Programming and frameworks for ML

Introduction to NoSQL Databases

About Me

Big Data Consultant at Indra / Big Data Lecturer

- More than 20 years of experience in different environments, technologies, customers, countries ...
- Passionate data and technology
- Enthusiastic Big Data world and NoSQL

Daniel Villanueva Jiménez

BigData Developer / Lecturer

INDRA • Universidad Pontificia de Salamanca

Objectives

- Tour of different database models
- Comparison of a relational database with NoSql databases
  - Key/Value
  - Documents
  - Column oriented
  - Graphs

Database Workshop

- Material
- Use case
- Relational Databases
- NoSQL
- Riak
- MongoDB
- Apache Cassandra
- Neo4j

Material - Virtual Machine

http://localhost:8001/

4.2.5

http://localhost:3100/

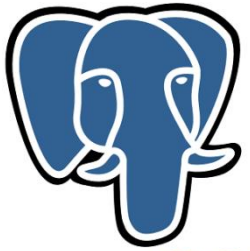
3.5.11

http://localhost:7474/

2.2.3

http://localhost:8098/

http://localhost:2222/



12.2

3.11

learner/learner

https://github.com/dvillaj/NoSQL-box

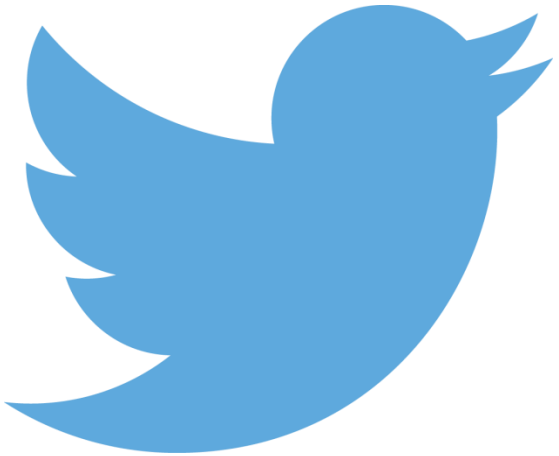
What are databases?

What are databases?

"A database is a **storehouse** that allows us to store **large amounts of information in an organized** manner so that we can easily **find** and **use it."**

Database Workshop

- Material
- Use case
- Relational Databases
- NoSQL
- Riak
- MongoDB
- Apache Cassandra
- Neo4j



Case Study - Black Cards

Gastos de los exdirectivos de Caja Madrid, uno a uno, con las 'tarjetas negras' (tabla)

CUARTOPODER | Publicado: 11/10/2014 07:59 - Actualizado: 16/5/2017 11:14

El juez de la Audiencia Nacional, Fernando Andréu, facilitó ayer a las partes el detalle de los pagos realizados con las llamadas 'tarjetas negras' por 86 exdirectivos de Caja Madrid: 15,5 millones de euros en total.

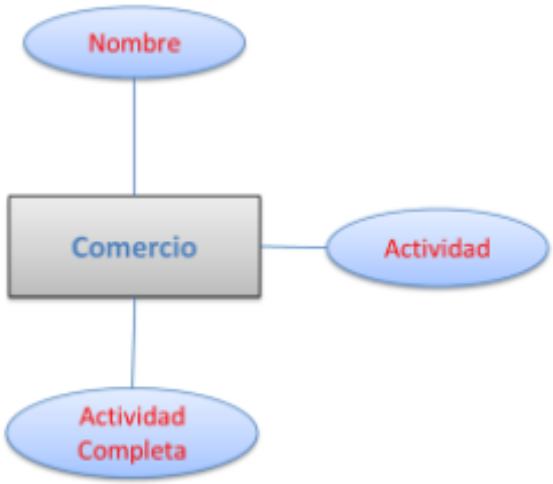
El informe de los gastos -86 tablas de Excel con miles de datos- ha sido realizado por Bankia y remitido a la fiscalía del caso ante la posibilidad de que quienes utilizaron las tarjetas pudieran haber cometido hechos delictivos.

cuartopoder ha desglosado la información de los gastos realizados por los exdirectivos de la entidad, uno por uno, para facilitar la consulta de los datos en esta tabla.

https://www.cuartopoder.es/

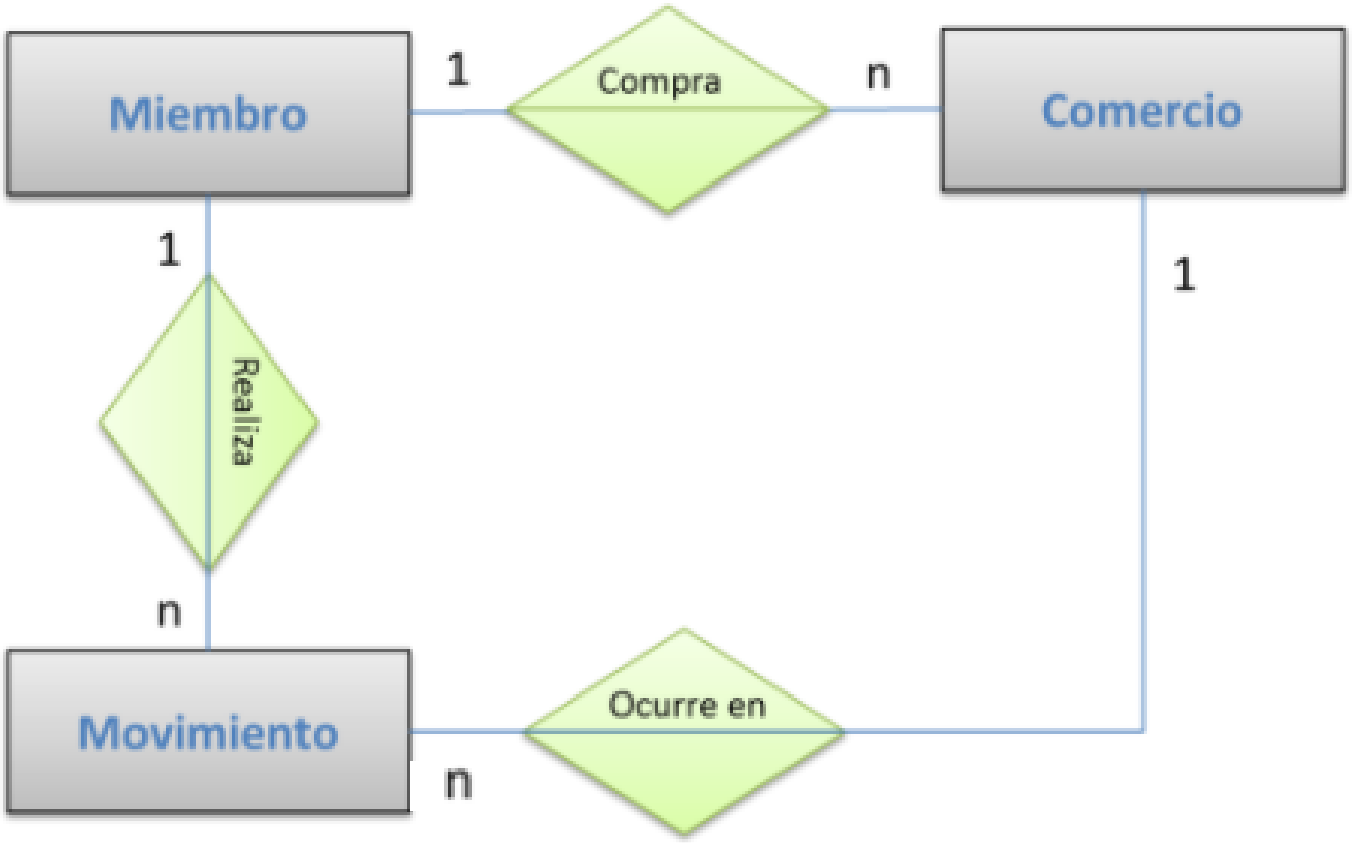
Conceptual model - Miembro

Conceptual model - Comercio



Conceptual Model - Movimiento

Entity Relationship Model



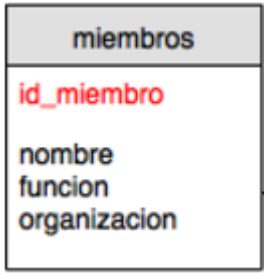
Database Workshop

- Material
- Use case
- Relational Databases
- NoSQL
- Riak
- MongoDB
- Apache Cassandra
- Neo4j

Elements of a relational database

- Tables
- Fields (or Columns)
- Primary Keys


```
CREATE TABLE miembros (
    id_miembro      int not null PRIMARY KEY,
    nombre          varchar(200),
    funcion         varchar(40),
    organizacion    varchar(200),
    CONSTRAINT pk_miembros UNIQUE(id_miembro)
);
```



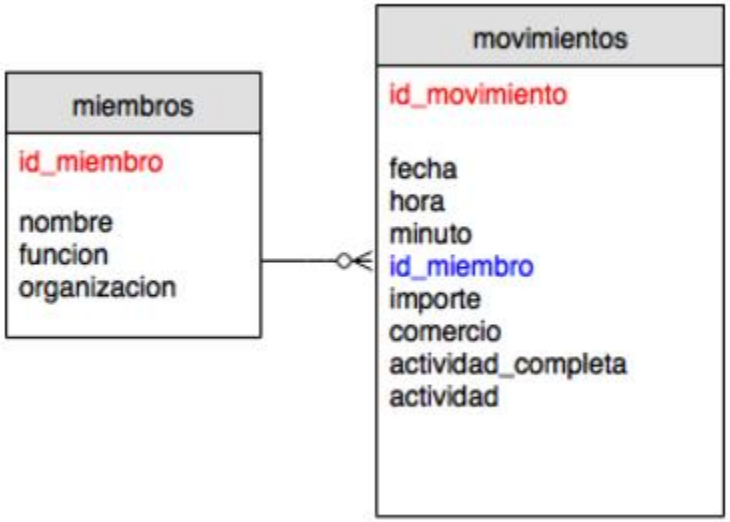
Elements of a relational database

- Records (or Rows)

id_miembro	nombre	funcion	organizacion
1	Alberto Recarte García Andrade	concejal	Partido Popular
2	Alejandro Couceiro Ojeda	concejal	CEIM
83	Ángel Eugenio Gómez del Pulgar Perales	concejal	PSOE
3	Angel Rizaldos González	concejal	Izquierda Unida
4	Antonio Cámara Eguinoa	concejal	Partido Popular
5	Antonio Rey de Viñas Sánchez-Majestad	concejal	CC OO
6	Antonio Romero Lázaro	concejal	PSOE
7	Arturo Luis Fernández Álvarez	concejal	CEIM
8	Beltrán Gutiérrez Moliner	concejal	Partido Popular
12	Cándido Cerón Escudero	concejal	Partido Popular

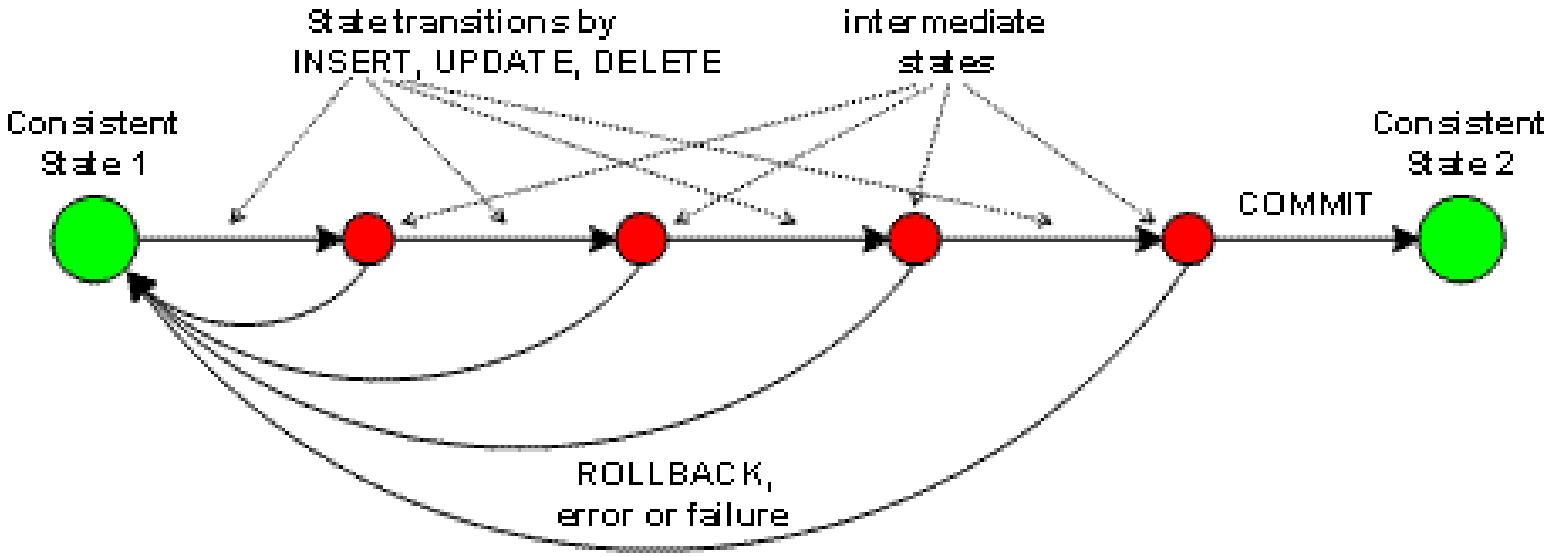
Elements of a relational database

- Relationships between tables
- Foreign Keys



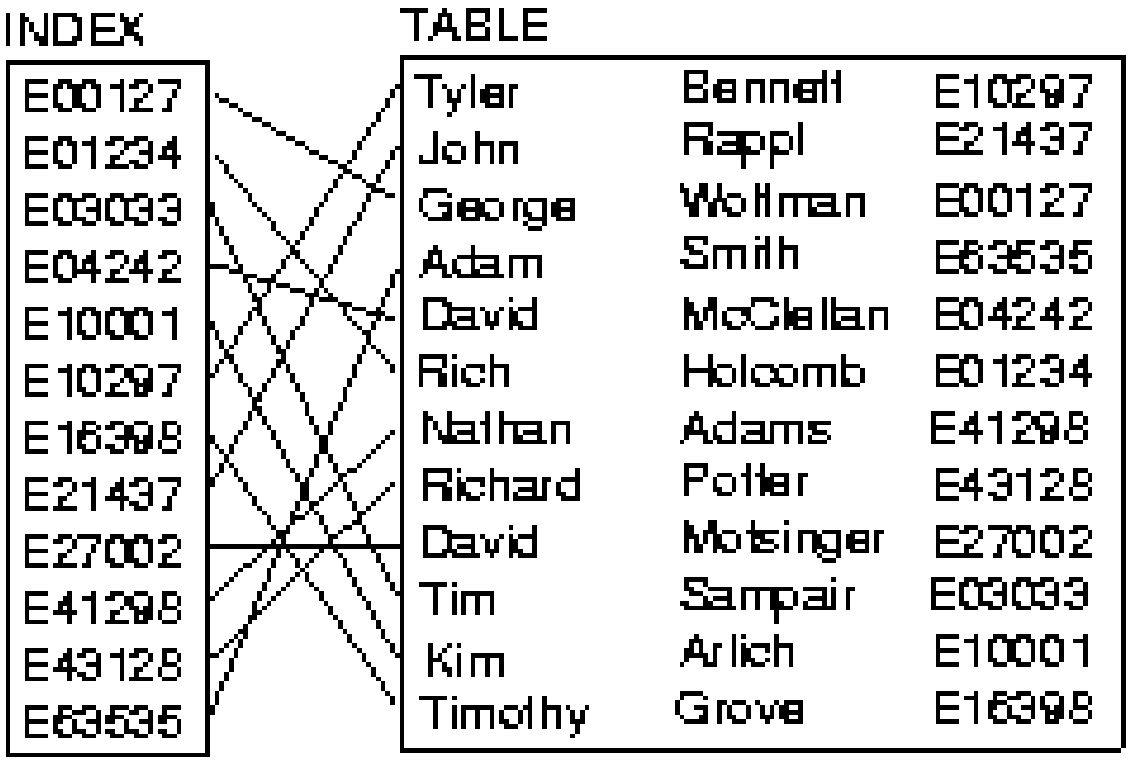
Elements of a relational database

- Transactions



Elements of a relational database

- Indexes



Elements of a relational database

- SQL Language

```
CREATE TABLE users (
    user_id          bigint not null PRIMARY KEY,
    screen_name      varchar(50) not null,
    name             varchar(50) null,
    created_at       timestamp with time zone null,
    description      varchar(200) null,
    retweet_count    int null,
    favorite_count   int null,
    friends_count    int null,
    followers_count  int null,
    statuses_count   int null,
    geo_enabled      boolean null,
    time_zone        varchar(50) null,
    profile_image_url varchar(300) null
);
```

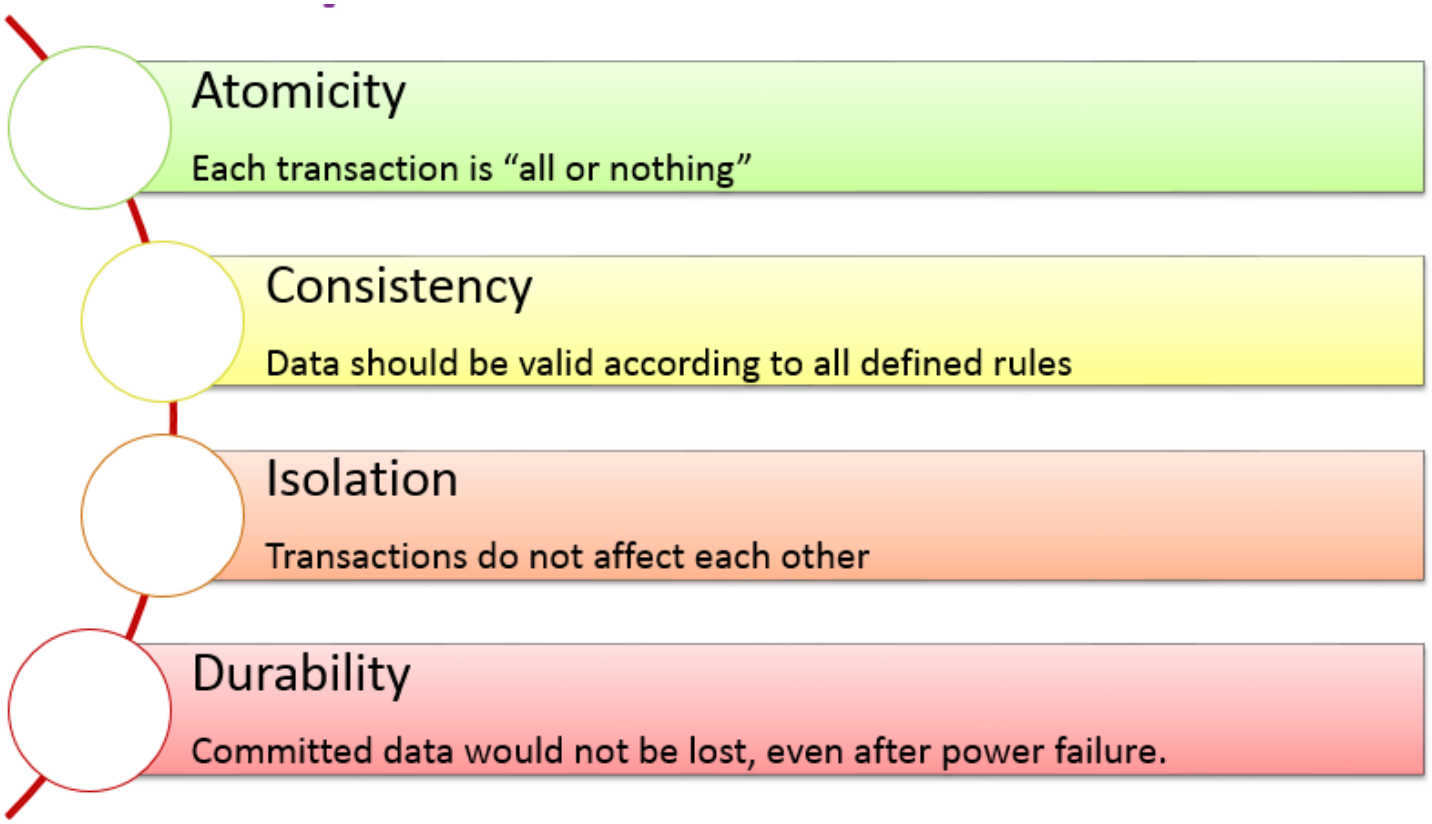
```
select user_id, count(*) as count
from tweet_usermention
group by 1
having count(*) > 0
order by 2 desc
```

```
CREATE INDEX id_user_screenname
ON users (screen_name)
```

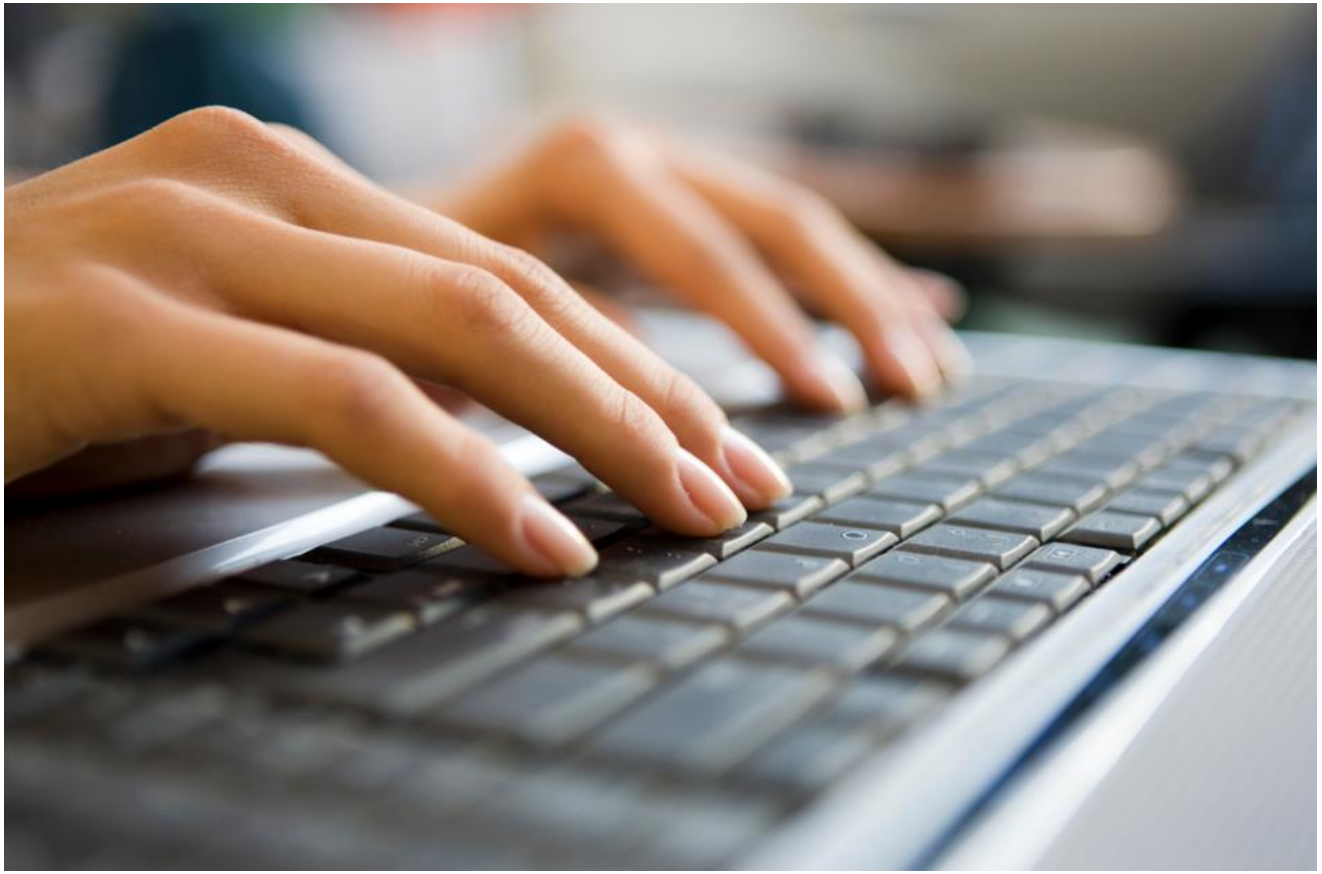
```
SELECT * FROM tweets
WHERE geo_type is not null
LIMIT 10
```

```
DELETE FROM users
WHERE user_id = 2012312
```

ACID properties associated to a Relational database



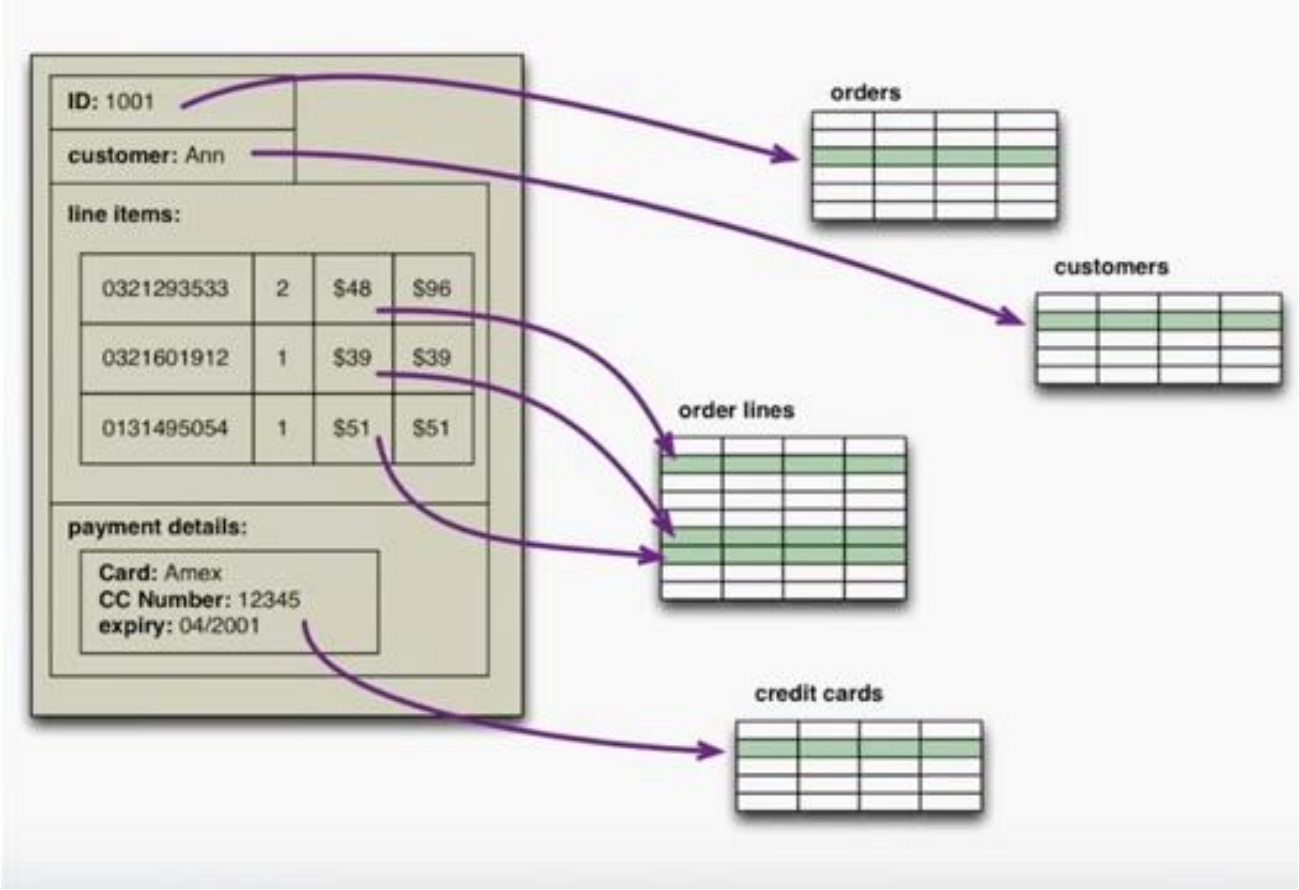
PostgreSQL - HandsOn



Cheat Sheet

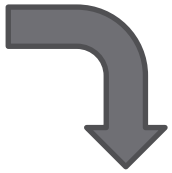
Problems?

Impedance Mismatch

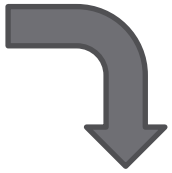


Rigid schemes

```
CREATE TABLE Customers (
    Customer_Id Int,
    Name Varchar(100),
    ...
)
```



```
BULK INSERT Customers
FROM 'CustFile.txt'
WITH FIELDTERMINATOR = ';'
```

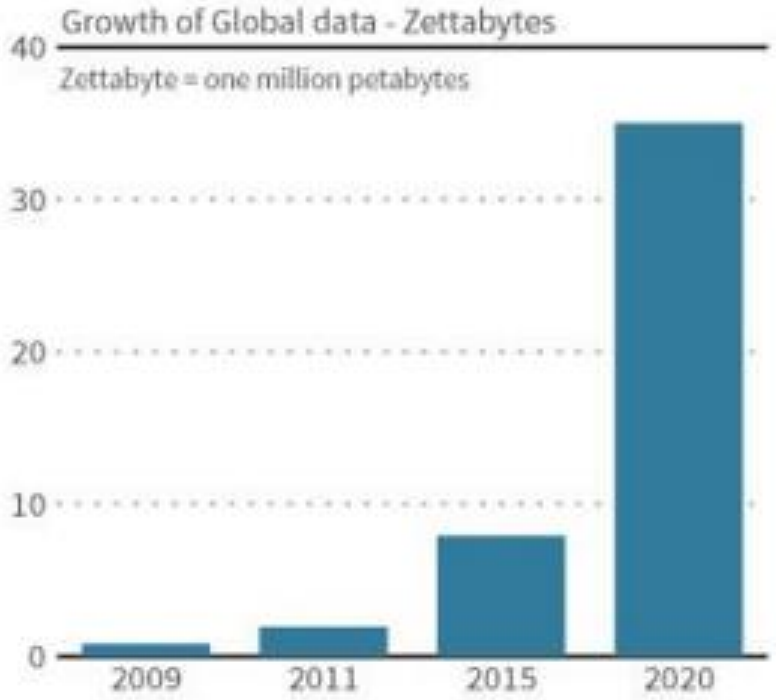


```
SELECT Customer_Id, Name
FROM Customers
```

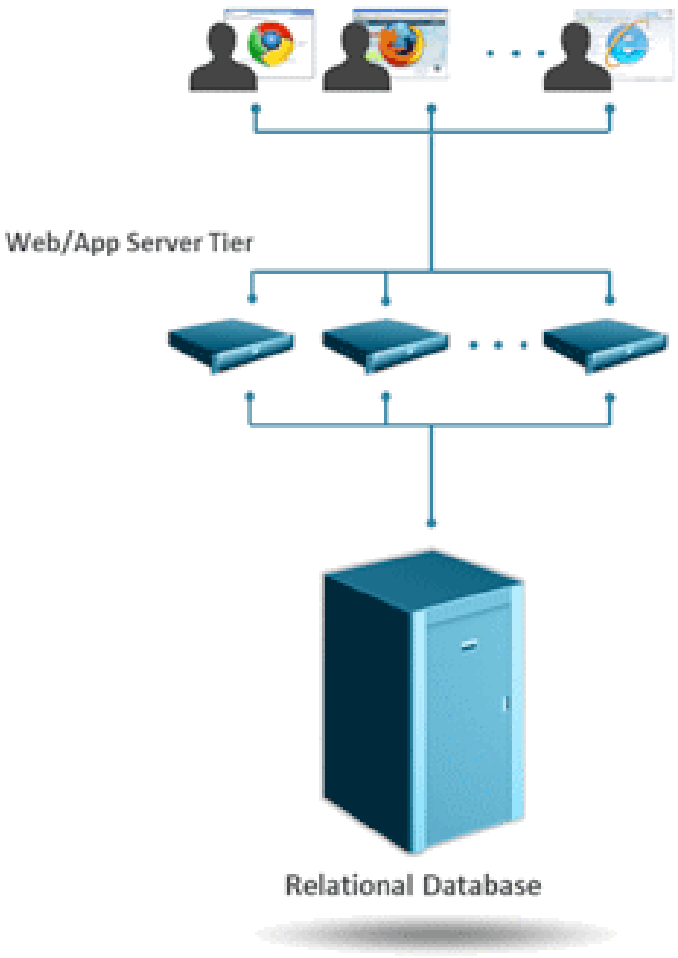
Rigid schemes

- You cannot load the information until you create the structure in the database
- You cannot create the structure until you understand the schema to be stored in the table
- What happens if the data changes?

Volume

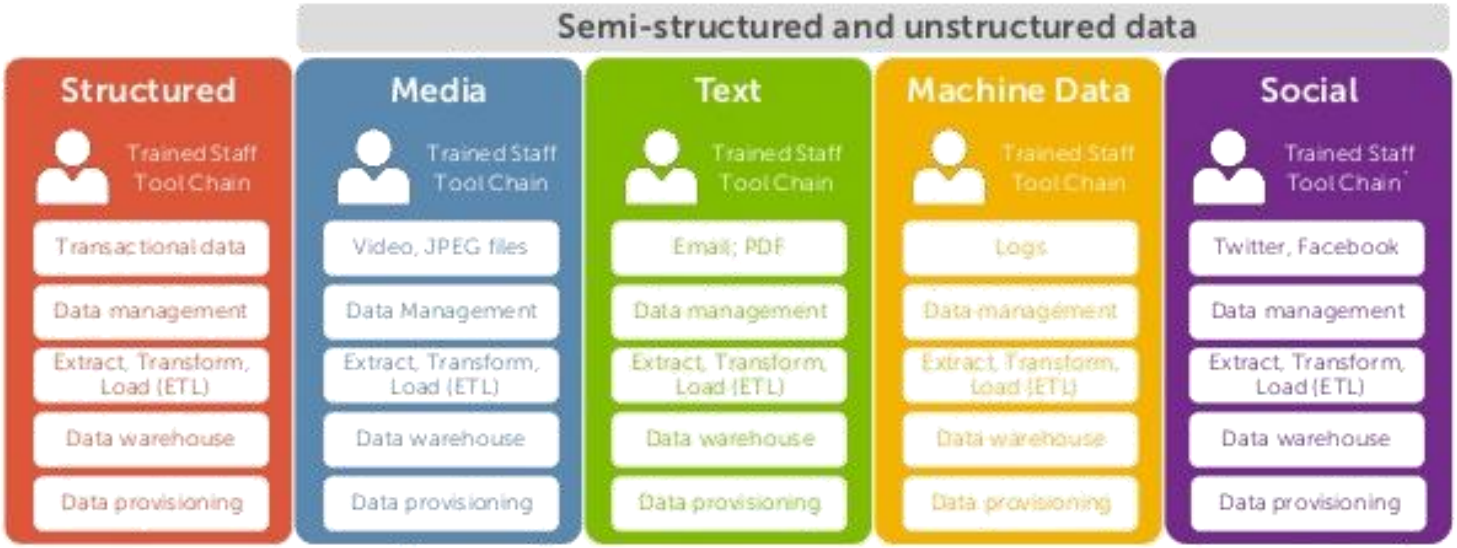

Growth of Global data - Zettabytes
Zettabyte = one million petabytes
40
30
20
10
0
2009
2011
2015
2020

Scalability



Scalability

Variety of information



Structured Data

model	mpg	cyl	disp	hp	drat
Mazda RX4	21	6	160	110	3.9
Mazda RX4 Wag	21	6	160	110	3.9
Datsun 710	22.8	4	108	93	3.85
Hornet 4 Drive	21.4	6	258	110	3.08
Hornet Sportabout	18.7	8	360	175	3.15
Valiant	18.1	6	225	105	2.76
Duster 360	14.3	8	360	245	3.21
Merc 240D	24.4	4	146.7	62	3.69
Merc 230	22.8	4	140.8	95	3.92
Merc 280	19.2	6	167.6	123	3.92
Merc 280C	17.8	6	167.6	123	3.92
Merc 450SE	16.4	8	275.8	180	3.07
Merc 450SL	17.3	8	275.8	180	3.07

Semi-Structured Data

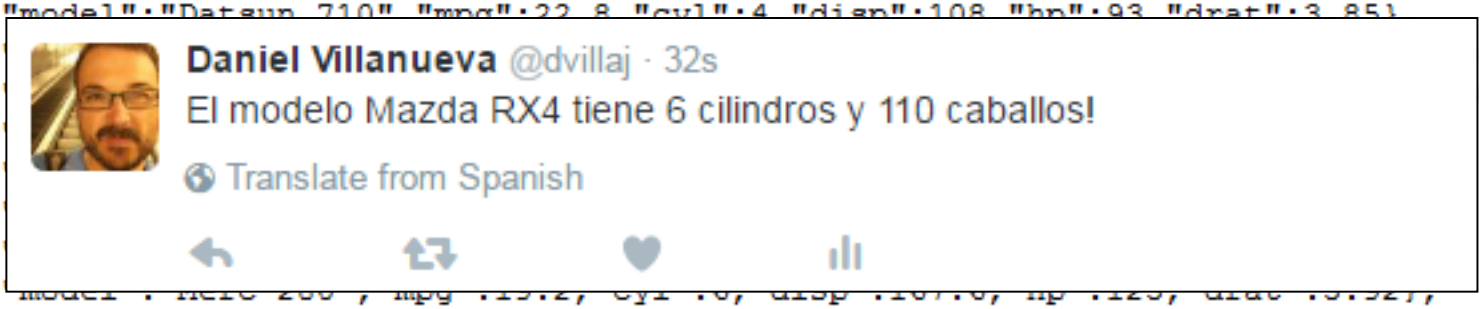
model	mpg	cyl	disp	hp	drat
Mazda RX4	21	6	160	110	3.9
Mazd					
Datsu					
Horne					
Horne					
Valia					
Duste					
Merc					
Merc					
Merc					
Merc					
Merc					
Merc					


```
[
    {"model":"Mazda RX4","mpg":21,"cyl":6,"disp":160,"hp":110,"drat":3.9},
    {"model":"Mazda RX4 Wag","mpg":21,"cyl":6,"disp":160,"hp":110,"drat":3.9},
    {"model":"Datsun 710","mpg":22.8,"cyl":4,"disp":108,"hp":93,"drat":3.85},
    {"model":"Hornet 4 Drive","mpg":21.4,"cyl":6,"disp":258,"hp":110,"drat":3.08},
    {"model":"Hornet Sportabout","mpg":18.7,"cyl":8,"disp":360,"hp":175,"drat":3.15},
    {"model":"Valiant","mpg":18.1,"cyl":6,"disp":225,"hp":105,"drat":2.76},
    {"model":"Duster 360","mpg":14.3,"cyl":8,"disp":360,"hp":245,"drat":3.21},
    {"model":"Merc 240D","mpg":24.4,"cyl":4,"disp":146.7,"hp":62,"drat":3.69},
    {"model":"Merc 230","mpg":22.8,"cyl":4,"disp":140.8,"hp":95,"drat":3.92},
    {"model":"Merc 280","mpg":19.2,"cyl":6,"disp":167.6,"hp":123,"drat":3.92},
    {"model":"Merc 280C","mpg":17.8,"cyl":6,"disp":167.6,"hp":123,"drat":3.92},
    {"model":"Merc 450SE","mpg":16.4,"cyl":8,"disp":275.8,"hp":180,"drat":3.07},
    {"model":"Merc 450SL","mpg":17.3,"cyl":8,"disp":275.8,"hp":180,"drat":3.07},
]
```


Unstructured Data

model	mpg	cyl	disp	hp	drat
Mazda RX4	21	6	160	110	3.9
Mazd					
Datsu					
Horne					
Horne					
Valia					
Duste					
Merc					
Merc					
Merc					
Merc					
Merc					
Merc					

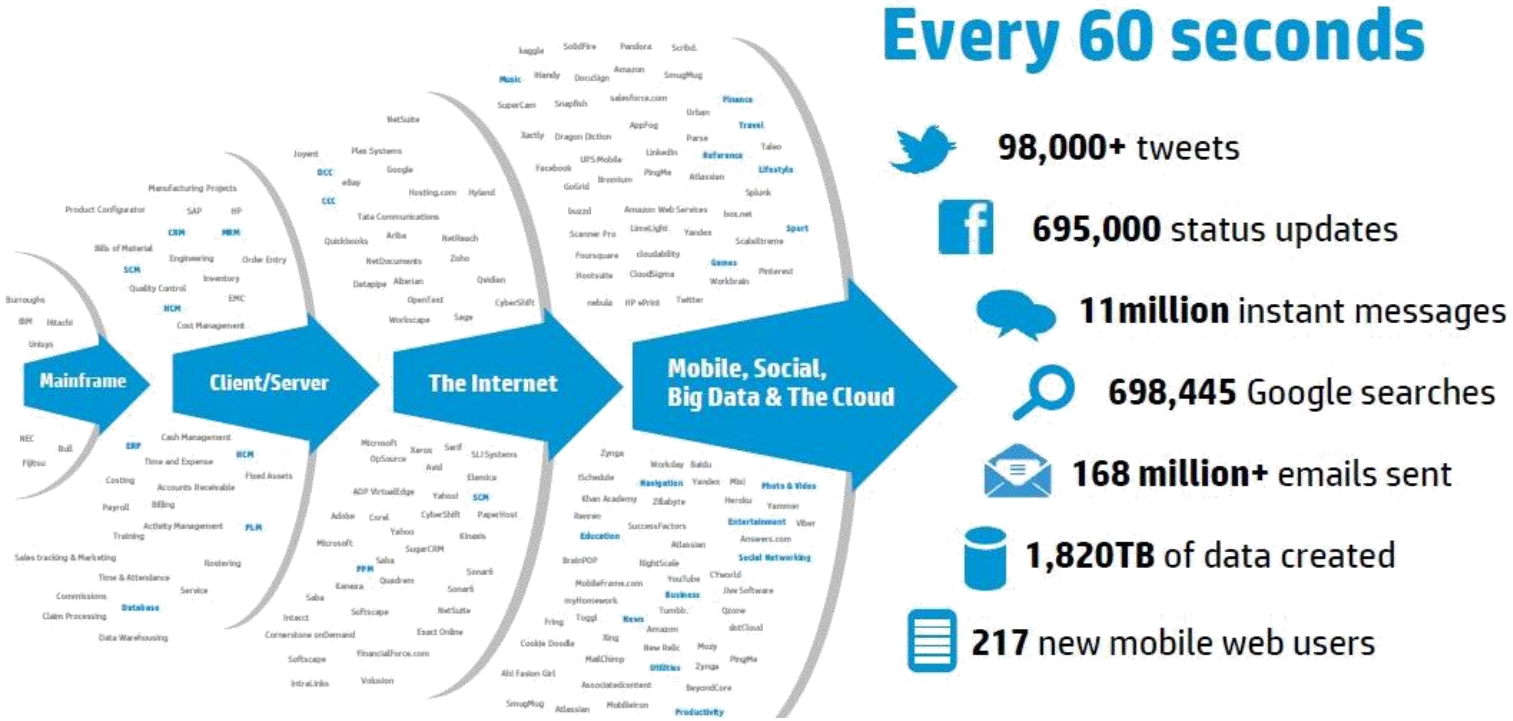

```
[
    {"model":"Mazda RX4","mpg":21,"cyl":6,"disp":160,"hp":110,"drat":3.9},
    {"model":"Mazda RX4 Wag","mpg":21,"cyl":6,"disp":160,"hp":110,"drat":3.9},
    {"model":"Datsun 710","mpg":22.8,"cyl":4,"disp":108,"hp":93,"drat":3.85},
    {"model":"Merc 280C","mpg":17.8,"cyl":6,"disp":167.6,"hp":123,"drat":3.92},
    {"model":"Merc 450SE","mpg":16.4,"cyl":8,"disp":275.8,"hp":180,"drat":3.07},
    {"model":"Merc 450SL","mpg":17.3,"cyl":8,"disp":275.8,"hp":180,"drat":3.07},
]
```

Velocity

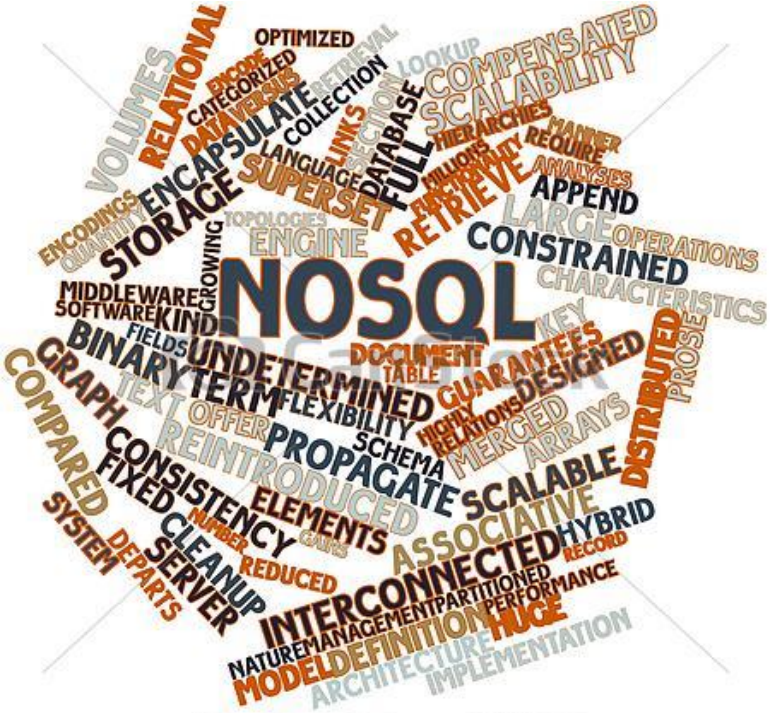
Tweeting

Velocity


Mainframe
Client/Server
The Internet
Mobile, Social, Big Data & The Cloud
Every 60 seconds
98,000+ tweets
695,000 status updates
11million instant messages
698,445 Google searches
168 million+ emails sent
1,820TB of data created
217 new mobile web users

Database Workshop

- Material
- Use case
- Relational Databases
- NoSQL
- Riak
- Apache Cassandra
- MongoDB
- Neo4j

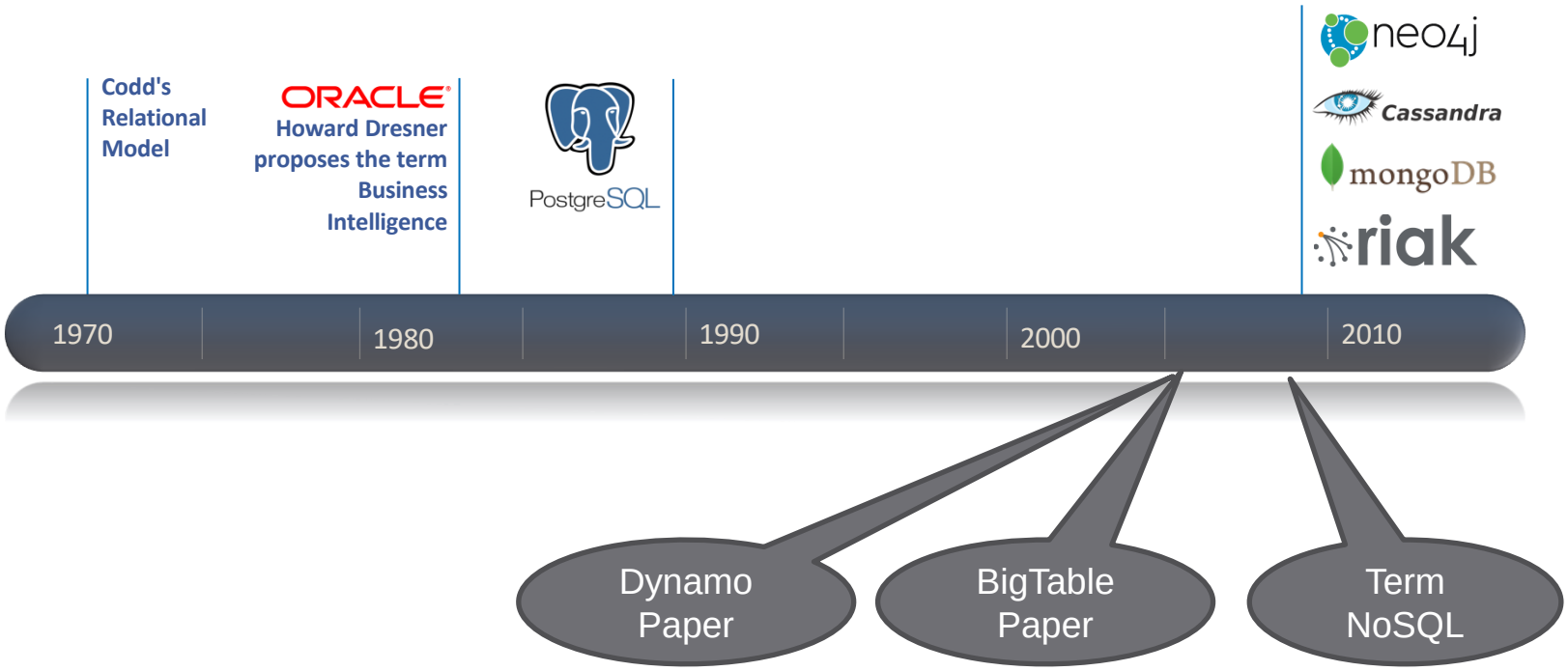


NoSQL

NoSQL is a broad class of **database management systems** that differs from the classic model of the relational database management system

- They usually **scale** well horizontally
- Do not use **SQL** as the main query language
- Stored data does not require **fixed structures** such as tables
- Normally do not support **JOIN** operations
- Not fully guaranteed by **ACID**
- Many of them are **Open Source**

A little bit of history



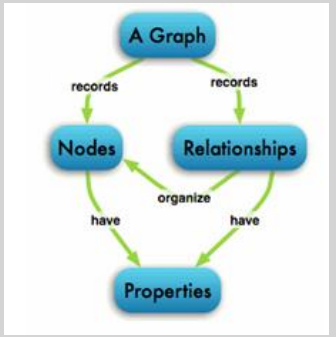
Types of NoSQL databases

Key / Value	Columnar	Documents	Graphs

Key	Value
1	New York
2	Boston
3	Mexico
4	Kansas
5	Detroit
6	California

101	email	name	tel	
	ab@c.to	otto	12345	
103	email	name	tel	tel2
	karl@a.b	karl	6789	12233
104	name			
	linda			

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [],
  "spouse": null
}
```



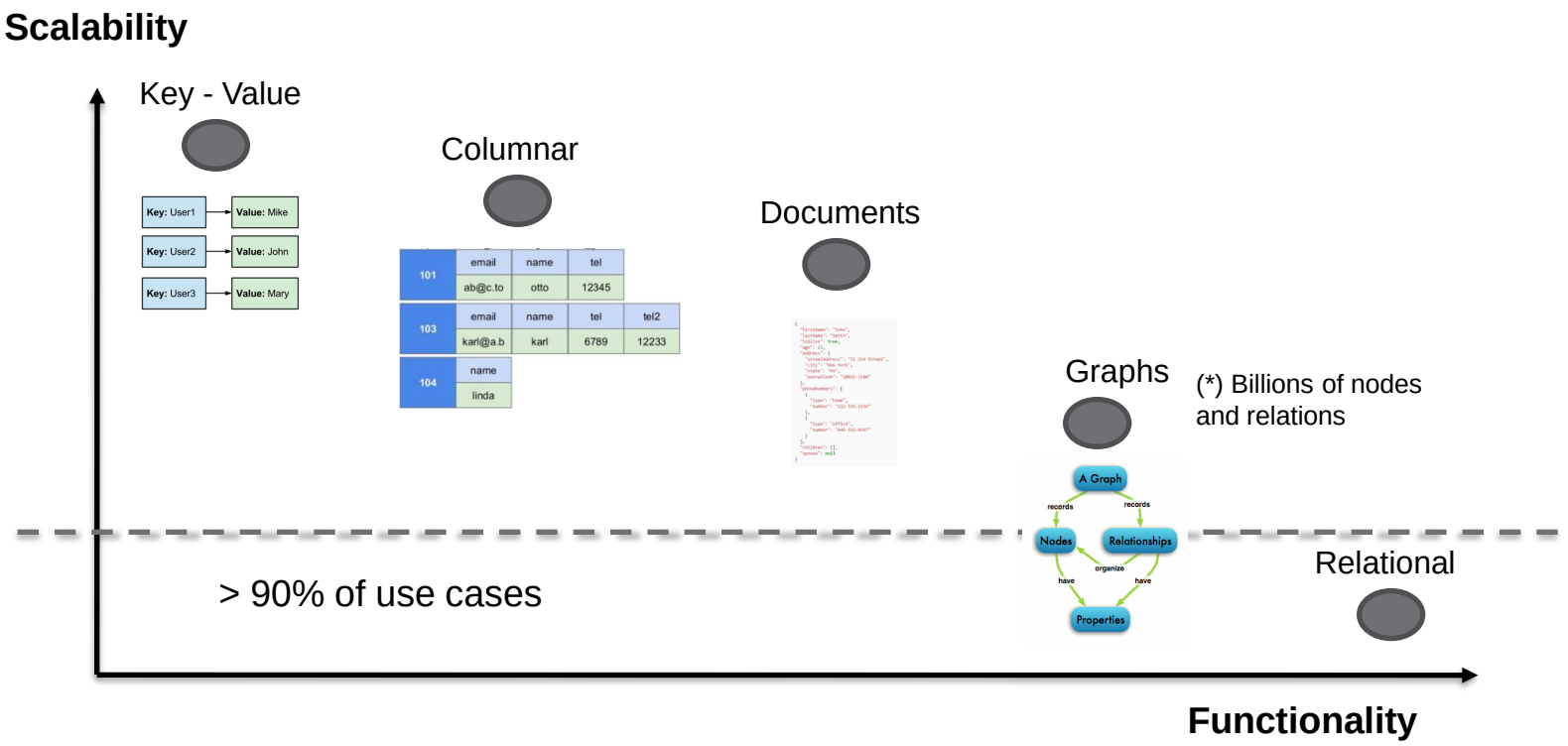
riak

Cassandra

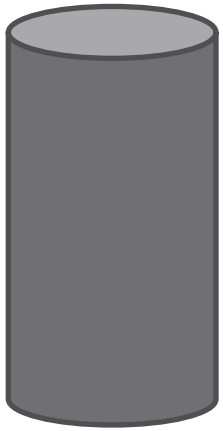
mongoDB

neo4j

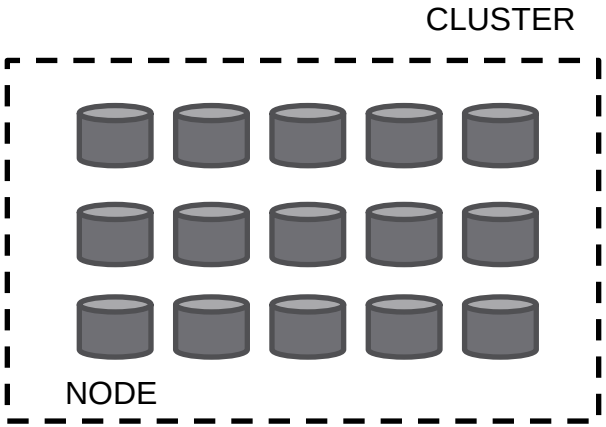
Size vs. Functionality



Vertical and horizontal scaling



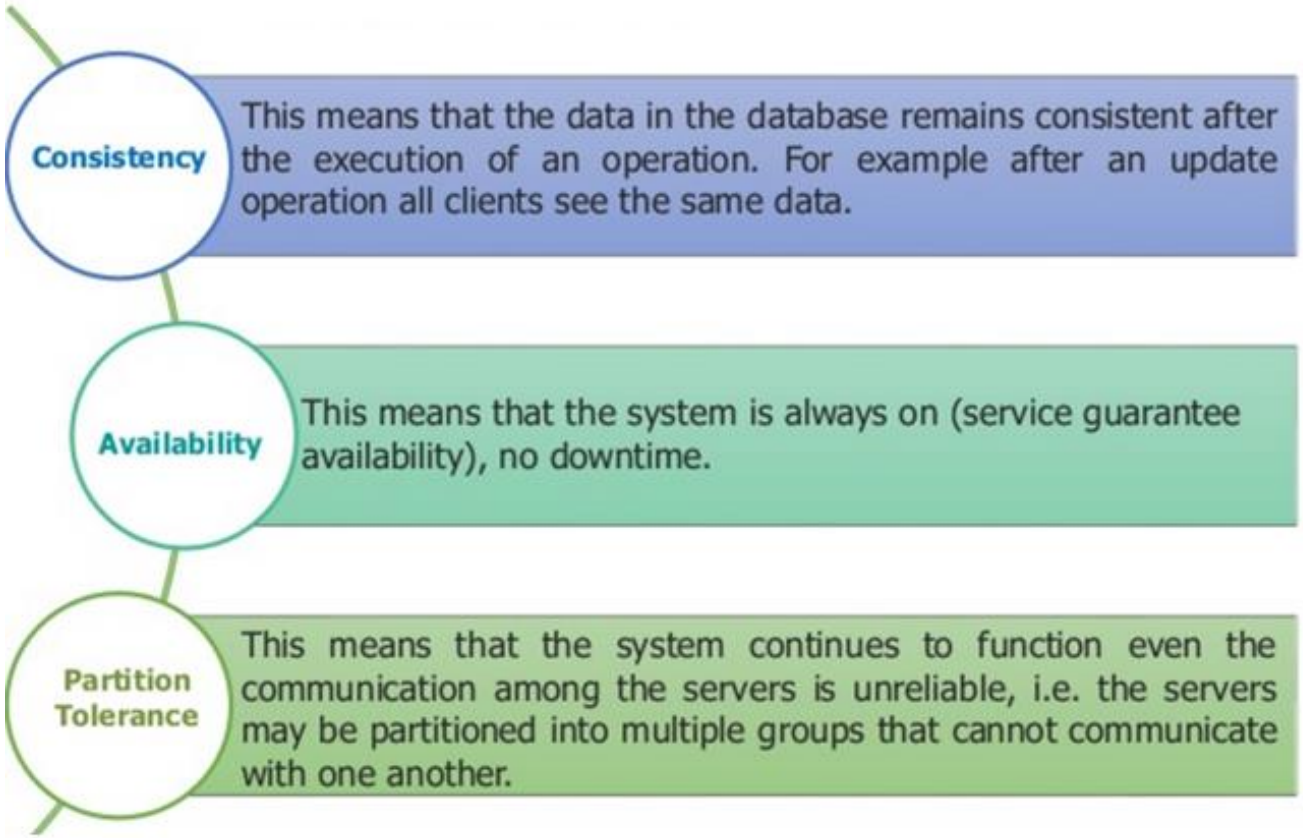
VERTICAL SCALING



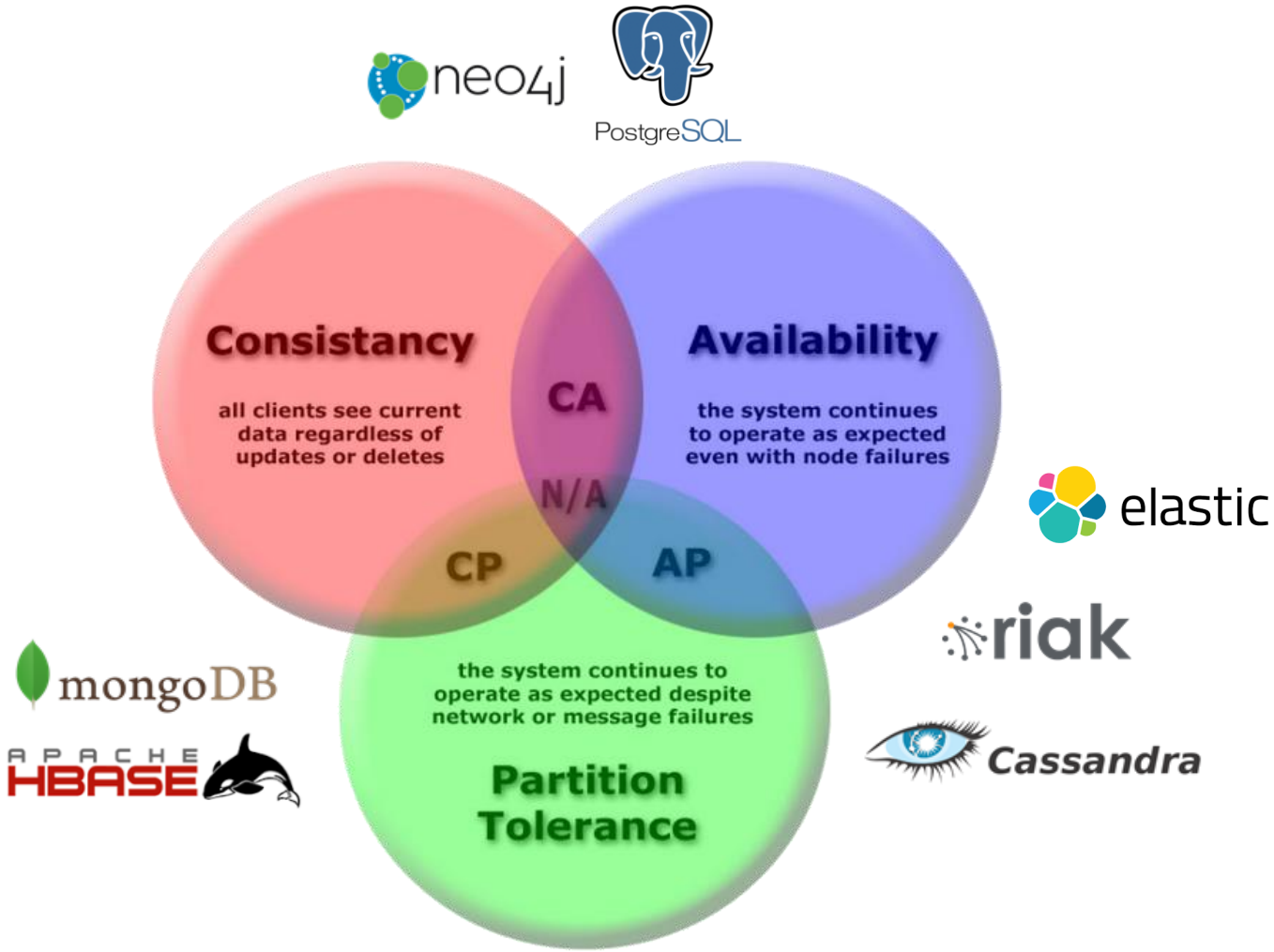
HORIZONTAL SCALING

CAP Theorem

- Requirements for distributed databases



CAP Theorem


neo4j
PostgreSQL
Consistency
all clients see current data regardless of updates or deletes
Availability
the system continues to operate as expected even with node failures
CA
N/A
CP
AP
the system continues to operate as expected despite network or message failures
Partition Tolerance
elastic
riak
Cassandra
mongoDB
APACHE HBASE

Database Workshop

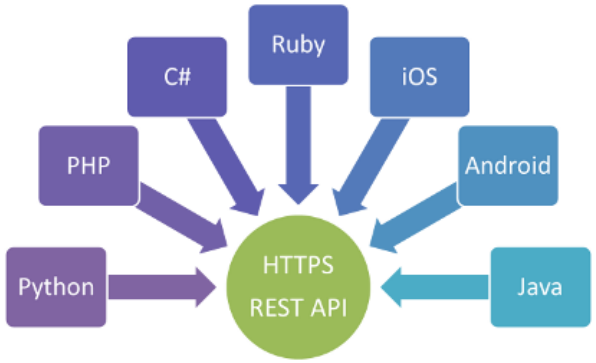
- Material
- Use case
- Relational Databases
- NoSQL
- Riak
- Apache Cassandra
- MongoDB
- Neo4j

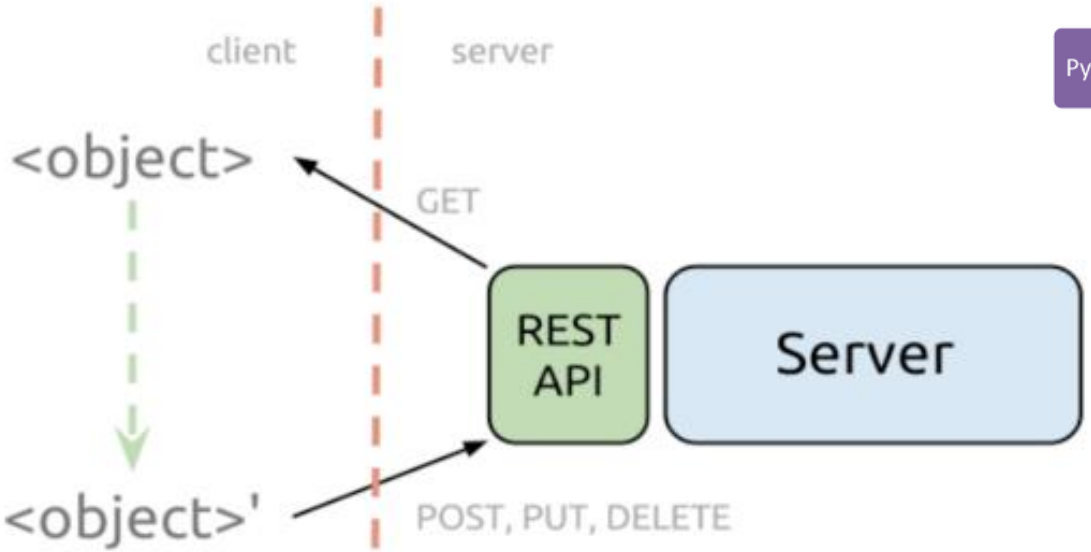
Key	Value
1	New York
2	Boston
3	Mexico
4	Kansas
5	Detroit
6	California

Riak

- Developed by Basho Technologies in Erlang
- Inspired by Amazon Dynamo
- Horizontal Distribution - Fault Tolerant
- Prioritizes availability - Tunable consistency
- No master node - No single point of failure
- Querys - Provides a REST API over HTTP
- Drivers in multiple languages - Java, Python, Ruby, etc.
- Storage options - Memory, disk or both.

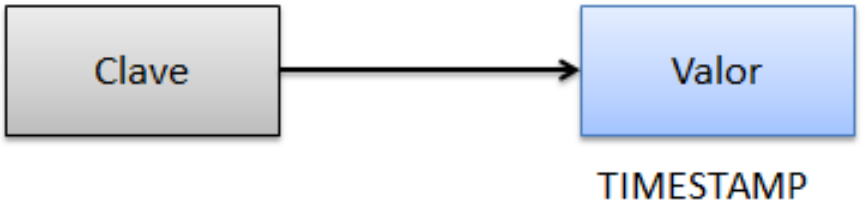
API REST


Ruby
C#
iOS
PHP
Android
HTTPS
REST API
Python
Java


client
server
<object>
GET
REST
API
Server
<object>'
POST, PUT, DELETE

Key and Value

- It's the most basic structure

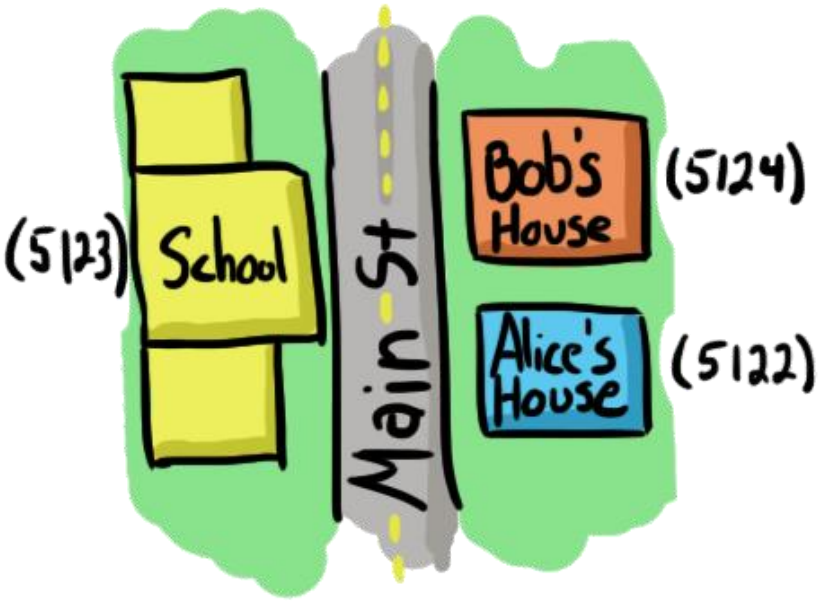


Key and Value

- For example:
  - Key: Address
  - Value: Tenant

```
hashtable = {}
hashtable["5124"] = "Bob"

print hashtable["5124"]
```

```
Bob
```



Buckets

- They allow to separate the same key according to a context
- Example: Streets

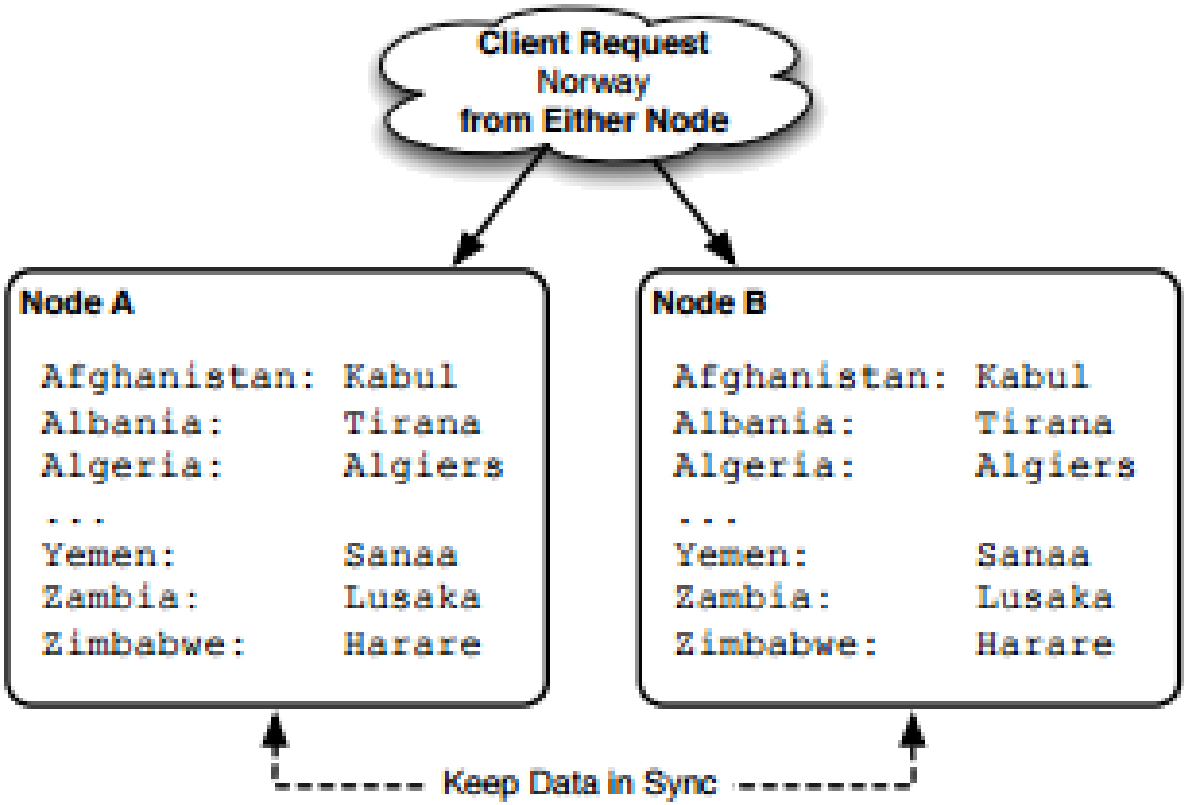
```
Street23rd = {}
Street6thAve = {}

Street23rd["5124"] = "Bob"
Street6thAve["5124"] = "Sara"
```

Data distribution in Riak

- Riak is kept available by distributing the data between different nodes
- There are 2 styles of layout ...

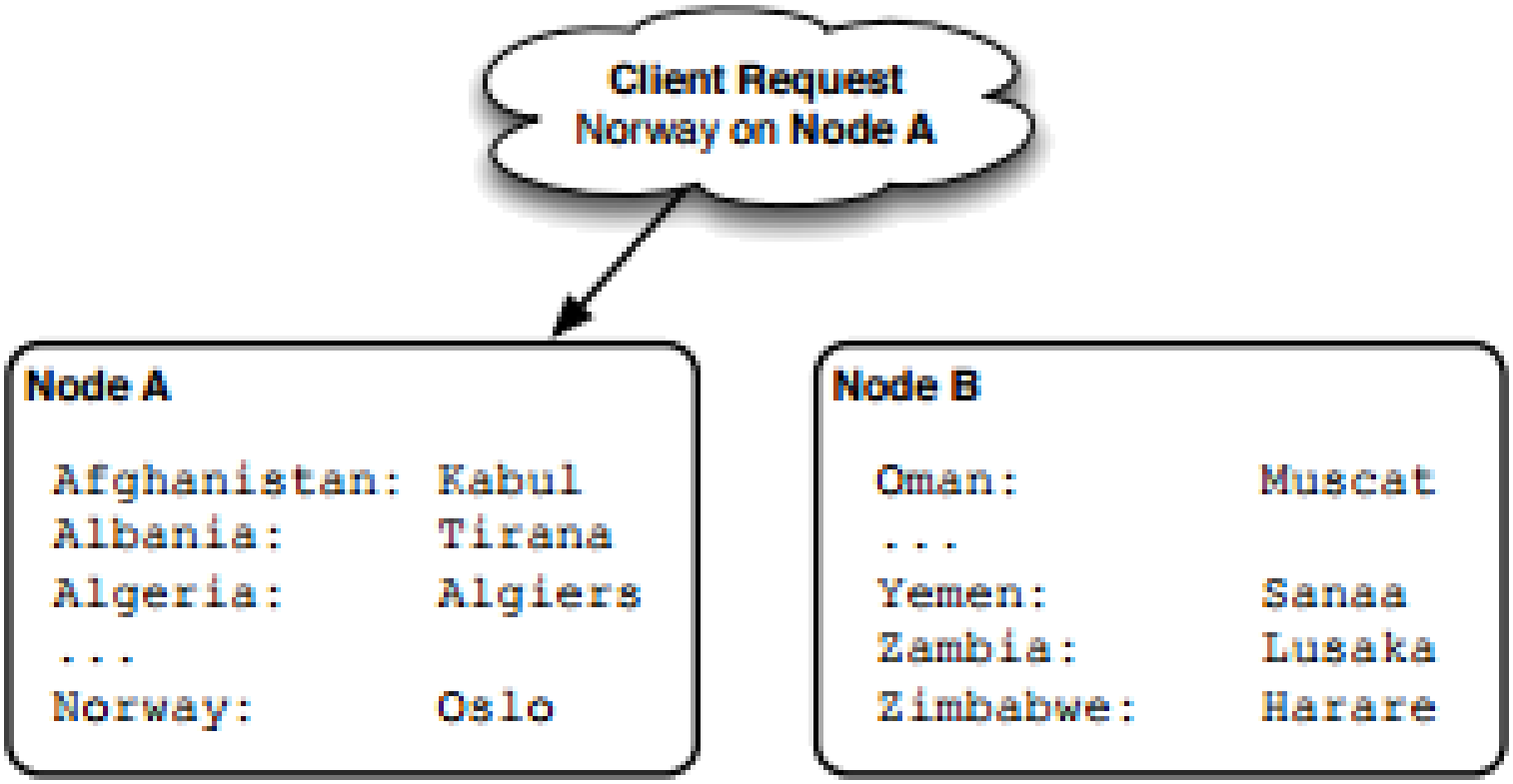
Data distribution in Riak

- Replication



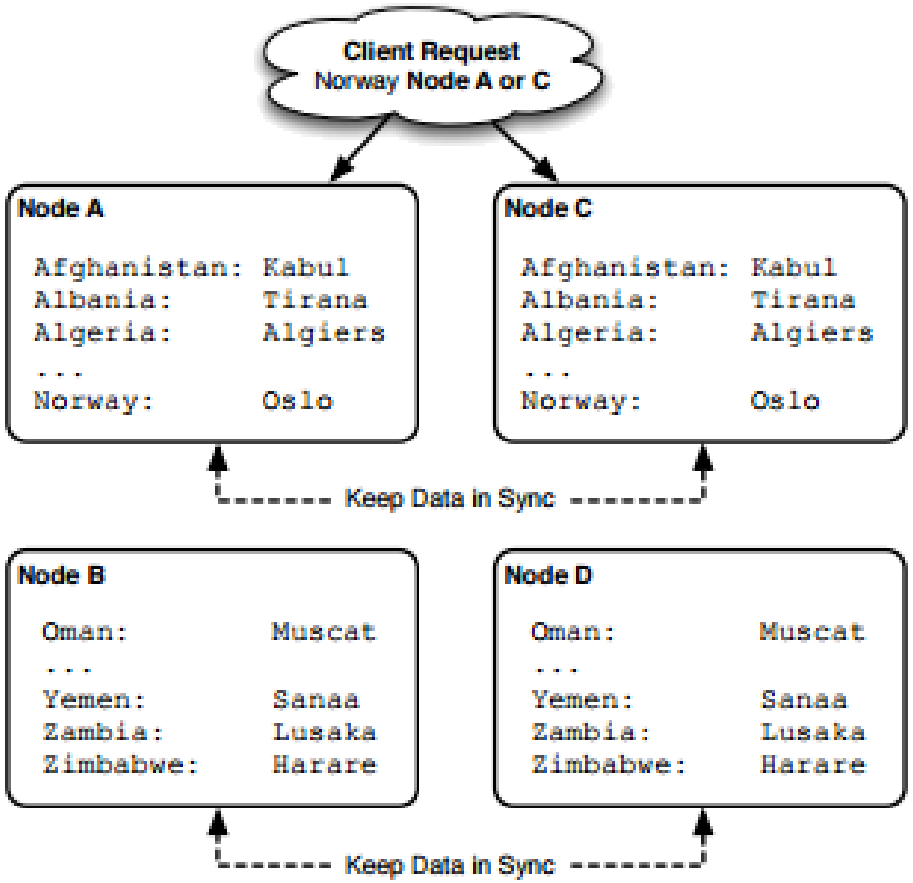
Data distribution in Riak

- Partitioned



Data distribution in Riak

- Riak uses Replication + Partitioning

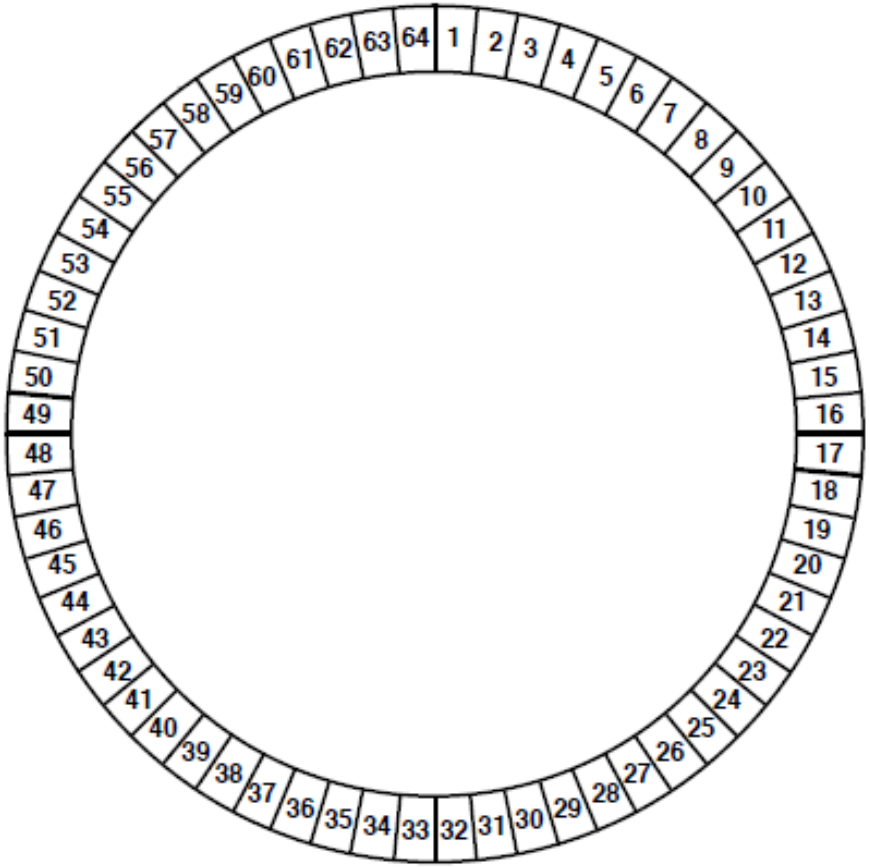


Hash function

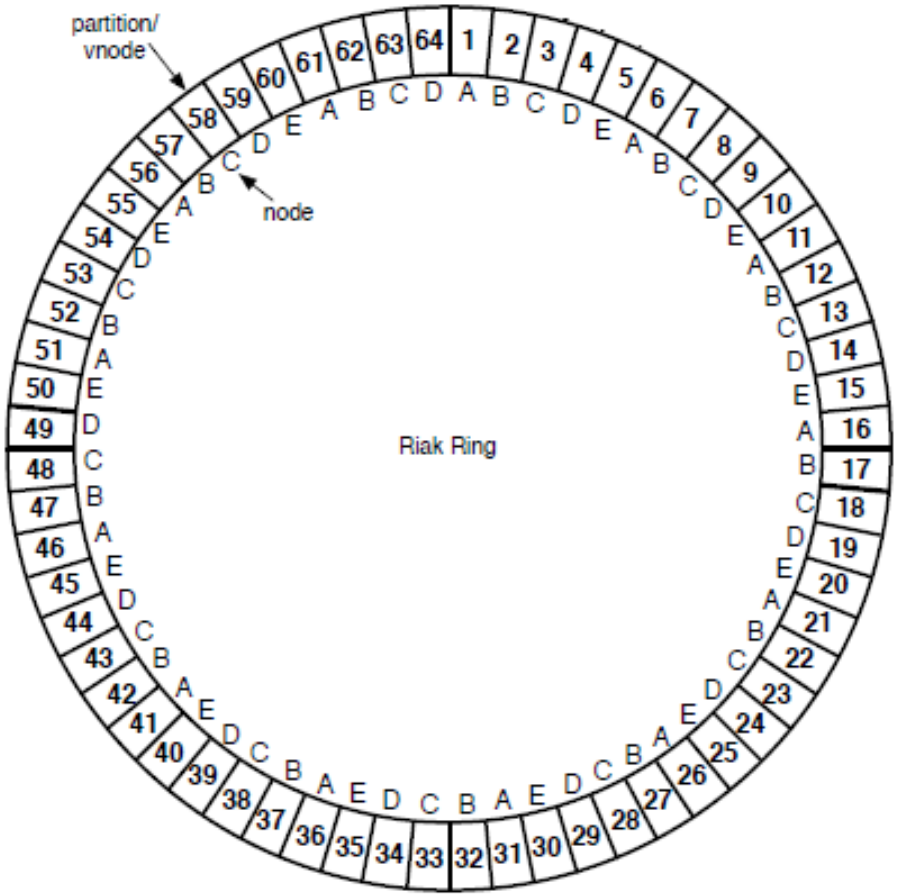
```
# hash for string
print("Hash for 'favorite' is:", hash('favorite'))
print("Hash for 'favorite' is:", hash('favorite'))
print("Hash for 'FAVORITE' is:", hash('FAVORITE'))
```

```
Hash for 'favorite' is: 7755465691136918617
Hash for 'favorite' is: 7755465691136918617
Hash for 'FAVORITE' is: -3146618751266618l3
```

The Riak Ring – The Cluster



The Riak Ring – The Cluster

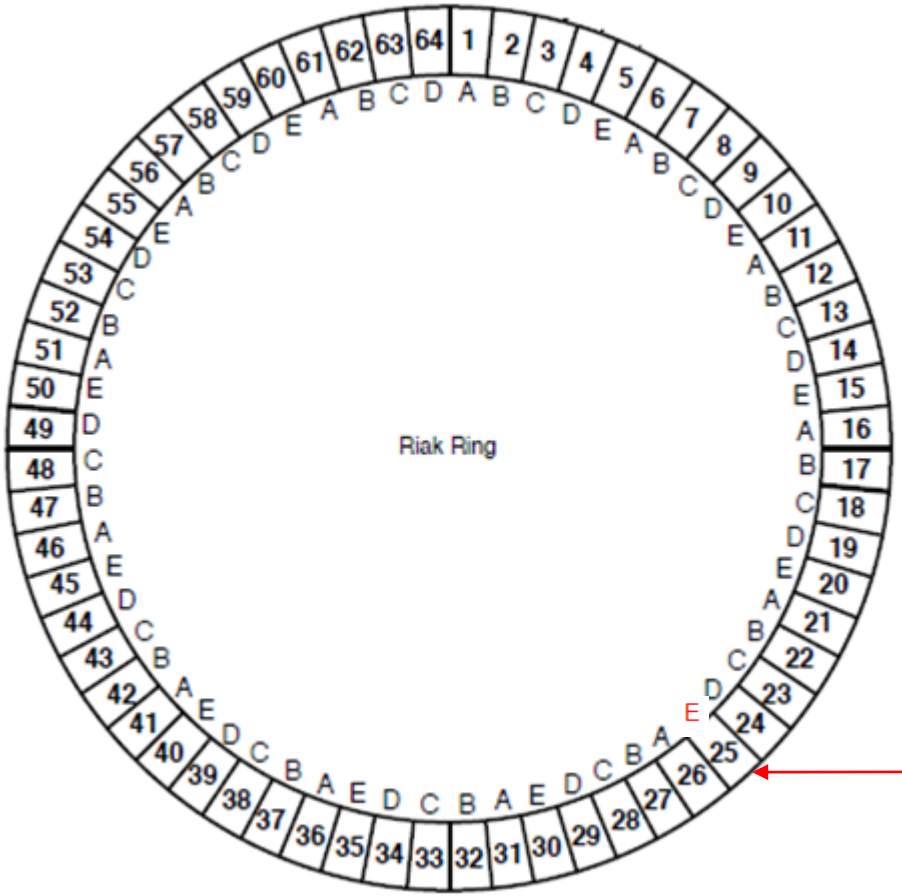


Hash function

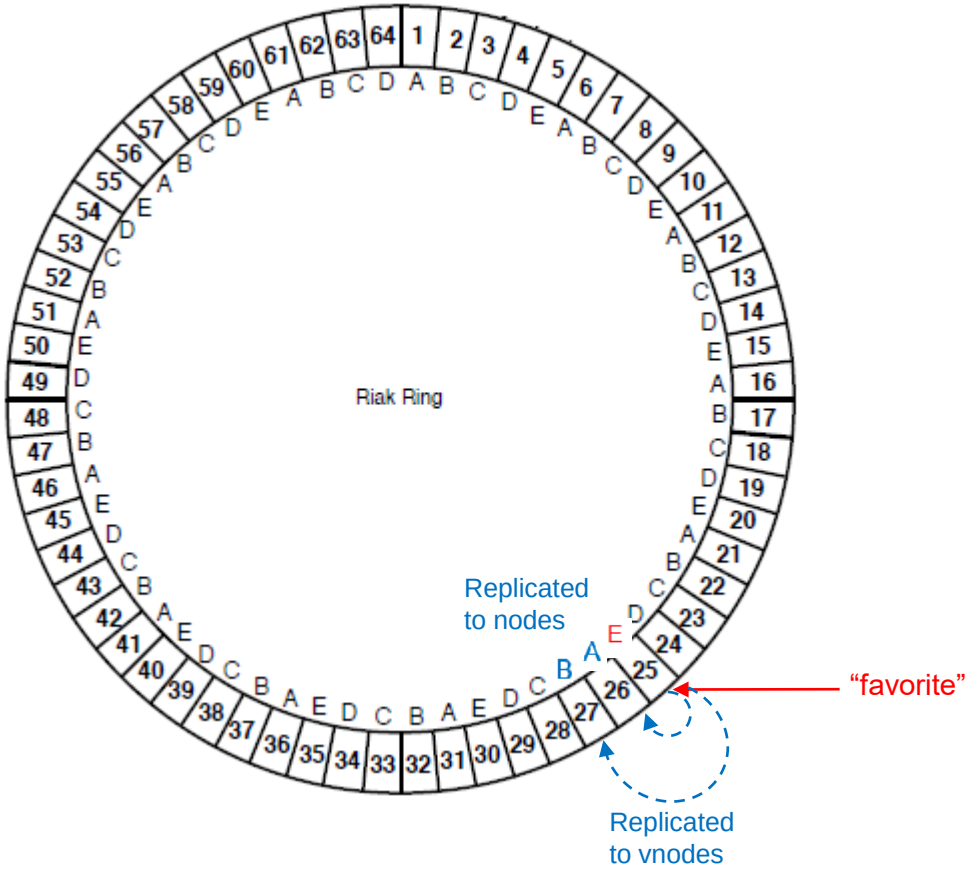
```
print("Hash for 'favorite' is:", hash('favorite'))
print("vnode for 'favorite' is:", abs(hash('favorite') % 64))
```

```
Hash for 'favorite' is: 7755465691136918617
VNode for 'favorite' is: 25
```

The Riak Ring – Replication

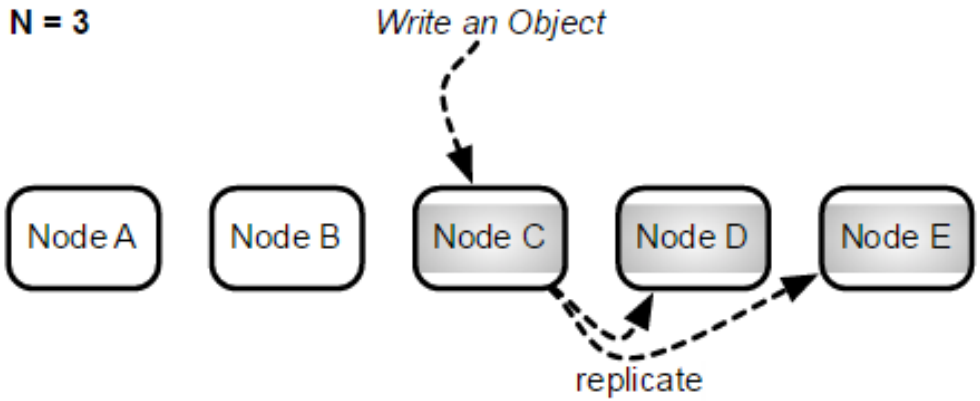


The Riak Ring – Replication



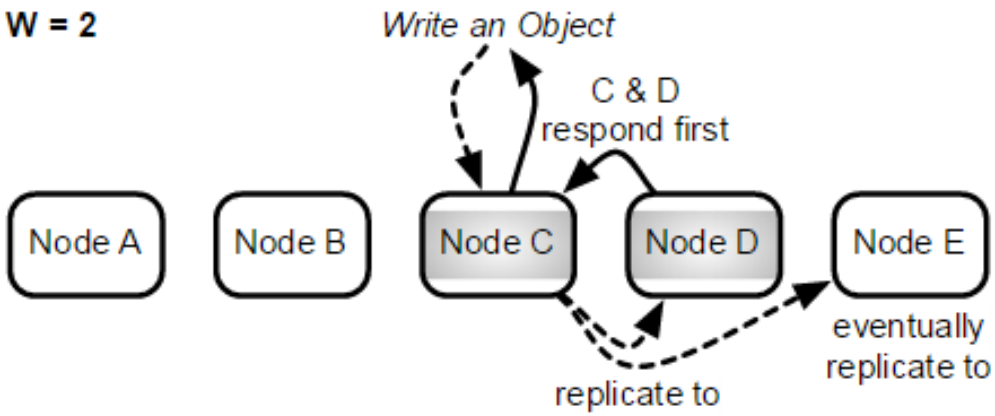
N/R/W

- N - Number of nodes in which the information is replicated



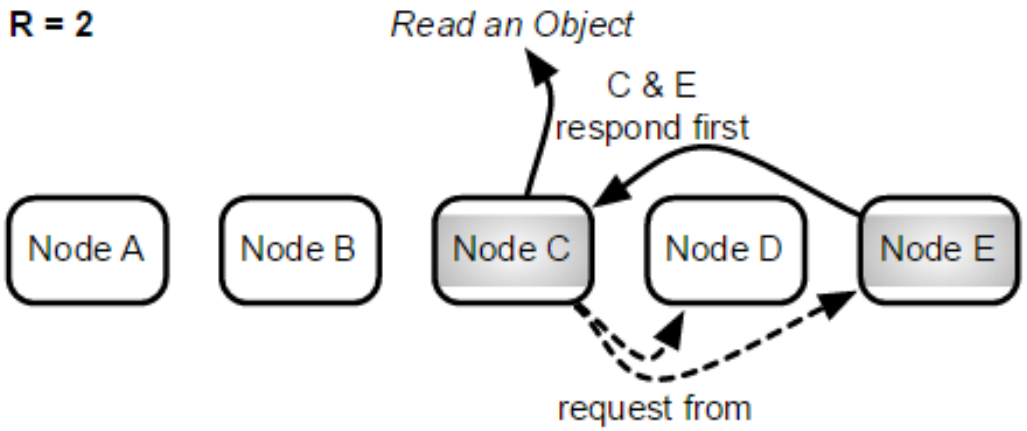
N/R/W

- W - Number of nodes to be written to before the operation is considered successful



N/R/W

- R - Number of nodes to be read from before returning the value



CRUD operations in Riak

- Inserting / Changing a password

```
curl -i -XPUT "http://localhost:8098/riak/food/favorite" \
-H "Content-Type:text/plain" \
-d "pizza"
```

```
HTTP/1.1 204 No Content
Vary: Accept-Encoding
Server: MochiWeb/1.1 WebMachine/1.10.8 (that head fake, tho)
Date: Fri, 13 Nov 2015 07:16:34 GMT
Content-Type: text/plain
Content-Length: 0
```

CRUD operations in Riak

- Reading a password

```
curl -i -XGET "http://localhost:8098/riak/food/favorite"
```

```
HTTP/1.1 200 OK
X-Riak-Vclock: a85hYGBgzGDKBVI8ypz/fkY4RxxgYPx9KYMpkTGPlSGIa9d5viwA
Vary: Accept-Encoding
Server: MochiWeb/1.1 WebMachine/1.10.8 (that head fake, tho)
Link: </riak/food>; rel="up"
Last-Modified: Fri, 13 Nov 2015 07:16:34 GMT
ETag: "5eBXvxQJPMoirlTo6QeXV5"
Date: Fri, 13 Nov 2015 07:16:35 GMT
Content-Type: text/plain
Content-Length: 5

pizza
```

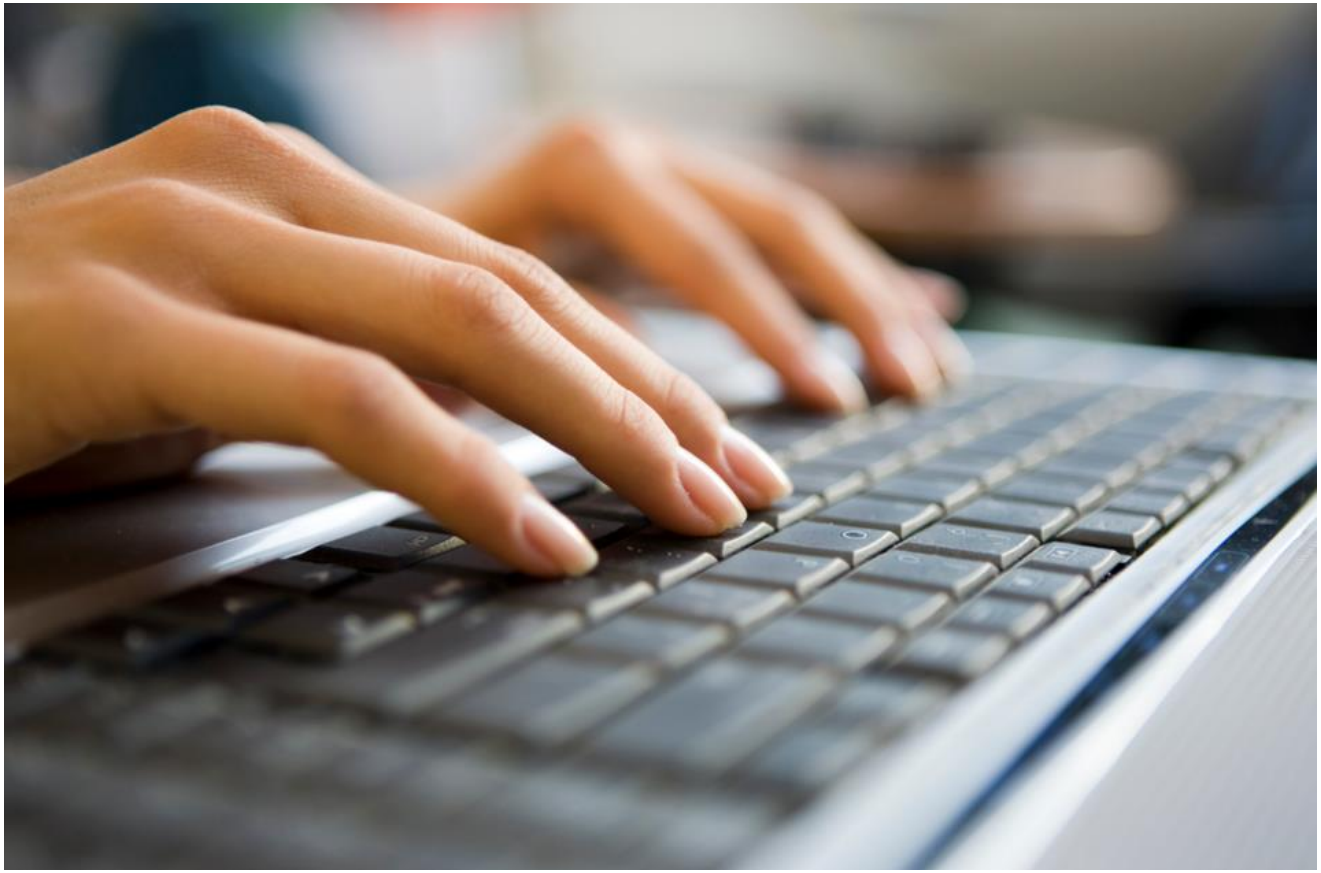
CRUD operations in Riak

- Deleting a key

```
curl -i -XDELETE "http://localhost:8098/riak/food/favorite"
```

```
HTTP/1.1 204 No Content
Vary: Accept-Encoding
Server: MochiWeb/1.1 WebMachine/1.10.8 (that head fake, tho)
Date: Fri, 13 Nov 2015 07:16:42 GMT
Content-Type: text/plain
Content-Length: 0
```

Riak - HandsOn



Use Cases

- Simple applications requiring high performance in read/write operations
- Applications that need the database to be always available
- Serving ads to web / mobile applications
- User preferences
- Session Storage

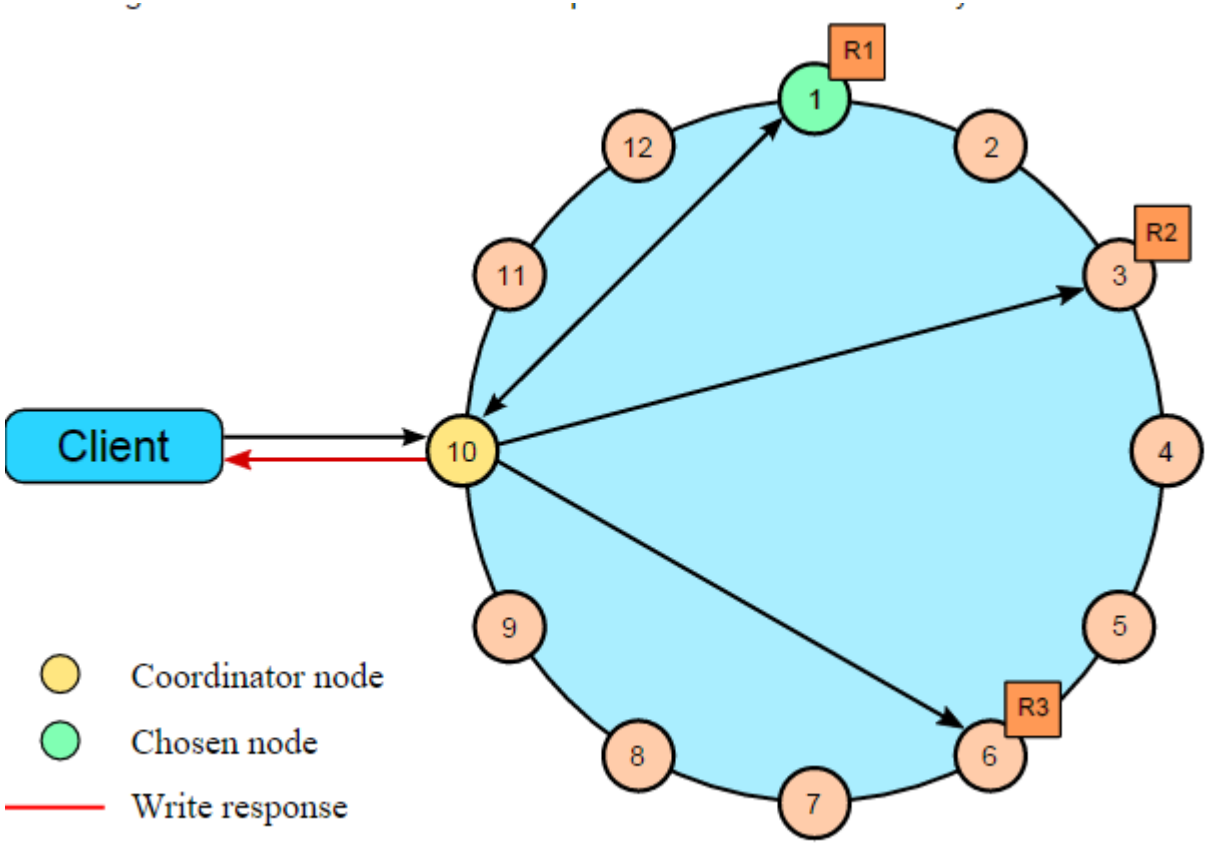
Database Workshop

- Material
- Use case
- Relational Databases
- NoSQL
- Riak
- Apache Cassandra
- MongoDB
- Neo4j

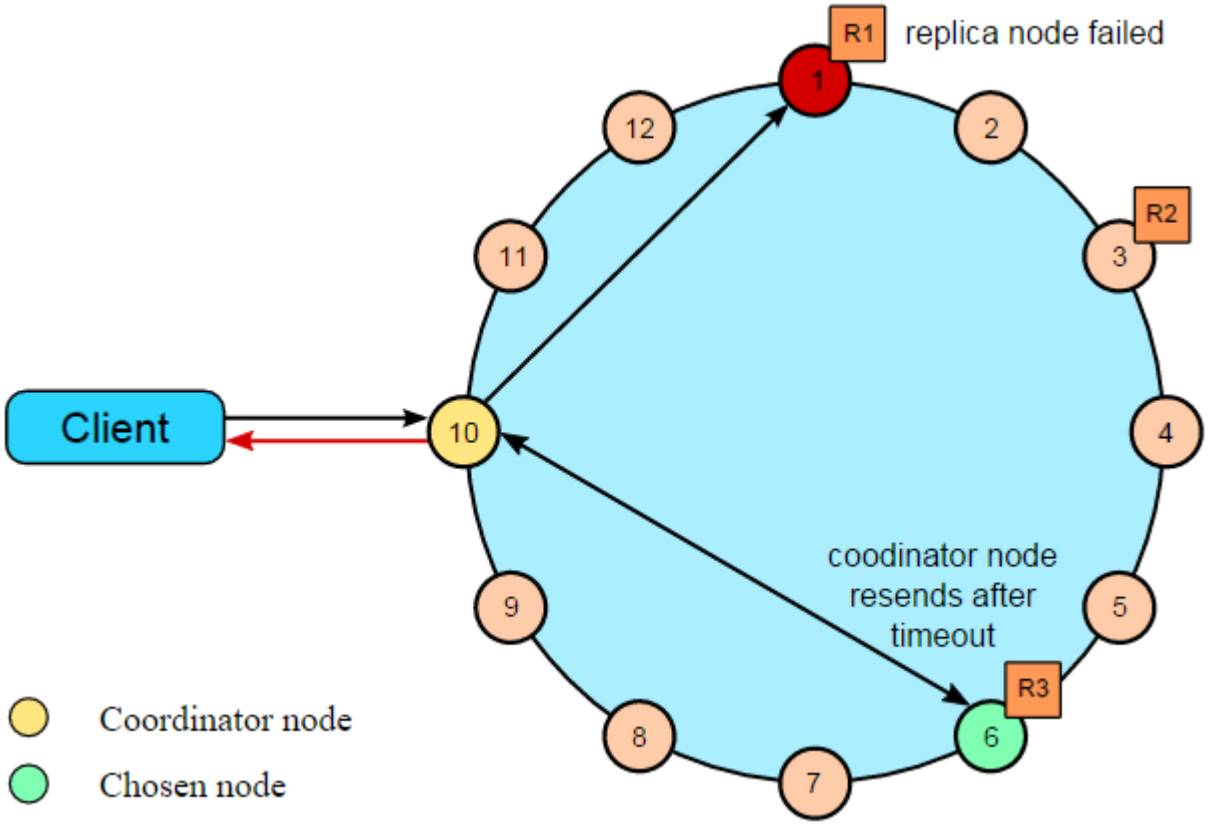
Apache Cassandra

- Developed in java by Facebook and donated to the Apache Foundation in 2008
- It is based on a key/value model by storing several columns per key.
- Inspired by Amazon's Dynamo (Same as Riak) and Google's BigTable (Column families)
- There's no central controller. No single point of failure
- Querys: CQL Language - Similar to SQL
- Compatible with Hadoop and Spark
- Supports multiple data centers
- Linear scalability

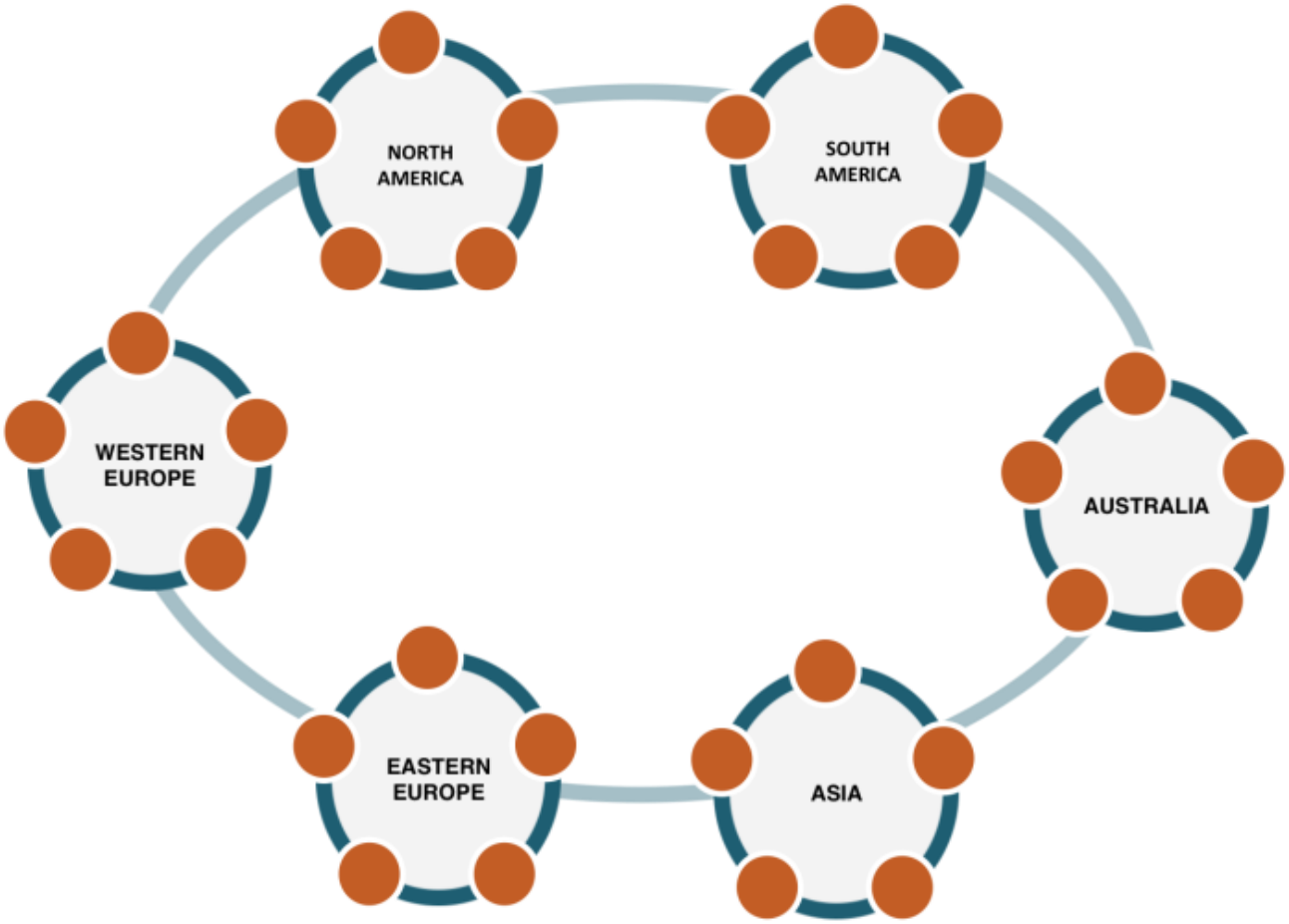
Architecture - Writing (W = 3)



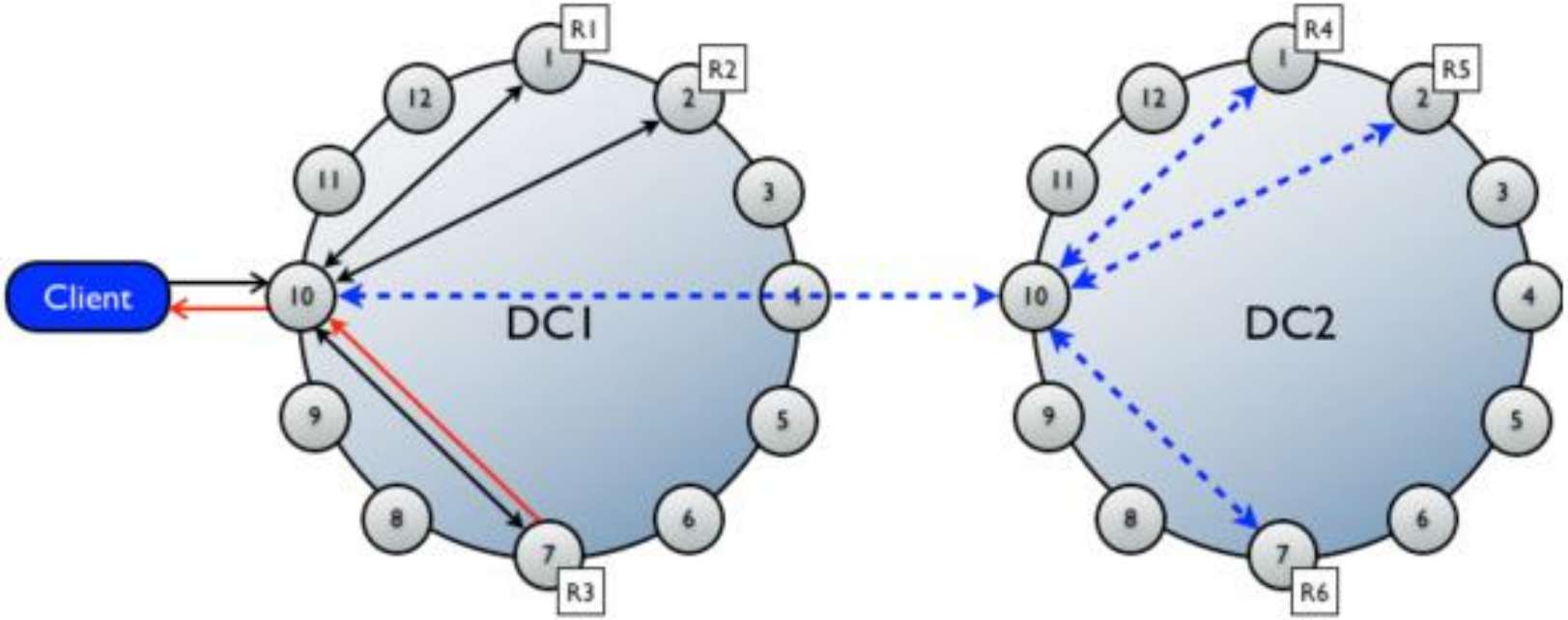
Architecture - Reading



Cassandra Cluster
Multiple DataCenters



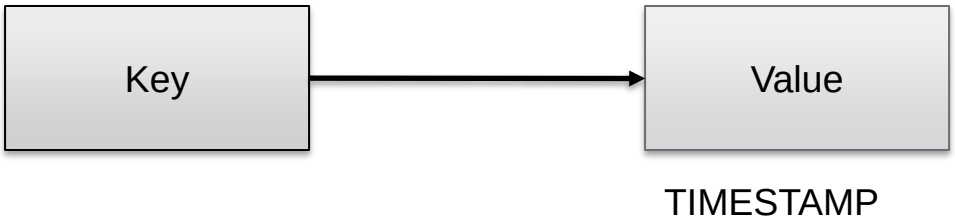
Cassandra Cluster
Multiple DataCenters



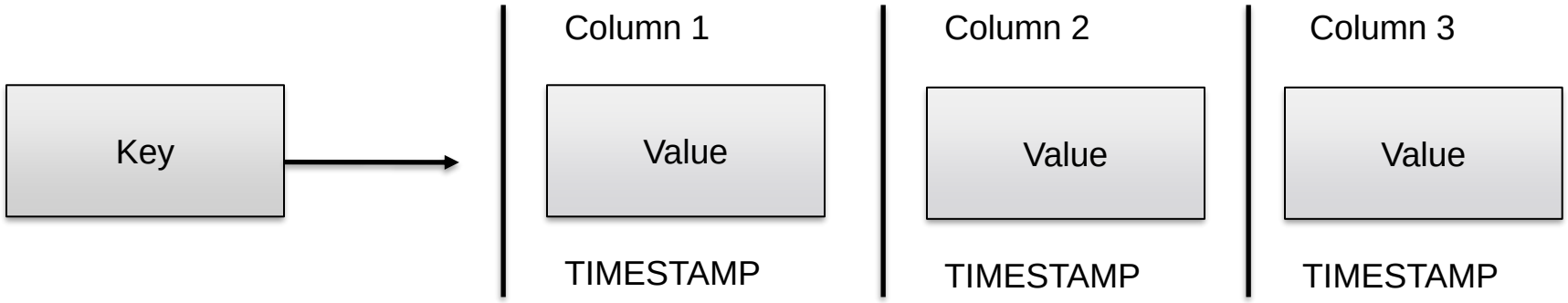
BigTable

- Data is stored in tables (Column Families)
- Tables are stored in separate databases (Keyspaces)
- Every table must have a primary key (Partition Key)
- Additionally a table can have composite keys

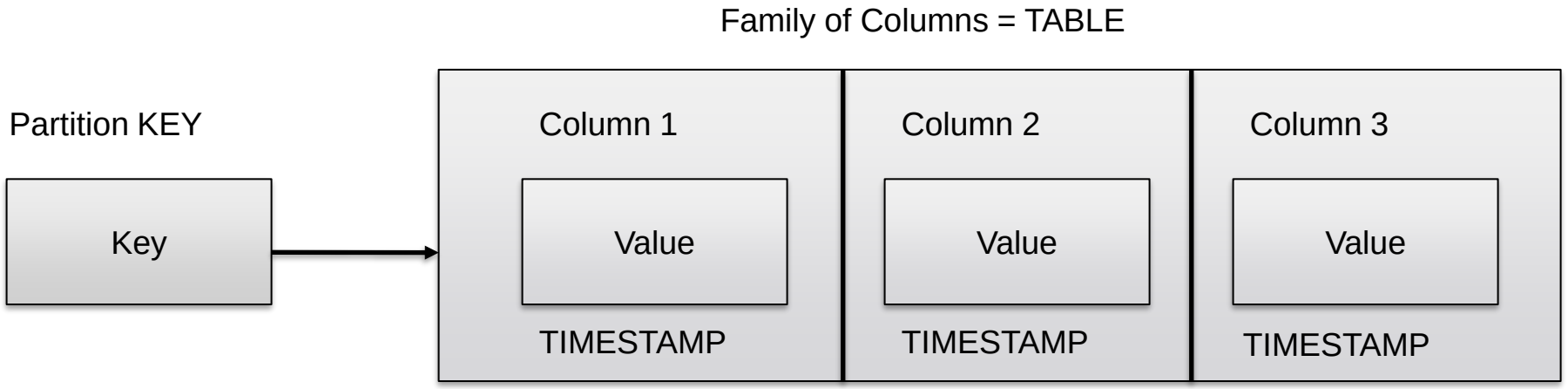
Remembering the Key - Value model



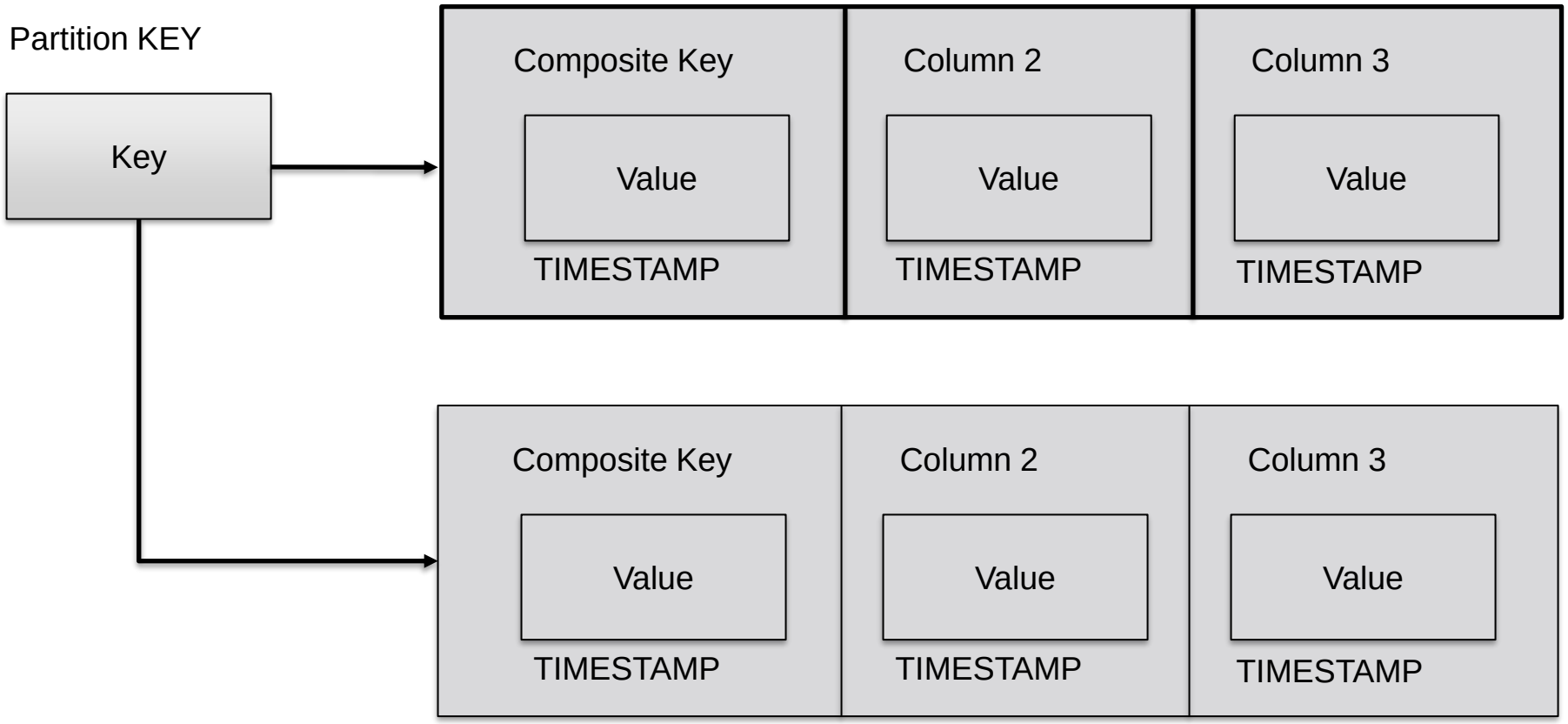
BigTable


Key
Column 1
Value
TIMESTAMP
Column 2
Value
TIMESTAMP
Column 3
Value
TIMESTAMP

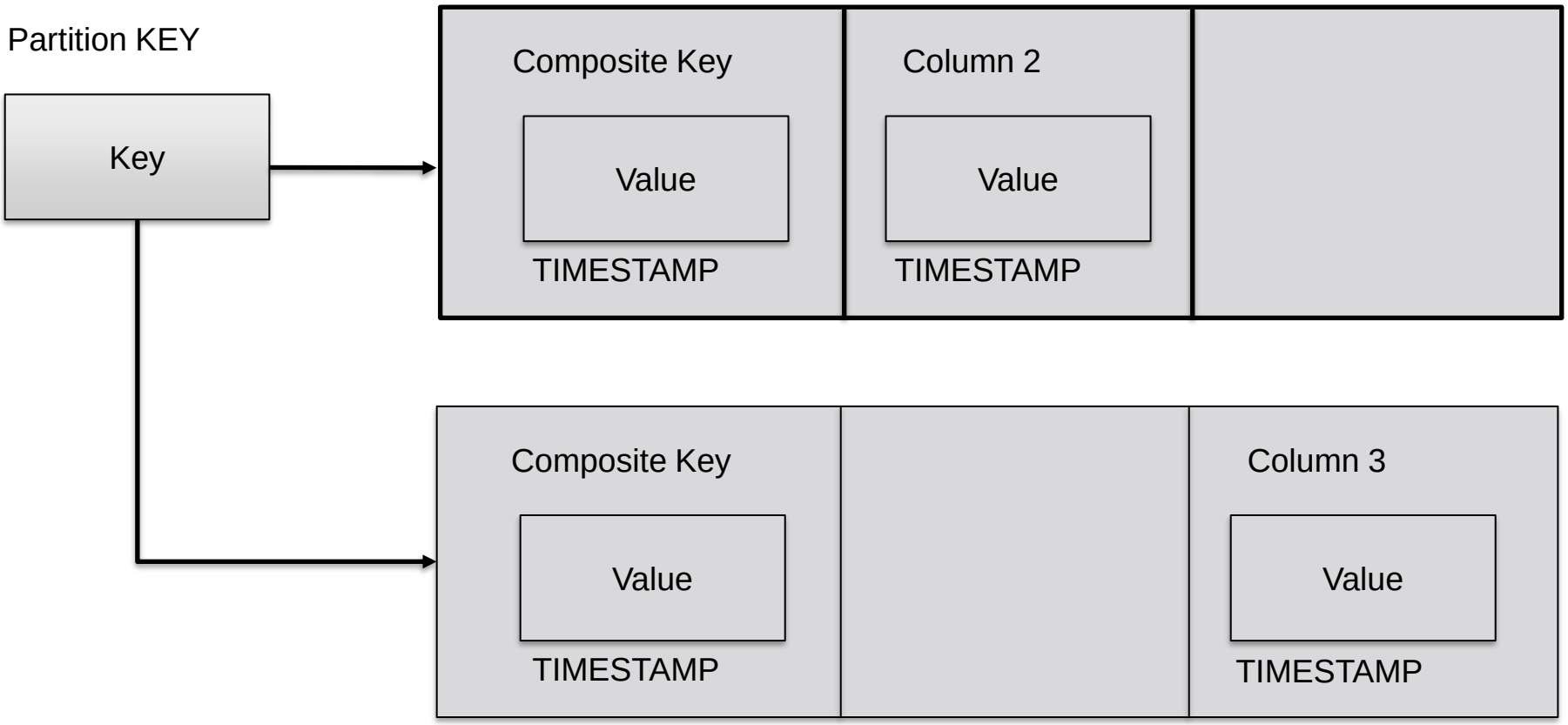
BigTable


Family of Columns = TABLE
Partition KEY
Key
Column 1
Value
TIMESTAMP
Column 2
Value
TIMESTAMP
Column 3
Value
TIMESTAMP

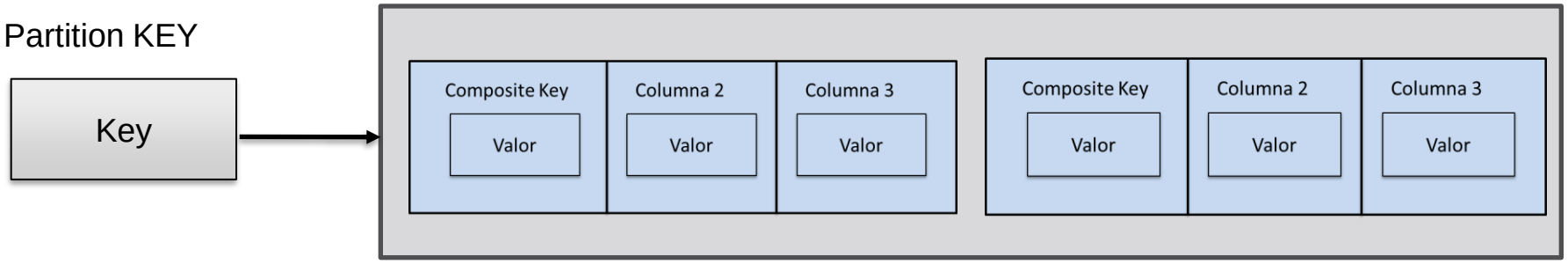
BigTable


Partition KEY
Key
Composite Key
Value
TIMESTAMP
Column 2
Value
TIMESTAMP
Column 3
Value
TIMESTAMP
Composite Key
Value
TIMESTAMP
Column 2
Value
TIMESTAMP
Column 3
Value
TIMESTAMP

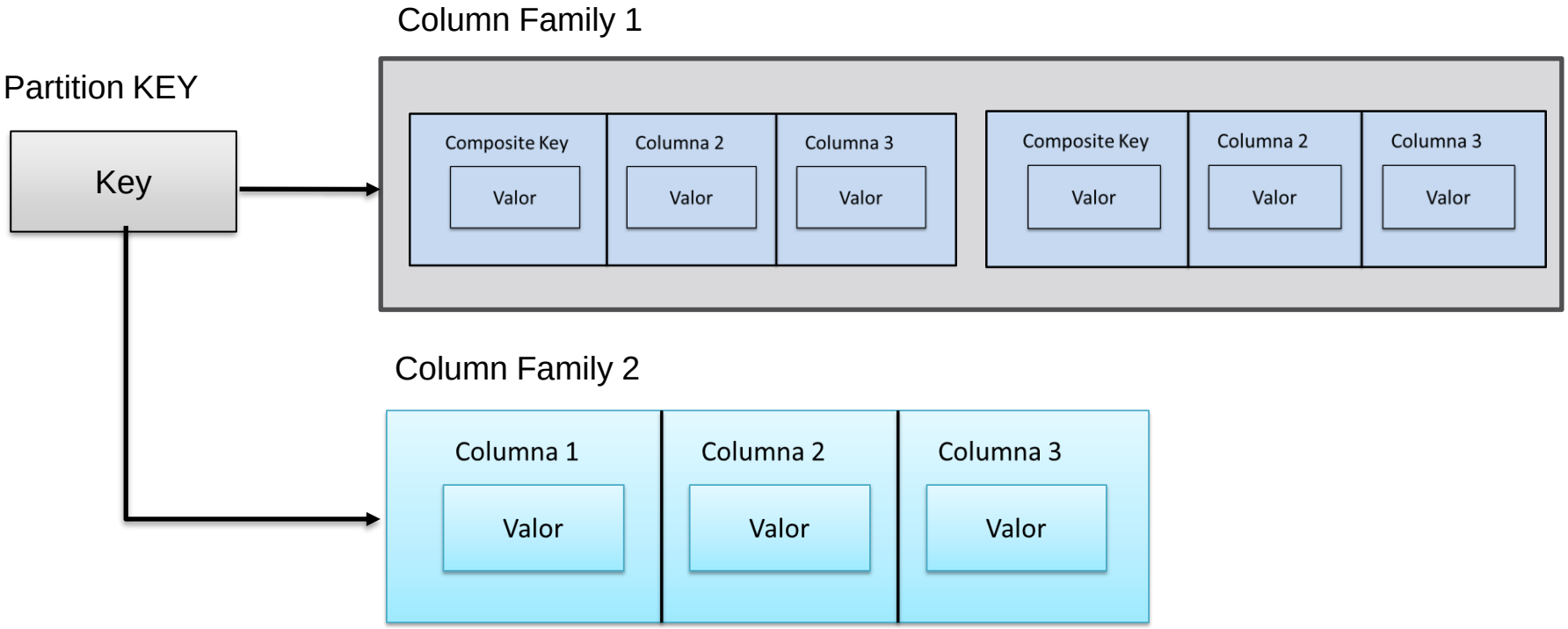
BigTable


Partition KEY
Key
Composite Key
Value
TIMESTAMP
Column 2
Value
TIMESTAMP
Composite Key
Value
TIMESTAMP
Column 3
Value
TIMESTAMP

BigTable

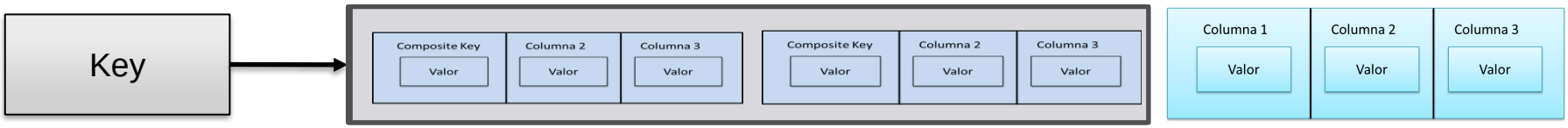

Partition KEY
Key
Composite Key
Valor
Columna 2
Valor
Columna 3
Valor
Composite Key
Valor
Columna 2
Valor
Columna 3
Valor

BigTable

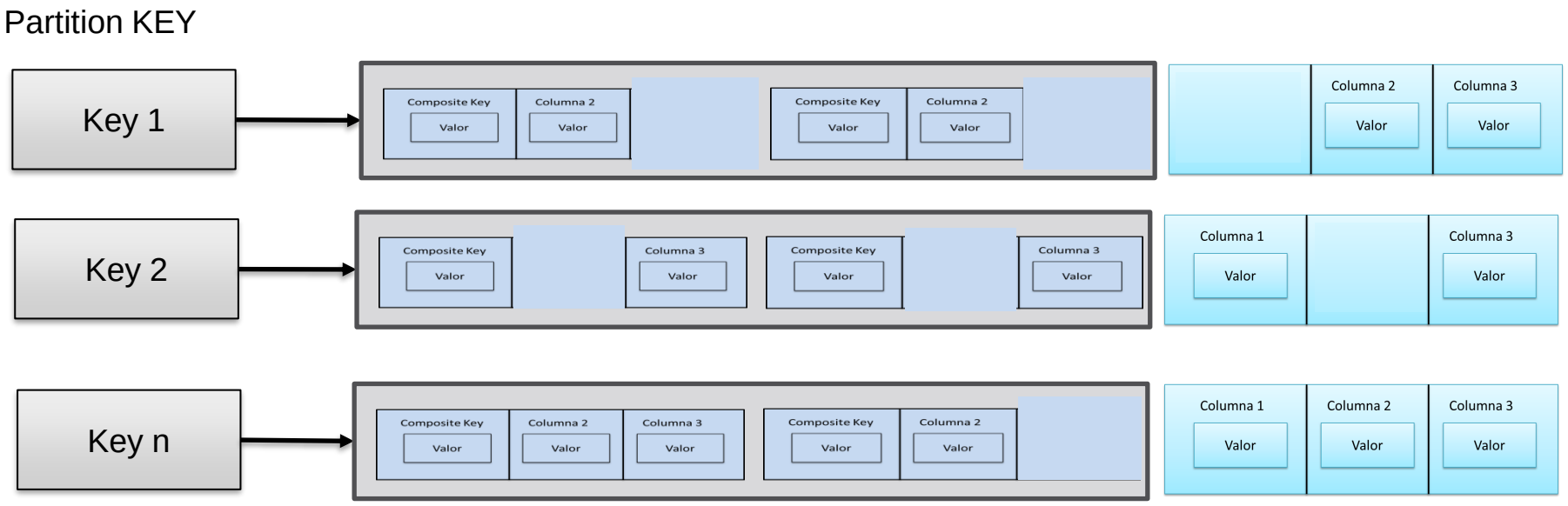

Column Family 1
Partition KEY
Key
Composite Key
Columna 2
Columna 3
Valor
Valor
Valor
Composite Key
Columna 2
Columna 3
Valor
Valor
Valor
Column Family 2
Columna 1
Columna 2
Columna 3
Valor
Valor
Valor

BigTable

Partition KEY



BigTable


Partition KEY
Key 1
Composite Key
Valor
Columna 2
Valor
Composite Key
Valor
Columna 2
Valor
Columna 2
Valor
Columna 3
Valor
Key 2
Composite Key
Valor
Columna 3
Valor
Composite Key
Valor
Columna 3
Valor
Columna 1
Valor
Columna 3
Valor
Key n
Composite Key
Valor
Columna 2
Valor
Columna 3
Valor
Composite Key
Valor
Columna 2
Valor
Columna 1
Valor
Columna 2
Valor
Columna 3
Valor

Cassandra Query Language CQL

- It looks like SQL but is much more limited
  - No JOINS
  - No GROUP BY
  - No ORDER BY

```
CREATE KEYSPACE demo
WITH replication = {'class':'SimpleStrategy', 'replication_factor': 1};
```

```
CREATE TABLE users (
    firstname text,
    lastname text,
    age int,
    email text,
    city text,
    PRIMARY KEY (lastname)
);
```

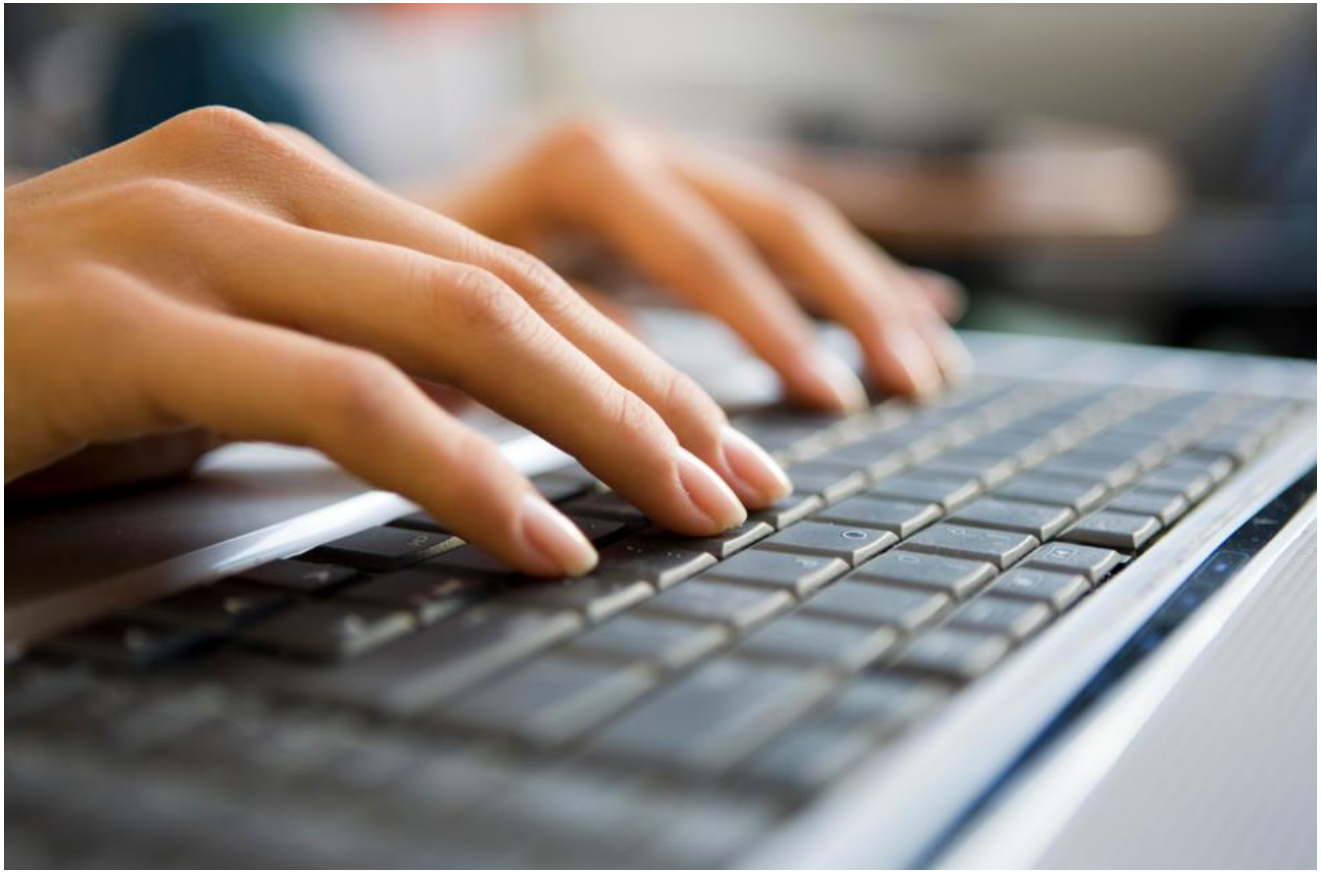
```
SELECT *
FROM users
WHERE lastname= 'Doe'
LIMIT 1;
```

```
UPDATE users
    SET city= 'San Jose'
WHERE lastname= 'Doe';
```

```
DELETE from users WHERE lastname = 'Doe';
```

```
INSERT INTO users (firstname, lastname, age, email, city)
VALUES ('John', 'Smith', 46, 'johnsmith@email.com', 'Sacramento');
```

Cassandra - HandsOn



Cheat Sheet

Use Cases

- Applications requiring very high real-time writing capabilities
- Business Intelligence systems that require a very fast database reading
- Decentralized applications that need to store large amounts of information
- Smart cities. Sensors and monitoring
- Content Delivery Network (CDN) - Highly distributed static content servers

Database Workshop

- Material
- Use case
- Relational Databases
- NoSQL
- Riak
- Apache Cassandra
- MongoDB
- Neo4j


```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [],
  "spouse": null
}
```

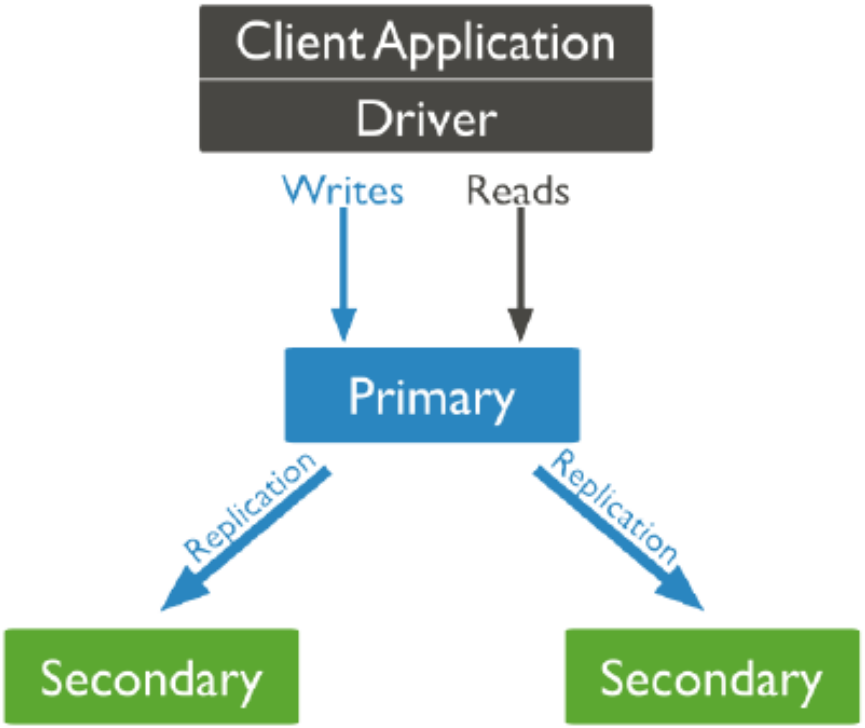

MongoDB

- It is a document-oriented database (Key Collections / value)
- It is very flexible in structuring the data
- Querys: Javascript with its own API based on high capabilities for information querying
- It has geospatial characteristics
- Prioritizes consistency over availability
- Master / Slave type replication
- Scale horizontally thanks to Sharding
- There is a connector for BI tools
- Compatible with Hadoop and Spark

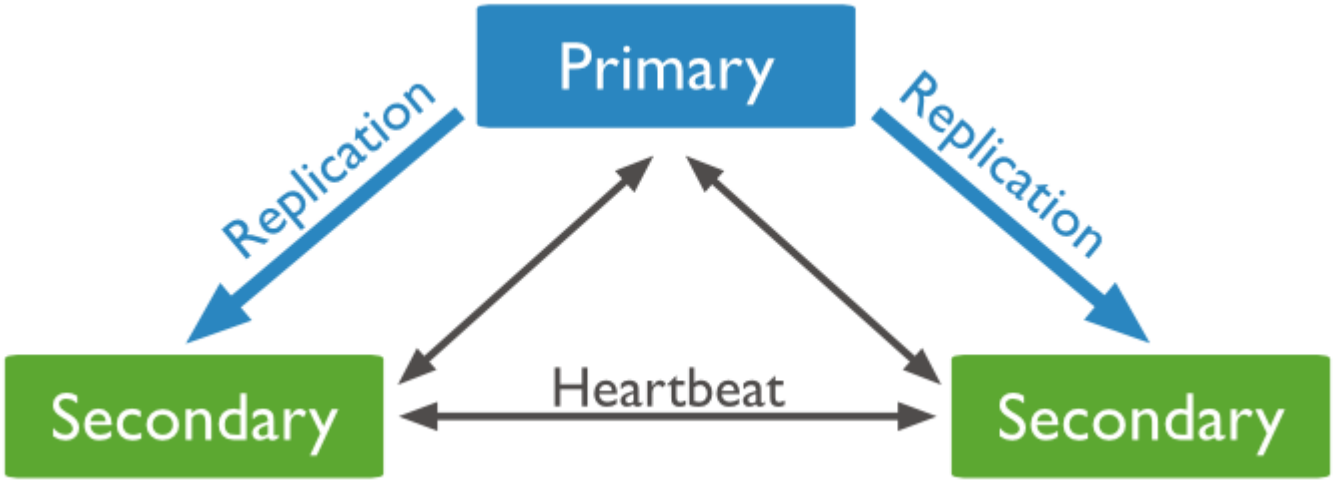
Nomenclature

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
Column/Attribute/Variable	Field
Table Join	Embedded Documents
Database Server and Client	
Primary Key	Primary Key (Default key _id provided by mongodb itself)
Mysqld/Oracle	mongod
mysql/sqlplus	mongo

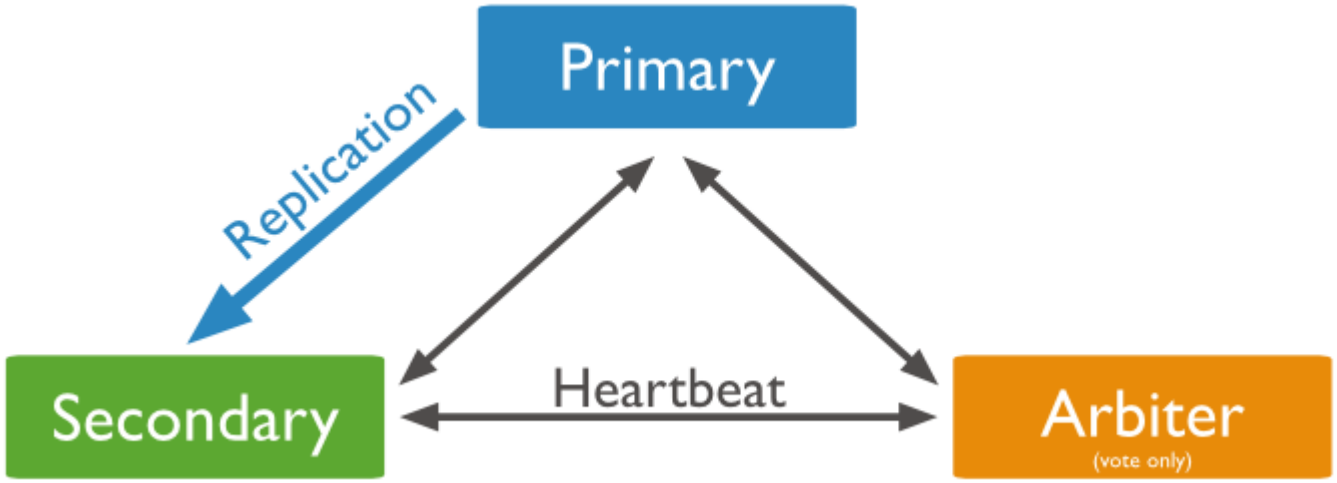
Replication - Scale the readings



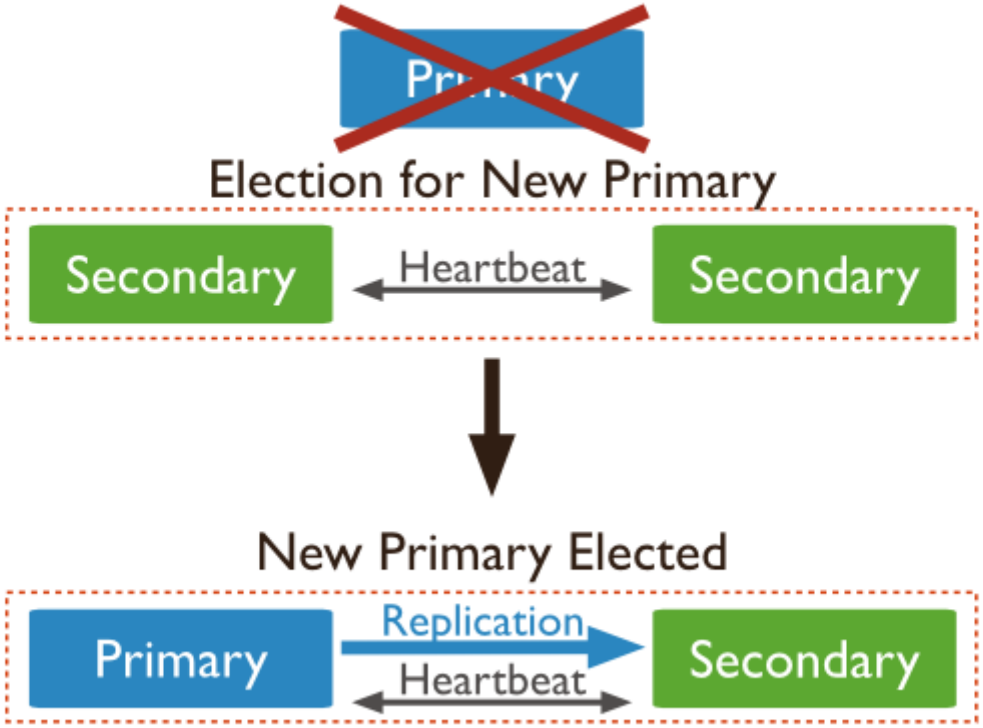
Replication



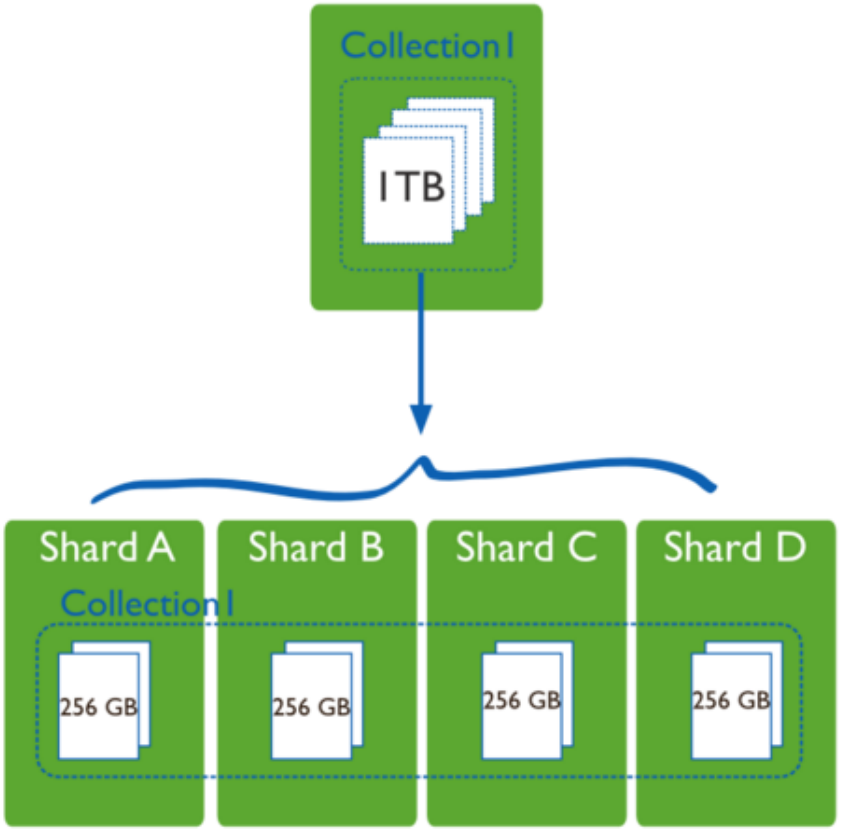
Replication



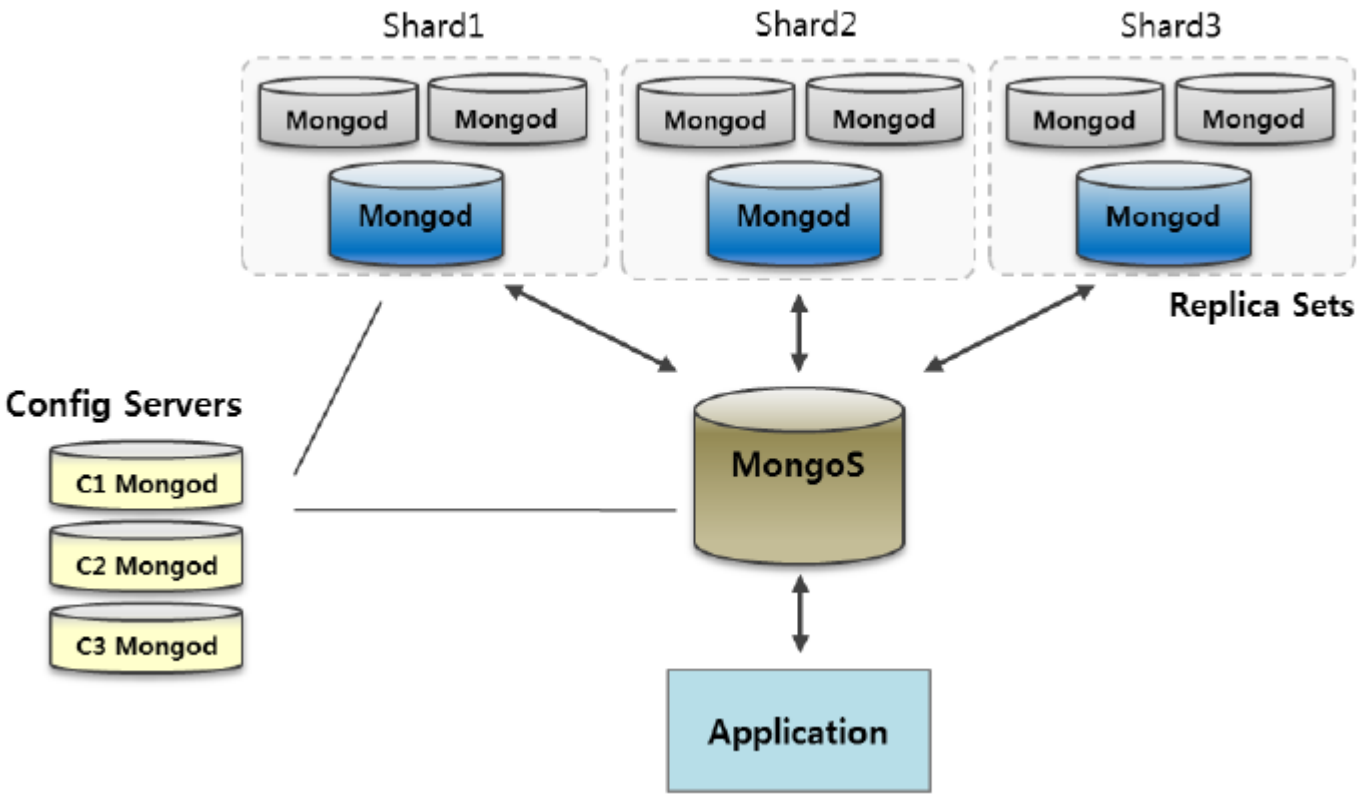
Replication



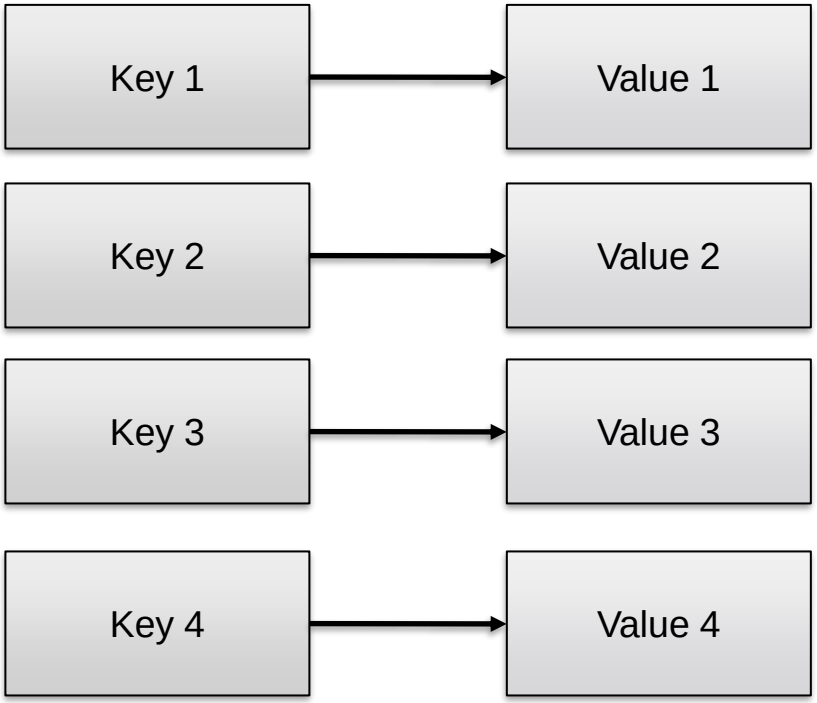
Sharding- Scale the scriptures



Complete map

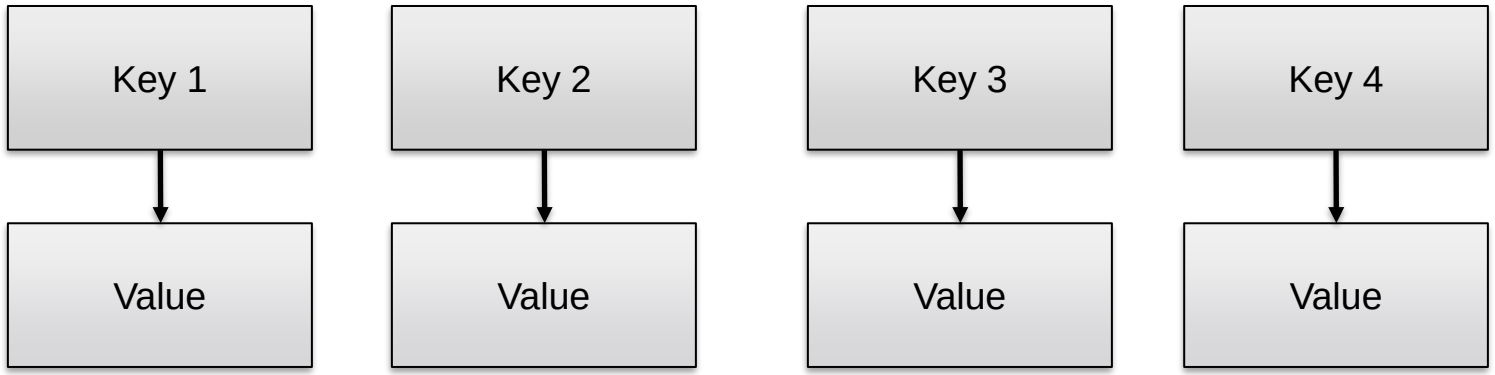


Document

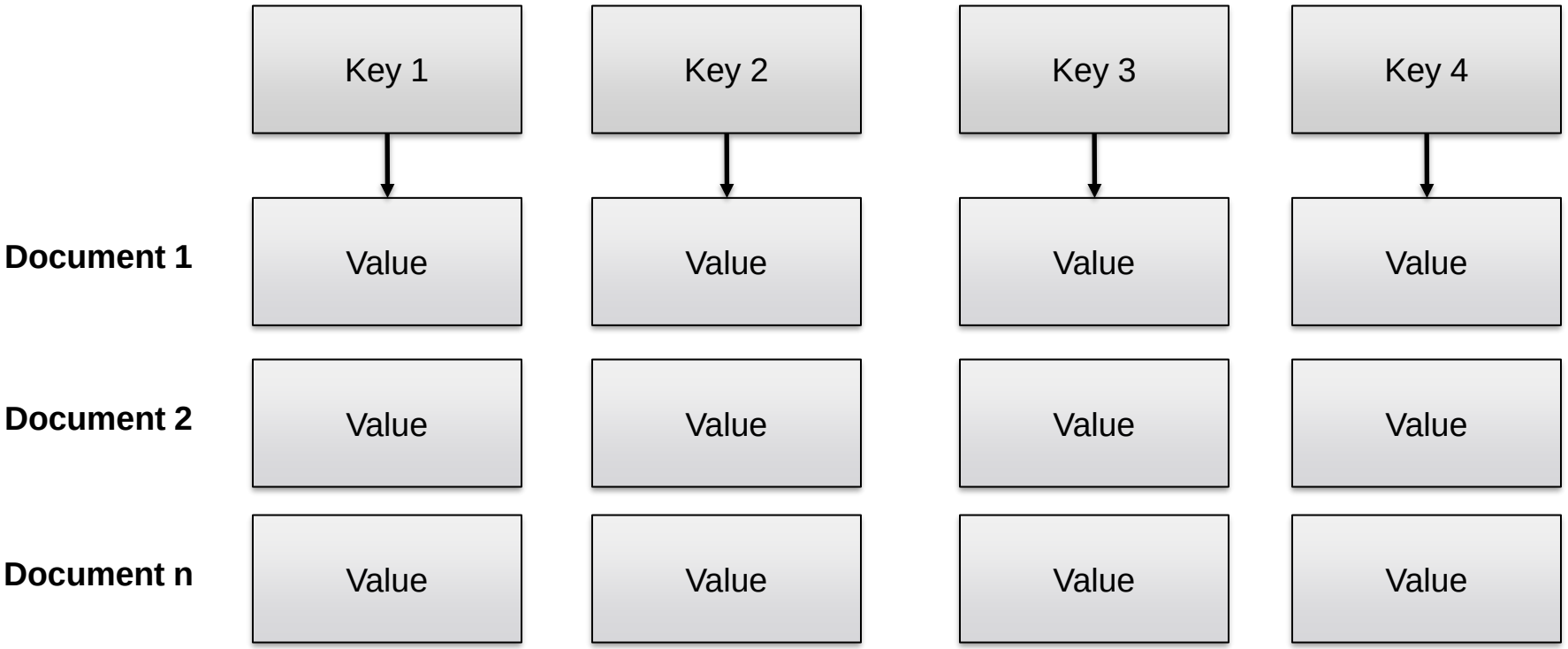


Document

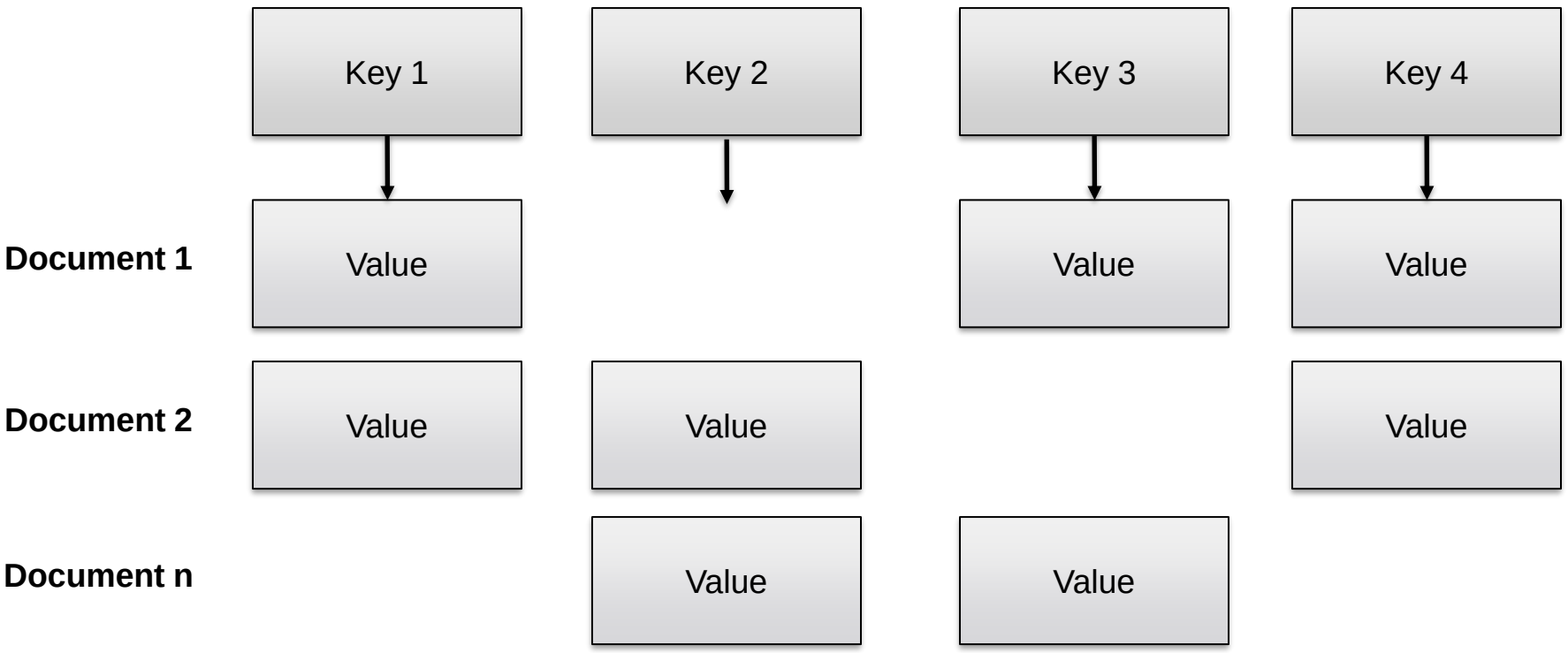
Document



Collection


Key 1
Key 2
Key 3
Key 4
Document 1
Value
Value
Value
Value
Document 2
Value
Value
Value
Value
Document n
Value
Value
Value
Value

Collection

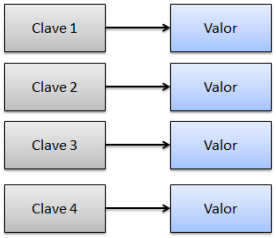

Key 1
Key 2
Key 3
Key 4
Document 1
Value
Value
Value
Document 2
Value
Value
Value
Document n
Value
Value

Documents

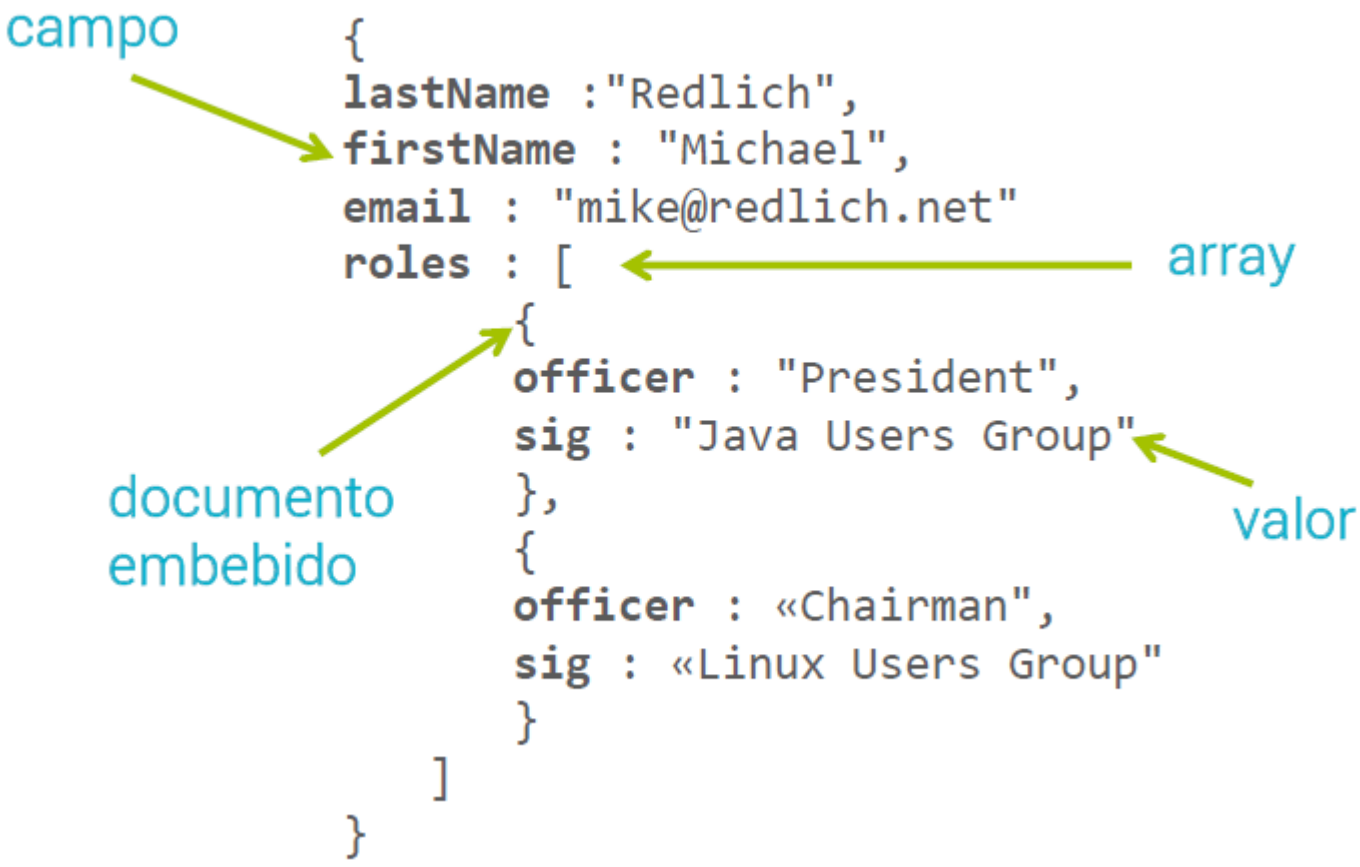
- The documents correspond to native data types in most programming languages.
- The ability to include other documents and arrays within the documents reduces the need for joins.
- Dynamic schemes allow support for any data structure in a collection

```
{
  name: "sue",
  age: 26,
  status: "A",
  groups: [ "news", "sports" ]
}
```

field: value
field: value
field: value
field: value

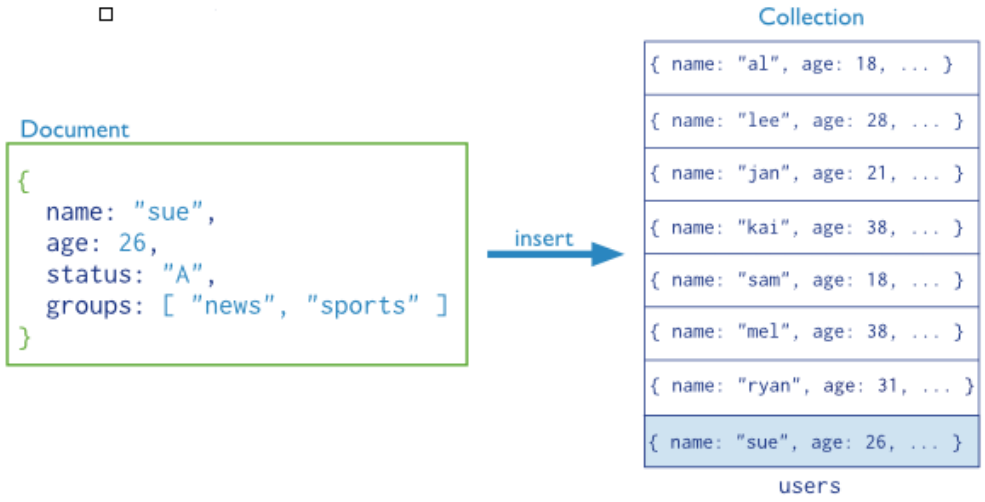


Document


campo
{
lastName :"Redlich",
firstName : "Michael",
email : "mike@redlich.net"
roles : [
array
{
officer : "President",
sig : "Java Users Group"
},
valor
documento
embebido
{
officer : «Chairman",
sig : «Linux Users Group"
}
]
}

CRUD operations

- Data insertion



Collection Document

```
db.users.insert(
                  {
                     name: "sue",
                      age: 26,
                   status: "A",
                   groups: [ "news", "sports" ]
                  }
                )
```

CRUD operations

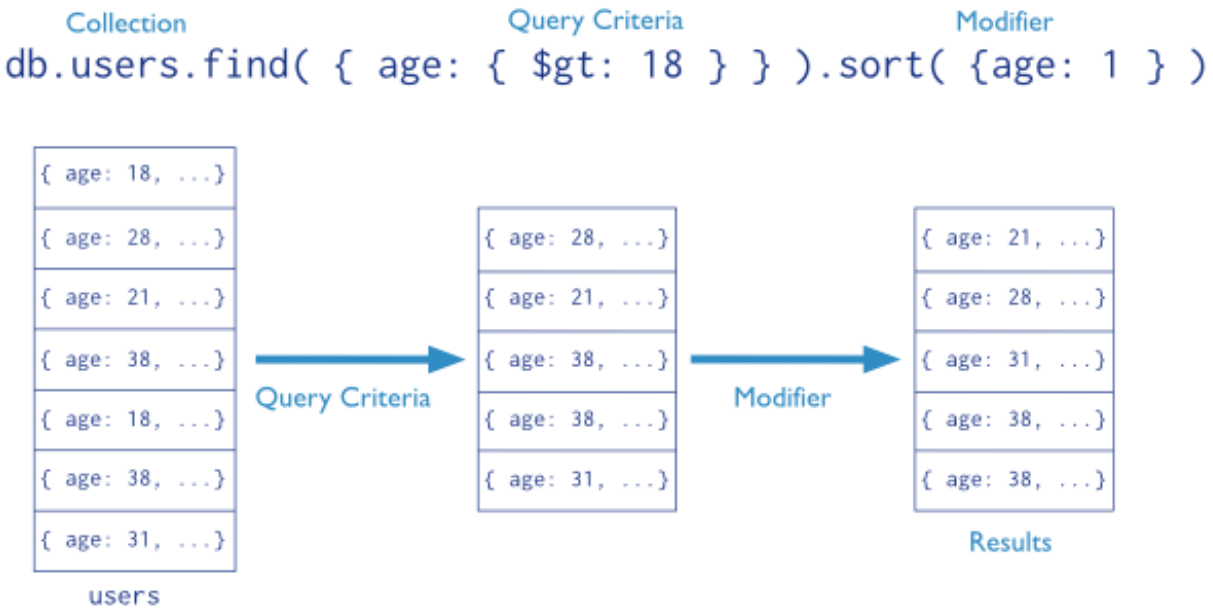
- Querys

```
SELECT _id, name, address   <-- projection
FROM   users                <-- table
WHERE  age > 18             <-- select criteria
LIMIT  5                    <-- cursor modifier
```

```
db.users.find(                          <-- collection
   { age: { $gt: 18 } },                <-- query criteria
   { name: 1, address: 1 }              <-- projection
).limit(5)                              <-- cursor modifier
```

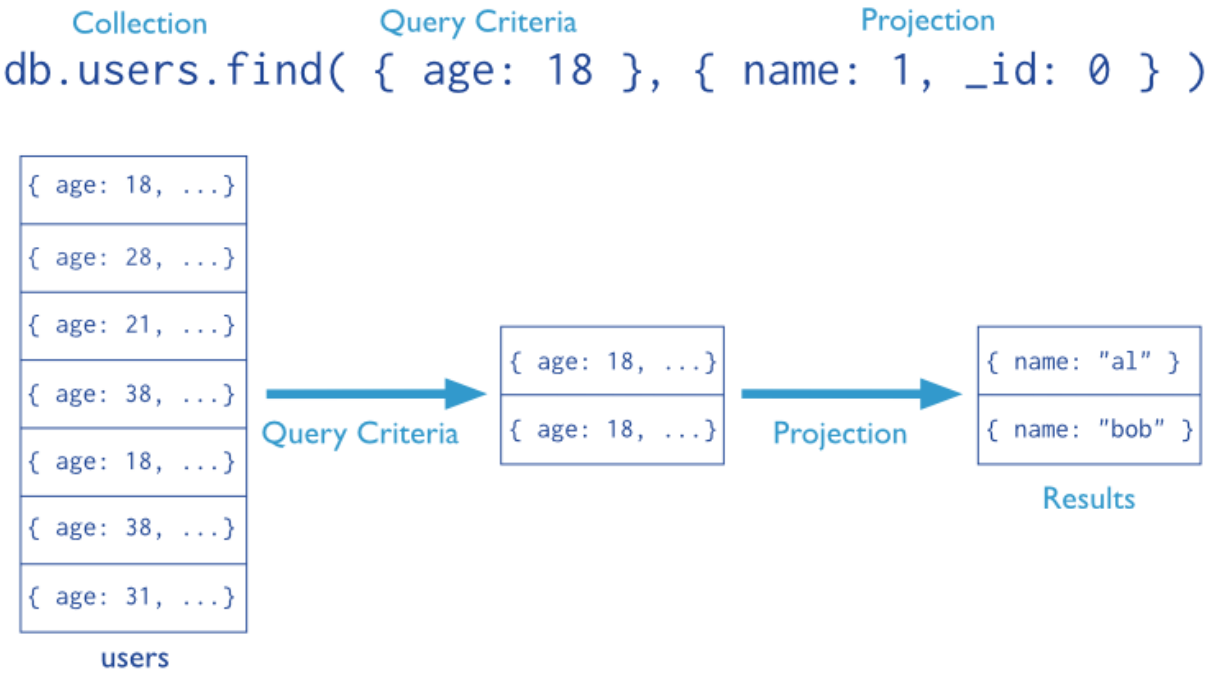
CRUD operations

- Ordered kerries



CRUD operations

- Kerries with projection



CRUD operations

- Modification of information

```
UPDATE users          <-- table
SET    status = 'A'   <-- update action
WHERE  age > 18       <-- update criteria
```

```
db.users.update(                    <-- collection
   { age: { $gt: 18 } },            <-- update criteria
   { $set: { status: "A" } },       <-- update action
   { multi: true }                  <-- update option
)
```

CRUD operations

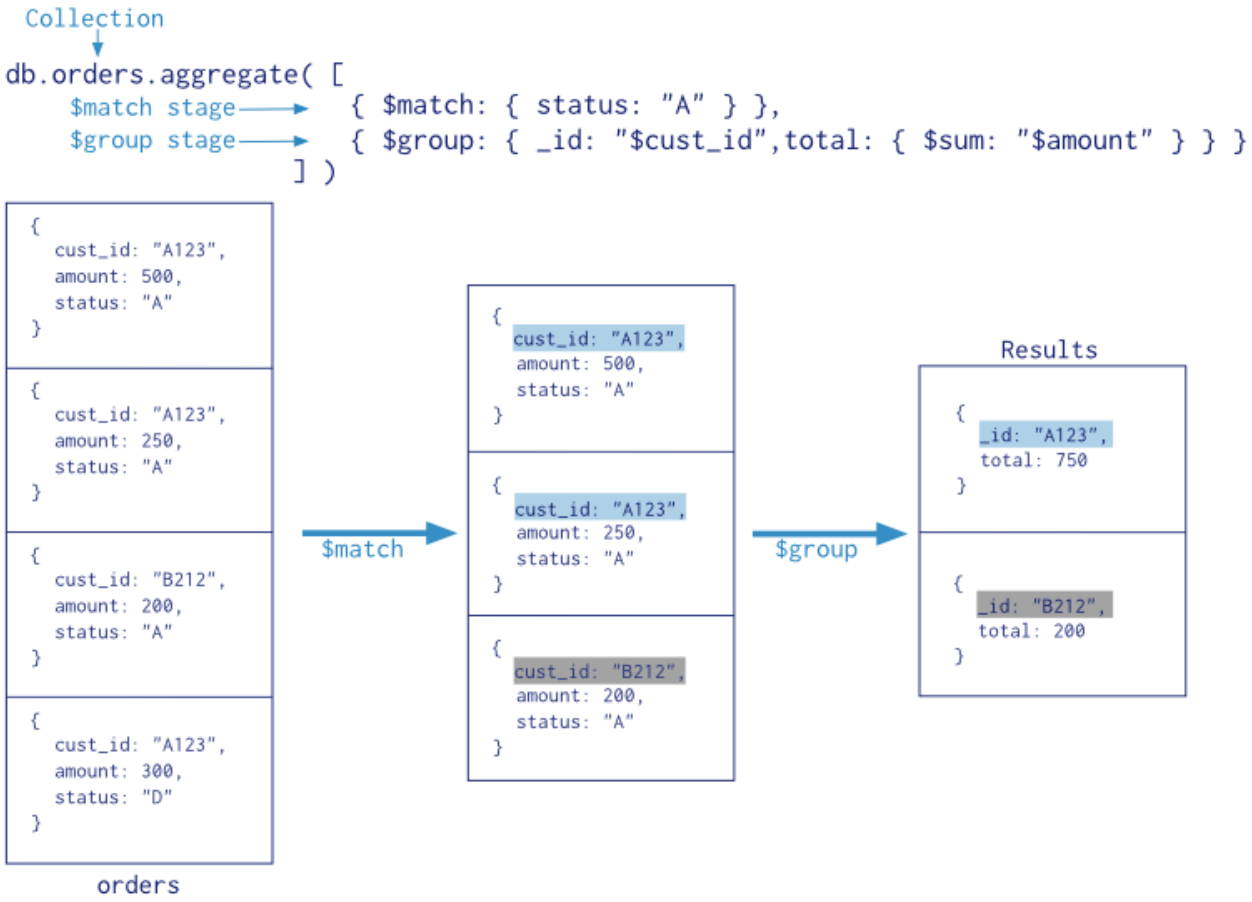
- Deletion of records

```
DELETE FROM users      <-- table
WHERE  status = 'D'    <-- delete criteria
```

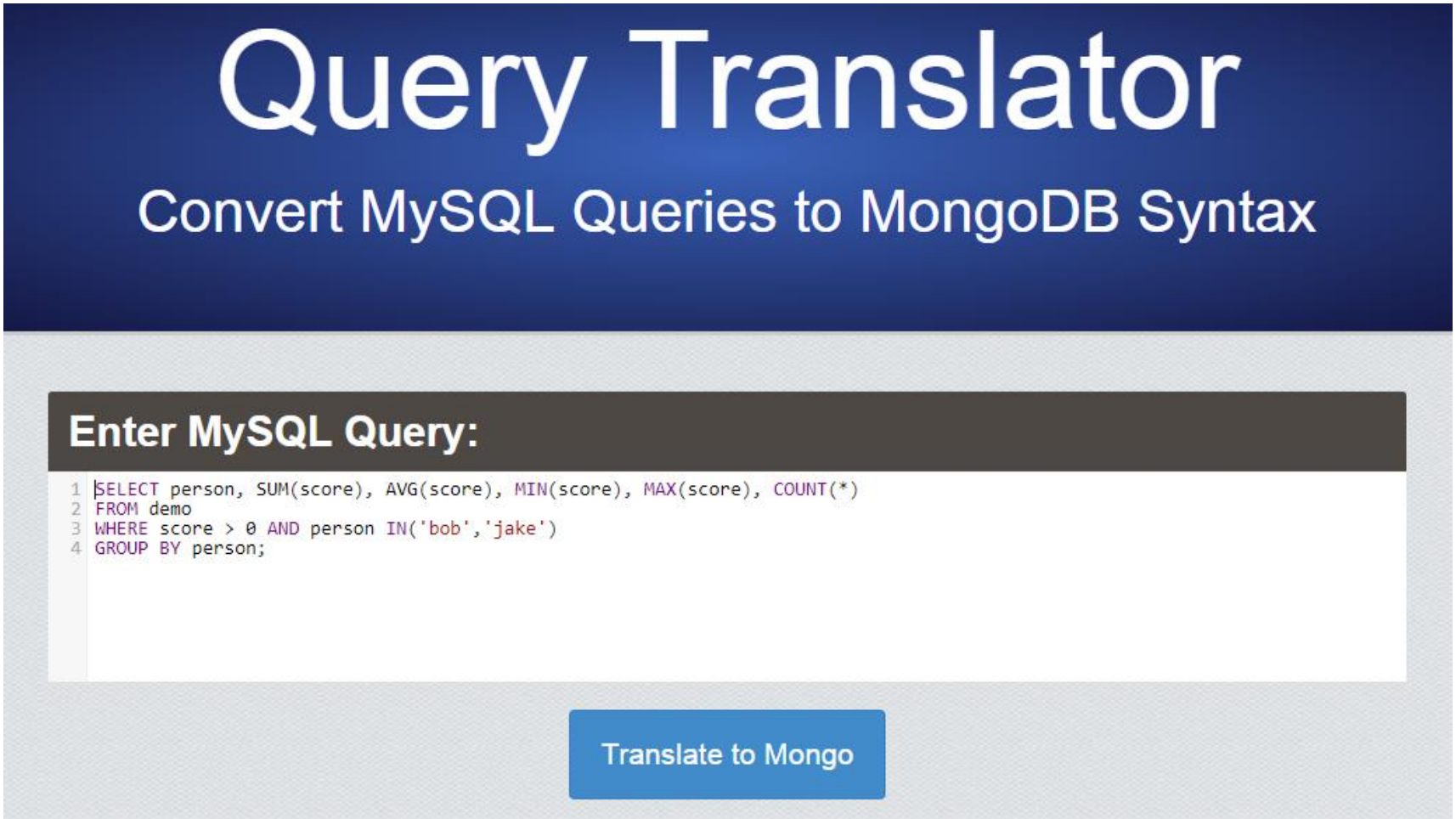
```
db.users.remove(       <-- collection
   { status: "D" }     <-- remove criteria
)
```

CRUD operations

- Aggregation - Equivalent to GROUP BY

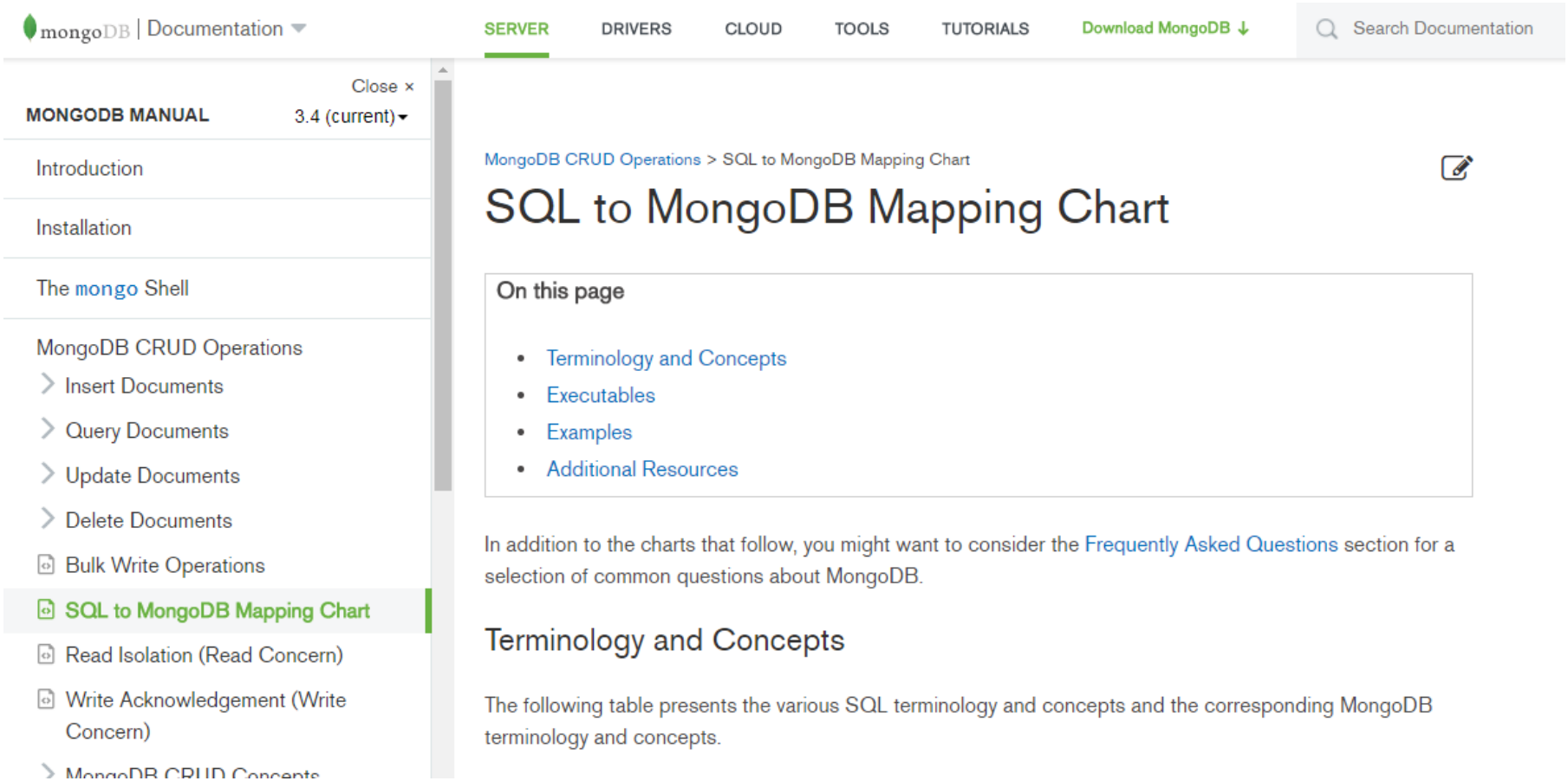


Converting from SQL to MongoDB



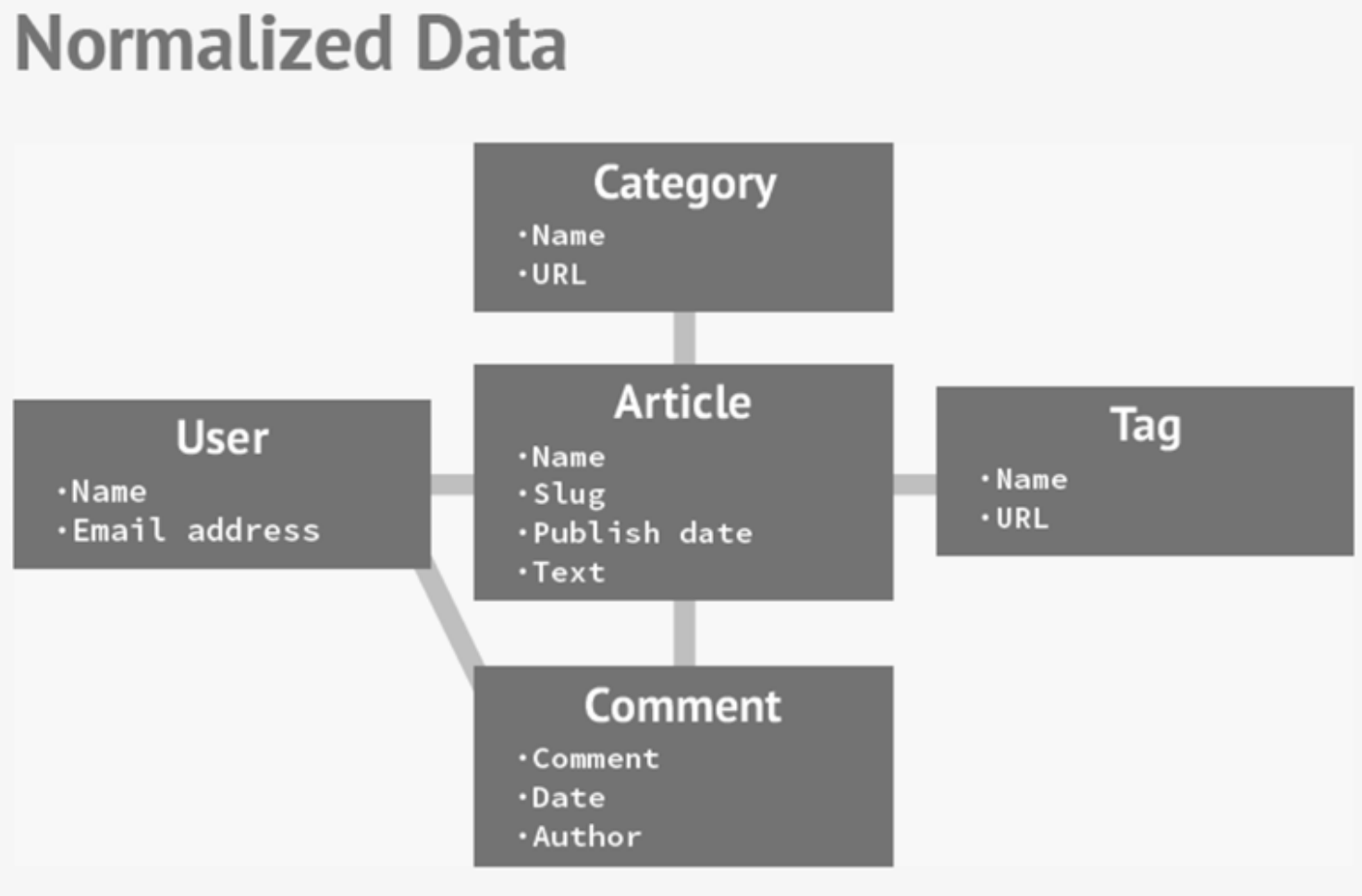
http://www.querymongo.com/

Converting from SQL to MongoDB

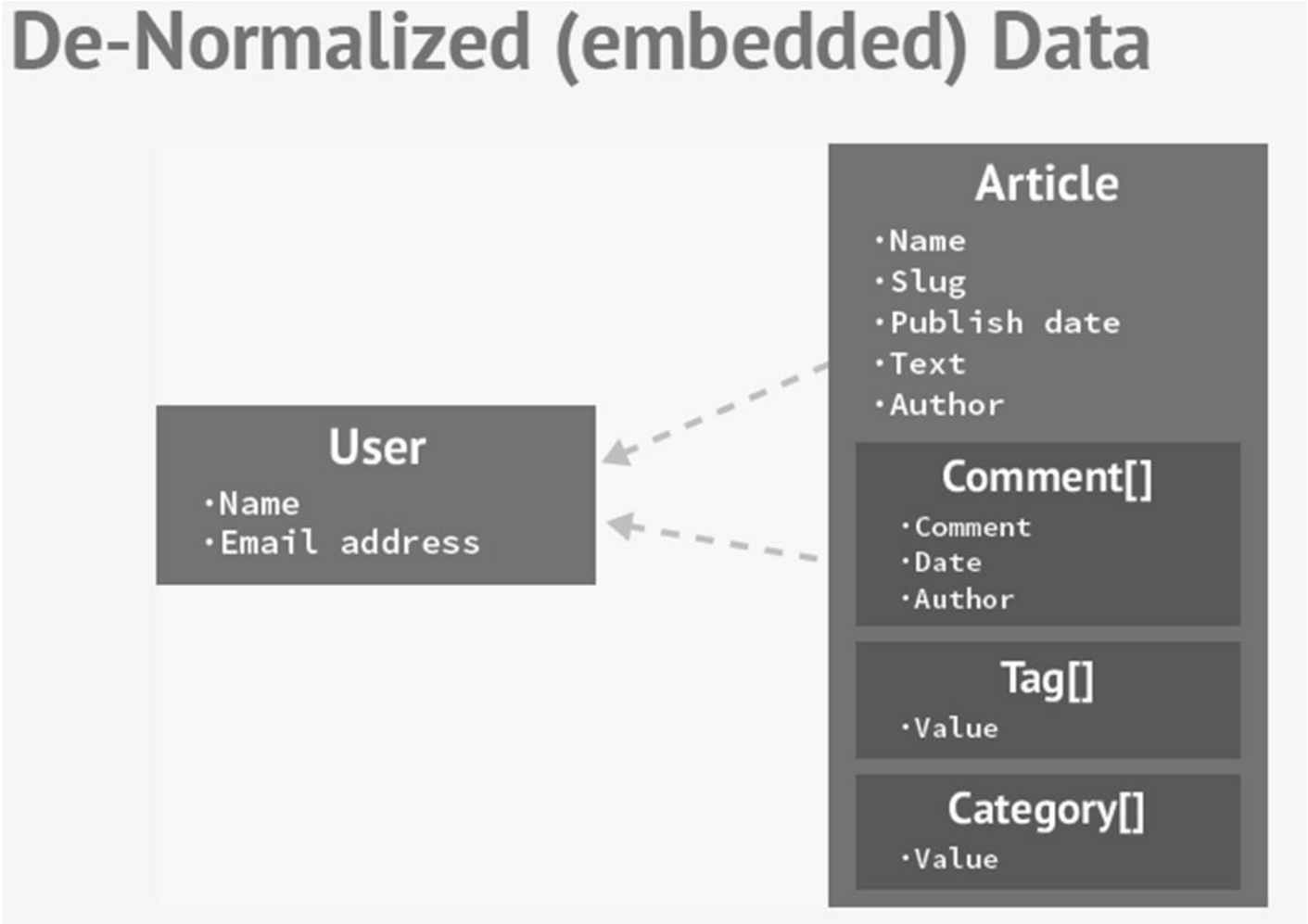


https://docs.mongodb.com/manual/reference/sql-comparison/

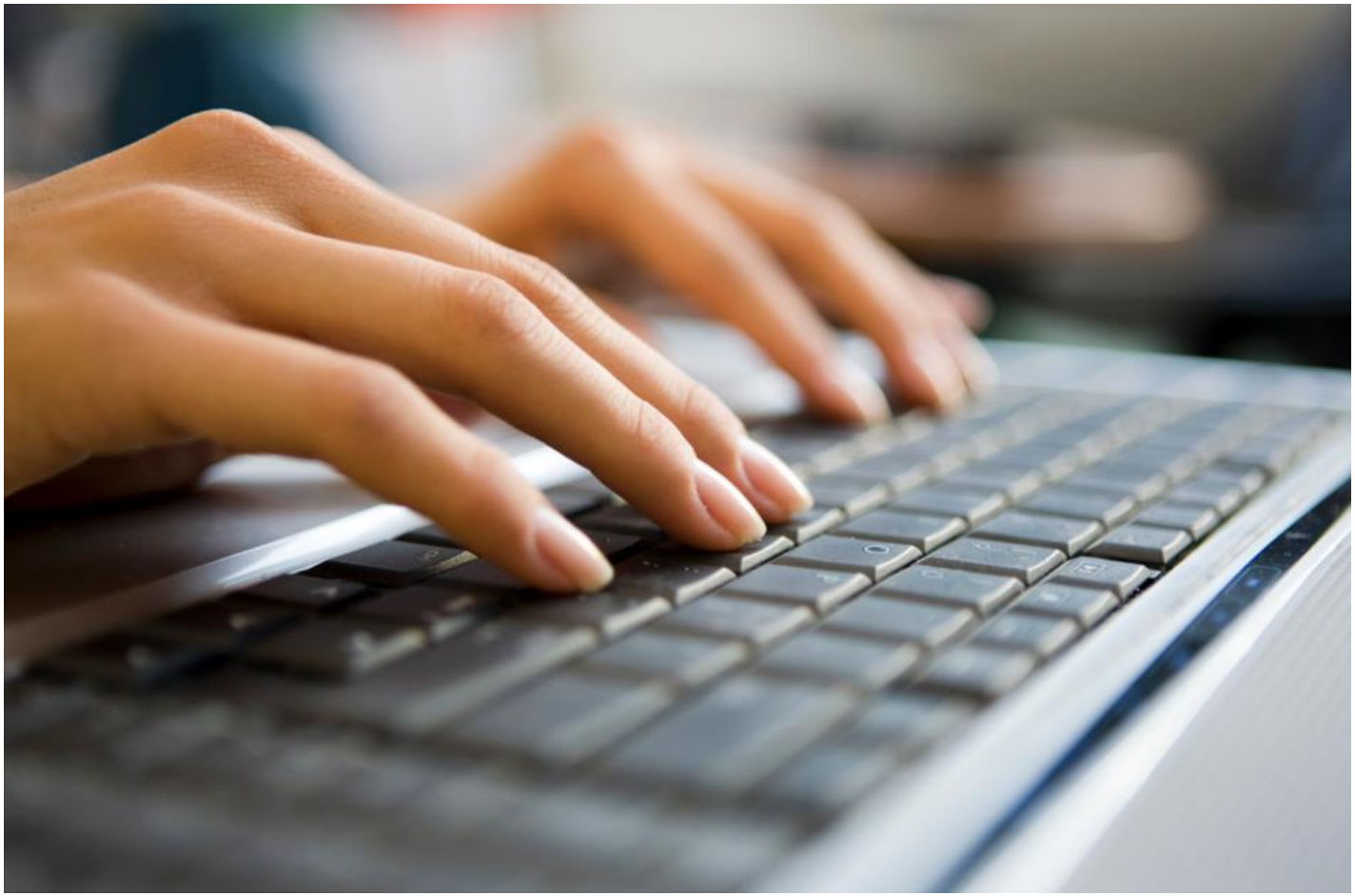
Schematic design



Scheme design



MongoDB - HandsOn



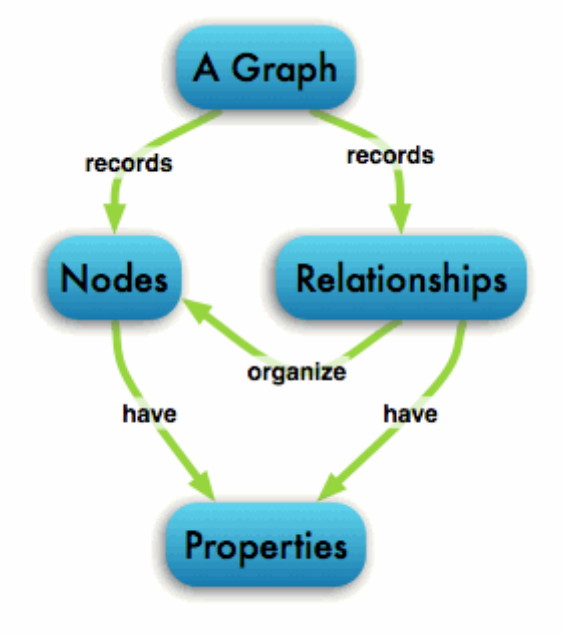
Cheat Sheet

Use Cases

- Any application that needs to use semi-structured data
- Applications with high volume of information
- Document and Content Management Systems
- Rapid development / Agile methodologies
- Machine-generated data (logs, sensors, etc.)
- It is not appropriate when there is more than one data center

Database Workshop

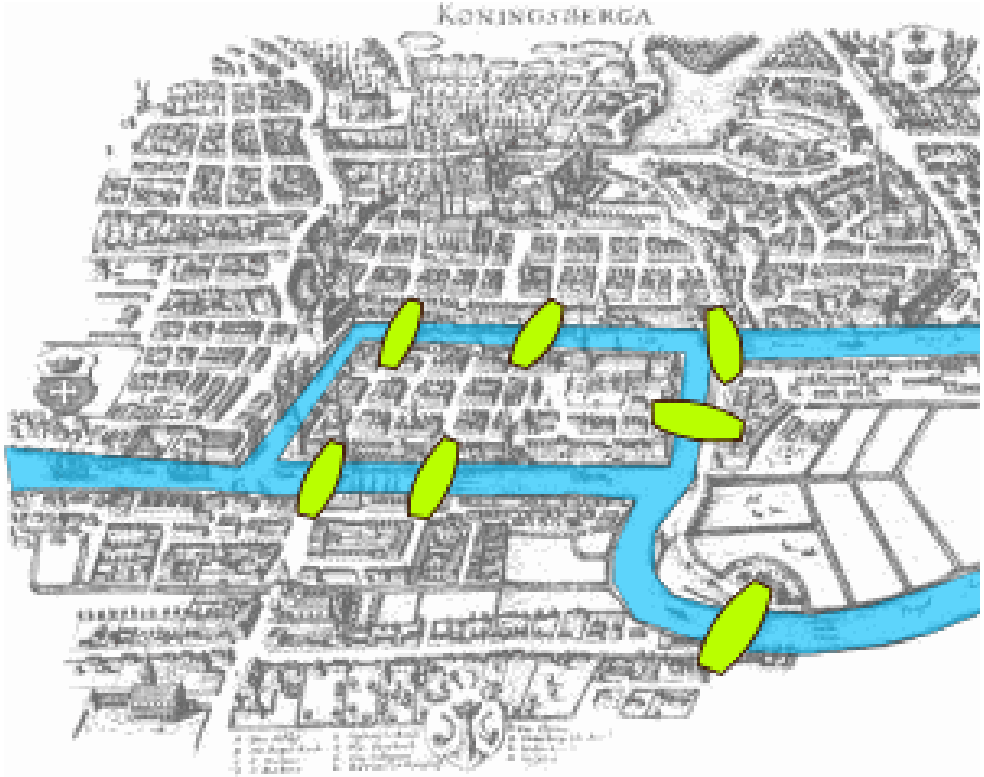
- Material
- Use case
- Relational Databases
- NoSQL
- Riak
- Apache Cassandra
- MongoDB
- Neo4j



Neo4j

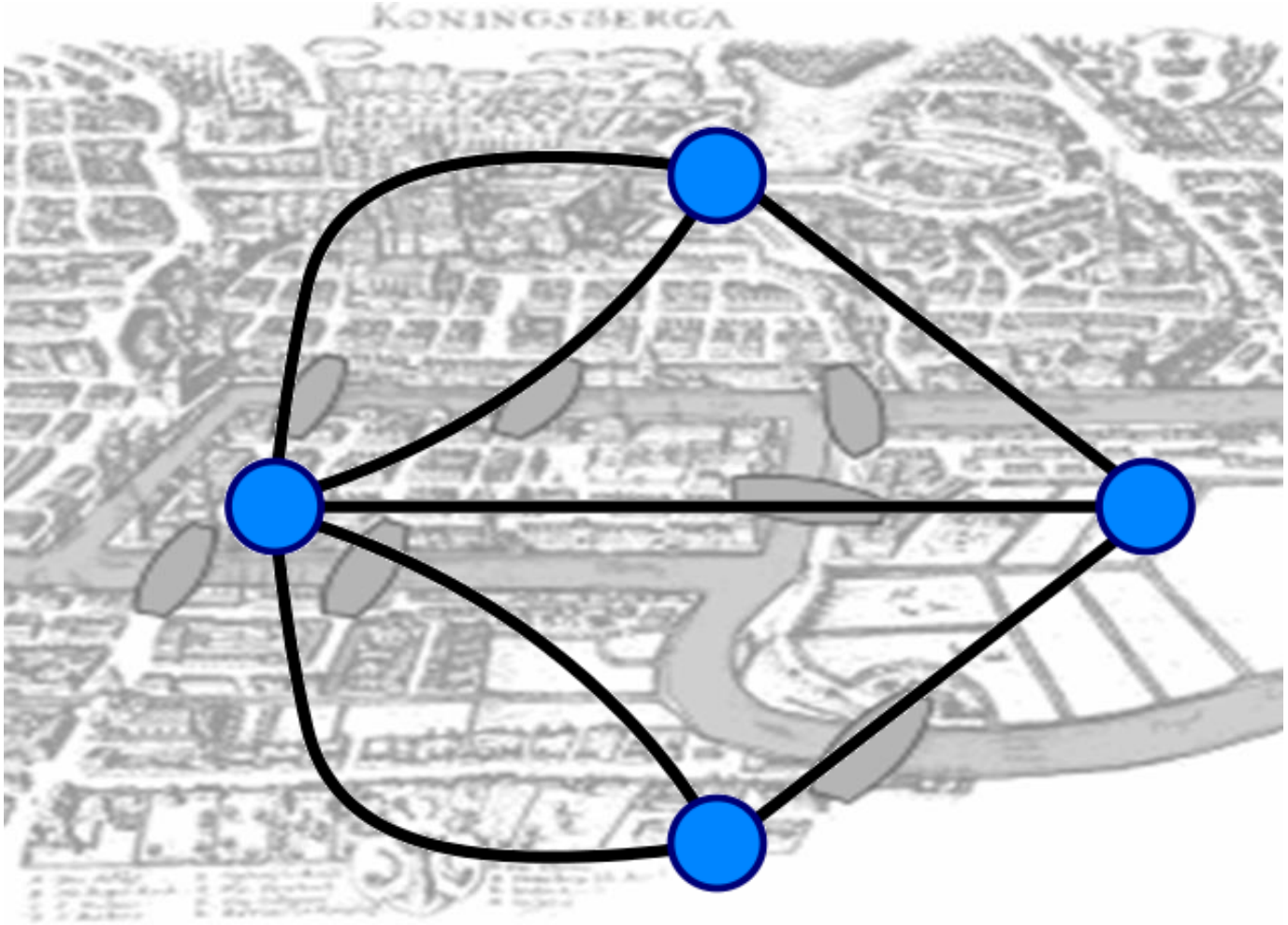
- It is a network oriented database (stores information as nodes and relationships)
- Implemented in java in 2010
- Querys: Proprietary language called Cypher that allows you to explore connections between information
- REST Interface
- It is not necessary to declare a scheme
- Prioritizes consistency and availability
- ACID

Graphite Theory - Königsberg Bridges



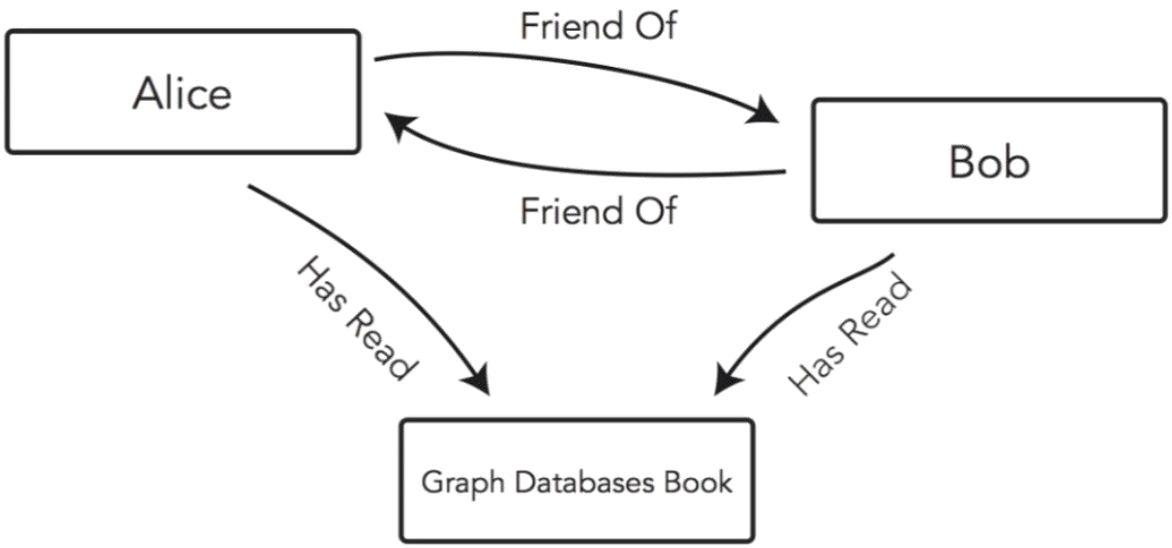
Leonhard Euler
1707-1783

Graphite Theory - Königsberg Bridges

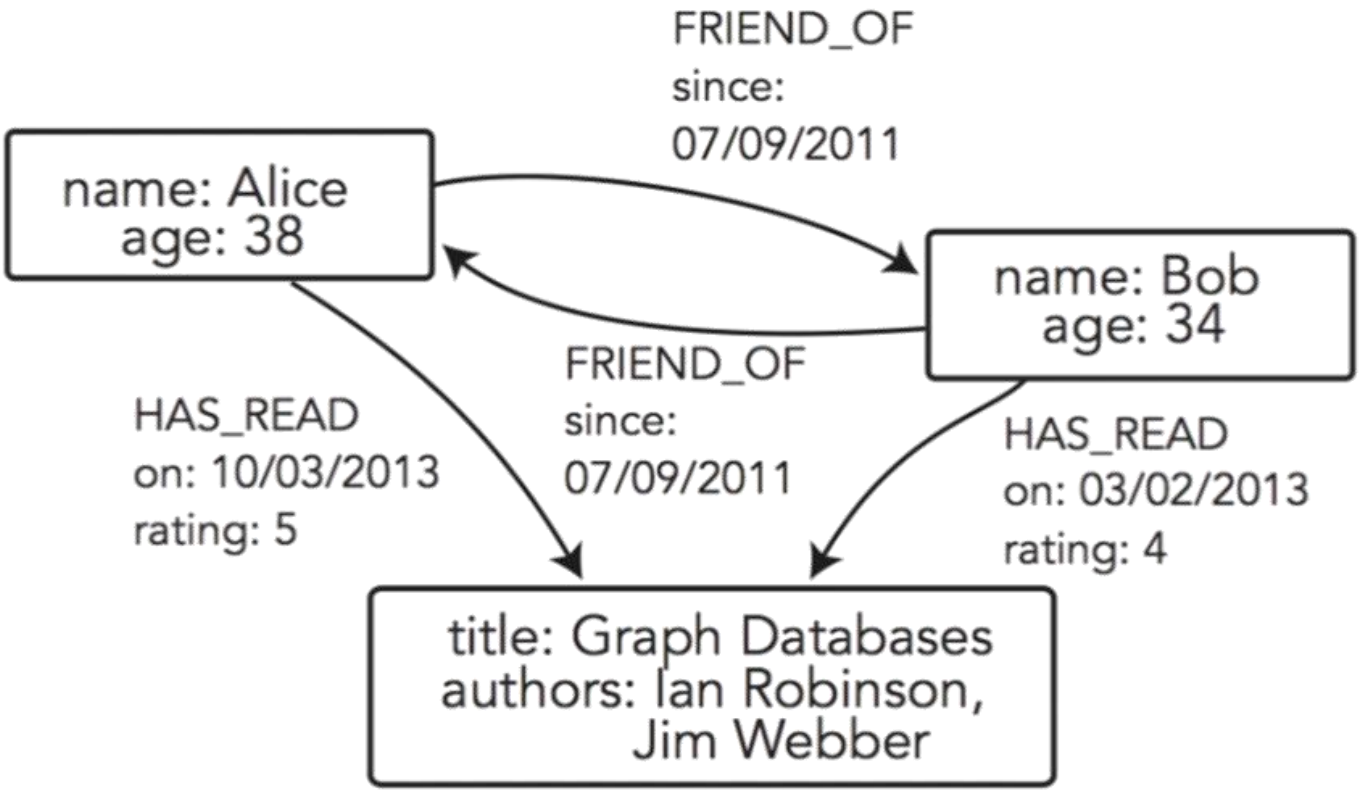


Nodes and Relations

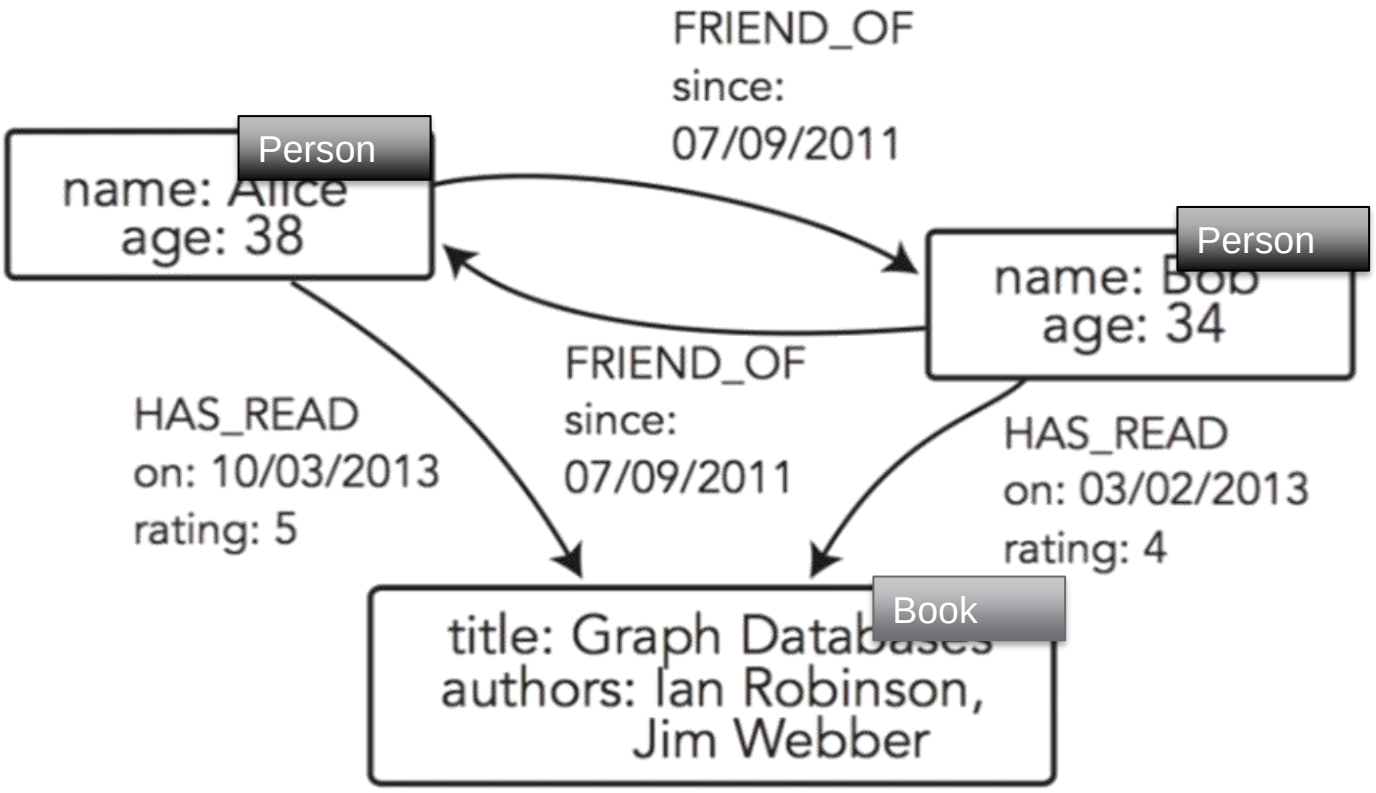
- A network is built the way people really think
- The nodes or vertices represent entities
- The edges represent relationships



Properties

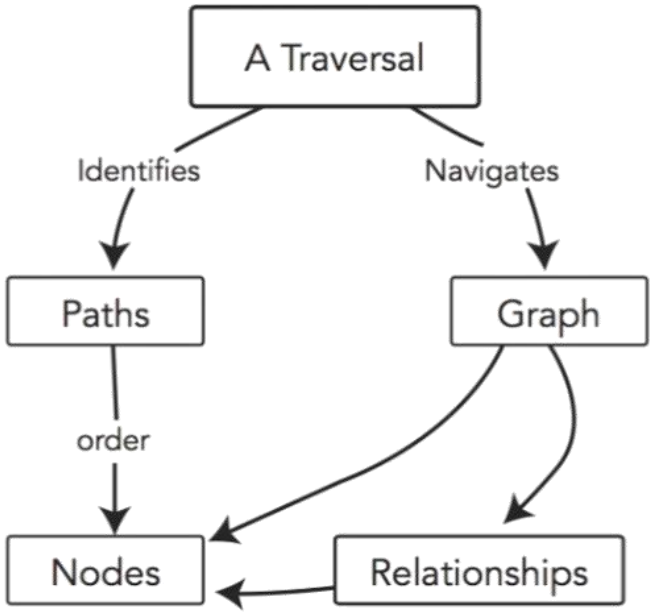

FRIEND_OF
since:
07/09/2011
name: Alice
age: 38
name: Bob
age: 34
FRIEND_OF
since:
07/09/2011
HAS_READ
on: 10/03/2013
rating: 5
HAS_READ
on: 03/02/2013
rating: 4
title: Graph Databases
authors: Ian Robinson,
Jim Webber

Tags


FRIEND_OF
since:
07/09/2011
Person
name: Alice
age: 38
Person
name: Bob
age: 34
FRIEND_OF
since:
07/09/2011
HAS_READ
on: 10/03/2013
rating: 5
HAS_READ
on: 03/02/2013
rating: 4
Book
title: Graph Databases
authors: Ian Robinson,
Jim Webber

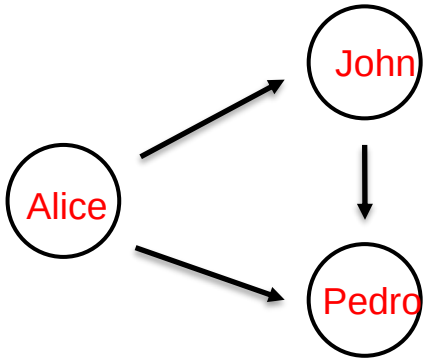
Searching for information

- Locate a node and explore other nodes through their relationships

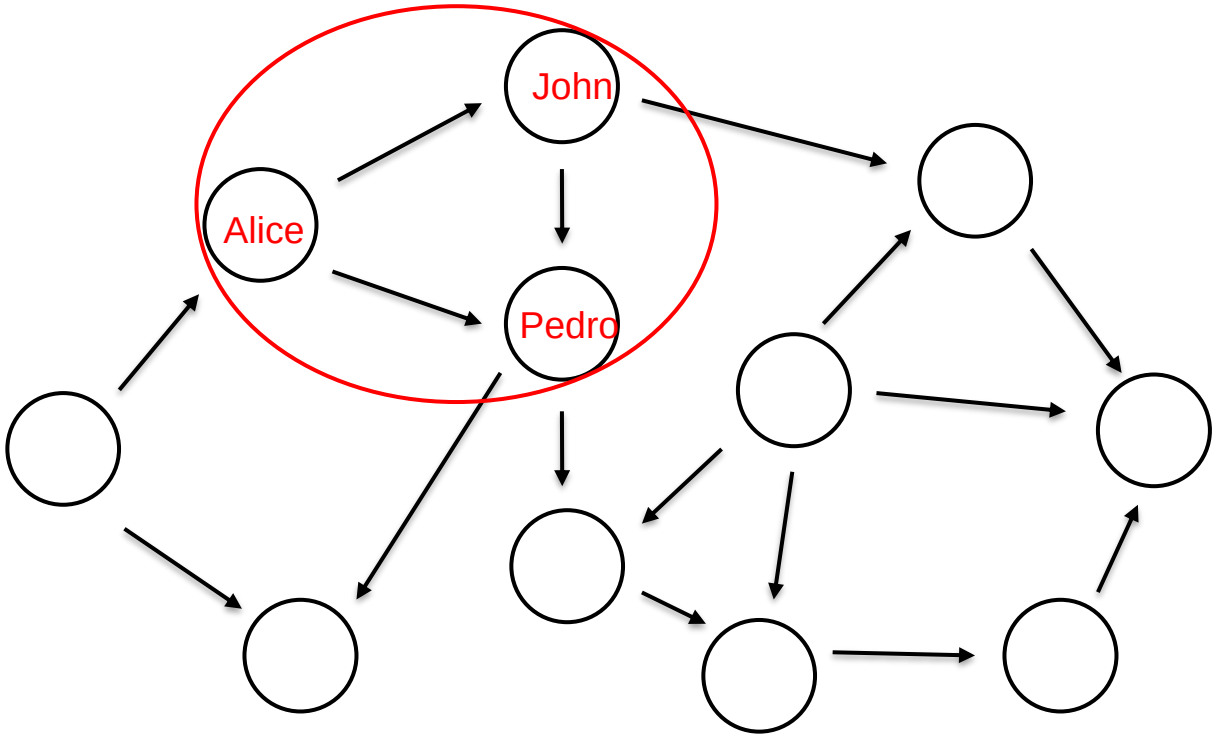


Searching for information

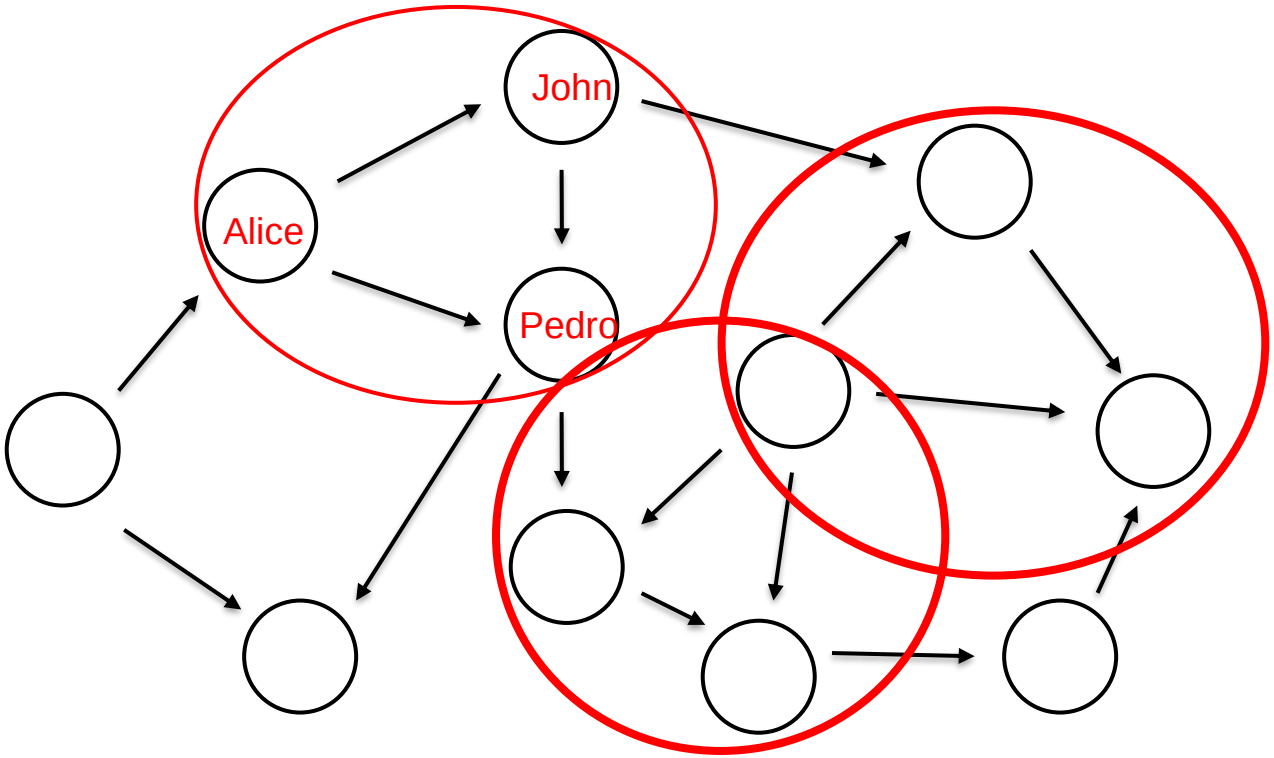
- Locate a node and explore other nodes through their relationships
- or we can identify patterns



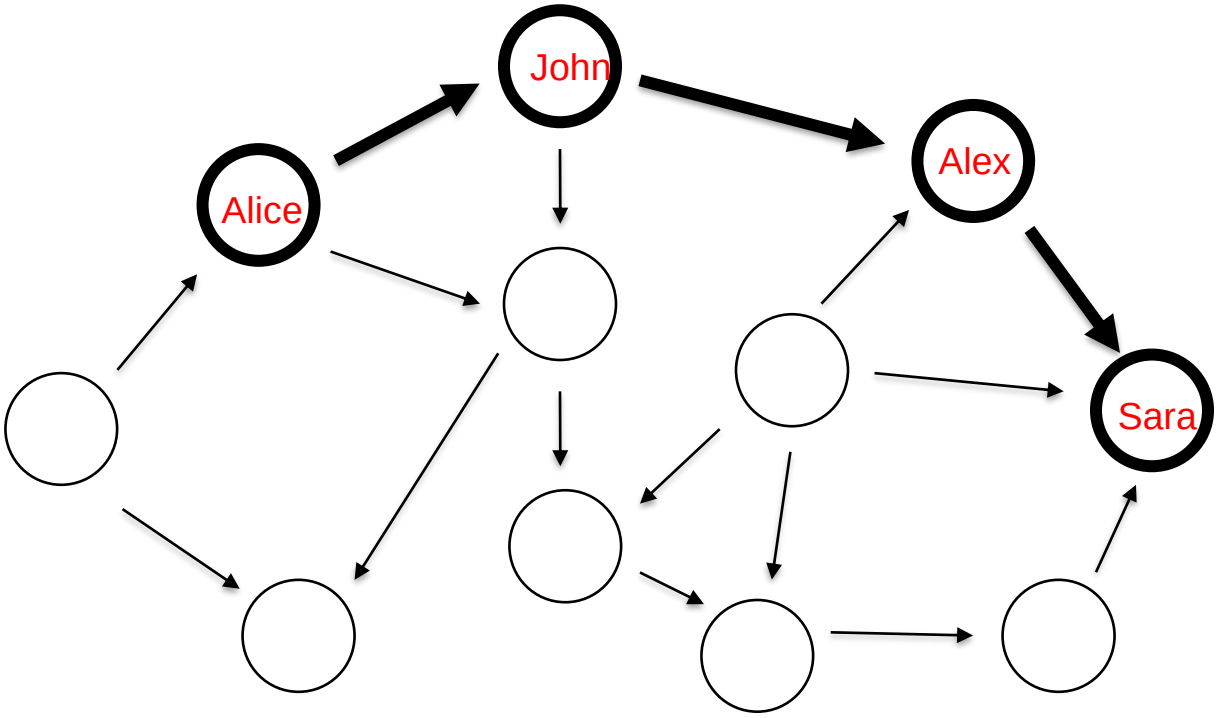
Patterns



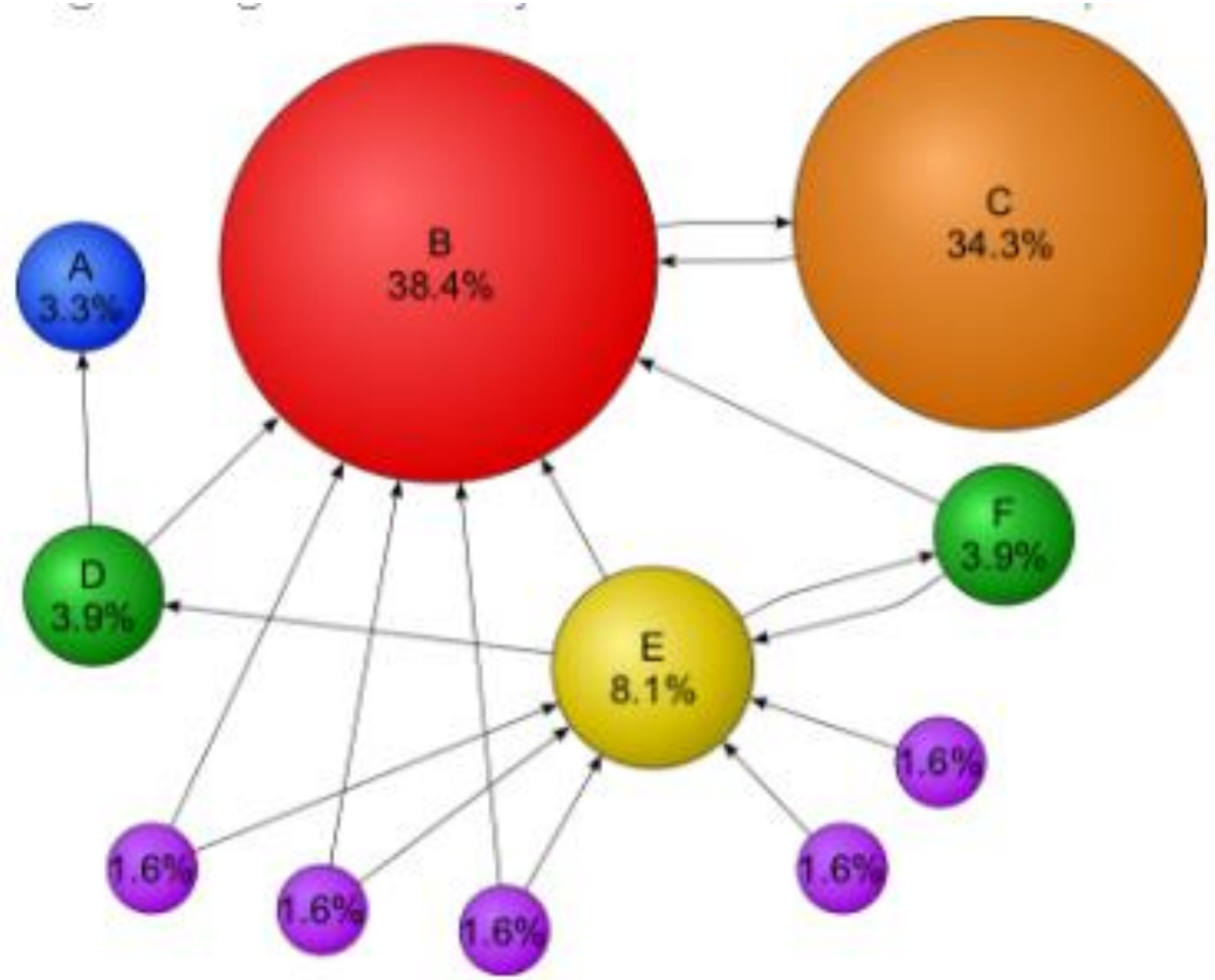
Patterns



Algorithms - The shortest path



Algorithms - PageRank



Cypher

- Language with a philosophy similar to SQL
- Allows you to create nodes and/or relationships, maintain them or delete them
- Finding patterns
- Execute algorithms implemented in the DB

```
CREATE (ann:Person { name: 'Ann' })
RETURN ann
```

```
MATCH p=shortestPath(
  (a:Person { name: 'Ann' })-[:KNOWS]-(b:Person { name: 'Dan'})
)
RETURN p
```

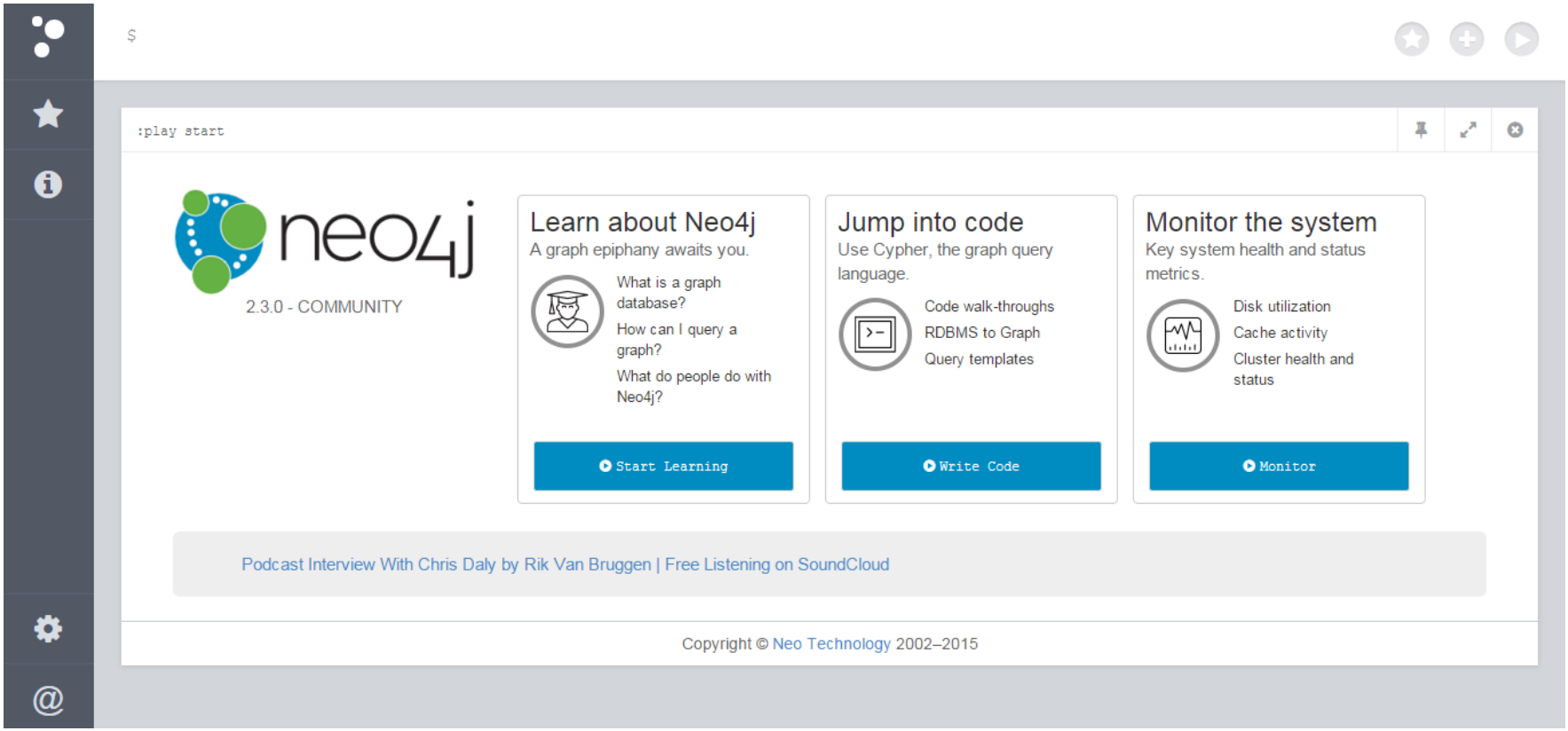
```
MATCH (a:Person { name: 'Ann' }),
    (b:Person { name: 'Dan' })
CREATE (a)-[:KNOWS]->(b)
```

```
MATCH (Alex:Person {name:"Alex"})
DELETE Alex
```

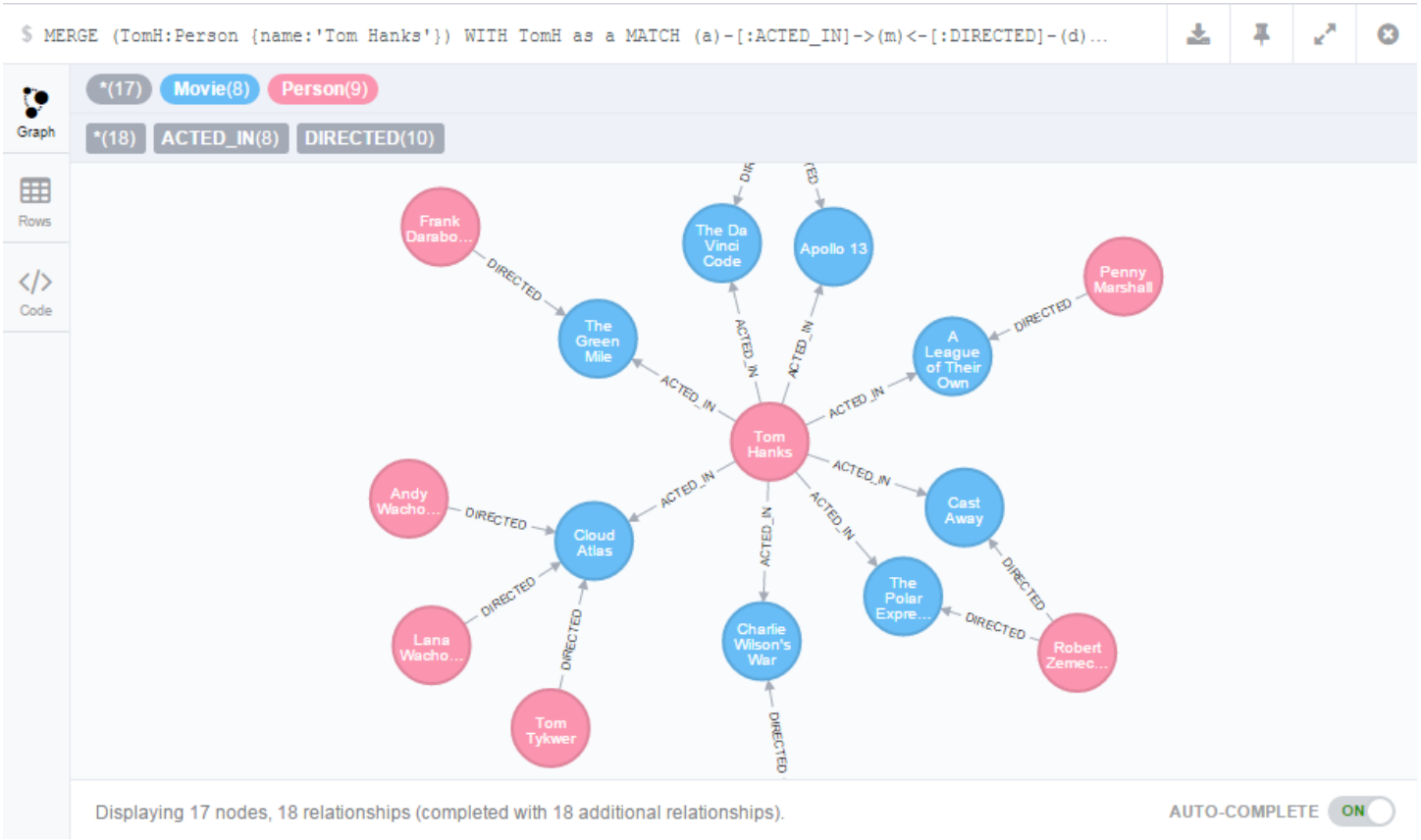
```
MATCH (n:Person { name: 'Ann' })
RETURN n
```

```
MATCH (n:Person {name : "Ann"})
SET n.hair = "Brown"
```

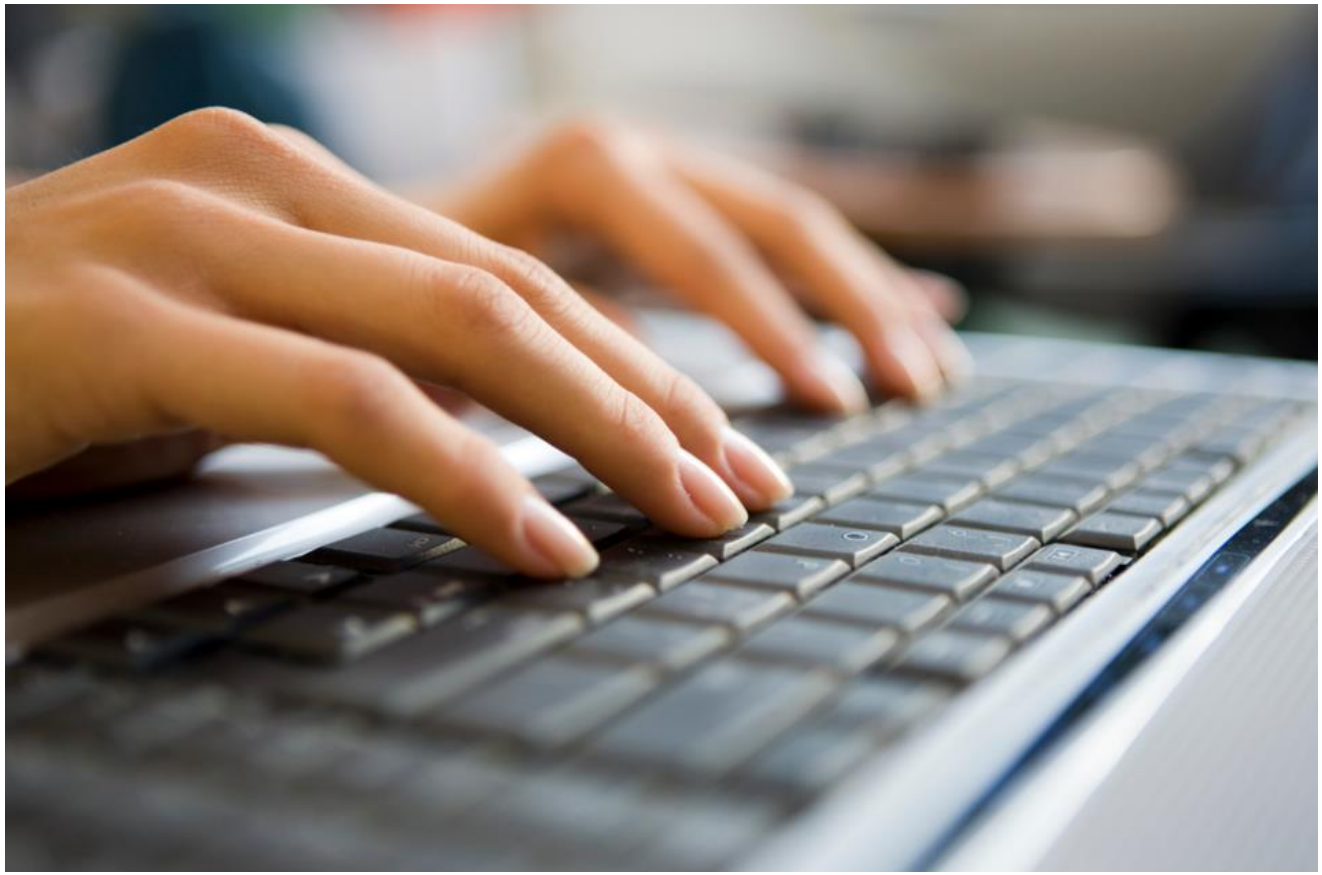
Neo4j


$
:play start
neo4j
2.3.0 - COMMUNITY
Learn about Neo4j
A graph epiphany awaits you.
What is a graph database?
How can I query a graph?
What do people do with Neo4j?
Start Learning
Jump into code
Use Cypher, the graph query language.
Code walk-throughs
RDBMS to Graph
Query templates
Write Code
Monitor the system
Key system health and status metrics.
Disk utilization
Cache activity
Cluster health and status
Monitor
Podcast Interview With Chris Daly by Rik Van Bruggen | Free Listening on SoundCloud
Copyright © Neo Technology 2002–2015

Neo4j - Visualization



Neo4j - HandsOn



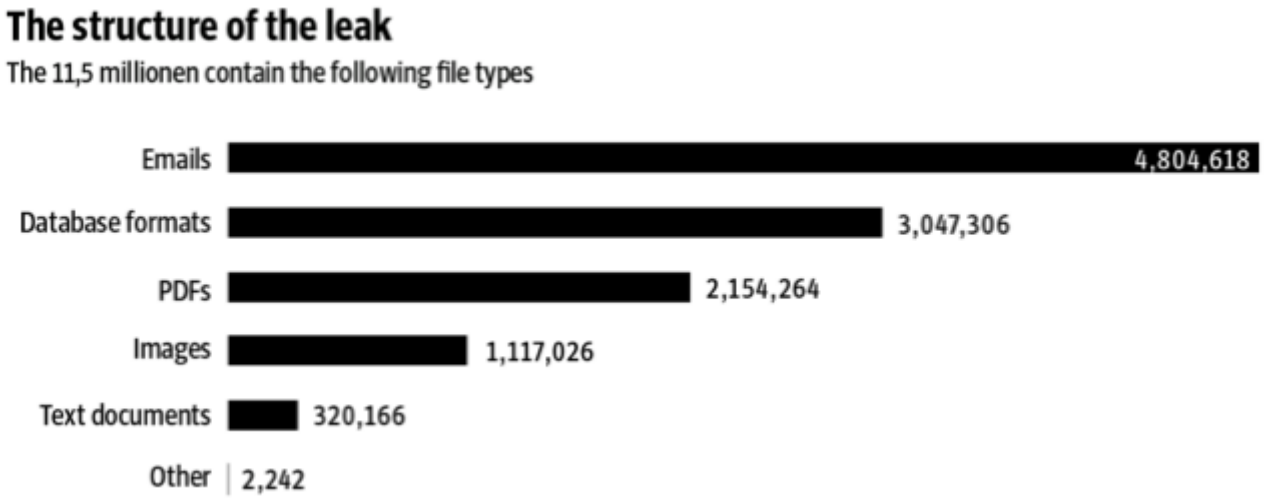
Cheat Sheet

Use Cases

- Optimal for applications that need to look for relationships in information
- Social networking
- Fraud detection by identifying patterns
- Real-time recommendations
- Data center management - devices, users, etc.
- Master Data Systems Management
- Identity and access management

Panama's papers

Panama's papers



Panama's papers

The Steps Involved in the Document Analysis

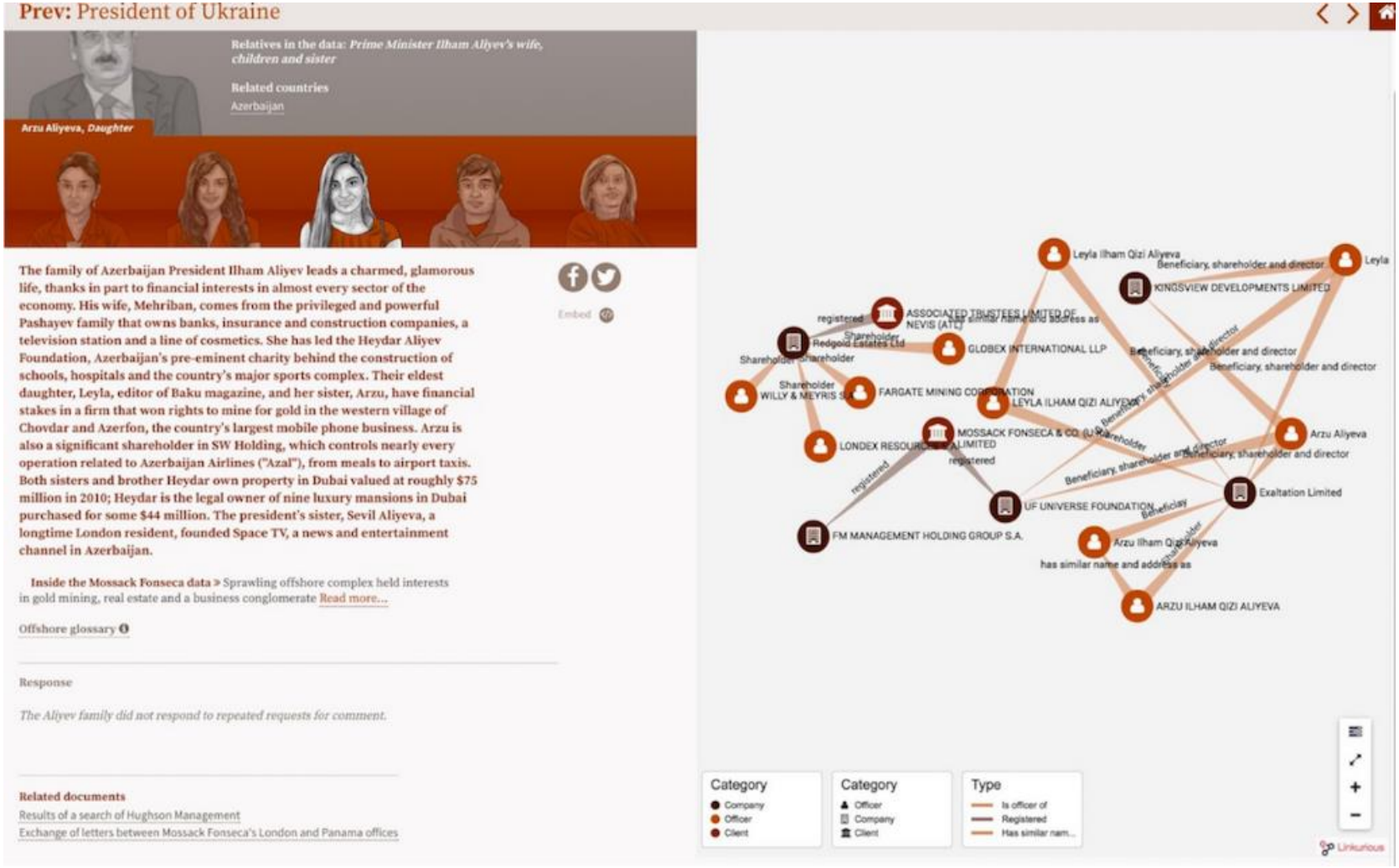
1. Acquire documents
2. Classify documents
    a. Scan / OCR
    b. Extract document metadata

3. Whiteboard domain
    a. Determine entities and their relationships
    b. Determine potential entity and relationship properties
    c. Determine sources for those entities and their properties

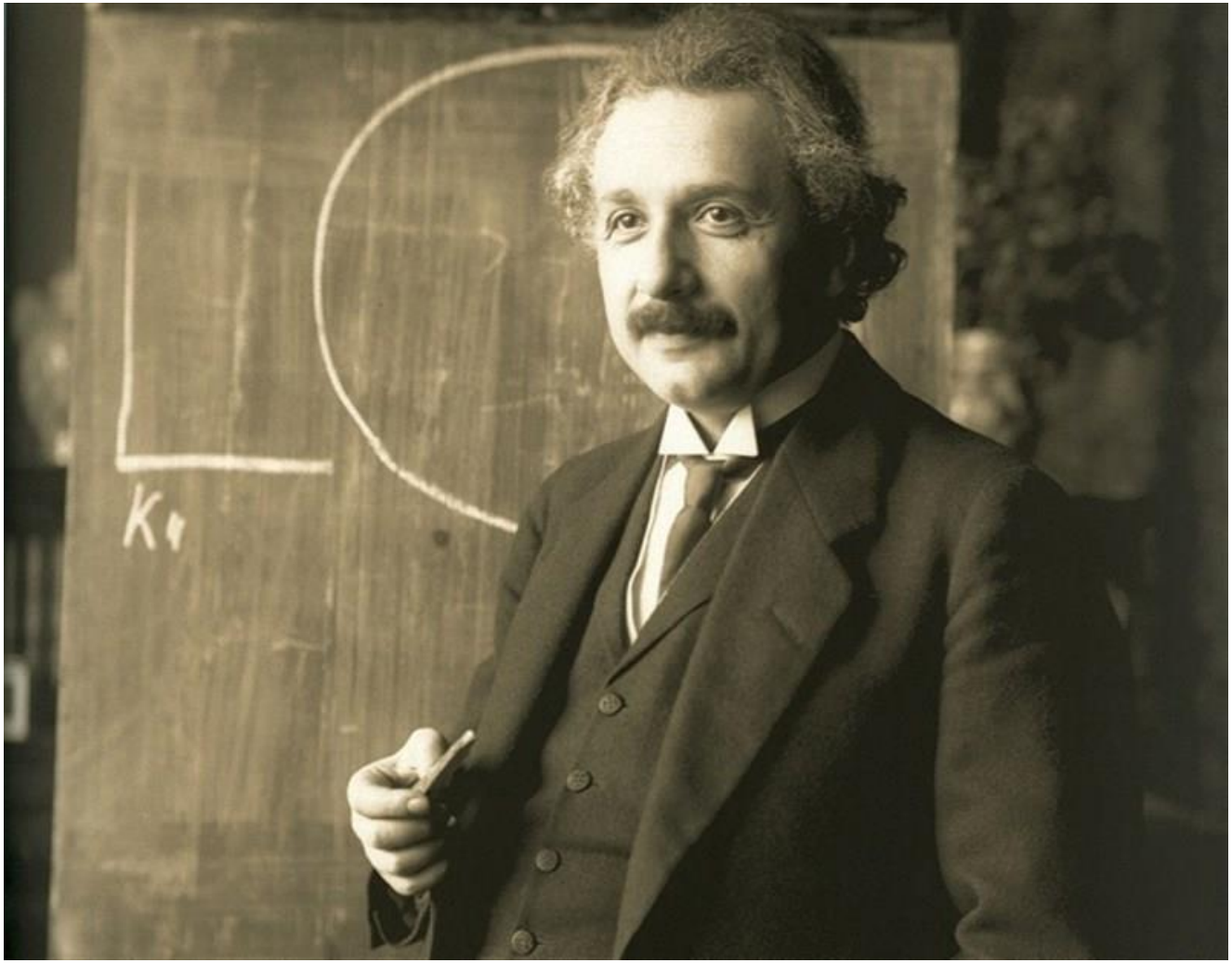
4. Work out analyzers, rules, parsers and named entity recognition for documents
5. Parse and store document metadata and document and entity relationships
    a. Parse by author, named entities, dates, sources and classification

6. Infer entity relationships
7. Compute similarities, transitive cover and triangles
8. Analyze data using graph queries and visualizations

Panama's papers



Einstein's Riddle



Einstein's Riddle

- This seemingly simple Einstein's riddle is based on a number of considerations and one question.
- These are about a group of five people of different nationalities, with five different pets, consuming a certain brand of tobacco, drinking a certain drink and living in a different house entirely in each case.

Who owns the fish?

Einstein's Riddle

- The Englishman lives in the red house.
- The Swede has a dog.
- The Dane drinks tea.
- The Norwegian lives in the first house.
- The German smokes Prince.
- The green house is immediately to the left of the white one.
- The owner of the green house drinks coffee.

Einstein's Riddle

- The person who smokes Pall Mall breeds birds.
- The owner of the yellow house smokes Durnhill.
- The man who lives in the house downtown drinks milk.
- The man who smokes Blends lives next door to the man who has a cat.
- The man who has a horse lives next to the man who smokes Dunhill.
- The man who smokes Bluemaster drinks beer.
- The man who smokes Blends is a neighbor of the man who drinks water.
- The Norwegian lives next door to the blue house.

THANKS FOR YOUR ATTENTION

Daniel Villanueva Jiménez
daniel.villanueva@immune.institute
@dvillaj