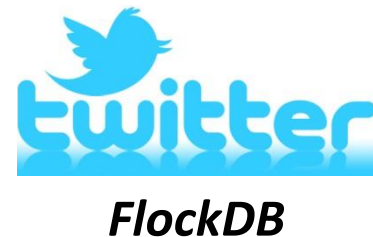
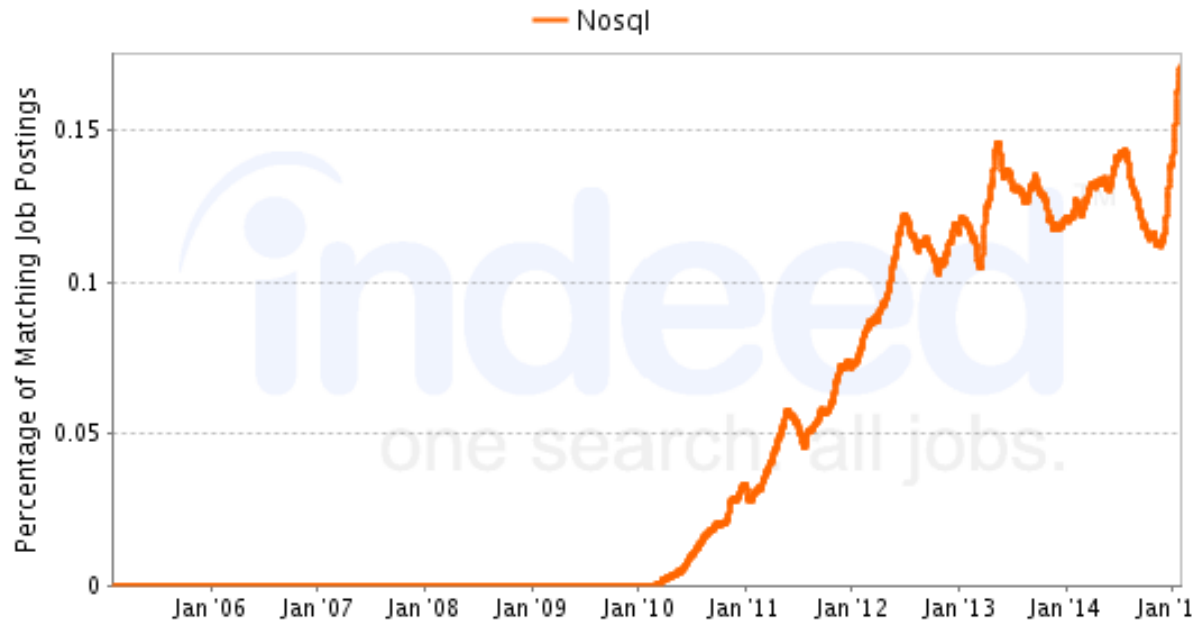


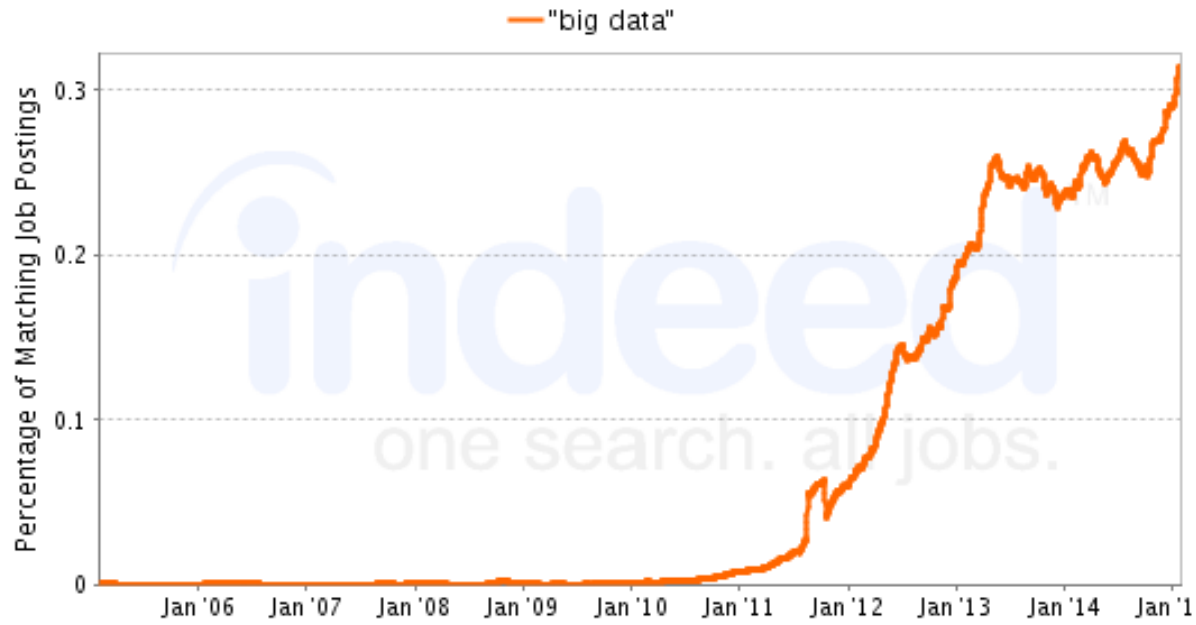
The NoSQL Movement



Job Trends from Indeed.com



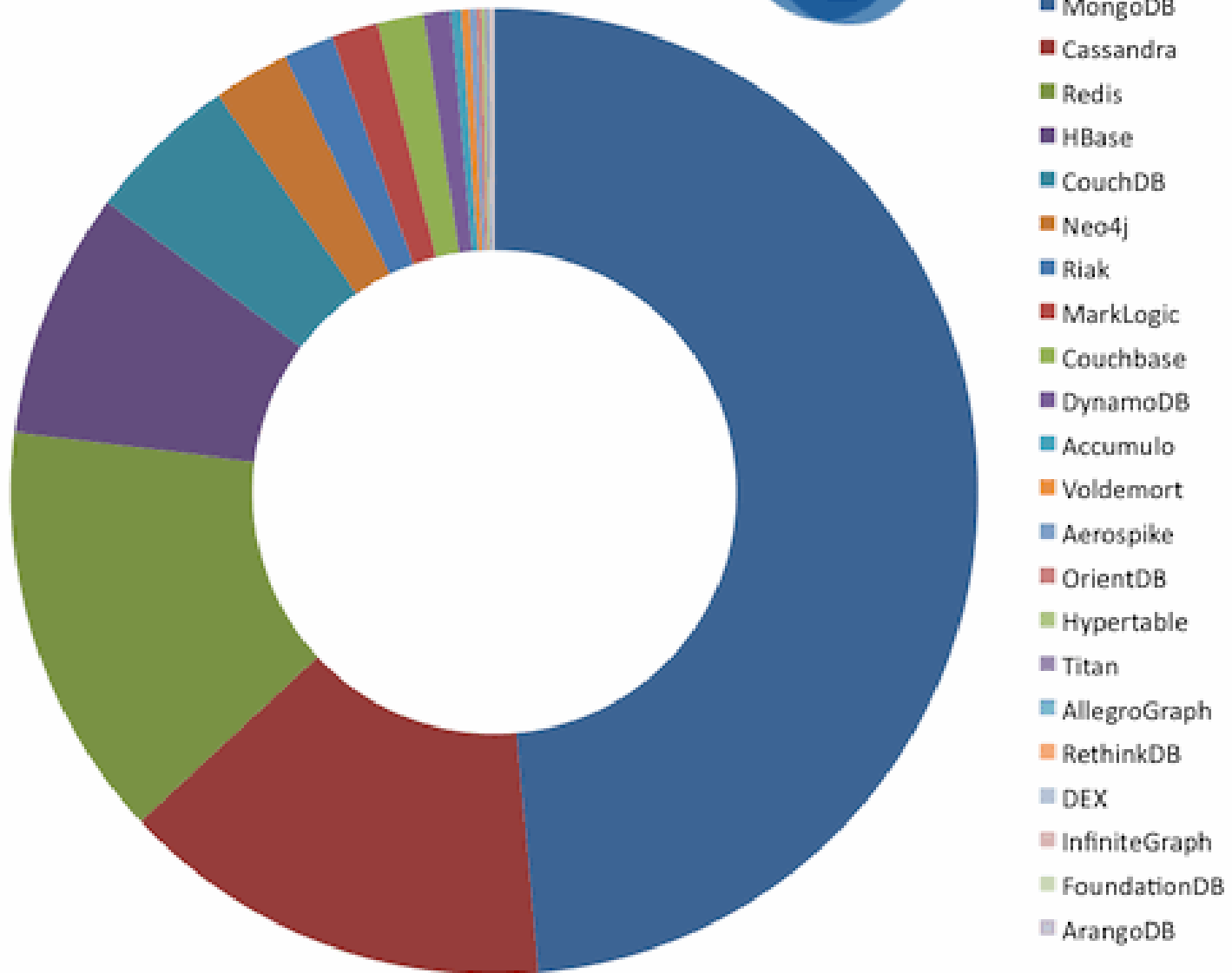
Job Trends from Indeed.com



Relative adoption of NoSQL skills - LinkedIn member search Sept 2013

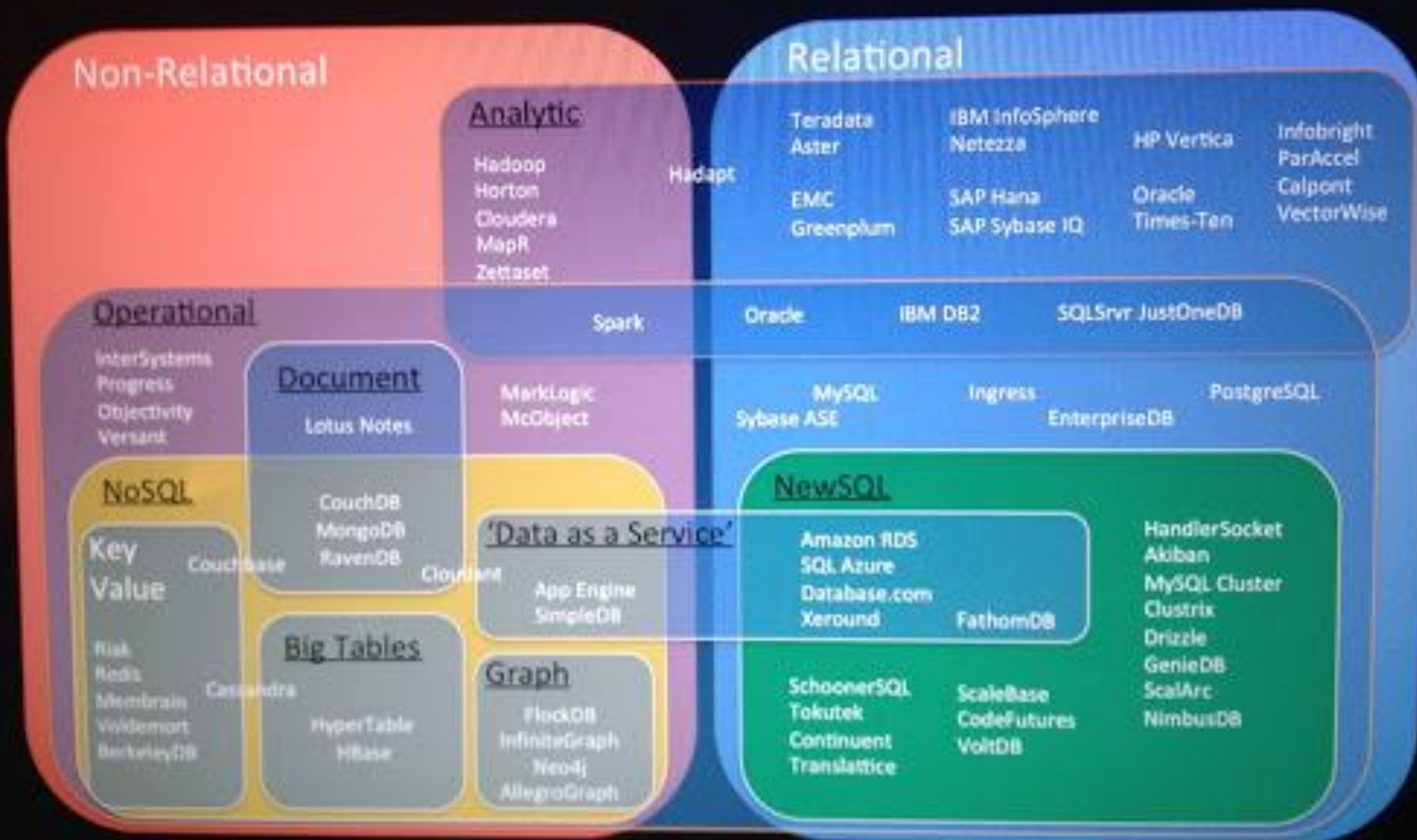
451

Research



Problem

One Size Does Not Fit All



Visual Guide to NoSQL Systems

Availability:
Each client can
always read
and write.

A

Data Models

Relational (comparison)
Key-Value
Column-Oriented/Tabular
Document-Oriented

CA

RDBMSs
(MySQL,
Postgres,
etc)

Aster Data
Greenplum
Vertica

AP

Dynamo
Voldemort
Tokyo Cabinet
KAI

Cassandra
SimpleDB
CouchDB
Riak

Pick Two

<http://blog.beany.co.kr/archives/275>

C

Consistency:
All clients always
have the same view
of the data.

CP

BigTable
Hypertable
Hbase

MongoDB
Terrastore
Scalaris

Berkeley DB
MemcacheDB
Redis

P

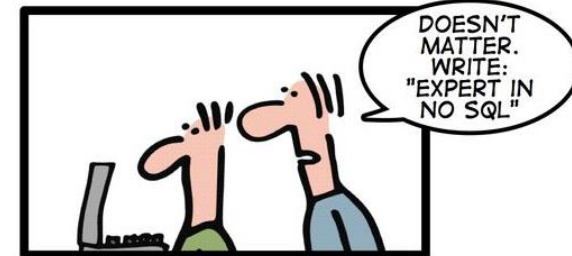
Partition Tolerance:
The system works
well despite physical
network partitions.

What's in a name?

- #nosql
- NoSQL:
 - Never SQL?
 - Not SQL?
 - No to SQL

Not
Only **SQL**

HOW TO WRITE A CV



Leverage the NoSQL boom

<http://geekandpoke.typepad.com/geekandpoke/2011/01/nosql.html>

The revolution will be polygamous

- **Polygot programming**, Neal Ford, 2006
 - "It's all about choosing **the right tool for the job** and leveraging it correctly...The times of writing an application in a single general purpose language is over."
- **Polygot persistence**, Martin Fowler, 2011
 - "any decent sized enterprise will have a **variety of different data storage technologies for different kinds of data**. There will still be large amounts of it managed in relational stores, but increasingly we'll be first asking how we want to manipulate the data and only then figuring out what technology is the best bet for it."



<http://martinfowler.com/bliki/PolygotPersistence.html>

What defines it?

- NoSQL characteristics:
 - Non-relational
 - Schema-less
 - Store whatever structure you like
 - Change it when you want
 - Cluster friendly
 - Parallelizable on clusters of commodity hardware
 - Enable web apps at massive scale
 - Open source (typically)
 - Variety of types / data models
 - No standard like with SQL

NoSQL advantages

Scalability of NoSQL Database vs Traditional Relational Database

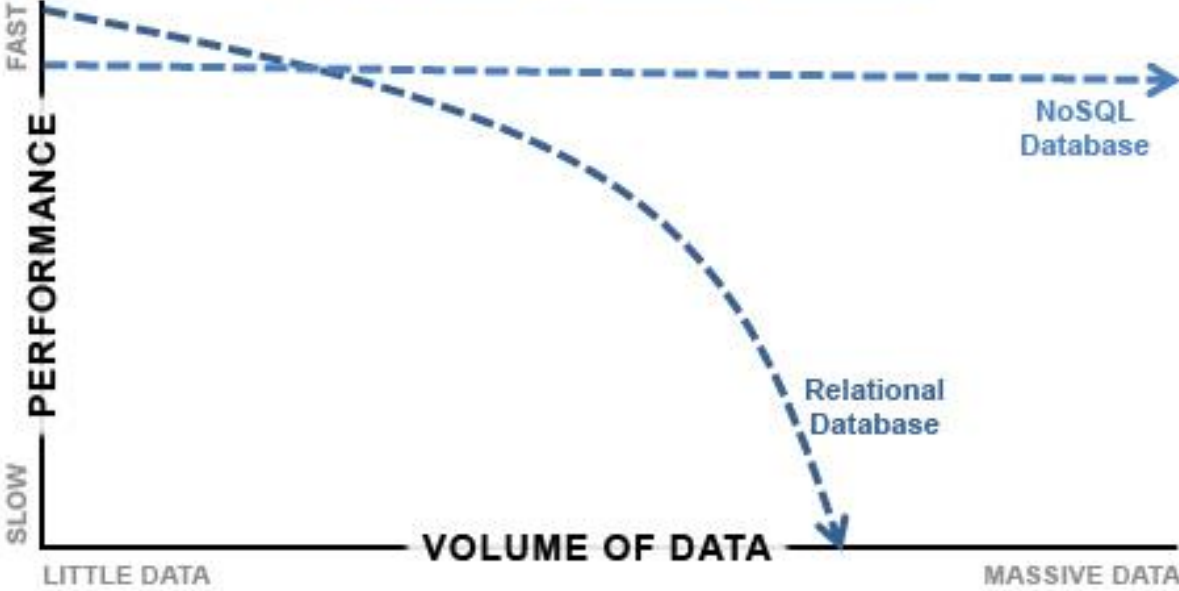


Image Credit: DataJobs.com

Horizontal scalability

Big data



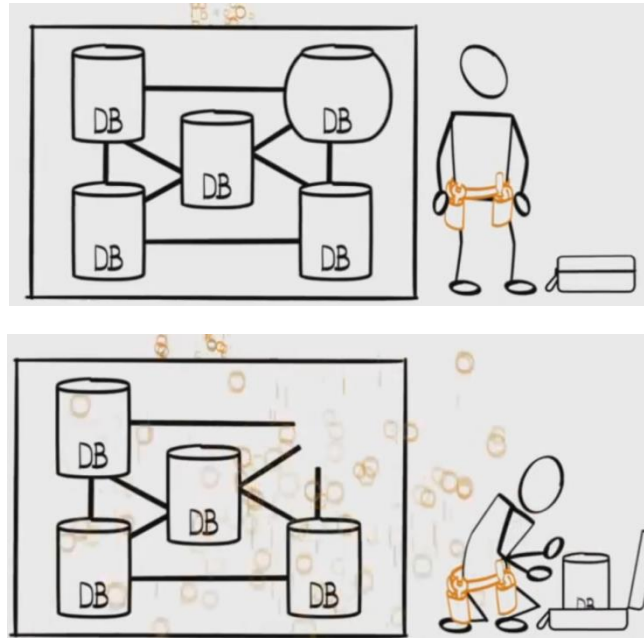
Cheaper

Availability



NoSQL advantages

**Goodbye
highly-trained
DBAs**



<https://www.youtube.com/watch?v=oz-7wJJ9HZ0>

Easier development:
malleable models
storing aggregates



NoSQL disadvantages

- **Maturity**
 - Don't have 20 years of experience as with relational DBs
- **Support**
 - Open source
- **Analytics, business intelligence**
 - Ad hoc queries require programming
- **Administration**
 - Takes skill to install and maintain (new form of DBAs?)
- **Developer expertise**
 - RDBMS expertise is standard with developers
 - Developers still learning NoSQL
 - Less consistent: many different data models and variants

How is data structured?

Key-value



Document



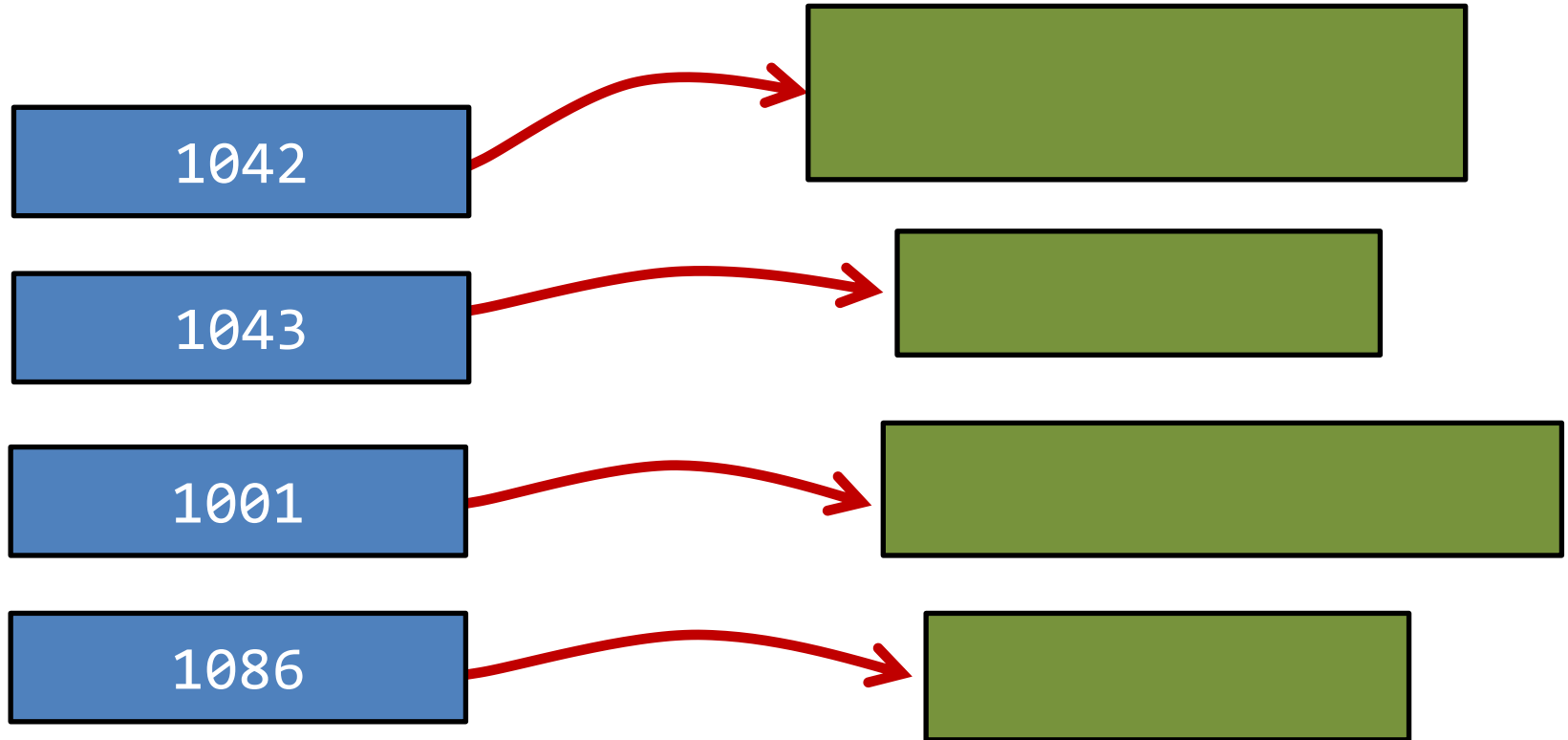
Column



Graph



FlockDB



Key

Value

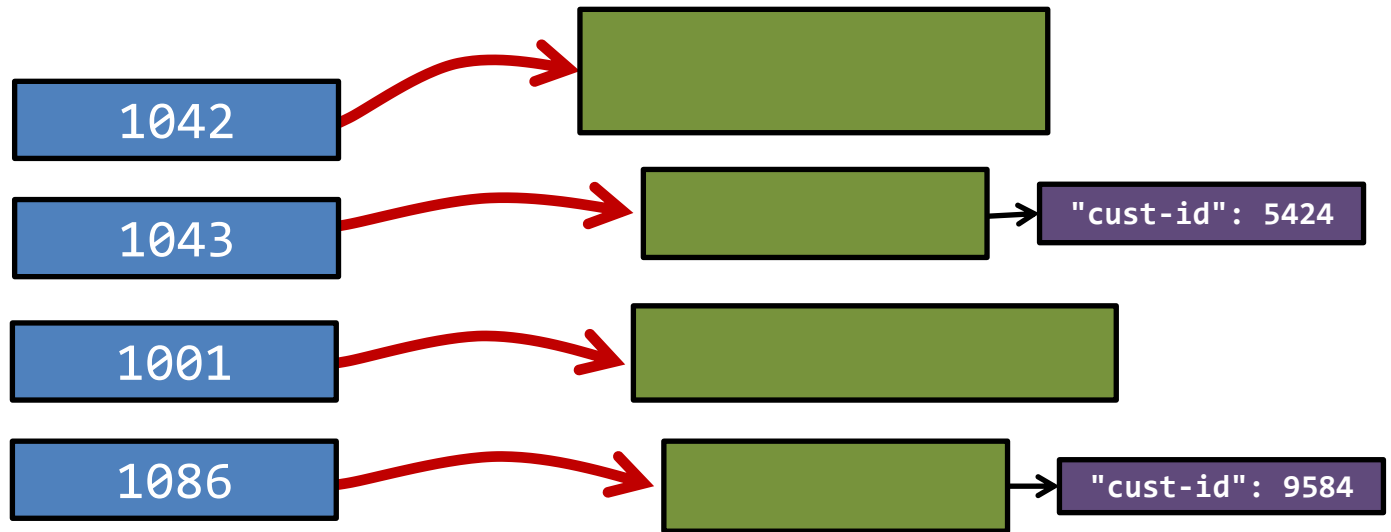
Opaque to DB: could be number, document, image, ...

A hash map that persists to disk

```
{ "id"       : 1001,  
  "cust-id" : 9584,  
  "line-items" : [  
    { "product-id": 5489, "quantity": 1 },  
    { "product-id": 5948, "quantity": 12 }  
  ]  
}
```

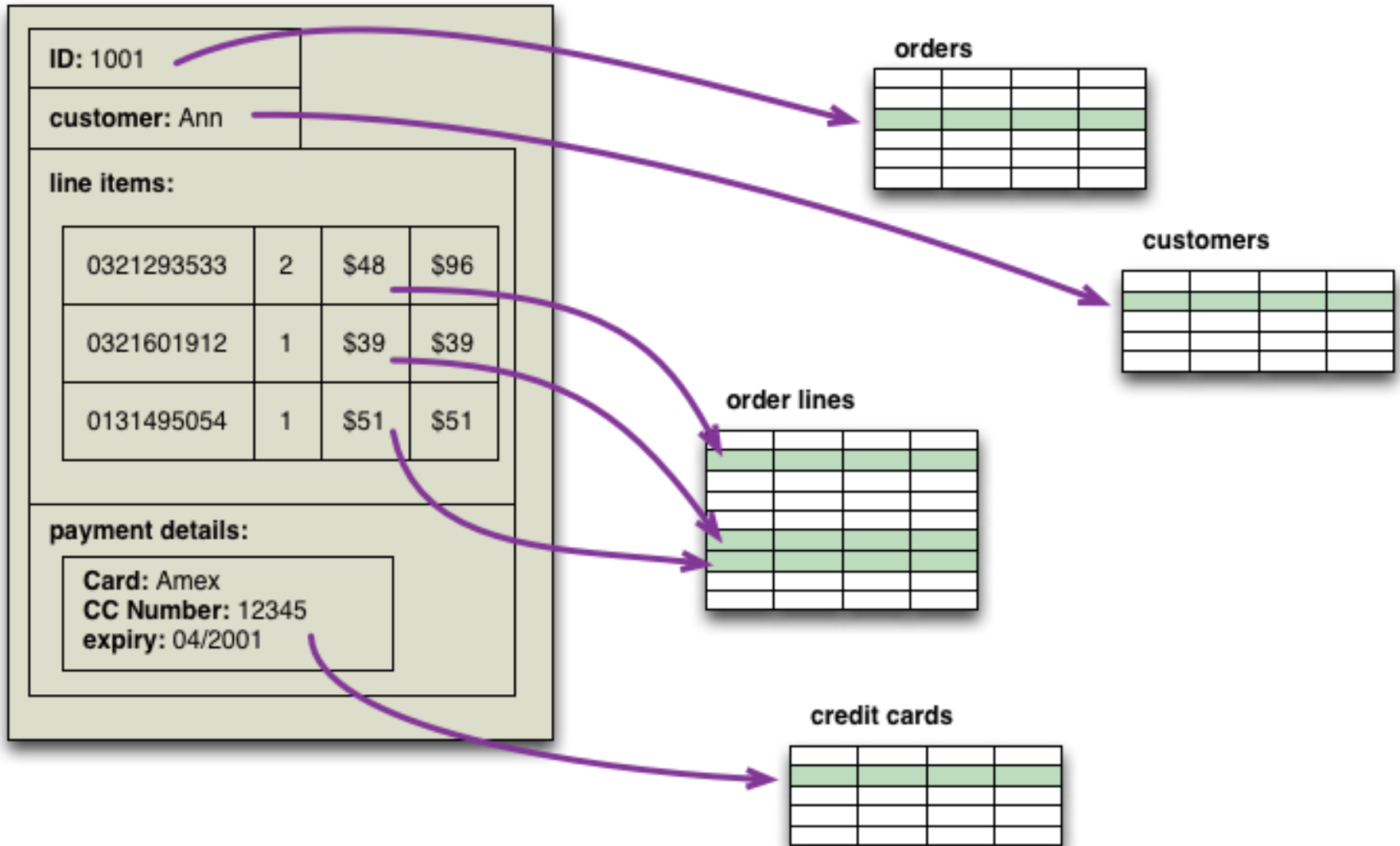
```
{ "id"       : 1002,  
  "cust-id" : 96586,  
  "line-items" : [  
    { "product-id": 8965, "quantity": 2,  
      "color": "Red" }  
  ],  
  "last-order" : "2014-01-03"  
}
```

No explicit schema



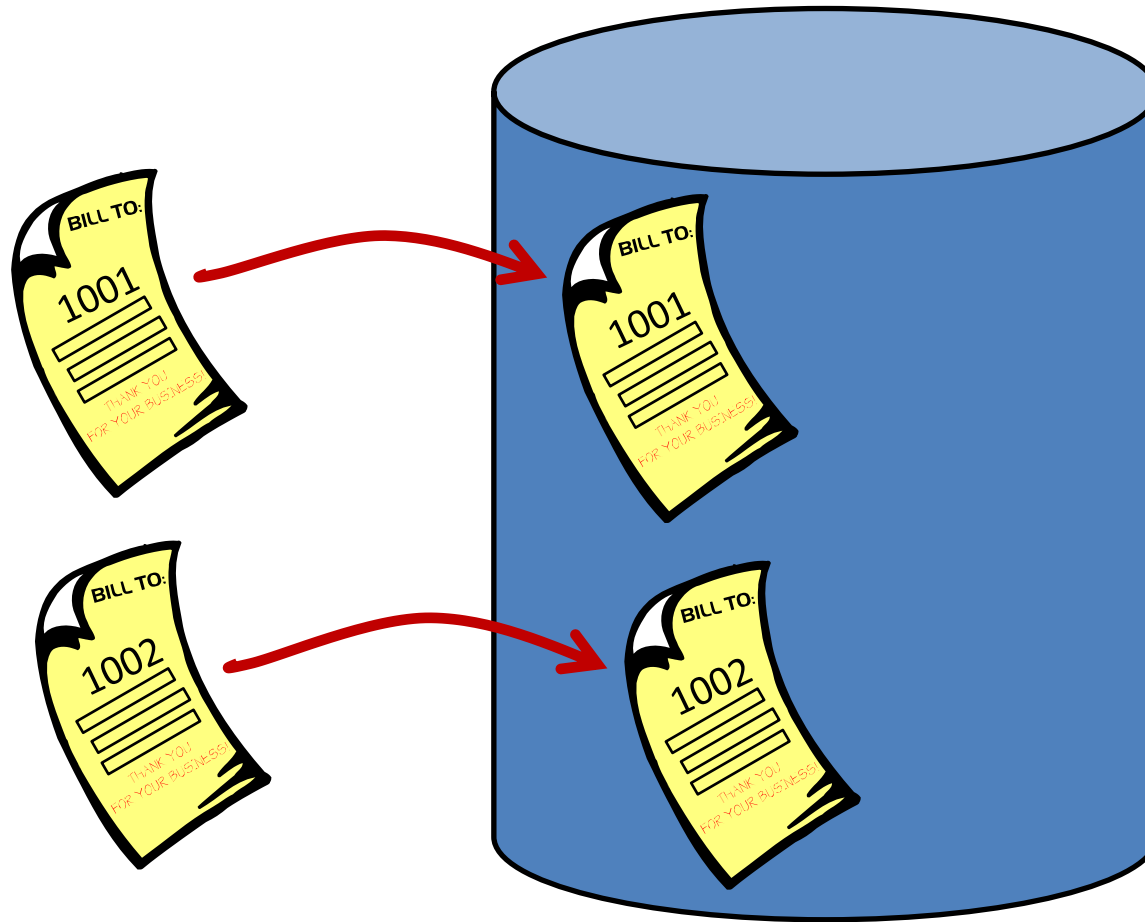
```
{  
  "id" : 1001,  
  "cust-id" : 9584,  
  "line-items" : [  
    {"product-id": 5489, "quantity": 1},  
    {"product-id": 5948, "quantity": 12}  
  ]  
}
```

Aggregates vs. RDMS



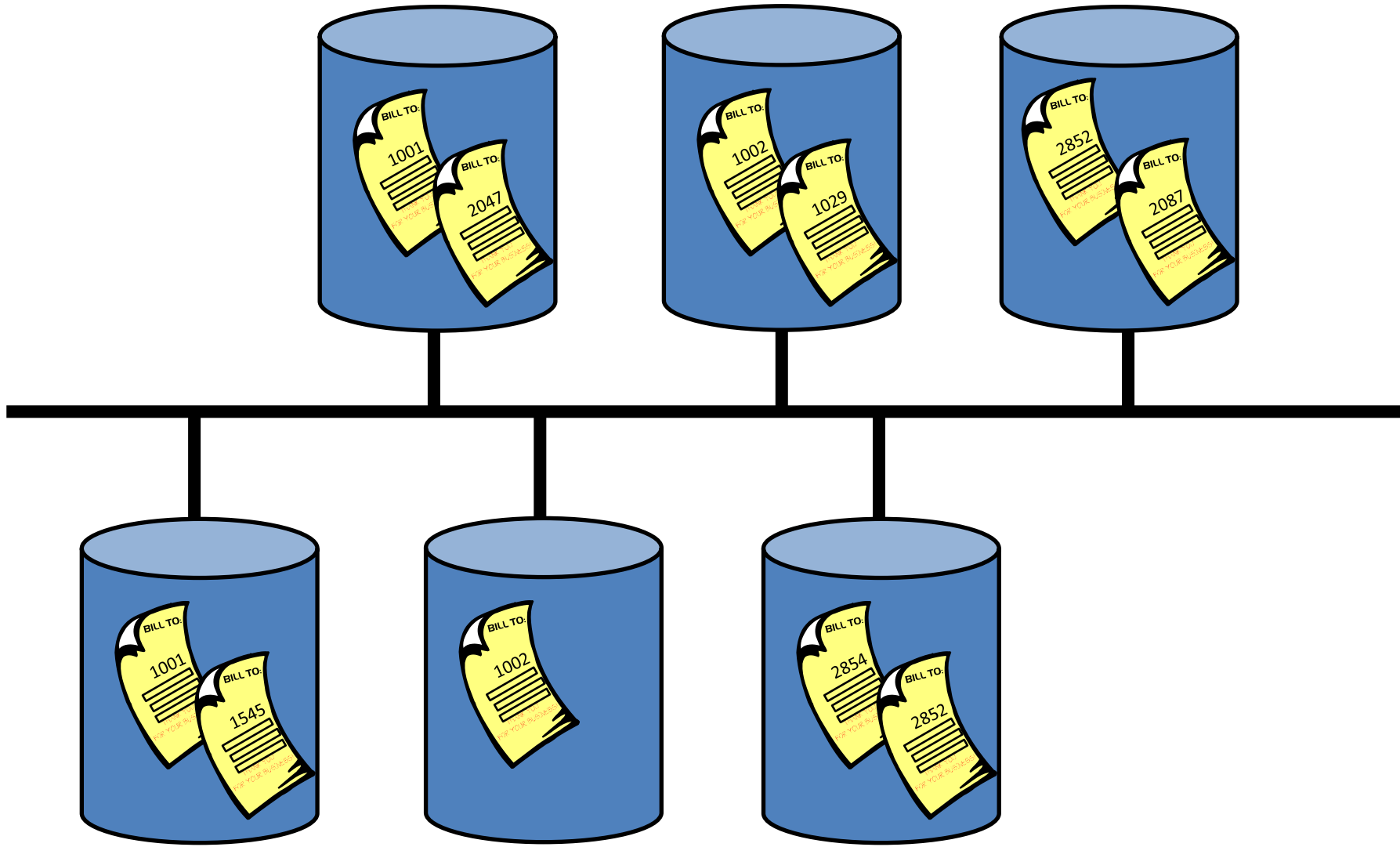
<http://martinfowler.com/bliki/AggregateOrientedDatabase.html>

Aggregates vs. NoSQL

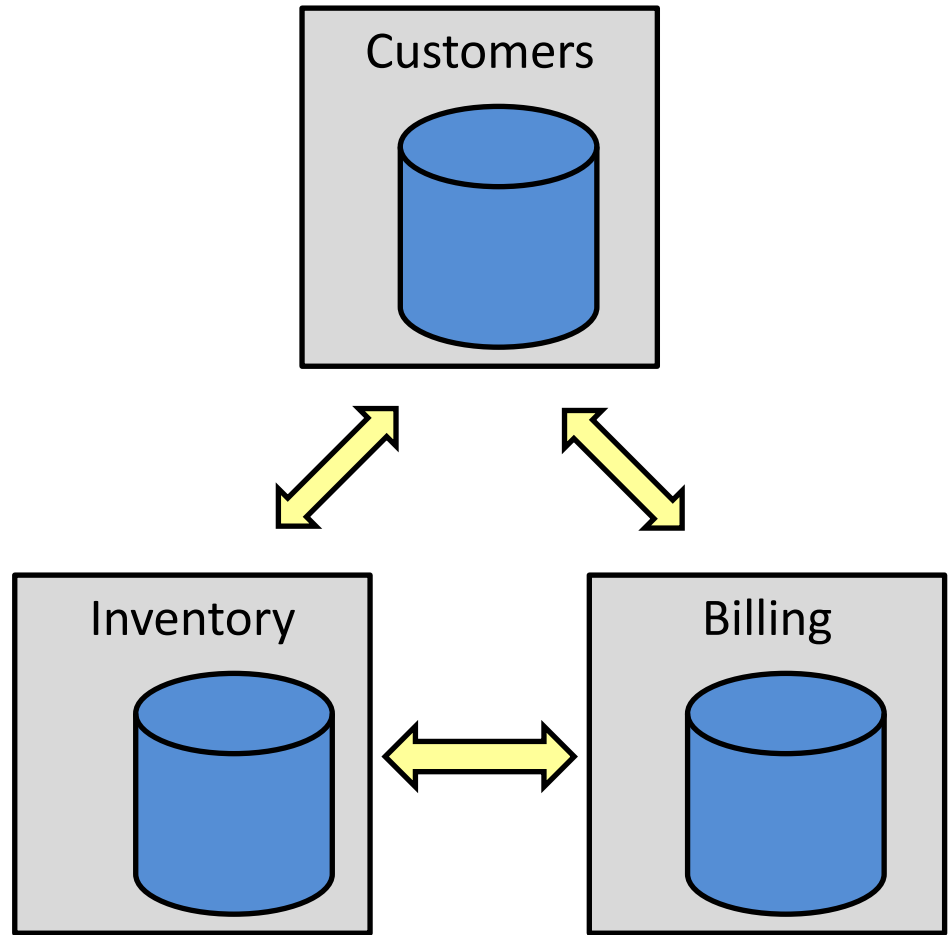
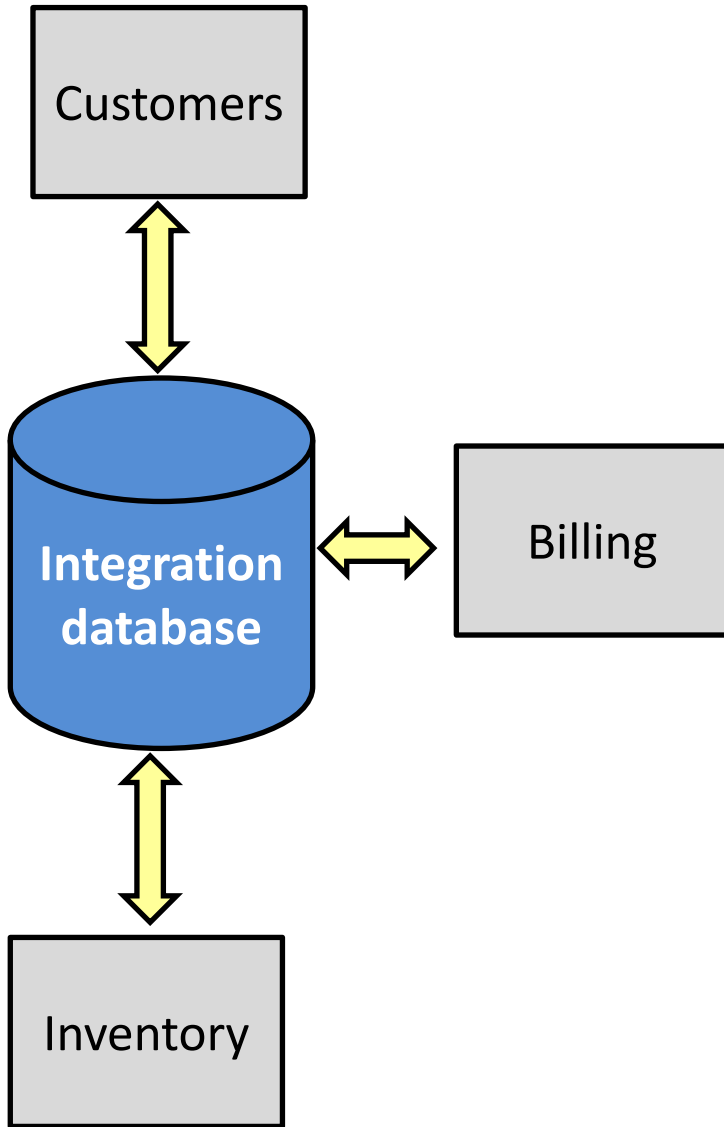


"works really well when **data access is aligned with the aggregates**, but what if you want to look at the data in a different way? Order entry naturally stores orders as aggregates, but analyzing product sales cuts across the aggregate structure." -Martin Fowler

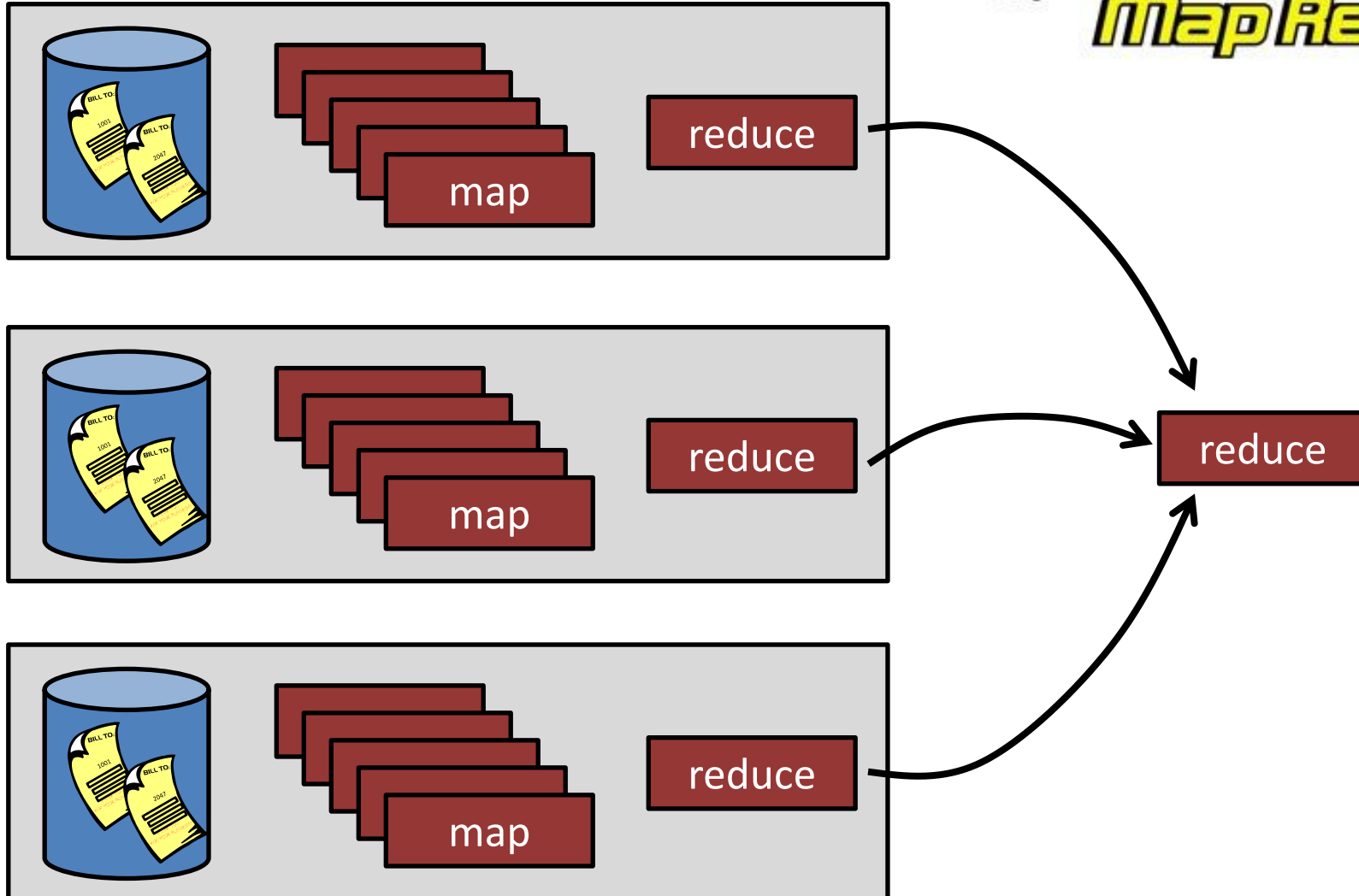
Aggregated-oriented DB: good for clusters



Changing architecture



Changing computation



Map reduce: programming model

- Input and output: set of key/value pairs
- Need to specify two functions:

`map(in_key, in_value) → list(out_key, interm_value)`

- Processes input key/value pair
- Produces set of intermediate pairs

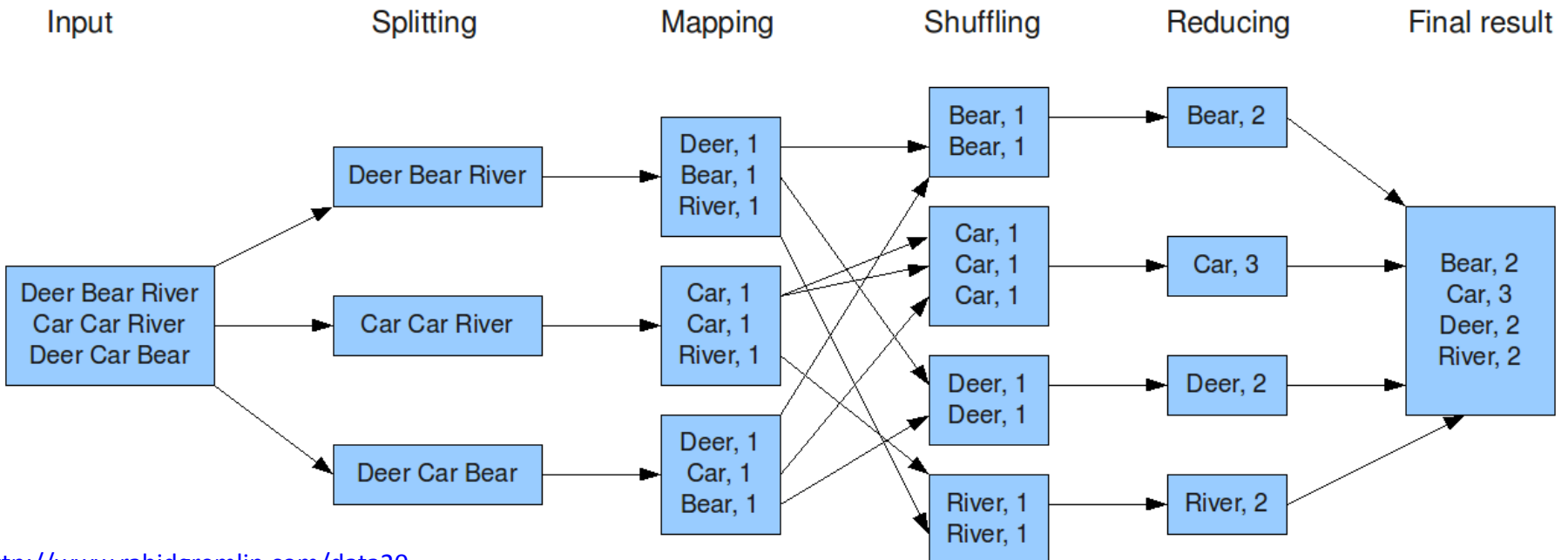
`reduce(out_key, list(interm_value)) → list(out_value)`

- Combine intermediate values for a particular key
- Produce a set of merged output values (usually one)

Map reduce: counting words

```
map(String input_key, String input_value):
  // input_key: document name
  // input_value: document contents
  for each word w in input_value:
    EmitIntermediate(w, "1");

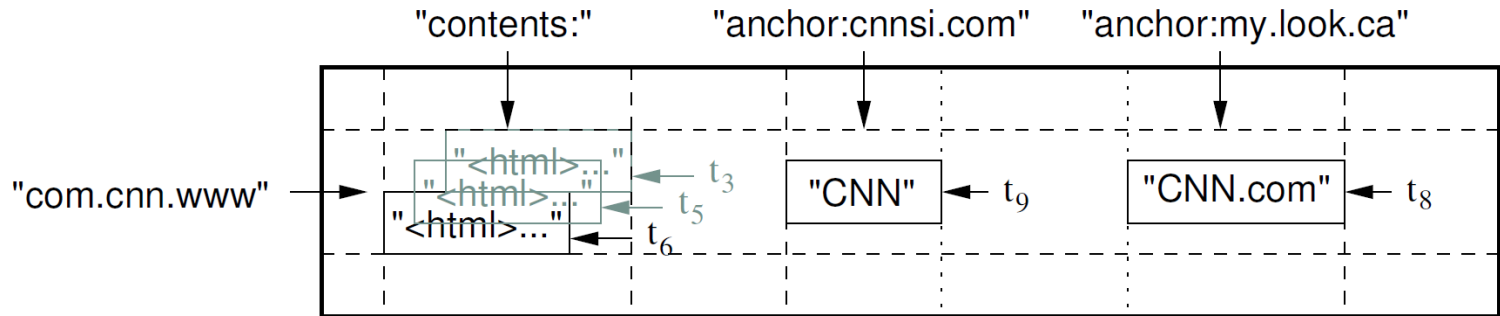
reduce(String output_key, Iterator intermediate_values):
  // output_key: a word
  // output_values: a list of counts
  int result = 0;
  for each v in intermediate_values:
    result += ParseInt(v);
  Emit(AsString(result));
```



Column



"a sparse, distributed, persistent, multi-dimensional, sorted map"



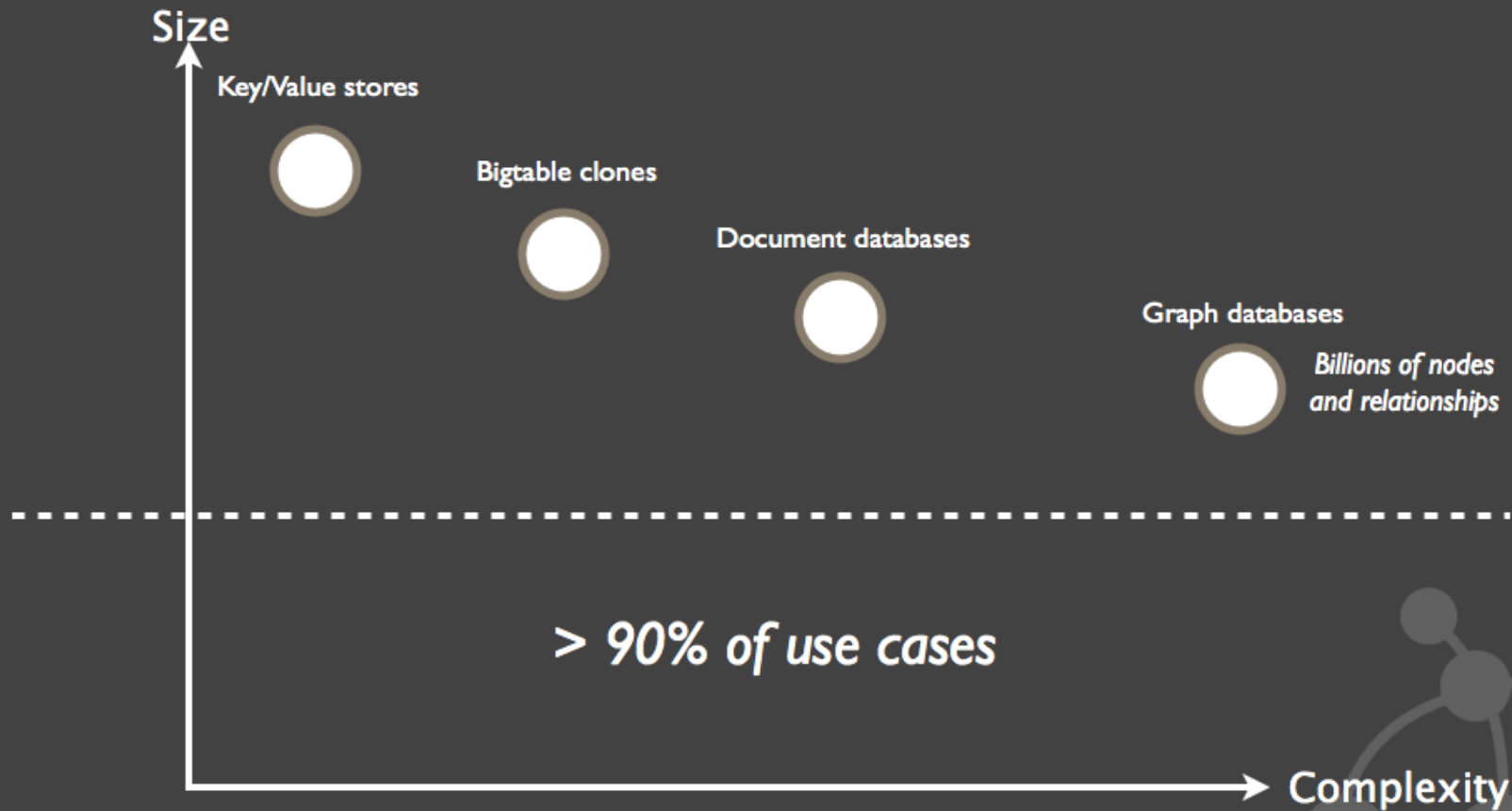
<http://research.google.com/archive/bigtable.html>

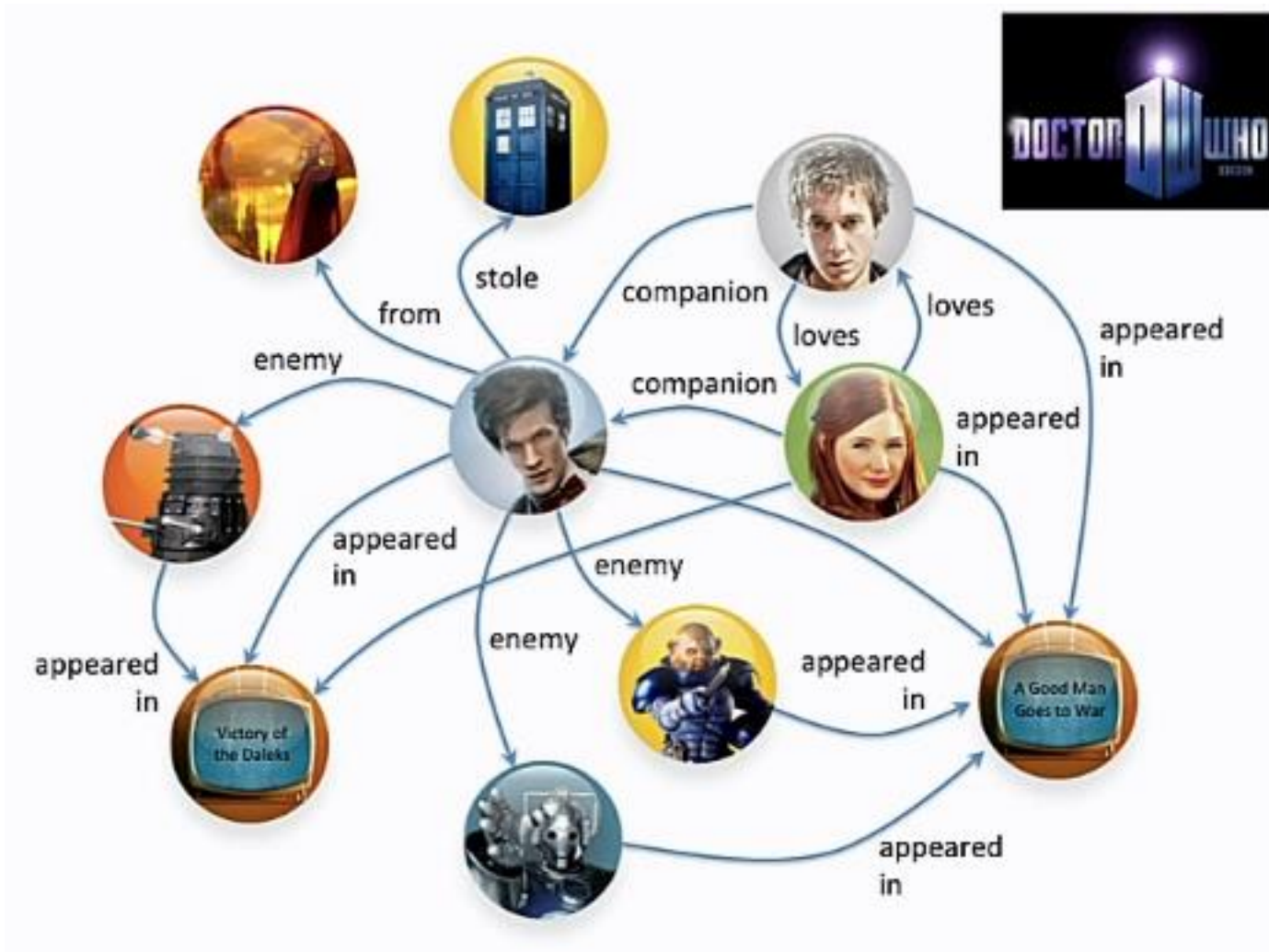
Sorted rows	row keys	column family	column family	column family	
		"language:"	"contents:"	anchor:cnnsi.com	anchor:mylook.ca
	com.aaa	EN	<!DOCTYPE html PUBLIC...		
	com.cnn.www	EN	<!DOCTYPE HTML PUBLIC...	"CNN"	"CNN.com"
	com.cnn.www/TECH	EN	<!DOCTYPE HTML>...		
	com.weather	EN	<!DOCTYPE HTML>...		

<http://www.cs.rutgers.edu/~pxk/417/notes/content/bigtable.html>



Scaling to size vs. Scaling to complexity





<http://www.neo4j.org/training>

Summary

- Relational databases

- Well understood, standard query language: SQL
- Sprays logical unit across many tables

- NoSQL

- Aggregate-oriented, large cohesive chunks
 - Key-value
 - Document
 - Column
- Graph database
 - Lots of small chunks with connections
- Map-reduce
 - Compute efficiently maintaining good data locality