Introduction to Graph Databases

NoSQL and Graph Databases

Dr. Alejandro Vaisman
avaisman@itba.edu.ar

Agenda

- 10.10.22. Introduction – Graph data models
- 13.10.22. Graph DB internals. Introduction to Neo4j
- 17.10.22. Querying Neo4j databases
- 20.10.22. Assignment 1. Graphs in relational databases
- 24.10.22. Assignment 2. Basic Cypher queries
- 27.10.22. Assignment 3. Advanced Cypher queries

Agenda

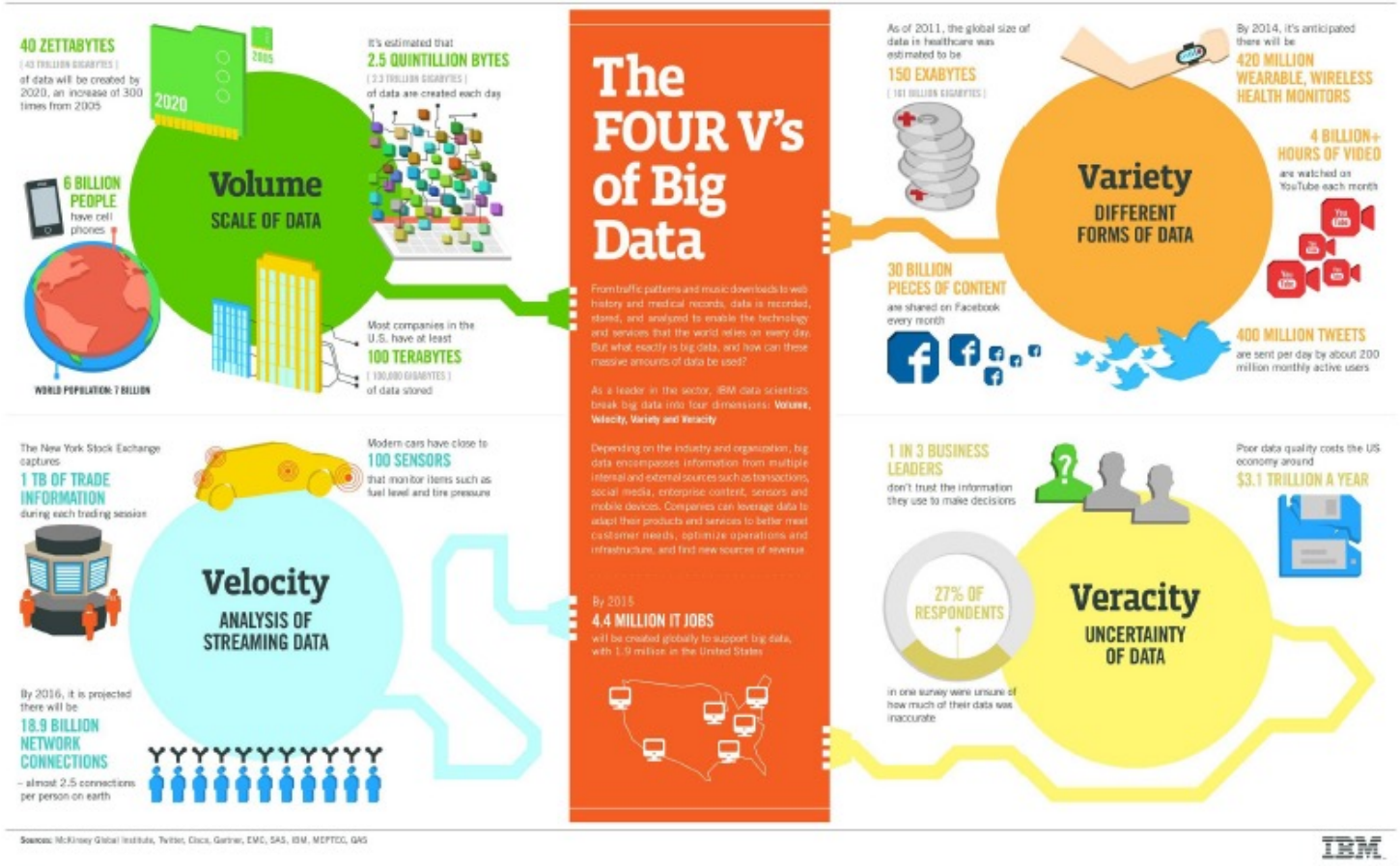
- 10.10.22. **Introduction – Graph data models**
- 13.10.22. Graph DB internals. Introduction to Neo4j
- 17.10.22. Querying Neo4j databases
- 20.10.22. Assignment 1. Graphs in relational databases
- 24.10.22. Assignment 2. Basic Cypher queries
- 27.10.22. Assignment 3. Advanced Cypher queries

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 PB of data of different kinds
  - Geographic
  - Text
  - Video, image
  - Audio

Volume, Velocity, Variety, Veracity



Main characteristic of these data

Social media monitoring

Social media monitoring

"Bullying" analysis on social networks

A world of interrelated information

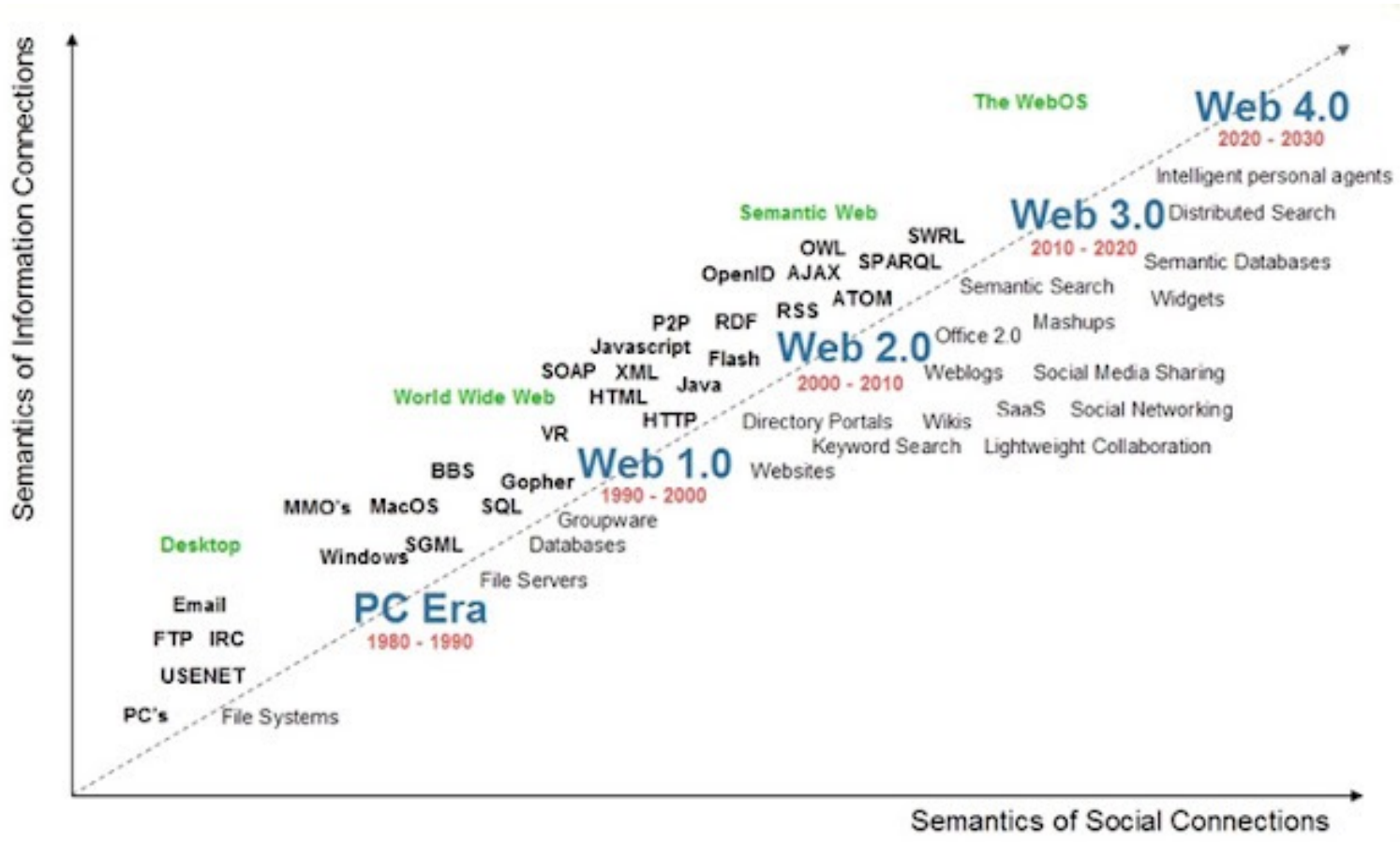
Data of 200 countries, 750 products, 50 years.
Visualizations produced by MIT media lab

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)

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

Trend : Connectedness



How do we deal with this?

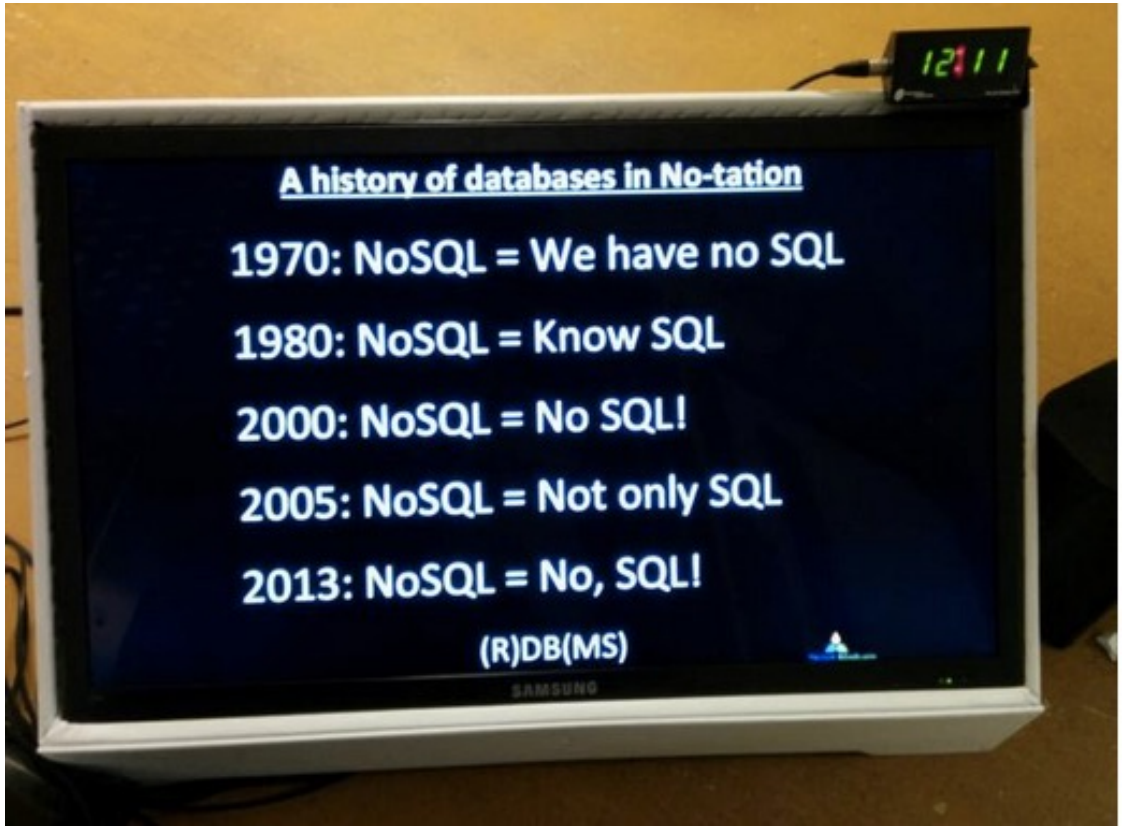
Is traditional DB technology enough?

We must address:

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

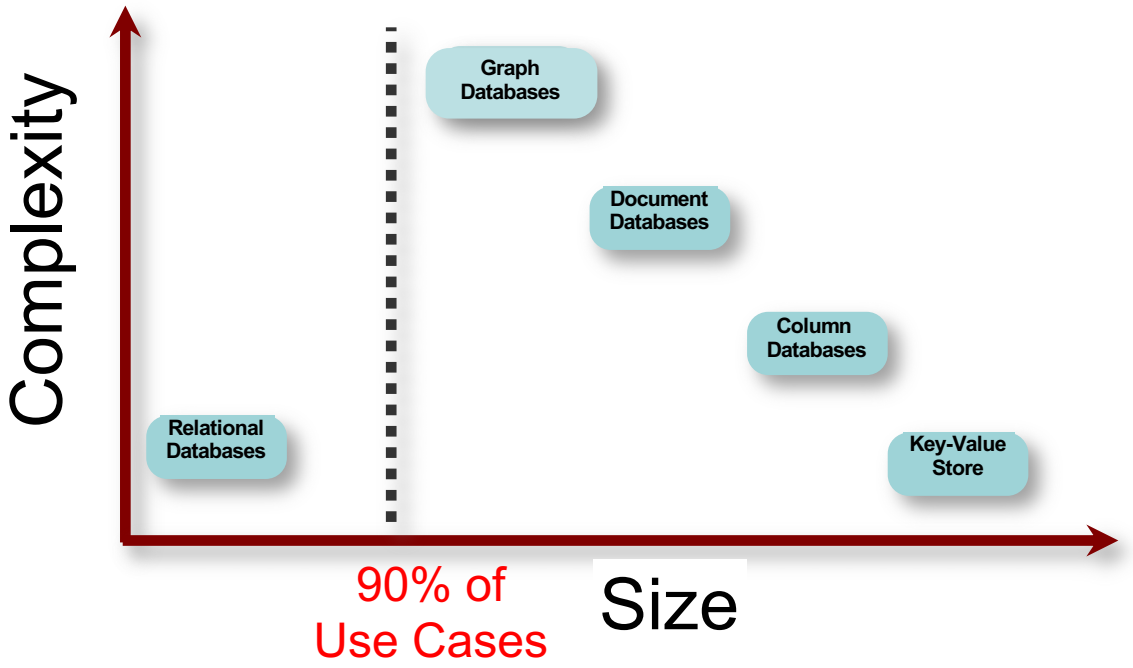
NoSQL technologies

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

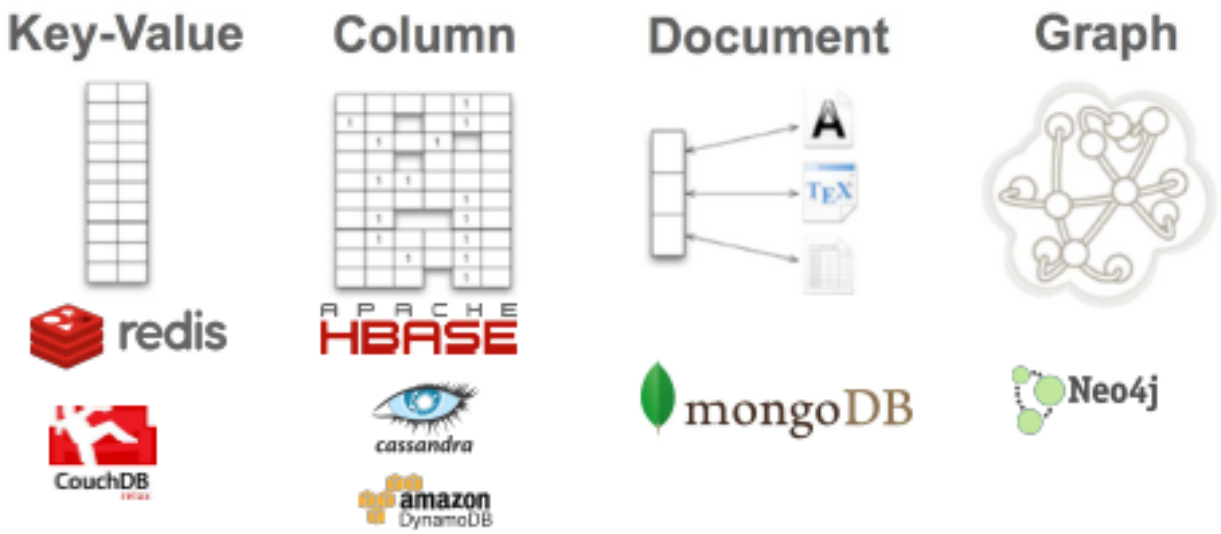
NoSQL characteristics

- **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



Recall the CAP theorem

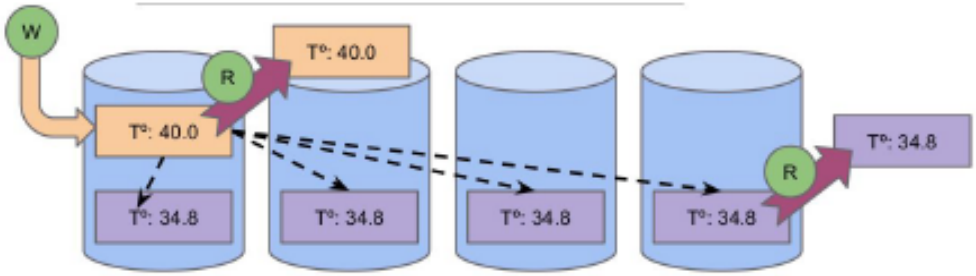
- **Consistency:** Strong consistency of updates
- **Avaliability:** Guarantees that every request receives a response whether it succeded or failed when retrieving data
- **Partition tolerance:** The system continues to operate despite arbitrary message lost or failure of part of the system
- **CAP Theorem: A distributed data system cannot guarantee the three properties above, but only any conbination of two of them.**

- In other words, the CAP theorem states that in the presence of a network partition, one has to choose between consistency and availability.

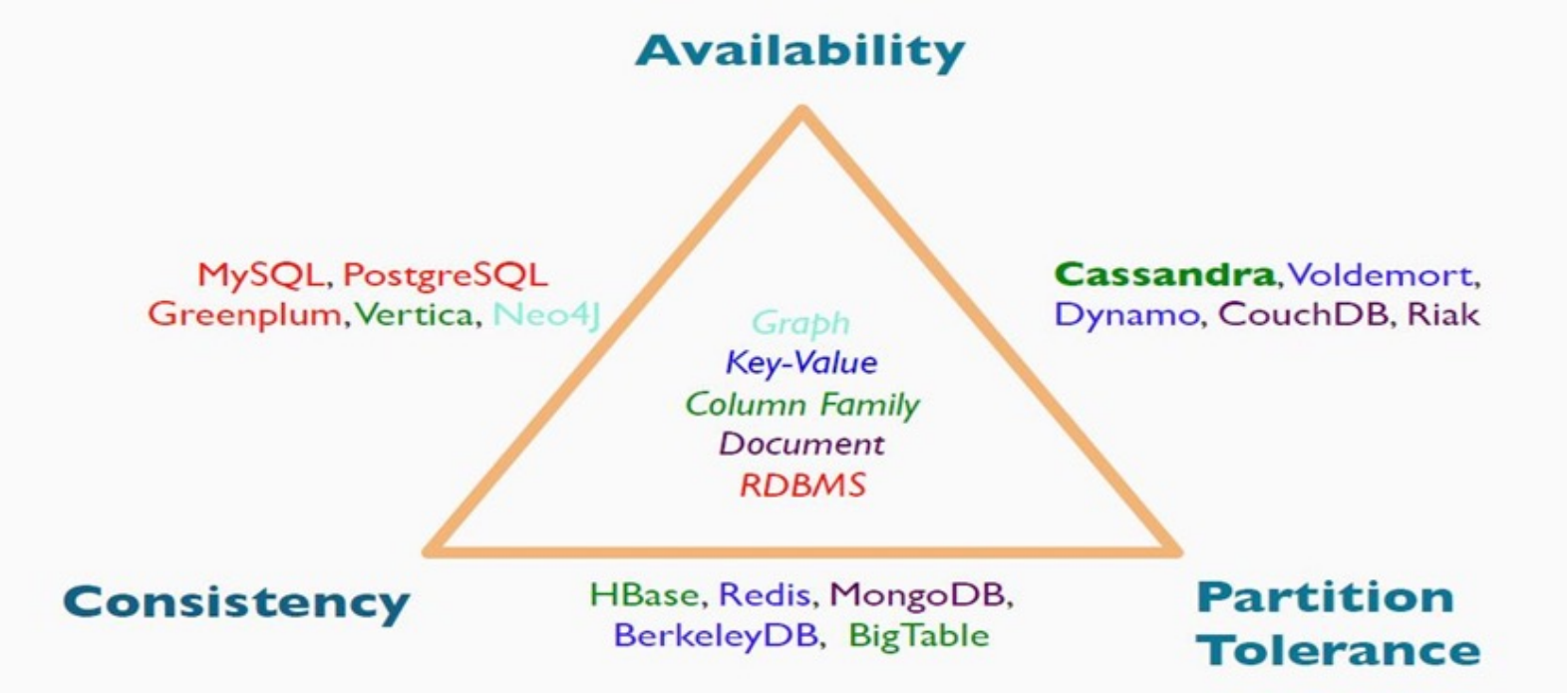
- **For example, a RDBMS can only guarantee CA**

Eventual consistency

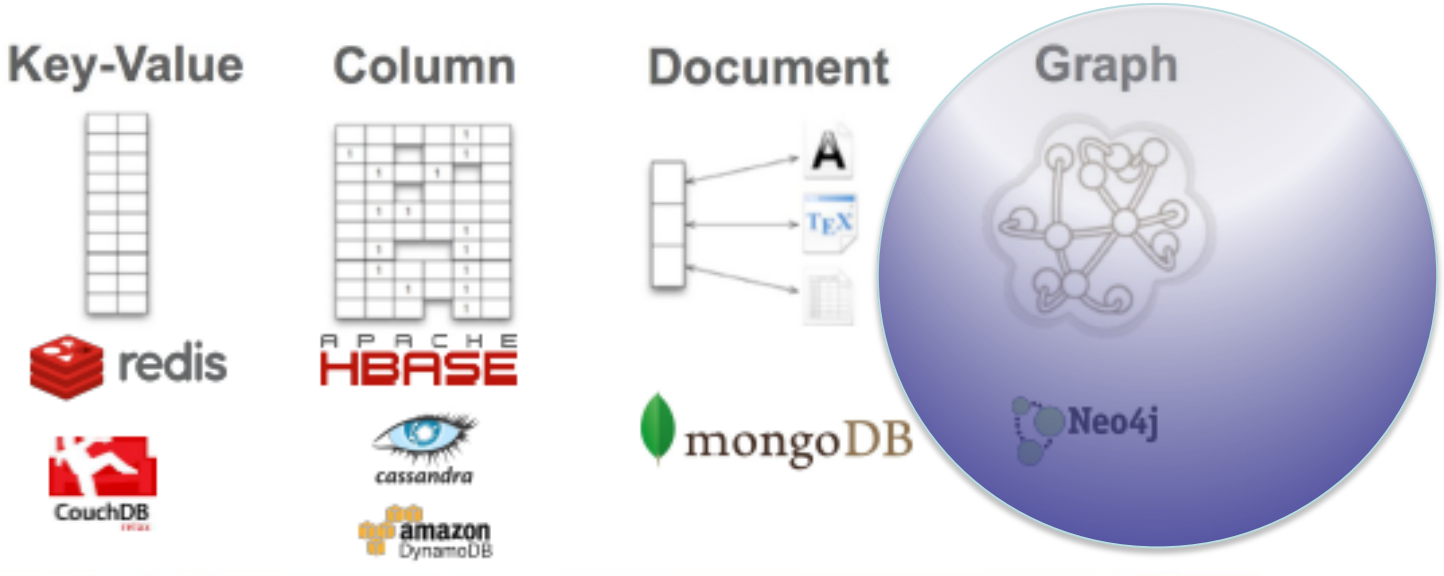
- **Assumption:** in the absence of new writes, **eventually** consistency will be achieved, and all replicas that are responsible for a data item will agree on the same version and return the last updated value.
- With fewer replicas, R/W operations complete more quickly, lowering latency.



CAP theorem

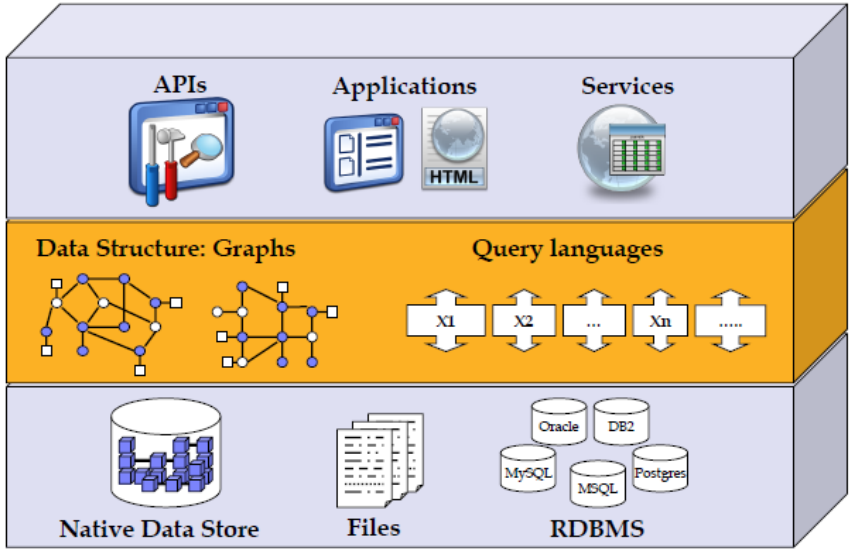

Availability
MySQL, PostgreSQL
Greenplum, Vertica, Neo4J
Cassandra, Voldemort,
Dynamo, CouchDB, Riak
Graph
Key-Value
Column Family
Document
RDBMS
Consistency
HBase, Redis, MongoDB,
BerkeleyDB, BigTable
Partition
Tolerance

In the remainder….



Introduction to Graph Databases

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

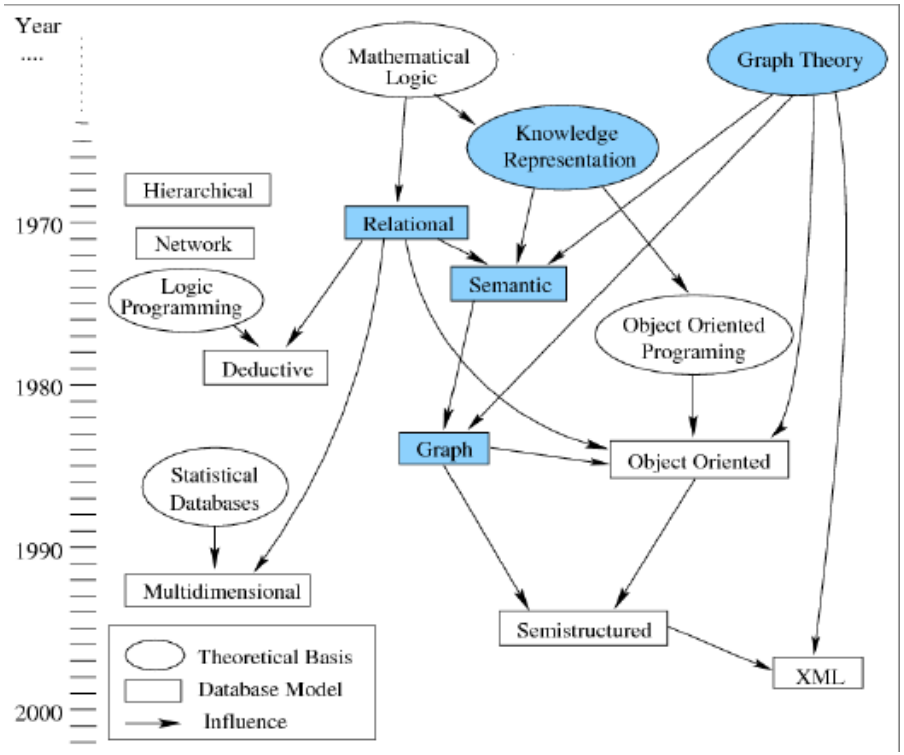
Enforce data consistency. Constraints can be grouped in schema-instance consistency, identity and referencial integrity, and functional and inclusion dependencies

Database models: (Codd)

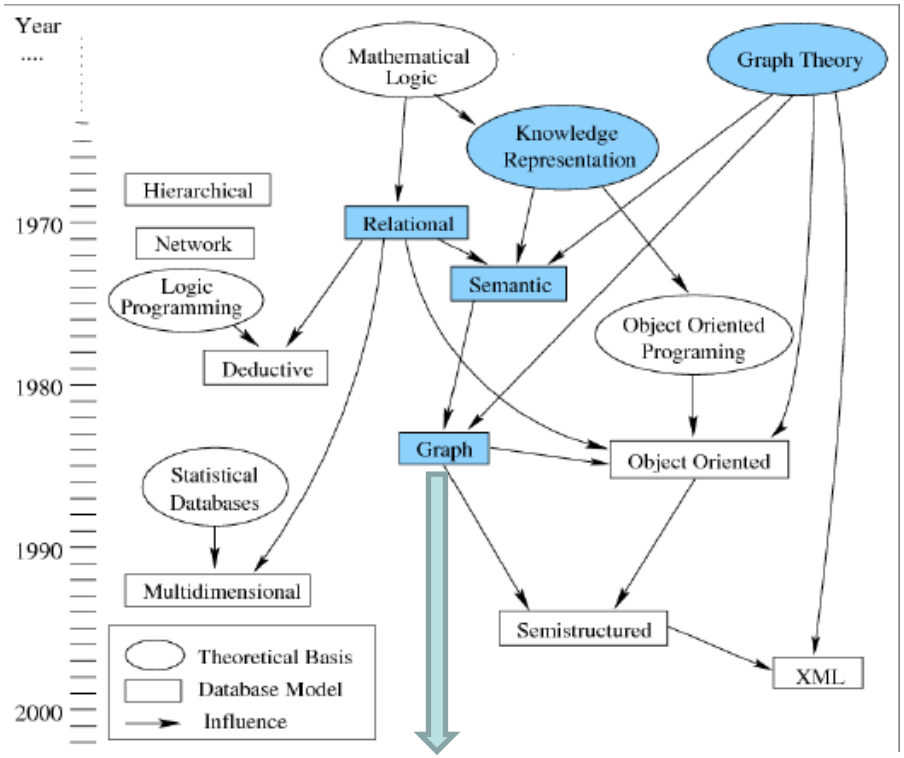
Query language

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.

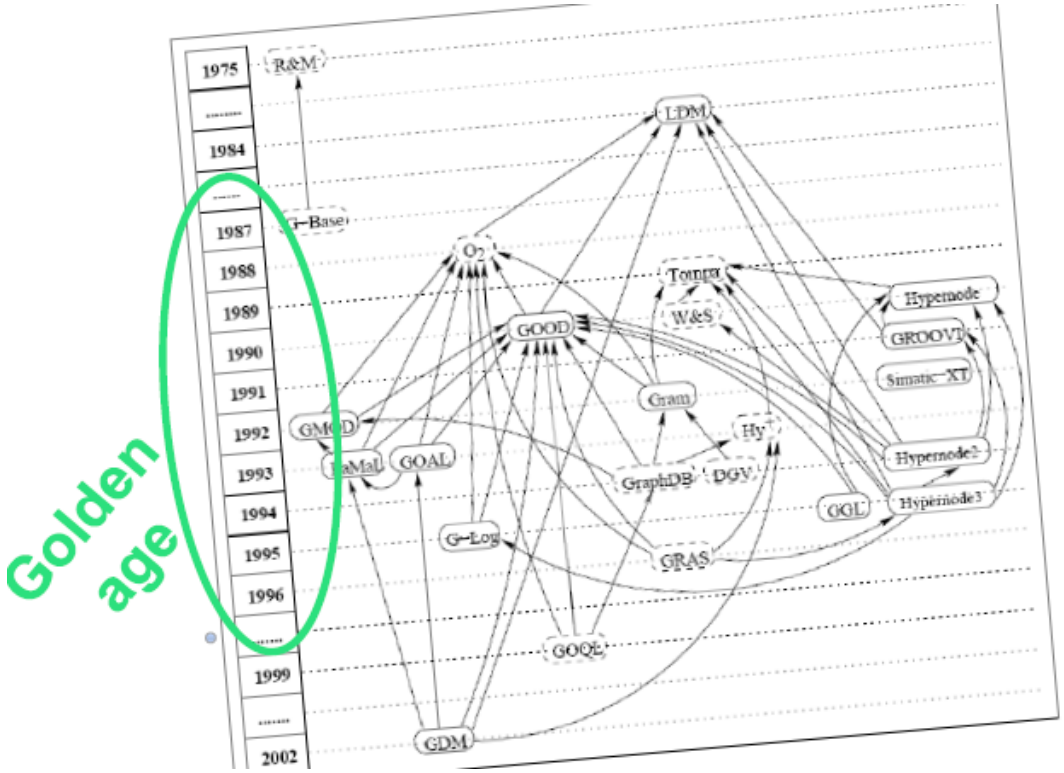
A history of database models (A. Mendelzon)



A history of database models (A. Mendelzon)

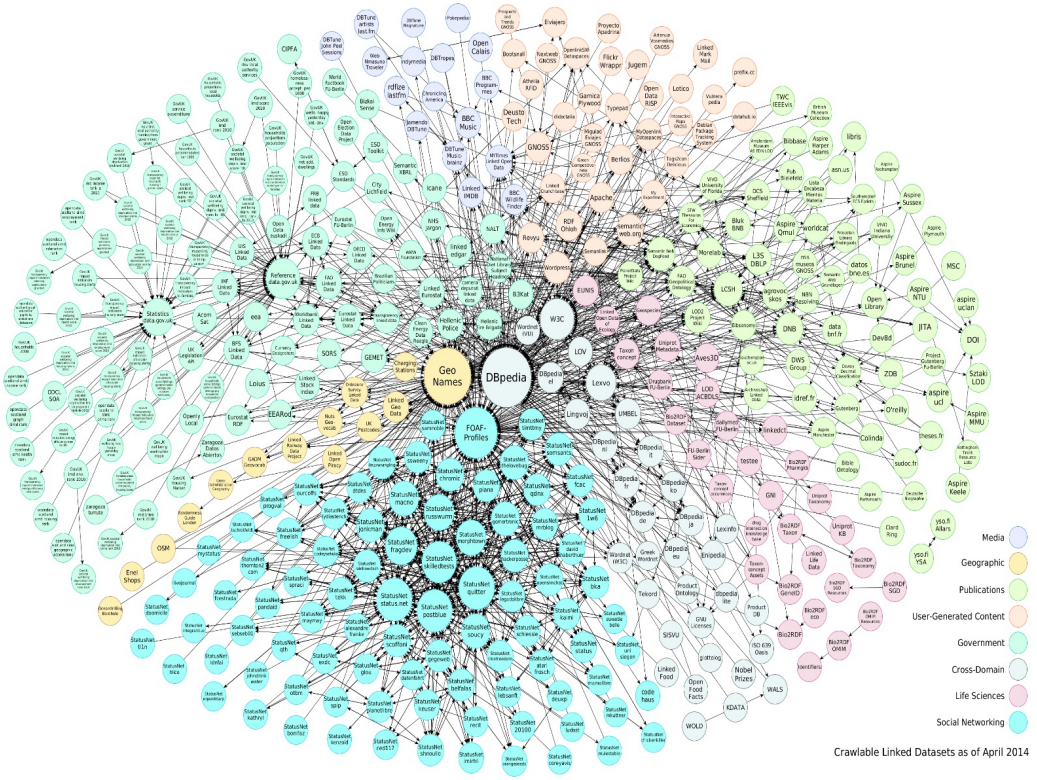


The Golden age of GDB



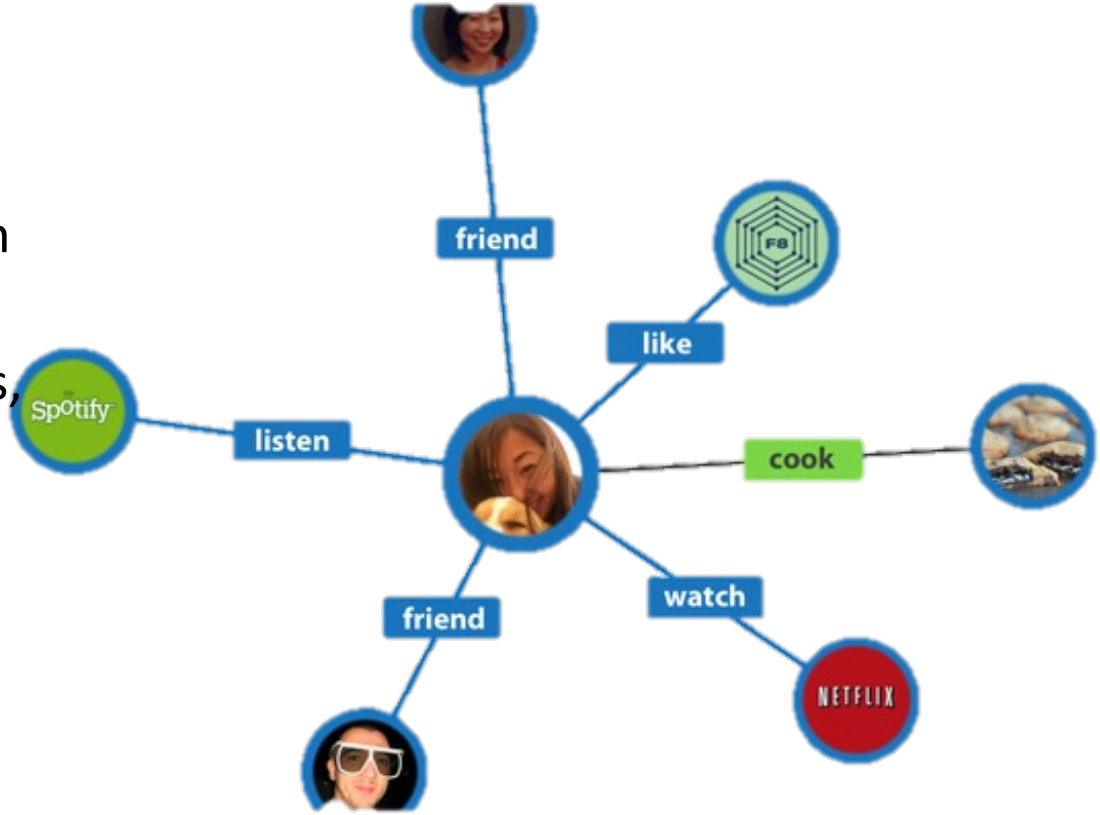
Data connectedness

- 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

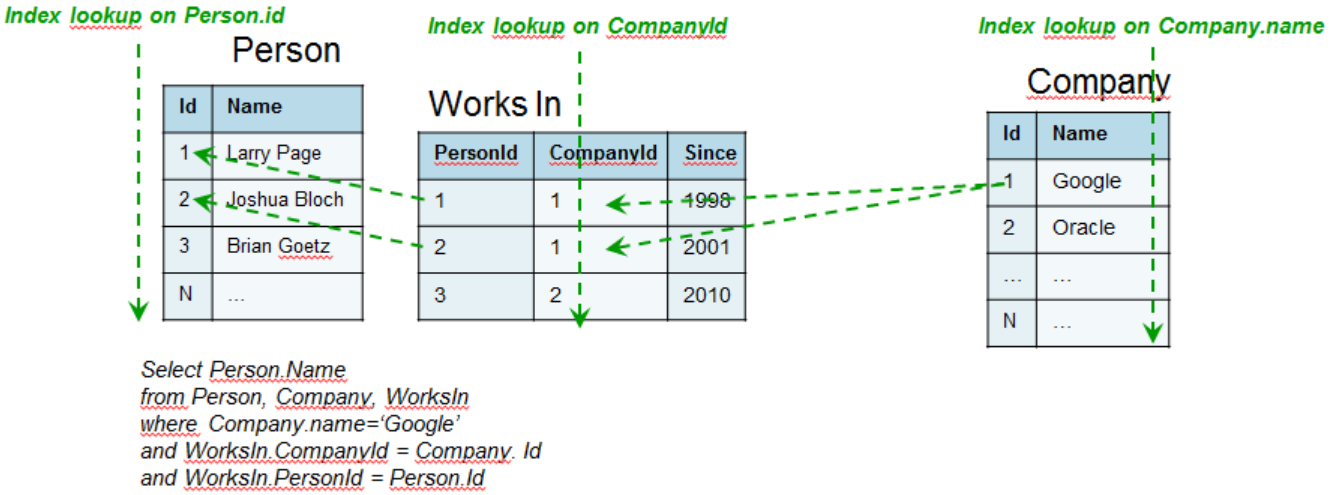
	# persons	query time
Relational database	1000	2000ms
Neo4j	1000	2ms

Social Network "path" performance

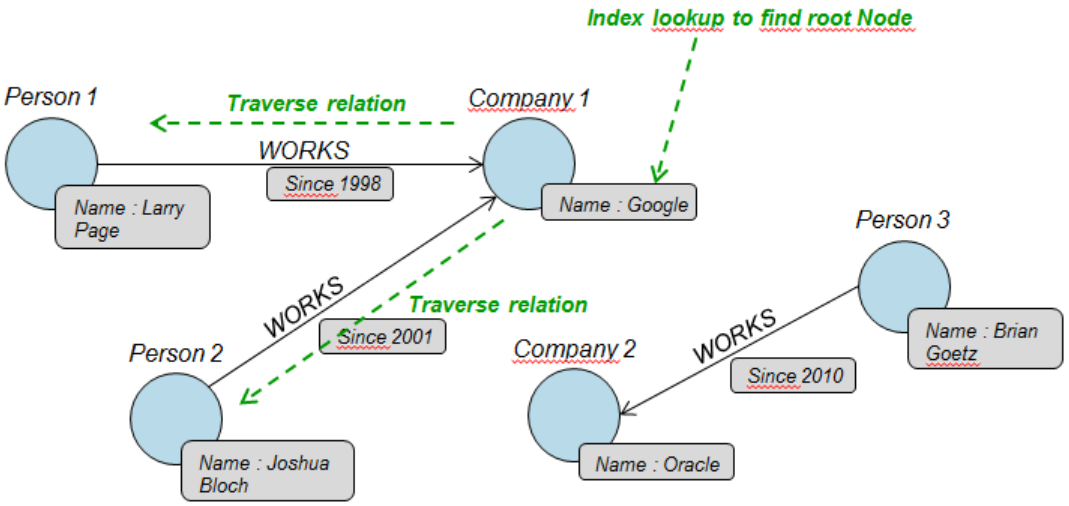
- Experiment:
  - ~1k persons
  - Average 50 friends per person
  - `pathExists(a,b)` limited to depth 4
  - Caches warm to eliminate disk IO

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

A typical SQL query



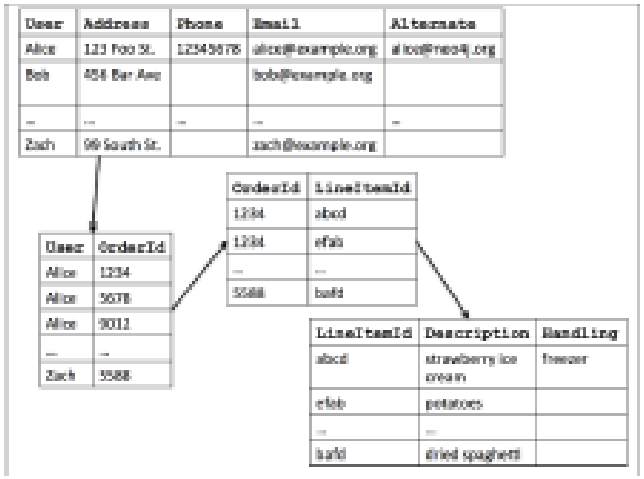
Same query on graphs



The deepest the navigation, the largest the difference with RDBs

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'
```

Cardinalities:

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

Query Cost?!

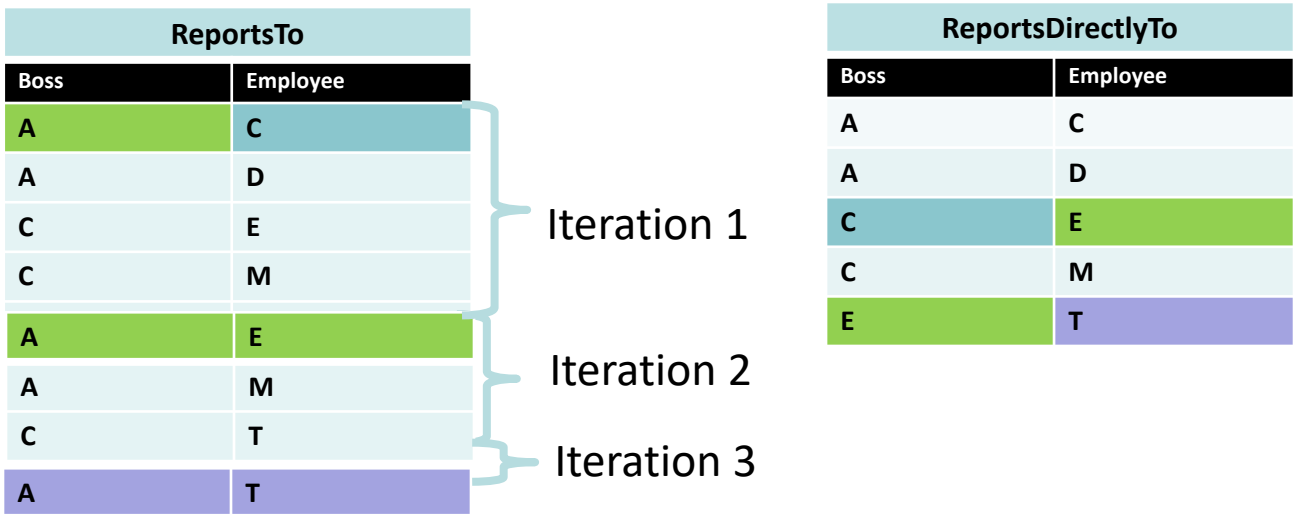
Traversing data in a GDB

Cardinalities:

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

Query Cost?!

O(N)



We want to compute the closure of the relation "ReportsDirectlyTo", that is, to whom someone "ReportsTo", either directly or indirectly. SQL supports these kinds of recursive queries. Recursively joining ReportsTo and ReportsDirectlyTo on RT.Employee=RDT.Boss.

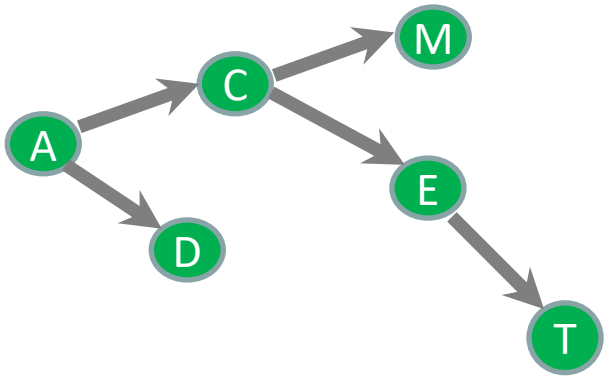
```
WITH recursive ReportsTo(Boss, Employee) AS
  (SELECT Boss, Employee
   FROM ReportsDirectlyTo
   UNION ALL
   SELECT ReportsTo.Boss, ReportsDirectlyTo.Employee
   FROM ReportsTo, ReportsDirectlyTo
   WHERE  ReportsTo.Employee = ReportsDirectlyTo.Boss )
SELECT * FROM ReportsTo
```

These queries are normally more expensive in the Relational Model, since they imply MULTIPLE JOINS.

Joins are expressed at the schema level rather than at the instance level.

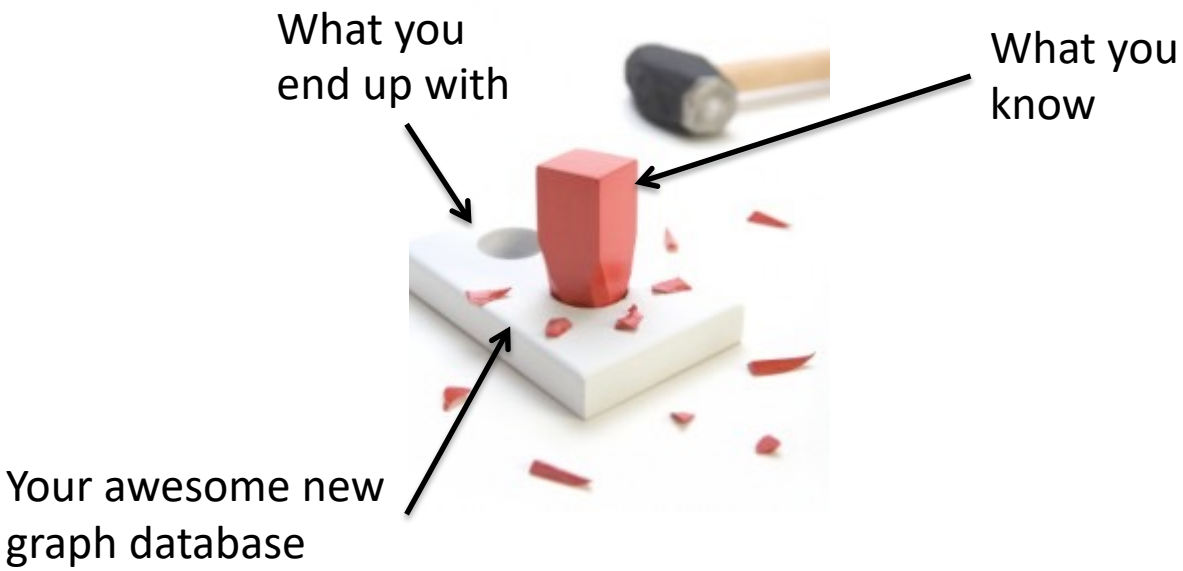
How would we represent this in a Graph Data Model?

ReportsDirectlyTo	
Boss	**Employee**
A	C
A	D
C	E
C	M
E	T



Paths in a graph are expressed at the instance level (there is no schema). Just check if there is an outgoing edge.

Forcing the relational model in a graph



Think in terms of nodes and edges!

Consequence: graph DB are **hot** again

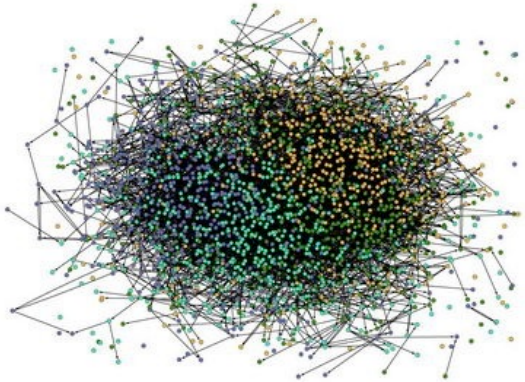
- The question is: how to process HUGE graphs?
- We now have hardware that can do this

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

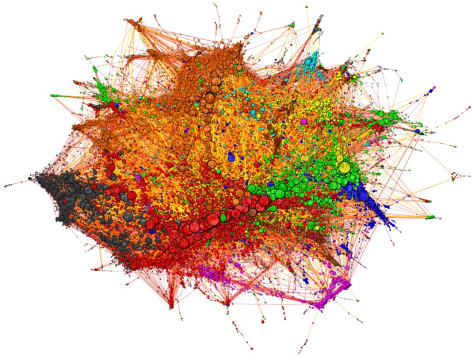
Open questions (almost all of the stuff)

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

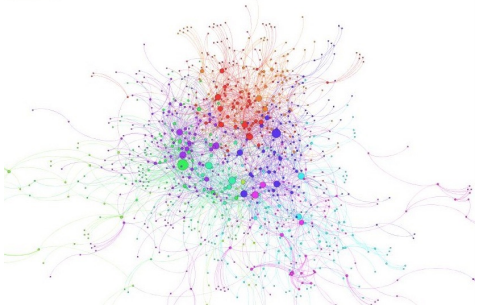
Use cases: social networks



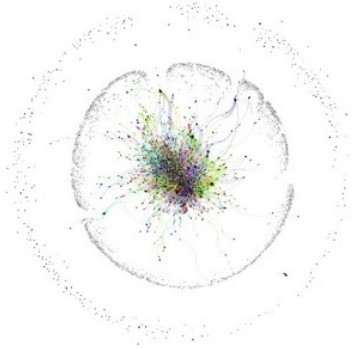
Network of Friends in a High School



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



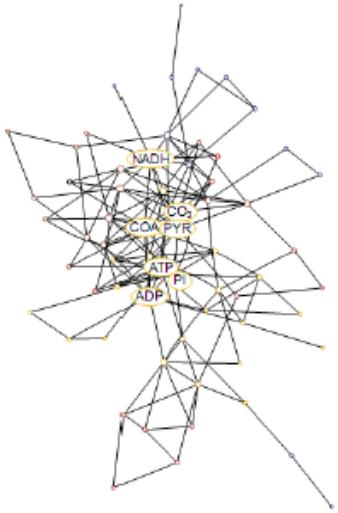
Network structure of music genres and their stylistic origin
http://www.infosysblogs.com/web2/2013/01/network_structure_of_music_gen.html



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

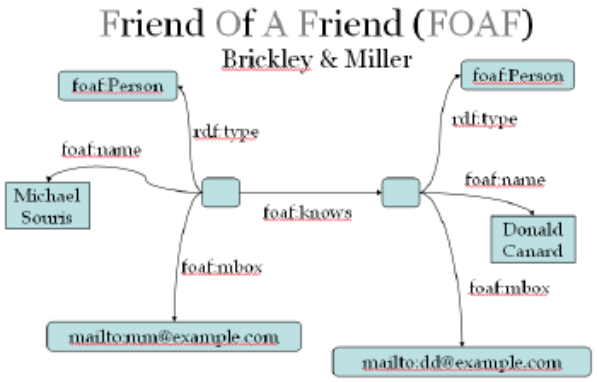
Use cases: biology

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

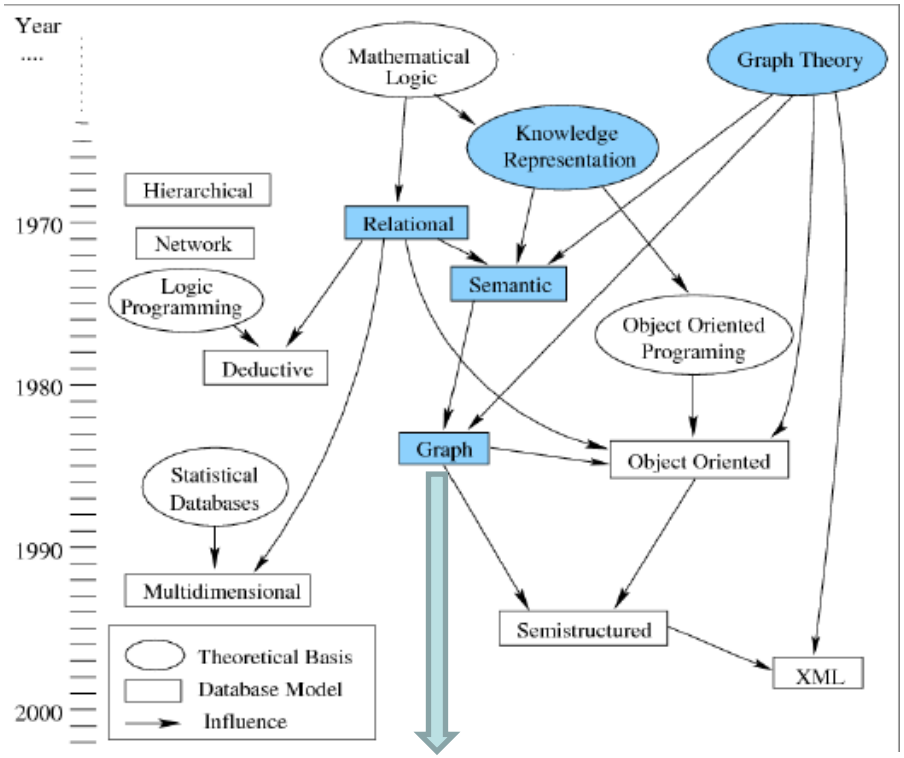


Use cases: the web

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 Erdös 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

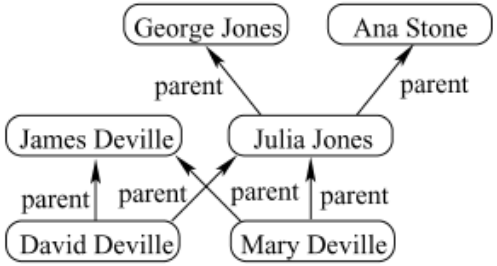


Graph database models

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



RDF Model

- RDF allows to express facts: Ana is the mother of Julia
- But we'd like to be able to express more generic knowledge
  - Mothers are female
  - If somebody has a daughter then that person is a parent
- This kind of knowledge is often called *schema* knowledge or *terminological* knowledge.
- RDF Schema allows us to do some schema knowledge modeling, OWL gives even more expressivity

RDF classes & instances

- Classes stand for sets of things. In RDF: Sets of URIs.
- book:uri is a member of the class ex:Textbook

```
book:uri    rdf:type    ex:Textbook .
```

- An URI can belong to several classes

```
book:uri    rdf:type    ex:Textbook .
book:uri    rdf:type    ex:WorthReading .
```

- Classes can be arranged in hierarchies: each textbook is a book

```
ex:Textbook    rdfs:subClassOf    ex:Book .
```

RDF pre-defined classes

- Every URI denoting a class is a member of rdfs:Class

```
ex:Textbook   rdf:type   rdfs:Class .
```

- This also makes rdfs:Class a member of rdfs:Class

```
rdfs:Class   rdf:type   rdfs:Class .
```

- rdfs:Resource (class of all URIs)
- rdf:Property (class of all properties)
- rdf:XMLLiteral
- rdfs:Literal (each datatype is a subclass)
- rdf:Bag, rdf:Alt, rdf:Seq, rdfs:Container , rdf:List, rdf:nil, rdfs:ContainerMembershipProperty
- rdfs:Datatype (contains all datatypes – a class of classes)
- rdf:Statement

RDF implicit knowledge

If an RDFS document contains

```
u    rdf:type    ex:Textbook .
```

and

```
ex:Textbook    rdfs:subClassOf    ex:Book .
```

then

```
u    rdf:type    ex:Book .
```

is *implicitly* also the case: it's a *logical consequence*. (We can also say it is *deduced* (deduction) or *inferred* (inference). We do not have to state this explicitly. Which statements are logical consequences is governed by the formal semantics.

RDF implicit knowledge (cont.)

As another example, from

```
ex:Textbook   rdfs:subClassOf   ex:Book .
```

and

```
ex:Book       rdfs:subClassOf   ex:PrintMedia .
```

then

```
ex:Textbook   rdfs:subClassOf   ex:PrintMedia .
```

I.e. rdfs:subClassOf is *transitive*.

RDF implicit knowledge (cont.)

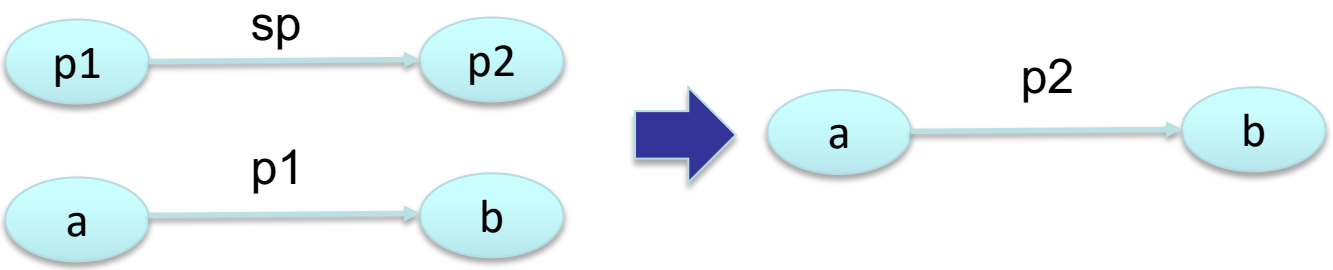
Property hierarchies

From
```
ex:isHappilyMarriedTo  rdf:subPropertyOf ex:isMarriedTo.
```

and
```
ex:markus   ex:isHappilyMarriedTo   ex:anja .
```

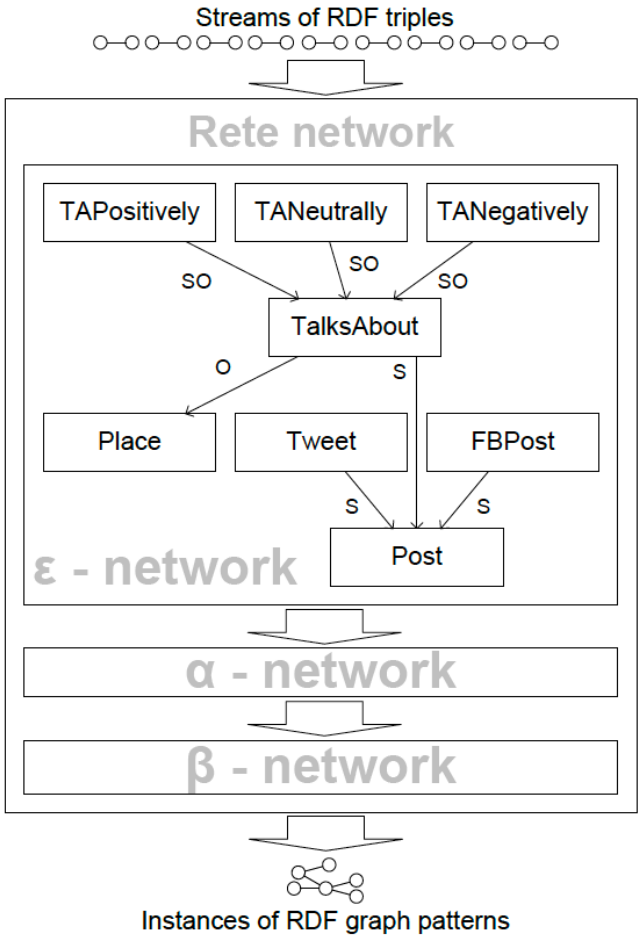
then
```
ex:markus   ex:isMarriedTo   ex:anja .
```

Thus, if p1 is subproperty of p2, and a p1 b, then a p2 b

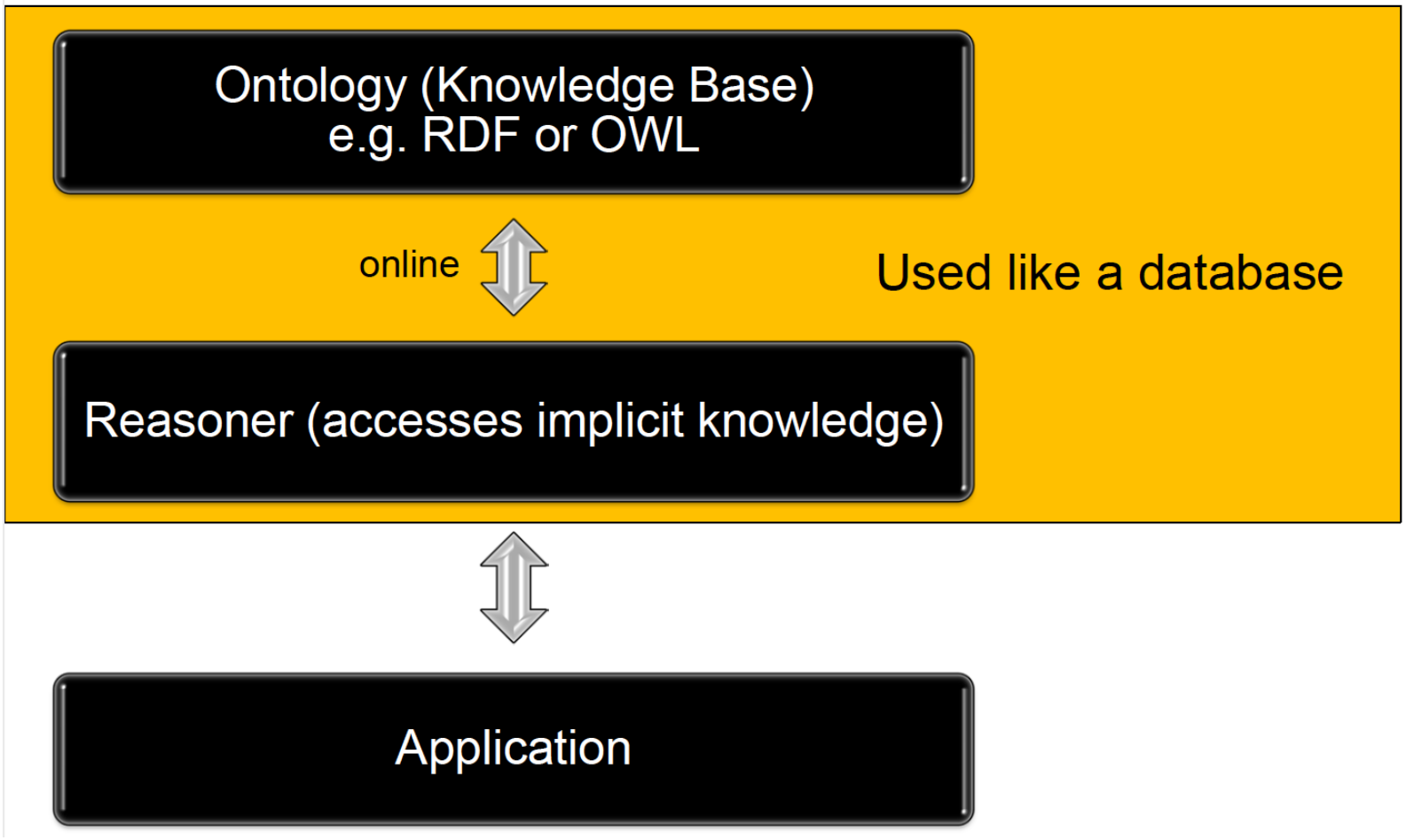


RDF model classes & properties

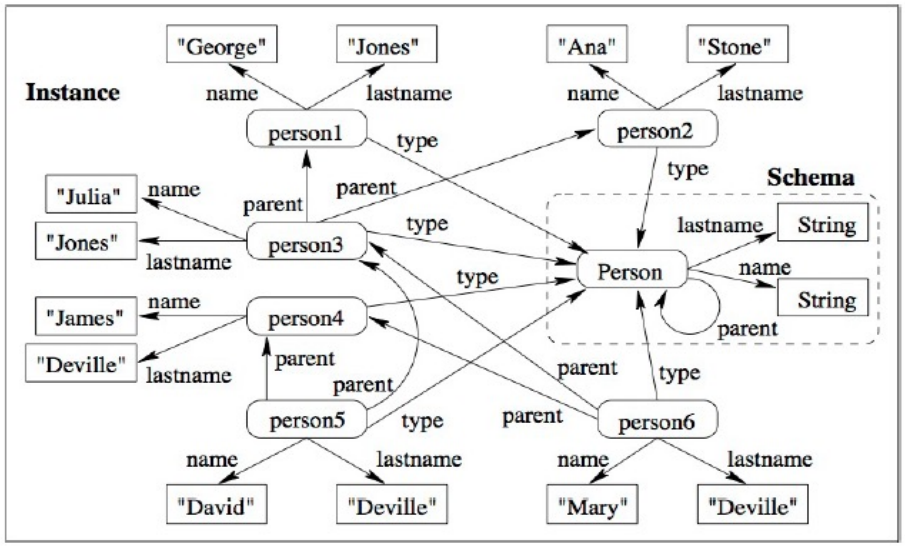
```
@prefix rdf:
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs:
  <http://www.w3.org/2000/01/rdf-schema#>.
@prefix :
  <http://example.org/ns#>

:Post           rdf:type            rdfs:Class.

:Tweet          rdf:type            rdfs:Class;
                rdfs:subClassOf     :Post.

:FBPost         rdf:type            rdfs:Class;
                rdfs:subClassOf     :Post.

:Place          rdf:type            rdfs:Class.

:TalksAbout     rdf:type            rdf:Property;
                rdfs:domain         :Post;
                rdfs:range          :Place.

:TAPositively   rdf:type            rdf:Property;
                rdfs:subPropertyOf  :TalksAbout.

:TANeutrally    rdf:type            rdf:Property;
                rdfs:subPropertyOf  :TalksAbout.

:TANegatively   rdf:type            rdf:Property;
                rdfs:subPropertyOf  :TalksAbout.
```



Using implicit knowledge

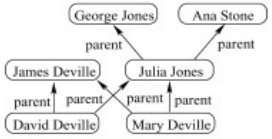


Genealogy – RDF model



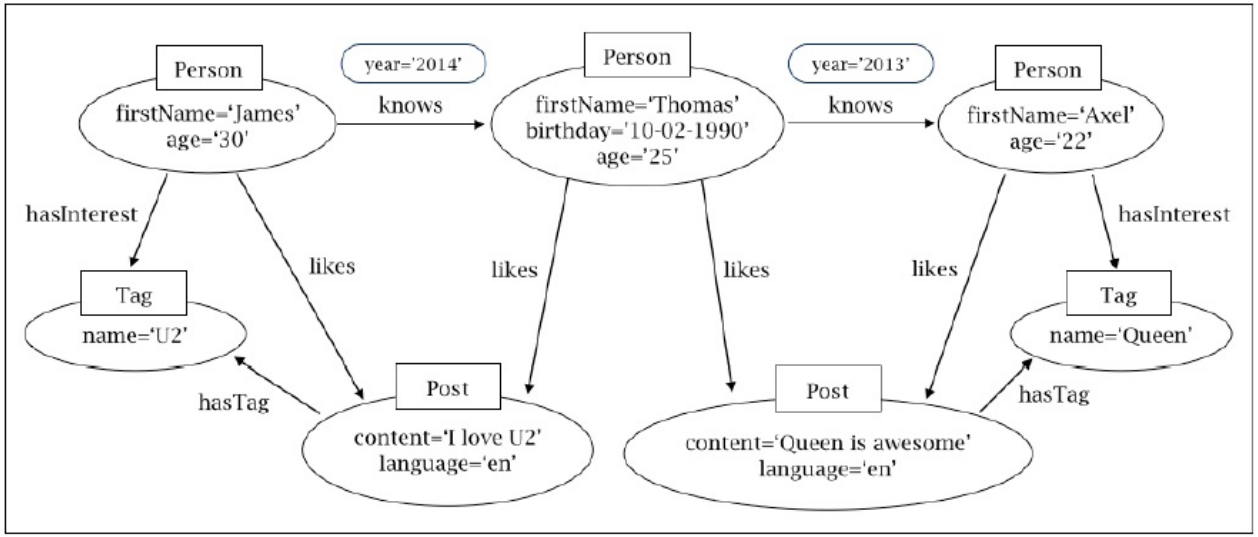
NAME	LASTNAME
George	Jones
Ana	Stone
Julia	Jones
James	Deville
David	Deville
Mary	Deville

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.

The property graph data model



Next, we get into GRAPH DB's