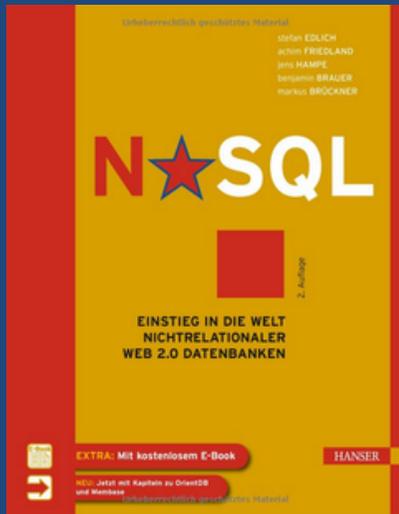




„NoSQL, NewSQL, ... Status Quo, Newcomers  
and some Future Predictions““



BOOKS



#1 WEBSITE



CONFERENCES

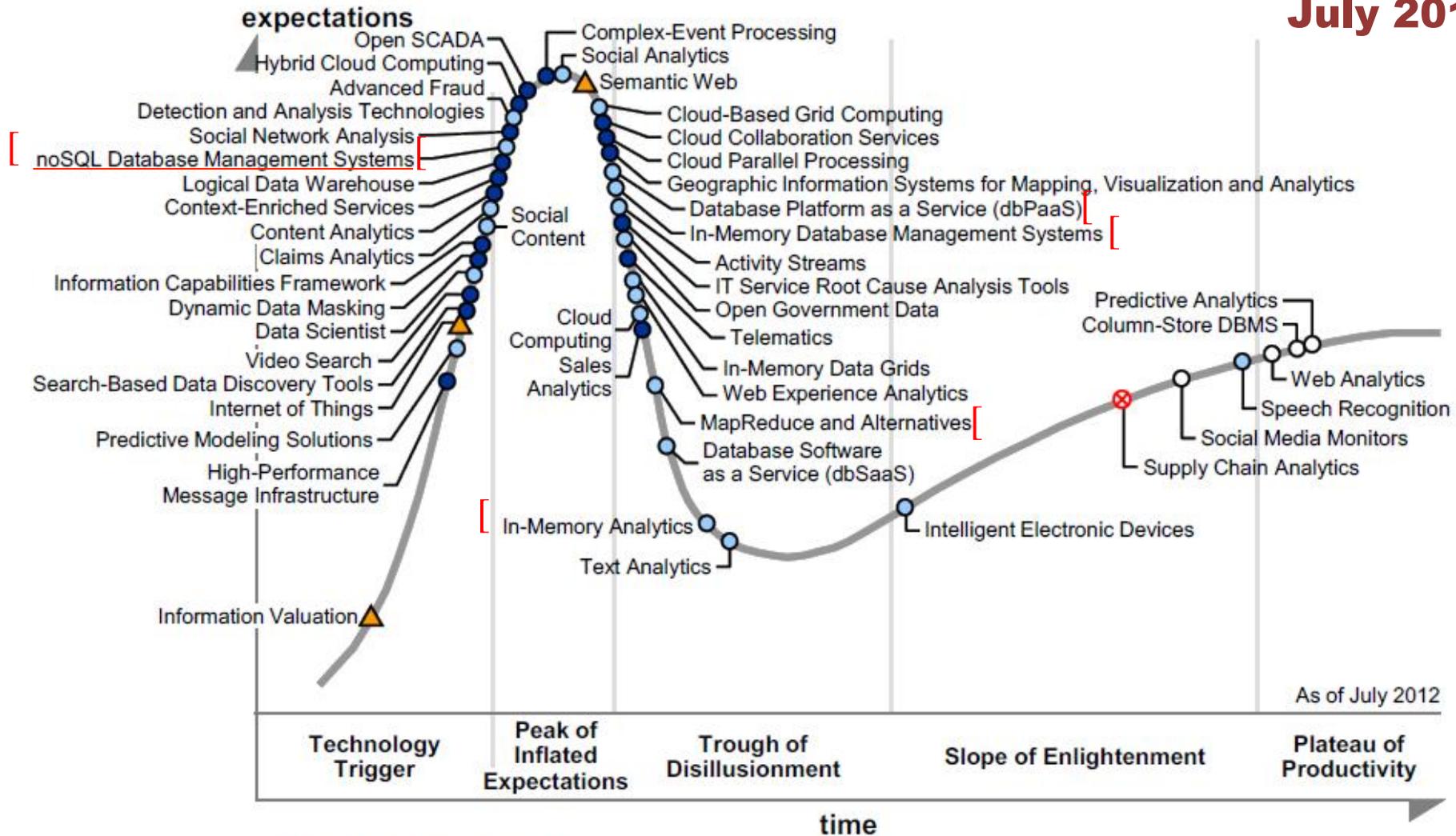


3y+ Consultant

1. Where are we?
2. What's new?
3. Some predictions for the future?

Figure 1. Hype Cycle for Big Data, 2012

July 2012



Plateau will be reached in:

- less than 2 years
- 2 to 5 years
- 5 to 10 years
- ▲ more than 10 years
- ⊗ obsolete before plateau

Source: Gartner (July 2012)

in 2009 „In the next years we will see a consolidation and many databases and DB companies will die!“

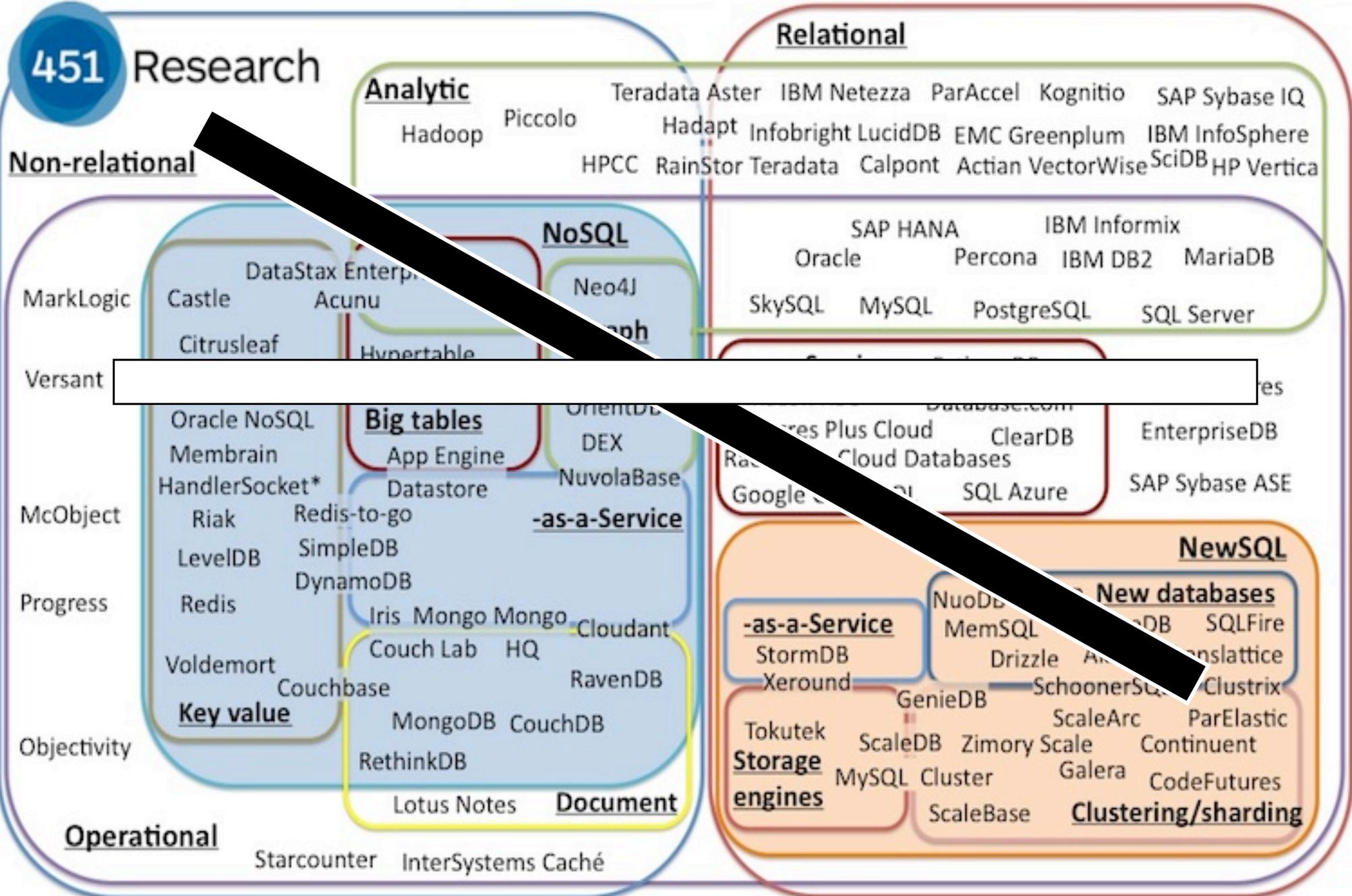


Rank	Last Month	DBMS	Database Model	Score	Changes
1.		1. <a href="#">Oracle</a>	Relational DBMS	1560.59	+27.20
2.	↑	3. <a href="#">MySQL</a>	Relational DBMS	1342.45	+47.24
3.	↓	2. <a href="#">Microsoft SQL Server</a>	Relational DBMS	1278.15	-40.21
4.		4. <a href="#">PostgreSQL</a>	Relational DBMS	174.09	-3.07
5.		5. <a href="#">Microsoft Access</a>	Relational DBMS	161.40	-8.77
6.		6. <a href="#">DB2</a>	Relational DBMS	155.02	-4.31
7.		7. <a href="#">MongoDB</a>	Document store	129.75	+5.52
8.	↑	9. <a href="#">SQLite</a>	Relational DBMS	88.94	+5.68
9.	↓	8. <a href="#">Sybase</a>	Relational DBMS	80.16	-5.25
10.		10. <a href="#">Solr</a>	Search engine	46.15	+2.99
11.		<a href="#">Teradata</a>	Relational DBMS	44.93	
12.		11. <a href="#">Cassandra</a>	Wide column store	38.57	+2.21
13.		12. <a href="#">Redis</a>	Key-value store	35.58	+3.15
14.		13. <a href="#">Memcached</a>	Key-value store	24.80	-0.17
15.		14. <a href="#">Informix</a>	Relational DBMS	24.00	+0.10
16.		15. <a href="#">HBase</a>	Wide column store	21.84	+1.40
17.		16. <a href="#">CouchDB</a>	Document store	18.72	+0.42
18.		17. <a href="#">Firebird</a>	Relational DBMS	12.24	-1.54
19.		<a href="#">Netezza</a>	Relational DBMS	11.14	
20.		18. <a href="#">Sphinx</a>	Search engine	9.55	+0.09
21.		19. <a href="#">Neo4j</a>	Graph DBMS	8.34	+0.90
22.		21. <a href="#">Elasticsearch</a>	Search engine	8.31	+1.56
23.		22. <a href="#">Riak</a>	Key-value store	7.20	+1.10
24.		20. <a href="#">Vertica</a>	Relational DBMS	6.98	-0.42
25.		<a href="#">Greenplum</a>	Relational DBMS	6.29	
26.		26. <a href="#">Couchbase</a>	Document store	5.33	+1.16

# The evolving database landscape

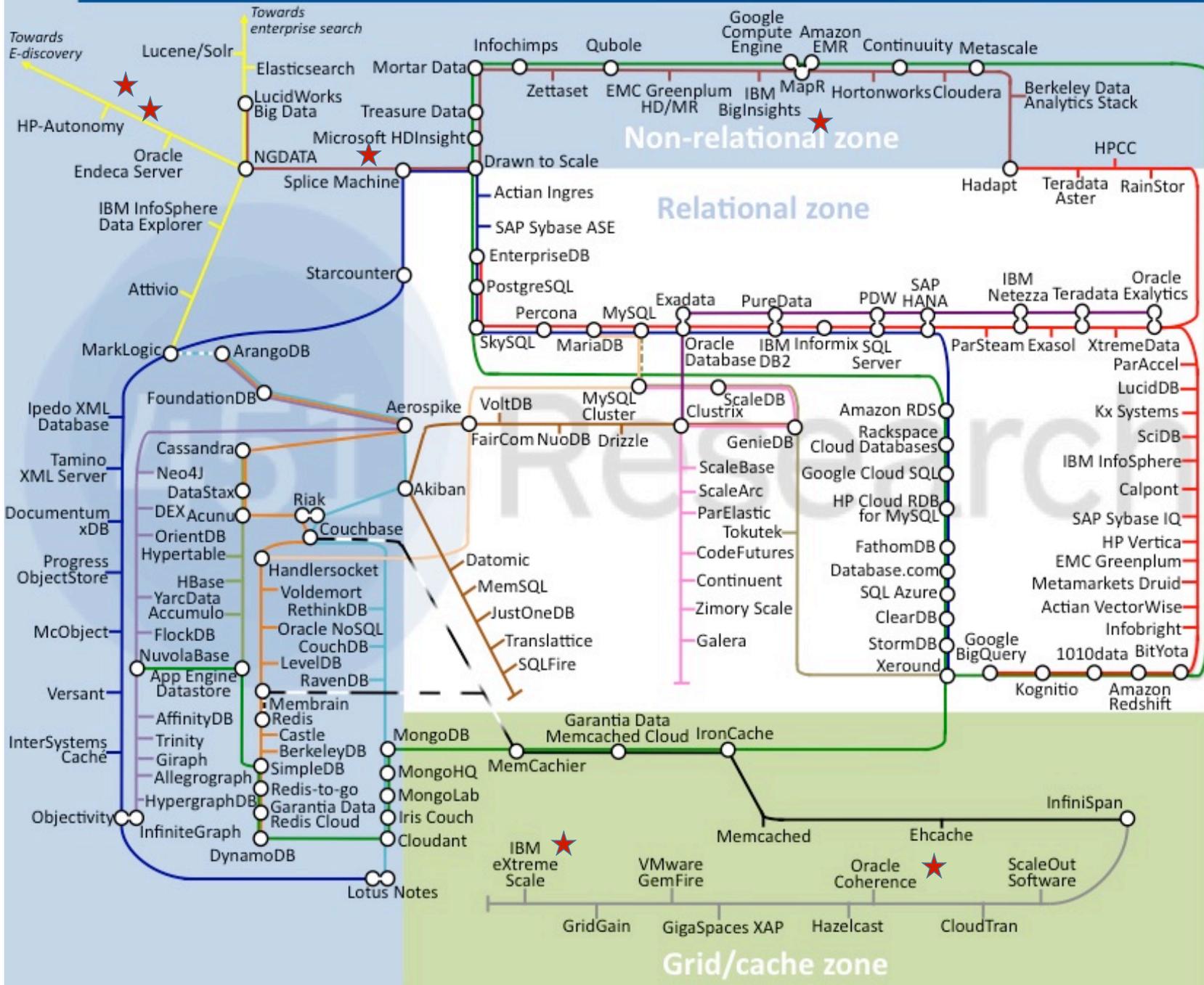
451 Research

**Non-relational**



# Database Landscape Map – December 2012

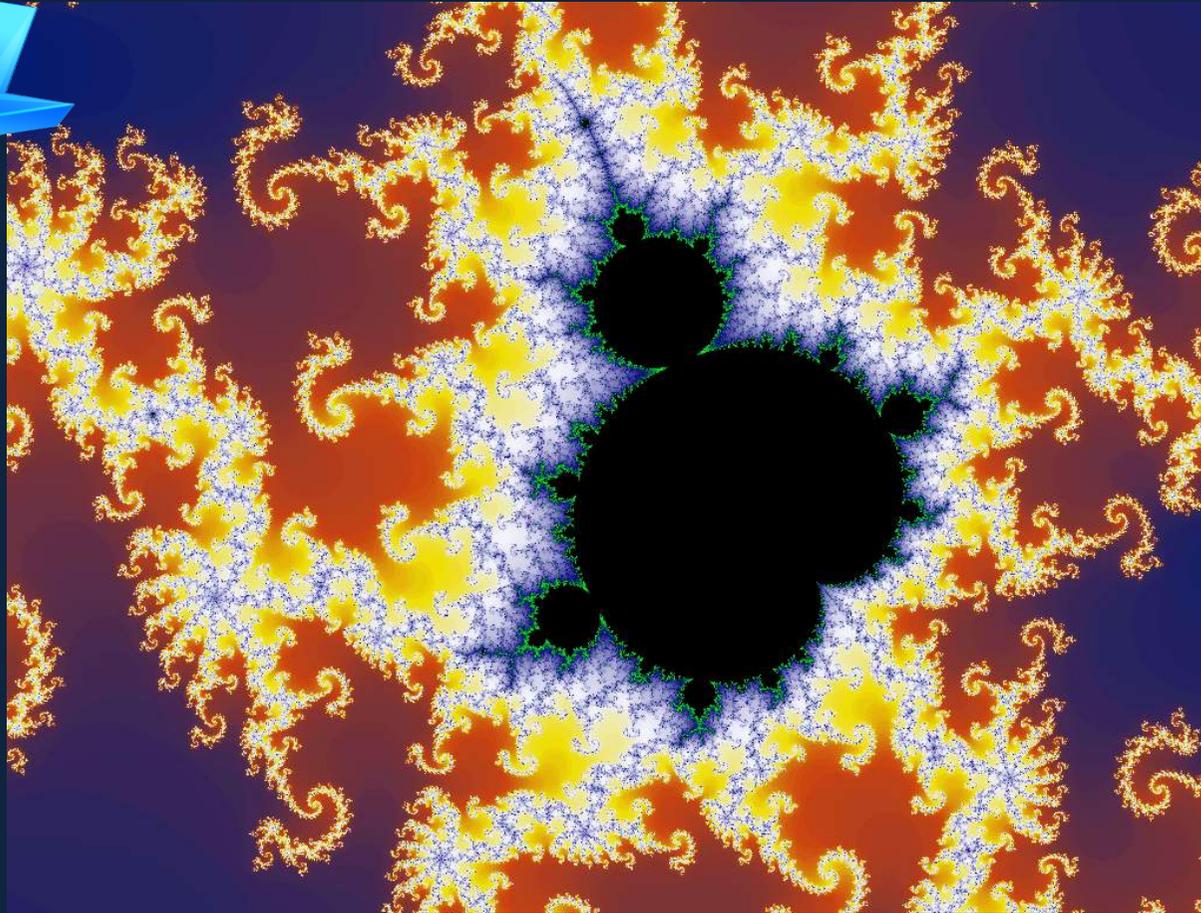
451 Research



- Key:**
- Operational
  - Analytic
  - as-a-Service
  - NoSQL extension
  - BigTables
  - Graph
  - Document
  - Key value stores
  - Key value direct access
  - Hadoop
  - NewSQL extension
  - Storage engines
  - Advanced clustering/sharding
  - New SQL databases
  - Data caching extension
  - Data caching
  - Data grid
  - Index-based data management
  - Appliances

www.451research.com  
@maslett

# Expanding Universe...



# In 5-N years... DBs like the smartphone market



„excessive  
feature  
exchange“

Microsoft  
PostgreSQL Oracle  
IBM SAP

JSON  
Hadoop Scalability  
relaxed-Schema  
NewSQL



Schema-support  
love-SQL  
Management-Support  
tunable-Consistency  
OLAP-Mining-Analytics

MongoDB Cassandra  
Aerospike Riak  
Graph-DBs

# analogy Programming Languages

multicore  
no state

OOP

functional

logic

3		2	4		6			
4						5	3	
1	8	9	6	3	5	4		
			8		2			
	7	4	9	6	8		1	
8	9	3	1	5	6		4	
		1	9	2	5			
2		3			7	4		
9	6	5			3		2	

```
:- use_module(library(clpfd)).
sudoku(Rows) :-
length(Rows, 9), maplist(length_9, Rows),
append(Rows, Vs), Vs ins 1..9,
maplist(all_distinct, Rows),
transpose(Rows, Columns), maplist(all_distinct, Columns),
Rows = [A,B,C,D,E,F,G,H,I],
blocks(A, B, C), blocks(D, E, F), blocks(G, H, I).
length_(L, Ls) :- length(Ls, L).
blocks([], [], []).
blocks([A,B,C|Bs1], [D,E,F|Bs2], [G,H,I|Bs3]) :-
all_distinct([A,B,C,D,E,F,G,H,I]),
blocks(Bs1, Bs2, Bs3).
```

Language / Programming  
Paradigm

Database  
Paradigm

Logic: rule based deduction



declarative querying  
abstract query composition

composition, abstractness

Functional: immutability



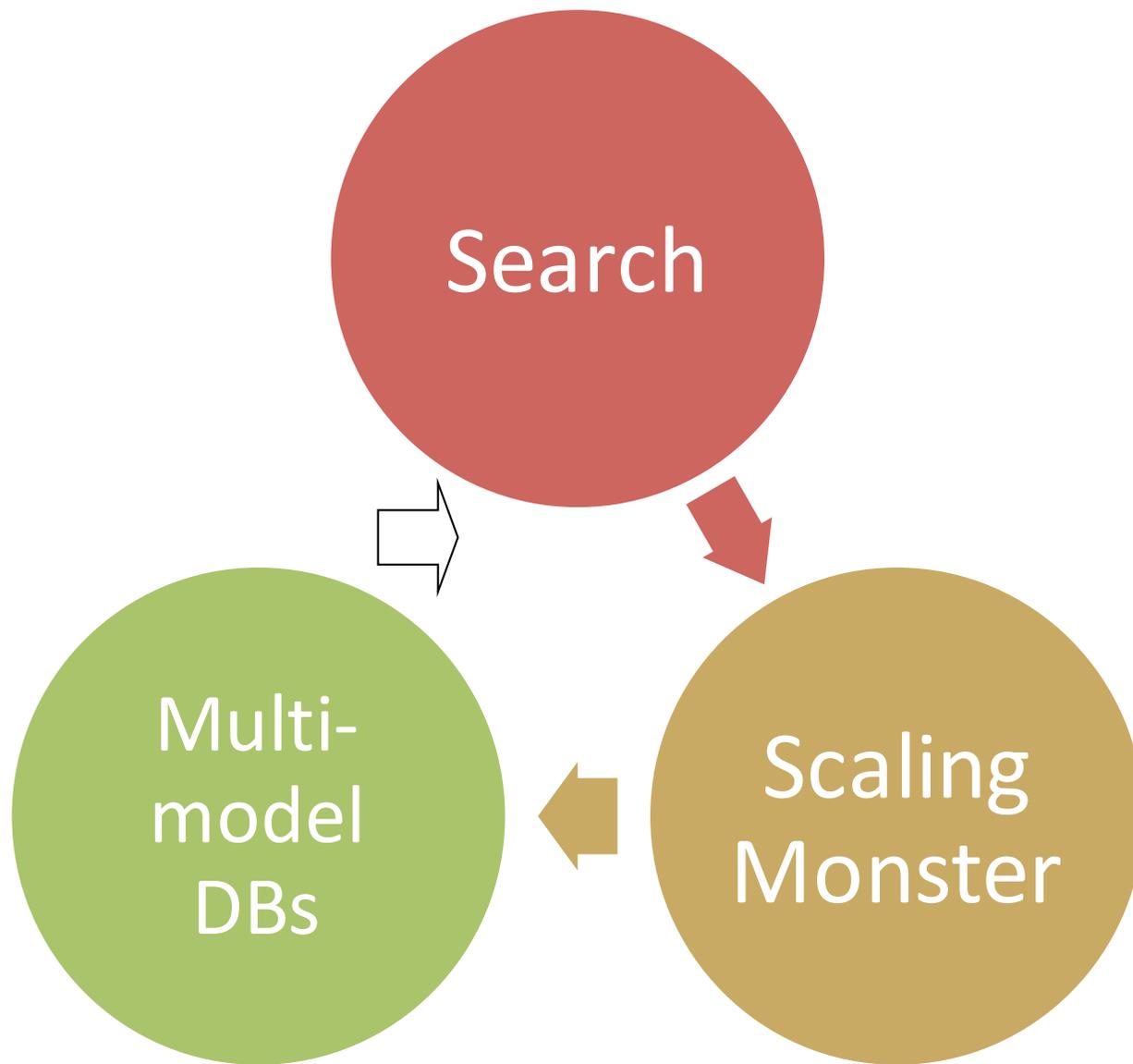
MVCC, append only, SSD ready  
„single source of truth“

OO / Imperative

uncontrolled state  
manipulation

## **Part II**

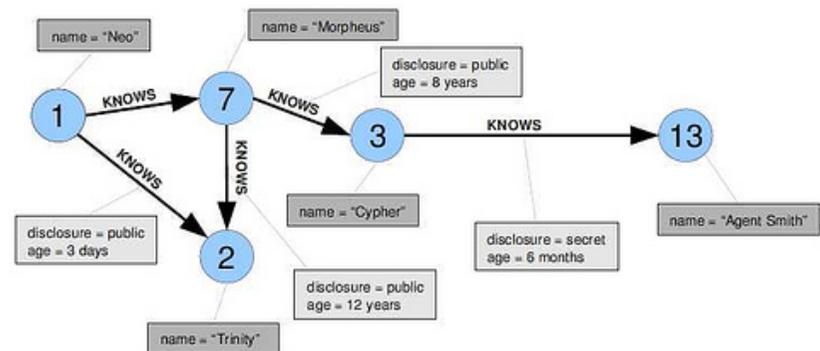
**new Ideas & Newcomers**



RAM? SSD?  
Disk?



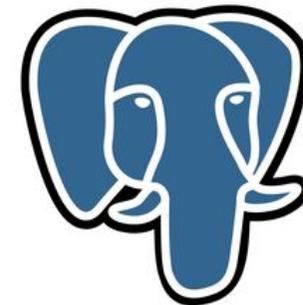
„The amount of  
Multimodel-Databases  
will increase dramatically!“



Microsoft Trinity?  
IBM RDF Store?!  
Object-Databases?  
SAP Hana Graph Engine

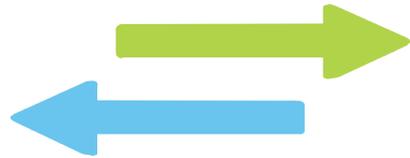
 **MarkLogic**

PostgreSQL



2nd Iteration

JSON  
KV



tables,

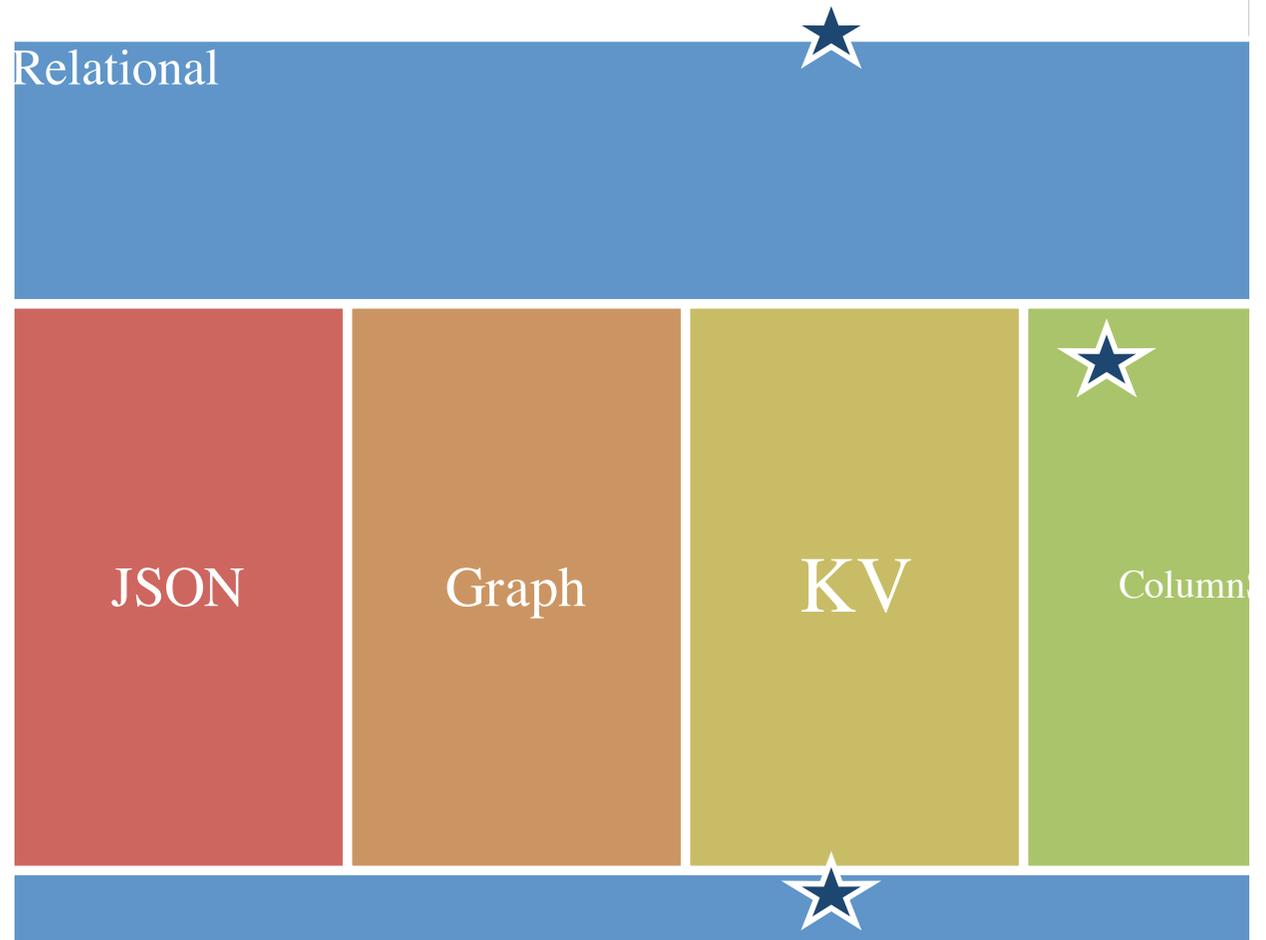


```
json_agg(anyrecord) -> json      => json array
to_json(any) -> json
hstore_to_json(hstore) -> json (also used as a cast)
hstore_to_json_loose(hstore) -> json
```

```
postgres=# SELECT json_agg(aa) FROM aa;
Access JSON Objects: -> and ->>
condense fields with: #> and #>>
```

```
postgres=# CREATE TABLE aa (id int, txt hstore);
CREATE TABLE
postgres=# INSERT INTO aa VALUES (1, 'f1=>t, f2=>2, f3=>"Hi", f4=>NULL');
INSERT 0 1
postgres=# SELECT id, txt::json, hstore_to_json(txt) FROM aa;
id | txt | hstore_to_json
-----+-----+-----
1 | {"f1": "t", "f2": "2", "f3": "Hi", "f4": null} | {"f1": "t", "f2": "2",
"f3": "Hi", "f4": null}
(1 row)
```

# Multimodel-Databases



Example: Handler Socket first 2010, now 2013 Oracle / MySQL

Premise: Atoms



inducible to MultimodelDBs



Labs do work on:  
**DB Personalities!**

```
1 db = openDatabase(id);  
2 db.selectPersonality(GRAPHDB);  
3 // db.selectModel(JSONDB)  
4 db.insert(graph-node1);  
5 db.insert(graph-node2);  
6 db.query(cypherQuery);  
7 ...
```

K/V rebirth?!





# Polyglot outside + Polyglot inside



Polyglot inside  
„Yes I can!“ 2-  
N Models



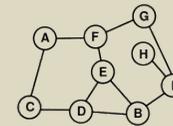
DB 1  
Model 1



DB 2 M

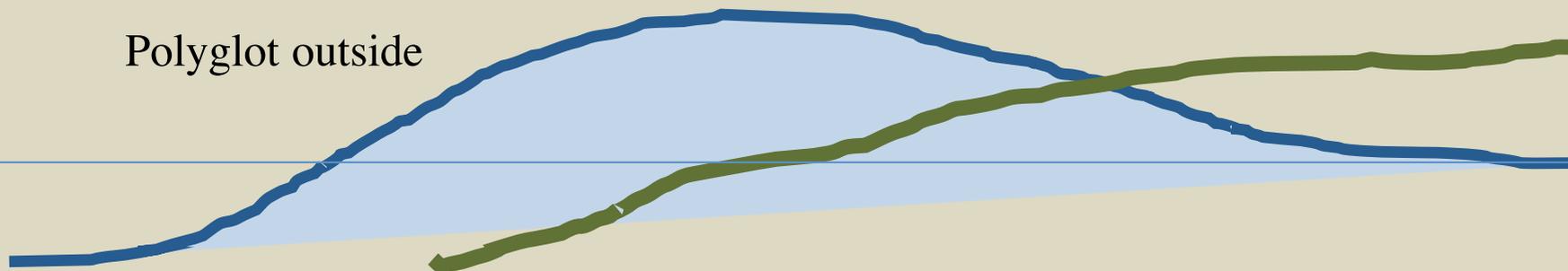


DB 3  
Model 3



Polyglot outside

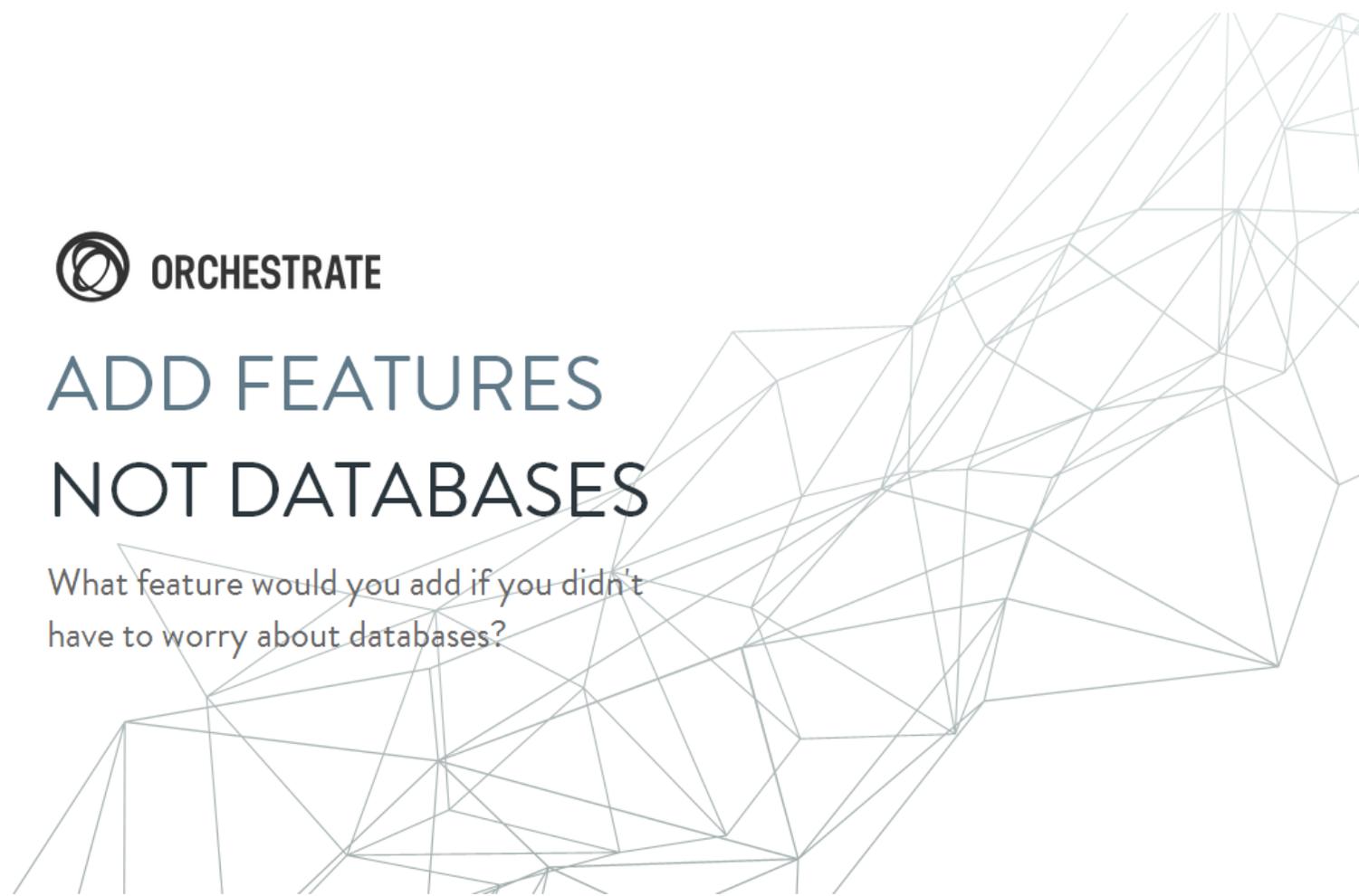
Polyglot inside





# ADD FEATURES NOT DATABASES

What feature would you add if you didn't  
have to worry about databases?





Datomic  
Data, meet Simple



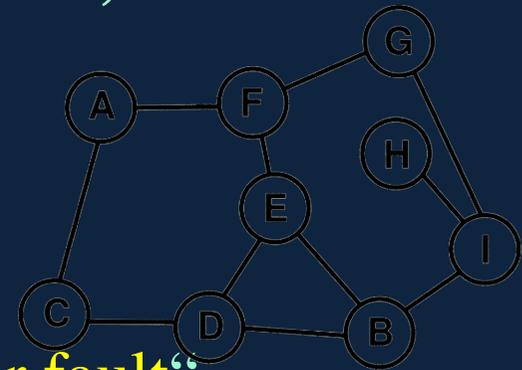
- ✓ Atoms => Datomic!
- ✓ Clojure Principles, 2-3 persons, 2 years work
- ✓ Immutable Data MVCC, replay, compare DBs, history juggling (query „as of“ or „time-windows“)
- ✓ Nathan Marz: „Single Source of Truth...“



Datomic  
Data, meet Simple

- ✓ ACID LAYER
- ✓ unlimited read scalability  
e.g. with DynamoDB
- ✓ Full-Text Search
- ✓ Smart Caching / Data Gravity

- ✓ minimal Schema / Metadata
- ✓ has cardinality
- ✓ references are „bi“ by default,  
=> graph ,alike‘ queries



Stuart Halloway „The impedance mismatch is our fault“



```
(unifier '[(?a * ?x | 2) + (?b * ?x) + ?c]
         '[?z + (4 * 5) + 3])
```

```
;=> [(?a * 5 | 2) + (4 * 5) + 3]
```

# ✓ Datalog + Queries!

✗ Rectangulation of Data

✗ Place Oriented Programming

P unification (MGU) + logic capabilities

P Code as Data

```
(datomic/q '[:find ?name ?url
            :where (lang-anchor ?name ?url)]
  [[-100 :language/name "Visi"]
   [-200 :language/name "Ioke"]
   [-3 :language/name "Frink"]
   [-4 :language/name "Roy"]
   [-100 :language/url #url "http://visi.io"]
   [-200 :language/url #url "http://ioke.org"]]
the-rules)
```



```
(def people [[ "ben" 35]
             [ "jerry" 41]])

(??- (<- [?name ?age]
        (people ?name ?age)
        (< ?age 40)))

;=> ((["ben" 35]))
```

- ✓ lessons from Cascalog
- ✓ logic querying
- ✓ recursive rules / queries
- ✓ implicit joins

# ✓ abstractions + composability!

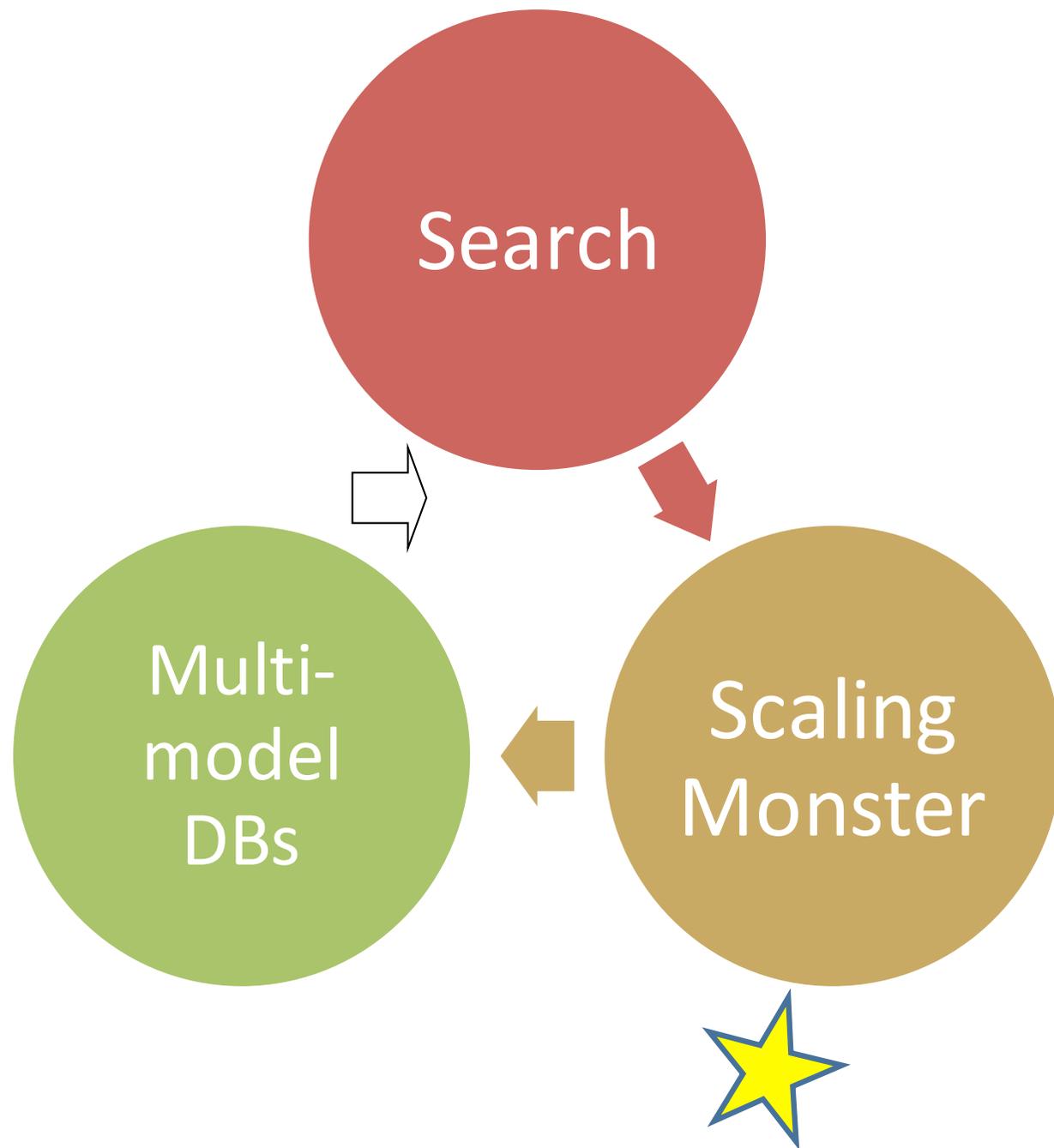
```
1 | (defn constrain
2 |   "Apply constraints to a query"
3 |   [query constraints]
4 |   (update-in query [:where] (partial apply concat) constraints))
```

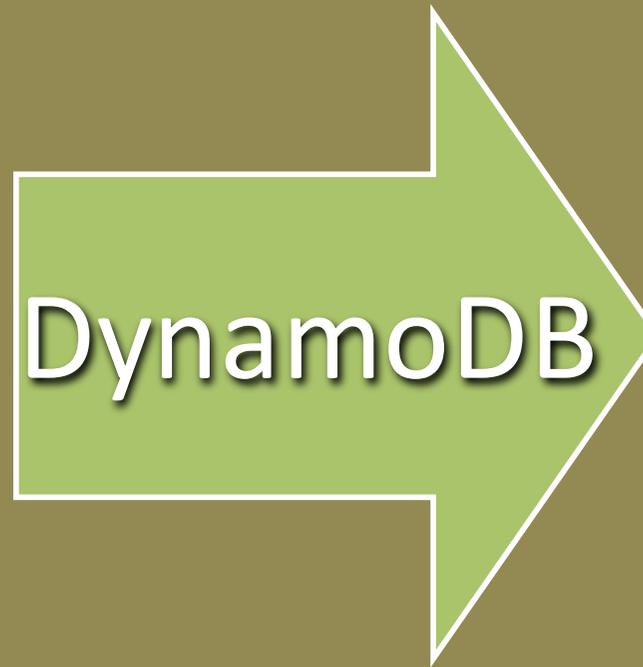
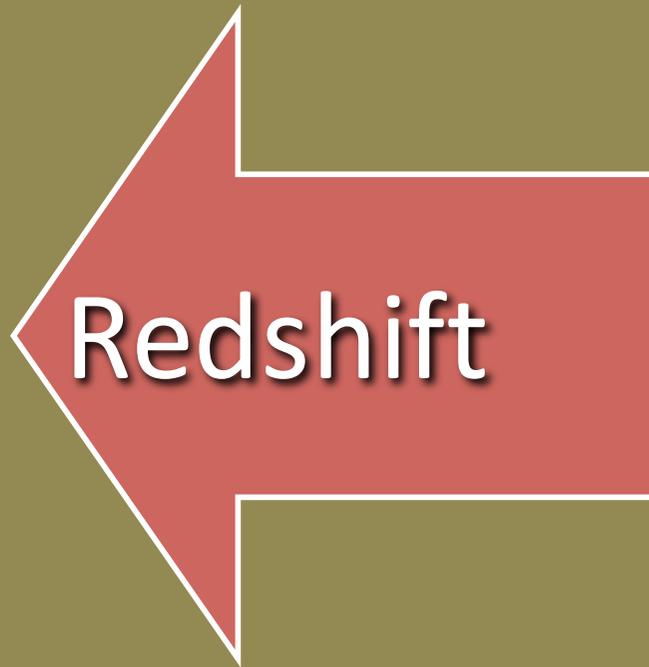
```
1 | {:find [?v], :in [$ %], :where [(vote-join ?v ?y ?p ?s)]}
```

```
1 | (constrain base-query [(year "2000")])
```

```
1 | {:find [?v], :in [$ %], :where ((vote-join ?v ?y ?p ?s) [?v :year "2000"])}
```







- SSD
- no maintenance
- zone replication
- backup & restore to S3
- 40 mio requests free
- ...



A fully managed NoSQL-like data storage service that scales with your users.

- JSON over http or ProtoBuf Interface
- ACID Transactions
- Partially strong or eventual consistent
- Multi Datacenter
- No downtime
- Autoscale
- 0.09\$ write, 0,0´6\$ read per 100k ops

SSD

?

Datomic, Lucene 4 / ElasticSearch, DynamoDB,

...

Pattern: write once + append only

RAM – SSD – Disk

Jim Gray, ... & Götz Gräfe

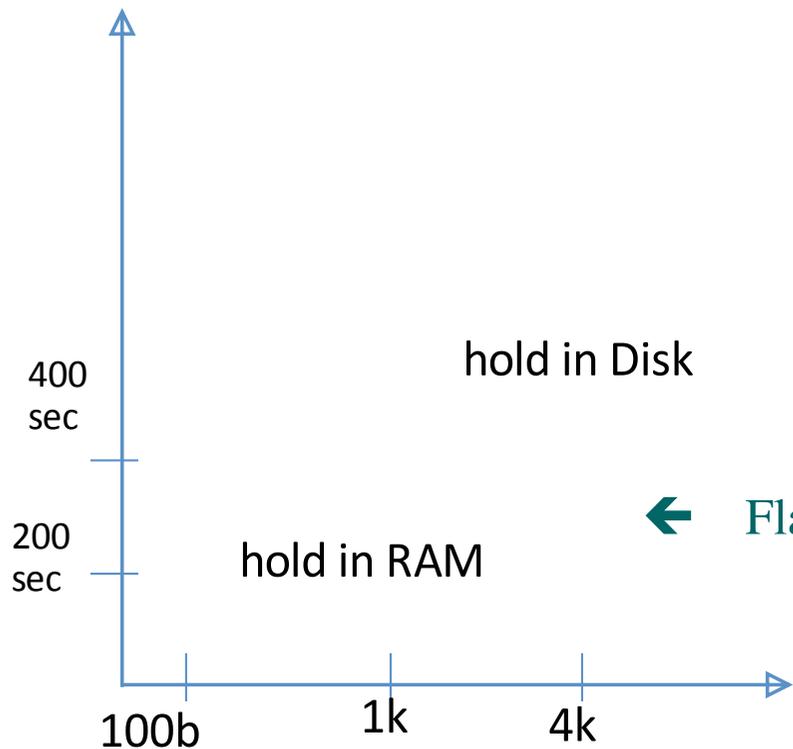
„The 5 Minute Rule for Disc Access and the 5 Byte Rule  
for

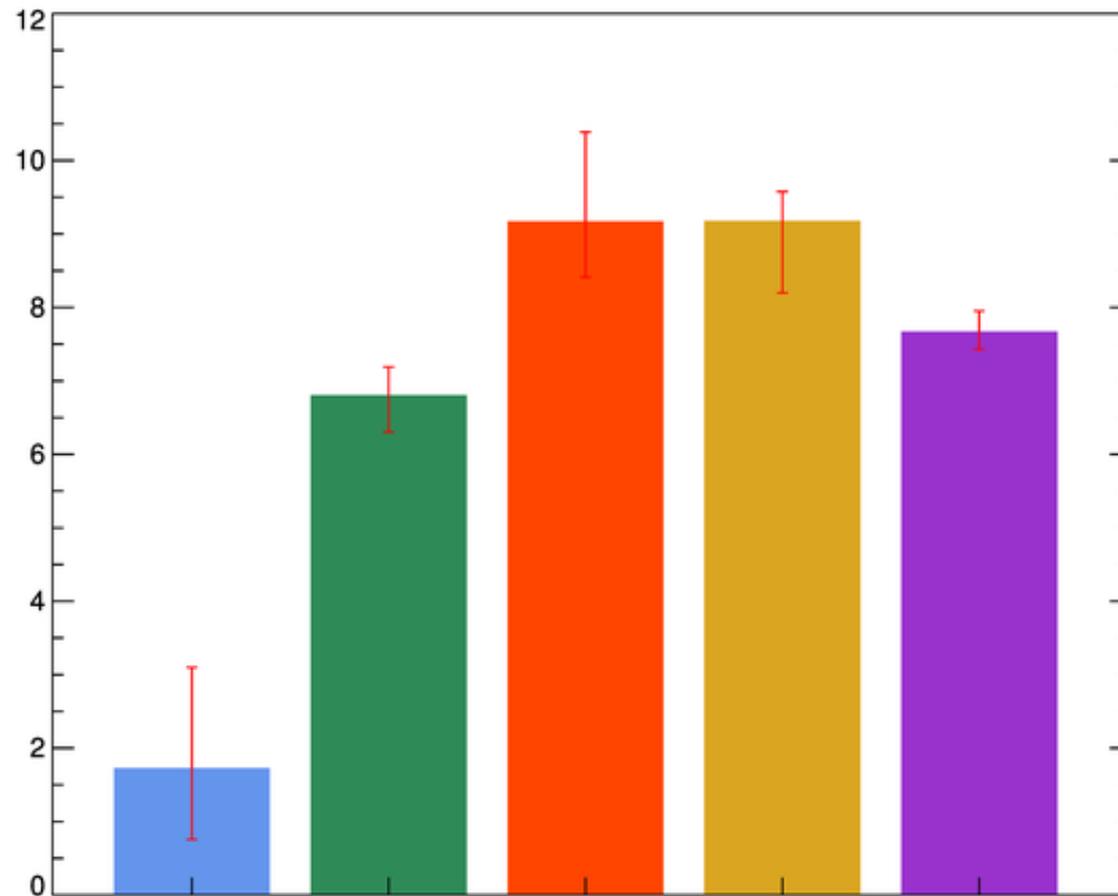
Trading Memory for CPU Time“ 1987, 1997, 2007

$$\text{BreakEvenInSeconds} = (\text{PagesPerMBofRAM} / \text{AccessesPerSecondPerDisk}) \times (\text{PricePerDiskDrive} / \text{PricePerMBofRAM})$$

- Hybrid Disks?!
- Slow vs. Fast Write SSDs?!
- Rule valid only for big blocks
- 5248 sec for 4k blocks
- 5M: big blocks: Ram -> Flash
- 5M: small blocks: RAM -> Disk
- ✓ 100.000 erase-write cycles: no problem
- ✓ important: energy efficiency
- ✓ append-only fits to, wear-leveling'
- Access Time and Access Patterns ☹

1987, 1997: 1K, <400sec => RAM





hot

fast

medium

low

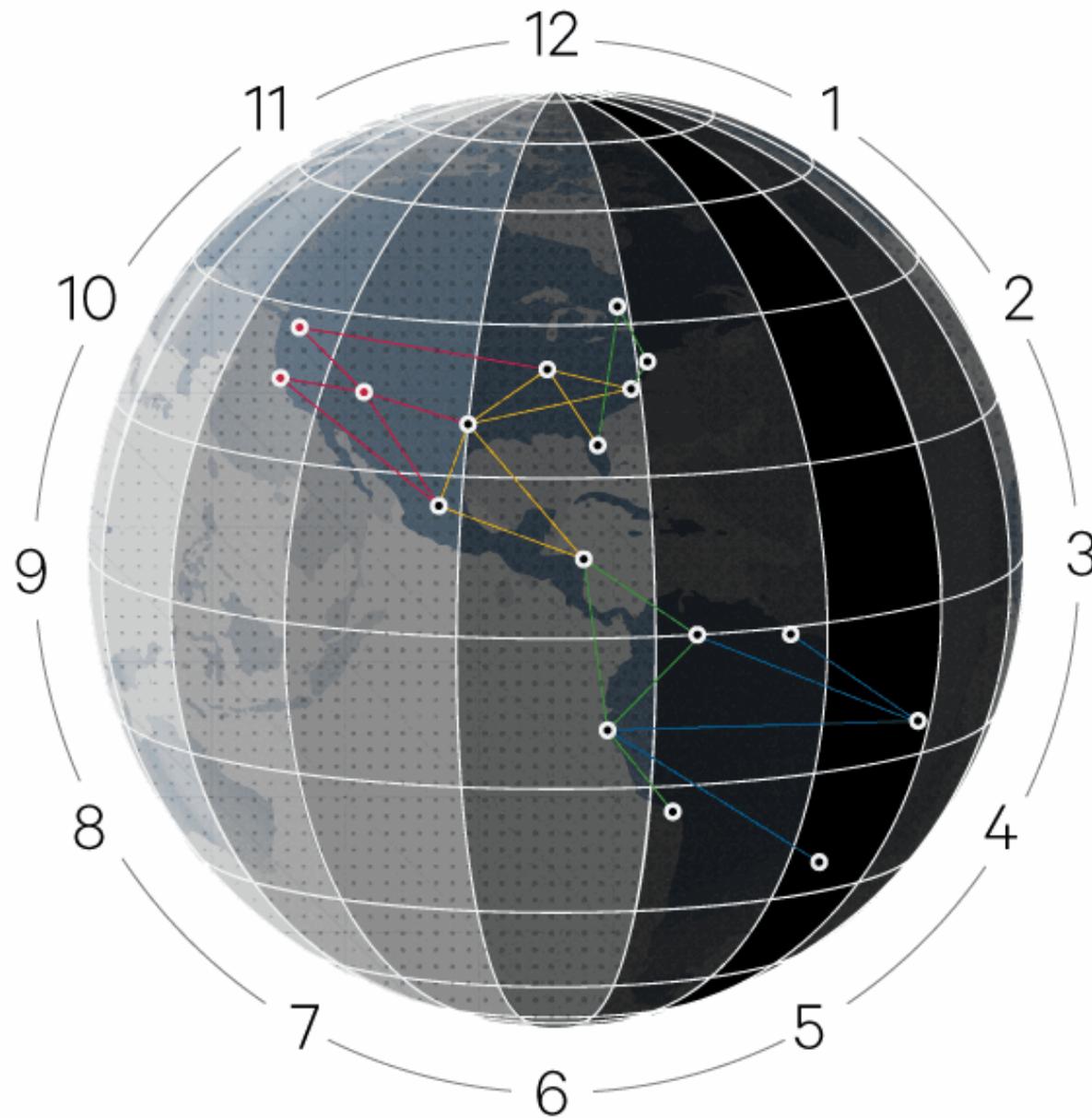
archive

delete!

RAM ⇒ SSD ⇒  
Disk

Register ⇒ L1 ⇒ L2 ⇒ RAM ⇒ SCM: PCM? ⇒ SSD ⇒ Disk ⇒ Tape ⇒  
/dev/null

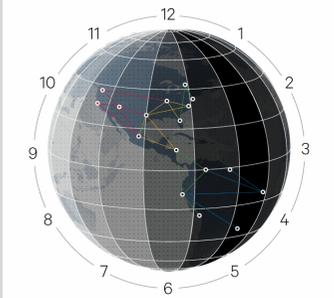
# Spanner





Jeff Dean 2009:

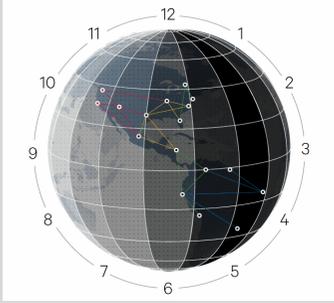
- many Exabyte ( $10^{18}$ )
- millions of machines
- around the globe



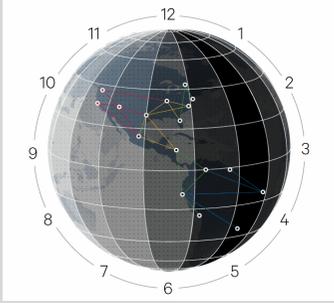
```
CREATE TABLE Users {  
  uid INT64 NOT NULL,  
  email STRING  
} PRIMARY KEY (uid), DIRECTORY;
```

```
CREATE TABLE ALBUMS {  
  uid INT64 NOT NULL, aid INT64 NOT NULL,  
  name STRING  
} PRIMARY KEY (uid, aid),  
INTERLEAVE IN PARENT Users ON DELETE CASCADE;
```

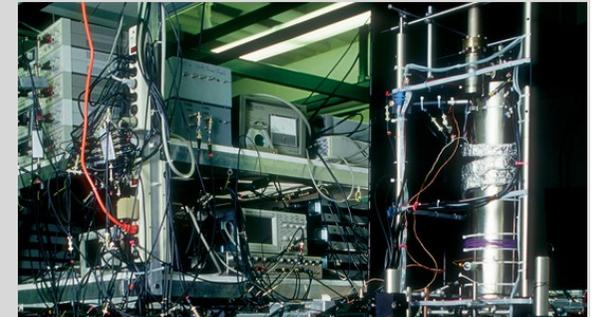
- protobuf encoded columns
- extended SQL syntax
- logically denormalized, physically one Objekt



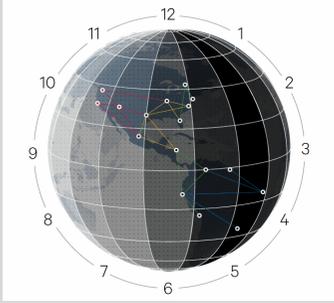
„Time synchronisation on this level  
is not possible!“



True-Time API: 1-7ms

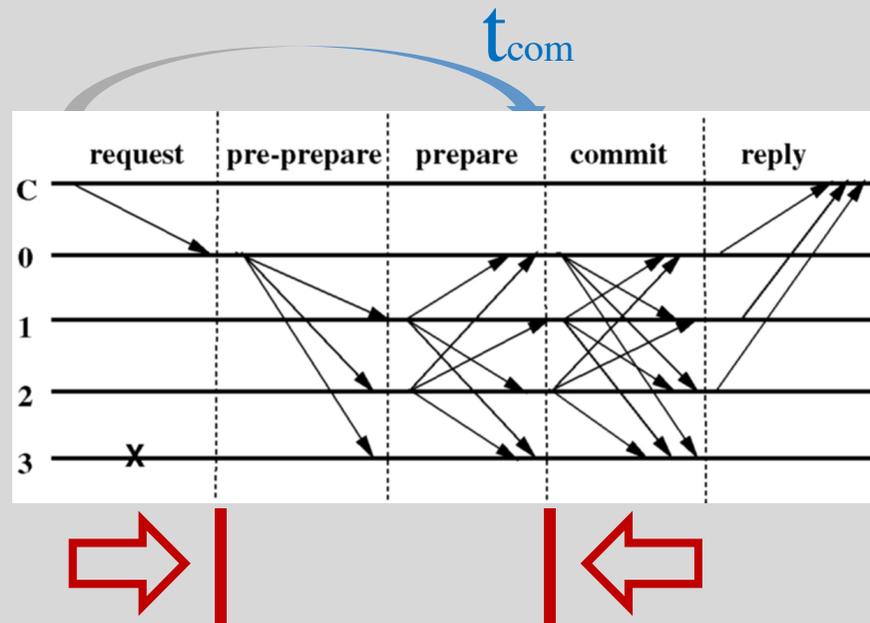


GPS-Antennas & Atomic-Clocks  
„An atomic clock is not that expensive!”



# Paxos wait minimal for RW-Transactions ✓

⇒ |  $2\varepsilon$  | ⇐

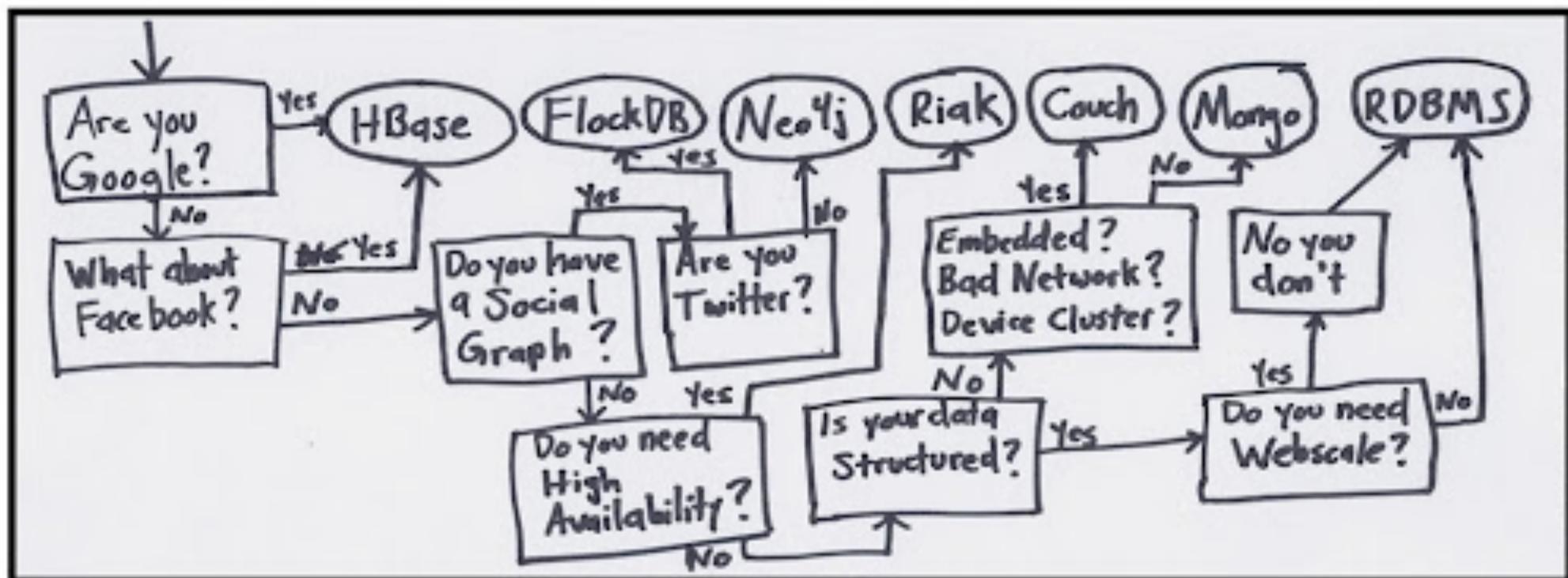


a dozen truly global databases till 2018



## **Part III**

**How to ...  
...tie up the loose ends?!**



## Cluster 1: Know & Segment your data

### Analyze & Categorize it:

- Domain-Data
- Log-Data
- Event-Data
- Message-Data
- critical Data
- Business-Data
- Meta-Data
- temp Data
- Session-Data
- Geo Data
- etc.

### Data- / Storage Model:

- relational
- column-o
- doc-alike
- graphs
- objects
- multivalued
- objects=ORM
- JSON
- BLOBS
- etc. (beyond bit-bucket)

### Data / Type constraints:

- Data-Navigation?
- Data Amount?
- Data Complexity (Deep XML?)
- Schema flexibility?
- Schema support needed?

### Persistence design: (Reference: (C) highscalability link to be inserted)

- Durability? On power failure?
- Memtable/SSTable; Append-only B-tree; B-tree; On-disk linked lists; In-memory replicated; In-memory snapshots; In-memory only; Hash; Pluggable.

## Cluster 2: Consistency Model

### Global consistency model:

- ACID / BASE / WATER?
- Ability to (fine) tune the consistency model

### CAP tradeoff:

- CP, AP, CA or tunable?

## Cluster 3: Performance Dimensions

- Latency / Request behaviour / distribution [High = 10, Low = 0]
- Throughput [High = 10, Low = 0]
- High Concurrency?

## Cluster 4: Query Requirements

- Typical queries look like?
- SQL needed? LINQ needed?
- BI / Analytic-Tools needed? (M/R sufficient?)
- Ad-Hoc Queries needed?
- Map/Reduce needed? Background data analytics?
- Secondary Indices
- Range queries
- Weird aggregations
- ColumnDB needed for Analytics?
- Views

## Cluster 5: Architecture and Patterns

### Architecture looks like:

- local, parallel, distributed / grid, service, cloud, mobile, p2p, ...
- Hosted? Cloud? Local? Datacenter?

### Data Access Patterns

- read / write distribution?
- random / sequential access?
- Access Design Patterns

+ Prototype

## Cluster 6: Non functional Requirements

- Replication needed? = Rubustness
- Automatic load balancing, partitioning, and repartitioning?
- Auto-Scaling
- Text search integration? Lucene / Solr?
- Refactoring Frequency?
- 24/7 System? Live add and remove?
- Developer Qualification
- DB simplicity? (installation, configuration, development, deployment, upgrade)
- Company restrictions?
- DB diversity (allowed?)
- Security? (authentication, authorization, validation?)
- Licence Model?
- Vendor trustworthiness?
- Community support?
- Documentation?
- Company and DB dev in the future?

### Costs:

- DB-Support? (responsiveness, SLA)
- Costs in general, Scaling Costs
- Sysadmin costs
- Operational Costs: (noOps)
- Safety / Backup & Restore
- Crash Resistance, Disaster Management
- Monitoring

> 75% relational fit!

<http://nosql-database.org>



# Architecture Vision - z.B. $\lambda$

## Speed

- High Latency updates
- Incremental algorithms

## Serving

- Updates vom Batch Layer
- Random Access

## Batch

- Source / „Source of Truth“
- Views

# Paradigm Shift

x10 more expensive!



early adopters win



# Executive Summary:



komplex „DB  
Smartpho  
Mash“



Immutability  
→ MVCC  
→ SSD



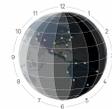
radika  
new



Multimodel &  
Global  
Databases



new  
Architectures &  
Checklists



contact [edlich@gmail.com](mailto:edlich@gmail.com)  
<http://edlich.de>

Thanks!

