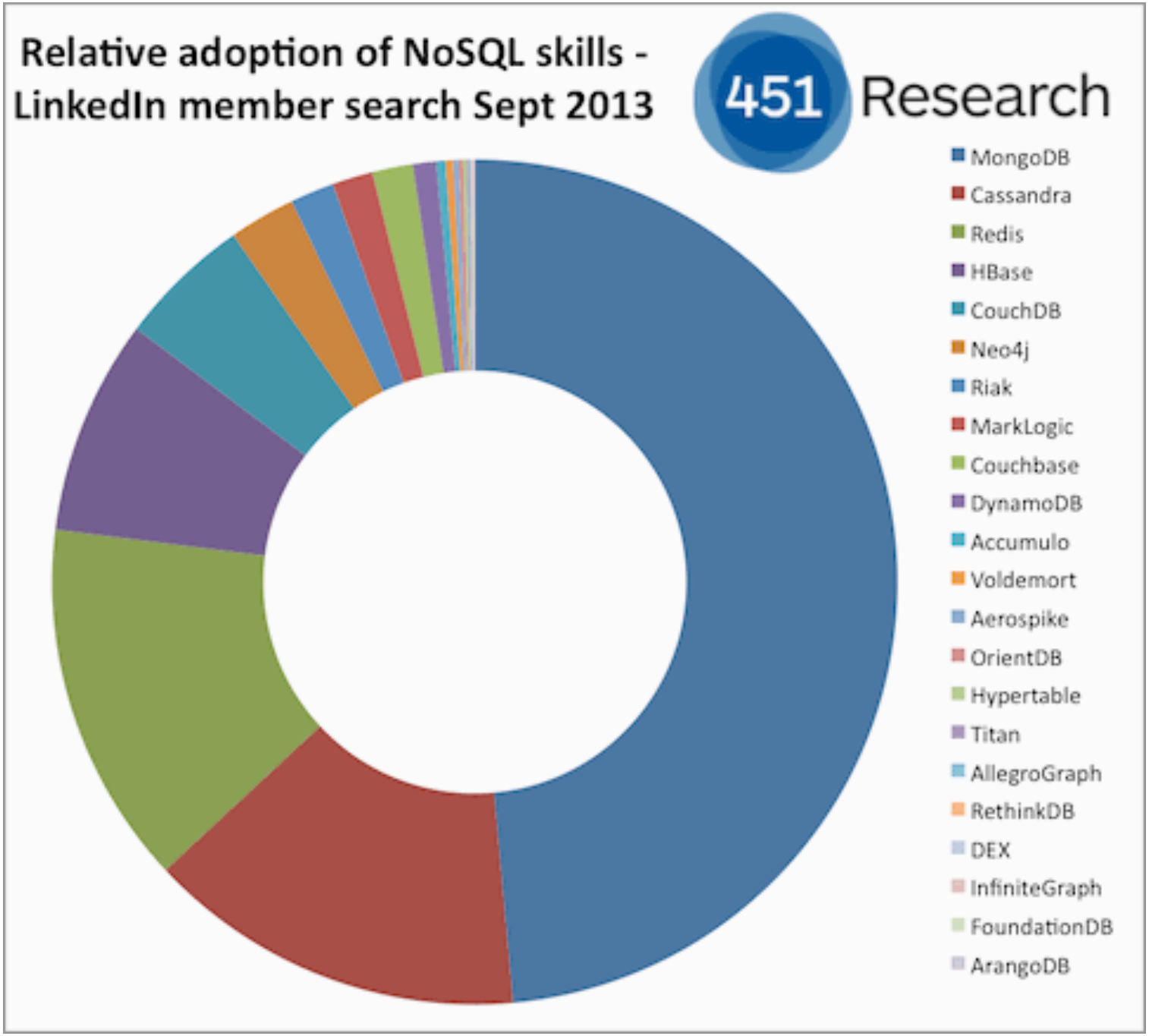
The NoSQL Movement

FlockDB


Job Trends from Indeed.com
Nosql
Percentage of Matching Job Postings
0.15
0.1
0.05
0
Jan '06
Jan '07
Jan '08
Jan '09
Jan '10
Jan '11
Jan '12
Jan '13
Jan '14
Jan '1


Job Trends from Indeed.com
"big data"
Percentage of Matching Job Postings
0.3
0.2
0.1
0
Jan '06
Jan '07
Jan '08
Jan '09
Jan '10
Jan '11
Jan '12
Jan '13
Jan '14
Jan '1

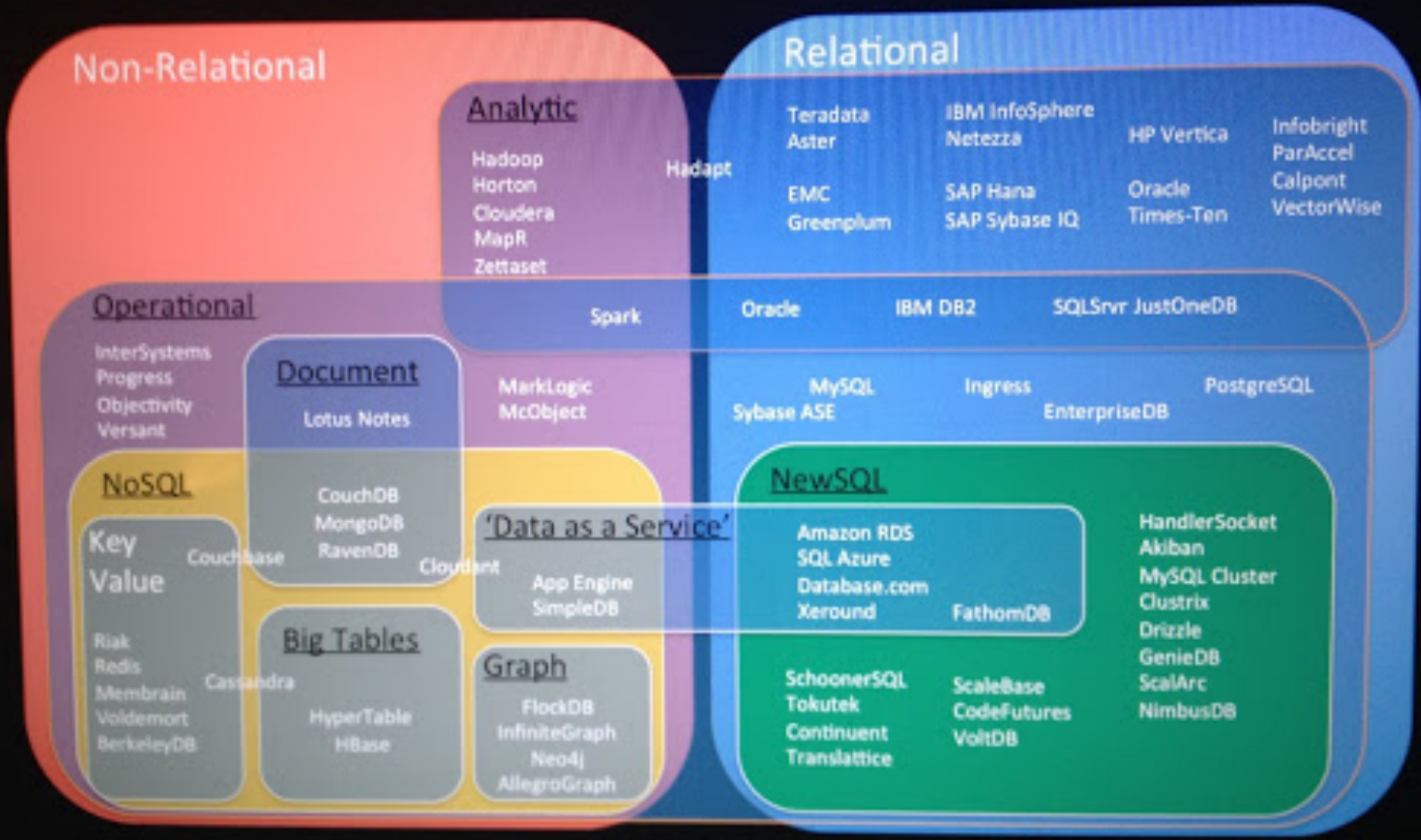

Relative adoption of NoSQL skills - LinkedIn member search Sept 2013
451 Research
MongoDB
Cassandra
Redis
HBase
CouchDB
Neo4j
Riak
MarkLogic
Couchbase
DynamoDB
Accumulo
Voldemort
Aerospike
OrientDB
Hypertable
Titan
AllegroGraph
RethinkDB
DEX
InfiniteGraph
FoundationDB
ArangoDB

http://techcrunch.com/2012/10/27/big-data-right-now-five-trendy-open-source-technologies/

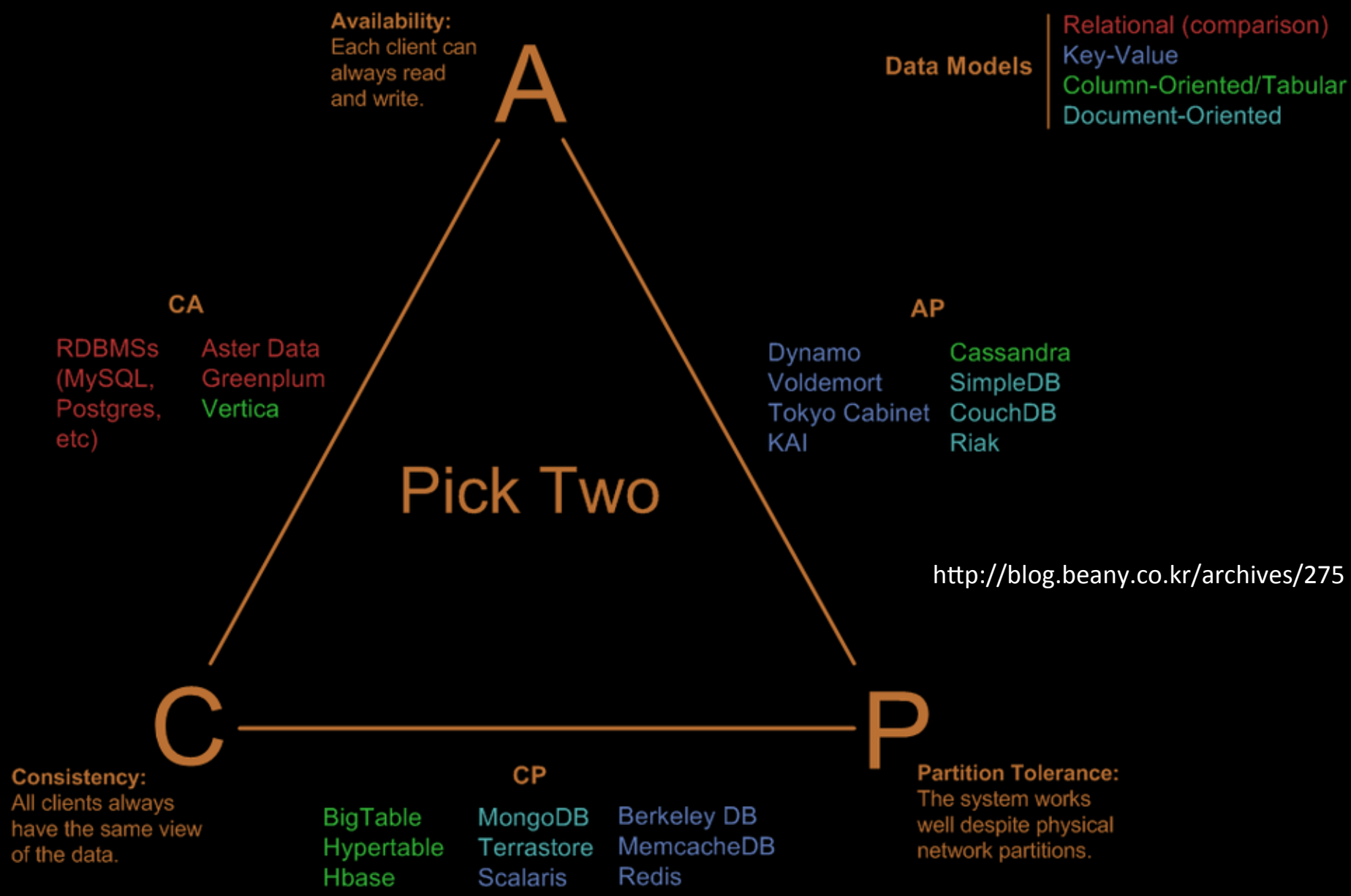
infochin

Problem

One Size Does Not Fit All



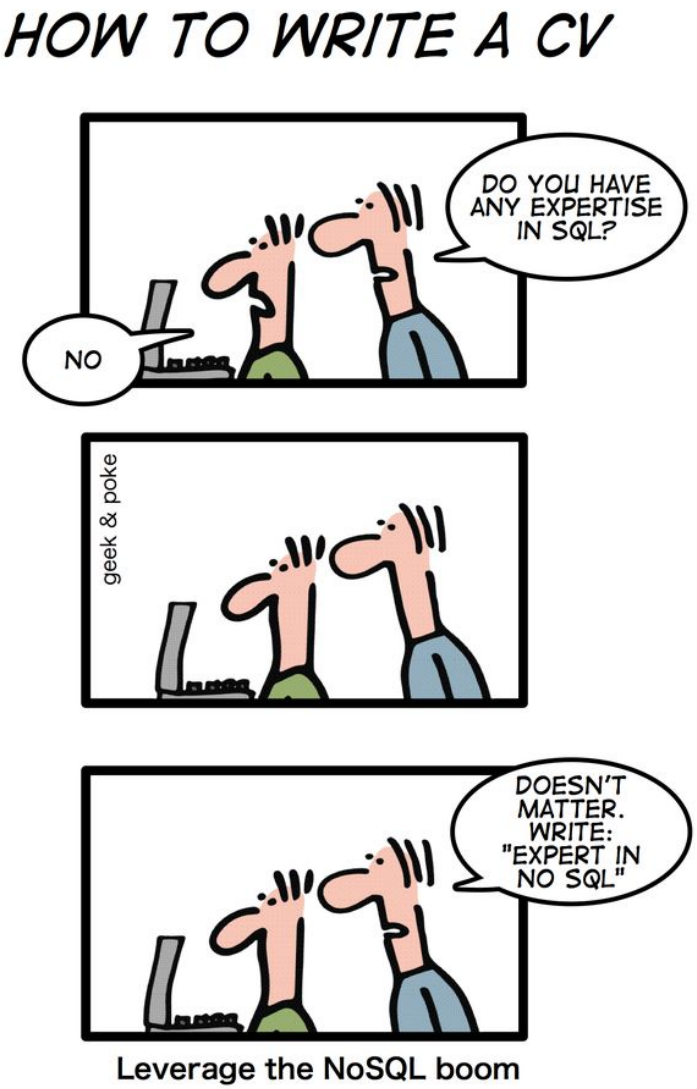
Visual Guide to NoSQL Systems



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

What's in a name?

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



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/PolyglotPersistence.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

Horizontal scalability

Big data

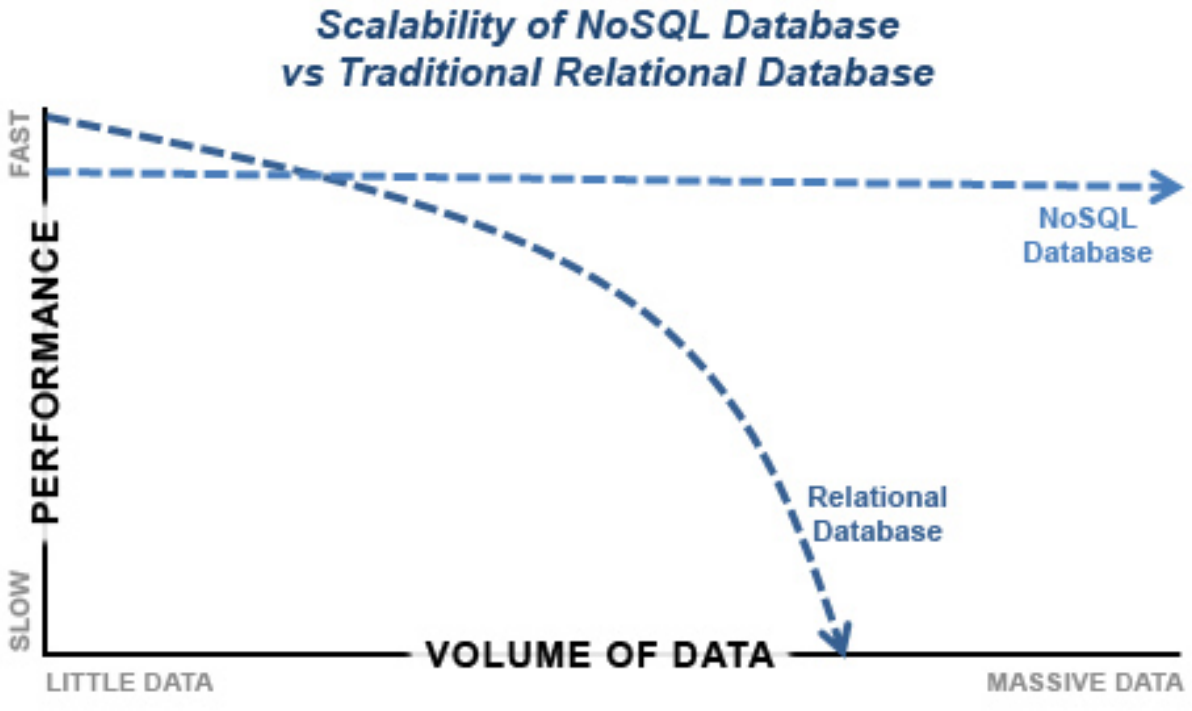
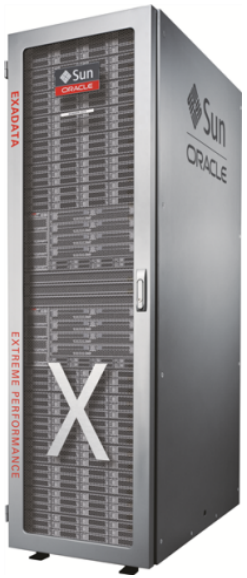


Image Credit: DataJobs.com



Cheaper

Availability

NoSQL advantages

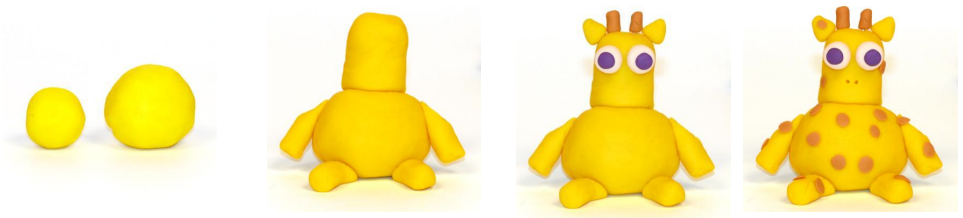
Goodbye highly- trained DBAs

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

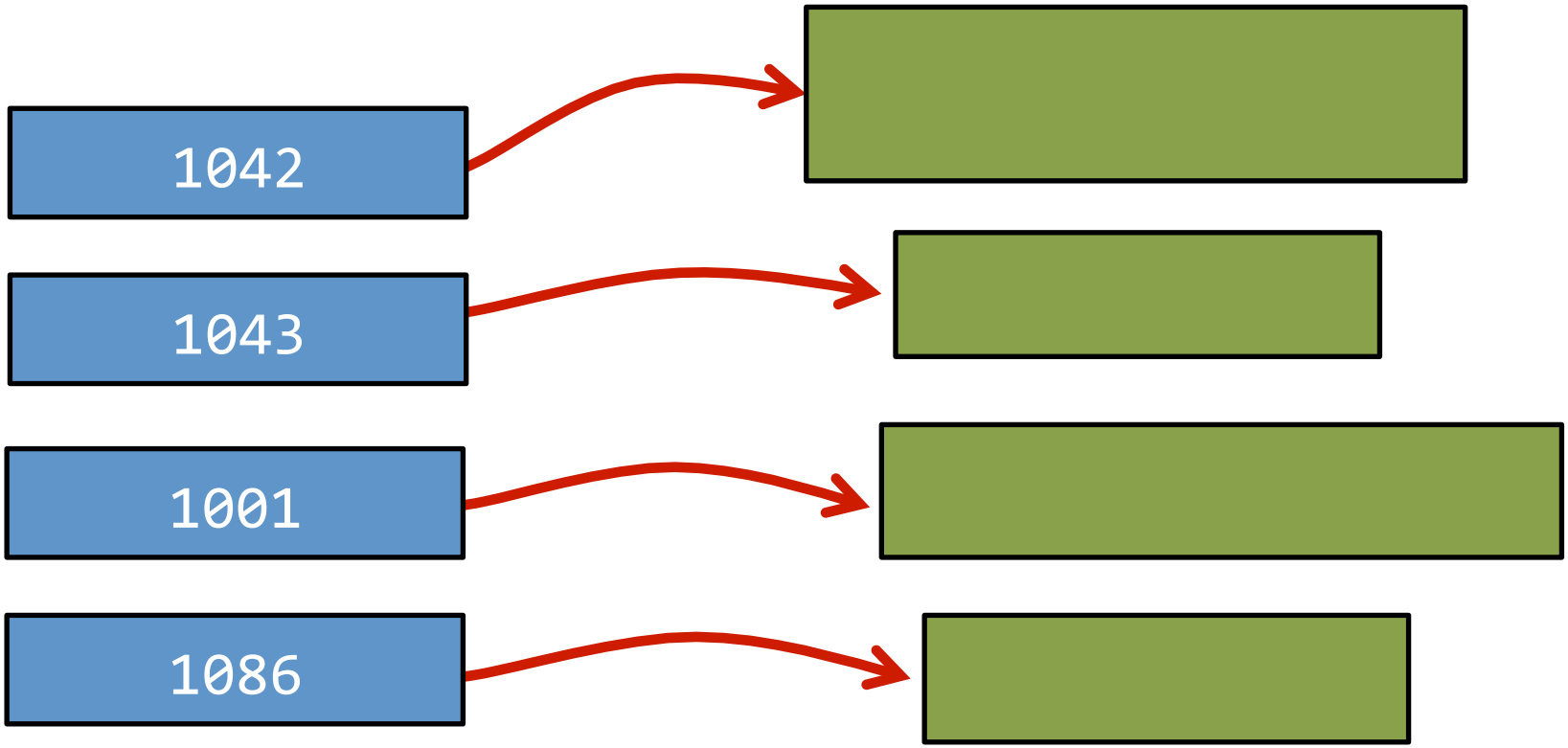
FlockDB

Key-value

amazon DynamoDB

Project Voldemort
A distributed database.

riak



Key

Value

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

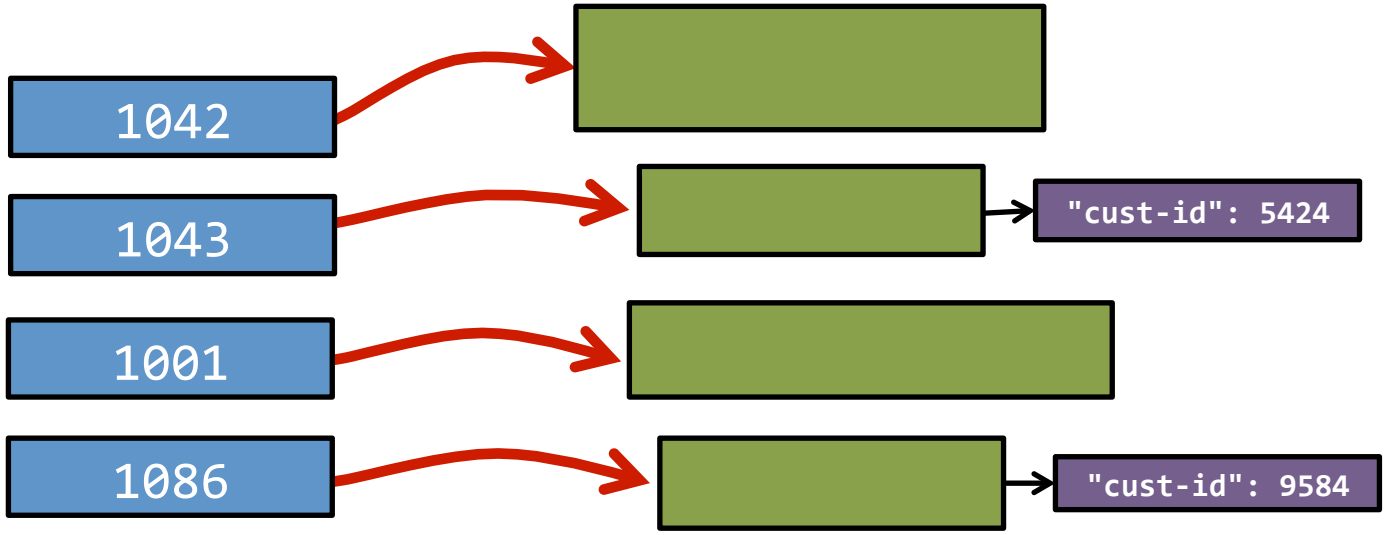
A hash map that persists to disk

Document


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

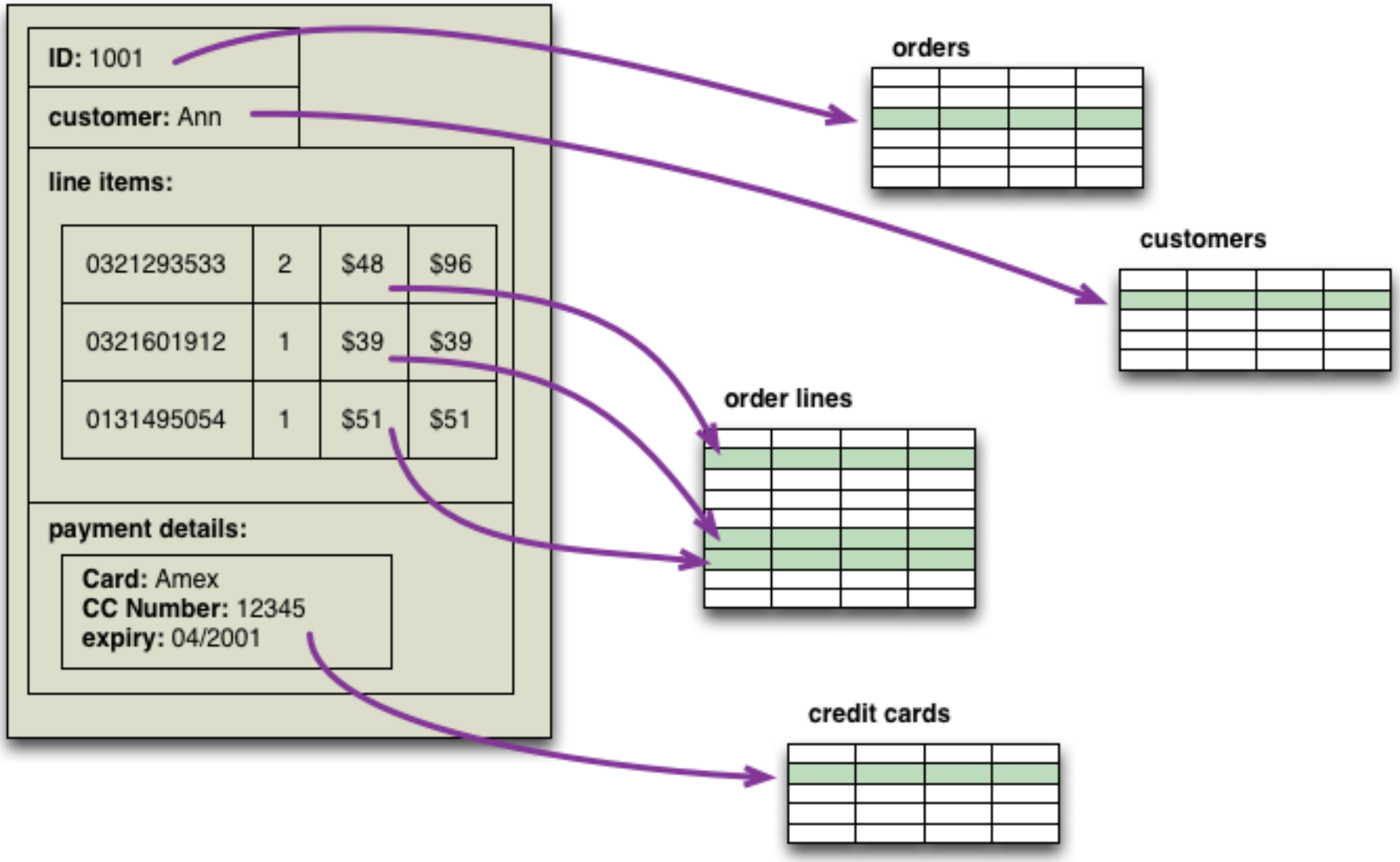
```
{"id"          : 1002,
 "cust-id"     : 96586,
 "line-items" : [
                  {"product-id": 8965, "quantity": 2,
                   "color": "Red"}
                ],
 "last-order" : "2014-01-03"
}
```

No explicit schema



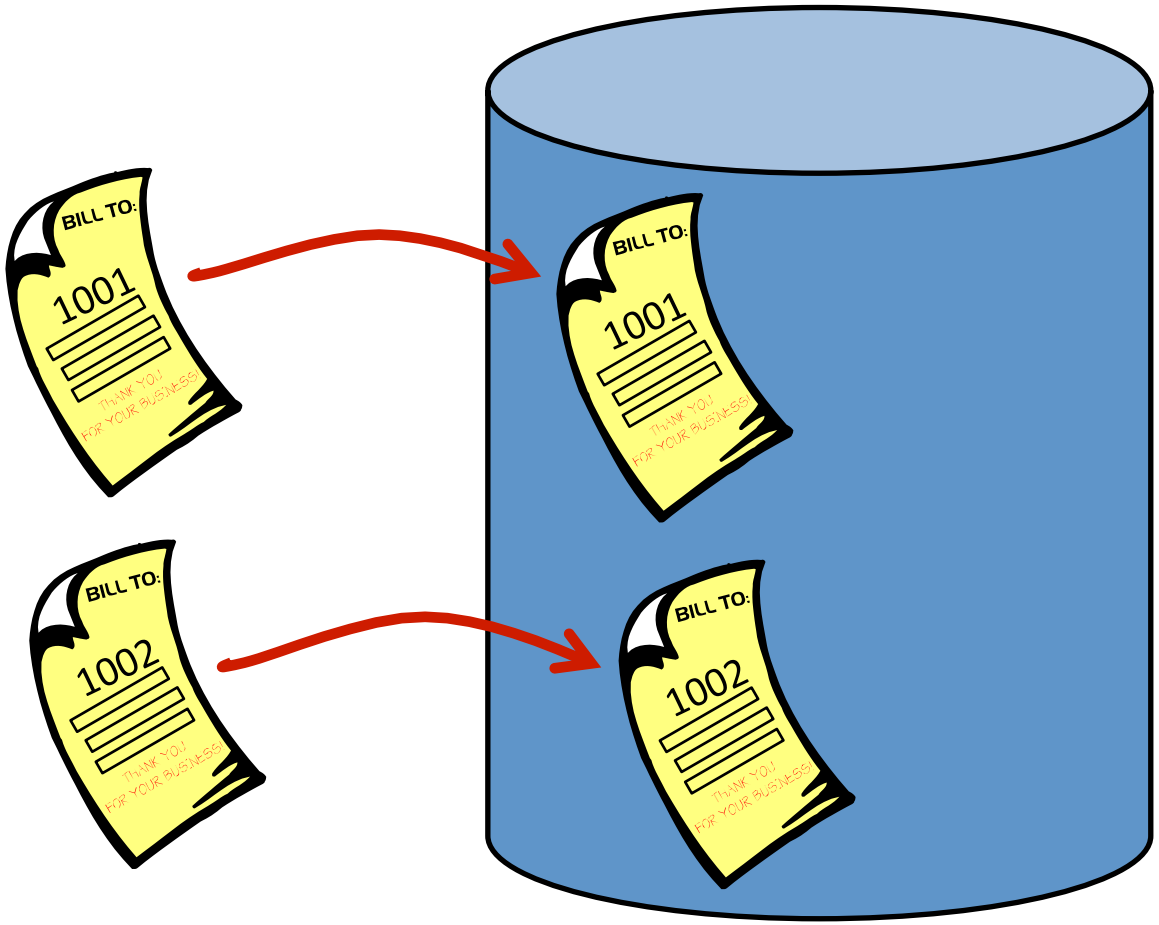

```
{"id"          : 1001,
 "cust-id"     : 9584,
 "line-items" : [
                  {"product-id": 5489, "quantity": 1},
                  {"product-id": 5948, "quantity": 12}
                ]
}
```

Aggregates vs. RDMS



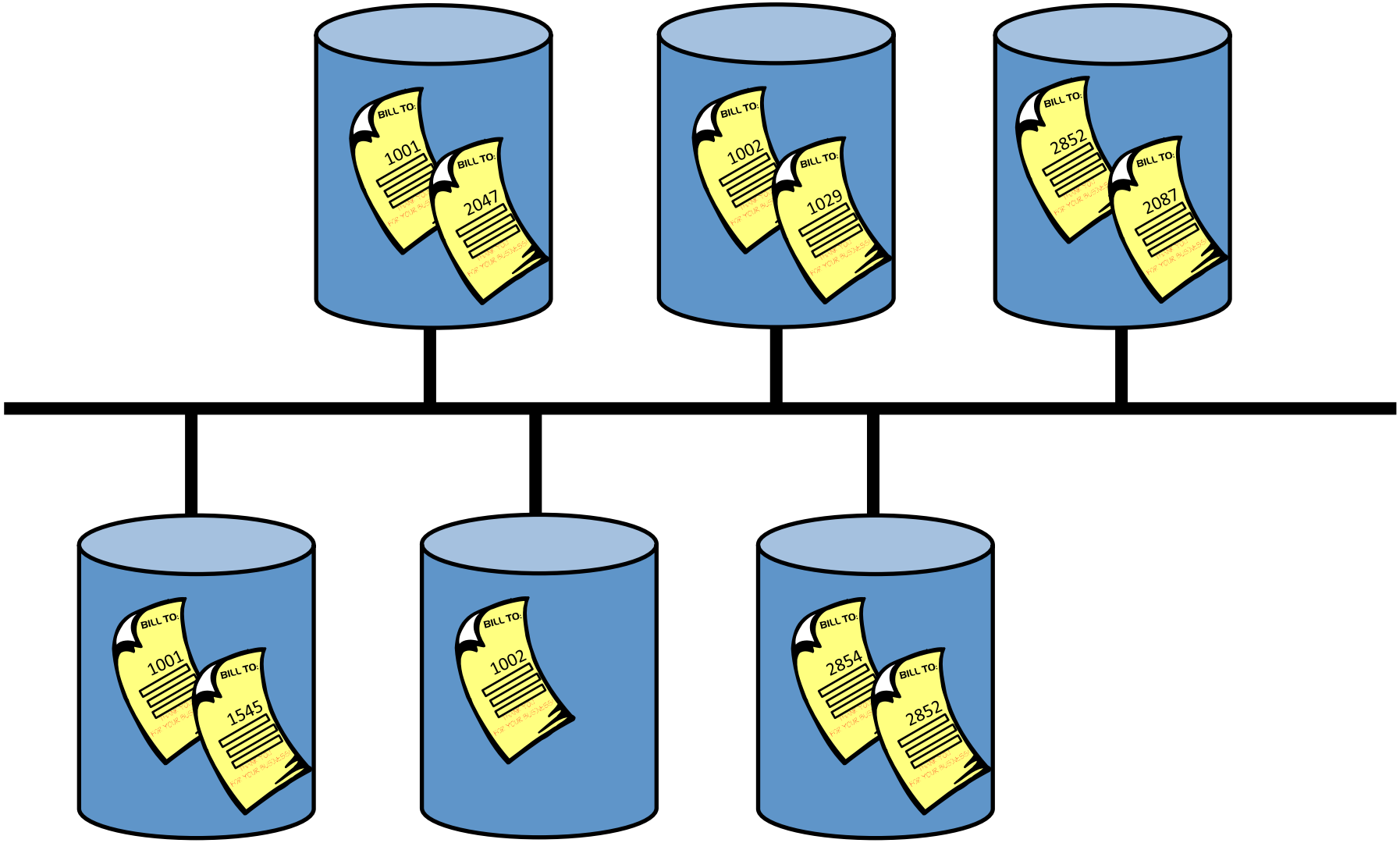
http://martinfowler.com/bliki/AggregateOrientedDatabase.html

Aggregates vs. NoSQL

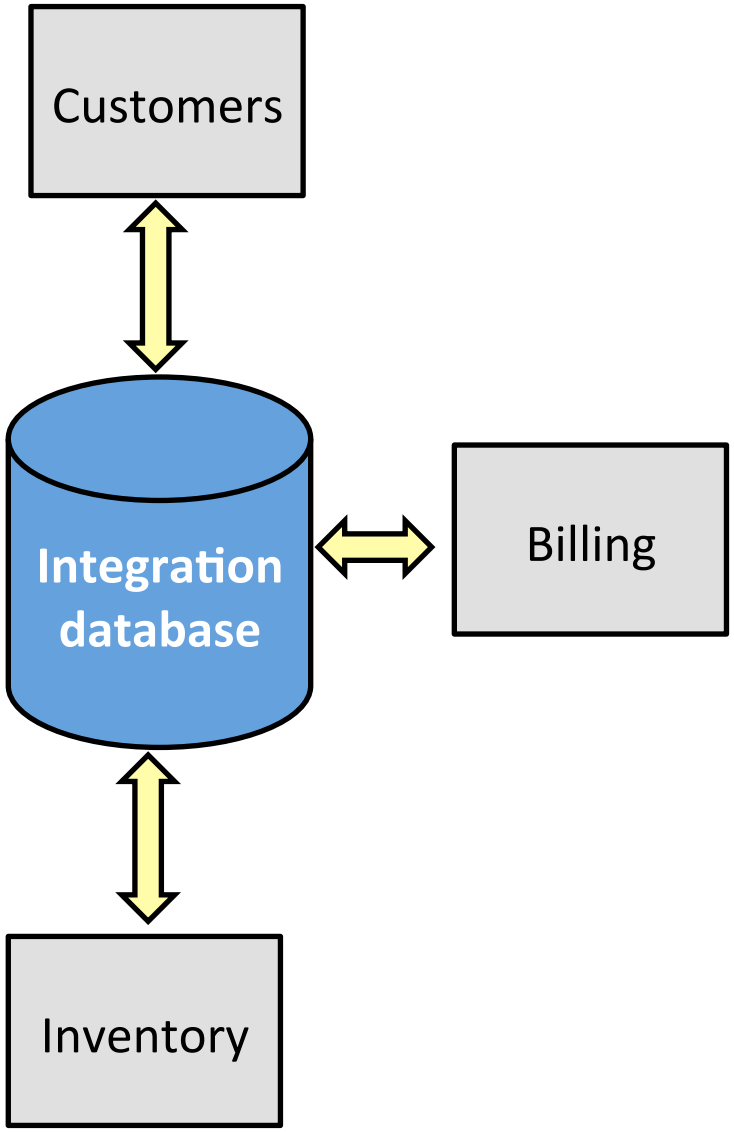


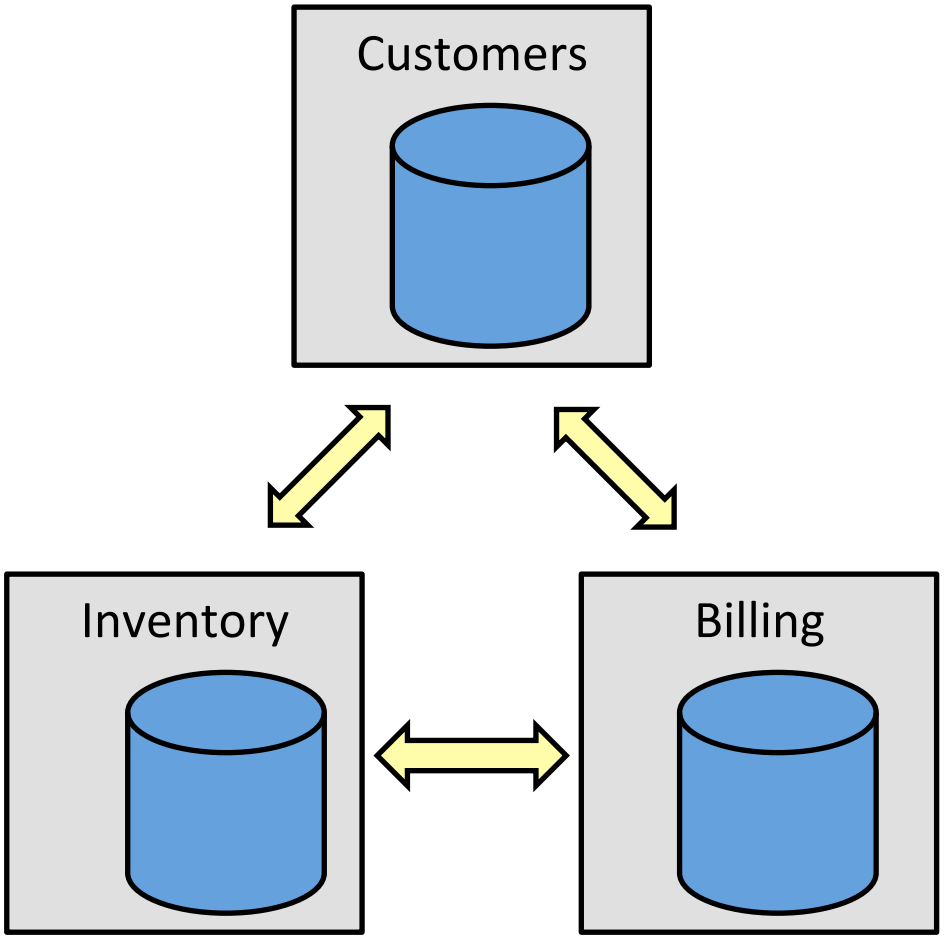
"works really well when data access is aligned with the aggregates, but what if you want to look at the data in a different way? Order entry naturally stores orders as aggregates, but analyzing product sales cuts across the aggregate structure." -Martin Fowler

Aggregated-oriented DB: good for clusters

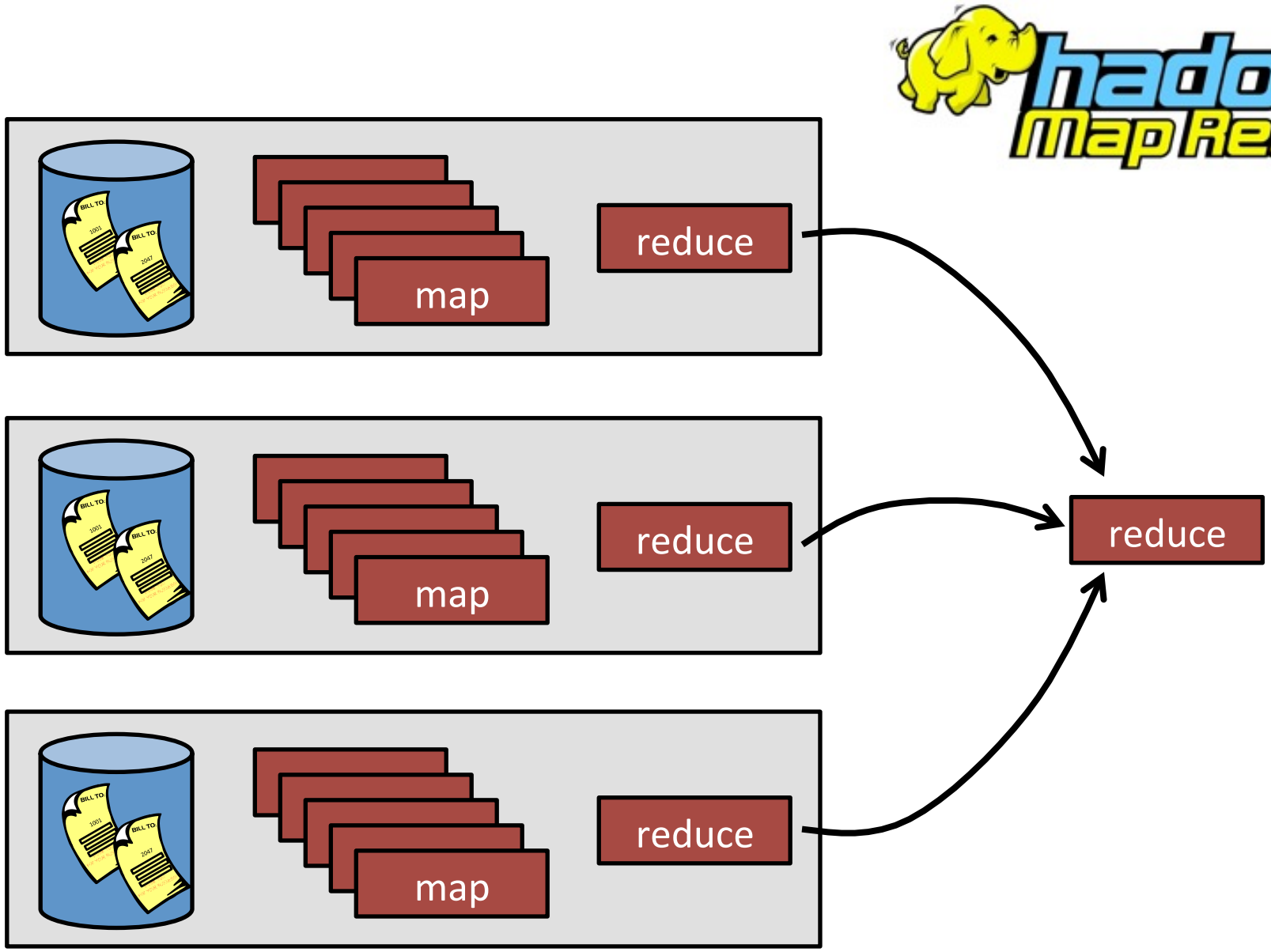


Changing architecture





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

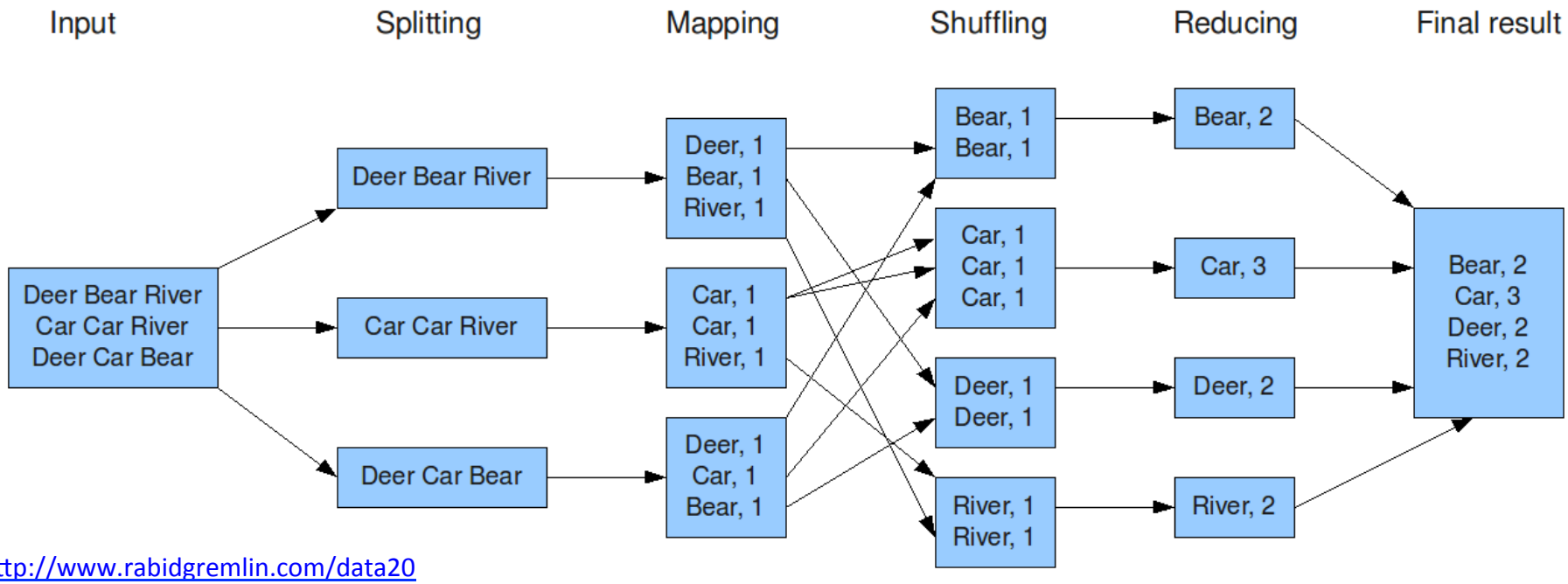
`reduce(out_key, list(interm_value)) → list(out_value)`

  - 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));
```



Map reduce: mutual friends

```
Person → [list of friends]
A → B C D
B → A C D E
C → A B D E
D → A B C E
E → B C D
Each line becomes arg to mapper
```

```
From A → B C D
(A B) → B C D
(A C) → B C D
(A D) → B C D
```

```
From B → A C D E
(A B) → A C D E
(B C) → A C D E
(B D) → A C D E
(B E) → A C D E
```

...

```
Shuffle:
(A B) → (A C D E) (B C D)
(A C) → (A B D E) (B C D)
(A D) → (A B C E) (B C D)
(B C) → (A B D E) (A C D E)
(B D) → (A B C E) (A C D E)
(B E) → (A C D E) (B C D)
(C D) → (A B C E) (A B D E)
(C E) → (A B D E) (B C D)
(D E) → (A B C E) (B C D)
```

```
Reduce:
(A B) → (C D)
(A C) → (B D)
(A D) → (B C)
(B C) → (A D E)
(B D) → (A C E)
(B E) → (C D)
(C D) → (A B E)
(C E) → (B D)
(D E) → (B C)
```

http://stevekrenzel.com/finding-friends-with-mapreduce

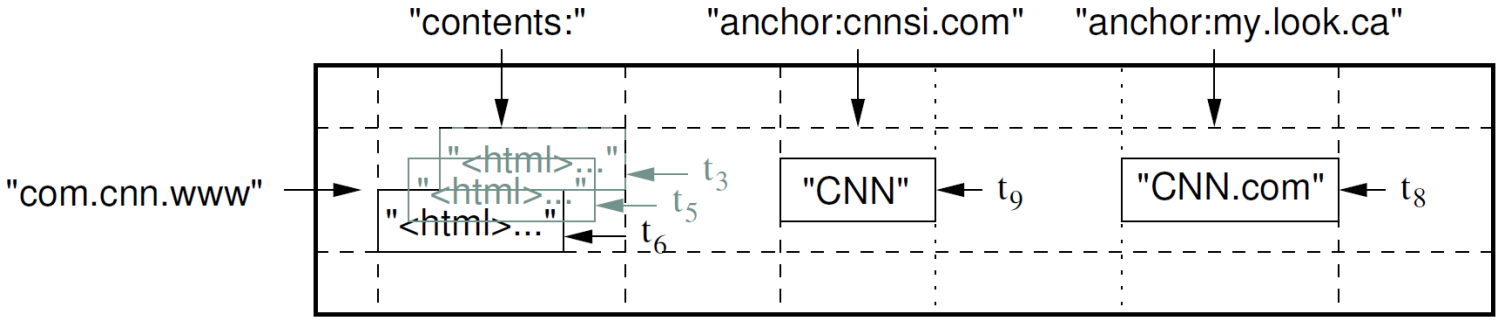
Column

Google BigTable

cassandra

APACHE HBASE

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

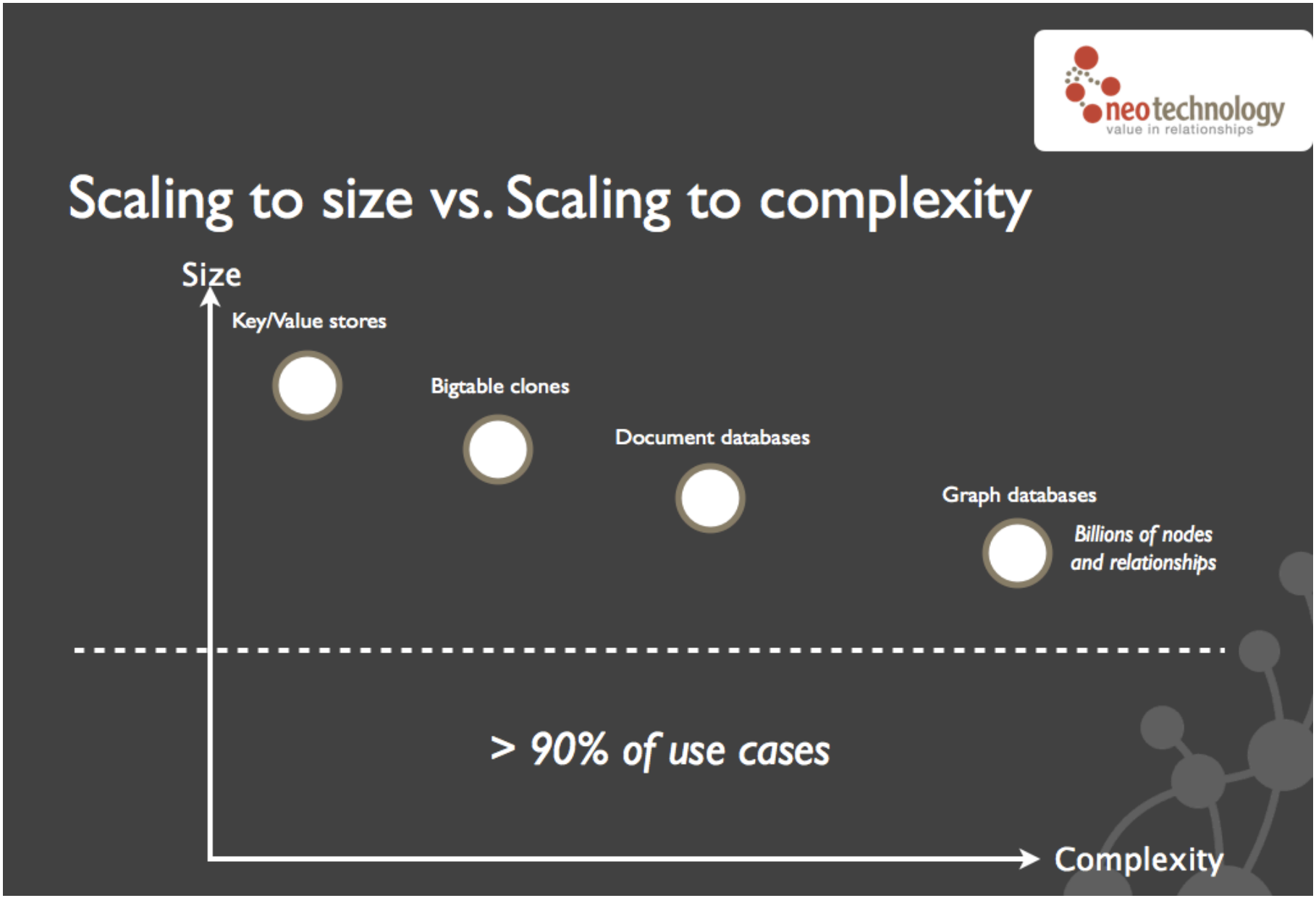


http://research.google.com/archive/bigtable.html

row keys	column family "language:"	column family "contents:"	column family anchor:cnnsi.com	column family 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>…		

Sorted rows

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

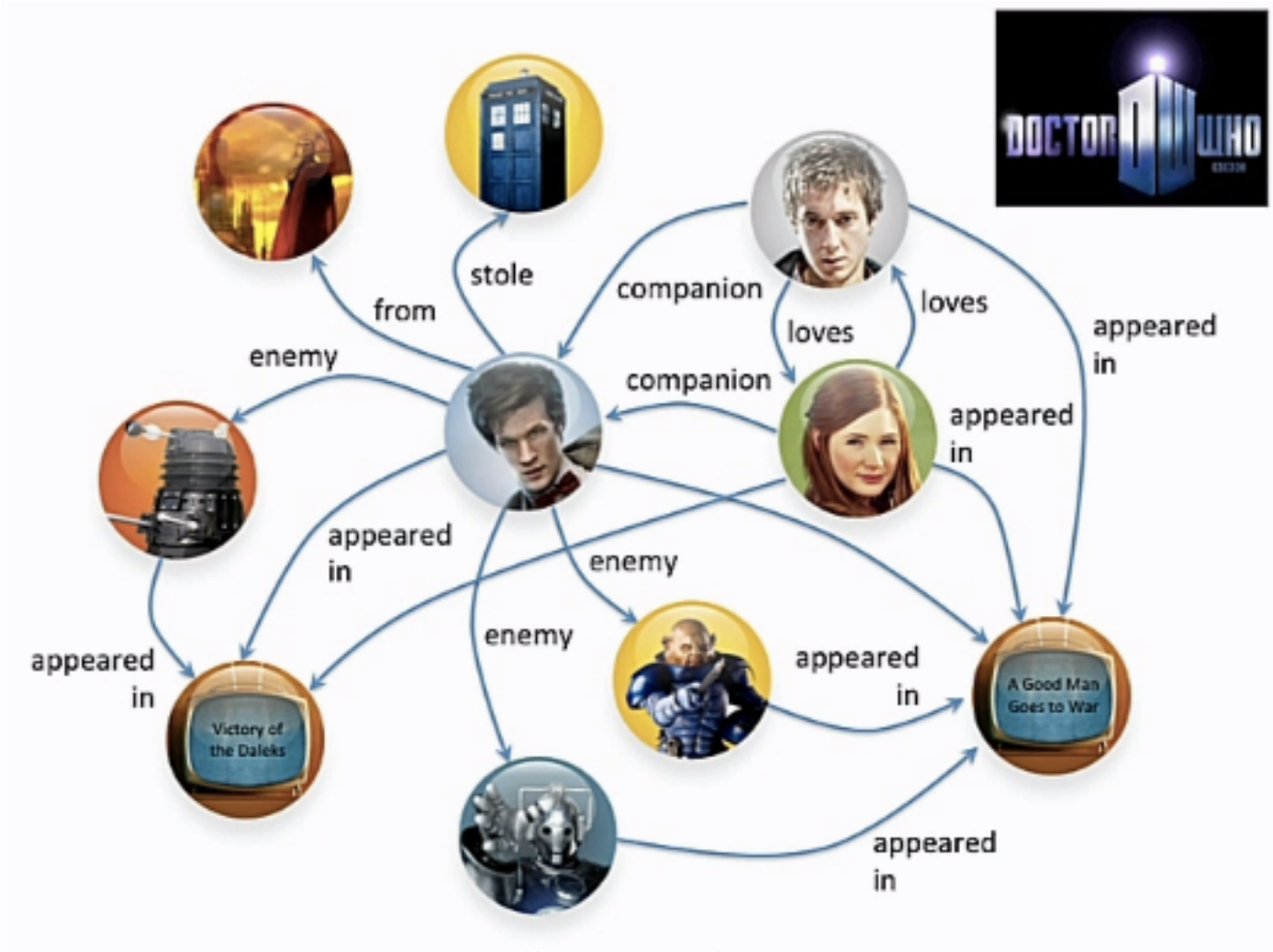

neotechnology
value in relationships
Scaling to size vs. Scaling to complexity
Size
Key/Value stores
Bigtable clones
Document databases
Graph databases
Billions of nodes and relationships
> 90% of use cases
Complexity

Graph

Neo4j

twitter

FlockDB



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