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

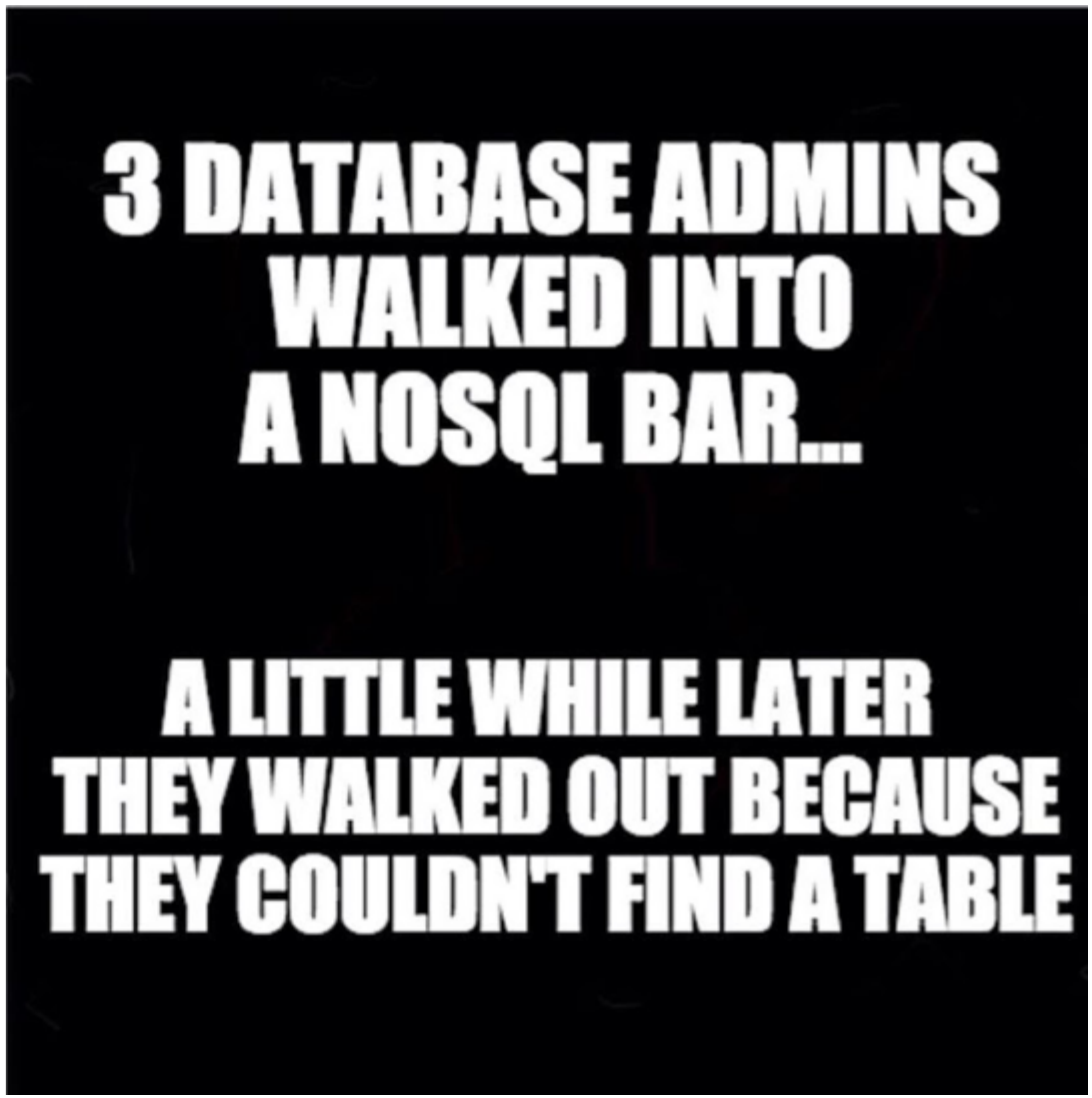
By Wolfgang Reinhardt, Eszter Gálicz & Stephan Schmid nosql

“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?



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:

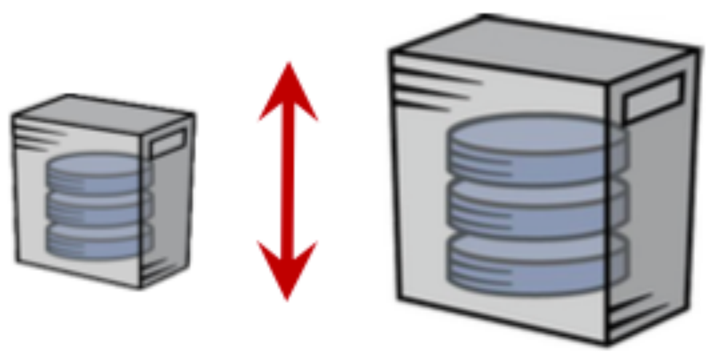
- ACID
- Strong promises,
  desirable for many domains (e.g. banks)
- Computationally expensive,
  in particular when distributed

Characteristics:

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

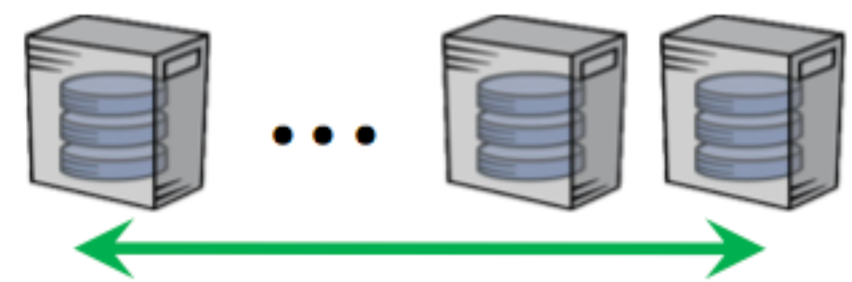
Scalability

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

riak

CouchDB relax

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

MemCacheDB Riak
MemCache Cassandra Redis
Voldemort SimpleDB

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)

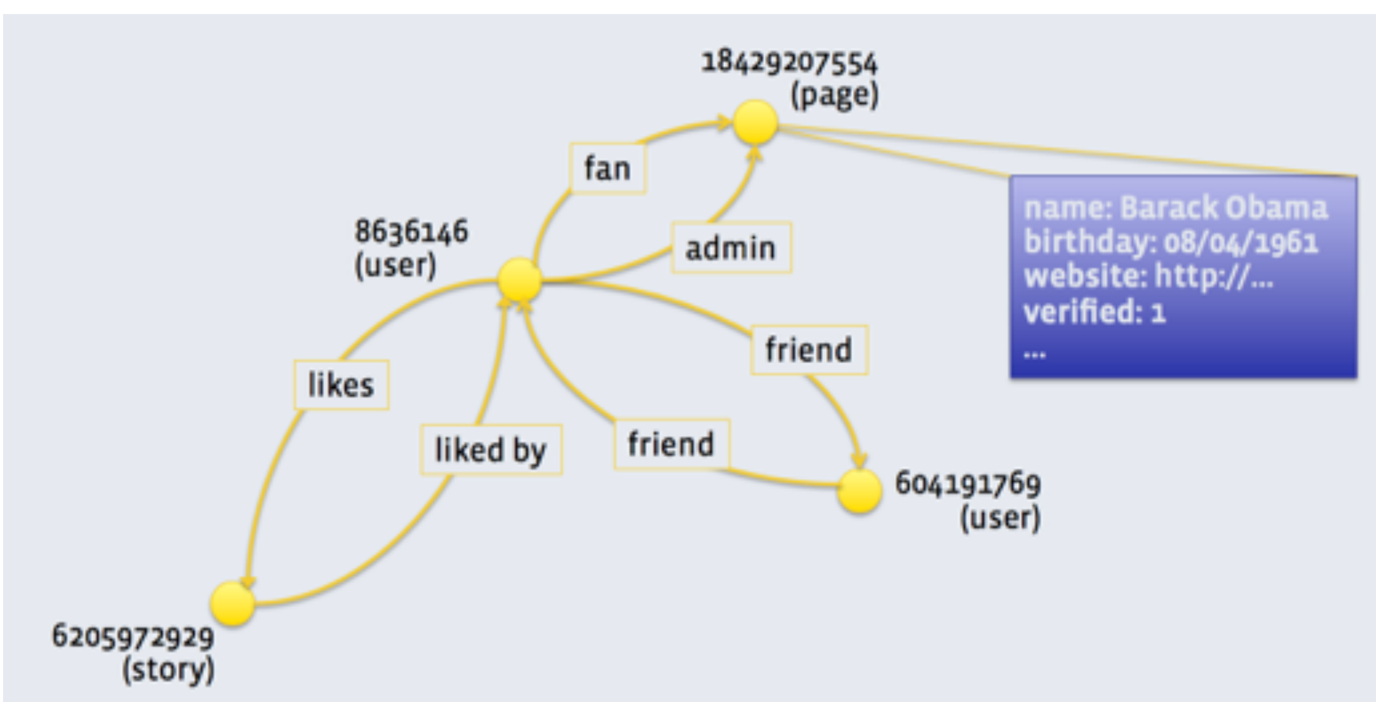
MongoDB

ElasticSearch

CouchDB

Graph Databases

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



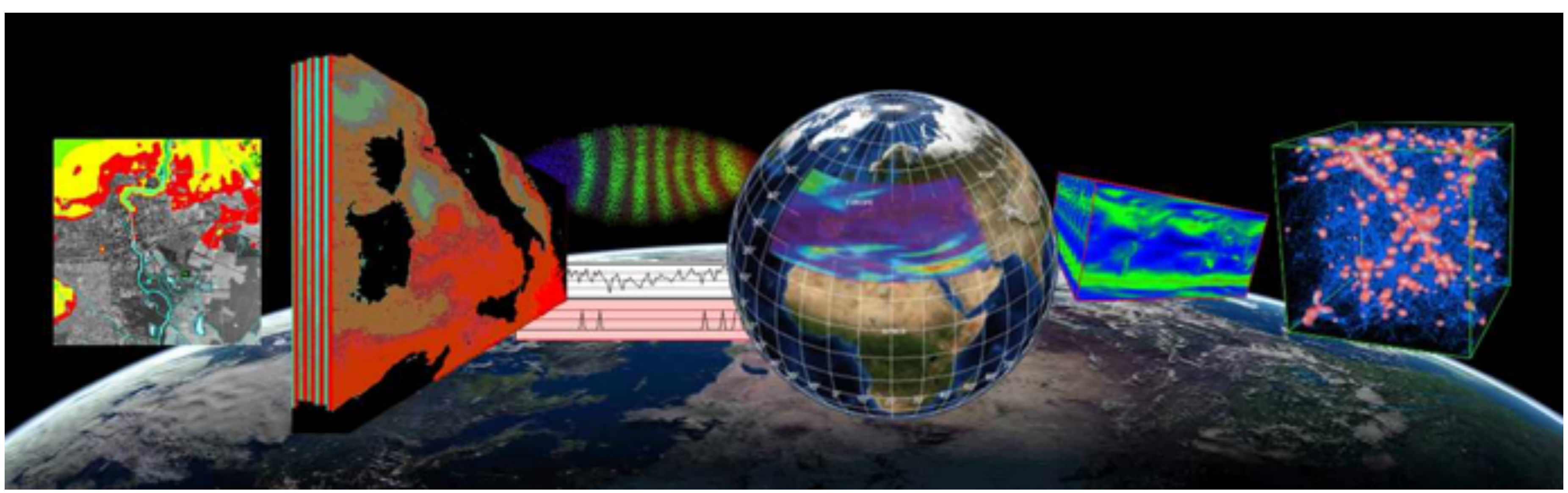
RDF/SPARQL

Facebook TAO | Neo4j

Array Databases

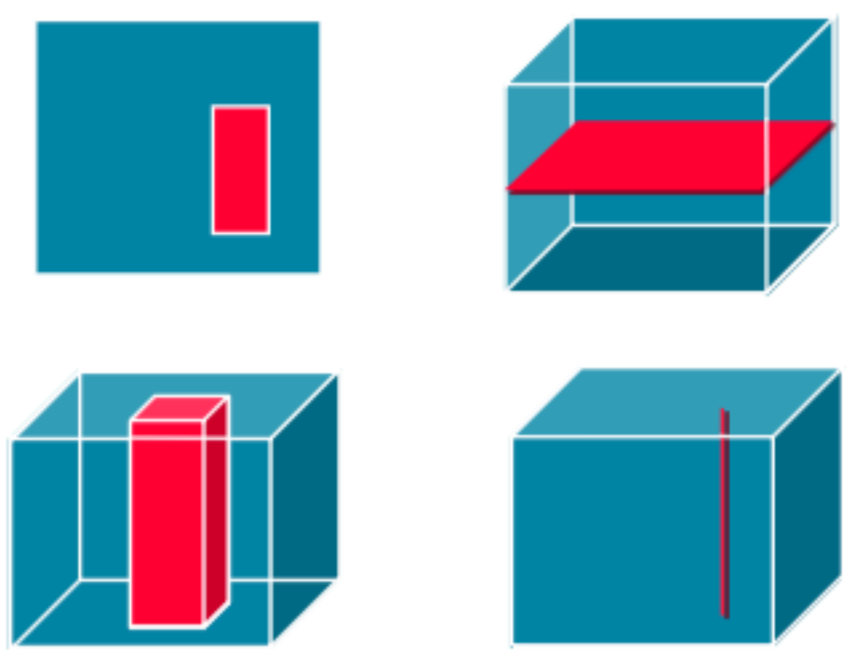
- Conceptual model: **n-D arrays**
- Application: satellite imagery & signal processing

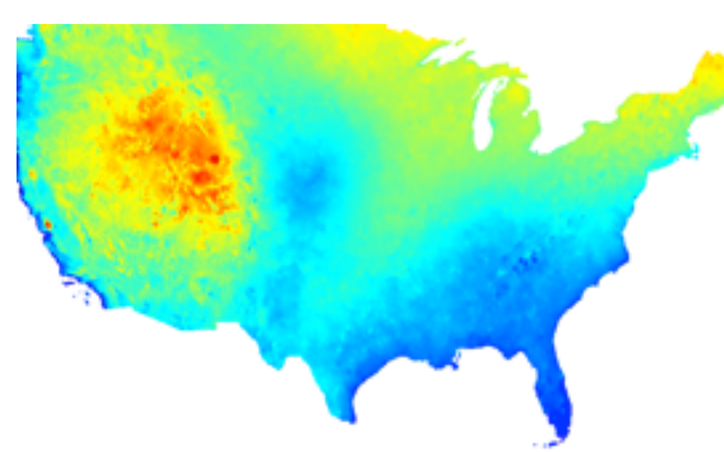
rasdaman DaggerDB



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





rasdaman

GDAL QGIS

OpenLayers Pyxis

ArcGIS OPeNDAP

MapServer GeoServer

NewSQL

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)

Sysbase IQ
KDB
MonetDB
HBase
Vertica
Druid

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	
Value	Row IDs
Beer	1, 2
Vodka	3, 6, 7
Whiskey	4, 5

CUSTOMER	
Value	Row IDs
Thomas	1, 2, 3
Christian	4, 5
Alexei	6, 7

DATE	
Value	Row IDs
2011-11-25	1, 2, 3, 4
2011-11-26	5, 6, 7

SALE	
Value	Row IDs
2 GBP	1, 2
10 GBP	3, 6, 7
5 GBP	4, 5

Main Memory DB

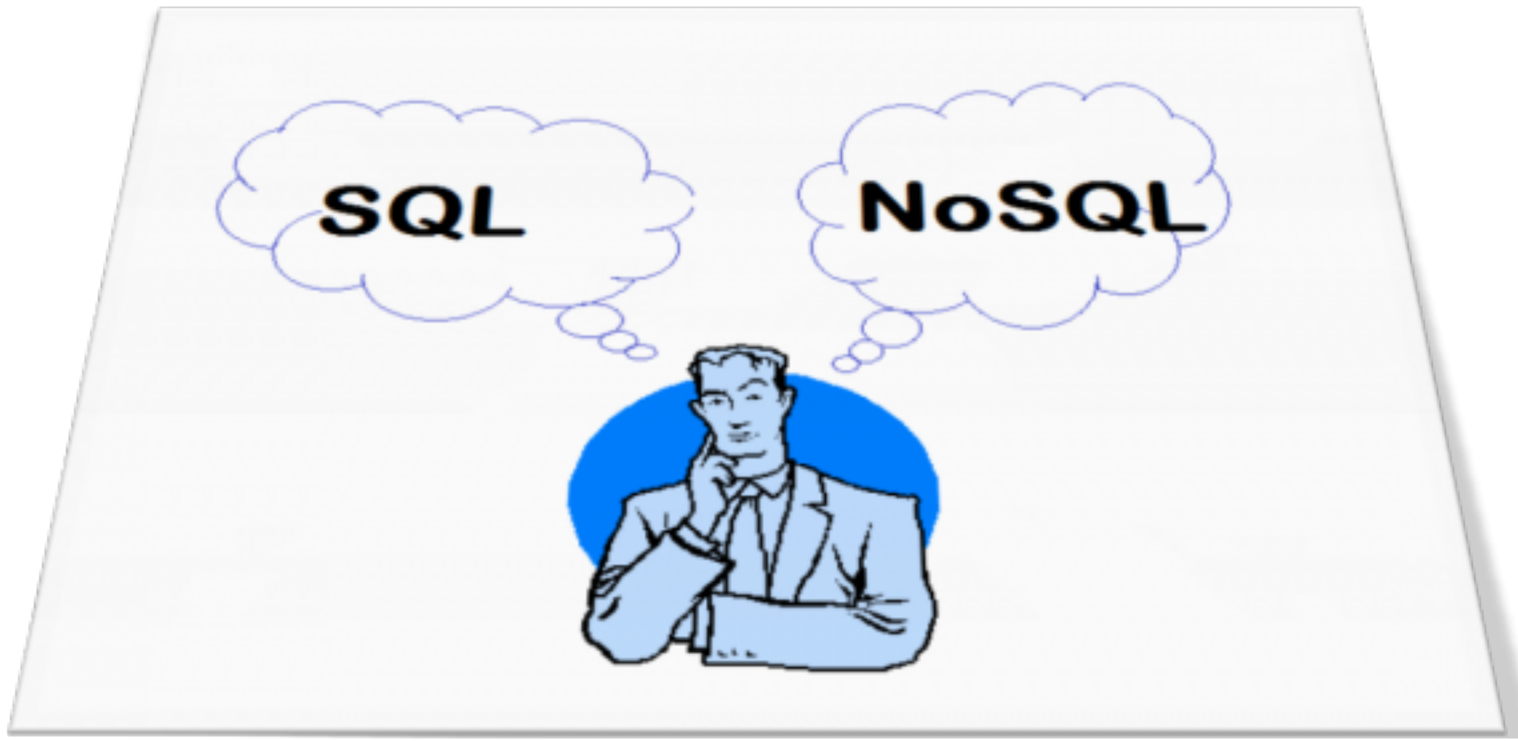
- RAM is faster than disk
  - Load the DB into RAM
  - Doable now with lower hardware price
  - Oracle ships 1PB RAM servers
- **Vertical scaling**, not horizontal

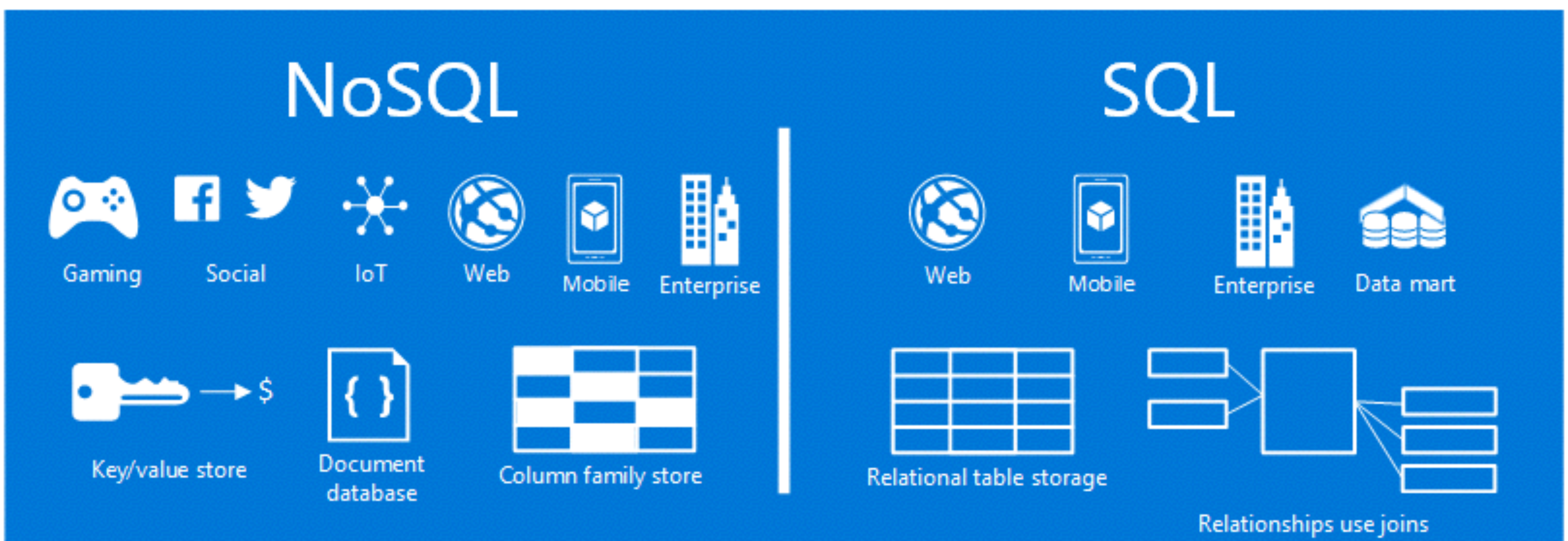
Oracle TimesTen | HyPer | Microsoft Hekaton | VoltDB | Calvin | IBM solidDB

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

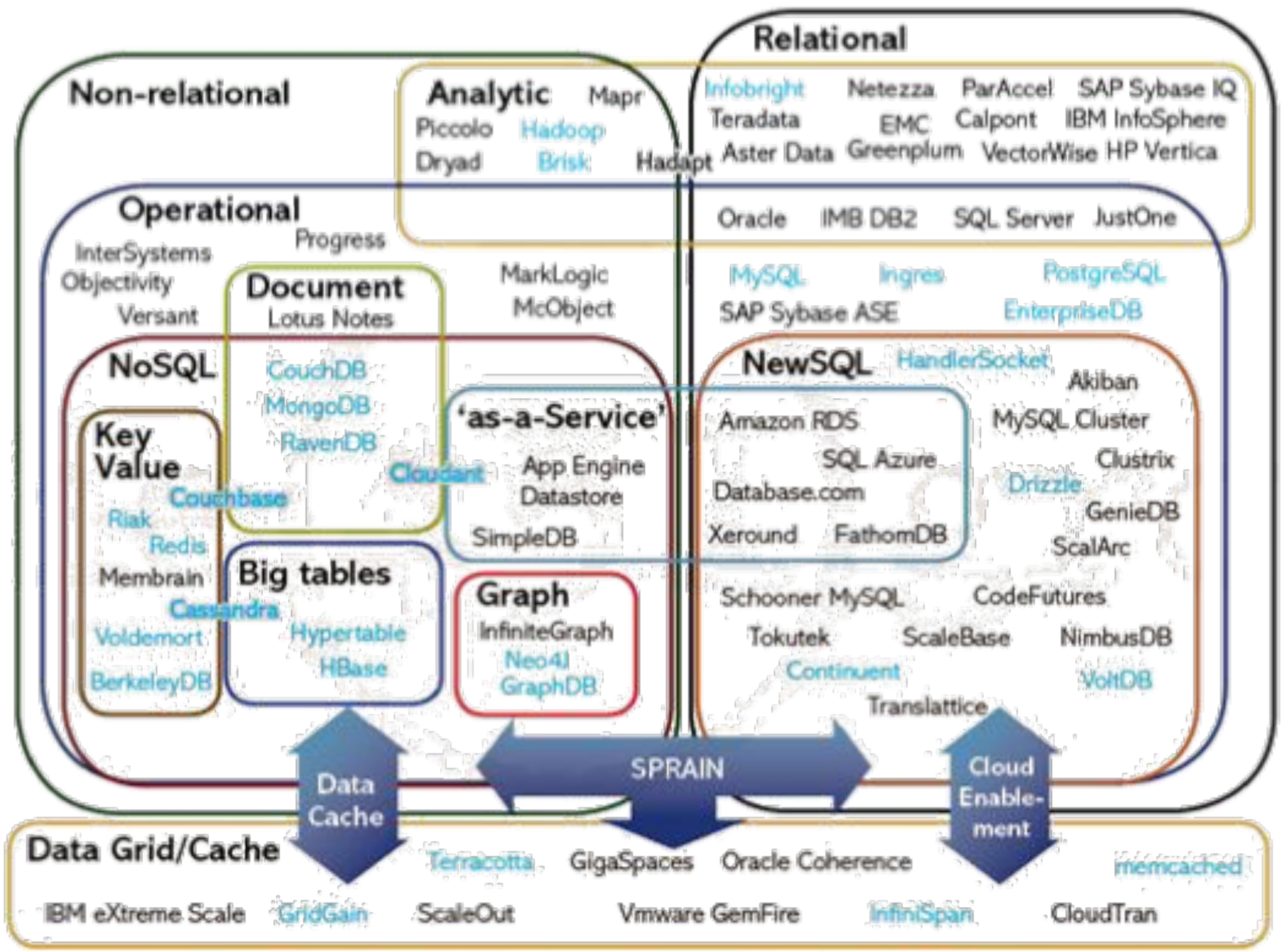




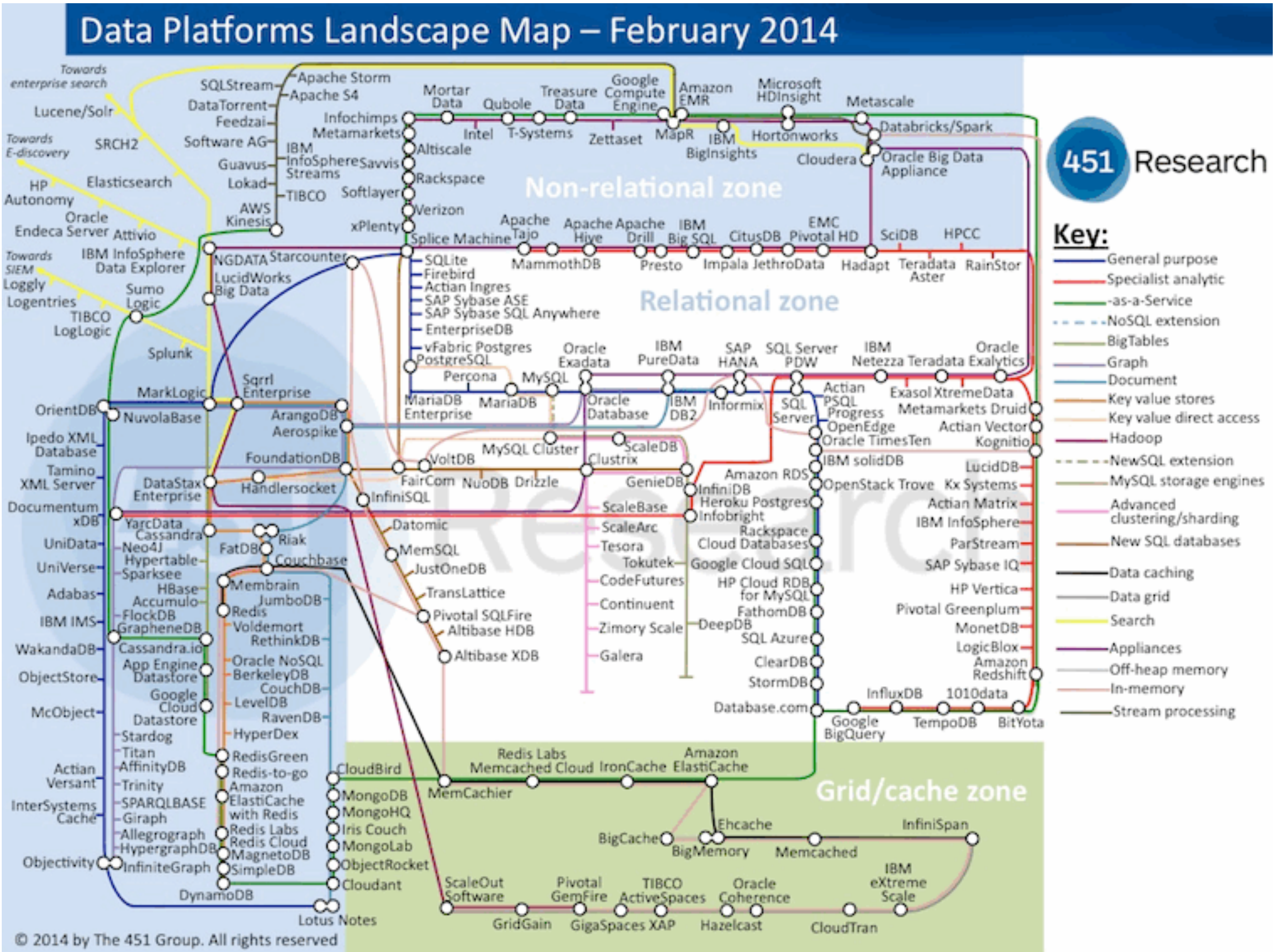
Misconceptions

- **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
451 Research
Non-relational zone
Relational zone
Grid/cache zone
Towards enterprise search
Lucene/Solr
Towards E-discovery
SRCH2
Elasticsearch
HP Autonomy
Oracle Endeca Server
Attivio
Towards SIEM
IBM InfoSphere Data Explorer
Loggly
Logentries
Sumo Logic
TIBCO LogLogic
Splunk
SQLStream
DataTorrent
Feedzai
Software AG
Guavus
Lokad
AWS Kinesis
Apache Storm
Apache S4
IBM InfoSphere Streams
TIBCO
Infochimps
Metamarkets
Savvis
Softlayer
xPlenty
Altiscale
Rackspace
Verizon
Splice Machine
Mortar Data
Qubole
Intel
T-Systems
Treasure Data
Google Compute Engine
Zettaset
Amazon EMR
MapR
IBM BigInsights
Microsoft HDInsight
Hortonworks
Cloudera
Metascale
Databricks/Spark
Oracle Big Data Appliance
Apache Tajo
Apache Hive
Apache Drill
IBM Big SQL
CitusDB
EMC Pivotal HD
SciDB
HPCC
NGDATA
Starcounter
LucidWorks Big Data
MammothDB
Presto
Impala
JethroData
Hadapt
Teradata Aster
RainStor
SQLite
Firebird
Actian Ingres
SAP Sybase ASE
SAP Sybase SQL Anywhere
EnterpriseDB
vFabric Postgres
PostgreSQL
Percona
MySQL
MariaDB Enterprise
MariaDB
Oracle Exadata
Oracle Database
IBM PureData
IBM DB2
Informix
SAP HANA
SQL Server PDW
SQL Server
IBM Netezza
Teradata
Oracle Exalytics
Actian PSQL
Progress OpenEdge
Oracle TimesTen
Exasol
XtremeData
Metamarkets Druid
Actian Vector
Kognitio
MarkLogic
Sqrrl Enterprise
OrientDB
NuvolaBase
ArangoDB
Aerospike
FoundationDB
Ipedo XML Database
Tamino XML Server
DataStax Enterprise
Handlersocket
Documentum xDB
MySQL Cluster
ScaleDB
Clustrix
VoltDB
FairCom
NuoDB
Drizzle
InfiniSQL
GenieDB
ScaleBase
ScaleArc
Tesora
Tokutek
CodeFutures
Continuent
Zimory Scale
Galera
Amazon RDS
InfiniDB
Heroku Postgres
Infobright
Rackspace Cloud Databases
Google Cloud SQL
HP Cloud RDB for MySQL
FathomDB
DeepDB
SQL Azure
ClearDB
StormDB
Database.com
IBM solidDB
OpenStack Trove
LucidDB
Kx Systems
Actian Matrix
IBM InfoSphere
ParStream
SAP Sybase IQ
HP Vertica
Pivotal Greenplum
MonetDB
LogicBlox
Amazon Redshift
InfluxDB
1010data
Google BigQuery
TempoDB
BitYota
UniData
UniVerse
Adabas
IBM IMS
WakandaDB
ObjectStore
McObject
Actian Versant
InterSystems Cache
Objectivity
YarcData
Cassandra
Neo4J
Hypertable
Sparksee
HBase
Accumulo
FlockDB
GrapheneDB
Cassandra.io
App Engine Datastore
Google Cloud Datastore
Stardog
Titan
AffinityDB
Trinity
SPARQLBASE
Giraph
Allegrograph
HypergraphDB
InfiniteGraph
DynamoDB
Lotus Notes
Riak
FatDB
Couchbase
Membrain
JumboDB
Redis
Voldemort
RethinkDB
Oracle NoSQL
BerkeleyDB
CouchDB
LevelDB
RavenDB
HyperDex
RedisGreen
Redis-to-go
Amazon ElastiCache with Redis
Redis Labs Redis Cloud
MagnetoDB
SimpleDB
CloudBird
MongoDB
MongoHQ
Iris Couch
MongoLab
ObjectRocket
Cloudant
Datomic
MemSQL
JustOneDB
TransLattice
Pivotal SQLFire
Altibase HDB
Altibase XDB
MemCachier
Redis Labs Memcached Cloud
IronCache
Amazon ElastiCache
BigCache
Ehcache
BigMemory
Memcached
InfiniSpan
ScaleOut Software
GridGain
Pivotal GemFire
GigaSpaces XAP
TIBCO ActiveSpaces
Hazelcast
Oracle Coherence
CloudTran
IBM eXtreme Scale
Key:
General purpose
Specialist analytic
-as-a-Service
NoSQL extension
BigTables
Graph
Document
Key value stores
Key value direct access
Hadoop
NewSQL extension
MySQL storage engines
Advanced clustering/sharding
New SQL databases
Data caching
Data grid
Search
Appliances
Off-heap memory
In-memory
Stream processing
© 2014 by The 451 Group. All rights reserved

GIS

Designing our future

esri

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/quadtree	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 (**BA**SE)

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

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"
  }
}
```

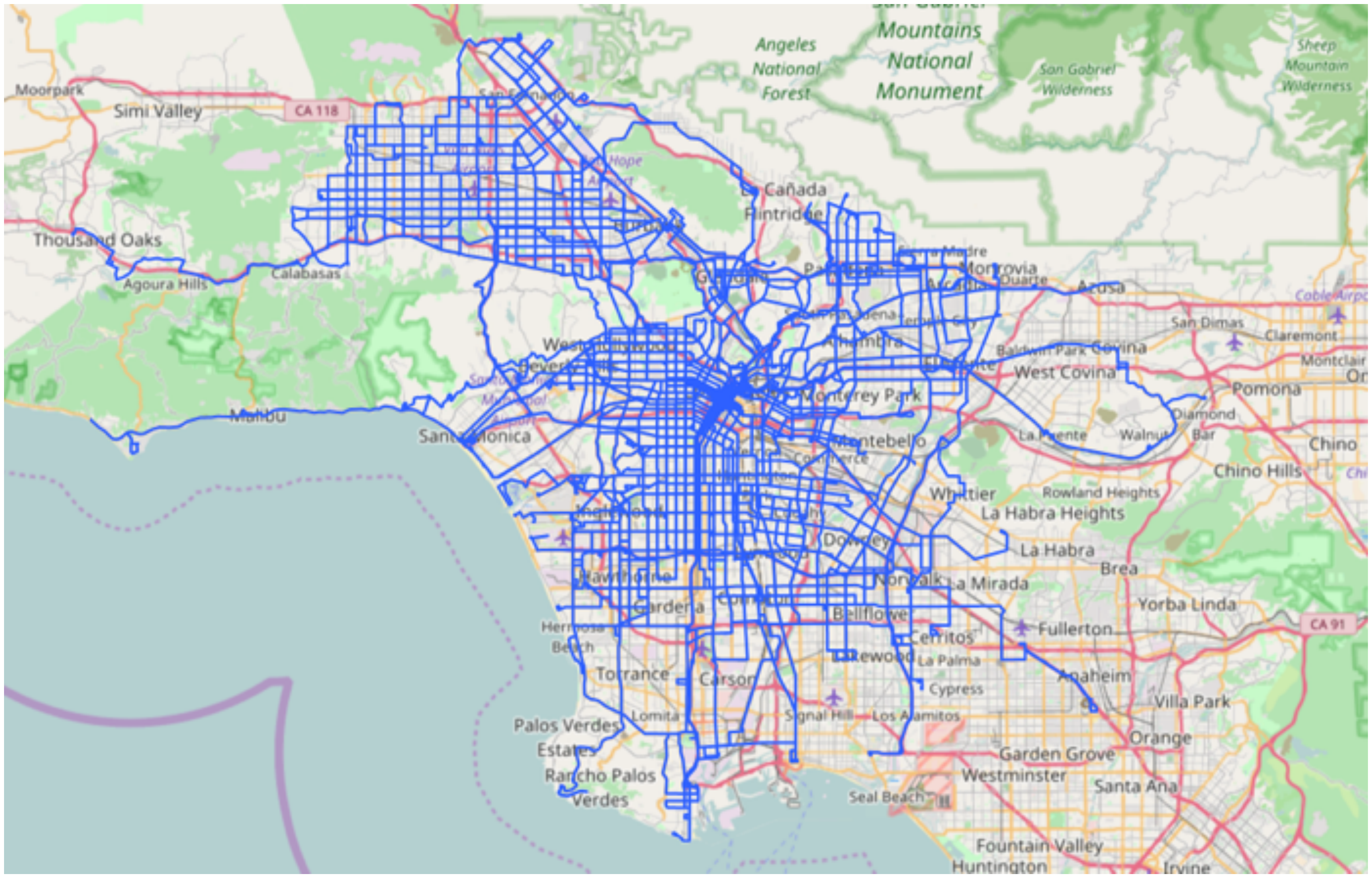
Available types: Point, LineString, Polygon, MultiPoint, MultiLineString, and MultiPolygon

http://geojson.org

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


EXERCISE
Some motivation required.

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

AngularJS by Google

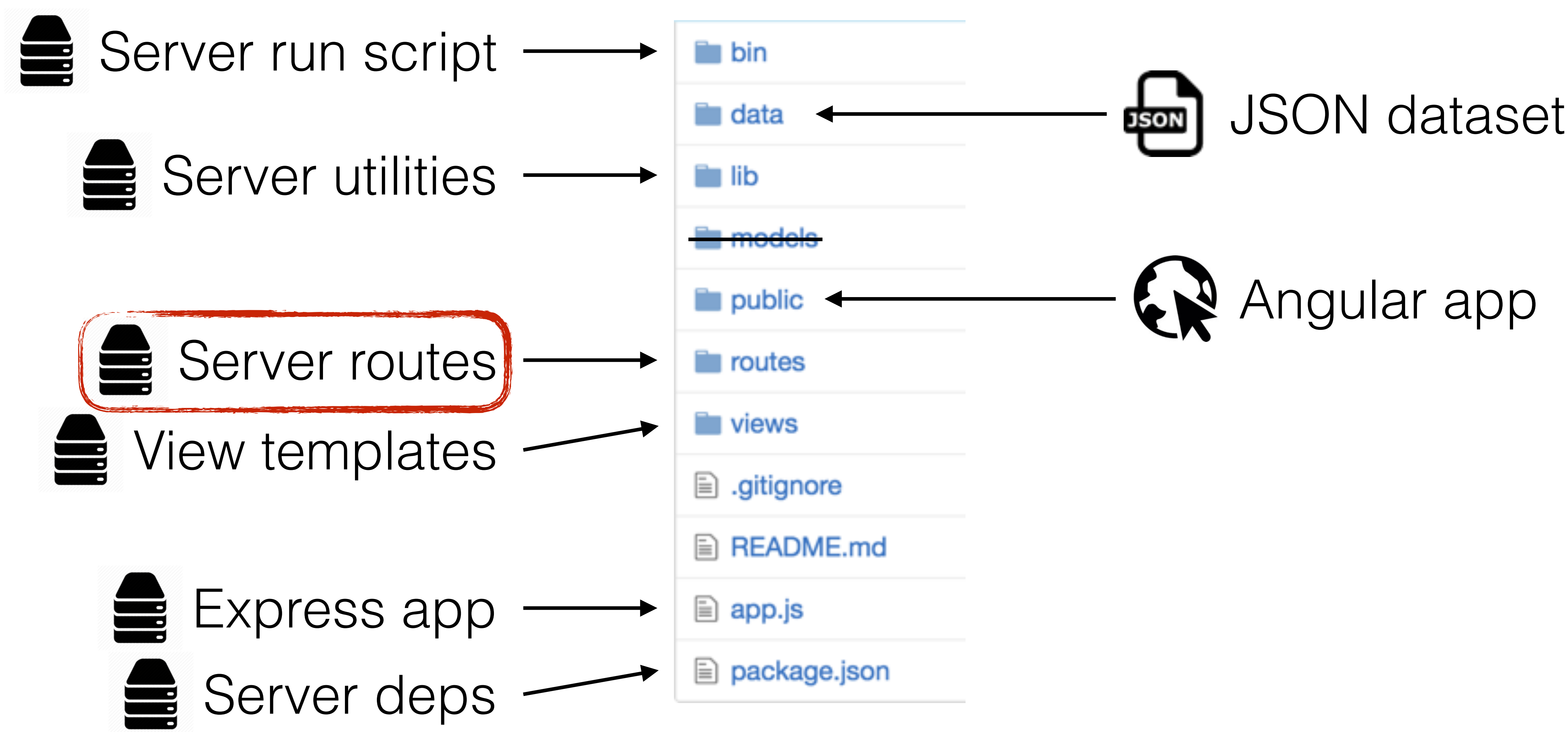
express + node JS

mongoDB

JavaScript

JSON

Exercise Web App



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

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