

# NoSQL & NewSQL

~~SQL~~

# Overview

- NoSQL
- Transactions
- NewSQL

# NoSQL

# Performance Comparison

- On > 50 GB data:
- MySQL
  - Writes 300 ms avg
  - Reads 350 ms avg
- Cassandra
  - Writes 0.12 ms avg
  - Reads 15 ms avg

# We Don't Want No SQL !

- NoSQL movement: SQL considered slow → only access by id („lookup“)
  - Deliberately abandoning relational world: „too complex“, „not scalable“
  - No clear definition, wide range of systems
  - Values considered black boxes (documents, images, ...)
  - simple operations (ex: key/value storage), horizontal scalability for those
  - ACID → CAP, „eventual consistency“

## ■ Systems



- Open source
  - Proprietary: Amazon, Oracle, Google , Oracle NoSQL
- See also: <http://glennas.wordpress.com/2011/03/11/introduction-to-nosql-john-nunemaker-presentation-from-june-2010/>

# NoSQL

- Previous „young radicals“ approaches subsumed under „NoSQL“
- = we want „no SQL“
- Well..., „not only SQL“
  - After all, a QL is quite handy
  - So, QLs coming into play again (and 2-phase commits = ACID!)
- Ex: MongoDB: „tuple“ = JSON structure

```
db.inventory.find(  
  { type: 'food',  
    $or: [ { qty: { $gt: 100 } }, { price: { $lt: 9.95 } } ]  
  } )
```

# Another View: Structural Variety in Big Data

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

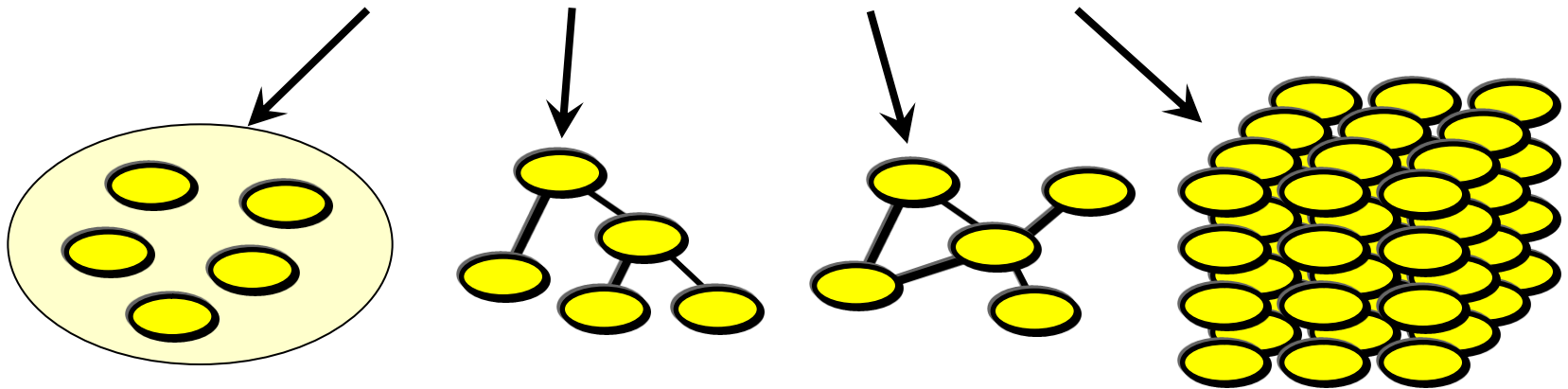
# Another View: Structural Variety in Big Data

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

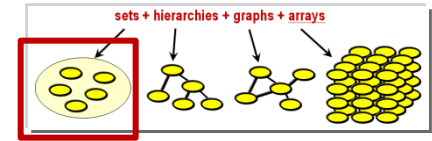


# Structural Variety in [Big] Data

**sets + hierarchies + graphs + arrays**



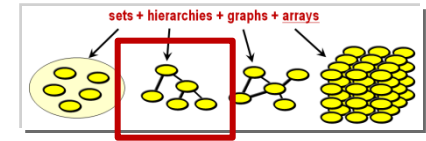
# Ex 1: Key/Value Store



- Conceptual model: key/value store = **set of key+value**
  - Operations: *Put(key,value)*, *value = Get(key)*
  - → large, distributed **hash table**
- Needed for:
  - twitter.com: tweet id -> information about tweet
  - kayak.com: Flight number -> information about flight, e.g., availability
  - amazon.com: item number -> information about it
- Ex: Cassandra (Facebook; open source)
  - Myriads of users, like:



# Ex 2: Document Stores

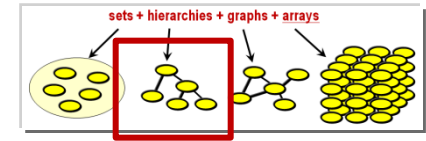


- Like key/value, but value is a **complex document**
  - Data model: set of nested records
- Added: **Search** functionality within document
  - Full-text search: Lucene/Solr, ElasticSearch, ...
- Application: content-oriented applications
  - Facebook, Amazon, ...
- Ex: MongoDB, CouchDB

```
db.inventory.find( { $or: [ { status: "A" }, { qty: { $lt: 30 } } ] } )
```

```
SELECT * FROM inventory WHERE status = "A" AND qty < 30
```

# Ex 3: Hierarchical Data



- Disclaimer: long before NoSQL!

```
doc("books.xml") /bookstore/book/title
```

```
doc("books.xml") /bookstore/book[price<30]
```

- Later more, time permitting!

```
<?xml version="1.0" encoding="UTF-8"?>

<bookstore>

<book category="COOKING">
  <title lang="en">Everyday Italian</title>
  <author>Giada De Laurentiis</author>
  <year>2005</year>
  <price>30.00</price>
</book>

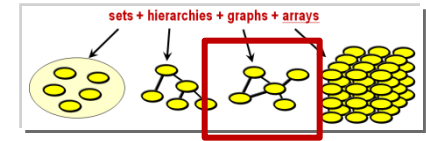
<book category="CHILDREN">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>

<book category="WEB">
  <title lang="en">XQuery Kick Start</title>
  <author>James McGovern</author>
  <author>Per Bothner</author>
  <author>Kurt Cagle</author>
  <author>James Linn</author>
  <author>Vaidyanathan Nagarajan</author>
  <year>2003</year>
  <price>49.99</price>
</book>

<book category="WEB">
  <title lang="en">Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
</book>

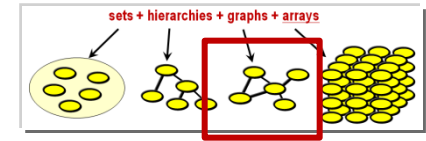
</bookstore>
```

# Ex 4: Graph Store

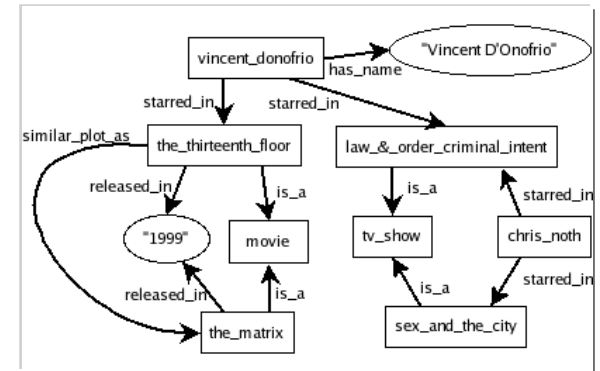


- Conceptual model: Labeled, directed, attributed graph
- Why not relational DB? can model graphs!
  - but “endpoints of an edge” already requires join
  - No support for global ops like transitive hull
- Main cases:
  - Small, heterogeneous graphs
  - Large, homogeneous graphs

# Ex 4a: RDF & SPARQL



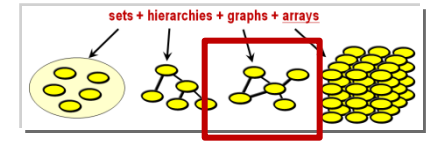
- Situation: **Small, heterogeneous** graphs
- Use cases: ontologies, knowledge graphs, Semantic Web
- Model:
  - Data model: graphs as triples
    - RDF (Resource Data Framework)
  - Query model: patterns on triples
    - SPARQL (see later, time permitting)



```

PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE
{
    ?x foaf:name ?name .
    ?x foaf:mbox ?mbox
}
  
```

# Ex 4b: Graph Databases

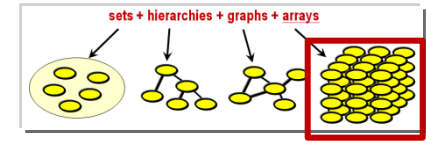


- Situation: **Large, homogeneous** graphs
- Use cases: Social Networks
- Common queries:
  - *My friends*
  - *who has no / many followers*
  - *closed communities*
  - *new agglomerations,*
  - *new themes, ...*
- Sample system: Neo4j with QL Cypher



```
MATCH (:Person {name: 'Jennifer'})-[:WORKS_FOR]->(company:Company)
RETURN company.name
```

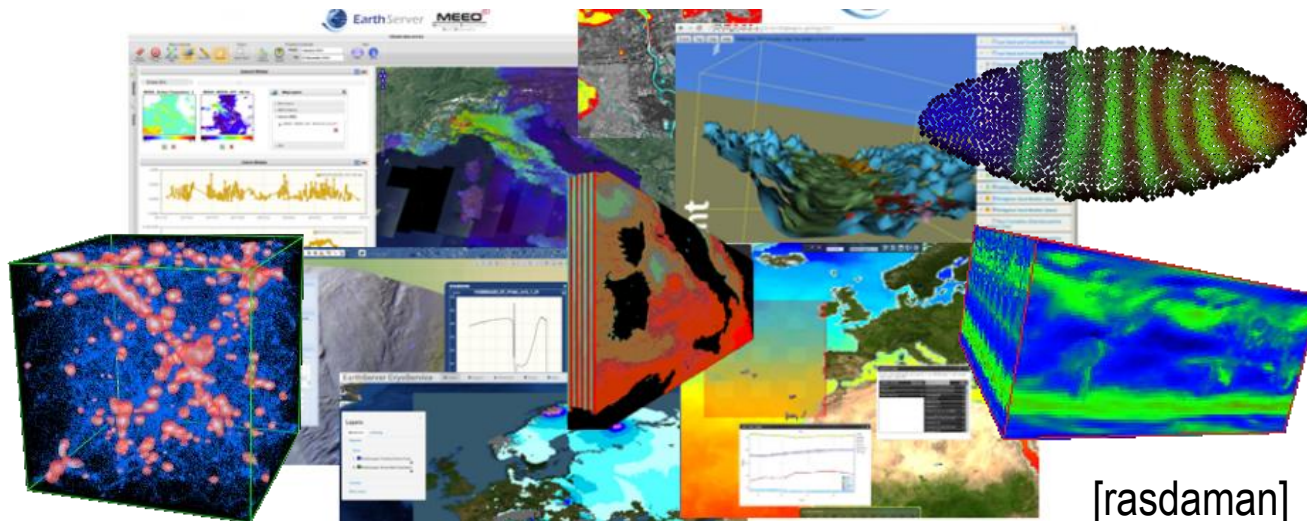
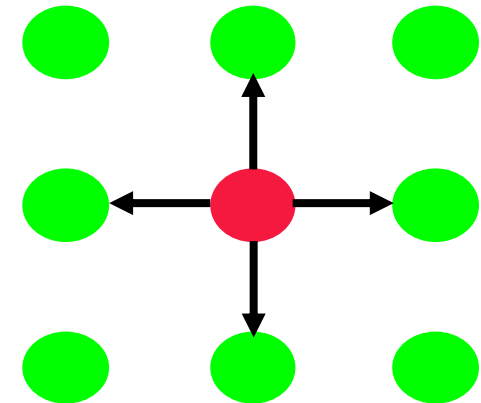
# Ex 5: Array Analytics



- **Array Analytics** :=  
*Efficient analysis on multi-dimensional arrays of a size several orders of magnitude above the evaluation engine's main memory*

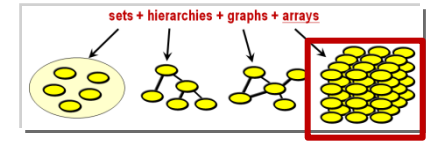
sensor, image [timeseries],  
simulation, statistics data

- Essential property:  $n$ -D Cartesian neighborhood





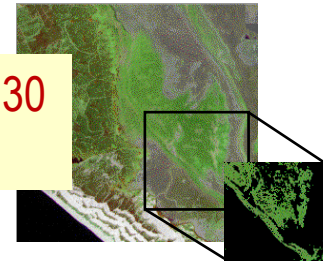
# Ex 5: Array Databases



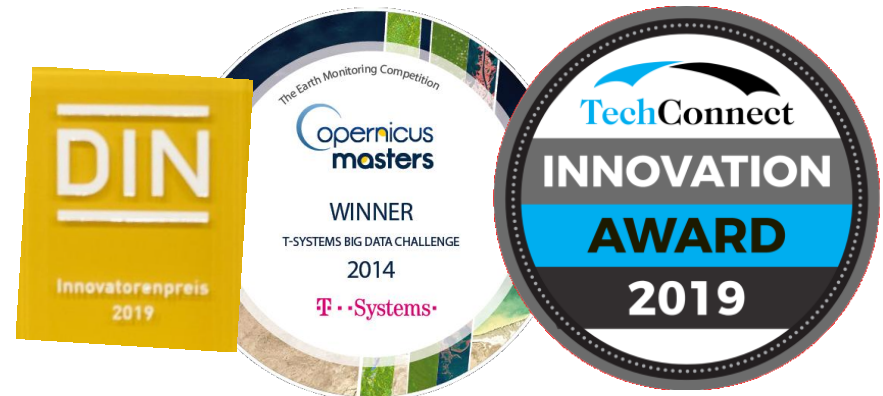
- Ex: rasdaman = **Array DBMS**

- Data model: n-D arrays as attributes
- Query model: Tensor Algebra
- Demo at <http://standards.rasdaman.org>

```
select img.raster[x0:x1,y0:y1] > 130
from LandsatArchive as img
```



- Multi-core, distributed, platform for EarthServer (<https://earthserver.xyz>)
- Relational? „Array DBMSs can be 200x RDBMS“ [Cudre-Maroux]



# Transactions

# No More ACID

- RDBMS provide ACID...locally
- Close to impossible to achieve in distributed situations
- Instead: **BASE**
  - Basically Available Soft-state Eventual Consistency
  - Prefers availability over consistency
- Ex: Cassandra

# Outlook: ACID vs BASE

- **BASE** = Basically Available Soft-state Eventual Consistency
  - availability over consistency, relaxing ACID
  - ACID model promotes **consistency** over **availability**,  
BASE promotes **availability** over **consistency**
- Comparison:
  - Traditional RDBMSs: Strong **consistency** over **availability** under a **partition**
  - Cassandra: Eventual (weak) **consistency**, **availability**, **partition-tolerance**
- **CAP Theorem** [proposed: Eric Brewer; proven: Gilbert & Lynch]:  
In a distributed system you can satisfy at most 2 out of the 3 guarantees
  - **Consistency**: all nodes have same data at any time
  - **Availability**: system allows operations all the time
  - **Partition-tolerance**: system continues to work in spite of network partitions

# Discussion: ACID vs BASE

- Justin Sheely: “eventual consistency in well-designed systems does not lead to inconsistency”
- Daniel Abadi: “If your database only guarantees eventual consistency, you have to make sure your application is well-designed to resolve all consistency conflicts. [...] Application code has to be smart enough to deal with any possible kind of conflict, and resolve them correctly”
  - Sometimes simple policies like “last update wins” sufficient
  - other apps far more complicated, can lead to errors and security flaws
  - Ex: ATM heist with 60s window
  - DB with stronger guarantees greatly simplifies application design

# CAP Theorem

- Proposed by Eric Brewer, UCB; subsequently proved by Gilbert & Lynch
- In a distributed system you can satisfy at most 2 out of the 3 guarantees
  - **Consistency**: all nodes have same data at any time
  - **Availability**: system allows operations all the time
  - **Partition-tolerance**: system continues to work in spite of network partitions
- Traditional RDBMSs
  - Strong consistency over availability under a partition
- Cassandra
  - Eventual (weak) consistency, Availability, Partition-tolerance

# NewSQL

# NewSQL: *The Empire Strikes Back*

- Michael Stonebraker: „*no one size fits all*“
- NoSQL: sacrificing functionality for performance – no QL, only key access
  - Single round trip fast, complex real-world problems slow
- Swinging back from NoSQL:  
declarative QLs considered good (again), but SQL often inadequate
- Definition 1: NewSQL = SQL with **enhanced performance** architectures
- Definition 2: NewSQL = SQL enhanced with, eg, **new data types**
  - Some call this NoSQL



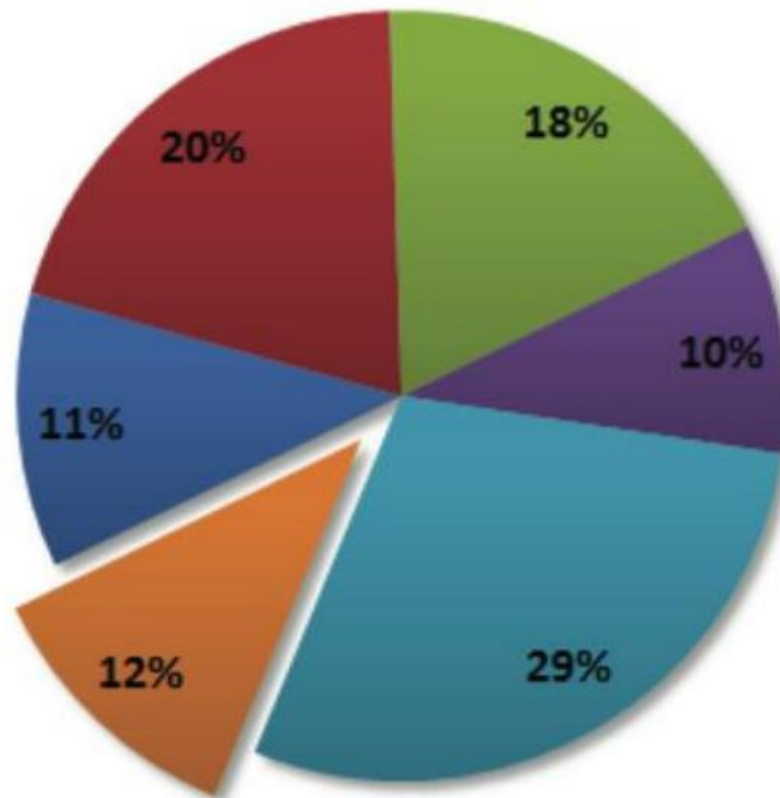
# What Makes an RDBMS Slow?

## General Purpose RDBMS Processing Profile

*OLTP Through the Looking Glass, and What We Found There*

Stavros Harizopoulos, Daniel Abadi, Samuel Madden, and Michael Stonebraker  
ACM SIGMOD 2008.

- Index Management
- Logging
- Locking
- Latching
- Buffer Management
- Useful Work

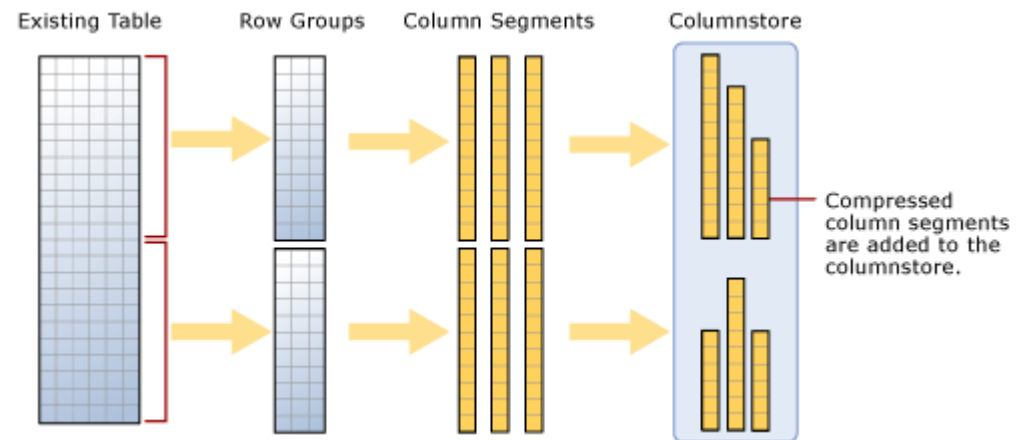


# Column-Store Databases

- Observation: fetching long tuples overhead when few attributes needed
- Brute-force decomposition: one value (plus key)

- Ex: Id+SNLRH → Id+S, Id+N, Id+L, Id+R, Id+H

- Column-oriented storage:  
each binary table separate file



- With clever architecture, reassembly of tuples pays off
  - system keys, contiguous, not materialized, compression, MMIO, ...

- Sample systems: MonetDB, Vertica, SAP HANA

# Main-Memory Databases

- RAM faster than disk → load data into RAM, process there
  - Multi-TB RAM
  - CPU, GPU, ...
- Largely giving up ACID's Durability → different approaches
  - Snapshots, transaction logging, non-volatile RAM (battery!), replication, ...
- Sample systems:
  - ArangoDB, HSQLDB, MonetDB, SAP HANA, VoltDB, Oracle TimesTen, ...



# Arrays in SQL



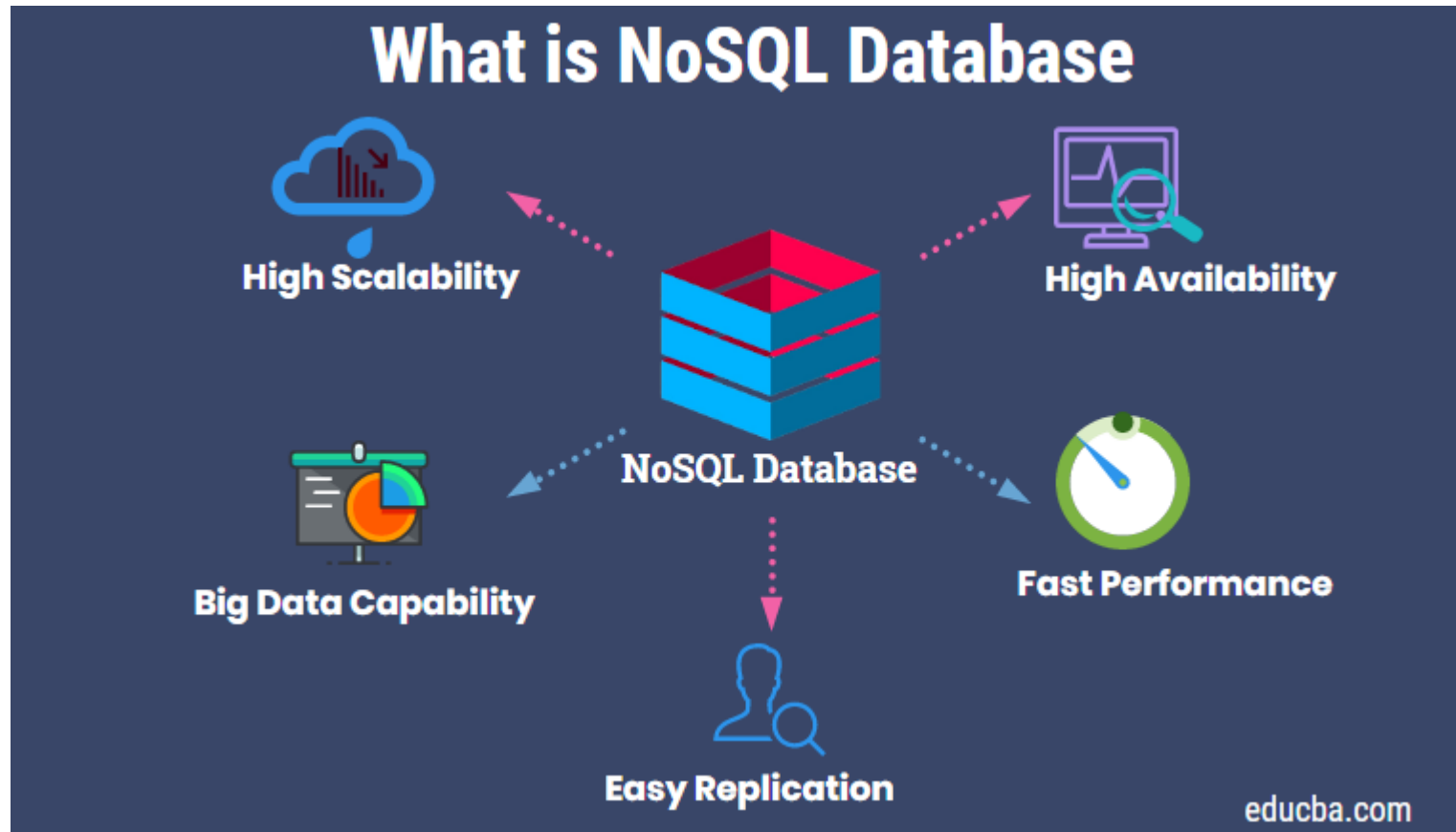
- 2014 - 2018
- rasdaman as blueprint

```
create table LandsatScenes(
  id: integer not null, acquired: date,
  scene: row( band1: integer, ..., band7: integer ) mdarray [ 0:4999,0:4999] )
```

```
select id, encode(scene.band1-scene.band2)/(scene.nband1+scene.band2)), „image/tiff“ )
from LandsatScenes
where acquired between „1990-06-01“ and „1990-06-30“ and
  avg( scene.band3-scene.band4)/(scene.band3+scene.band4)) > 0
```

# Summary

# Assessment: A NoSQL View on NoSQL



# Ex: Common NoSQL Misconception

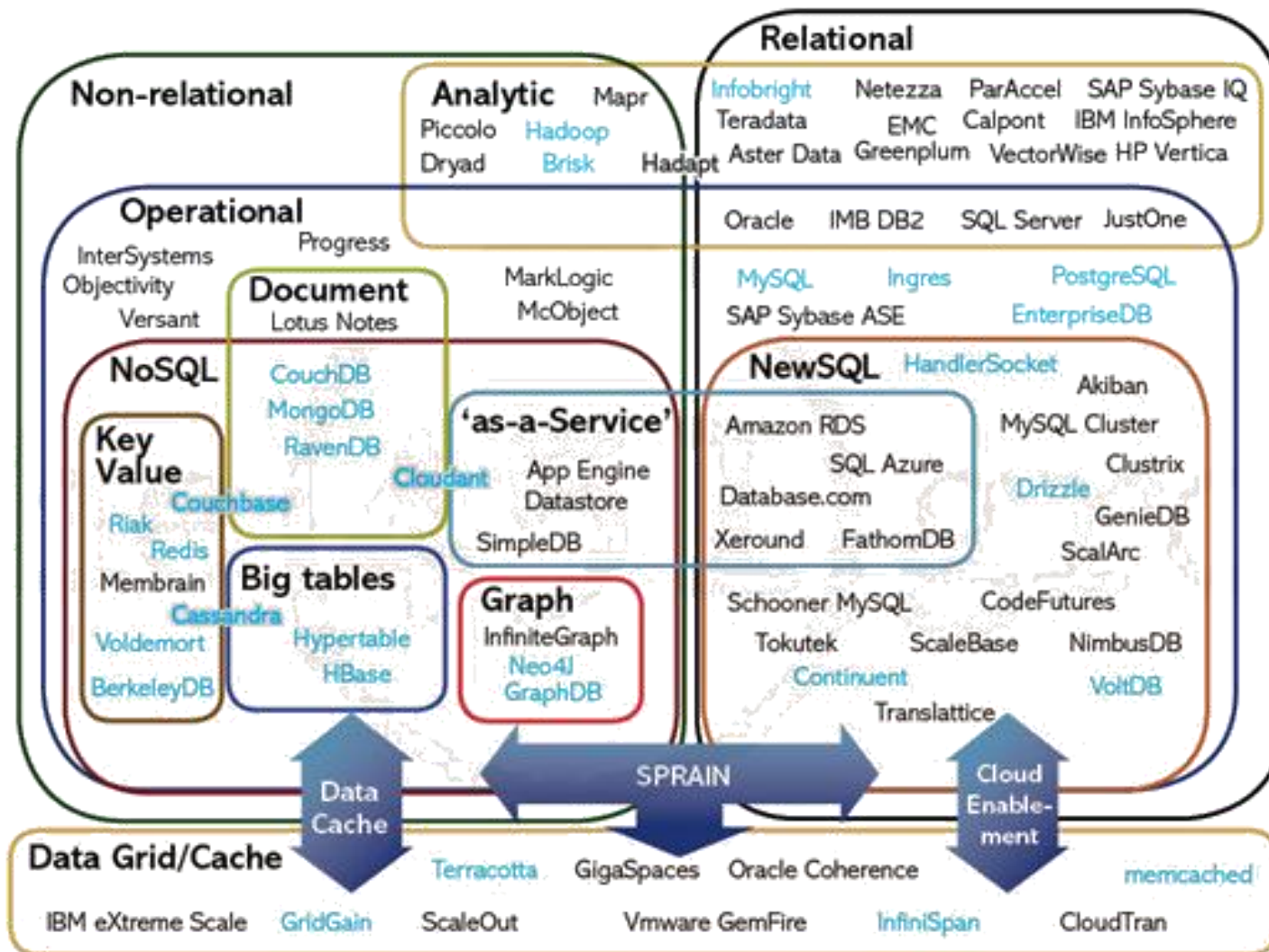
- “NoSQL virtually can store any kind of data whether it’s structured or unstructured. It can store text, audio, video, graphics or any kind of files.”
- ...no better than a file system.
- Ever heard of BLOBs in RDBMSs?
- NoSQL can answer question...on metadata. Just like any RDBMS.
- No QL support for these data types: cannot ask for
  - „audio containing music in minor?“
  - „someone saying yes my dear?“
  - „hate speech?“

# Top 10 NoSQL Databases in 2022





# The Explosion of DBMSs

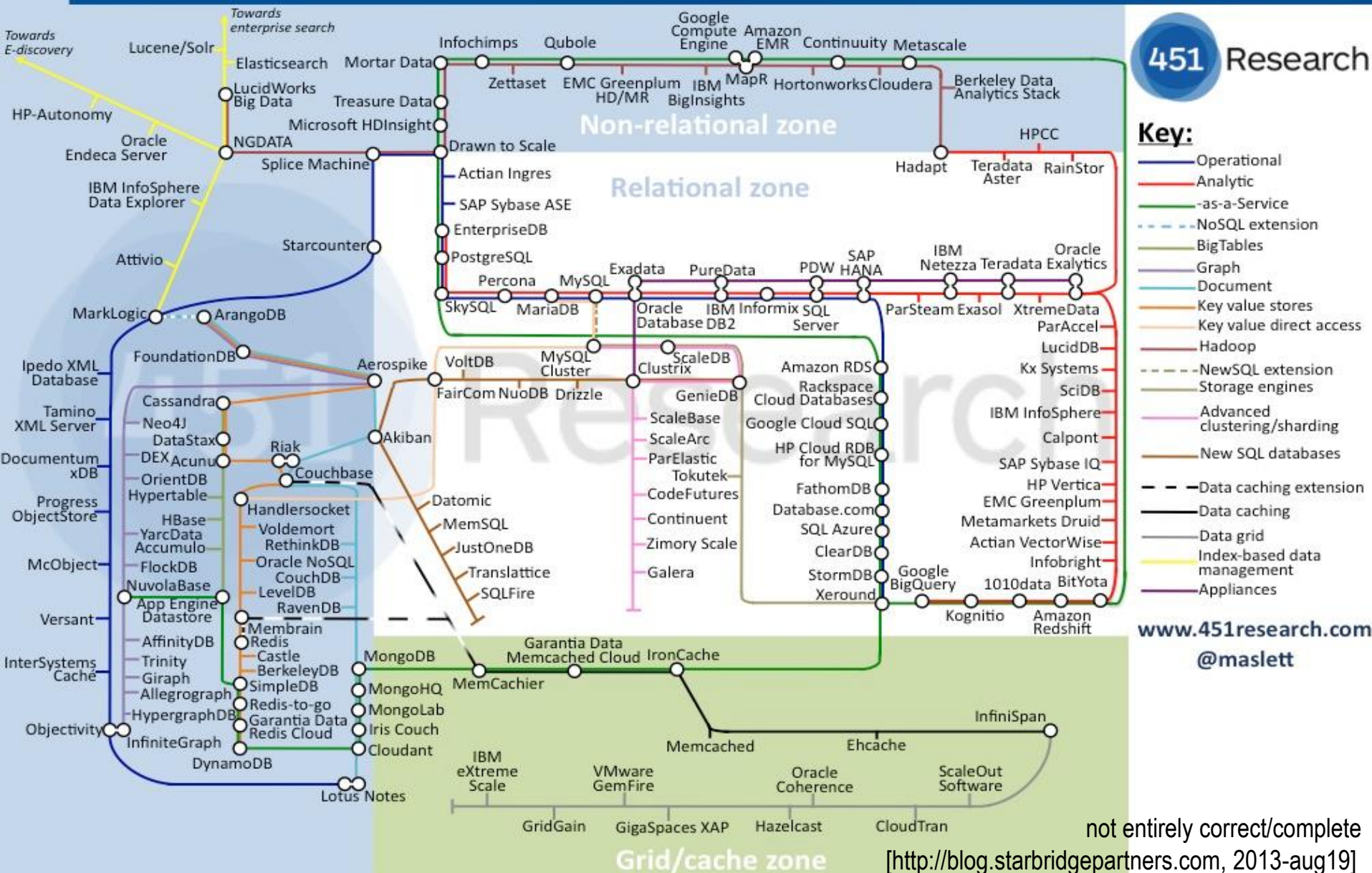


[451 group]

...not  
entirely correct

# Database Landscape Map – December 2012

451 Research



not entirely correct/complete

[<http://blog.starbridgepartners.com>, 2013-aug19]

# Summary & Outlook

- Fresh approach to scalable data services: NoSQL, NewSQL
  - Diversity of technology → pick best of breed for specific problem
- The future:
  - Avenue 1: **Modular data frameworks** to coexist - Heterogeneous model coupling barely understood - needs research
  - Avenue 2: concepts **assimilated by relational vendors** - Like fulltext, object-oriented, SPARQL, ... cf „Oracle NoSQL“
  - “SQL-as-a-service”: Amazon RDS, Microsoft SQL Azure, Google Cloud SQL
- Caveat: NoSQL **not interoperable**, „vendor“ lock-in