CC5212-1
Procesamiento Masivo de Datos
Otoño 2016

Lecture 11: NoSQL II

Aidan Hogan
aidhog@gmail.com

RECAP: NOSQL

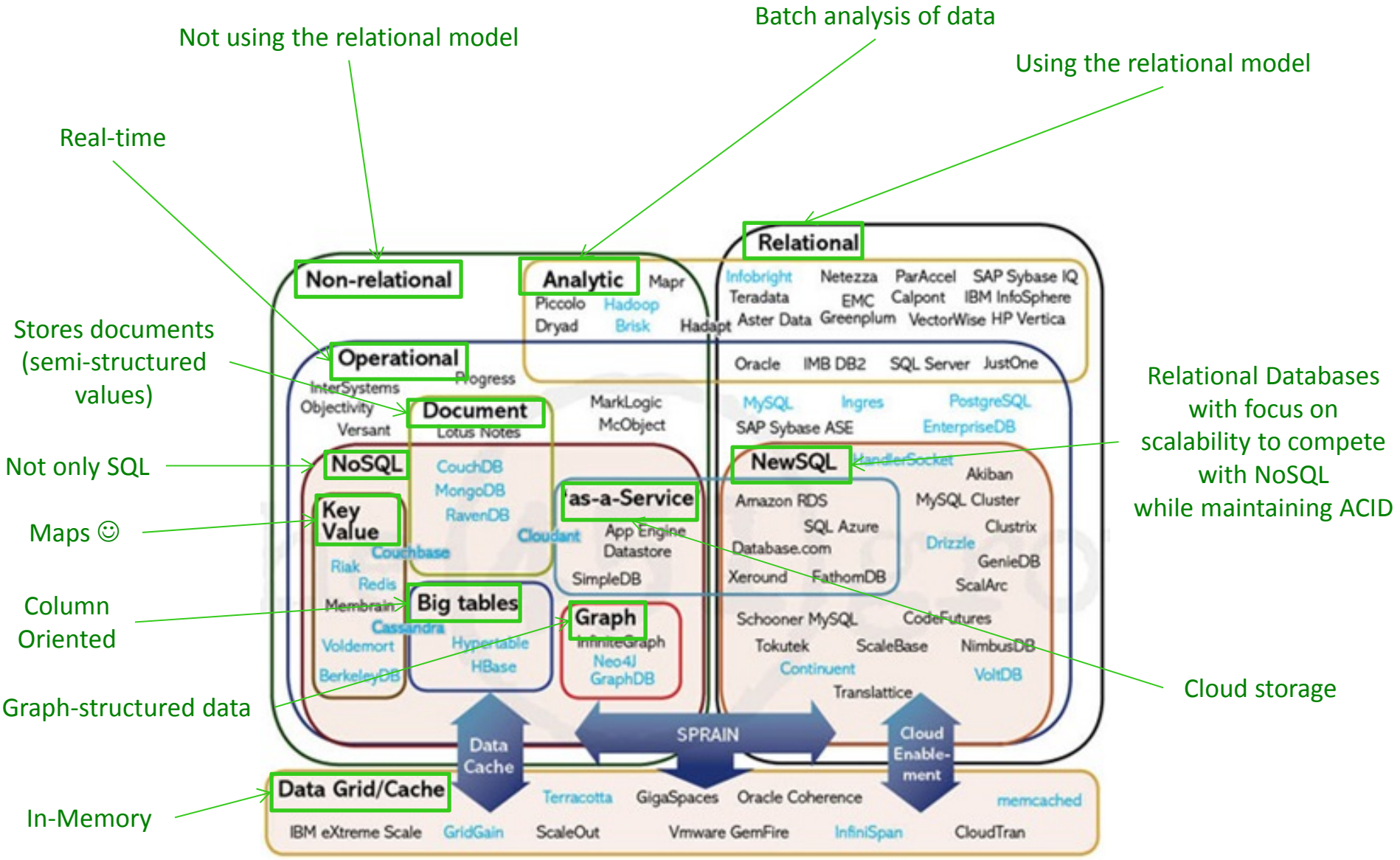
NoSQL

NoSQL vs. Relational Databases

What are the big differences between relational databases and NoSQL systems?

What are the trade-offs?

The Database Landscape



RECAP: KEY–VALUE

Key–Value = a Distributed Map

Key	Value
country:Afghanistan	capital@city:Kabul,continent:Asia,pop:31108077#2011
country:Albania	capital@city:Tirana,continent:Europe,pop:3011405#2013
…	…
city:Kabul	country:Afghanistan,pop:3476000#2013
city:Tirana	country:Albania,pop:3011405#2013
…	…
user:10239	basedIn@city:Tirana,post:{103,10430,201}
…	…

Amazon Dynamo(DB): Model

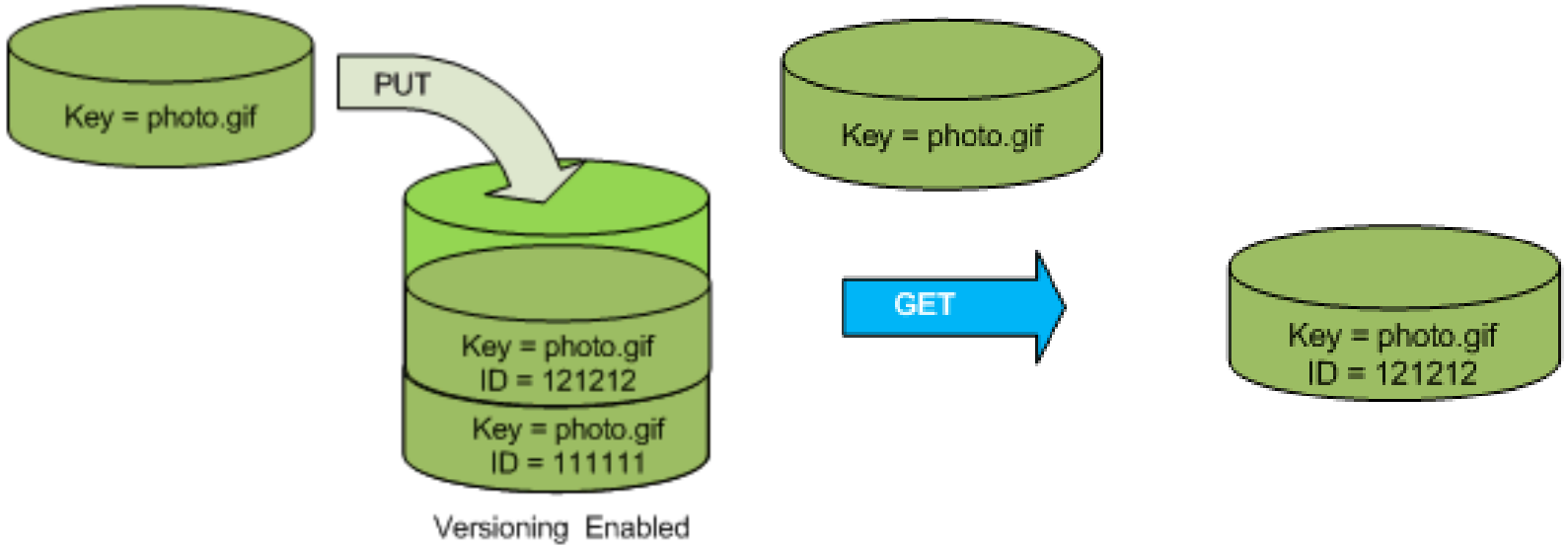
- Named table with primary key and a value

Countries	
Primary Key	**Value**
Afghanistan	capital:Kabul,continent:Asia,pop:31108077#2011
Albania	capital:Tirana,continent:Europe,pop:3011405#2013
…	…

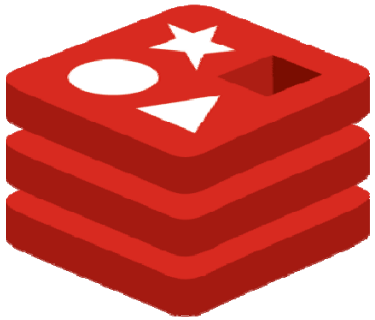
Cities	
Primary Key	**Value**
Kabul	country:Afghanistan,pop:3476000#2013
Tirana	country:Albania,pop:3011405#2013
…	…

Amazon Dynamo(DB): Object Versioning

- Object Versioning (per bucket)
  - PUT doesn't overwrite: pushes version
  - GET returns most recent version



Other Key–Value Stores



redis

RECAP: DOCUMENT STORES

Key–Value Stores: Values are Documents

Key	Value
country:Afghanistan	<country> <capital>city:Kabul</capital> <continent>Asia</continent> <population> <value>31108077</value> <year>2011</year> </population> </country>
…	…

- Document-type depends on store
  - XML, JSON, Blobs, Natural language
- Operators for documents
  - e.g., filtering, inv. indexing, XML/JSON querying, etc.

MongoDB: JSON Based

Key	Value (Document)
6ads786a5a9	{ "_id" : **ObjectId**("6ads786a5a9") , "name" : "Afghanistan" , "capital": "Kabul" , "continent" : "Asia" , "population" : { "value" : 31108077, "year" : 2011 } }
…	…

- Can invoke Javascript over the JSON objects
- Document fields can be indexed

```
db.inventory.find({ continent: { $in: [ 'Asia', 'Europe' ]}})
```

Document Stores

TABLULAR / COLUMN FAMILY

Key–Value = a Distributed Map

Countries	
Primary Key	**Value**
Afghanistan	capital:Kabul,continent:Asia,pop:31108077#2011
Albania	capital:Tirana,continent:Europe,pop:3011405#2013
…	…

Tabular = Multi-dimensional Maps

Countries				
Primary Key	**capital**	**continent**	**pop-value**	**pop-year**
Afghanistan	Kabul	Asia	31108077	2011
Albania	Tirana	Europe	3011405	2013
…	…	…	…	…

Bigtable: The Original Whitepaper

Why did they write another paper?
MapReduce solves everything, right?

MapReduce authors

Bigtable: A Distributed Storage System for Structured Data

Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach
Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber
{fay,jeff,sanjay,wilsonh,kerr,m3b,tushar,fikes,gruber}@google.com
Google, Inc.

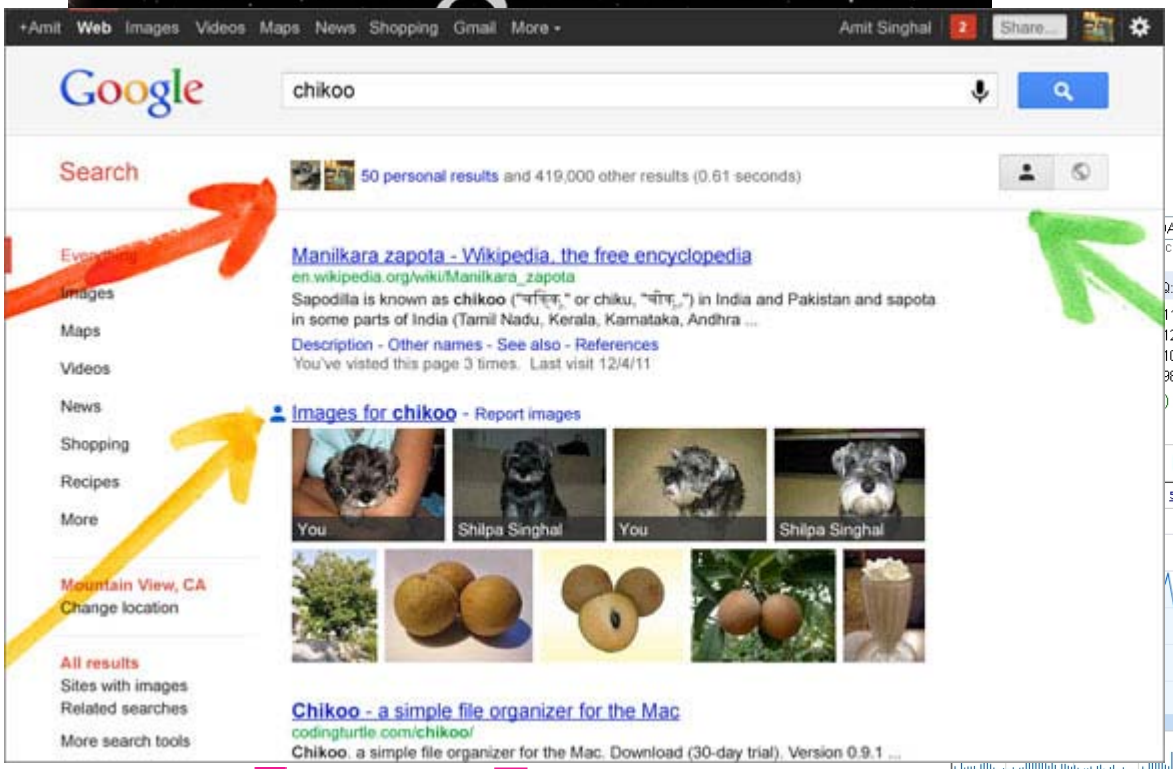
Abstract

Bigtable is a distributed storage system for managing structured data that is designed to scale to a very large size: petabytes of data across thousands of commodity servers. Many projects at Google store data in Bigtable, including w... Google Earth, and Google Finance. T...se applications place very different demands on Big...le, both in terms of data size (from URLs to web p...s to satellite imagery) and ...tency ...iremen... (from ...ckend bulk processing to ...l-time da... ...rvi...). Despi...ese varied demands, B...ble has su...ss... provid... flexible, high-... ...orm... ...e solution... r a... these Go... products. In ... pape... descri...e the sim... ple data model...p... ...d by Bigtable, which gives clients dynamic control over data layout and format, and we describe the design and implementation of Bigtable.

achieved scalability and high performance, but Bigtable provides a different interface than such systems. Bigtable does not support a full relational data model; instead, it provides clients with a simple data model that supports dynamic control over data layout and format, and allows clients to reason abou...ocality properties of the data represented in the und...ing storage. Data is indexed using row and colum...mes that can be arbitrary s...gs. Bigtab...ats ... as uni...p...d strings, alth...h c...nts oft...eria... vari...s form...f struc-tured ...d ...i-struct...d da...nto ...rings. Clients can ...tro... loc...ty of ...eir ... through careful choi...s in their ...mas. P...lly, B...ble schema parameters let clien...amically control ...er to serve data out ...emory d...om disk.

Sectio... describes t...e data model in more detail, and Section 3 p... ...overview of the client API. Sec-

Bigtable used for …



orkut by Google

• • •

Bigtable: Data Model

"*a sparse, distributed, persistent, multi-dimensional, sorted map*."

- *sparse*: not all values form a dense square
- *distributed*: lots of machines
- *persistent*: disk storage (GFS)
- *multi-dimensional*: values with columns
- *sorted*: sorting lexicographically by row key
- *map*: look up a key, get a value

Bigtable: in a nutshell

(row, column, time) → value

- row: a row id string
  - e.g., "`Afganistan`"
- column: a column name string
  - e.g., "`pop-value`"
- time: an integer (64-bit) version time-stamp
  - e.g., `18545664`
- value: the element of the cell
  - e.g., "`31120978`"

Bigtable: in a nutshell

(row, column, time) → value

(`Afganistan`, `pop-value`, t_4) → `31108077`

Primary Key	capital		continent		pop-value		pop-year	
Afghanistan	t_1	Kabul	t_1	Asia	t_1	31143292	t_1	2009
					t_2	31120978		
					t_4	31108077	t_4	2011
Albania	t_1	Tirana	t_1	Europe	t_1	2912380	t_1	2010
					t_3	3011405	t_3	2013
…	…		…			…		…

Bigtable: Sorted Keys

SORTED

Primary Key	capital		pop-value		pop-year	
Asia:Afghanistan	t_1	Kabul	t_1	31143292	t_1	2009
			t_2	31120978		
			t_4	31108077	t_4	2011
Asia:Azerbaijan	…	…	…	…	…	…
…	…	…	…	…	…	…
Europe:Albania	t_1	Tirana	t_1	2912380	t_1	2010
			t_3	3011405	t_3	2013
Europe:Andorra	…	…	…	…	…	…
…	…	…	…	…	…	…

Benefits of sorted keys vs. hashed keys?

Bigtable: Tablets

Primary Key	capital		pop-value		pop-year	
Asia:Afghanistan	t_1	Kabul	t_1	31143292	t_1	2009
			t_2	31120978		
			t_4	31108077	t_4	2011
Asia:Azerbaijan	…	…	…	…	…	…
…	…	…	…	…	…	…
Europe:Albania	t_1	Tirana	t_1	2912380	t_1	2010
			t_3	3011405	t_3	2013
Europe:Andorra	…	…	…	…	…	…
…	…	…	…	…	…	…

Take advantage of locality of processing!

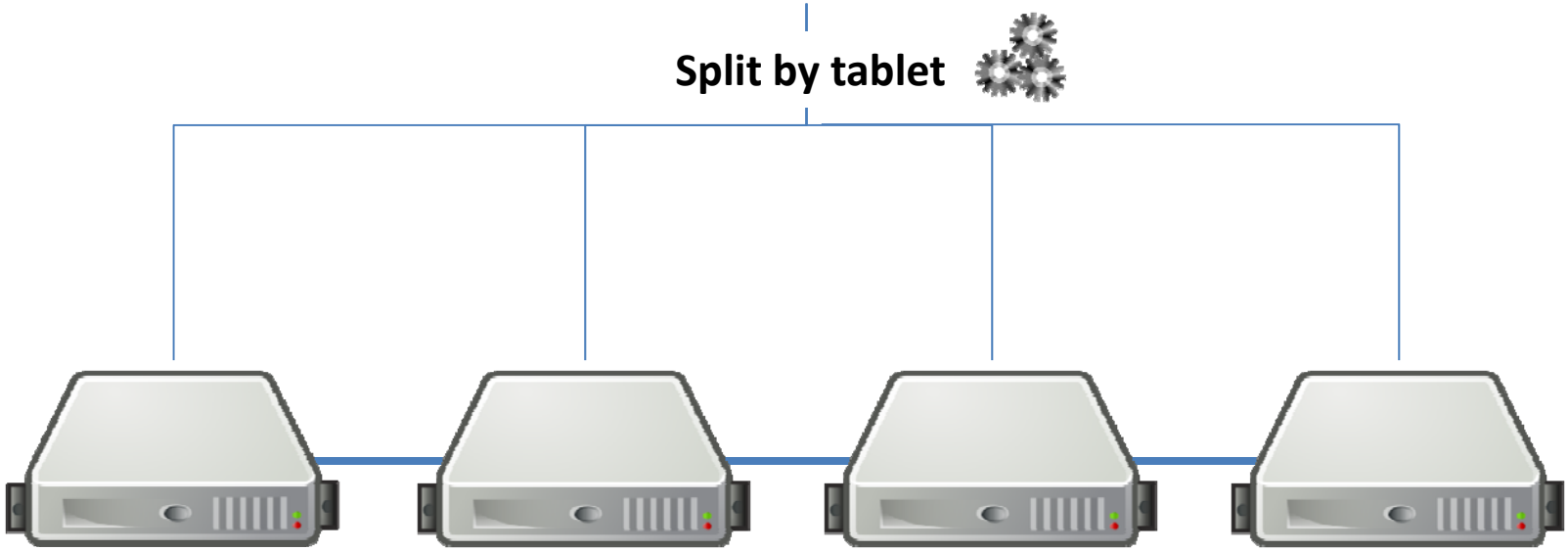
A real-world example of locality/sorting

Primary Key	language		title		links	
com.imdb	t_1	en	t_1	IMDb Home	t_1	…
			t_2	IMDB - Movies		
			t_4	IMDb	t_4	…
com.imdb/title/tt2724064/	t_1	en	t_2	Sharknado	t_2	…
com.imdb/title/tt3062074/	t_1	en	t_2	Sharknado II	t_2	
…	…	…	…	…	…	…
org.wikipedia	t_1	multi	t_1	Wikipedia	t_1	…
			t_3	Wikipedia Home	t_3	…
org.wikipedia.ace	t_1	ace	t_1	Wikipèdia bahsa Acèh	…	…
…	…	…	…	…	…	…

Bigtable: Distribution

Asia:Afghanistan	t_1	Kabul	t_1	31143292	t_1	2009
			t_2	31120978		
			t_4	31108077	t_4	2011
Asia:Azerbaijan	...	...	...	...	...	...
...	...	...	...	...	...	...
Europe:Albania	t_1	Tirana	t_1	2912380	t_1	2010
			t_3	3011405	t_3	2013
Europe:Andorra	...	...	...	...	...	...
...	...	...	...	...	...	...

Pros and cons versus hash partitioning?



***Horizontal** range partitioning*

Bigtable: Column Families

Primary Key	pol:capital		demo:pop-value		demo:pop-year	
Asia:Afghanistan	t_1	Kabul	t_1	31143292	t_1	2009
			t_2	31120978		
			t_4	31108077	t_4	2011
Asia:Azerbaijan	...	...	...	...	...	...
...	...	...	...	...	...	...
Europe:Albania	t_1	Tirana	t_1	2912380	t_1	2010
			t_3	3011405	t_3	2013
Europe:Andorra	...	...	...	...	...	...
...	...	...	...	...	...	...

- Group logically similar columns together
  - Accessed efficiently together
  - Access-control and storage: column family level
  - If of same type, can be compressed

Bigtable: Versioning

- Similar to Apache Dynamo
  - Cell-level
  - 64-bit integer time stamps
  - Inserts push down current version
  - Lazy deletions / periodic garbage collection
  - Two options:
    - keep last *n* versions
    - keep versions newer than *t* time

Bigtable: SSTable Map Implementation

- 64k blocks (default) with index in footer (GFS)
- Index loaded into memory, allows for seeks
- Can be split or merged, as needed

How to handle writes?

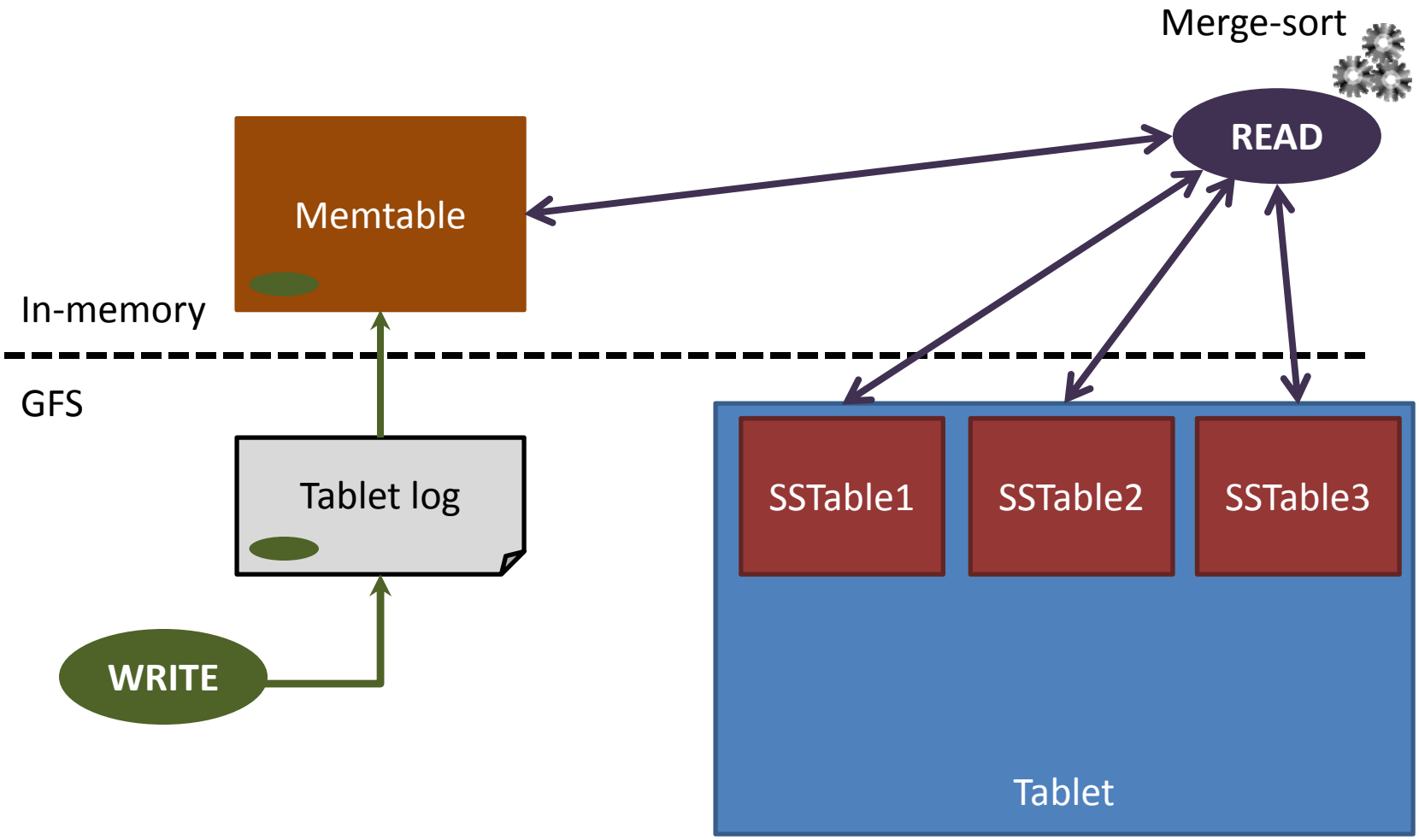
	Primary Key	pol:capital		demo:pop-value		demo:pop-year	
0	Asia:Afghanistan	t_1	Kabul	t_1	31143292	t_1	2009
				t_2	31120978		
				t_4	31108077	t_4	2011
	Asia:Azerbaijan	...	...	...	...	...	...
	...	...	...	...	...	...	...
65536	Asia:Japan	...	...	...	...	...	...
	Asia:Jordan	...	...	...	...	...	...
	...	...	...	...	...	...	...

Index:

Block 0 / Offset 0 / Asia:Afghanistan
Block 1 / Offset 65536 / Asia: Japan

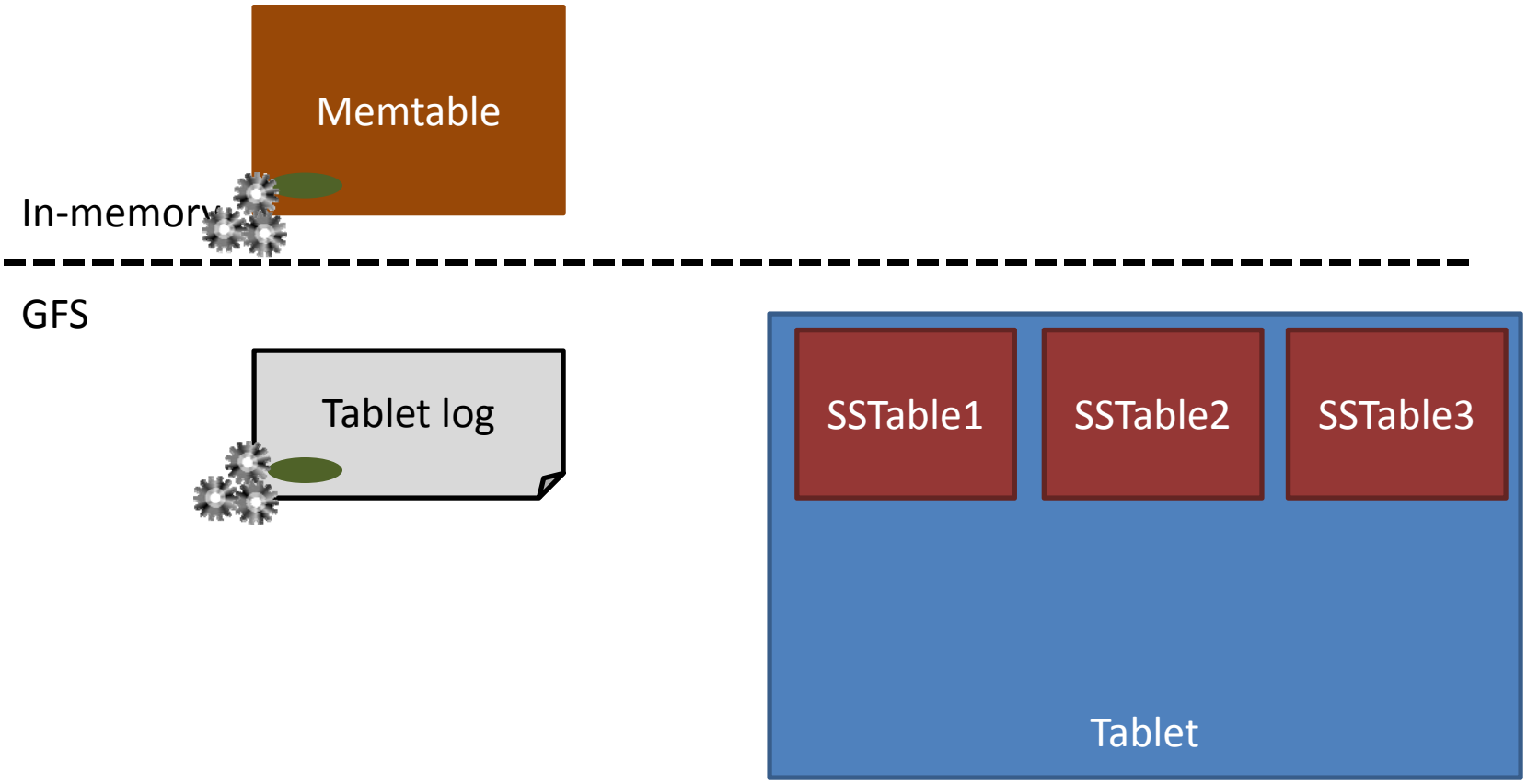
Bigtable: Buffered/Batched Writes

What's the danger here?



Bigtable: Redo Log

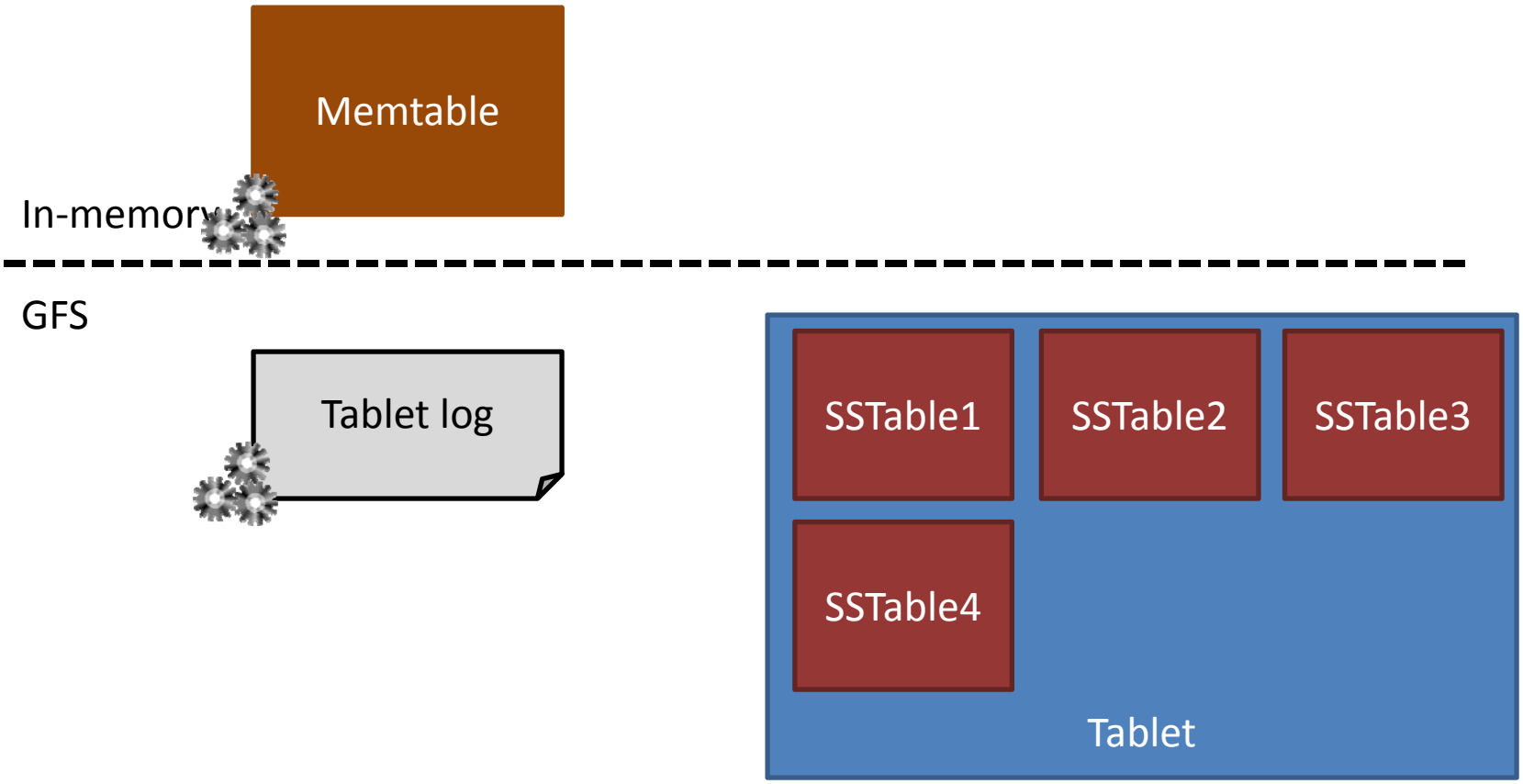
- If machine fails, Memtable redone from log



Bigtable: **Minor** Compaction

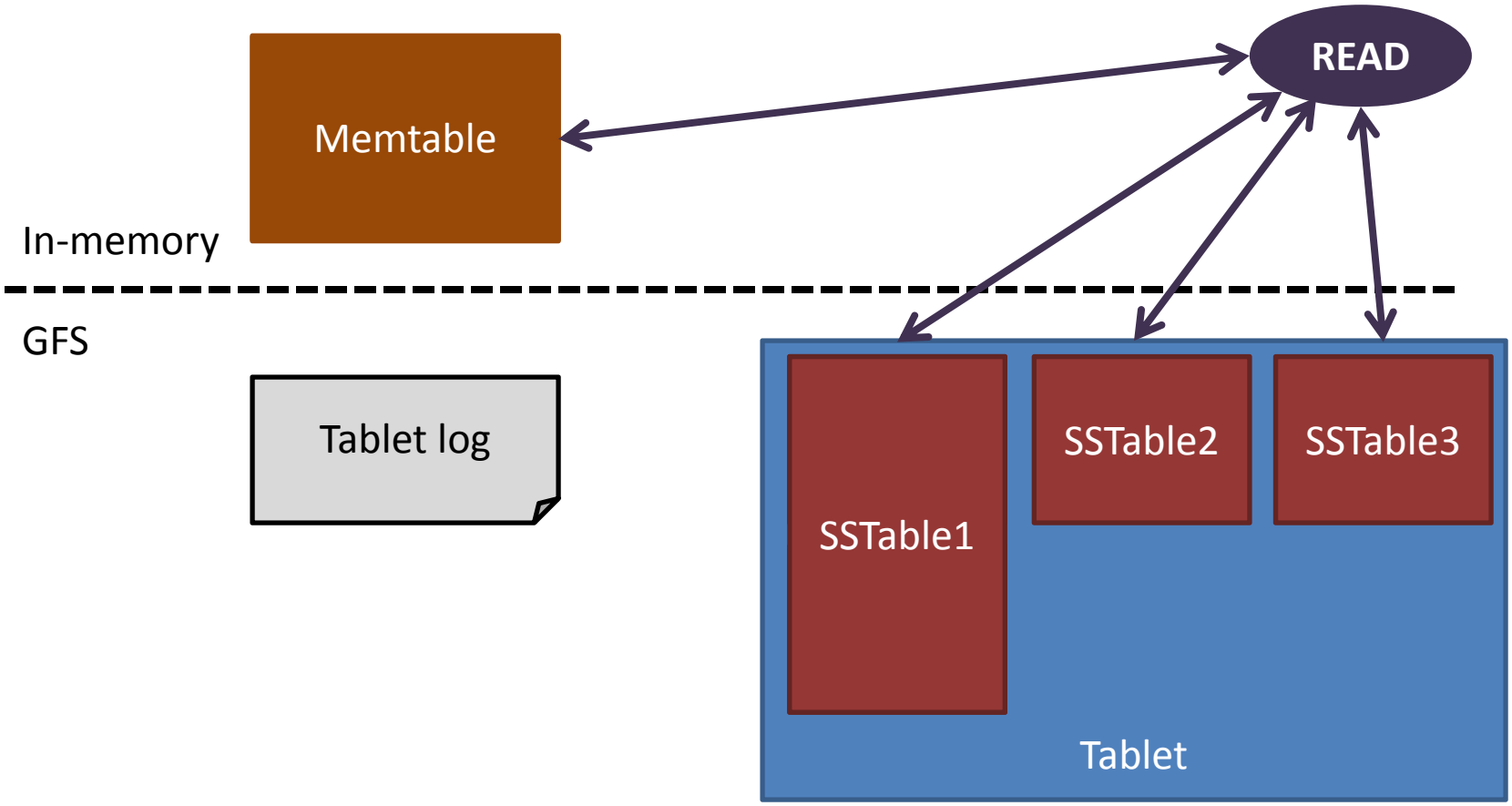
- When full, write Memtable as SSTable

What's the problem here?



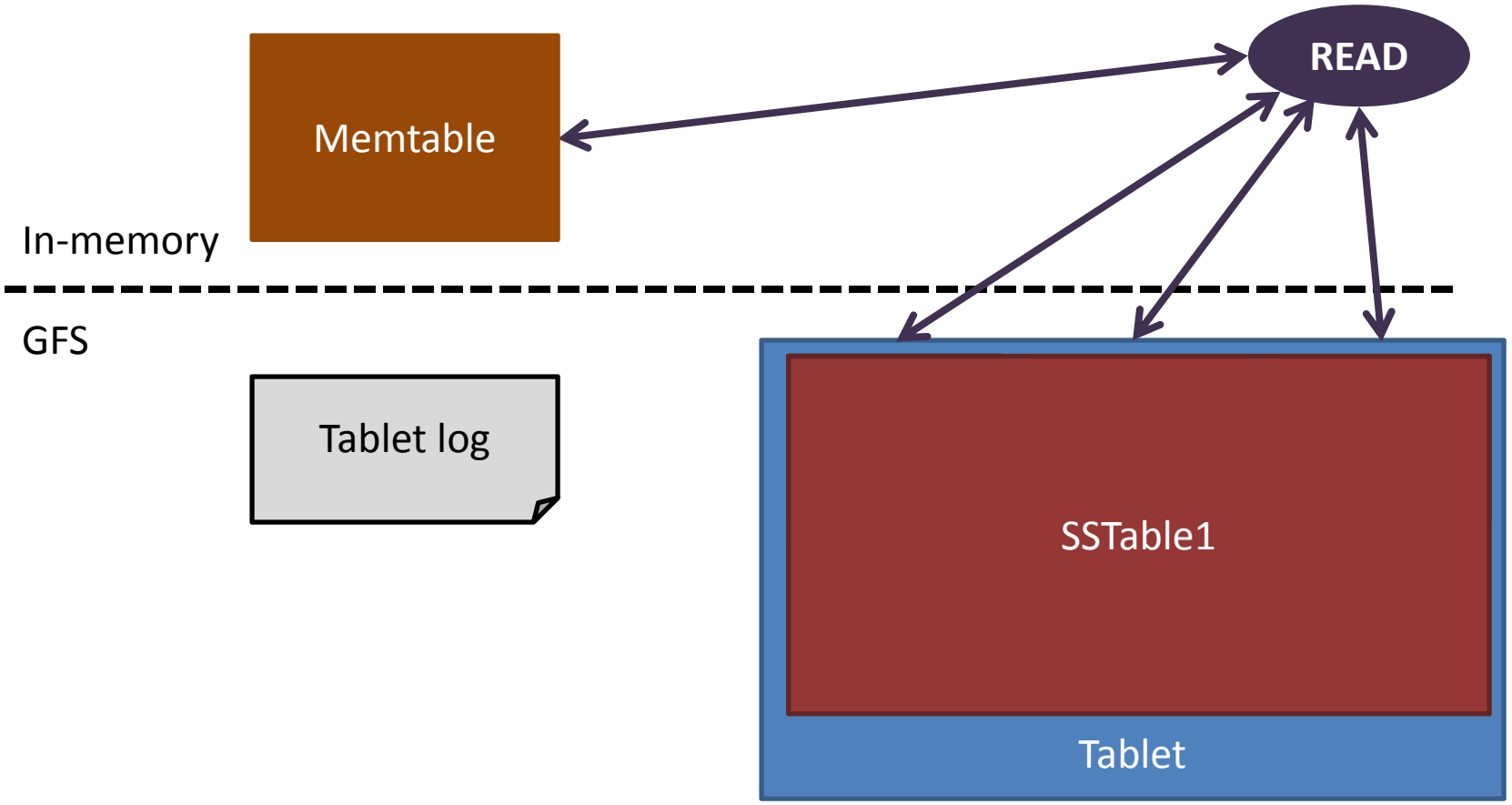
Bigtable: **Merge** Compaction

- Merge some of the SSTables (and the Memtable)



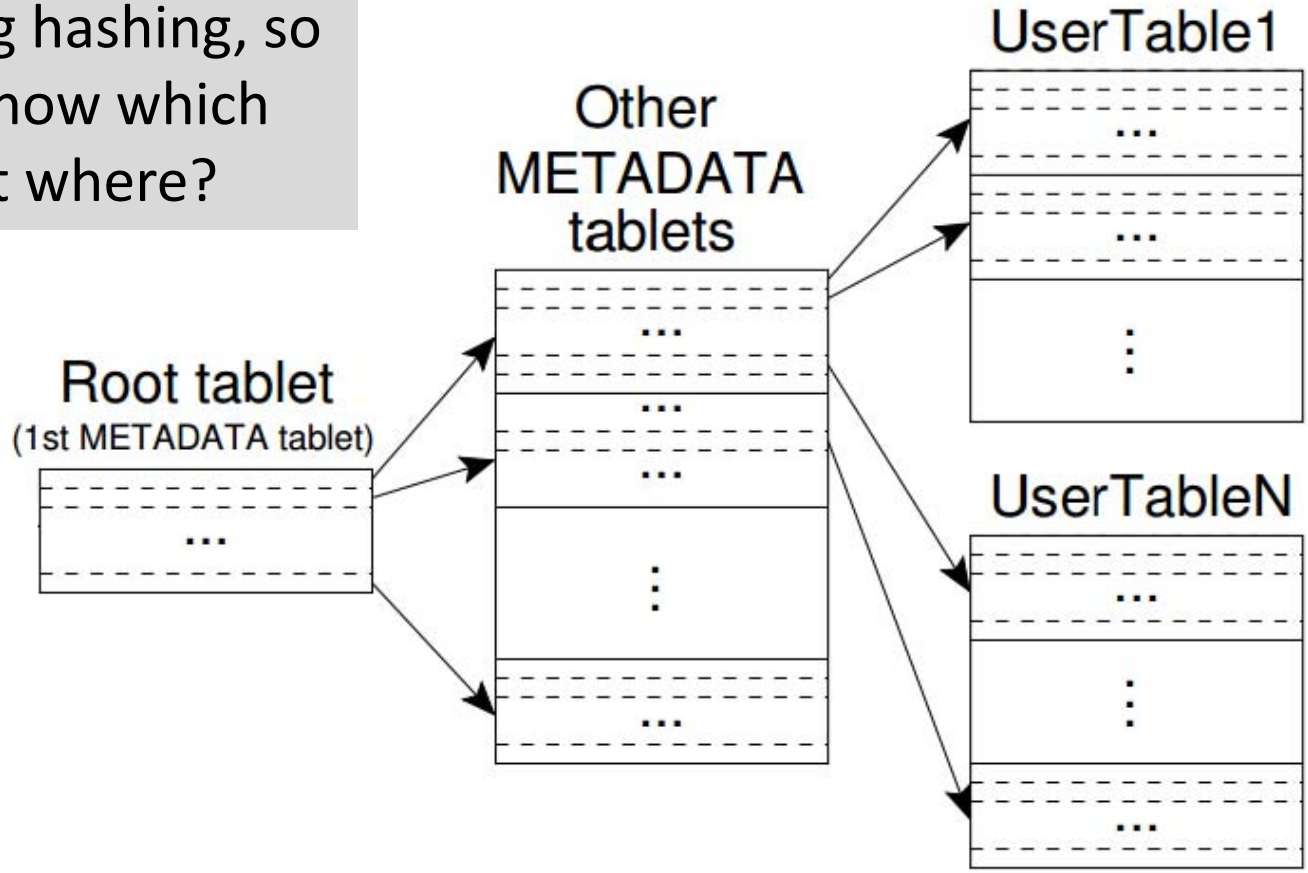
Bigtable: **Major** Compaction

- Merge all SSTables (and the Memtable)
- Makes reads more efficient!



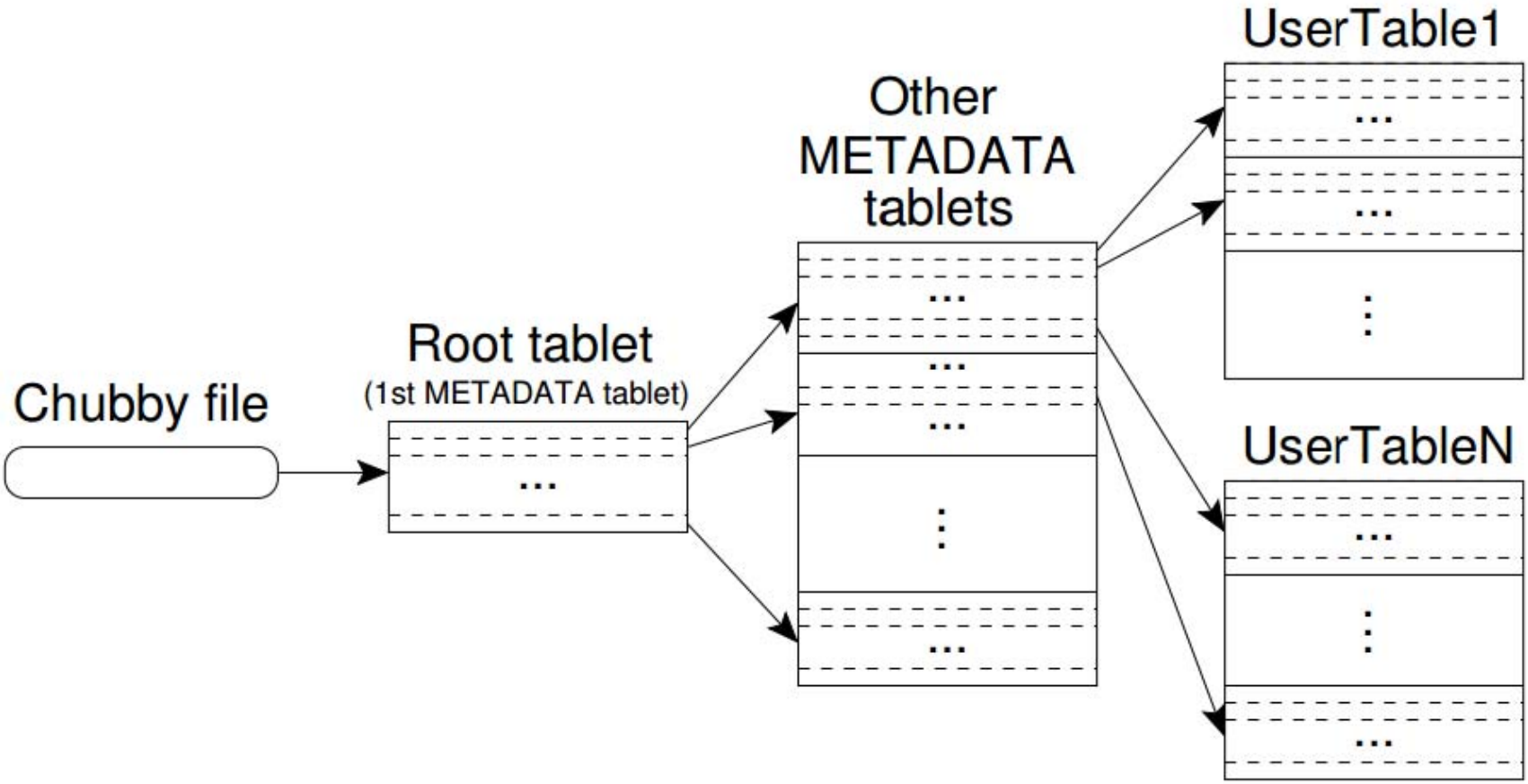
Bigtable: Hierarchical Structure

We're not using hashing, so how do we know which tablet went where?



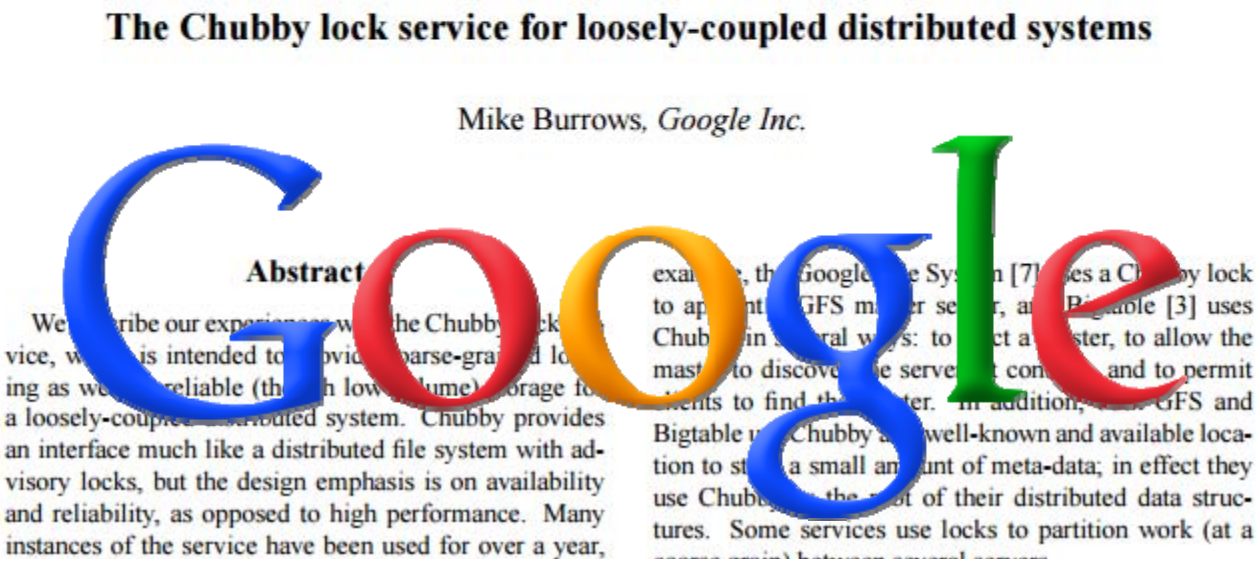
How do we choose the location of the Root tablet?

Bigtable: Hierarchical Structure



Bigtable: Consistency

- CHUBBY: Distributed consensus tool based on **PAXOS**
  - Quorum of five:
    - one master (chosen automatically) and four slaves
  - Co-ordinates distributed locks
  - Stores location of main “root tablet”
  - Holds other global state information (e.g., active servers)



The Chubby lock service for loosely-coupled distributed systems

Mike Burrows, *Google Inc.*

Abstract

We ribe our exp w he Chubb ck vice, w is intended to vid arse-gra d lo ing as we reliable (th h low ume) orage fo a loosely-coupled distributed system. Chubby provides an interface much like a distributed file system with advisory locks, but the design emphasis is on availability and reliability, as opposed to high performance. Many instances of the service have been used for over a year,

exa , th Google e Sys n [7] ses a Ch y lock to ap nt GFS m r se r, a Bi able [3] uses Chub in s ral w s: to ct a ster, to allow the mas to discov e serve con and to permit clients to find th ter. In addition, GFS and Bigtable u Chubby a well-known and available location to st a small an unt of meta-data; in effect they use Chubby the ot of their distributed data structures. Some services use locks to partition work (at a

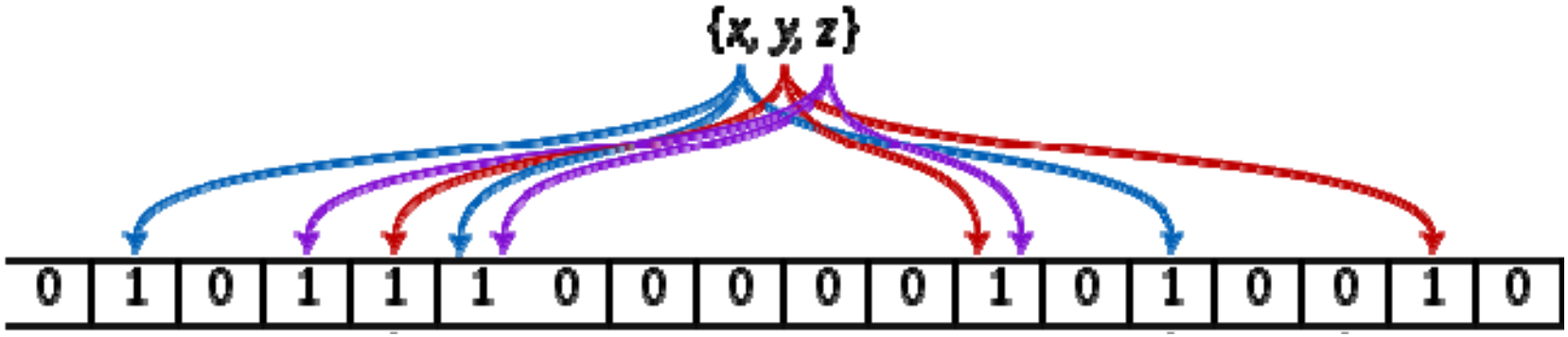
Bigtable: A Bunch of Other Things

- Locality groups: Group multiple column families together; assigned a separate SSTable
- Select storage: SSTables can be persistent or in-memory
- Compression: Applied on SSTable blocks; custom compression can be chosen
- Caches: SSTable-level and block-level
- Bloom filters: Find negatives cheaply …

Aside: Bloom Filter

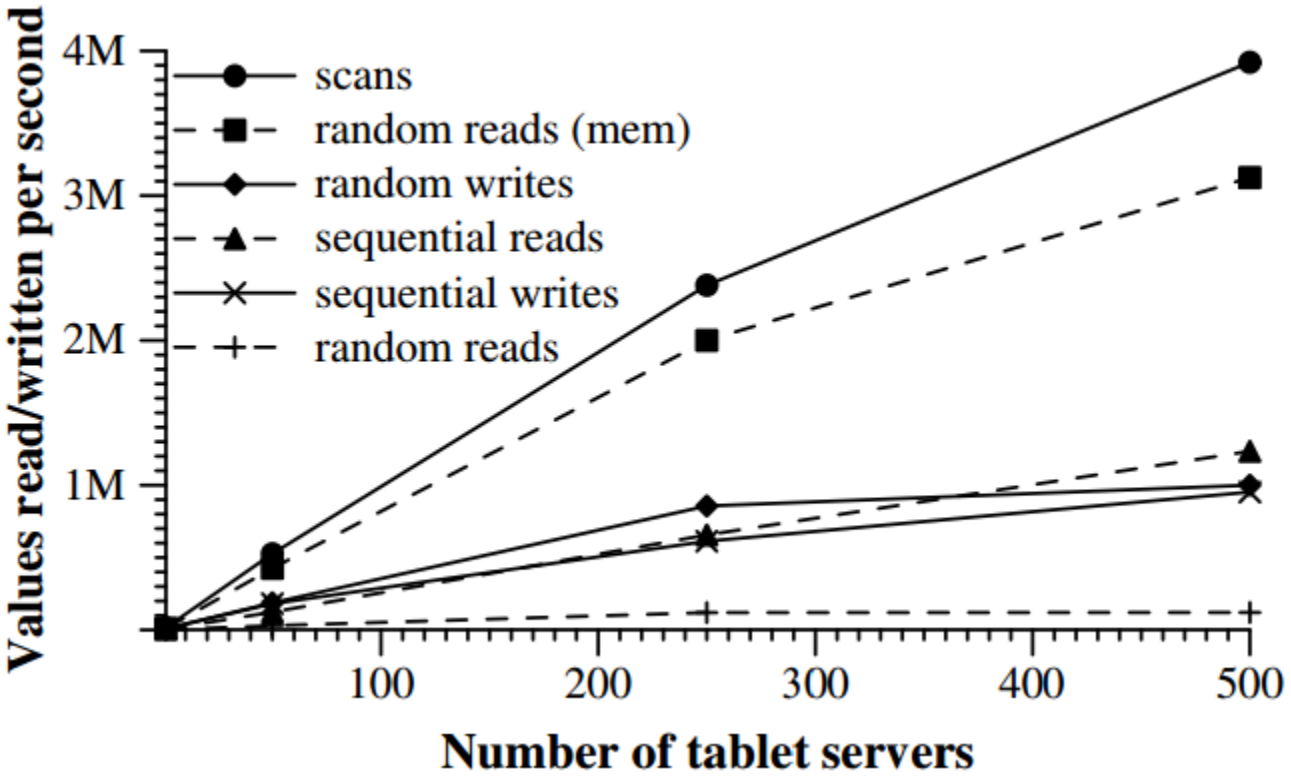
Reject "empty" queries using very little memory!

- Create a bit array of length m (init to 0's)
- Create k hash functions that map an object to an index of m (even distribution)
- Index o: set $m[\text{hash}_1(o)]$, …, $m[\text{hash}_k(o)]$ to 1
- Query o:
  - **any** $m[\text{hash}_1(o)]$, …, $m[\text{hash}_k(o)]$ set to **0** = not indexed
  - **all** $m[\text{hash}_1(o)]$, …, $m[\text{hash}_k(o)]$ set to **1** = **might** be indexed



Bigtable: an idea of performance [2006]

- Values are 1 Kb in size (blocks 64 Kb)
- Values are aggregate over all machines



Why are random (disk) reads so slow?

The read sizes are 1 kb, but a different 64 kb block must be sent over the network (almost) every time

Bigtable: an idea of performance [2006]

- Values are 1 Kb in size (blocks 64 Kb)
- Average values/second per server:

Experiment	# of Tablet Servers			
	1	50	250	500
random reads	1212	593	479	241
random reads (mem)	10811	8511	8000	6250
random writes	8850	3745	3425	2000
sequential reads	4425	2463	2625	2469
sequential writes	8547	3623	2451	1905
scans	15385	10526	9524	7843

- Adding more machines does add a cost!
- But overall performance does increase

Bigtable: examples in Google [2006]

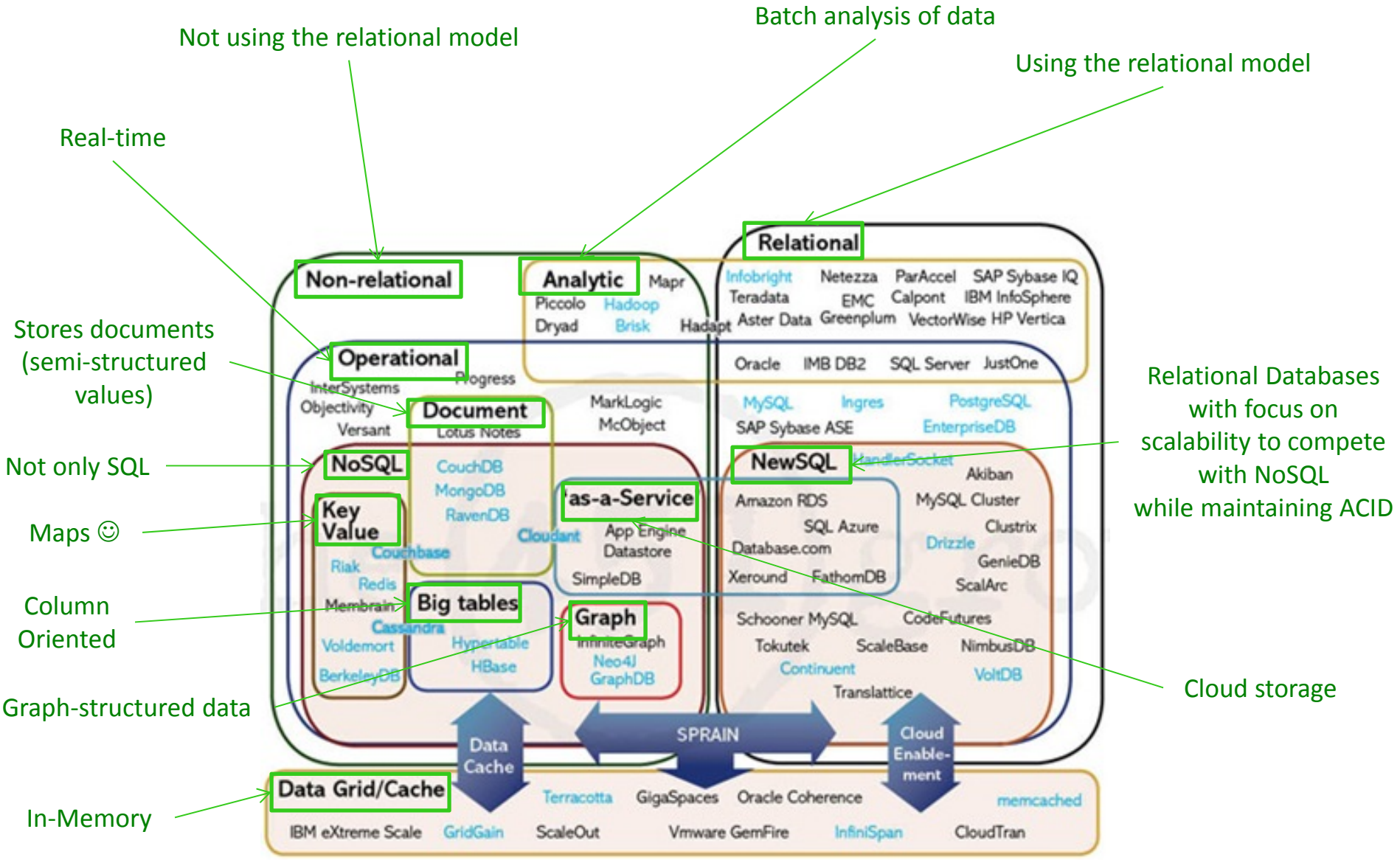
Project name	Table size (TB)	Compression ratio	# Cells (billions)	# Column Families	# Locality Groups	% in memory	Latency-sensitive?
Crawl	800	11%	1000	16	8	0%	No
Crawl	50	33%	200	2	2	0%	No
Google Analytics	20	29%	10	1	1	0%	Yes
Google Analytics	200	14%	80	1	1	0%	Yes
Google Base	2	31%	10	29	3	15%	Yes
Google Earth	0.5	64%	8	7	2	33%	Yes
Google Earth	70	–	9	8	3	0%	No
Orkut	9	–	0.9	8	5	1%	Yes
Personalized Search	4	47%	6	93	11	5%	Yes

Bigtable: Apache HBase

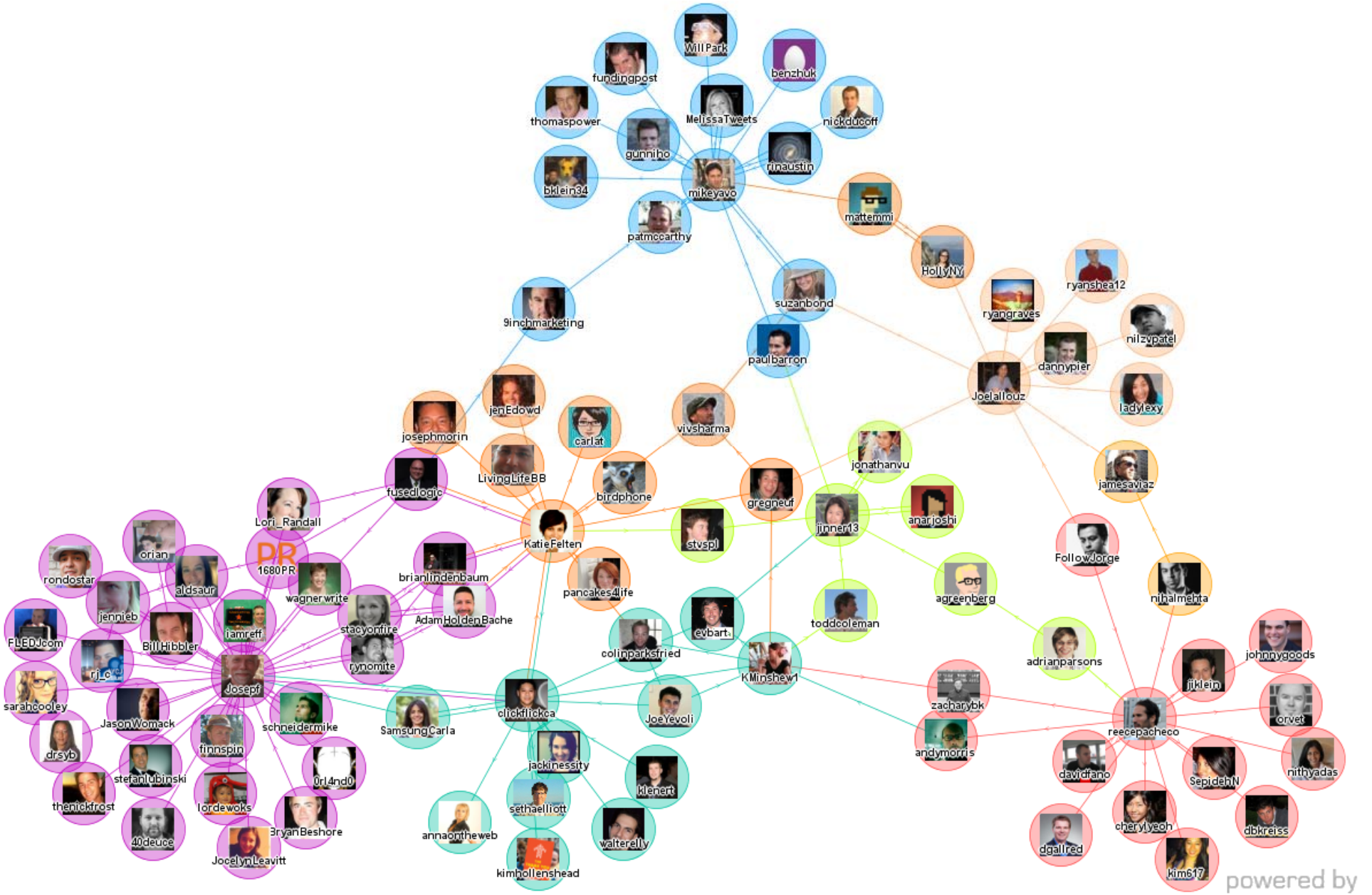
Open-source implementation of Bigtable ideas

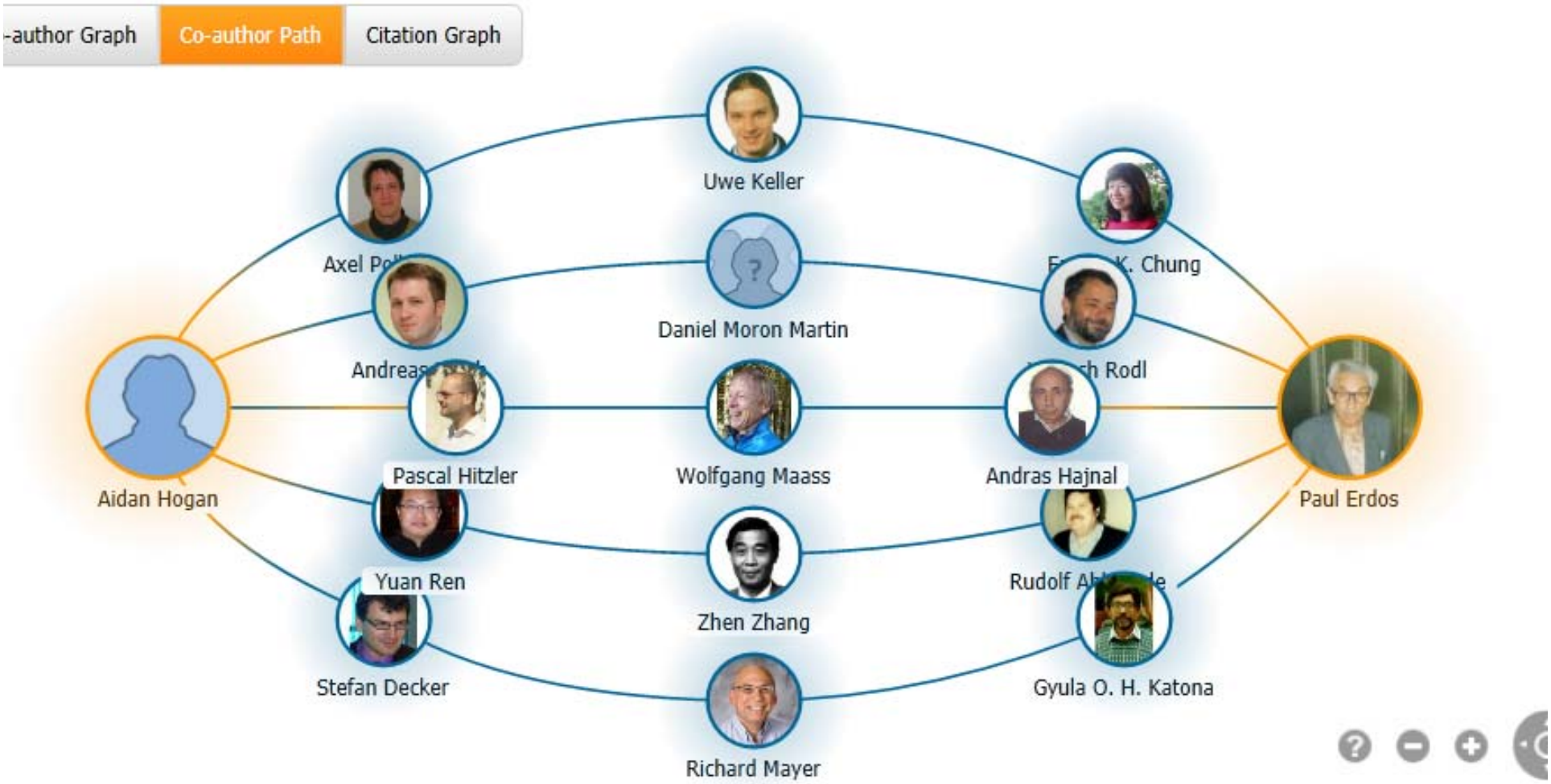
Similar ideas in Cassandra

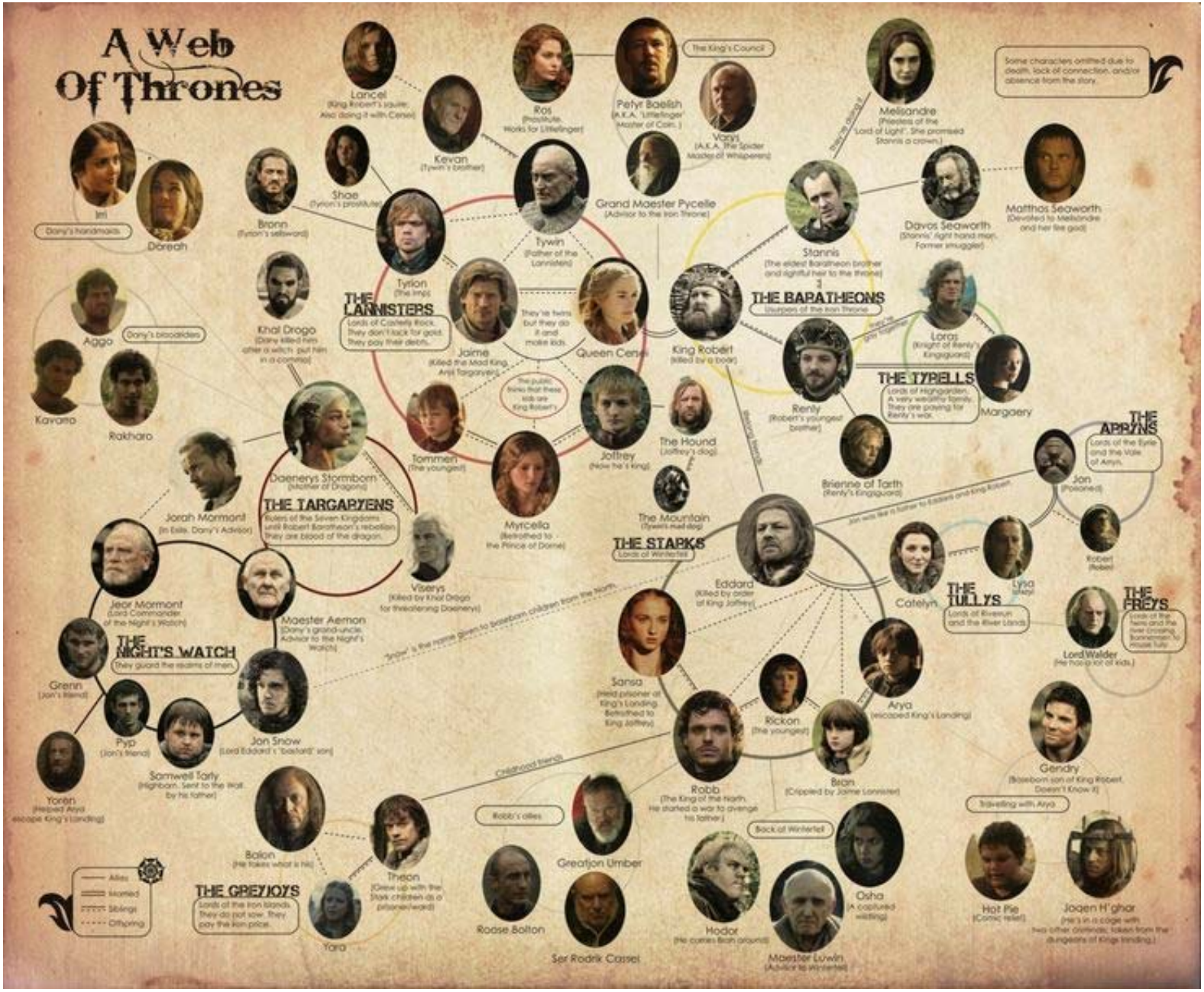
The Database Landscape



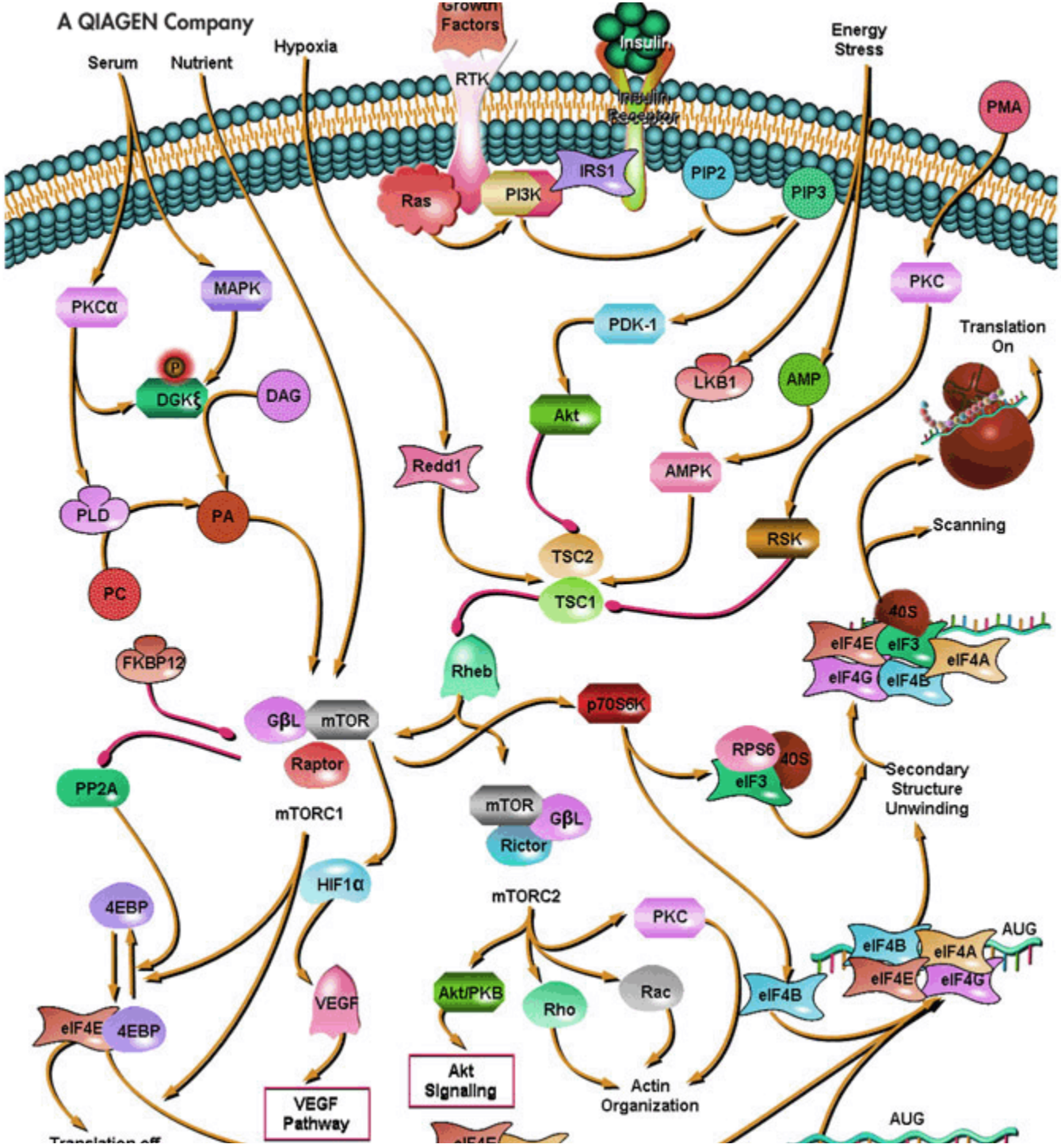
GRAPH DATABASES



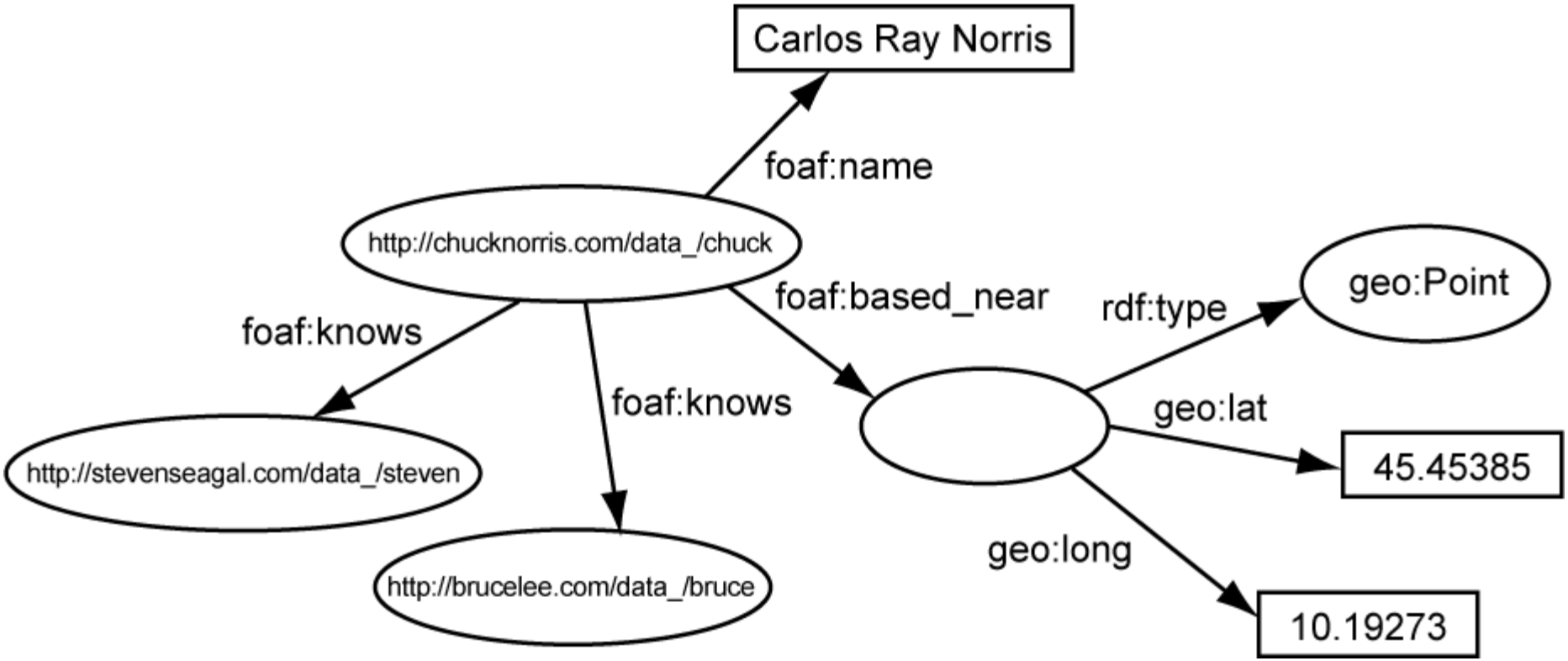

-author Graph
Co-author Path
Citation Graph
Uwe Keller
Axel Po
Daniel Moron Martin
K. Chung
Andreas
h Rodl
Aidan Hogan
Pascal Hitzler
Wolfgang Maass
Andras Hajnal
Paul Erdos
Yuan Ren
Zhen Zhang
Rudolf Ah
Stefan Decker
Gyula O. H. Katona
Richard Mayer


A Web Of Thrones
Some characters omitted due to death, lack of connection, and/or absence from the story.
The King's Council
Lancel
(King Robert's squire. Also doing it with Cersei)
Ros
(Prostitute. Works for Littlefinger)
Petyr Baelish
(A.K.A. 'Littlefinger' Master of Coin.)
Varys
(A.K.A. 'The Spider' Master of Whisperers)
Melisandre
(Priestess of the 'Lord of Light'. She promised Stannis a crown.)
Kevan
(Tywin's brother)
Shae
(Tyrion's prostitute)
Grand Maester Pycelle
(Advisor to the Iron Throne)
Matthos Seaworth
(Devoted to Melisandre and her fire god)
Davos Seaworth
(Stannis' right hand man. Former smuggler)
Stannis
(The eldest Baratheon brother and rightful heir to the throne)
Irri
(Dany's handmaids)
Doreah
Bronn
(Tyrion's sellsword)
Tywin
(Father of the Lannisters)
Tyrion
(The Imp)
THE LANNISTERS
Lords of Casterly Rock. They don't lack for gold. They pay their debts.
Jaime
(Killed the Mad King. Ayra Targaryen)
They're twins but they do it and make kids.
Queen Cersei
King Robert
(killed by a boar)
THE BARATHEONS
(Usurpers of the Iron Throne)
Loras
(Knight of Renly's Kingsguard)
Renly
(Robert's youngest brother)
THE TYRELLS
Lords of Highgarden. A very wealthy family. They are paying for Renly's war.
Margaery
Brienne of Tarth
(Renly's Kingsguard)
Khal Drogo
(Dany killed him after a witch put him in a coma)
Aggo
(Dany's bloodriders)
Kovarro
Rakharo
Tommen
(The youngest)
The public thinks that these kids are King Robert's
Joffrey
(Now he's king)
The Hound
(Joffrey's dog)
The Mountain
(Tywin's mad dog)
Myrcella
(Betrothed to the Prince of Dorne)
Daenerys Stormborn
(Mother of Dragons)
THE TARGARYENS
Rulers of the Seven Kingdoms until Robert Baratheon's rebellion. They are blood of the dragon.
Jorah Mormont
(In Exile. Dany's Advisor)
THE ARRYNS
Lords of the Eyrie and the Vale of Arryn.
Jon
(Poisoned)
Robert
(Robert)
Jeor Mormont
(Lord Commander of the Night's Watch)
Maester Aemon
(Dany's grand-uncle. Advisor to the Night's Watch)
Viserys
(Killed by Khal Drogo for threatening Daenerys)
THE STARKS
Lords of Winterfell
Eddard
(Killed by order of King Joffrey)
Catelyn
Lysa
(Wife)
THE TULLYS
Lords of Riverrun and the River Lands
THE FREYS
Lords of the twins and the river crossing. Bannermen to House Tully.
Lord Walder
(He has a lot of kids.)
"Snow" is the name given to bastard children from the North.
Grenn
(Jon's friend)
THE NIGHT'S WATCH
They guard the realms of men.
Pyp
(Jon's friend)
Jon Snow
(Lord Eddard's 'bastard' son)
Sansa
(Held prisoner at King's Landing. Betrothed to King Joffrey)
Arya
(escaped King's Landing)
Rickon
(The youngest)
Bran
(Crippled by Jaime Lannister)
Robb
(The King of the North. He started a war to avenge his father.)
Gendry
(Bastard son of King Robert. Doesn't know it)
Samwell Tarly
(Highborn. Sent to the Wall by his father)
Yoren
(Helped Arya escape King's Landing)
Childhood friends
Traveling with Arya
Balon
(He takes what is his)
Theon
(Grew up with the Stark children as a prisoner/ward)
Greatjon Umber
Robb's allies
Back at Winterfell
Osha
(A captured wildling)
Hot Pie
(Comic relief)
Joaen H'ghar
(He's in a cage with two other criminals, taken from the dungeons of Kings Landing.)
Roose Bolton
Ser Rodrik Cassel
Hodor
(He carries Bran around)
Maester Luwin
(Advisor to Winterfell)
Yara
THE GREYJOYS
Lords of the Iron Islands. They do not sow. They pay the iron price.
Allies
Married
Siblings
Offspring


N
DISTANCIAS GEOGRÁFICAS DISTORSIONADAS
CIUDAD EMPRESARIAL
2
VESPUCIO NORTE
Zapadores
Dorsal
Einstein
Cementerios
Cerro Blanco
Patronato
Puente Cal y Canto
SANTA ANA
Plaza de Armas
Bellas Artes
Catedral
1
SAN PABLO
Lo Prado
Blanqueado
Gruta de Lourdes
Quinta Normal
Cumming
Pudahuel
Barrancas
Laguna Sur
Las Parcelas
Monte Tabor
Del Sol
Santiago Bueras
5
PLAZA DE MAIPÚ
Neptuno
Pajaritos
Las Rejas
Ecuador
San Alberto Hurtado
U. de Santiago
Estación Central
U.L.A.
República
LOS HÉROES
La Moneda
U. de Chile
Santa Lucía
U. Católica
CERRO SANTA LUCÍA
BAQUEDANO
Salvador
M. Montt
P. de Valdivia
Los Leones
El Golf
Alcántara
Escuela Militar
Manquehue
H. de Magallanes
LOS DOMINICOS
TOBALABA
4
C. Colón
Francisco Bilbao
Príncipe de Gales
Simón Bolívar
Plaza Egaña
Los Orientales
Grecia
Los Presidentes
Quilín
Las Torres
Macul
VICUÑA MACKENNA
4A
VICENTE VALDÉS
5
Rojas Magallanes
Trinidad
San José de la Estrella
Los Quillayes
Elisa Correa
Hospital Sótero del Río
Protectora de la Infancia
Toesca
Parque O`Higgins
Rondizzoni
Franklin
El Llano
San Miguel
Lo Vial
Departamental
Ciudad del Niño
Lo Ovalle
El Parrón
LA CISTERNA
2
4A
P. Bustamante
Santa Isabel
Irarrázaval
Ñuble
Rodrigo de Araya
Carlos Valdovinos
Camino Agrícola
San Joaquín
Pedrero
Mirador
B. de La Florida
Santa Julia
La Granja
San Ramón
Santa Rosa
Aeropuerto Internacional AMB
Costanera Norte
Río Mapocho
Carrascal
Mapocho
Alameda
Av. Providencia
Av. Apoquindo
Av. Francisco Bilbao
Av. Grecia
Av. Departamental
Av. Malta
Av. Américo Vespucio
Av. Lo Espejo
Av. Gabriela
Lo Ovalle
Av. 5 de Abril
Autopista del Sol
Ruta 68
Av. San Pablo
Av. Teniente Cruz
Av. Pajaritos
Av. Las Rejas
Av. Gral. Velásquez
Cerrillos
Av. Presidente Jorge Alessandri - Ruta 5 Sur
Ruta 5 Norte
Av. Independencia
Av. Recoleta
Av. Zapadores
Av. El Salto
Av. Perú
CERRO SAN CRISTÓBAL
Av. Santa María
Av. Vitacura
Av. Kennedy
Av. Manquehue Norte
Padre Hurtado
Av. Tobalaba
Av. José Pedro Alessandri
Av. Vicuña Mackenna
Gran Avenida
Av. Santa Rosa


A QIAGEN Company
Growth Factors
Insulin
Serum
Nutrient
Hypoxia
Energy Stress
RTK
Insulin Receptor
PMA
Ras
PI3K
IRS1
PIP2
PIP3
PKCα
MAPK
PKC
PDK-1
Translation On
LKB1
AMP
DGKξ
DAG
Akt
Redd1
AMPK
PLD
PA
RSK
Scanning
TSC2
PC
TSC1
40S
eIF4E
eIF3
eIF4A
eIF4G
eIF4B
FKBP12
Rheb
GβL
mTOR
p70S6K
Raptor
RPS6
40S
eIF3
PP2A
Secondary Structure Unwinding
mTOR
GβL
Rictor
mTORC1
HIF1α
mTORC2
4EBP
PKC
AUG
eIF4B
eIF4A
eIF4E
eIF4G
Akt/PKB
Rho
Rac
VEGF
eIF4E
4EBP
Akt Signaling
Actin Organization
VEGF Pathway
AUG


Google
The Knowledge Graph
Learn more about one of the key breakthroughs
behind the future of search.


Carlos Ray Norris
foaf:name
http://chucknorris.com/data_/chuck
foaf:based_near
foaf:knows
foaf:knows
rdf:type
geo:Point
geo:lat
45.45385
geo:long
10.19273
http://stevenseagal.com/data_/steven
http://brucelee.com/data_/bruce

Data = Graph

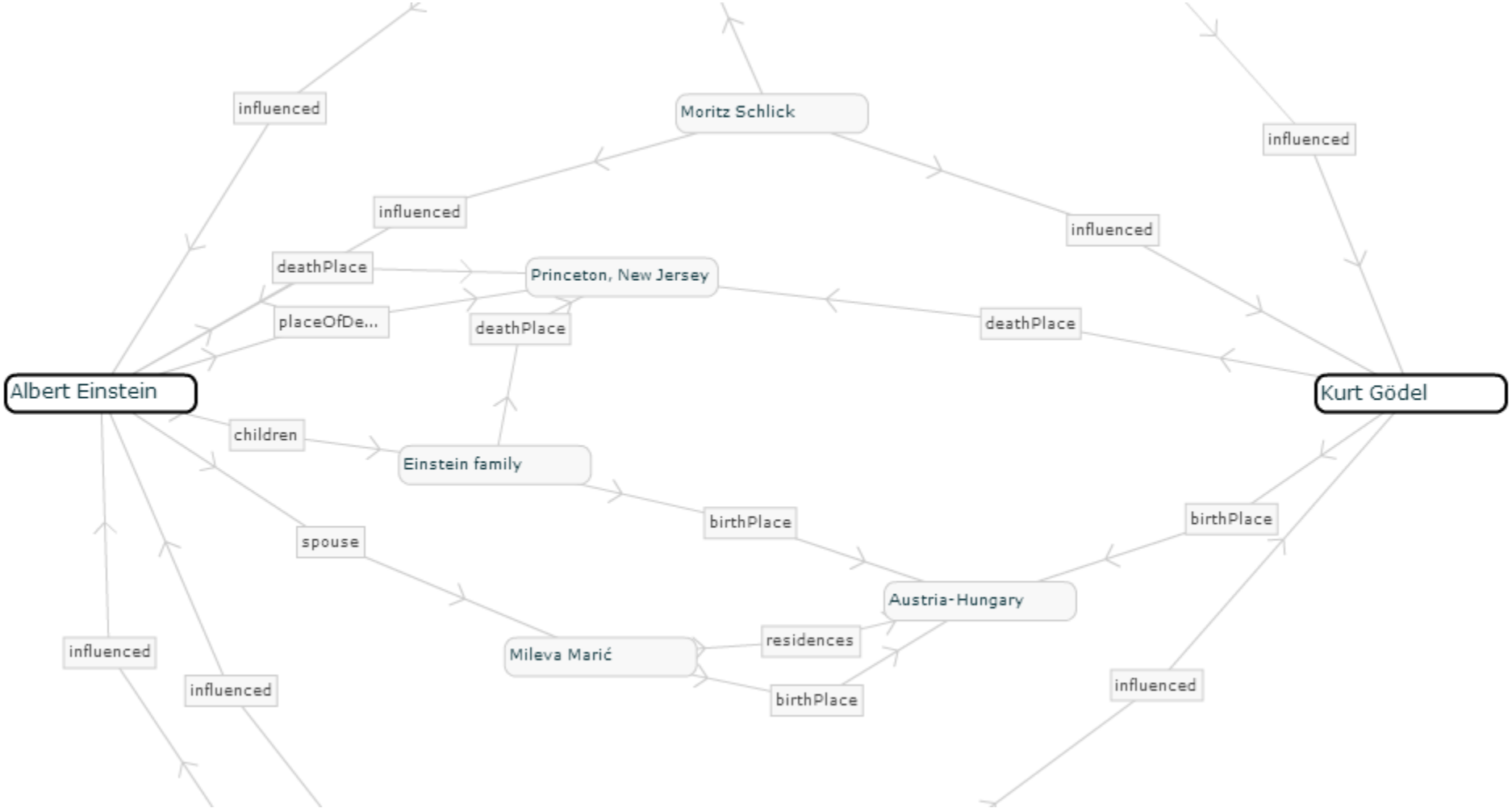
- Any data can be represented as a directed labelled graph (not always neatly)]

When is it a good idea to consider data as a graph?

When you want to answer questions like:

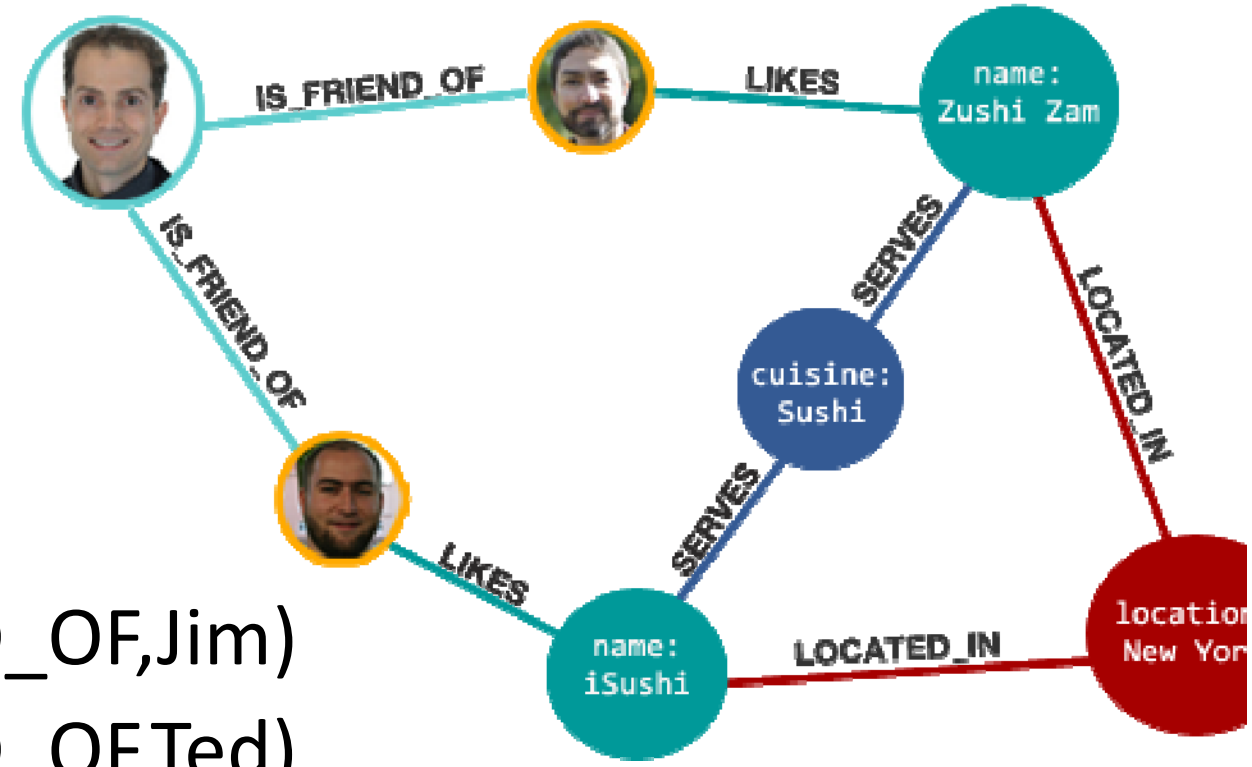
- How many social hops is this user away?
- What is my Erdős number?
- What connections are needed to fly to Perth?
- How are Einstein and Godel related?

RelFinder


influenced
Moritz Schlick
influenced
influenced
influenced
deathPlace
Princeton, New Jersey
placeOfDe...
deathPlace
deathPlace
Albert Einstein
Kurt Gödel
children
Einstein family
spouse
birthPlace
birthPlace
Austria-Hungary
influenced
Mileva Marić
residences
influenced
birthPlace
influenced

Leading Graph Database

Graph Databases



(Fred,IS_FRIEND_OF,Jim)

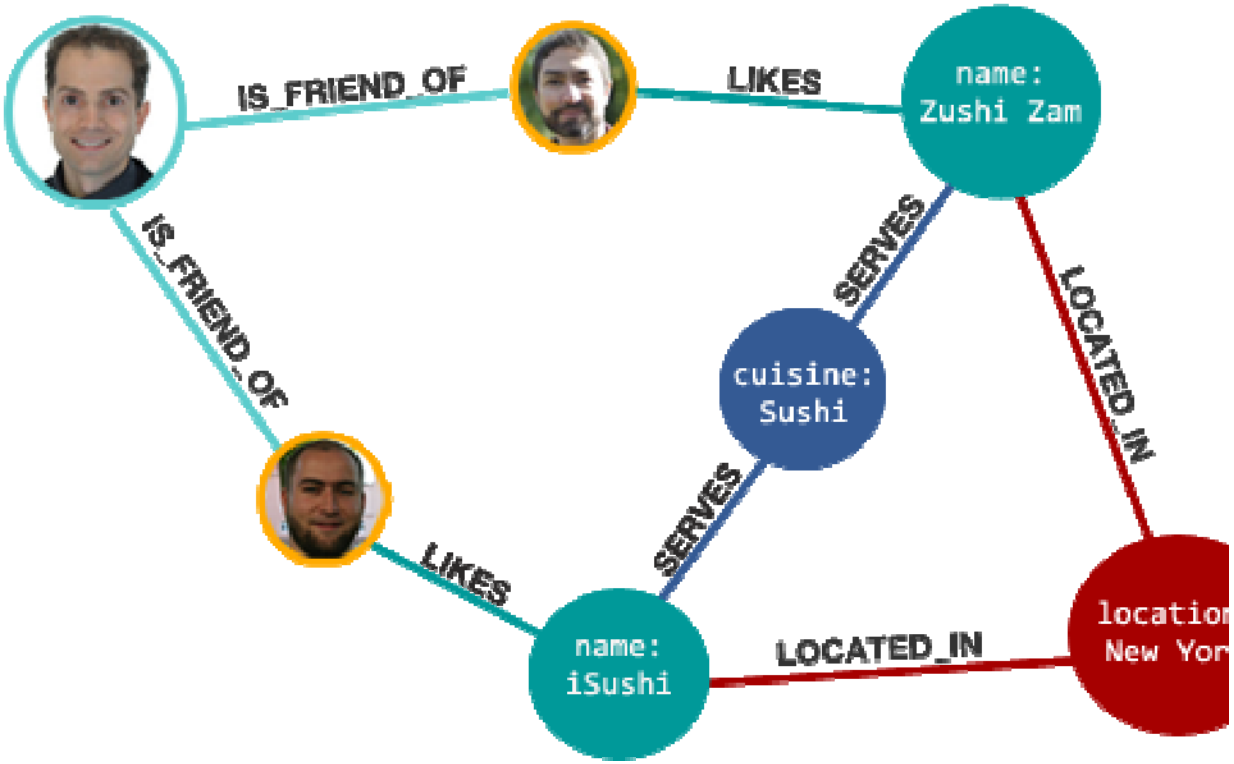
(Fred,IS_FRIEND_OF,Ted)

(Ted,LIKES,Zushi_Zam)

(Zuzhi_Zam,SERVES,Sushi)

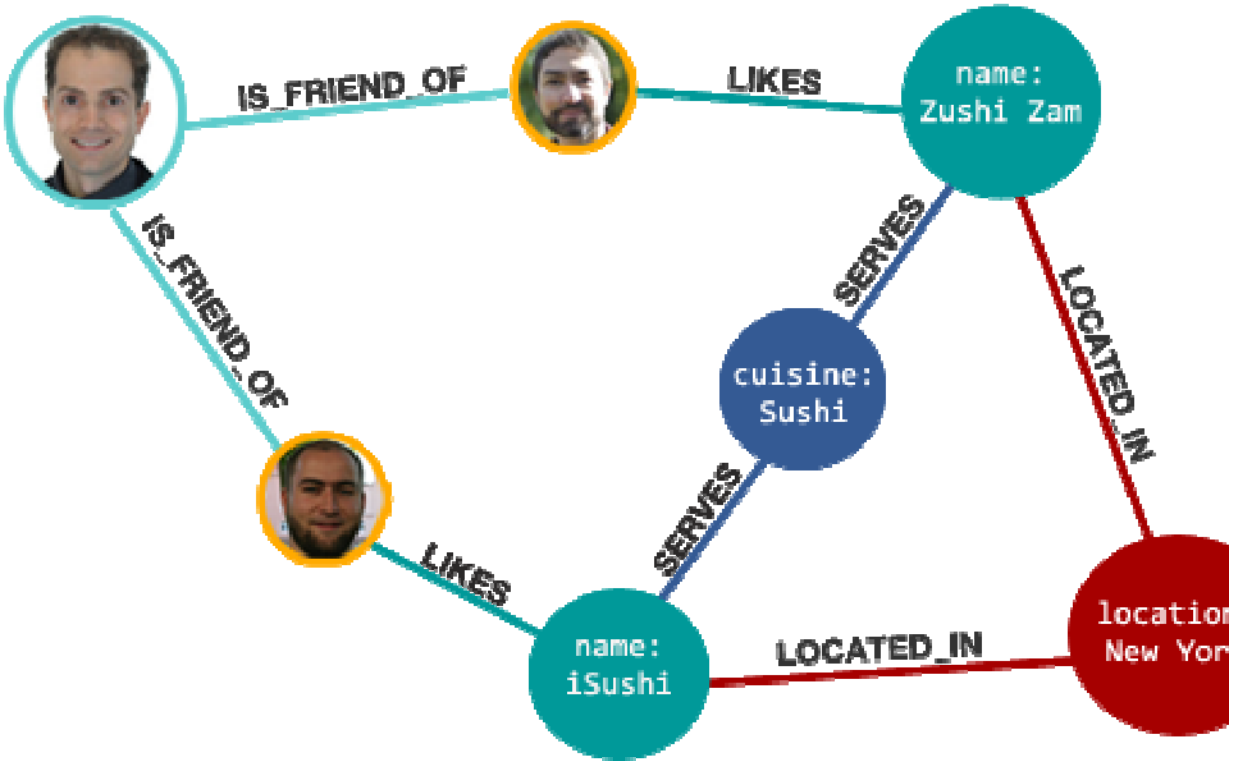
…

Graph Databases: Index Nodes



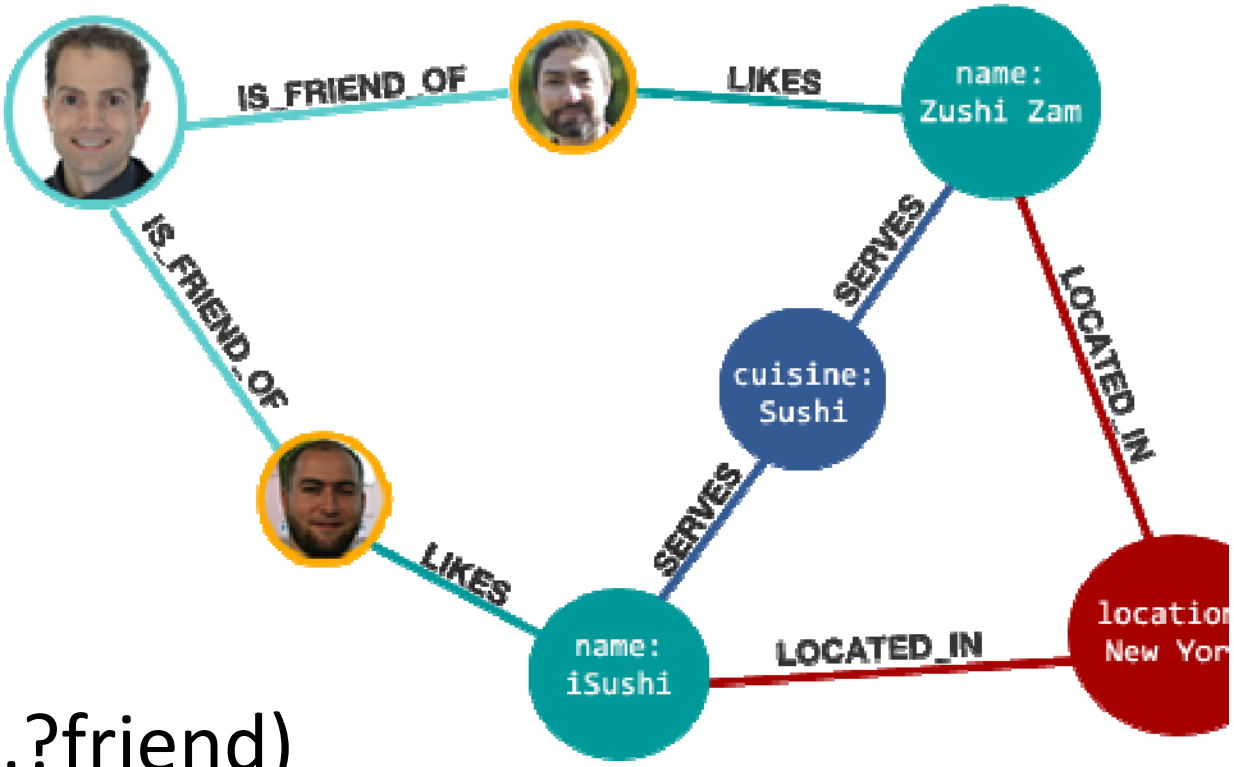
Fred -> (Fred,IS_FRIEND_OF,Jim)
(Fred,IS_FRIEND_OF,Ted)

Graph Databases: Index Relations



LIKES -> (Ted,LIKES,Zushi_Zam)
(Jim,LIKES,iSushi)

Graph Databases: Graph Queries



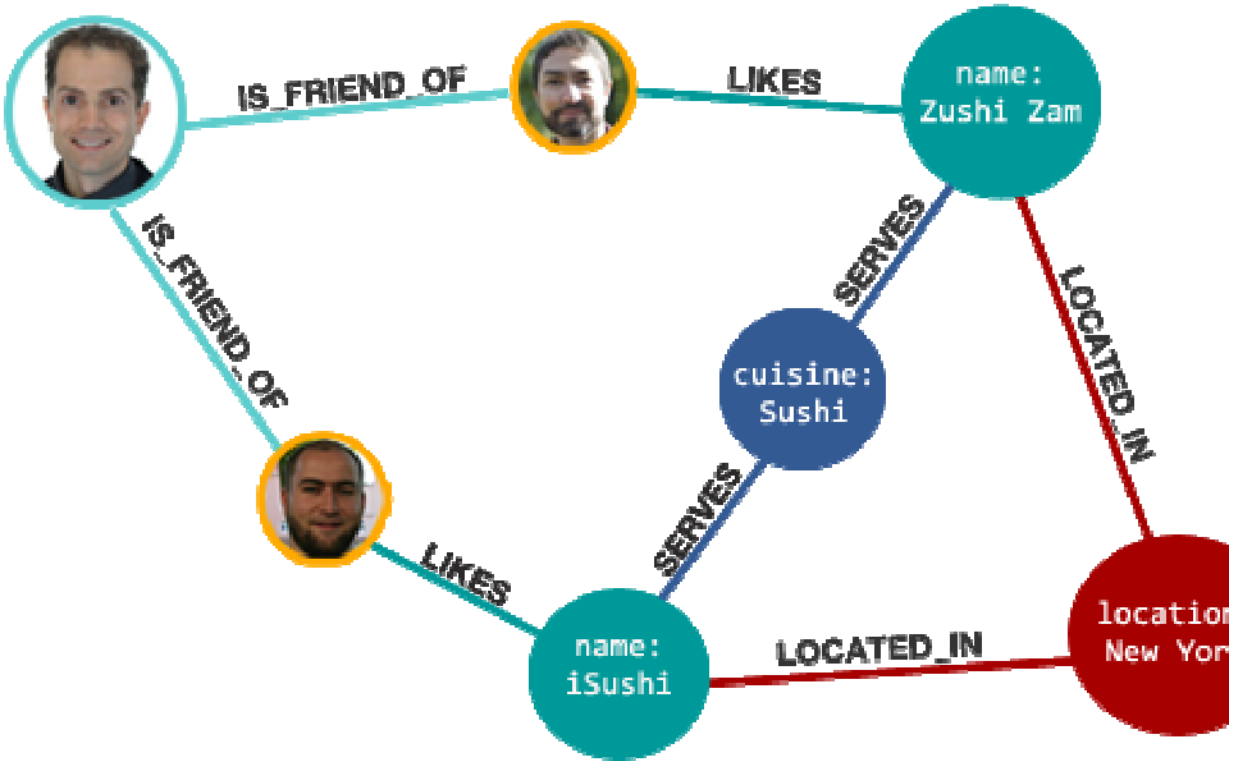
(Fred,IS_FRIEND,?friend)

(?friend,LIKES,?place)

(?place,SERVES,?sushi)

(?place,LOCATED_IN,New_York)

Graph Databases: Path Queries

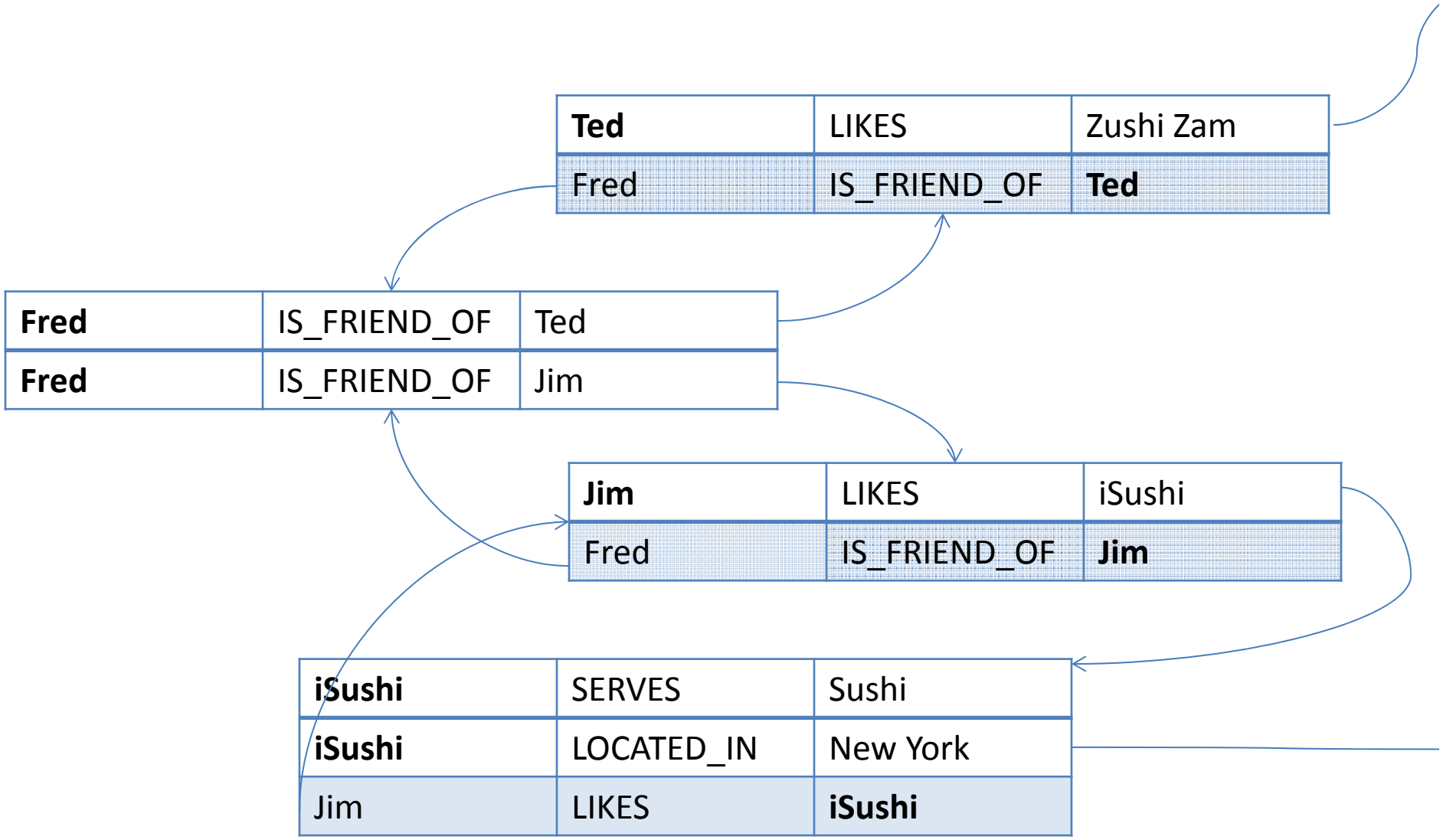


What about scalability?

(Fred,IS_FRIEND*,?friend_of_friend)

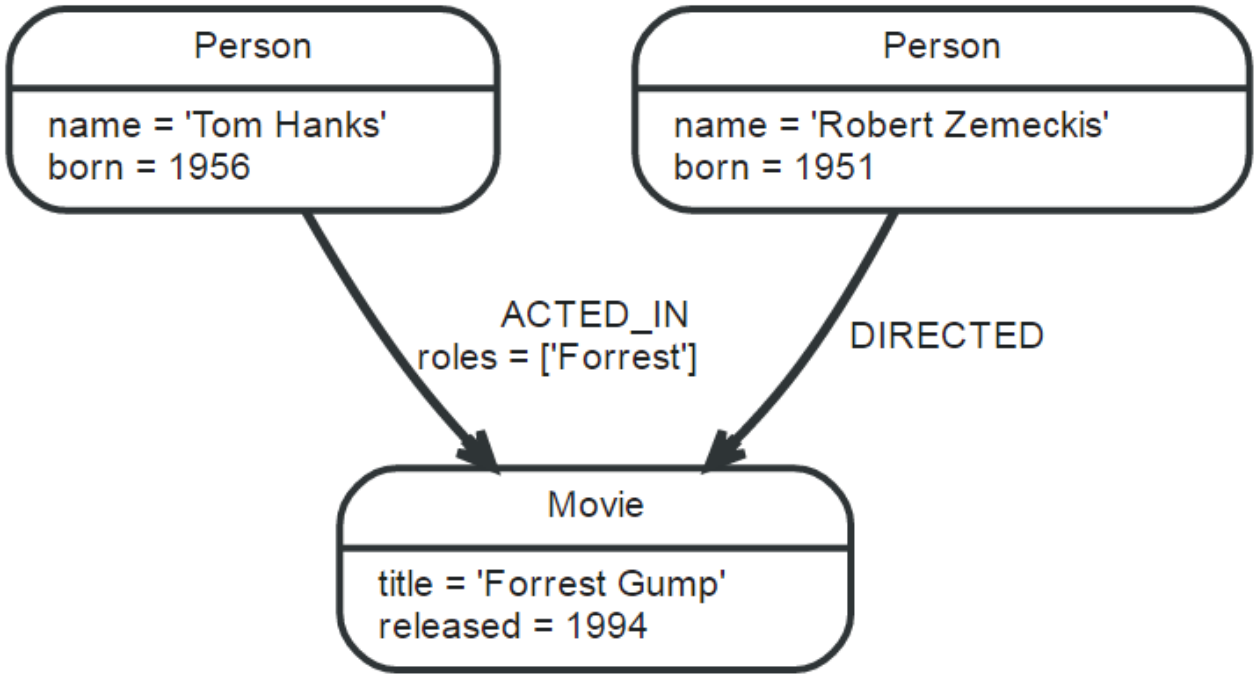
(?friend_of_friend,LIKES,Zushi_Zam)

Graph Database: Index-free Adjacency

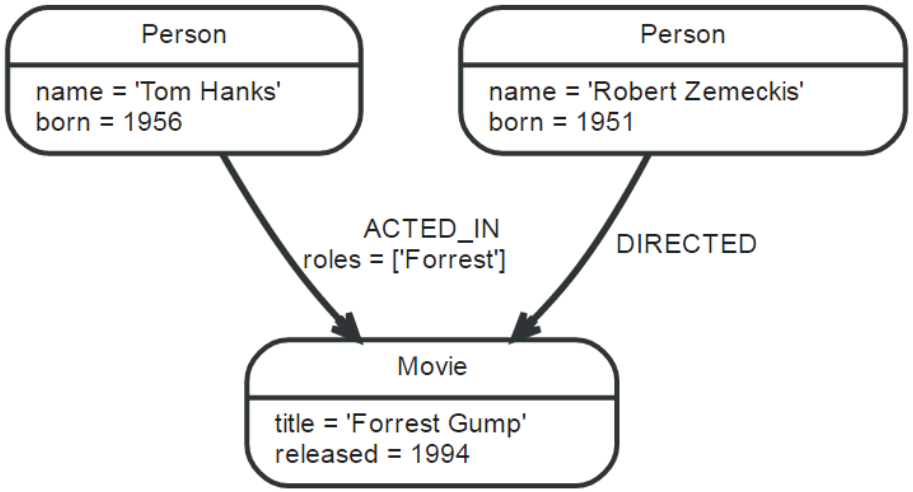


Property Graph Model

- Graph nodes and edges can have
  - Multiple attributes like name, born, roles
  - Multiple labels like Person, Actor, Movie



Query Language: Cypher




```
MATCH (p:Person { name:"Tom Hanks" })-[r:ACTED_IN]->(m:Movie)
RETURN m.title, r.roles
```

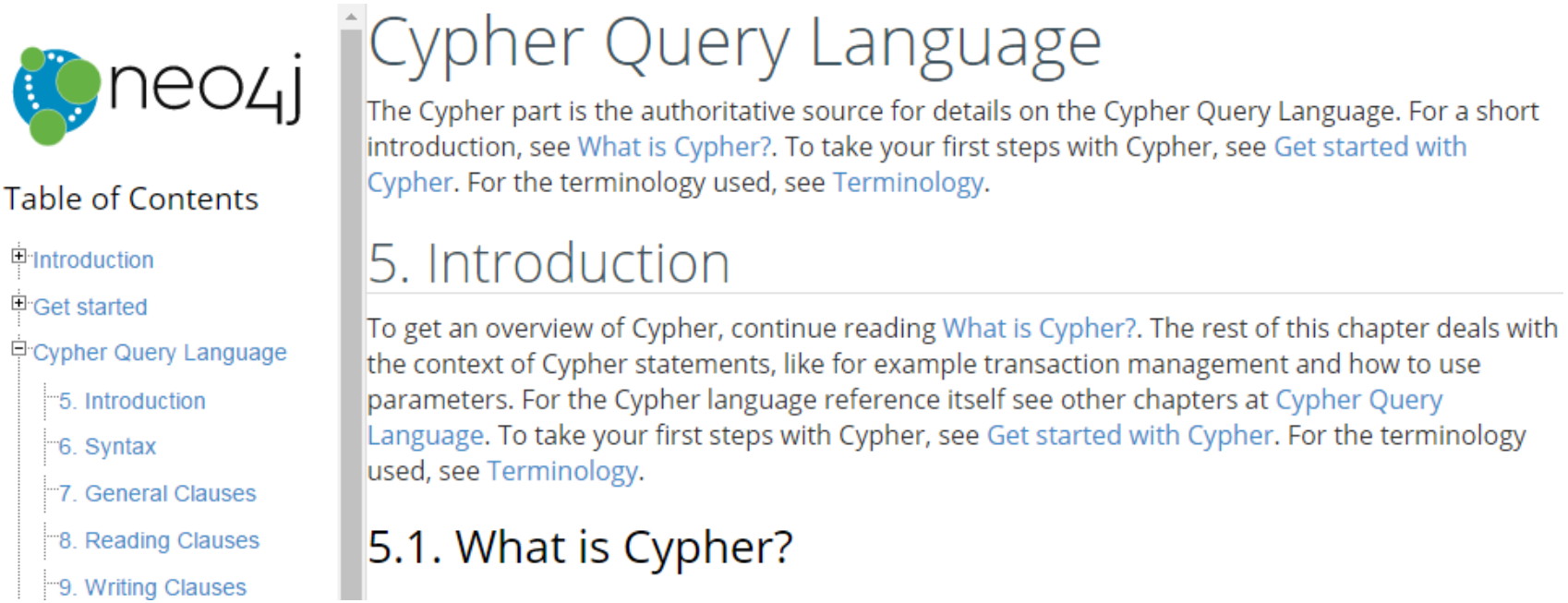
What will this return?

```
+-----------------------------+
| m.title         | r.roles   |
+-----------------------------+
| "Forrest Gump" | ["Forrest"] |
+-----------------------------+
1 row
```

Query Language: Cypher

- Relational query features (UNION, FILTER, ORDER BY, GROUP BY, COUNT, etc.)
- Path query features (*, 1..5, etc.)

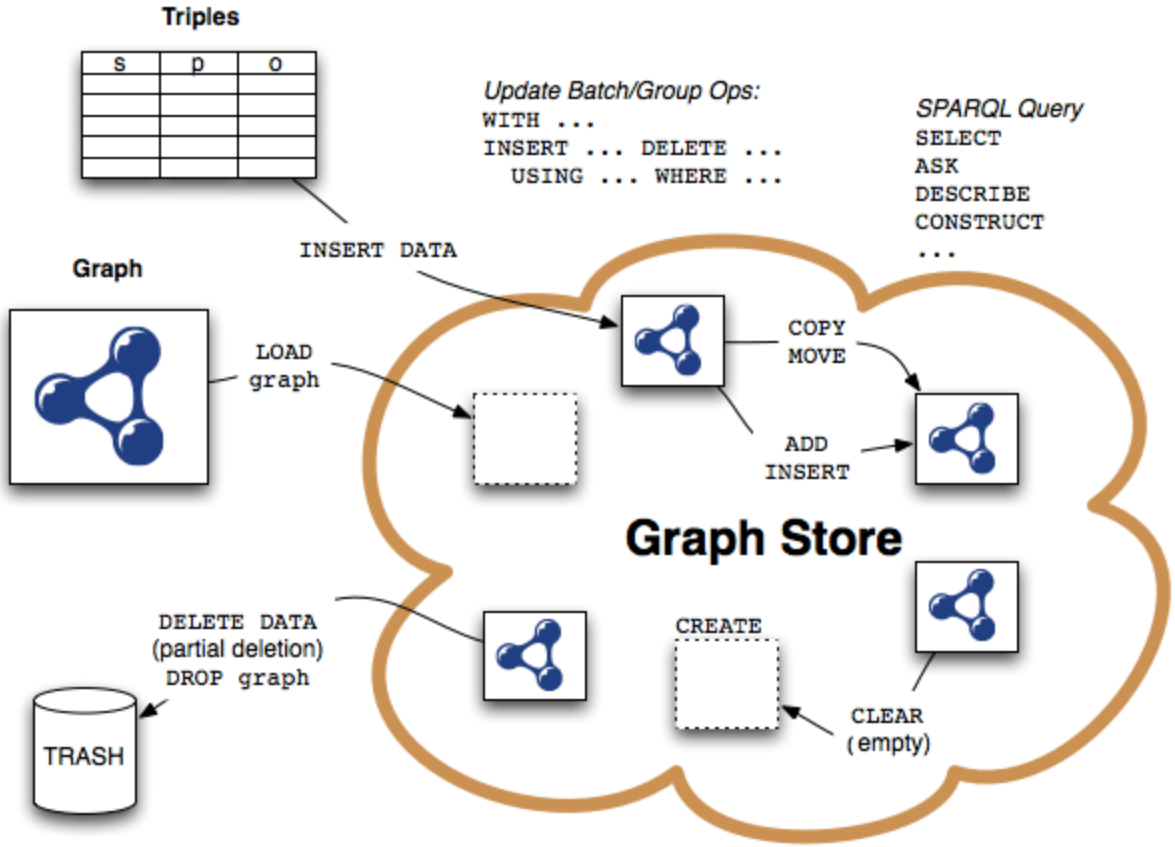
http://neo4j.com/docs/developer-manual/current/#cypher-query-lang



Rank			DBMS	Database Model	Score		
May 2016	Apr 2016	May 2015			May 2016	Apr 2016	May 2015
1.	1.	1.	Oracle	Relational DBMS	1462.02	-5.51	+19.93
2.	2.	2.	MySQL	Relational DBMS	1371.83	+1.72	+77.56
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1142.82	+7.77	+11.79
4.	4.	4.	MongoDB	Document store	320.22	+7.78	+42.90
5.	5.	5.	PostgreSQL	Relational DBMS	307.61	+3.89	+34.09
6.	6.	6.	DB2	Relational DBMS	185.96	+1.87	-15.09
7.	↑ 8.	↑ 8.	Cassandra	Wide column store	134.50	+4.83	+27.95
8.	↓ 7.	↓ 7.	Microsoft Access	Relational DBMS	131.58	-0.39	-14.00
9.	9.	↑ 10.	Redis	Key-value store	108.24	-3.00	+13.51
10.	10.	↓ 9.	SQLite	Relational DBMS	107.26	-0.70	+2.10
11.	11.	↑ 14.	Elasticsearch	Search engine	86.31	+3.73	+21.48
12.	↑ 13.	↑ 13.	Teradata	Relational DBMS	73.74	+1.48	+3.62
13.	↓ 12.	↓ 11.	SAP Adaptive Server	Relational DBMS	71.48	-1.84	-14.01
14.	14.	↓ 12.	Solr	Search engine	65.62	-0.40	-17.31
15.	15.	15.	HBase	Wide column store	51.84	+0.35	-9.87
16.	16.	↑ 17.	Hive	Relational DBMS	47.51	-1.57	+2.94
17.	17.	↓ 16.	FileMaker	Relational DBMS	46.71	+0.60	-6.19
18.	18.	18.	Splunk	Search engine	44.31	+1.96	+3.59
19.	19.	↑ 21.	SAP HANA	Relational DBMS	41.37	+1.02	+8.59
20.	↑ 21.	↑ 25.	MariaDB	Relational DBMS	33.97	+2.38	+10.37
21.	↓ 20.	↑ 22.	Neo4j	Graph DBMS	32.61	+0.70	+3.97
22.	22.	↓ 19.	Informix	Relational DBMS	30.58	-0.94	-6.16
23.	23.	↓ 20.	Memcached	Key-value store	27.90	-0.11	-5.21
24.	24.	24.	Couchbase	Document store	24.29	-0.73	-0.95
25.	25.	↑ 30.	Amazon DynamoDB	Multi-model	23.60	+0.48	+8.73

http://db-engines.com/en/ranking

SPARQL


Triples
S
P
O
Update Batch/Group Ops:
WITH ...
INSERT ... DELETE ...
USING ... WHERE ...
SPARQL Query
SELECT
ASK
DESCRIBE
CONSTRUCT
...
INSERT DATA
Graph
LOAD
graph
COPY
MOVE
ADD
INSERT
Graph Store
DELETE DATA
(partial deletion)
DROP graph
TRASH
CREATE
CLEAR
(empty)

Rank			DBMS	Database Model	Score		
May 2016	Apr 2016	May 2015			May 2016	Apr 2016	May 2015
1.	1.	1.	Oracle	Relational DBMS	1462.02	-5.51	+19.93
2.	2.	2.	MySQL	Relational DBMS	1371.83	+1.72	+77.56
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1142.82	+7.77	+11.79
4.	4.	4.	MongoDB	Document store	320.22	+7.78	+42.90
5.	5.	5.	PostgreSQL	Relational DBMS	307.61	+3.89	+34.09
6.	6.	6.	DB2	Relational DBMS	185.96	+1.87	-15.09
7.	↑ 8.	↑ 8.	Cassandra	Wide column store	134.50	+4.83	+27.95
8.	↓ 7.	↓ 7.	Microsoft Access	Relational DBMS	131.58	-0.39	-14.00
9.	9.	↑ 10.	Redis	Key-value store	108.24	-3.00	+13.51
10.	10.	↓ 9.	SQLite	Relational DBMS	107.26	-0.70	+2.10
11.	11.	↑ 14.	Elasticsearch	Search engine	86.31	+3.73	+21.48
12.	↑ 13.	↑ 13.	Teradata	Relational DBMS	73.74	+1.48	+3.62
13.	↓ 12.	↓ 11.	SAP Adaptive Server				
14.	14.	↓ 12.	Solr				
15.	15.	15.	HBase				
16.	16.	↑ 17.	Hive				
17.	17.	↓ 16.	FileMaker				
18.	18.	18.	Splunk				
19.	19.	↑ 21.	SAP HANA				
20.	↑ 21.	↑ 25.	MariaDB				
21.	↓ 20.	↑ 22.	Neo4j				
22.	22.	↓ 19.	Informix				
23.	23.	↓ 20.	Memcached				
24.	24.	24.	Couchbase				
25.	25.	↑ 30.	Amazon DynamoDB				

http://db-engines.com/en/ranking

RECAP

Recap

- Relational Databases don’t solve everything
  - SQL and ACID add overhead
  - Distribution not so easy

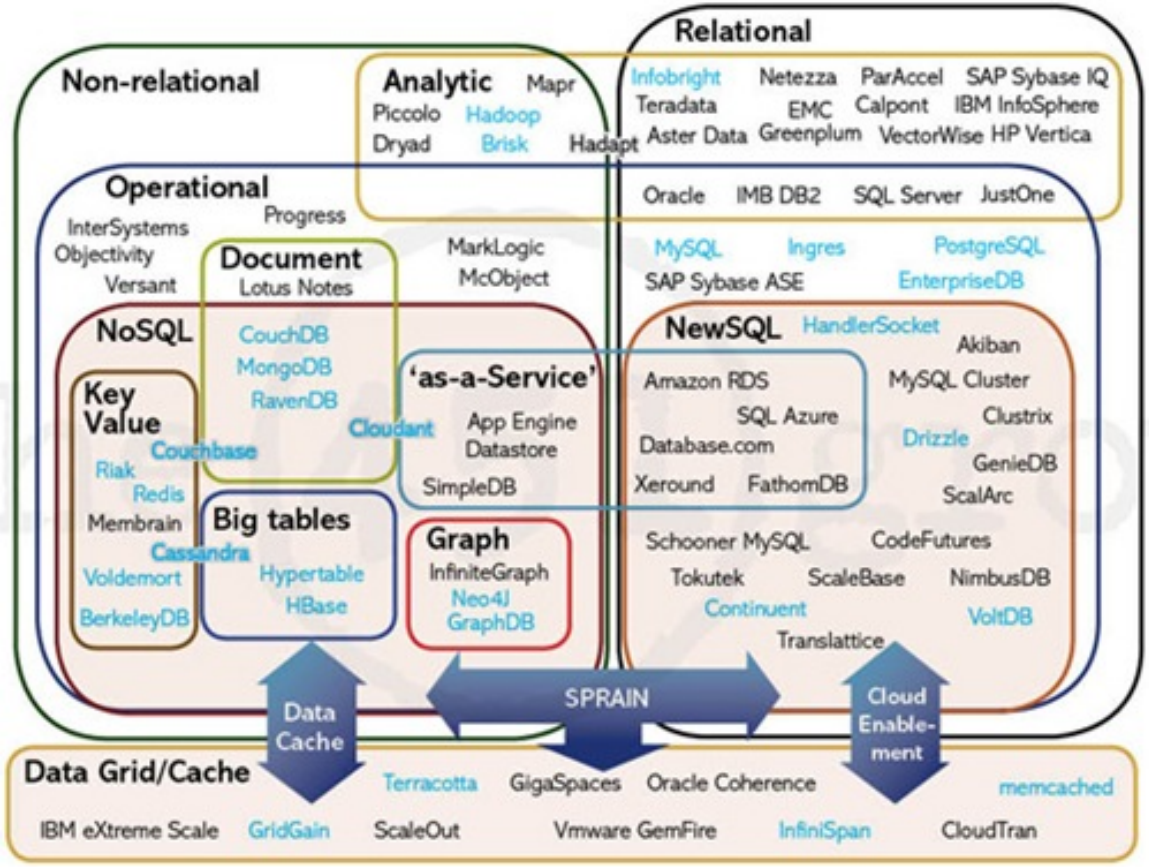
- NoSQL: what if you don’t need SQL or ACID?
  - Something simpler
  - Something more scalable
  - Trade efficiency against guarantees

NoSQL: Trade-offs

- **Simplified transactions** (no ACID)
- **Simplified** (or no) **query language**
  - Procedural *or* a subset of SQL
- **Simplified query alegbra**
  - Often no joins
- **Simplified data model**
  - Often map-based
- **Simplified replication**
  - Consistency vs. Availability

Simplifications enable scale to thousands of machines. But a lot of relational database features are lost!

NoSQL Overview Map



Types of NoSQL Store

- **Key–Value Stores** (e.g., Dynamo):
  - Distributed unsorted maps
  - Some have secondary indexes
- **Document Stores** (e.g., MongoDB):
  - Map values are documents (e.g., JSON, XML)
  - Built-in document functions/indexable fields
- **Table/Column-Based Stores** (e.g., Bigtable):
  - Distributed multi-dimensional sorted maps
  - Distribution by Tablets/Column-families
- **Graph Stores** (e.g., Neo4J)
  - Stores vertices and relations: Index-free adjacency
  - Query languages for paths, reachability, etc.
- **Hybrid/mix/other**

Categories are far from clean. But aside from graph stores, most NoSQL stores are just fancy (distributed) maps. ☺

Bigtable

- Column family store: (row, column, time) → value
- Sorted map, range partitioned
- PAXOS for locks, root table
- Tablets: horizontal table splits
- Column family: logical grouping of columns stored close together
- Locality groups: grouping of column families
- SSTable: sequence of 64k blocks
- Batch writes
- Compactions: merge SSTables

Graph stores

- Indexes on nodes, relations, (attributes)
- Relational-style queries
- Path-style queries
- Index-free adjacency
- Neo4J:
  - Property graph data model
  - Cypher query language

Questions