

NoSQL and the evolution of databases

MongoDB & GIS

Simon Oulevay
simon.oulevay@heig-vd.ch
2016, Comem+

Inspired by the "GIS with NoSQL" workshop at the Helsinki 2016 AGILE conference:
<https://www.unibw.de/inf4/professors/geoinformatics/agile-2016-workshop-gis-with-nosql>

By Wolfgang Reinhardt, Eszter Gálicz & Stephan Schmid



“NoSQL market expected to reach \$4.2 Billion globally by 2020”

“Graph based NoSQL market poised to grow at a rapid CAGR of 40.8%, during 2014-2020”

-- GISuser, April 2015,
citing Allied Market Research report

What is NoSQL?

**3 DATABASE ADMINS
WALKED INTO
A NOSQL BAR...**

**A LITTLE WHILE LATER
THEY WALKED OUT BECAUSE
THEY COULDN'T FIND A TABLE**

Structural Variety in Big Data

Historically, SQL favored:

- Business data = money
- In-house operation

Tabular & relational data:
customers, stock items,
transactions, etc.

However, there's more data (and Big Data):

- Stock trading: 1D sequences, i.e. **arrays**
- Social networks: large, homogeneous **graphs**
- Ontologies: small, heterogeneous **graphs**
- Climate modelling: 4D/5D **arrays**
- Satellite imagery: 2D/3D **arrays**
- Genome: long string **arrays**
- Particle physics: **sets** of events
- Bio taxonomies: **hierarchies** (e.g. XML)
- Documents: **key/value** stores



What is NoSQL?

Modern web-scale databases

non-relational
distributed
open-source
horizontally scalable

NoSQL databases are **next-generation databases** addressing **some of these points**.

schema-free
easy replication support
simple API
eventually consistent / BASE (not ACID)
a huge amount of data
structural variety

And often some of these points as well.

<http://nosql-database.org>

NoSQL is *Fast*

On >50GB of data:

MySQL:
Writes: **300ms** avg
Reads: 350ms avg



Cassandra:
Writes: **0.12ms** avg
Reads: 15ms avg



VS.



Atomicity
Consistency
Isolation
Durability

Basically **A**vailable
Soft state
Eventually consistent

Characteristics:

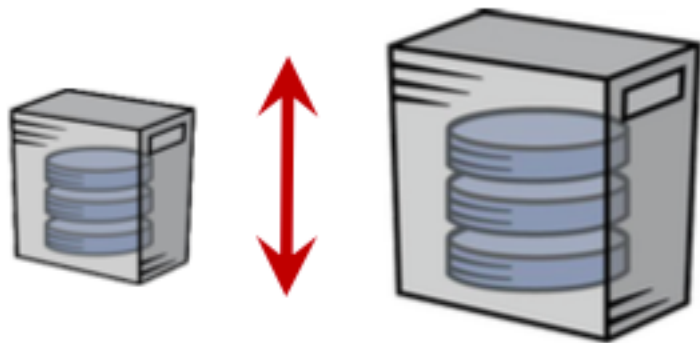
- ACID
- Strong promises, desirable for many domains (e.g. banks)
- Computationally expensive, in particular when distributed

Characteristics:

- Service availability first
- Approximate answers OK (write not guaranteed)
- No transactions
- Weak consistency (stale data OK)
- Simpler & faster

Scalability

Vertical scaling
performance on single server



- RAM, avoid random disk I/O
- Minimize overhead for locking & latching
- Minimize network calls between servers

Horizontal scaling
over multiple servers



- Partition
- Replicate
- Automatic failure recovery
- Zero downtime

nosql

nosql

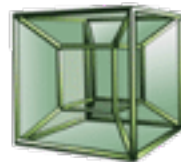
APACHE
HBASE

 **Cassandra**


CouchDB
relax

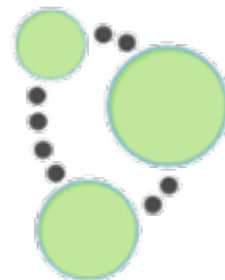


riak



mongoDB

HYPERTABLE INC



Neo4j

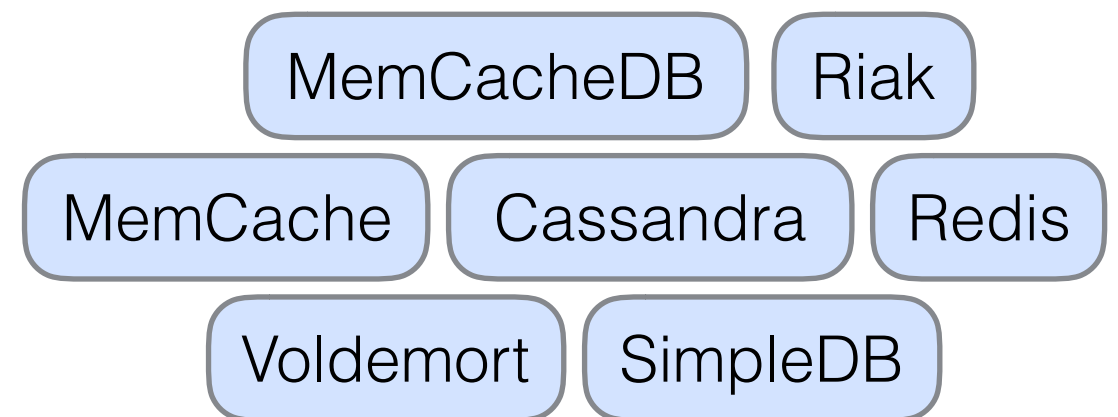


redis

Key/Value Stores

- Essentially a (distributed) **hash table**
- Fast, direct access to (often smallish) data values
- Operations:
 - Put(key, value)
 - value = Get(key)
 - More operations for some (e.g. Redis)

Key	Value
K1	AAA,BBB,CCC
K2	AAA,BBB
K3	AAA,DDD
K4	AAA,2,01/01/2015
K5	3,ZZZ,5623



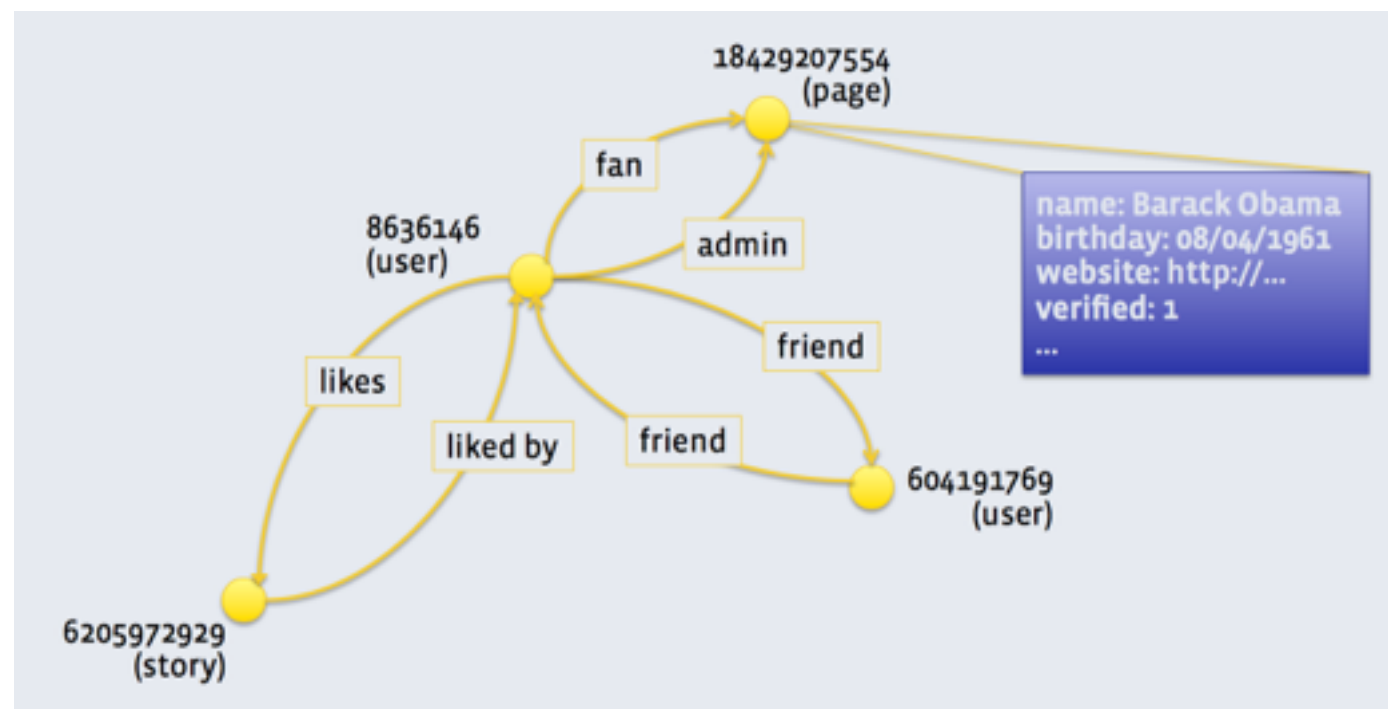
Document-oriented Databases

- Like key/value stores, but the value is a (often large) **document**
- Semi-structured data
- Searchable (e.g. full text search)
- Useful for content-oriented applications (e.g. Facebook, Amazon)



Graph Databases

- Conceptual model: **attributed graph**
- Application: ontologies, social networks, Internet of Things, geometric modeling



RDF/SPARQL

Facebook TAO

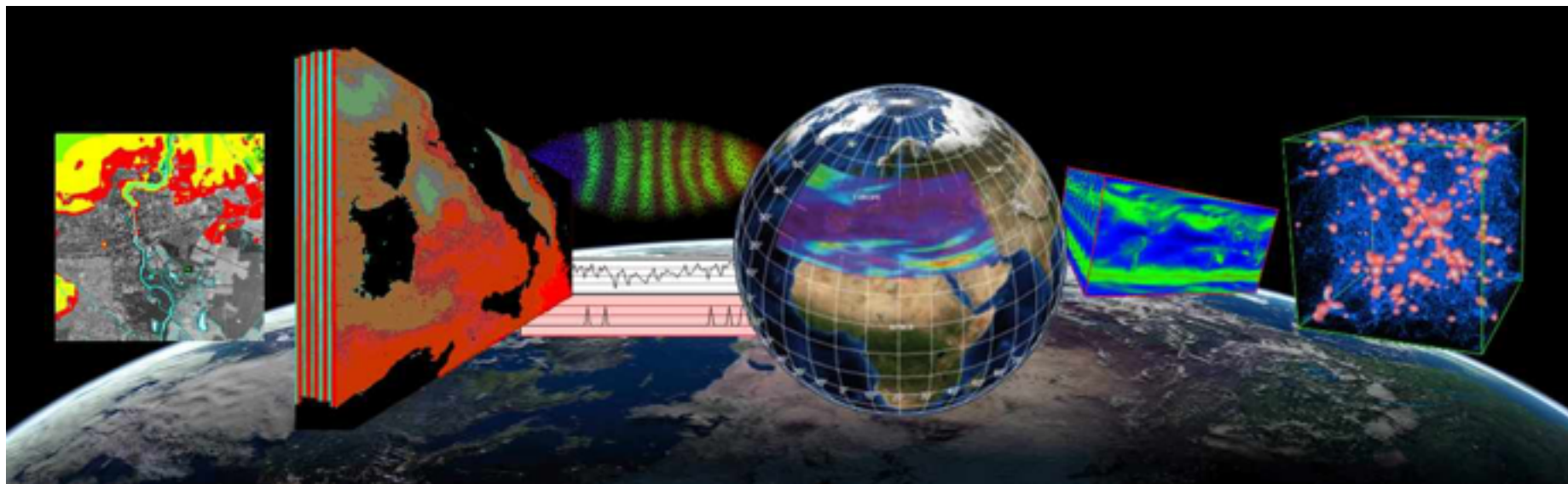
Neo4j

Array Databases

- Conceptual model: **n-D arrays**
- Application: satellite imagery & signal processing

rasdaman

DaggerDB



Geospatial Processing

- OGC Web Coverage Service (WCS)
 - Digital geospatial information representing space/time-varying phenomena
 - Formatting on-the-fly (e.g. trim, slice)
- OGC Web Coverage Processing Service (WCPS)
 - Spatio-temporal datacube analytics language
 - Time series analysis, Image processing, sensor fusion, pattern mining



NewSQL

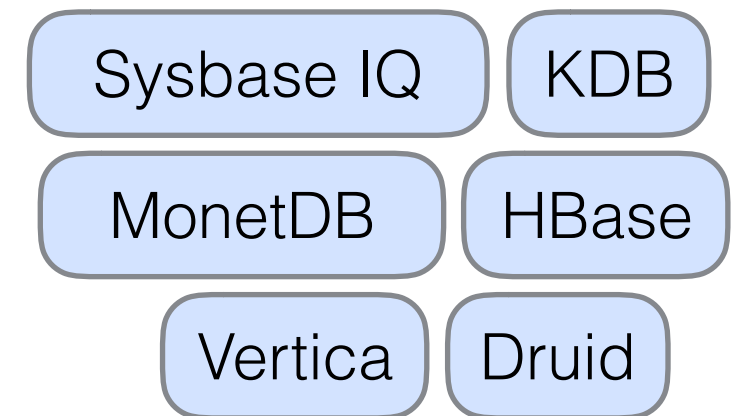
**THE
EMPIRE
STRIKES BACK**

NewSQL

- *Goal:* same **scalable performance** as NoSQL for common **read-write workloads** while maintaining **ACID guarantees** of traditional DBMSs.
- How?
 - Column stores
 - Main memory DB
 - New datatypes & functionality in SQL

Column Stores

- Relational DBMS with:
 - Columnar** storage architecture
 - Compression & novel processing
- Faster than row-based stores? Only if you don't often need to retrieve entire rows. So it depends on your workload.
- Not a new idea (TAXIR, 1969; KDB, 1993)



row based

ID	Product	Customer	Date	Sale
1	Beer	Thomas	2011-11-25	2 GBP
2	Beer	Thomas	2011-11-25	2 GBP
3	Vodka	Thomas	2011-11-25	10 GBP
4	Whiskey	Christian	2011-11-25	5 GBP
5	Whiskey	Christian	2011-11-26	5 GBP
6	Vodka	Alexei	2011-11-26	10 GBP
7	Vodka	Alexei	2011-11-26	10 GBP

column based

PRODUCT	CUSTOMER	DATE	SALE
Value	Value	Value	Value
Row IDs	Row IDs	Row IDs	Row IDs
Beer	Thomas	2011-11-25	2 GBP
Vodka	Christian	2011-11-26	10 GBP
Whiskey	Alexei		5 GBP

Main Memory DB

- RAM is faster than disk
 - Load the DB into RAM
 - Doable now with lower hardware price
 - Oracle ships 1PB RAM servers
- **Vertical scaling**, not horizontal

Oracle TimesTen

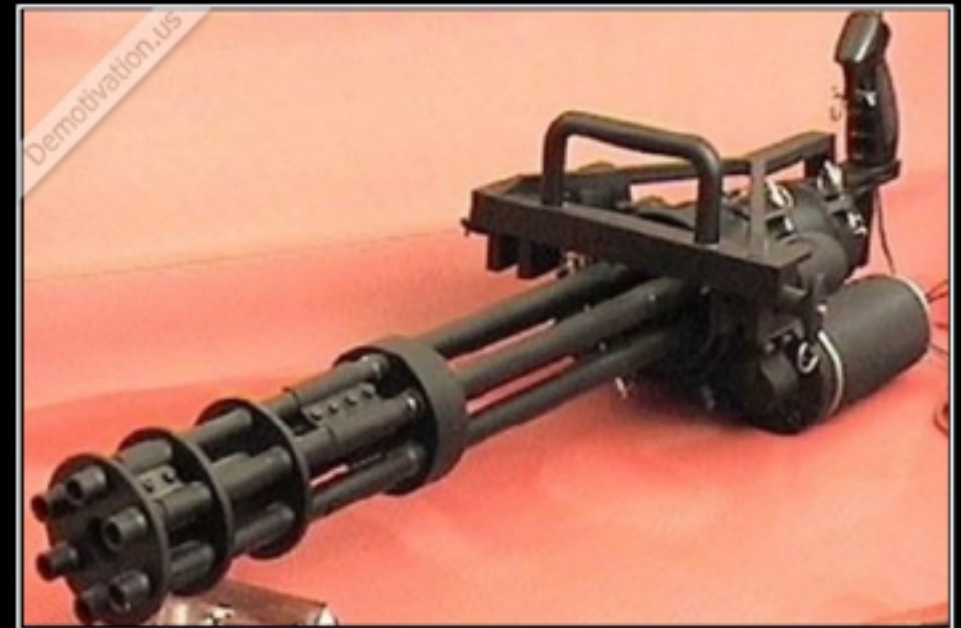
HyPer

Microsoft Hekaton

VoltDB

Calvin

IBM solidDB



BRUTE FORCE

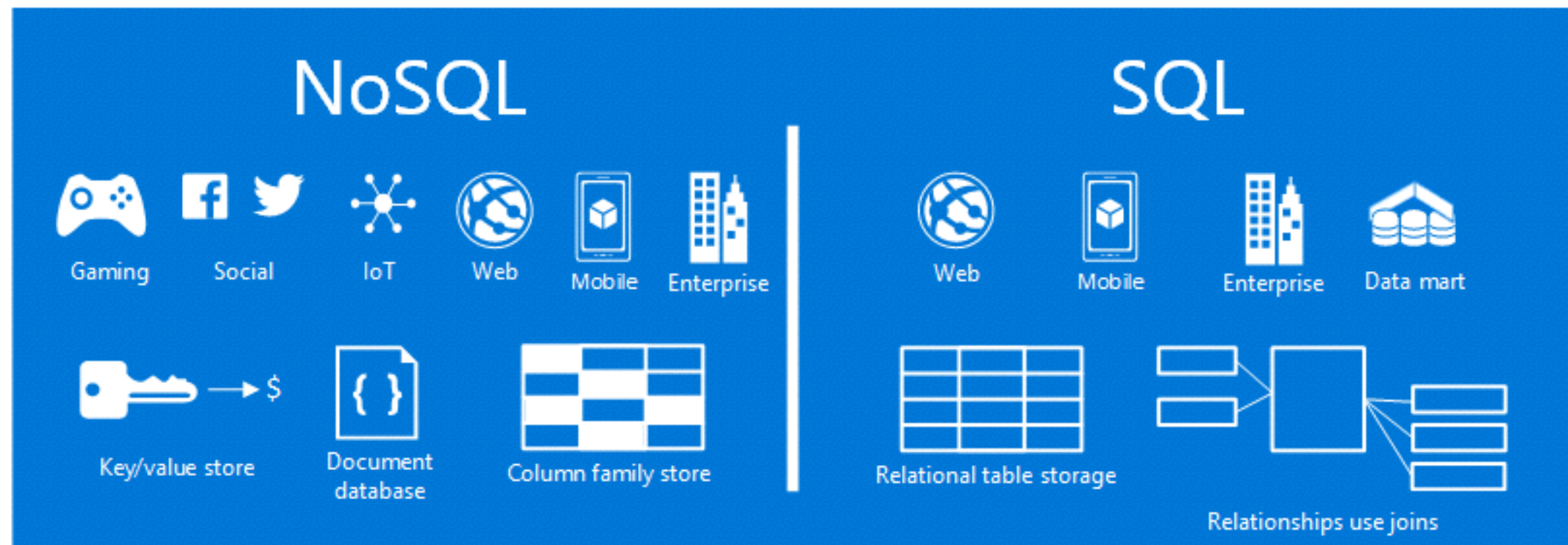
If it doesn't work, you're just not using enough.

New SQL Superpowers

- Array, map & JSON columns in PostgreSQL
- Additional analytics: trigonometric and logarithm functions
- Multi-dimensional arrays (SQL/MDA)
- Row pattern recognition
- Polymorphic table functions



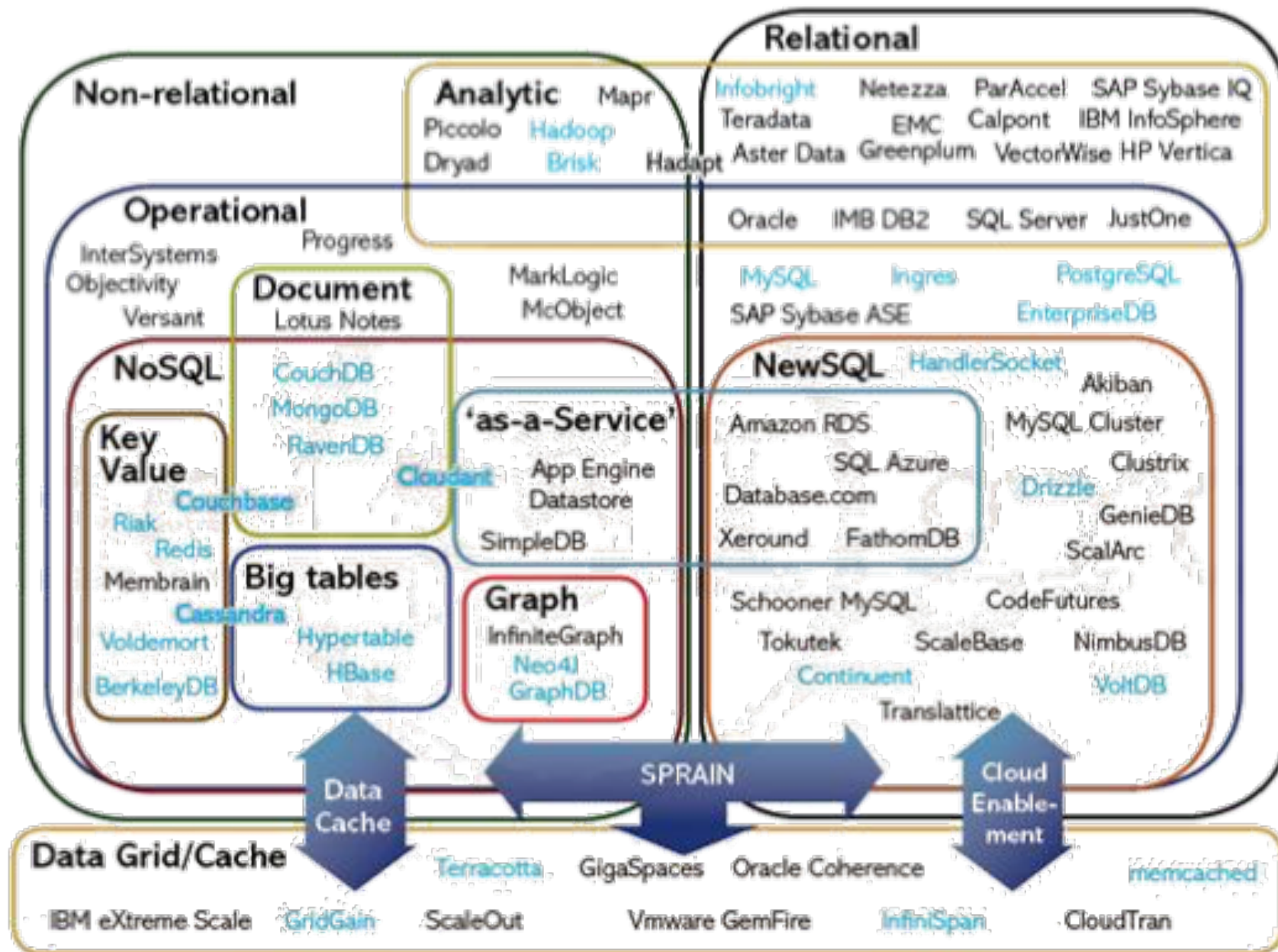
(No)SQL Final Thoughts



Misconceptions

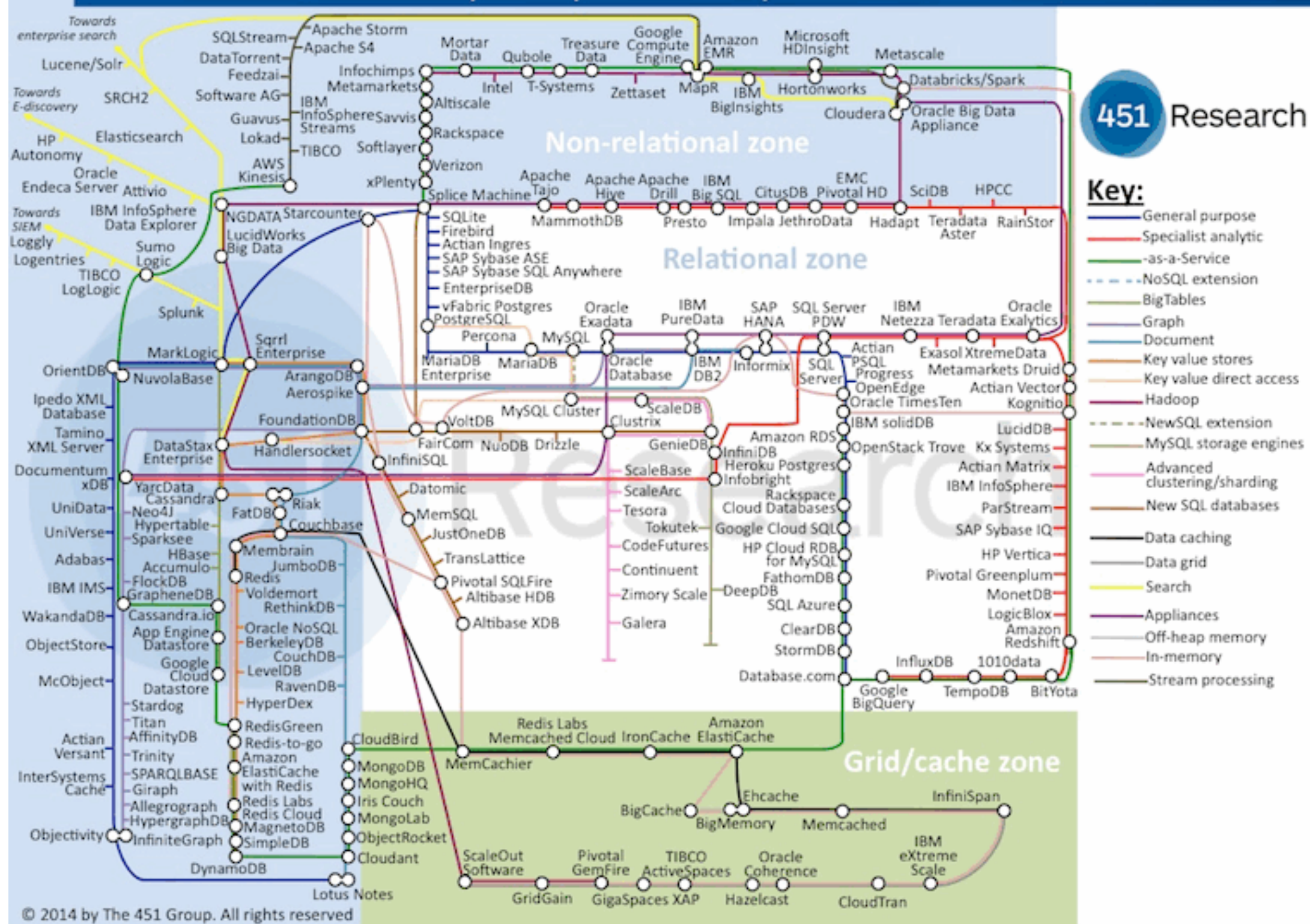
- **NoSQL is new**
 - dbm (1979), Internet's Domain Name System (1982), Berkeley DB (1991). The name "NoSQL" dates from 1998.
- **NoSQL is schema-free, so it eases migrations**
 - No standards, vendor lock-in (DBMS migration). Data structure still evolves (DB migration).
- **No joins & no query language = simpler & more performance**
 - Less flexibility & power at the database level = more application burden, more data transport.
- **NoSQL is fast**
 - Yes but generally at the price of weak consistency. Also, setting up horizontal scalability is a considerable overhead.
- **Relational databases are dead**
 - NoSQL databases mark the end of the era of relational databases, but they will not become the new dominators. Relational will still be popular, and used in many situations, but it will no longer be the automatic choice.

NoSQL is a lot of things



And more....

Data Platforms Landscape Map – February 2014





GIS & Databases

- GIS = database + visual interface
 - "Without a way to store, retrieve and analyze data, you have nothing more than a static map."
- GIS today deals with large and complex data (a.k.a. Big Data), and needs specialized data management

GIS & NoSQL

A partial survey of NoSQL stores with geo support

Name	Index strategy	Data types	Query types
Amazon DynamoDB	geohash	point	bbox, radius
GeoCouch	R-tree	point, line, poly	bbox, radius
IBM Cloudant (CouchDB)	R*-tree	GeoJSON types	bbox, radius, arbitrary shape
Lucene/Solr	geohash	point	bbox, radius
Orchestrate.io	geohash	point	bbox, radius
Microsoft DocumentDB	-	-	-
MongoDB	geohash/quadtrees	GeoJSON types	bbox, radius, arbitrary shape



mongoDB®

MongoDB: JSON documents

- It's a **document-oriented database**
- It stores **JSON** data (actually BSON)
- JSON **documents** (analogous to *rows*) are stored in **collections** (analogous to *tables*).



MongoDB: schema-less

- It's a **schema-less** database
- You can insert anything into a collection, even if the objects have a completely different structure (not necessarily a good idea)
- The database is created on-the-fly when you connect to it (if doesn't yet exist); commands are queued until the database is ready
- Collections (like **things**) are also created on-the-fly when you first access them

```
db.things.insert({
  'firstName': 'John',
  'lastName': 'Doe',
  'age': 30
})

db.things.insert({
  'wheels': 4,
  'speed': '200mps'
})

db.things.insert({
  'event': 'registration',
  'email': 'john.smith@example.com'
})
```

Basically Available (**BASE**)

MongoDB: find

- `db.collection.find(query)`
- MongoDB's **find** is like SQL's **SELECT**
- Queries are also JSON objects

```
// Find all documents
db.things.find()
// Equality: find all documents with an _id equal to 5
db.things.find({ _id: 5 })
// Equality: find all documents where qty.ordered is equal to 5
db.items.find({ qty: { ordered: 12 } })
// Operators: find all documents where age is greater than 25
db.people.find({ age: { $gt: 25 } })
// Ranges: find all documents where age is between 25 and 30
db.people.find({ age: { $gt: 25, $lt: 30 } })
```

<https://docs.mongodb.com/v3.2/reference/method/db.collection.find/>

MongoDB: Geospatial

Query Selectors

Name	Description
<code>\$geoWithin</code>	Selects geometries within a bounding GeoJSON geometry . The 2dsphere and 2d indexes support <code>\$geoWithin</code> .
<code>\$geoIntersects</code>	Selects geometries that intersect with a GeoJSON geometry . The 2dsphere index supports <code>\$geoIntersects</code> .
<code>\$near</code>	Returns geospatial objects in proximity to a point. Requires a geospatial index. The 2dsphere and 2d indexes support <code>\$near</code> .
<code>\$nearSphere</code>	Returns geospatial objects in proximity to a point on a sphere. Requires a geospatial index. The 2dsphere and 2d indexes support <code>\$nearSphere</code> .

<https://docs.mongodb.com/v3.2/reference/operator/query-geospatial/>

MongoDB: GeoJSON

GeoJSON is a format for encoding a variety of **geographic data structures**.

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [125.6, 10.1]
  },
  "properties": {
    "name": "Dinagat Islands"
  }
}
```

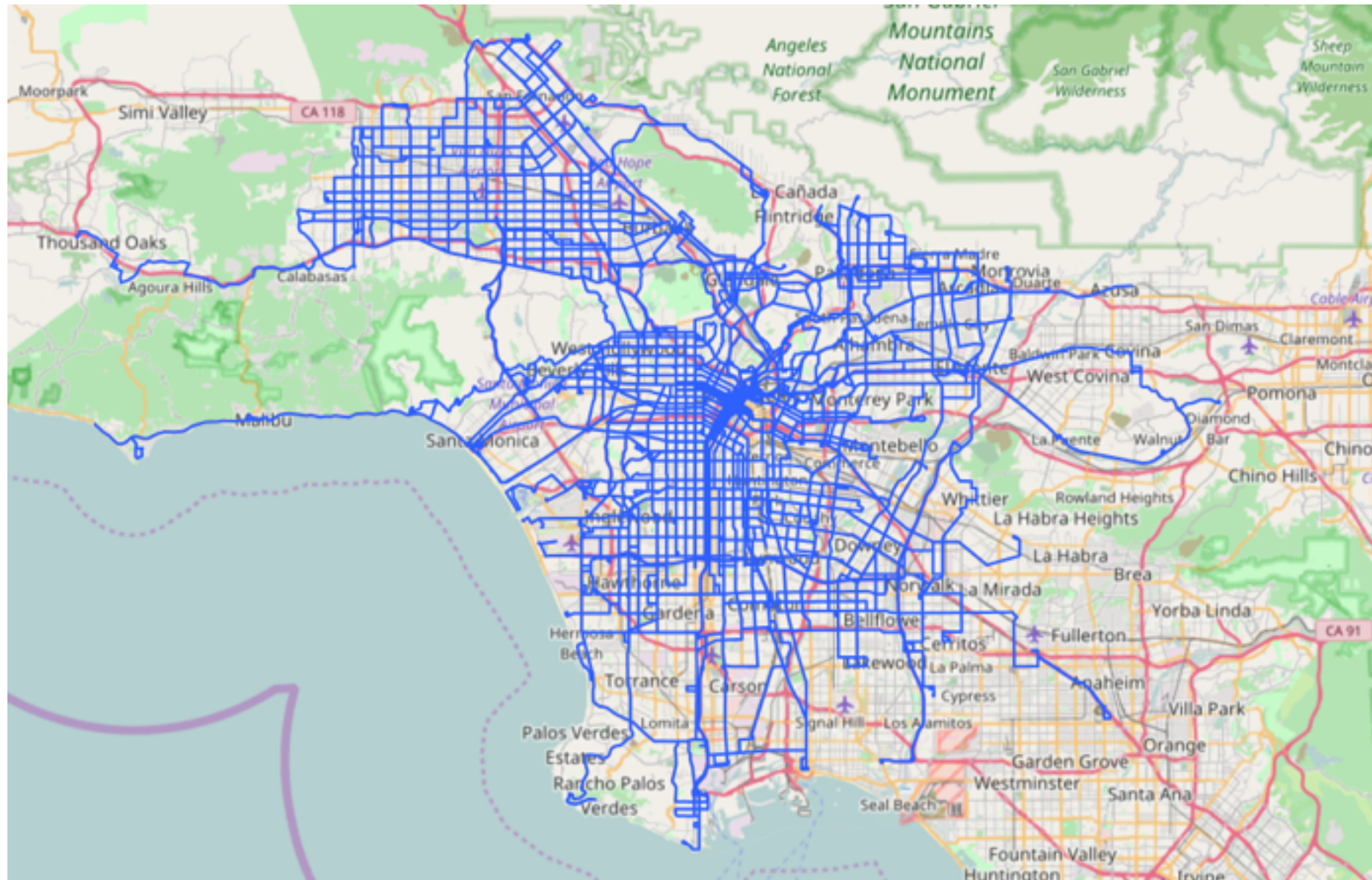
Available types: Point, LineString, Polygon, MultiPoint, MultiLineString, and MultiPolygon

<http://geojson.org>

<http://geojson.org/geojson-spec.html>



Data set



Los Angeles County Metropolitan Transportation Authority

<https://github.com/jdorfman/awesome-json-datasets>

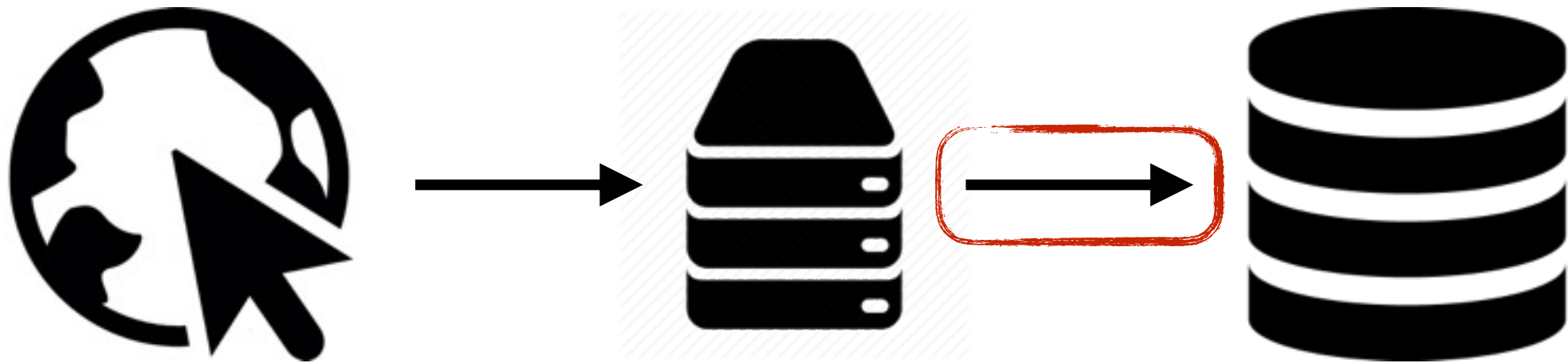
Data set: format

The database contains one collection: **features**.
Each document in the collection is a **GeoJSON feature**:

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [125.6, 10.1]
  },
  "properties": {
    "name": "Dinagat Islands"
  }
}
```

- The operator area is represented by **one** feature with a **Polygon** geometry.
- Transportation routes are represented by **multiple** features with a **MultiLineString** geometry.
- Train/bus stops are represented by **multiple** features with a **Point** geometry.

Exercise Web App



Exercise Web App



<http://stark-mountain-72407.herokuapp.com/>



<https://github.com/MediaComem/comem-geoinf-2016-mongodb-gis>