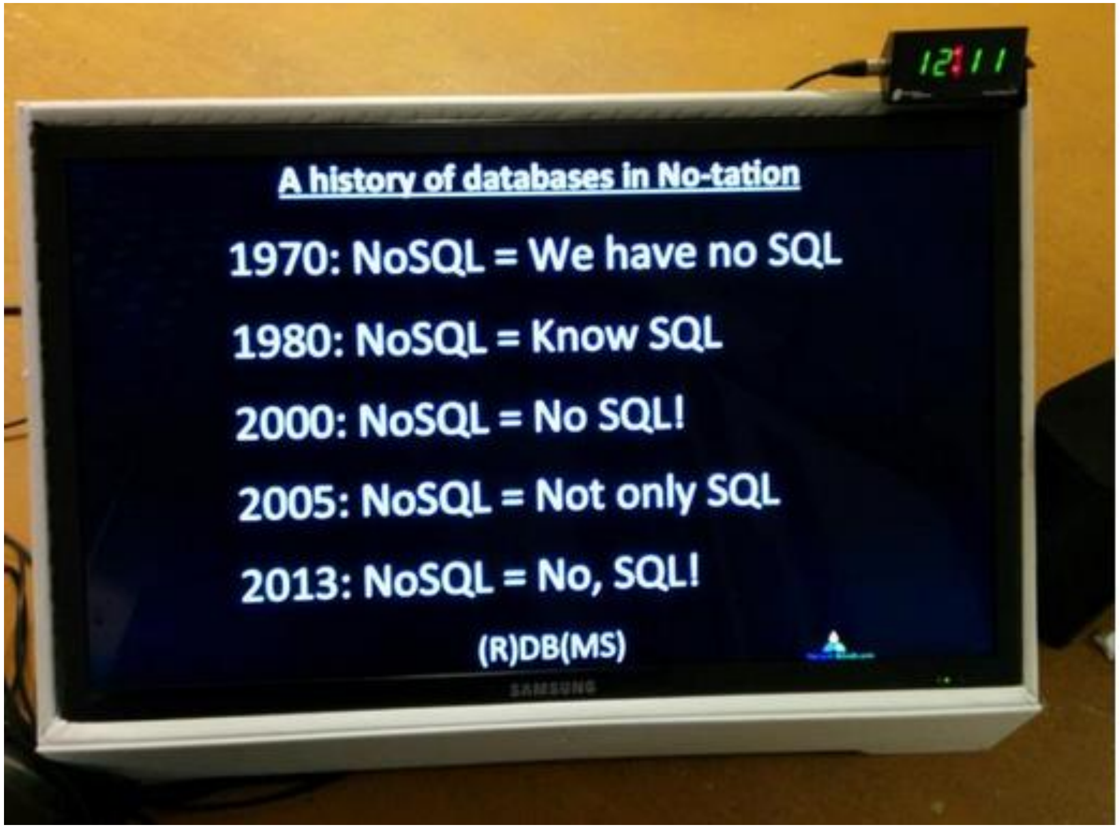
Is traditional DB technology enough?

We must address:

- Connectedness
- Unstructured data
- High Volumes
- Real-time

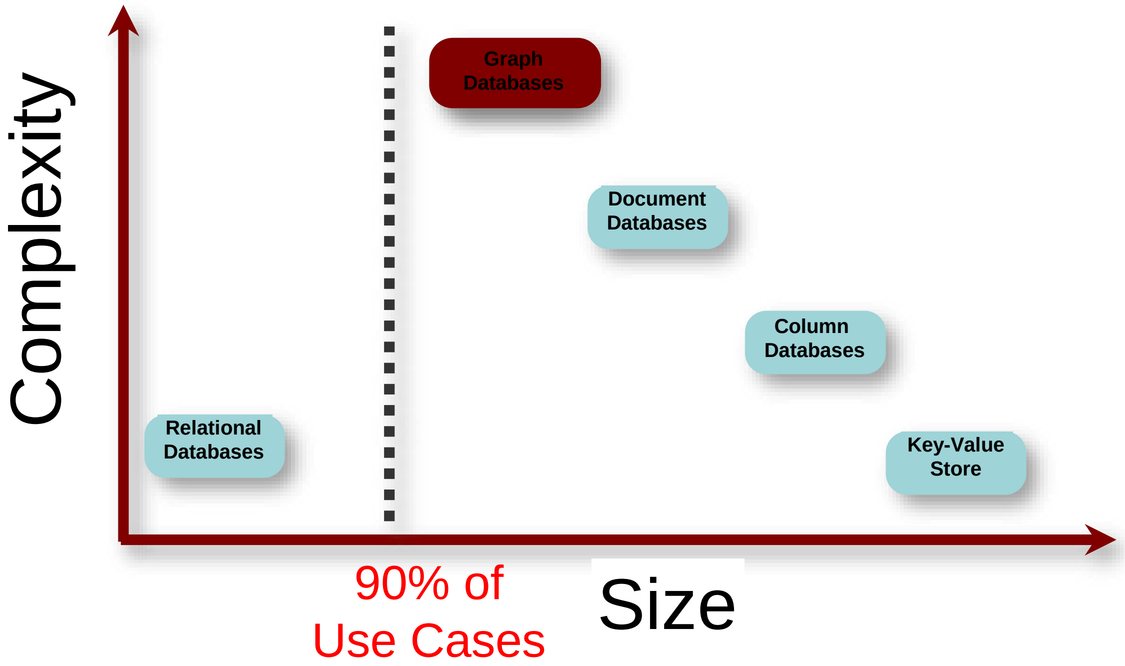
NoSQL technologies

The NoSQL paradigm



- Is SQL the future of NoSQL?

Living in a NoSQL world



NoSQL Motivation

- RDBMS **too rigid** for Big Data scenarios
- Not the best to store **unstructured data**
- **One-size-fits-all approach** no longer valid in many scenarios
- RDBMS **hard to scale** for billions of rows
- Data structures used in RDBMS optimized for systems with **small amounts of memory**

NoSQL characteristics

NoSQL characteristics

- **Not** using the **relational model** for storing data

NoSQL characteristics

- **Not** using the **relational model** for storing data
- **Not** using **SQL** for retrieving data

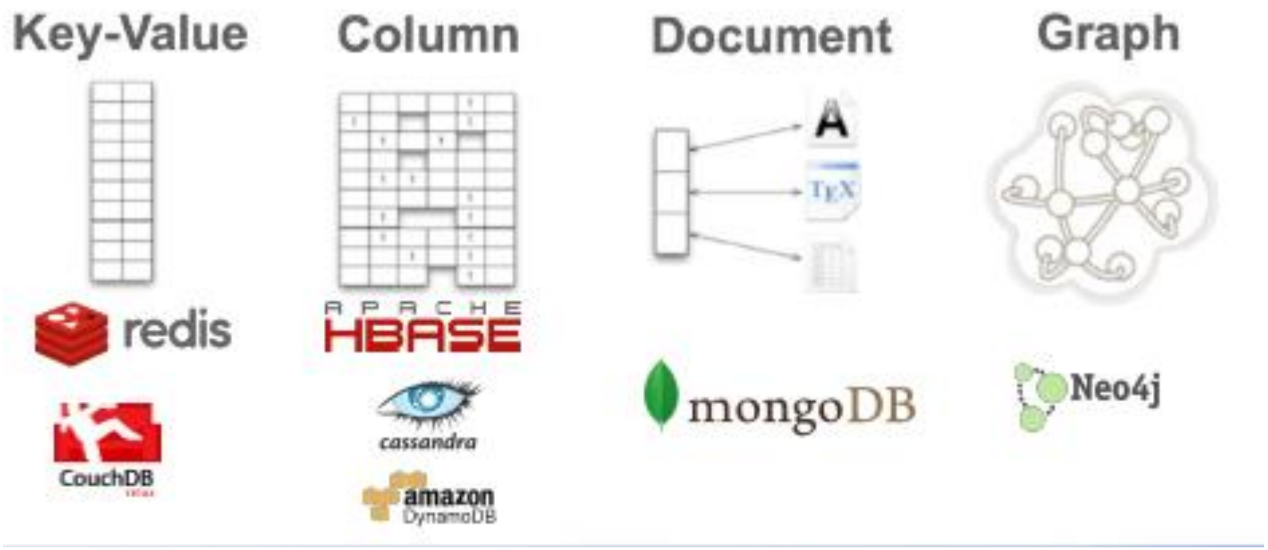
NoSQL characteristics

- **Not** using the **relational model** for storing data
- **Not** using **SQL** for retrieving data
- **No schema**, allowing fields to be added to any record, without control

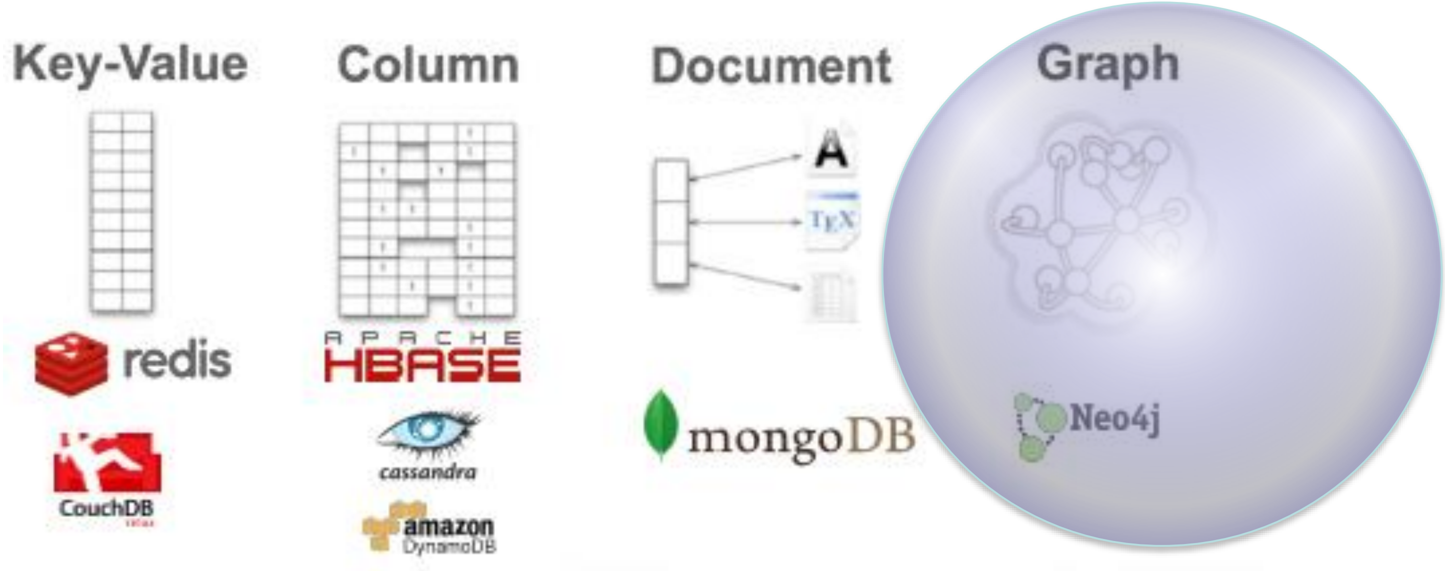
NoSQL characteristics

- **Not** using the **relational model** for storing data
- **Not** using **SQL** for retrieving data
- **No schema**, allowing fields to be added to any record, without control
- Ability to run on **clusters** of commodity hardware
- Ability to web-scale, with **horizontal scalability** in mind
- Trade-off traditional consistency for other useful properties (e.g., **no ACID** support most of the time)

Types of NoSQL Stores



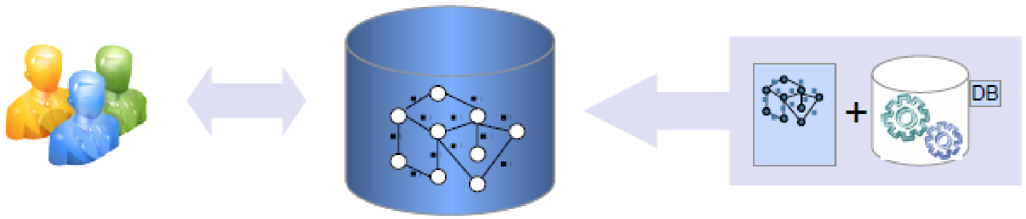
In the remainder….



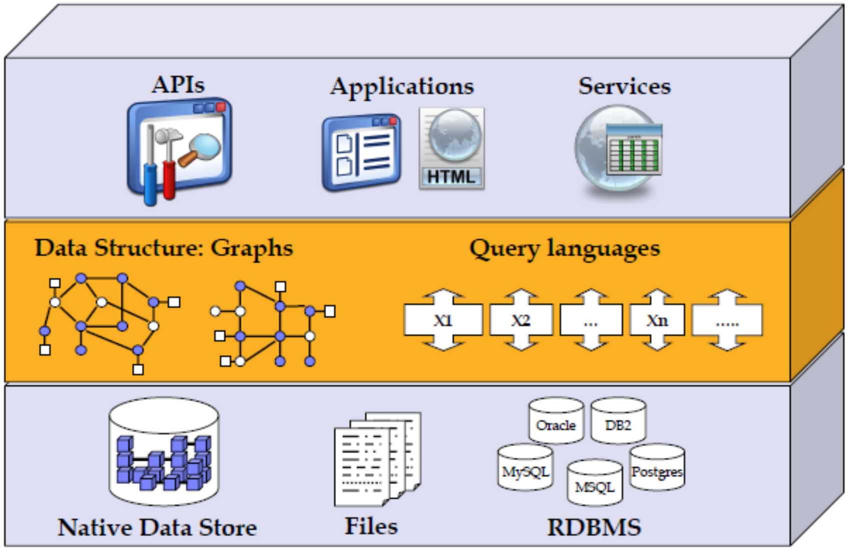
Introduction to Graph Databases

The database approach

- Manage huge data volumes with logics precision
- Separate model and implementation levels



Architecture



But first something some reminders

- Database models (Codd)

Data structures

Integrity constraints

Query Language

Database models

Database model	Abstraction level	Data structure	Information focus
Network	Physical	Pointers,records	Records
Relational	Logical	Relations	Data, attributes
Semantic	User	Graph	Schema, relations
OO	Physical/logical	Objects	Objects, methods
Semi-structured	Logical	Tree	Data,components
Graph	Logical/user	Graph	Data, relations

Database models: graphs

Data structure

Data and/or schema are represented by graphs, or by data structures generalizing the notion of graph (hypergraphs or hypernodes)

Database models (Codd)

Integrity constraints

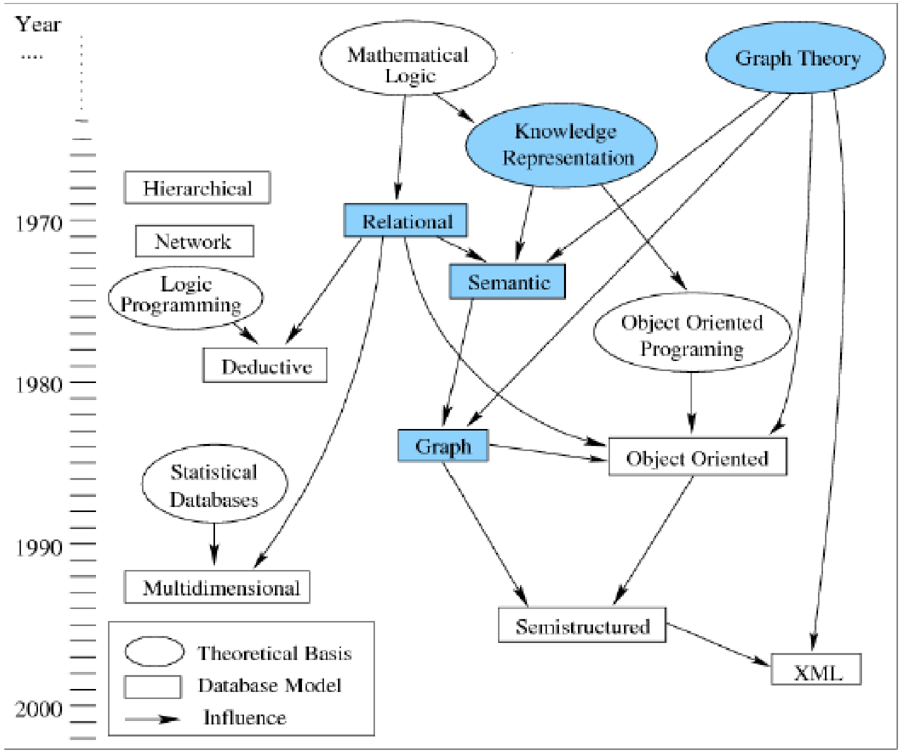
Enforce data consistency. Constraints can be grouped in schema-instance consistency, identity and referencial integrity, and functional and inclusion dependencies

Database models: (Codd)

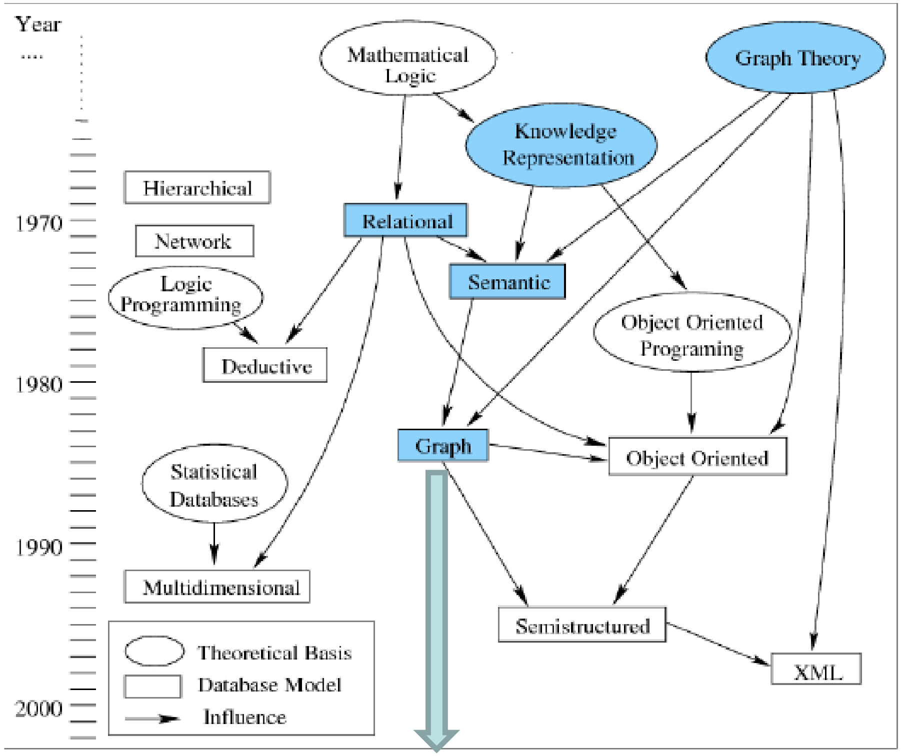
Query language

Data manipulation is expressed by graph transformations, or by operations whose main primitives are on graph features, like paths, neighborhoods, subgraphs, connectivity, and graph satistics.

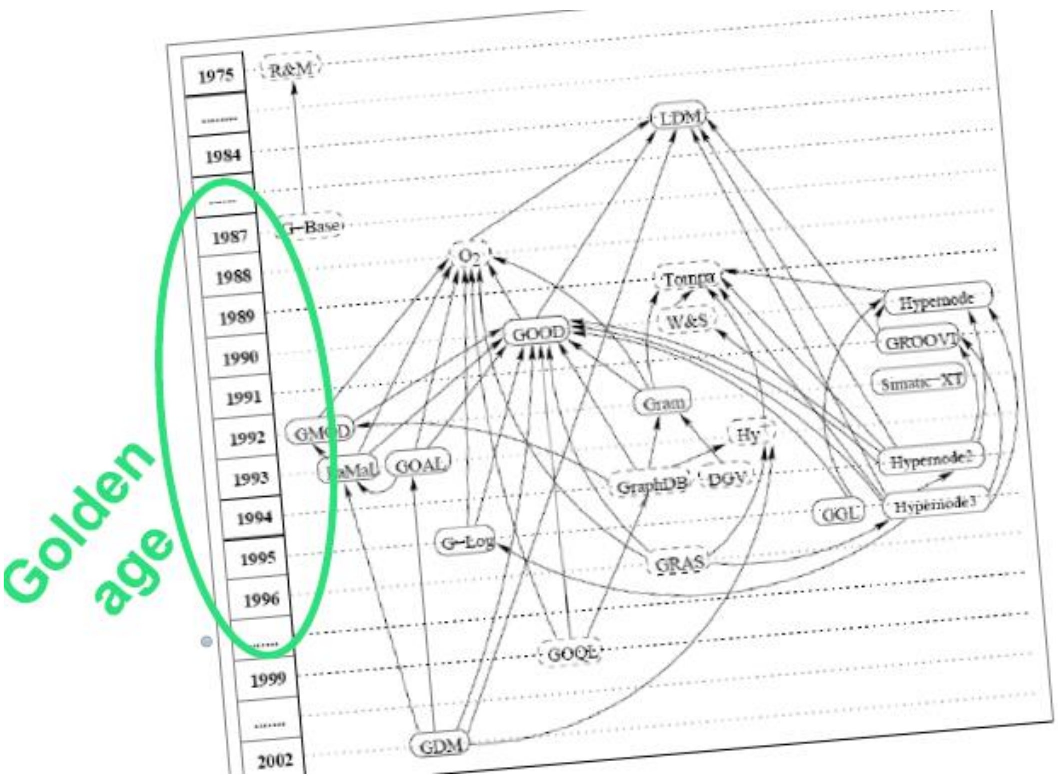
A history of database models (A. Mendelzon)



A history of database models (A. Mendelzon)

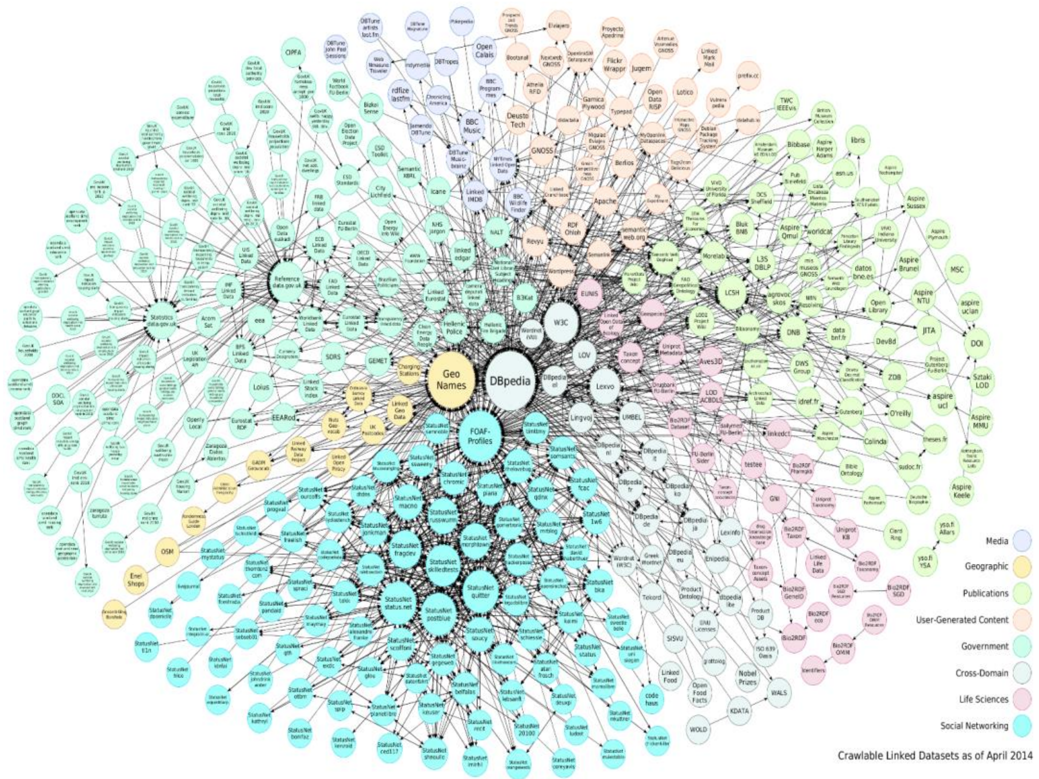


The Golden age of GDB



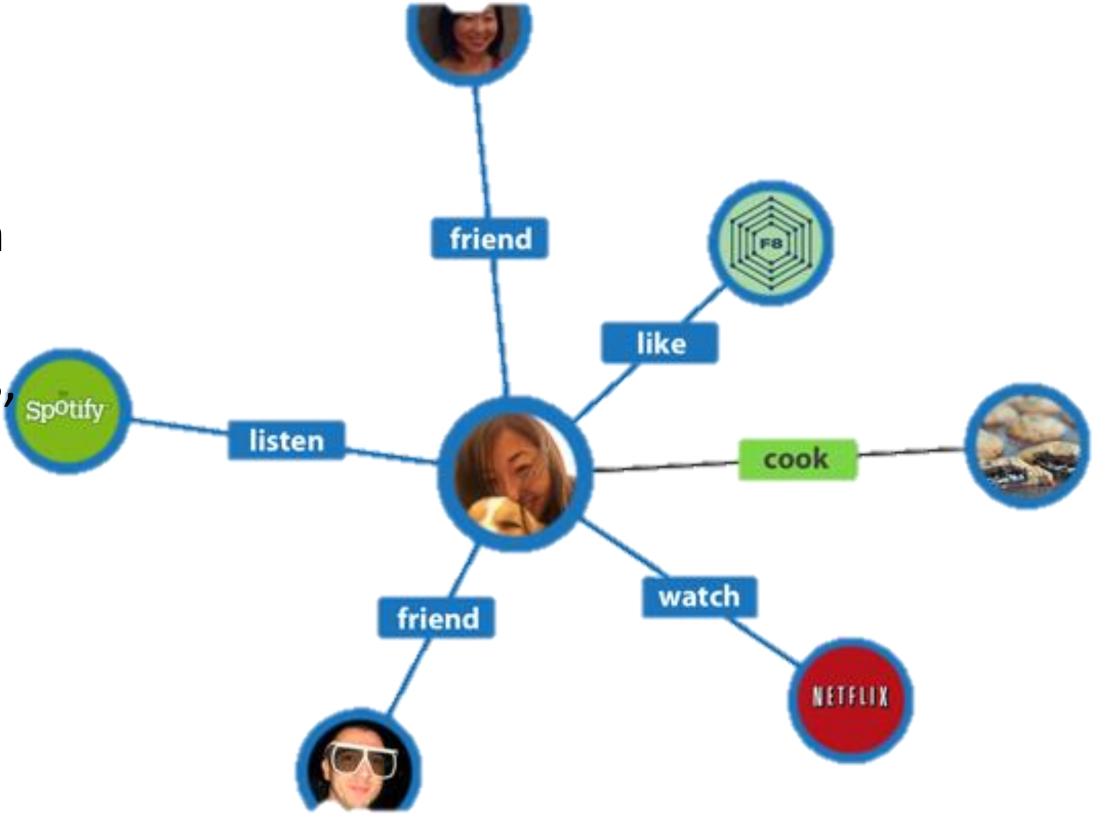
Data connectedness

- Facebook Graph
- LinkedIn Graph
- Linked Data
- Blogs/Tagging



Data connectedness

- Modeling the FB Graph
- Persons, friendships, photos, locations, apps, pages, ads, interests, age range, etc.



Social Network "path" performance

- Experiment:
  - ~1k persons
  - Average 50 friends per person
  - `pathExists(a,b)` limited to depth 4
  - Caches warm to eliminate disk IO

	# persons	query time
Relational database	1000	2000ms

Social Network “path” performance

- Experiment:
  - ~1k persons
  - Average 50 friends per person
  - `pathExists(a,b)` limited to depth 4
  - Caches warm to eliminate disk IO

	# persons	query time
Relational database	1000	2000ms
Neo4j	1000	2ms

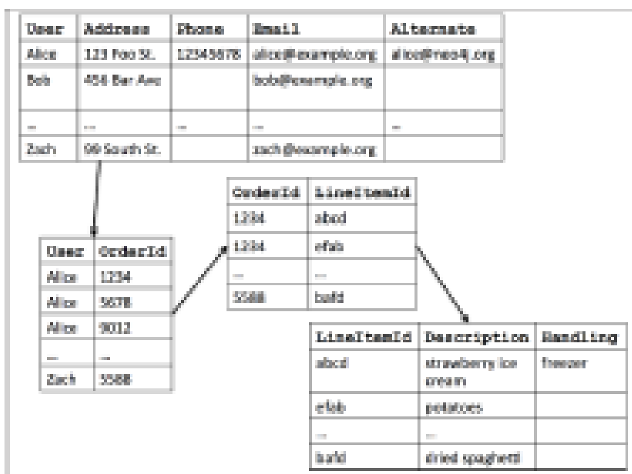
Social Network "path" performance

- Experiment:
  - ~1k persons
  - Average 50 friends per person
  - `pathExists(a,b)` limited to depth 4
  - Caches warm to eliminate disk IO

	# persons	query time
Relational database	1000	2000ms
Neo4j	1000	2ms
Neo4j	1000000	2ms

Traversing data in a RDBMS

- Based on joining and selecting data



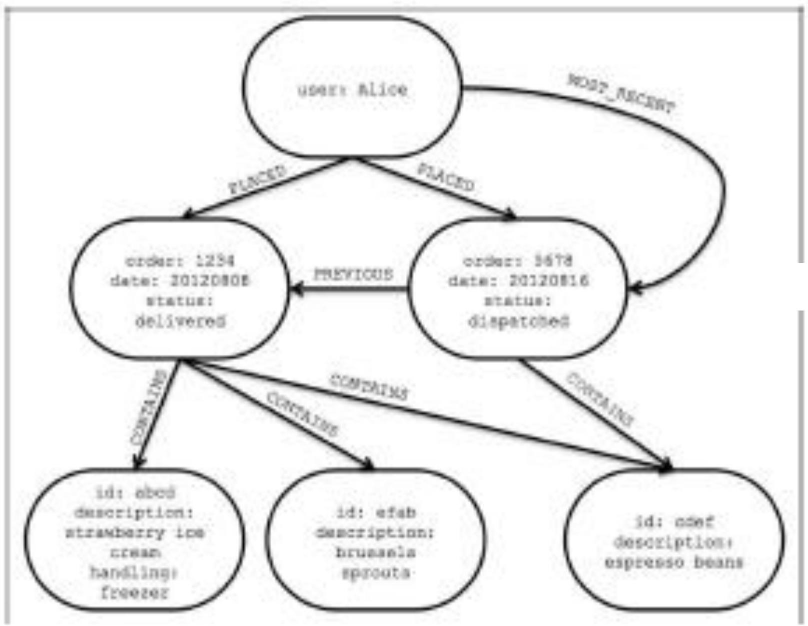

```
SELECT *
FROM user u, user_order uo,
orders o, items i
WHERE u.user = uo.user AND
uo.orderId = o.orderId AND
i.lineItemId = i.LineItemId
AND u.user = 'Alice'
```

<u>**Cardinalities:**</u>

|User|: 5.000.000
|UserOrder|: 100.000.000
|Orders|: 1.000.000.000
|Item|: 35.000

Query Cost?!

Traversing data in a GDB



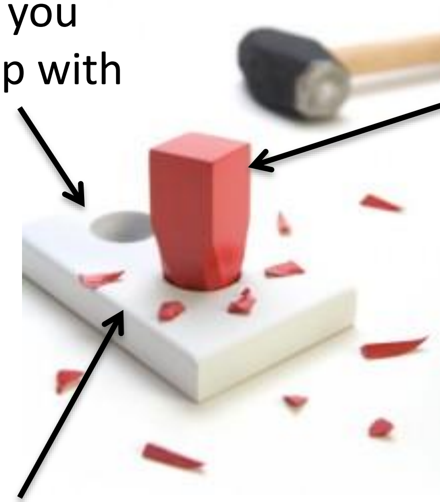
Cardinalities:

|User|: 5.000.000
|Orders|: 1.000.000.000
|Item|: 35.000

Query Cost?!

O(N)

Forcing the relational model in a graph



Think in terms of nodes and edges!

Consequence: graph DB are **hot** again

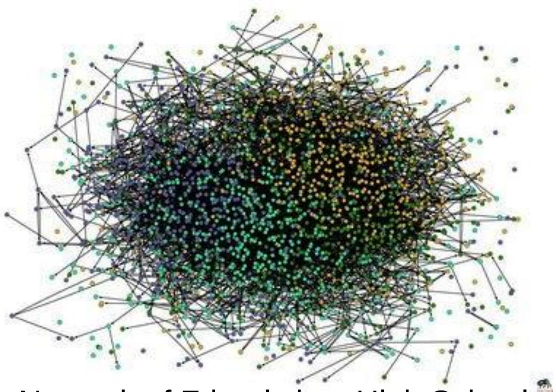
- The question is: how to process HUGE graphs?
- We now have hardware that can do this

- As usual
  - Software runs behind hardware;
  - Theory and models behind software
  - Current status: problem understanding

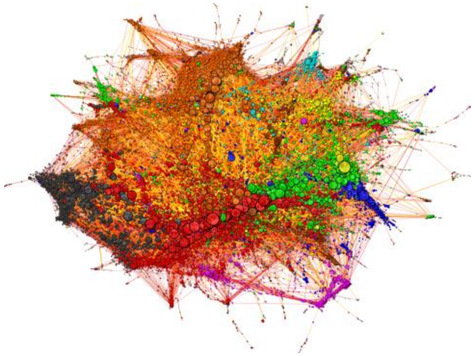
Open questions (almost all of the stuff)

- Use cases
- Data structures
- Query languages and operators
- Benchmarks
- Open world vs. closed world
- Centralized or distributed?
- Dynamics – transactions

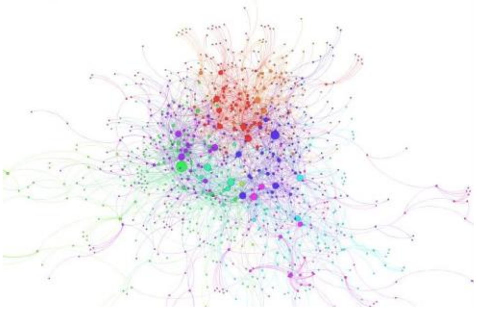
Use cases: social networks



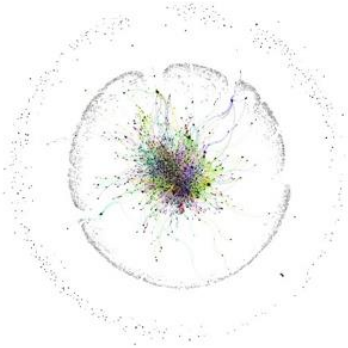
Network of Friends in a High School



Relationships between artists in Last.fm
http://sixdegrees.hu/last.fm/



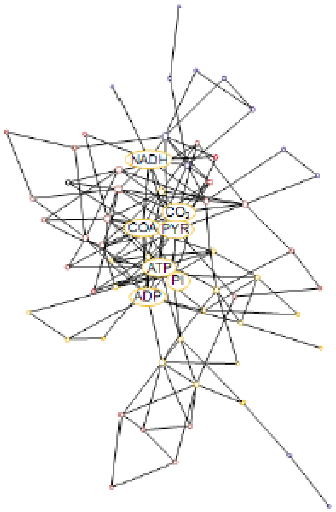
Network structure of music genres and their stylistic origin
http://www.infosysblogs.com/web2/2013/01/network_structure_of_music_gen.html



Network structure of Patent Citations
http://www.infosysblogs.com/web2/2013/07/

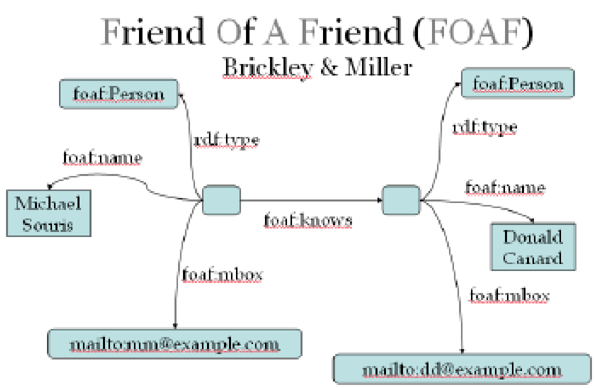
Use cases: biology

Use Case	Graph Query
Chemical structure associated with a node	Node matching
Find the difference in metabolisms between two microbes	Graph intersection, union, difference
To combine multiple protein interaction graphs	Majority graph query
To construct pathways from individual reactions	Graph composition
To connect pathways, metabolism of co-existing organisms	Graph composition
Identify "important" paths from nutrients to chemical outputs	Shortest path queries
Find all products ultimately derived from a particular reaction	Transitive Closure
Observe multiple products are co-regulated	Least common ancestor
To find biopathways graph motifs	Frequent subgraph recognition
Chemical info retrieval	Subgraph isomorphism
Kinaze enzyme	Subgraph homomorphism
Enzyme taxonomies	Subsumption testing
To find biopathways graph motifs	Frequent subgraph recognition

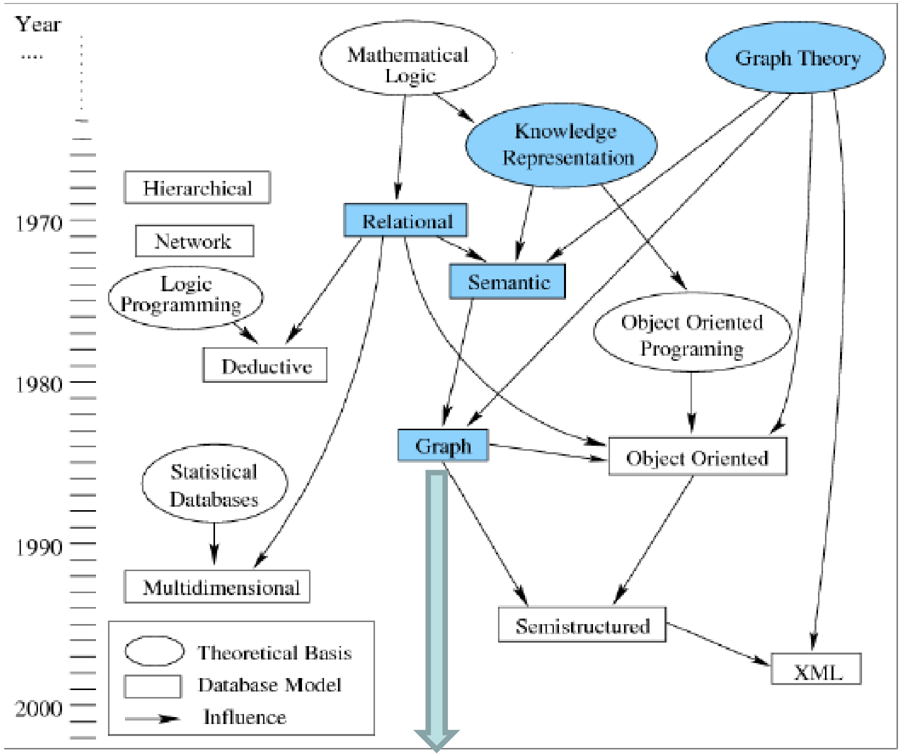


Use cases: the web

Use Case	Graph Query
What is/are the most cited paper/s?	Degree of a node
What is the influence of article D?	Paths
What is the Erdõs distance between authos X and author Y?	Distance
Are suspects A and B related?	Paths
All relatives of degree one of Alice	Adjacency



Graph database models

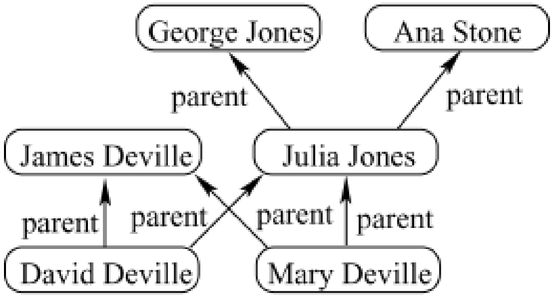


Graph database models

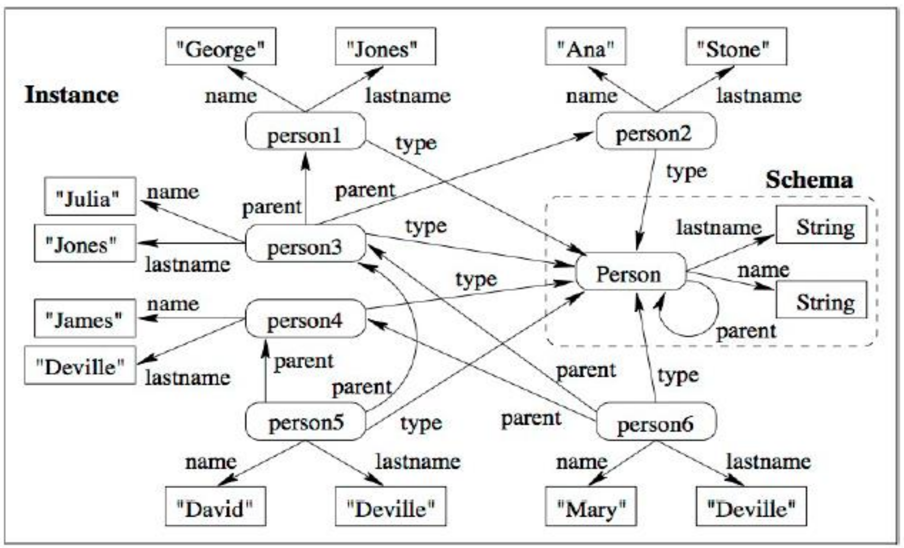
Example: genealogy and data diagram

NAME	LASTNAME
George	Jones
Ana	Stone
Julia	Jones
James	Deville
David	Deville
Mary	Deville

PERSON	PARENT
Julia	George
Julia	Ana
David	James
David	Julia
Mary	James
Mary	Julia

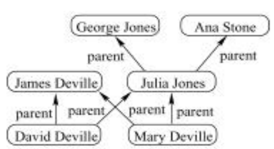


The RDF graph data model



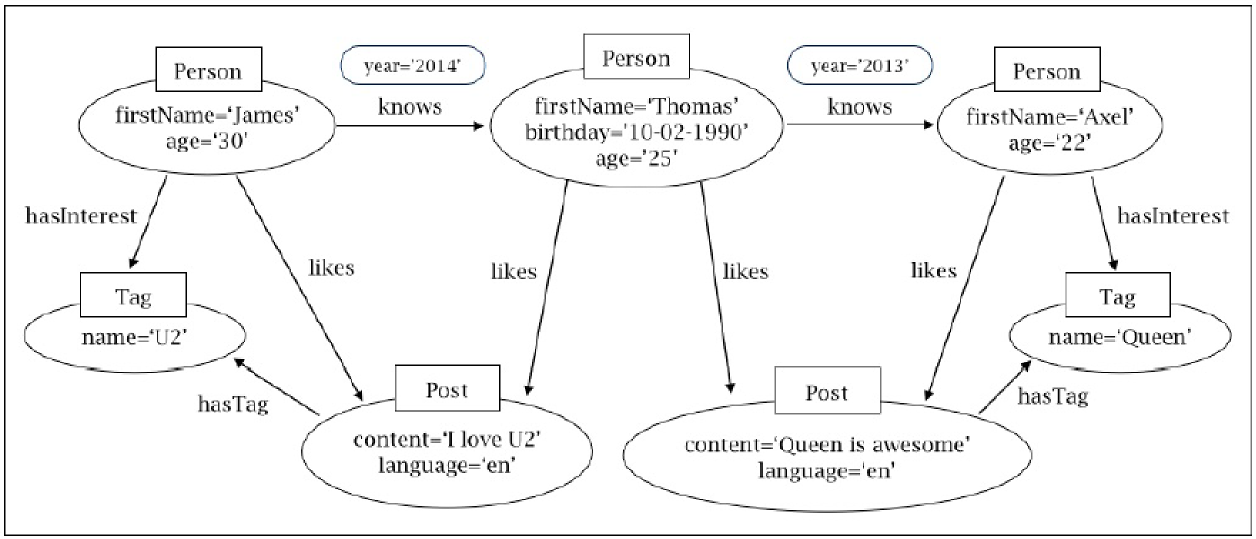
NAME	LASTNAME
George	Jones
Ana	Stone
Julia	Jones
James	Deville
David	Deville
Mary	Deville

PERSON	PARENT
Julia	George
Julia	Ana
David	James
David	Julia
Mary	James
Mary	Julia



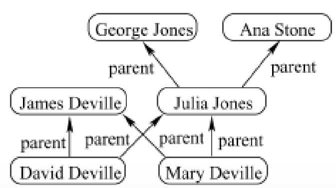
- Not oriented explicitly to model connectivity.
- Originally deviced to represent metadata.
- Represents resources and relations between resources.
- No assumption of the application domain.
- Generally implemented over a RDB as triples/quadruples

The property graph data model



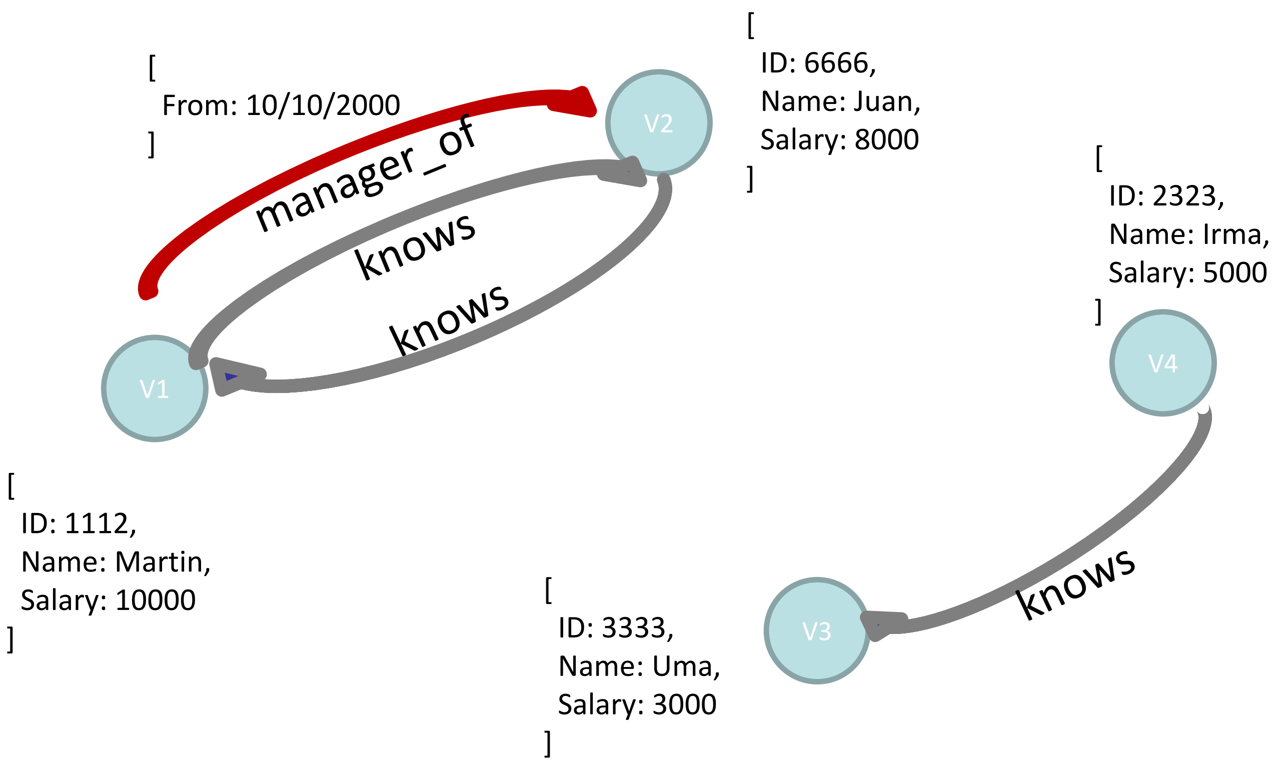
NAME	LASTNAME
George	Jones
Ana	Stone
Julia	Jones
James	Deville
David	Deville
Mary	Deville

PERSON	PARENT
Julia	George
Julia	Ana
David	James
David	Julia
Mary	James
Mary	Julia



- Appropriate for path traversal
- Typically implemented over native graph dbs, with specialized data structure

Property graphs - example



Typical SQL query

Index lookup on Person.id

Person

Id	Name
1	Larry Page
2	Joshua Bloch
3	Brian Goetz
N	…

Index lookup on CompanyId

Works In

PersonId	CompanyId	Since
1	1	1998
2	1	2001
3	2	2010

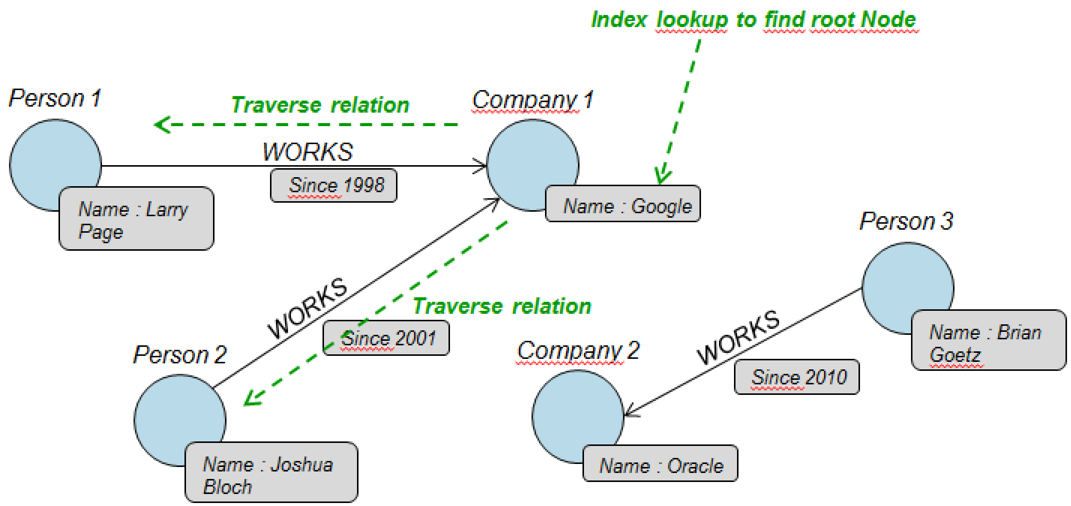
Index lookup on Company.name

Company

Id	Name
1	Google
2	Oracle
…	…
N	…

Select Person.Name
from Person, Company, WorksIn
where Company.name='Google'
and WorksIn.CompanyId = Company. Id
and WorksIn.PersonId = Person.Id

Same query on property graphs



The deepest the navigation, the largest the difference with RDBs

Graph databases

- Address needs for managing graph data
- Architecture goals inspired (as always) by classic DBMSs
- Persistent storage of graph data
- Address transactionality
- Closed world
- Efficiency (over scalability)
- Portability of data (near future)
- Declarative query languages (near future)

Graph query languages

- Basic graph queries
  - Content-based queries
    - Get a node, get the value of a node / edge attribute, etc.
    - A typical case are summarization queries (i.e., aggregations)
- Pattern matching
- Adjacency/neighbourhood (out-degree, in-degree)
    - Find all friends of a person
    - Airports with a direct connection
- Reachability/connectivity
    - Fixed-length paths (fixed #edges and nodes)
    - Regular simple paths (restrictions as regular expressions)
    - Hybrid if the restriction is in the *content*
    - Shortest path
    - Examples:
      - Friend-of-a-friend
      - Flight connections

Graph query languages

- **Analytical queries**
  - Centrality measures
  - Diameter and other global properties
  - Various statistics
  - Graph summarization
  - Graph OLAP

Graph query languages: what do we need

- Genericity (Independence of how data are coded)
- Expressive power
- Simple syntax
- Clear and simple semantics
- Compositionality
- Limited number of (simple) constructors
- User friendliness
- Standards

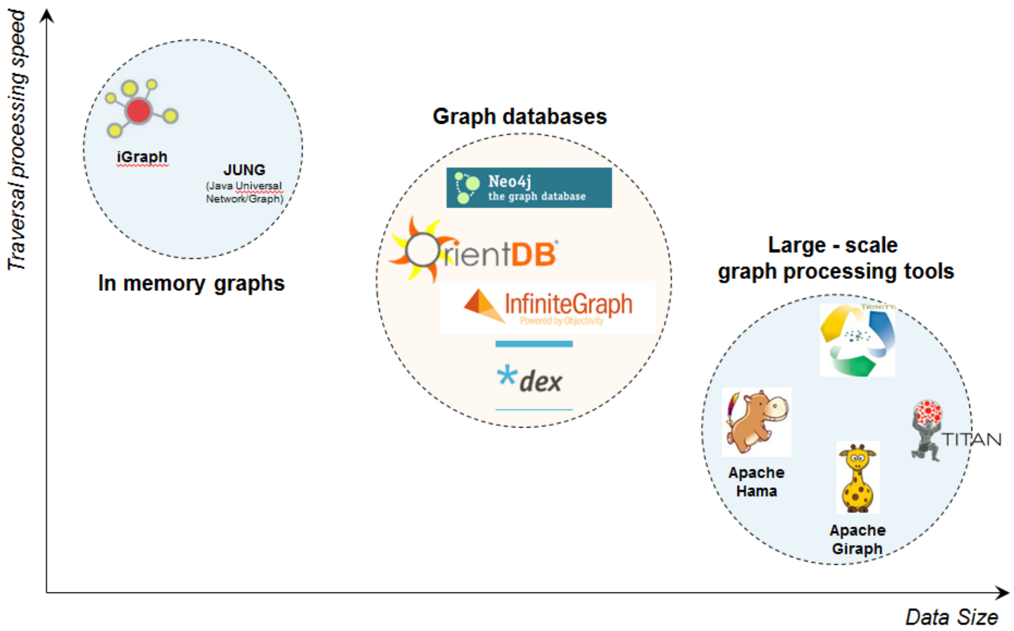
Graph processing frameworks

1. Batch processing
2. Analysis of large graphs
3. Facilities for graph analytical algorithms
4. Distributed environment
5. Multiple machines
6. API or programming as user access

Graph processing frameworks

- Pregel
- Apache Giraph
- GraphLab
- Catch the Wind
- GPS
- Mizan
- Power Graph
- GraphX
- TurboGraph
- GraphChi
- …

Graph DB vs. Graph processing



Graph DB Technology

- Will they be adopted?
- How do we select DB technologies?

	RDBMS	OO	Graph
Work with 1000s of objects and 1 to many relations, properly indexed?			
Allow for Pattern Matching and Recursive Graph Search?			
Change structure of data on a regular basis?			
Work with rules and reasoning?			
Can I find the programmers and DBA's to deal with these new technologies			
Will it work with the existing reporting tools?			
Will it work with my existing RDBMS?			

Tech reasons

Business Considerations

More questions (practical)

- Will they be adopted?
- How do we select DB technologies?

	RDBMS	OO	Graph
Work with 1000s of objects and 1 to many relations, properly indexed	-	+	++
Allow for Pattern Matching and recursive Graph Search?	-	++	++
Change structure of your data on a regular basis?	-	+	++
Work with rules and reasoning?	-	+	++
Can I find the programmers and DBA's to deal with these new technologies	++	-	-
Will it work with the existing reporting tools?	++	-	-
Will it work with my existing RDBMS?	++	-	+

Next, we get into GRAPH DB’s