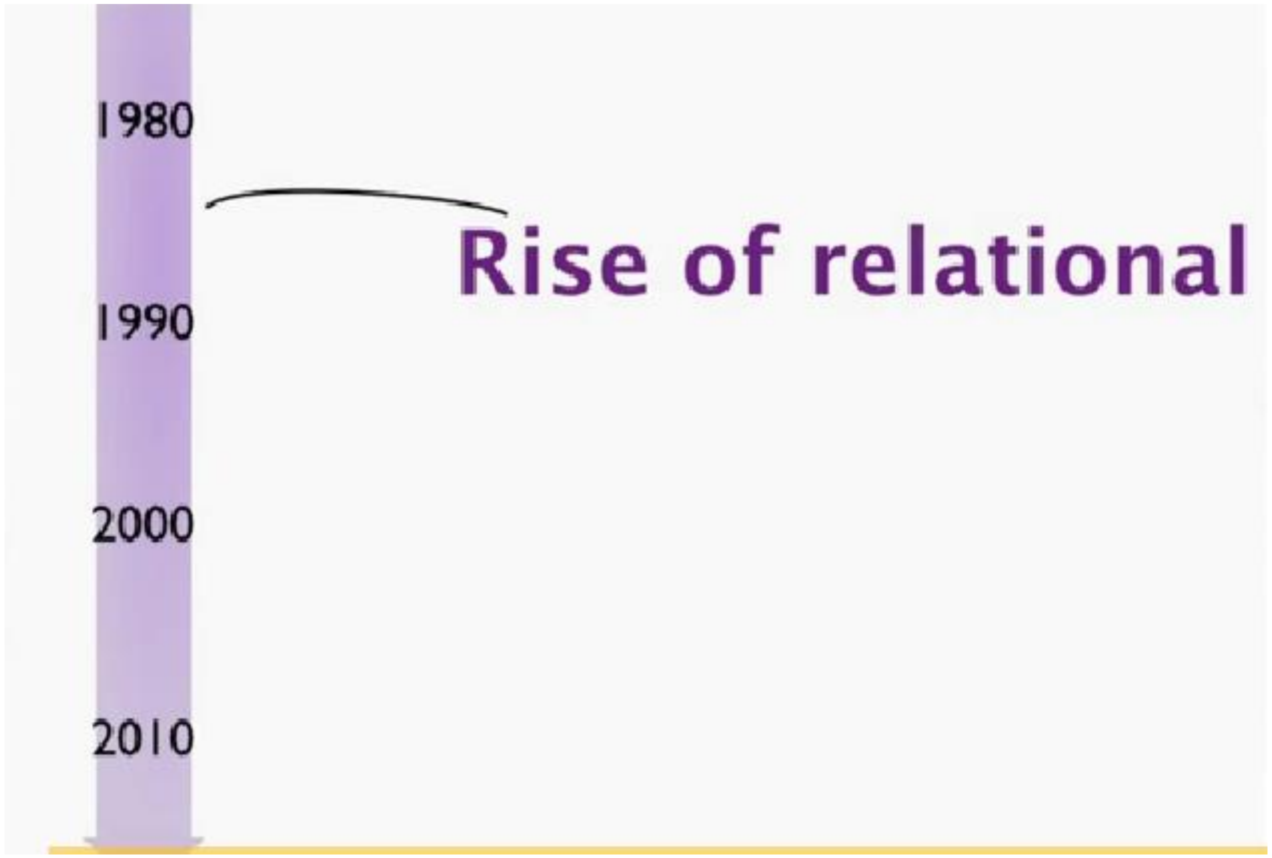
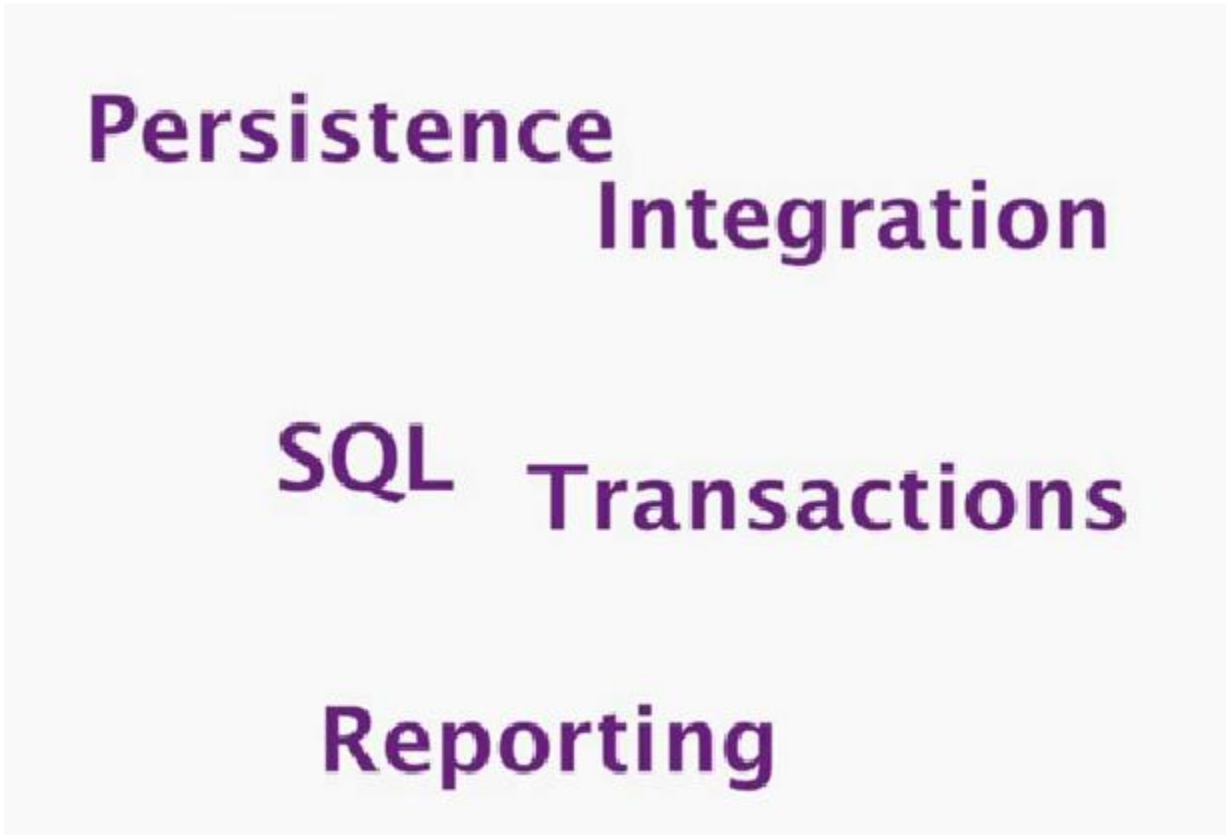
NoSQL

Sameer Dehadrai

HISTORY



Benefits of RDBMS

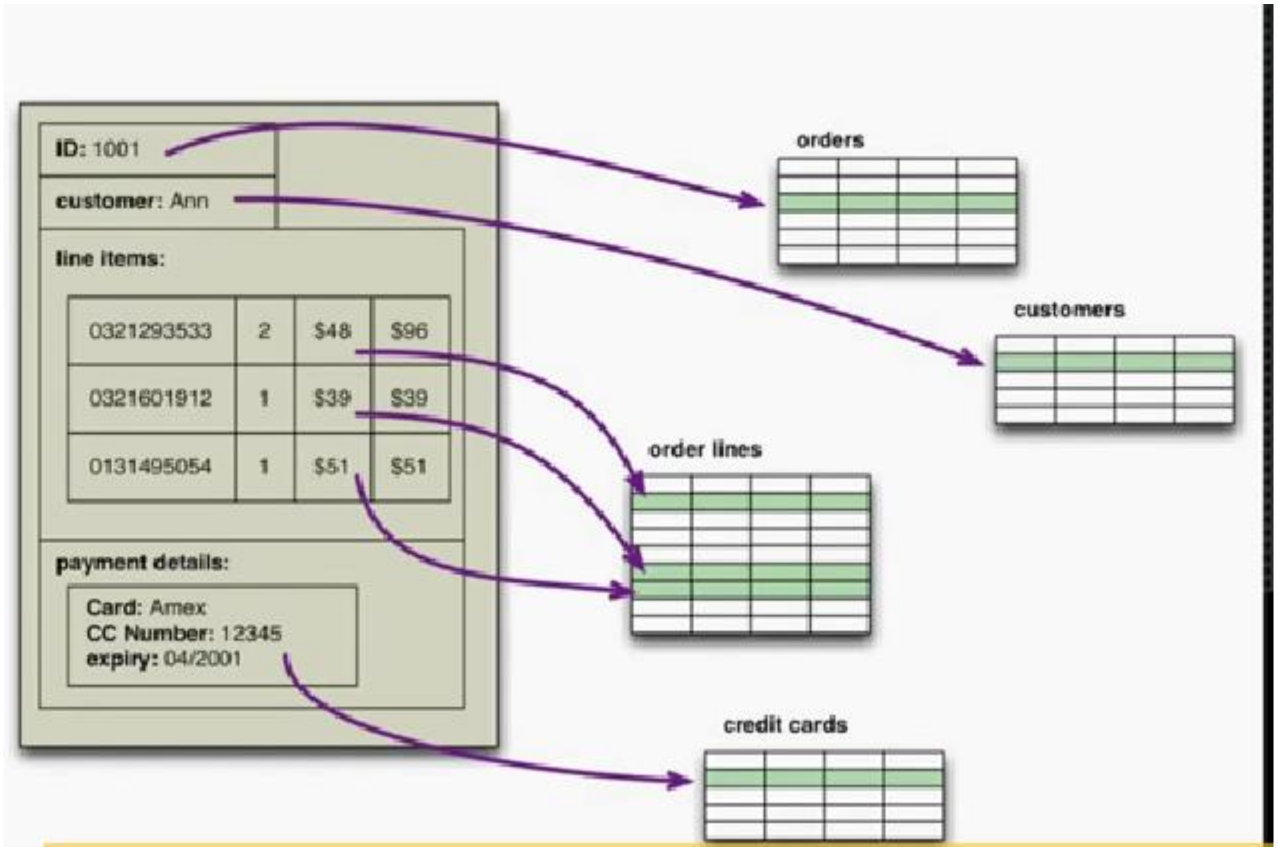


Persistence
Integration

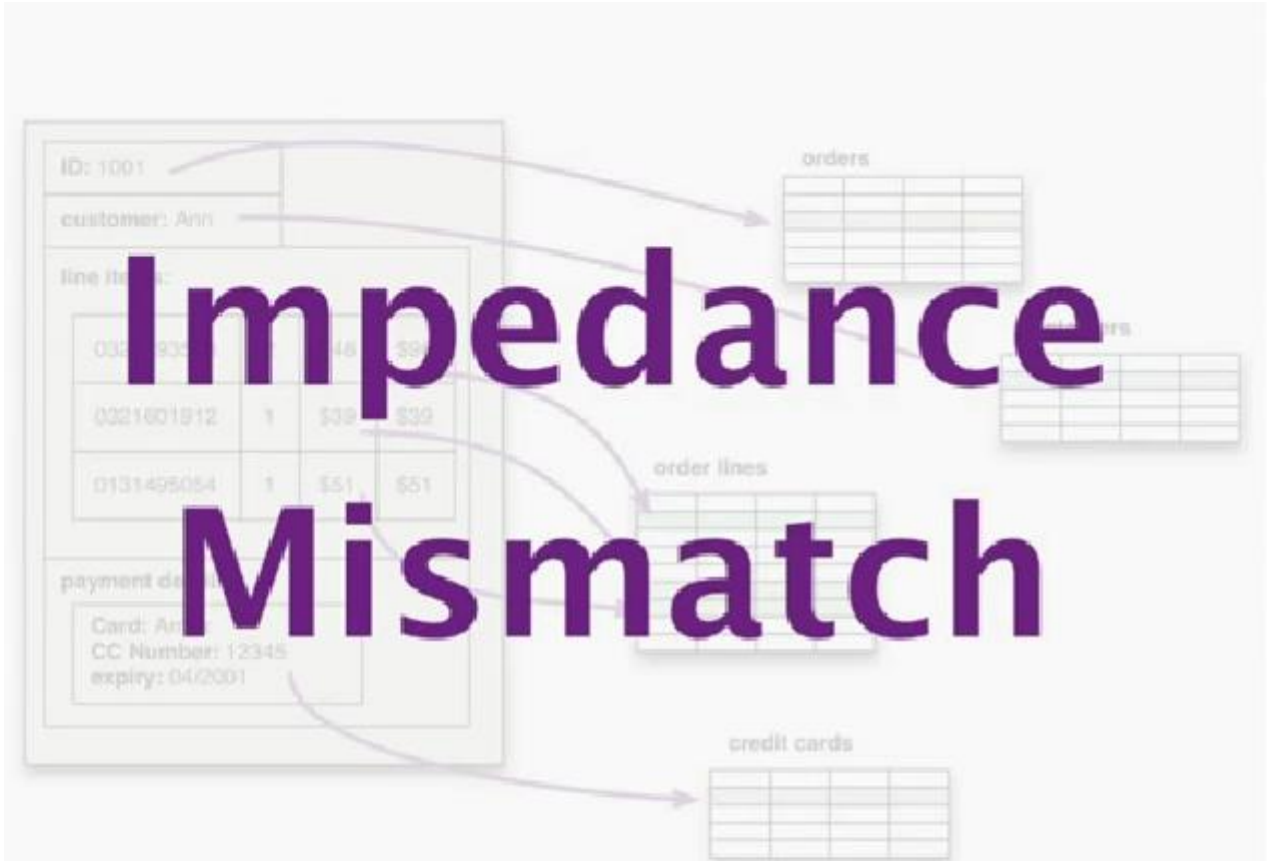
SQL
Transactions

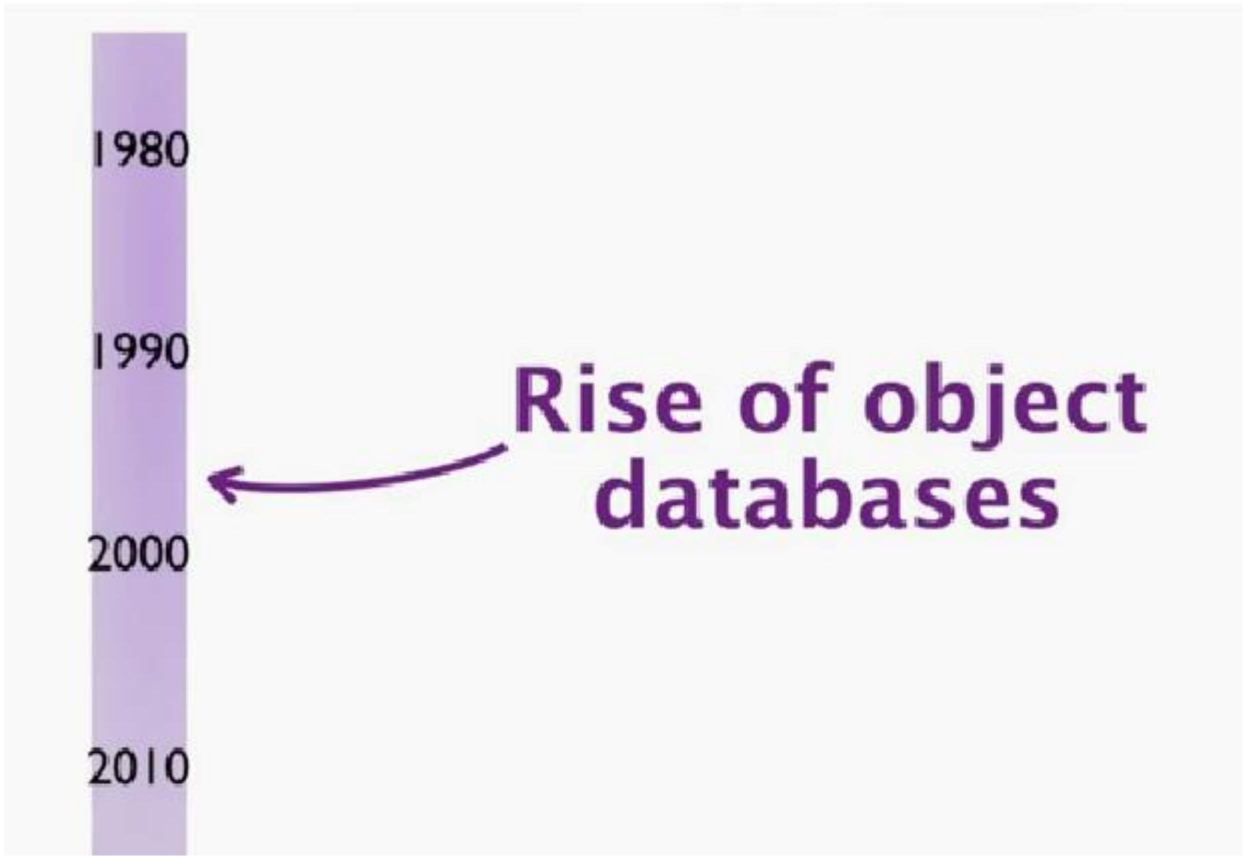
Reporting

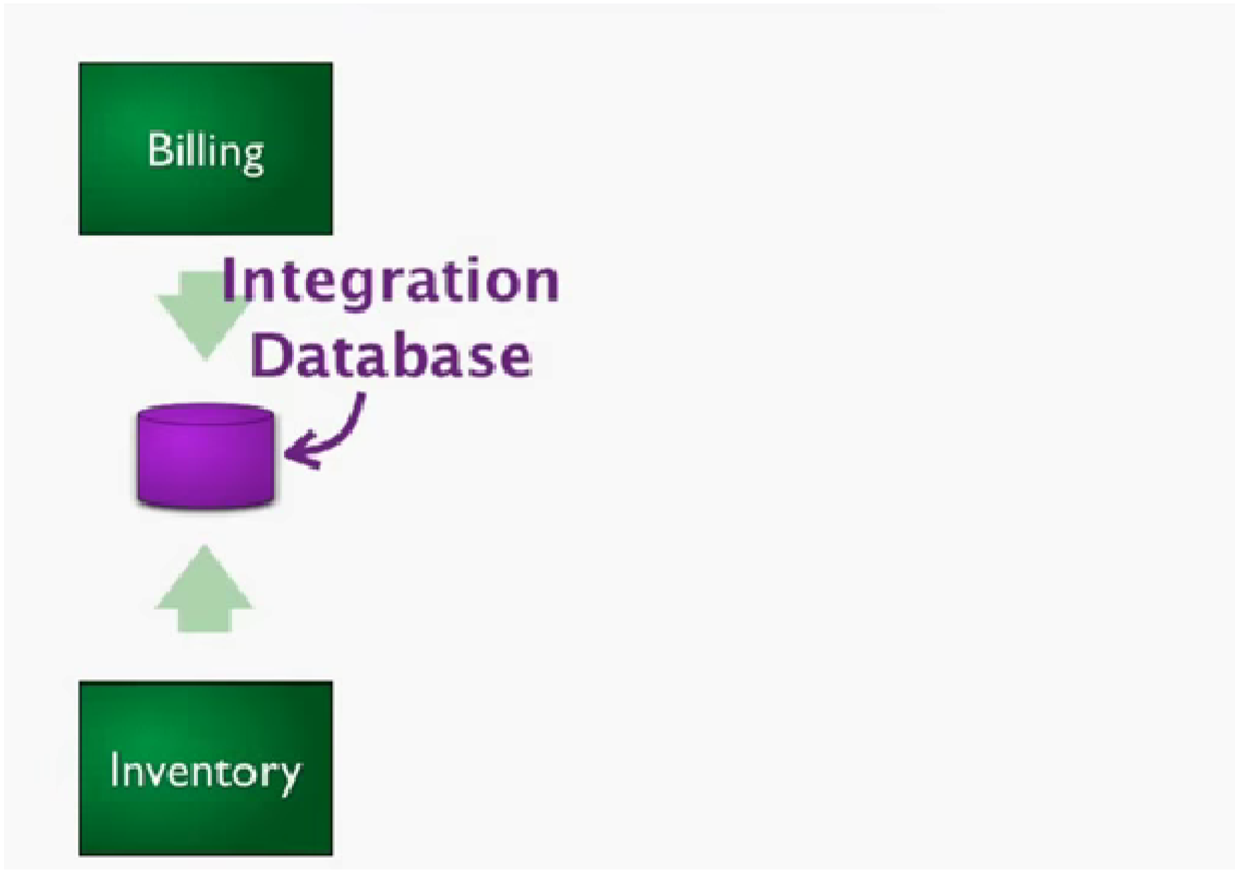
Problems with RDBMS model



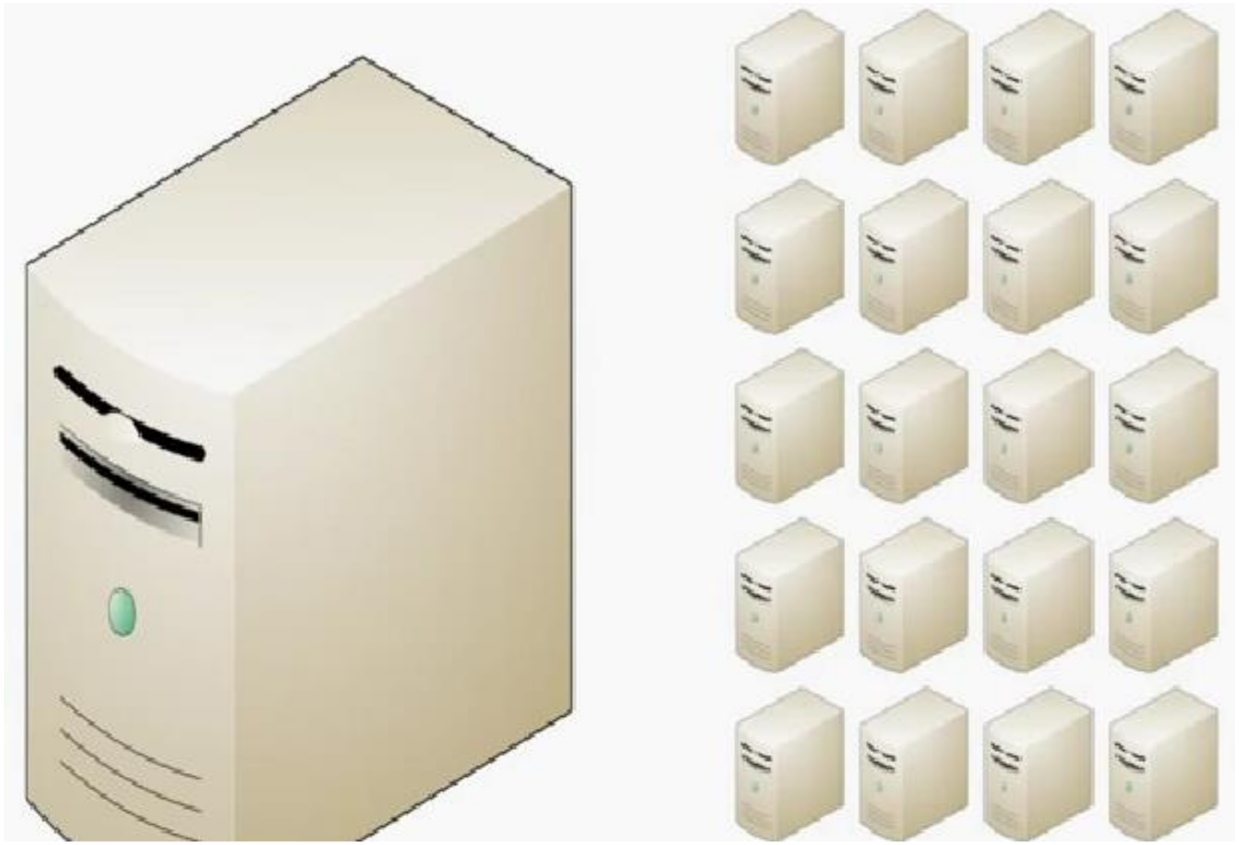
Impedance Mismatch




1980
1990
2000
2010
Rise of object databases


Billing
Integration Database
Inventory

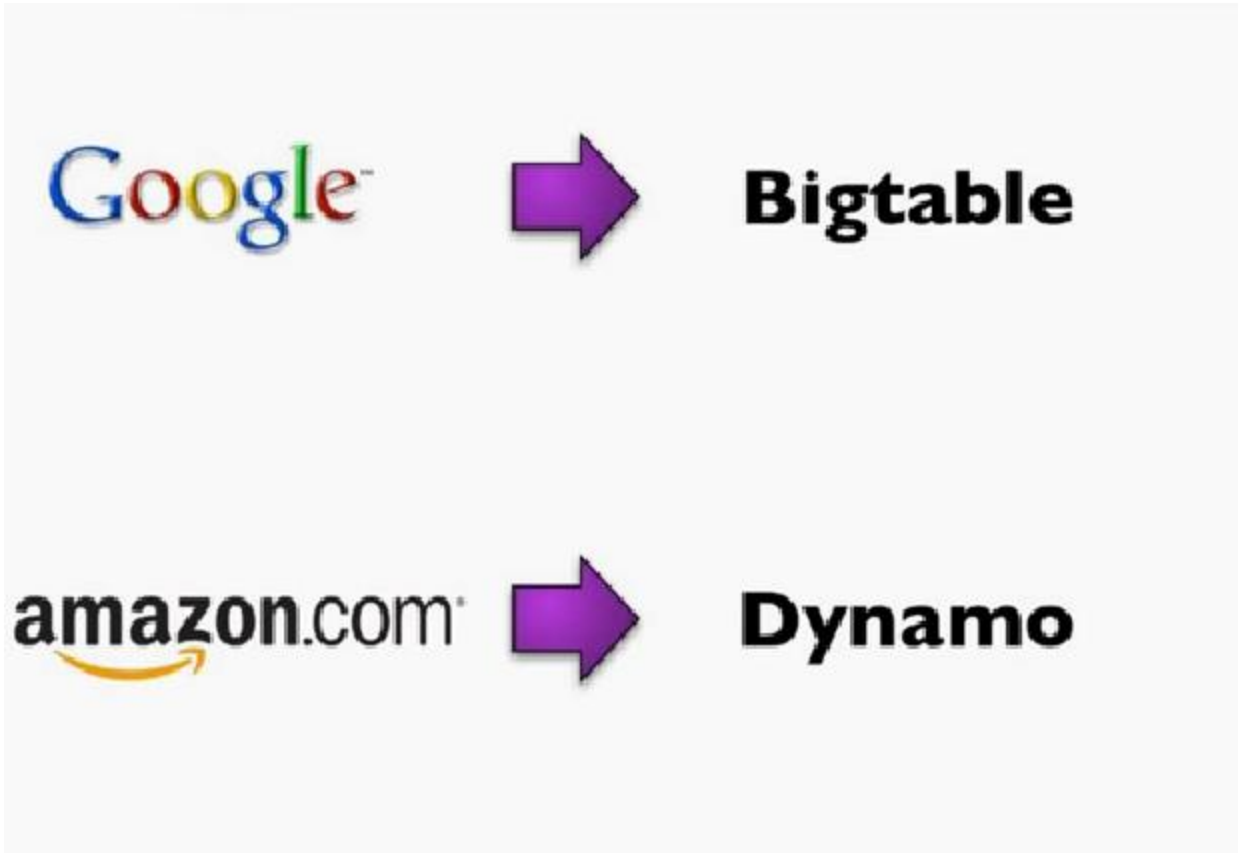
Lots of Traffic



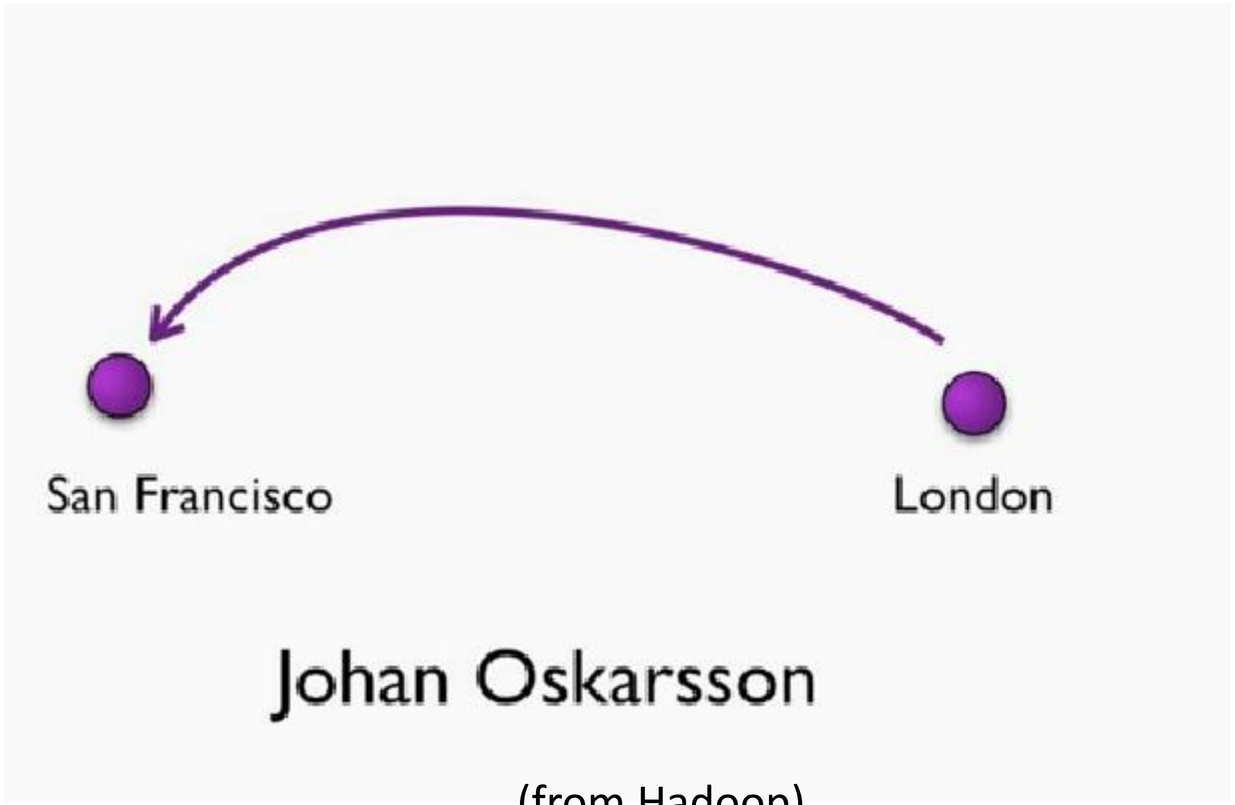
SQL
SQL

Google

amazon.com


Google
Bigtable
amazon.com
Dynamo

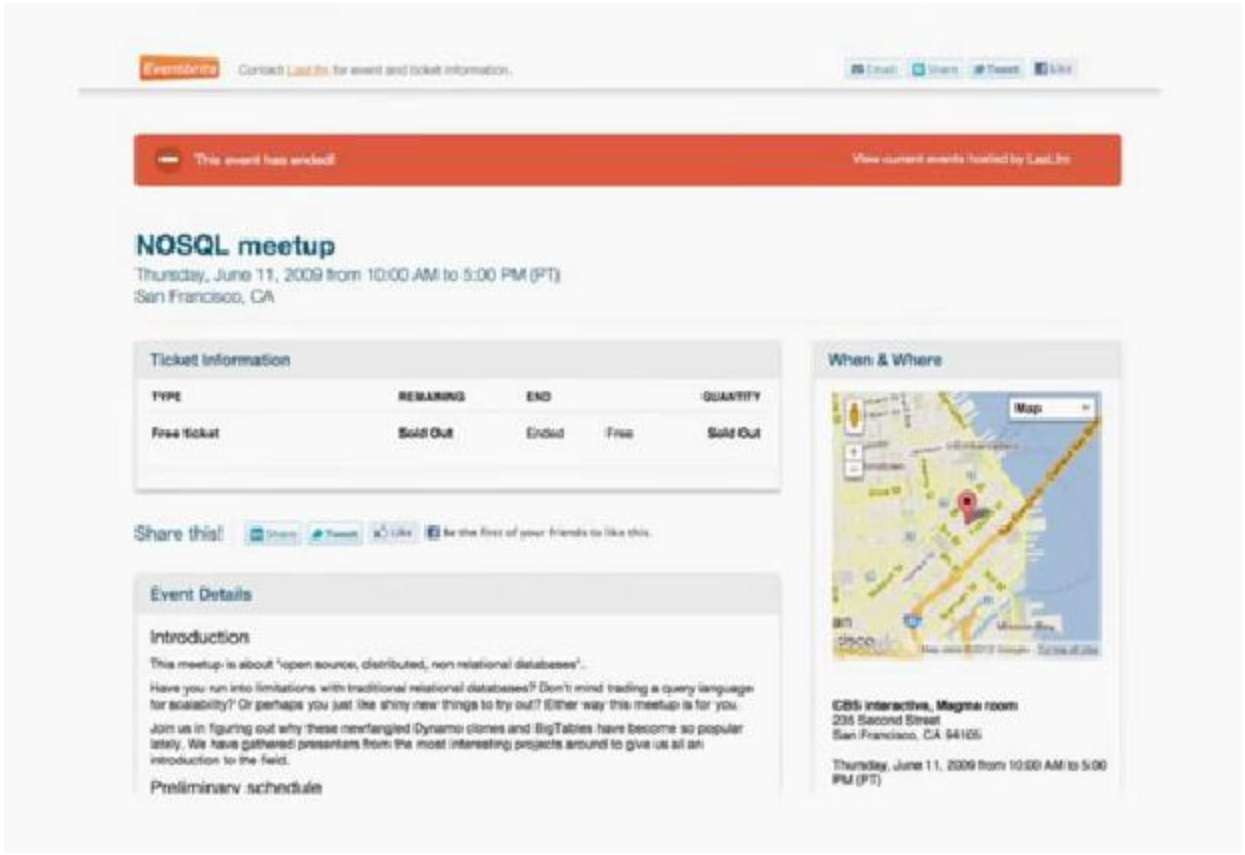
"NoSQL"



(from Hadoop)

Twitter hashtag

#nosql



Eventbrite
Contact Last.fm for event and ticket information.
Email
Share
Tweet
Like
This event has ended!
View current events hosted by Last.fm

NOSQL meetup
Thursday, June 11, 2009 from 10:00 AM to 5:00 PM (PT)
San Francisco, CA

Ticket Information

TYPE	REMAINING	END		QUANTITY
Free ticket	Sold Out	Ended	Free	Sold Out

Share this!
Share
Tweet
Like
Be the first of your friends to like this.

Event Details

Introduction

This meetup is about "open source, distributed, non relational databases".

Have you run into limitations with traditional relational databases? Don't mind trading a query language for scalability? Or perhaps you just like shiny new things to try out? Either way this meetup is for you.

Join us in figuring out why these newfangled Dynamo clones and BigTables have become so popular lately. We have gathered presenters from the most interesting projects around to give us all an introduction to the field.

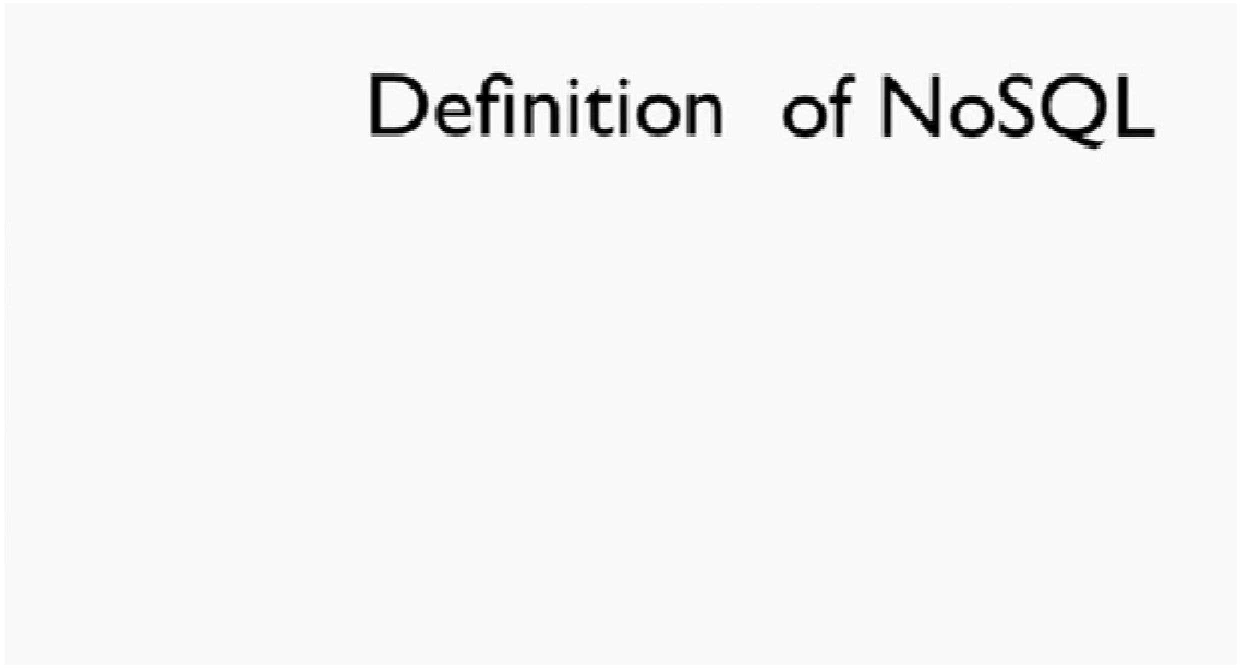
Preliminary schedule

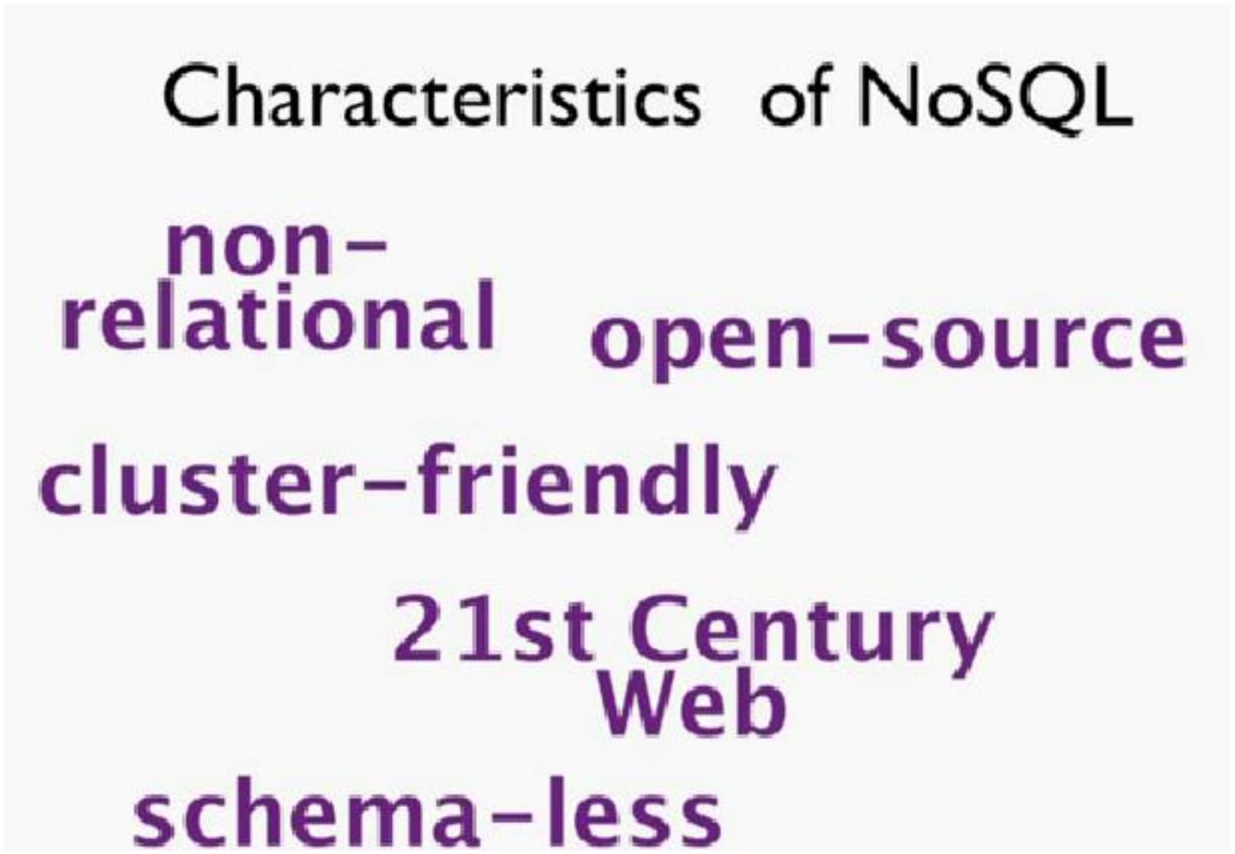
When & Where

CBS interactive, Magma room
235 Second Street
San Francisco, CA 94105

Thursday, June 11, 2009 from 10:00 AM to 5:00 PM (PT)

mongoDB
CouchDB
relax
Project Voldemort
A distributed database.
Cassandra
HYPERTABLE
Dynomite
APACHE
HBASE


Definition of NoSQL



Characteristics of NoSQL

non-relational

open-source

cluster-friendly

21st Century Web

schema-less

Data Model

mongoDB

RAVEN DB

CouchDB
relax

Cassandra

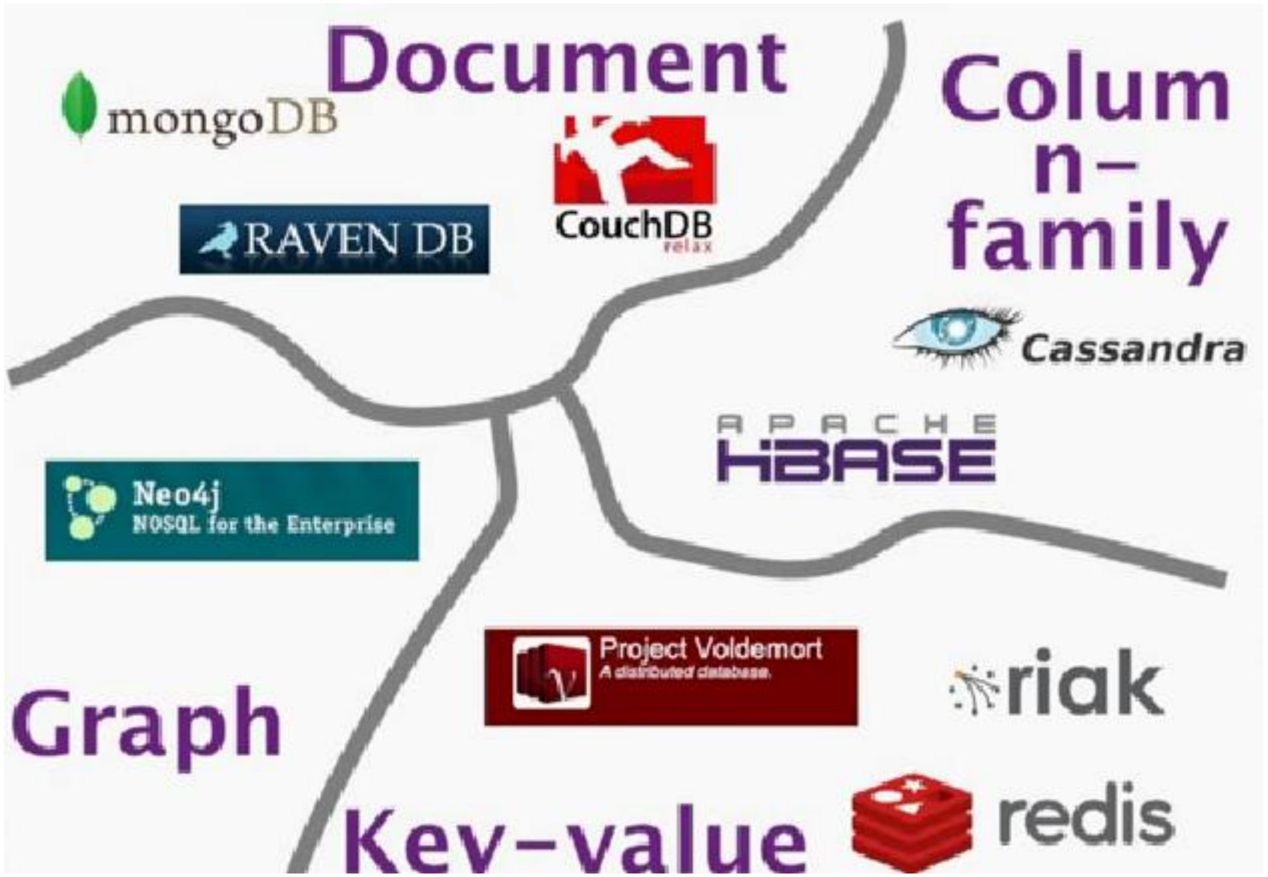
APACHE
HBASE

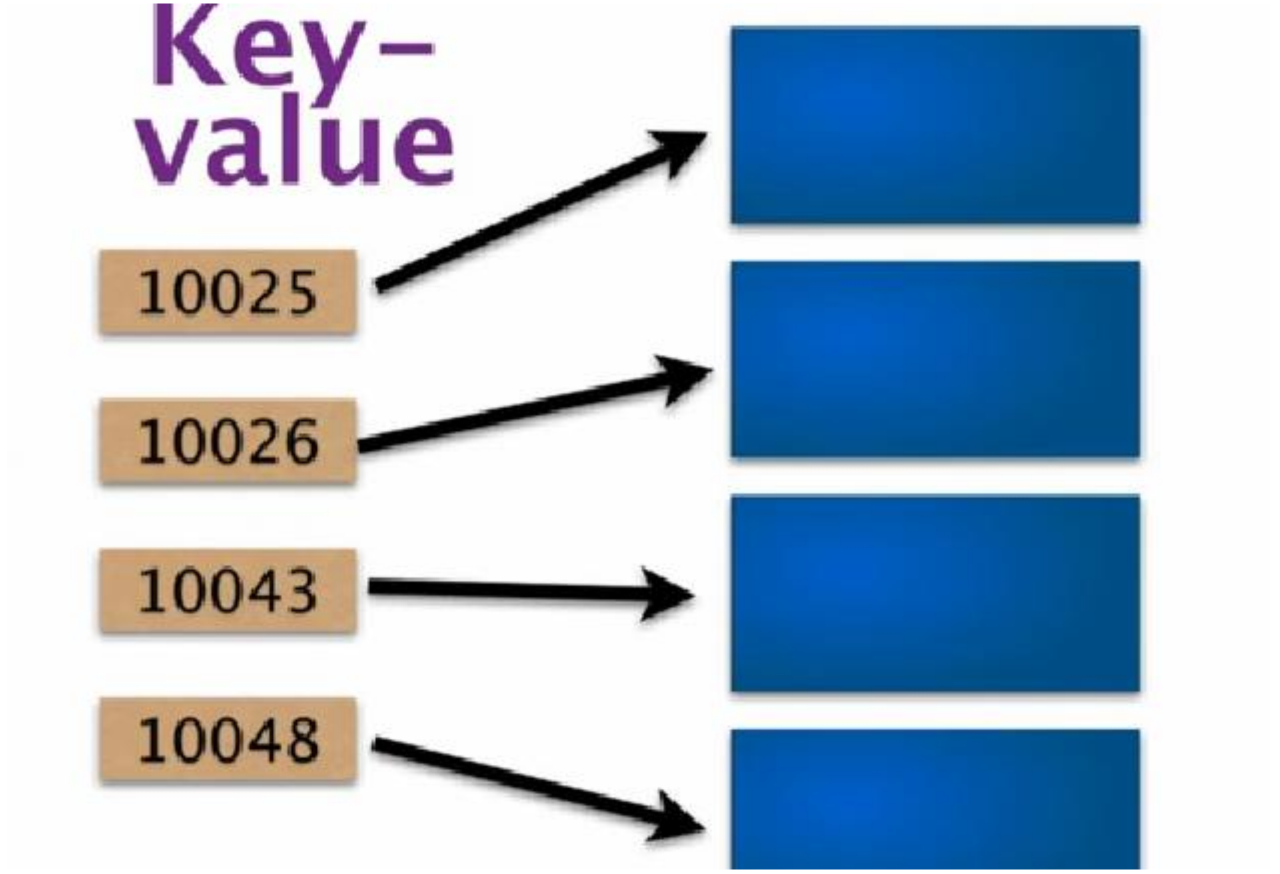
Neo4j
NOSQL for the Enterprise

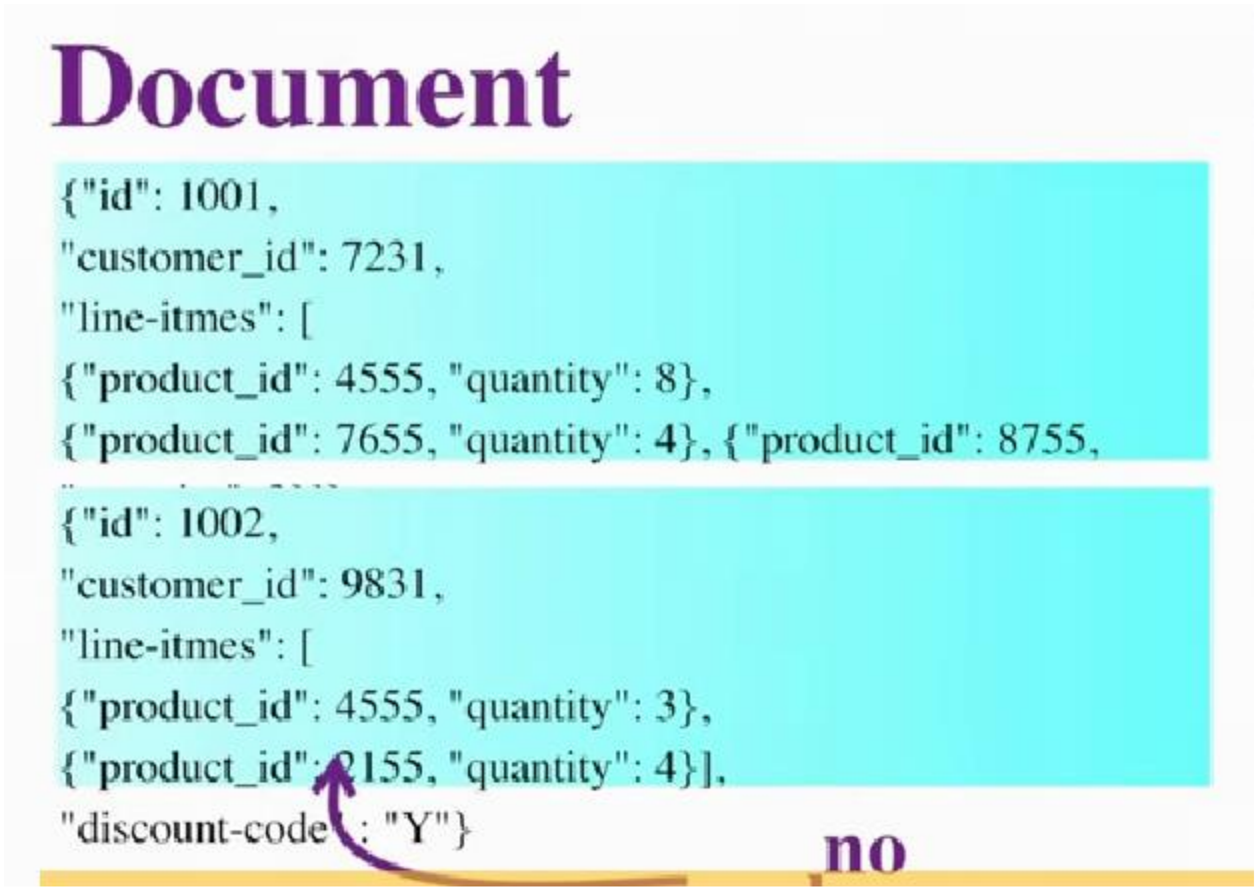
Project Voldemort
A distributed database.

riak

redis


Document
mongoDB
RAVEN DB
CouchDB relax
Colum n-family
Cassandra
APACHE HBASE
Neo4j NOSQL for the Enterprise
Project Voldemort A distributed database.
riak
Graph
Key-value
redis


Key-value
10025
10026
10043
10048

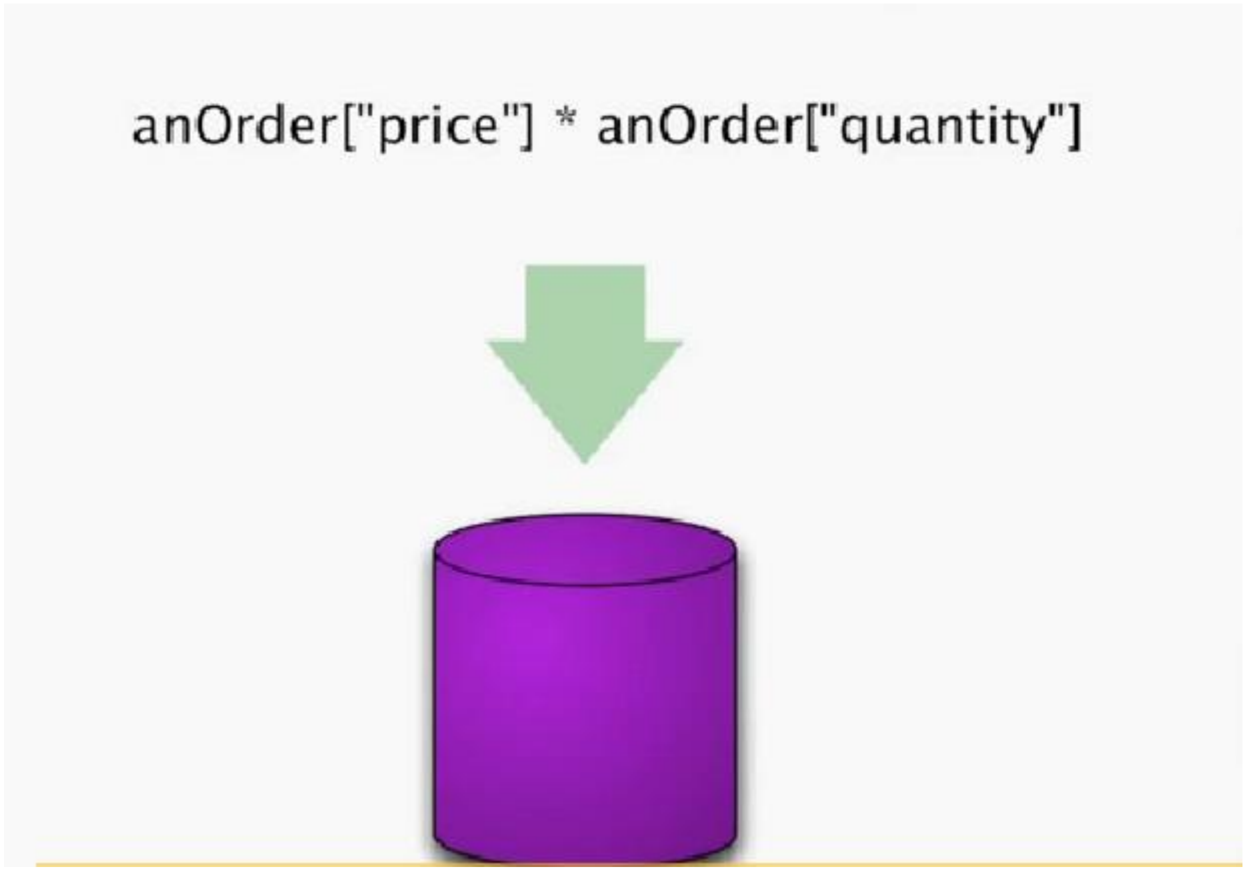


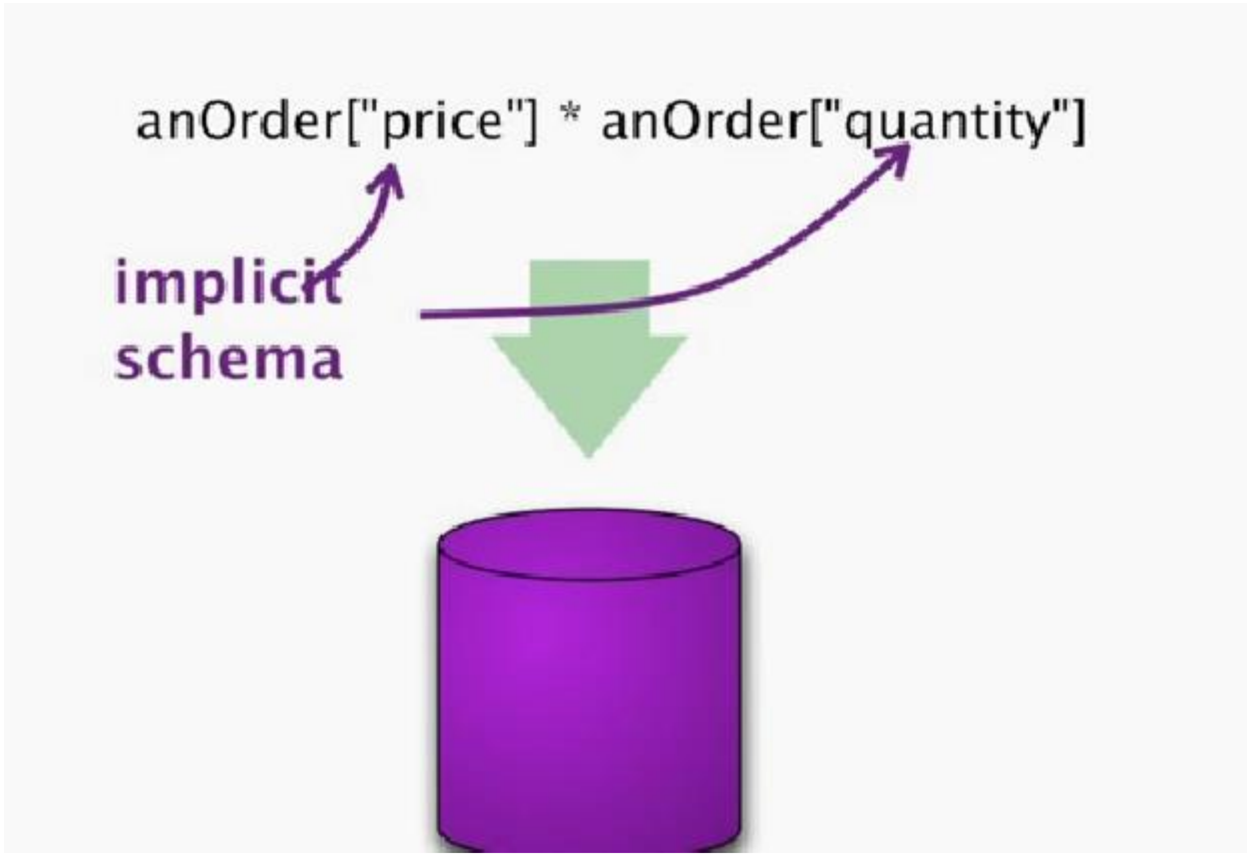
Document

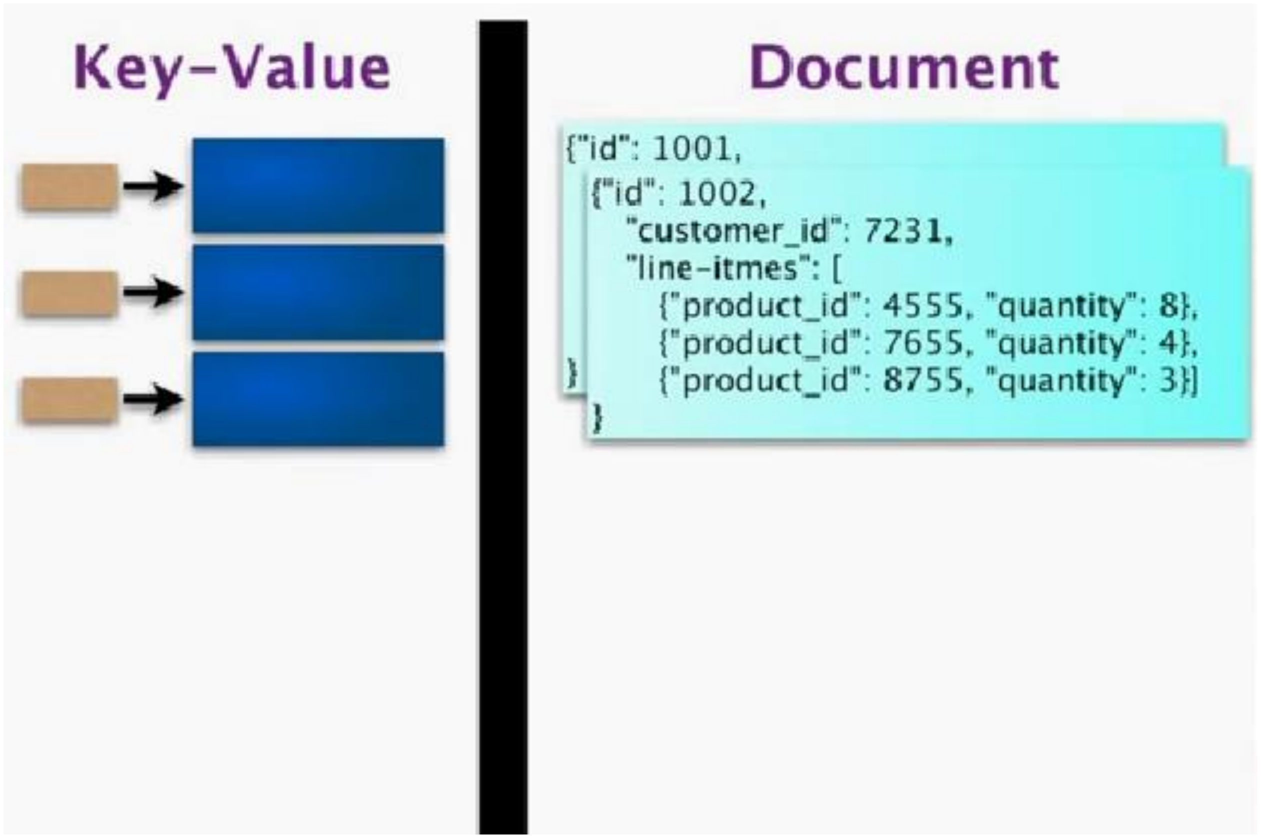
```
{"id": 1001,
"customer_id": 7231,
"line-itmes": [
{"product_id": 4555, "quantity": 8},
{"product_id": 7655, "quantity": 4}, {"product_id": 8755,
```

```
{"id": 1002,
"customer_id": 9831,
"line-itmes": [
{"product_id": 4555, "quantity": 3},
{"product_id": 2155, "quantity": 4}],
"discount-code": "Y"}
```

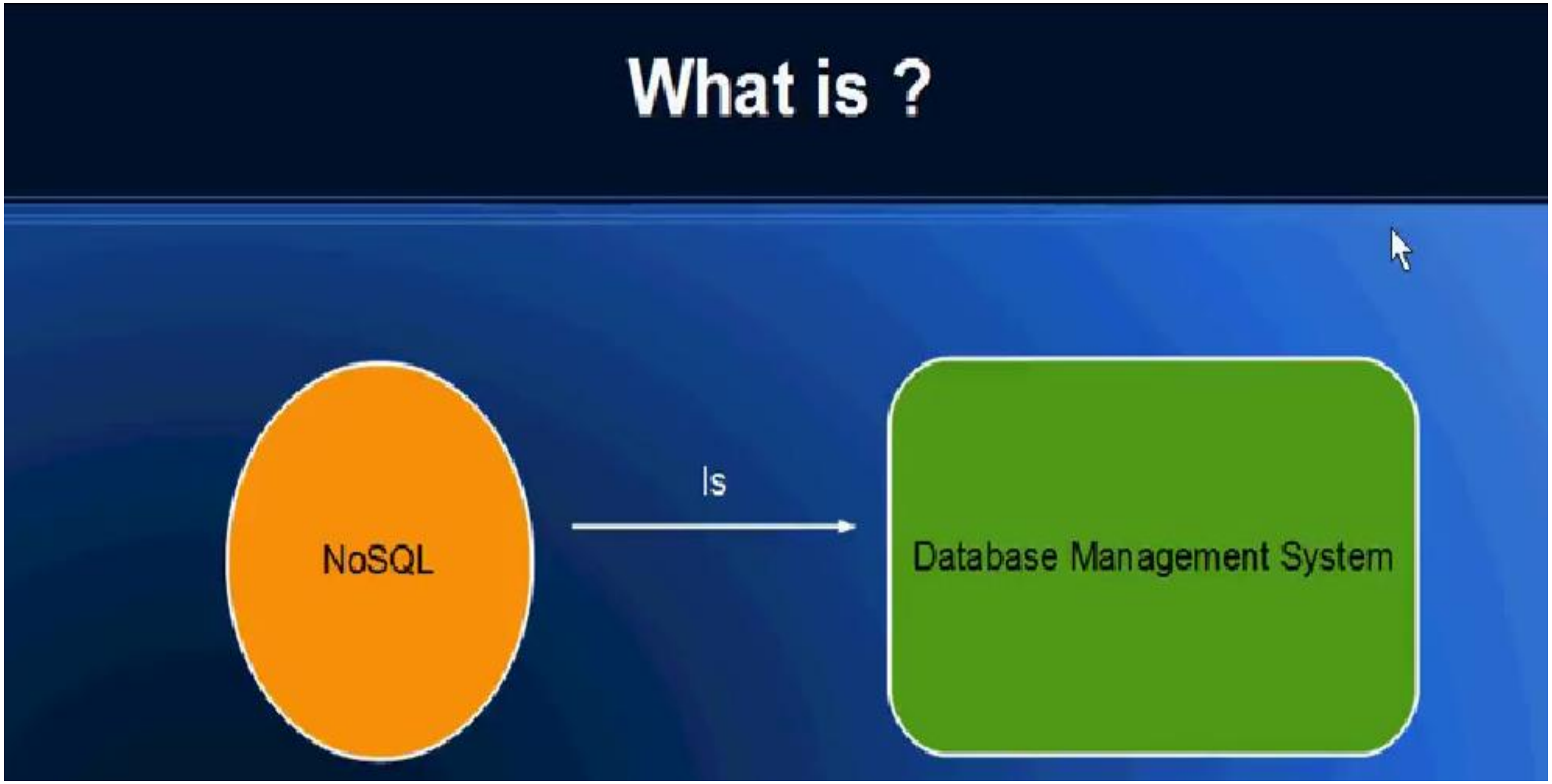
no


anOrder["price"] * anOrder["quantity"]

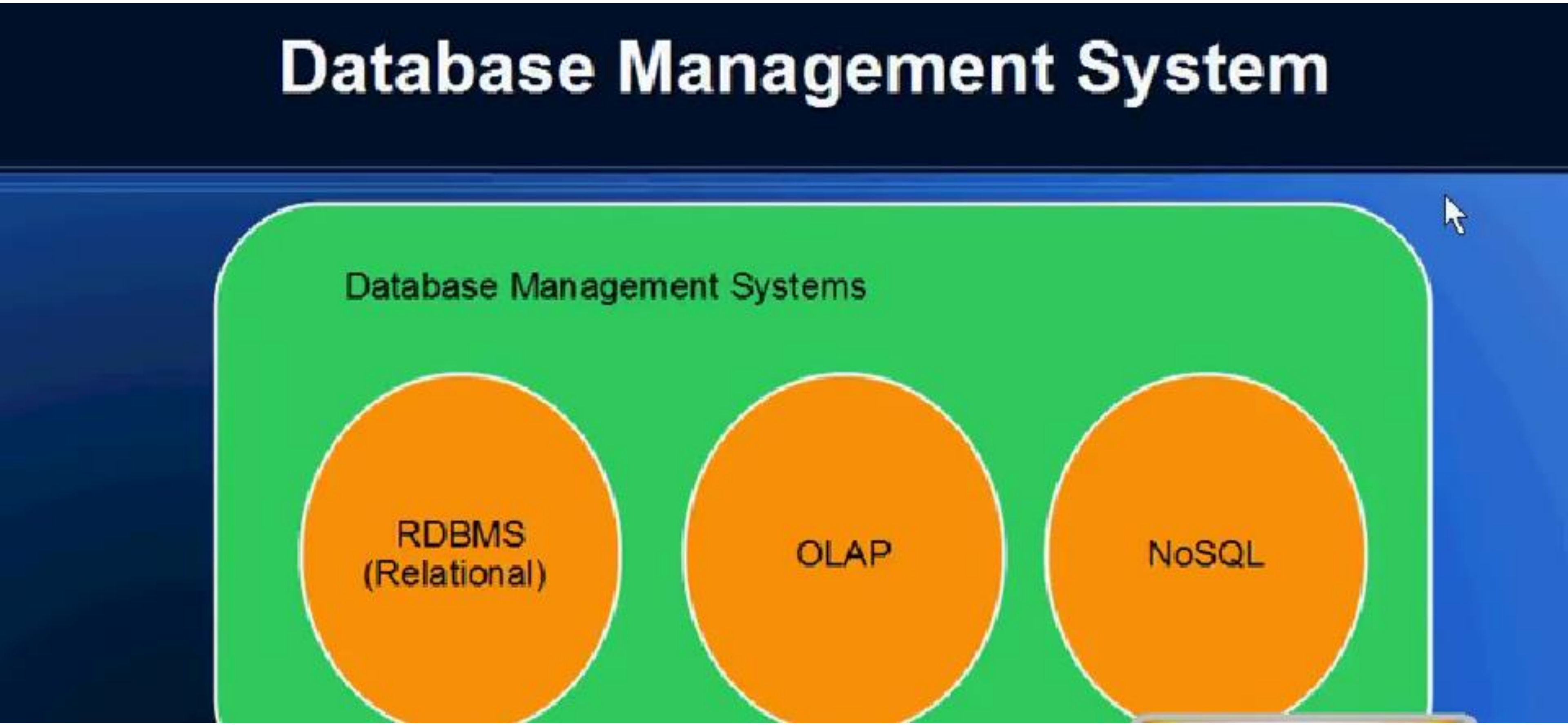

anOrder["price"] * anOrder["quantity"]
implicit schema



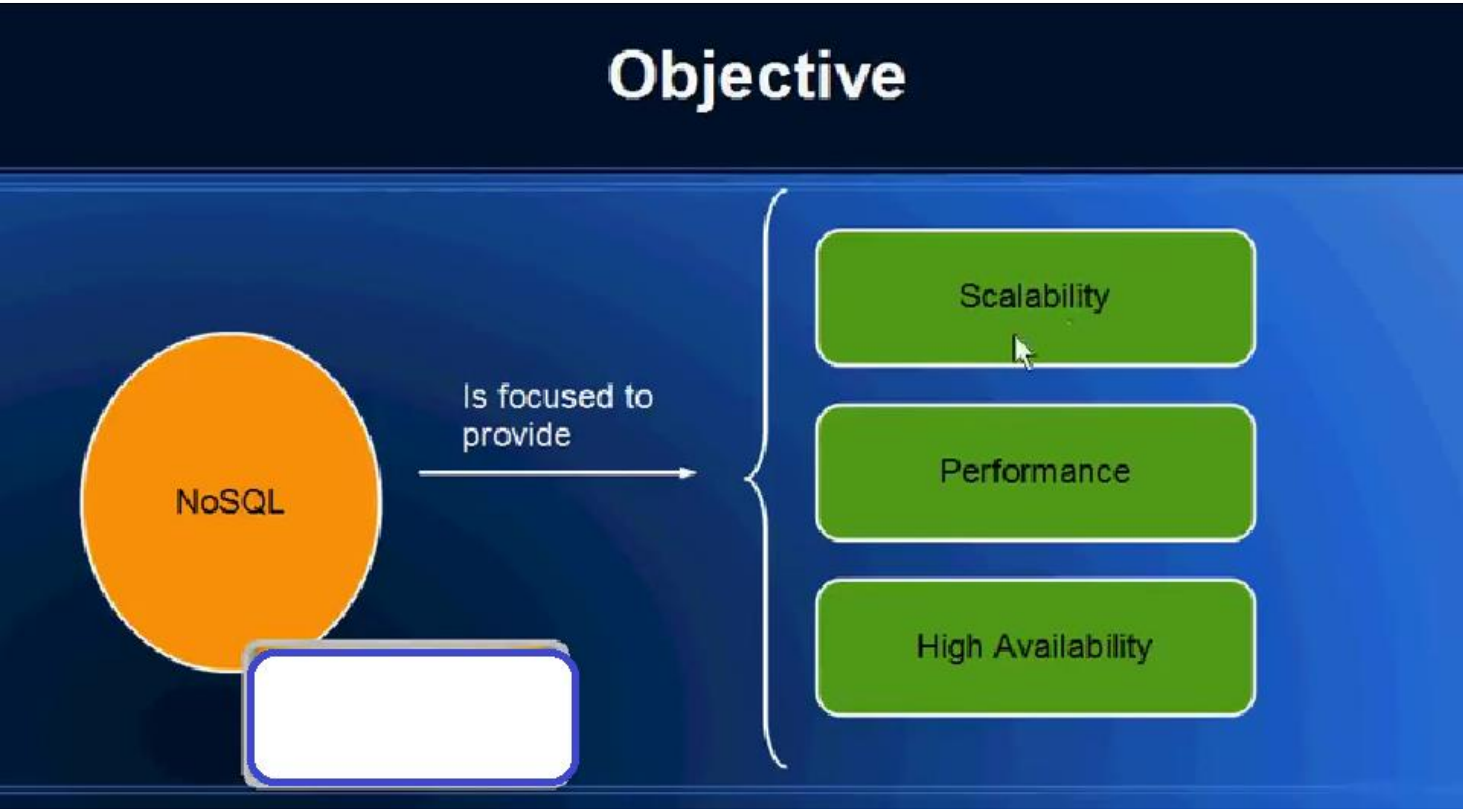
```
{"id": 1001,
}
{"id": 1002,
   "customer_id": 7231,
   "line-itmes": [
      {"product_id": 4555, "quantity": 8},
      {"product_id": 7655, "quantity": 4},
      {"product_id": 8755, "quantity": 3}]
}
```



What is ?

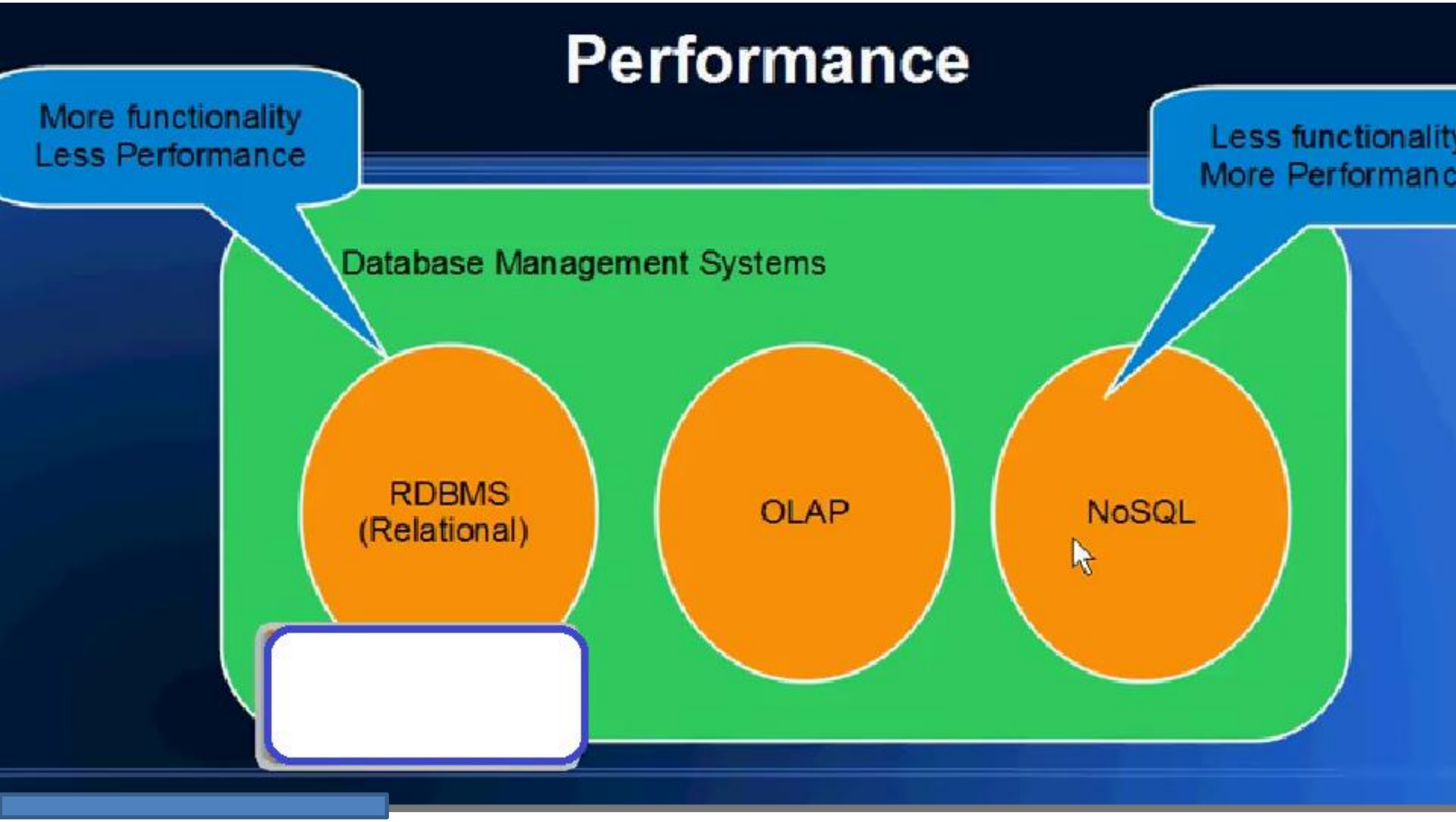


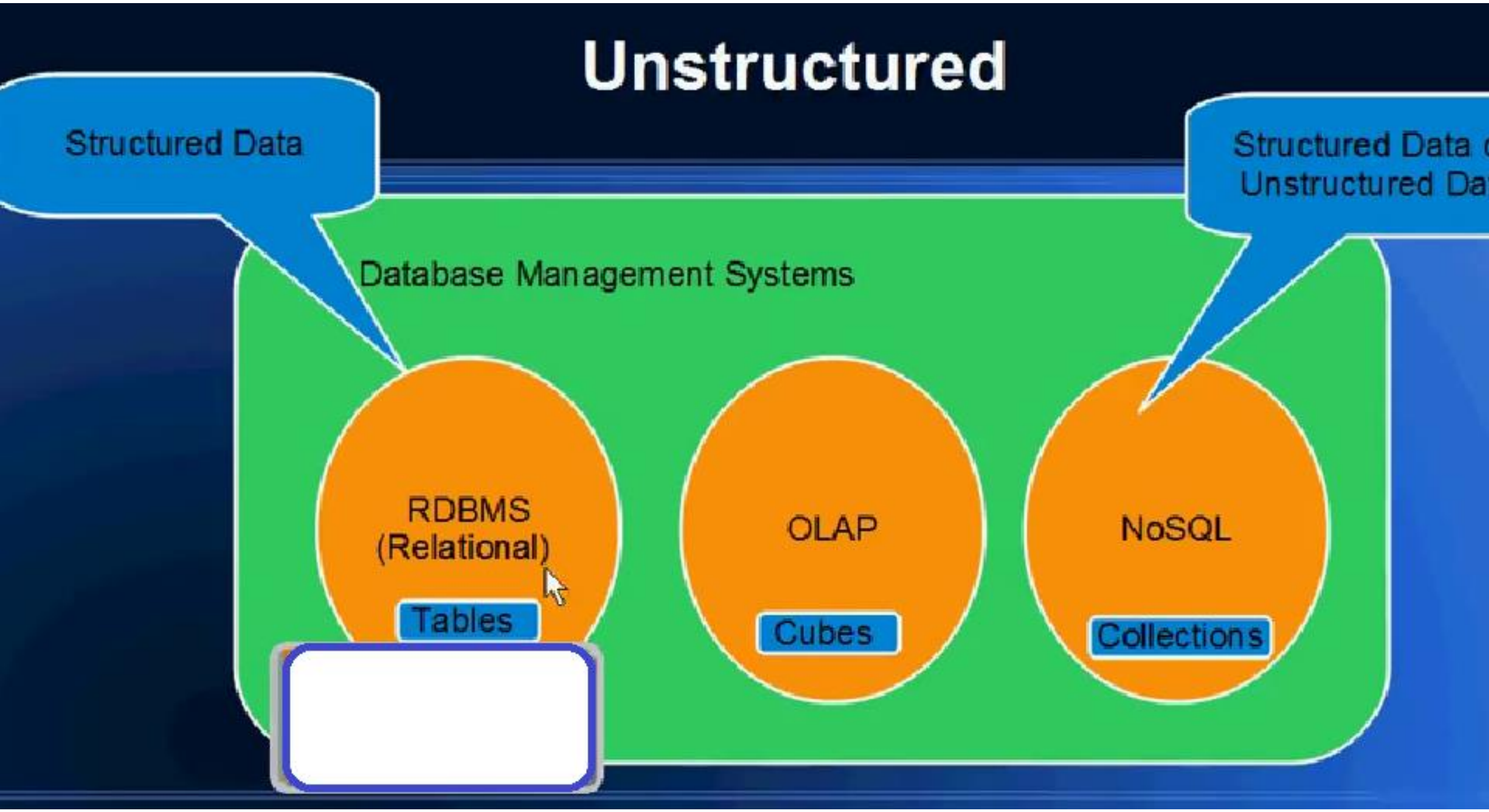
Database Management System

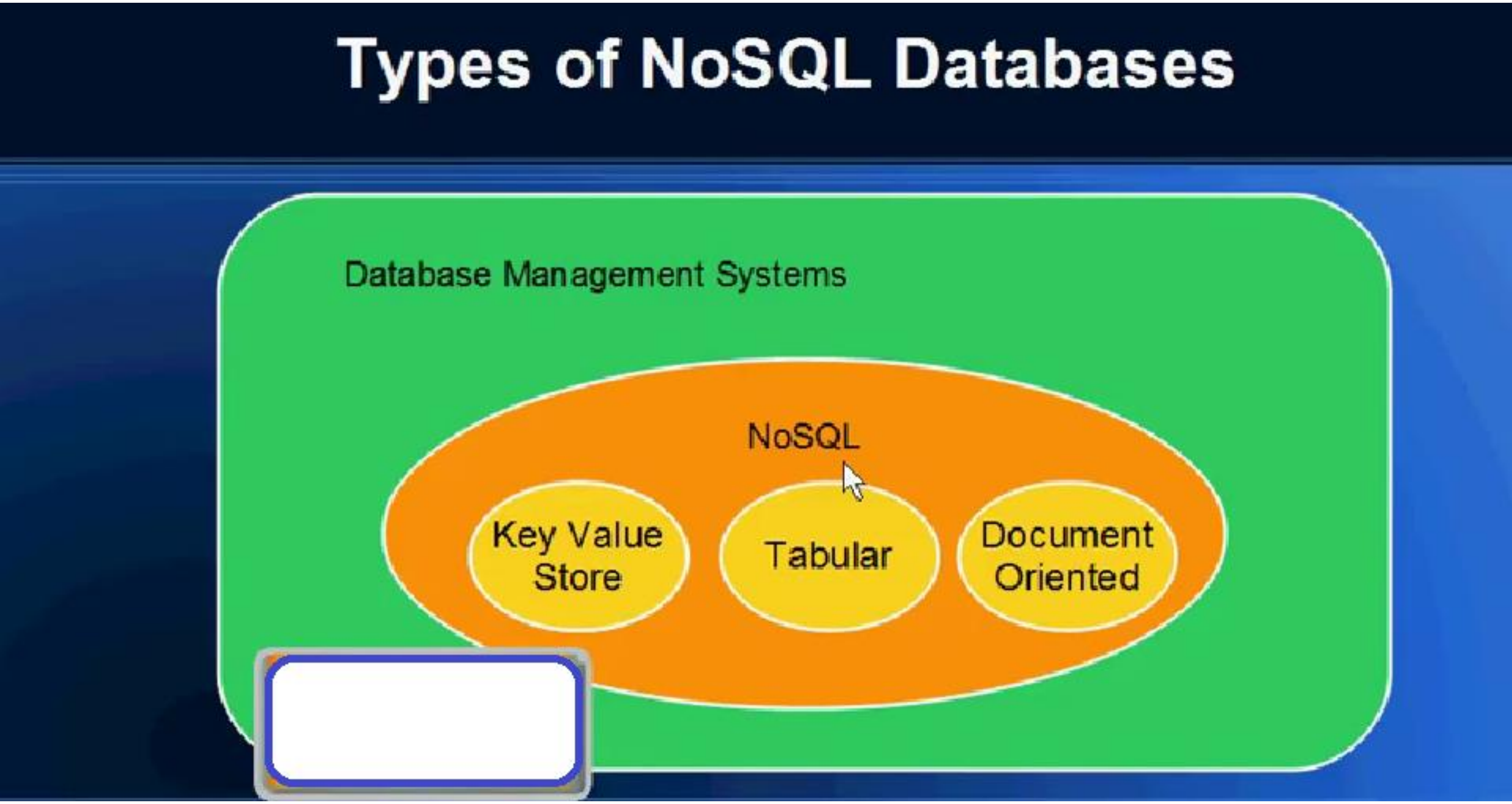


Objective

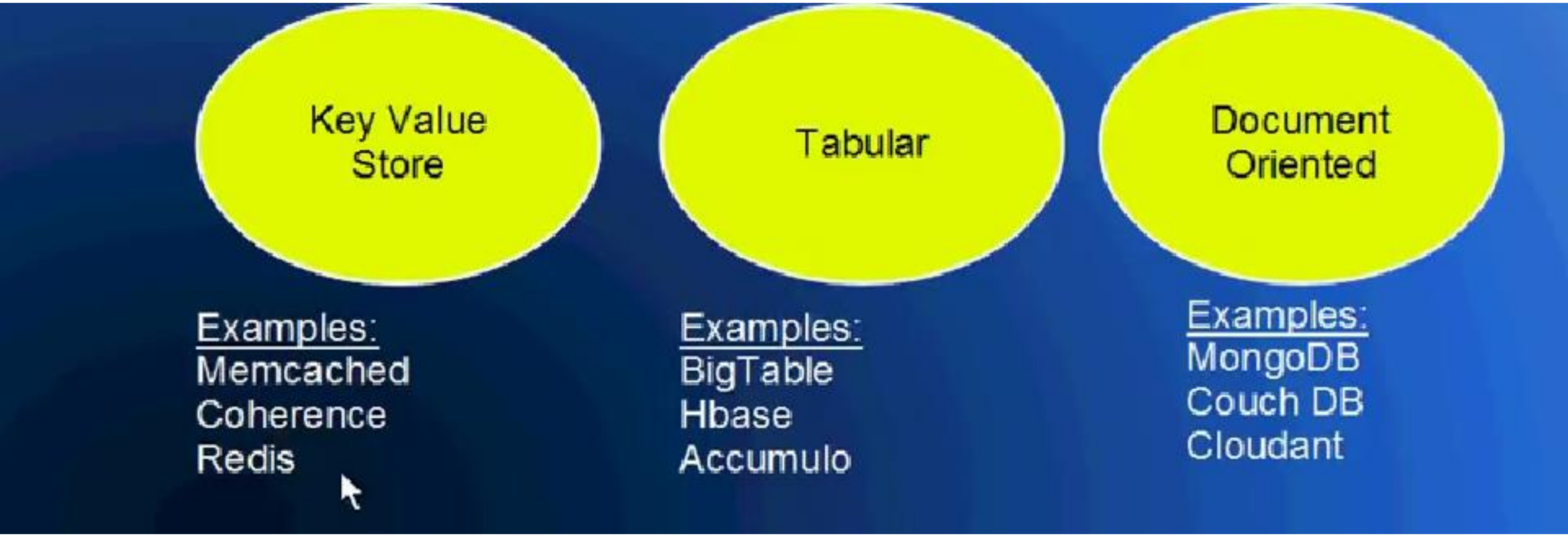
NoSQL
Is focused to provide
Scalability
Performance
High Availability


Performance
More functionality
Less Performance
Less functionalit
More Performanc
Database Management Systems
RDBMS
(Relational)
OLAP
NoSQL


Unstructured
Structured Data
Structured Data
Unstructured Da
Database Management Systems
RDBMS (Relational)
Tables
OLAP
Cubes
NoSQL
Collections



Types of NoSQL Databases

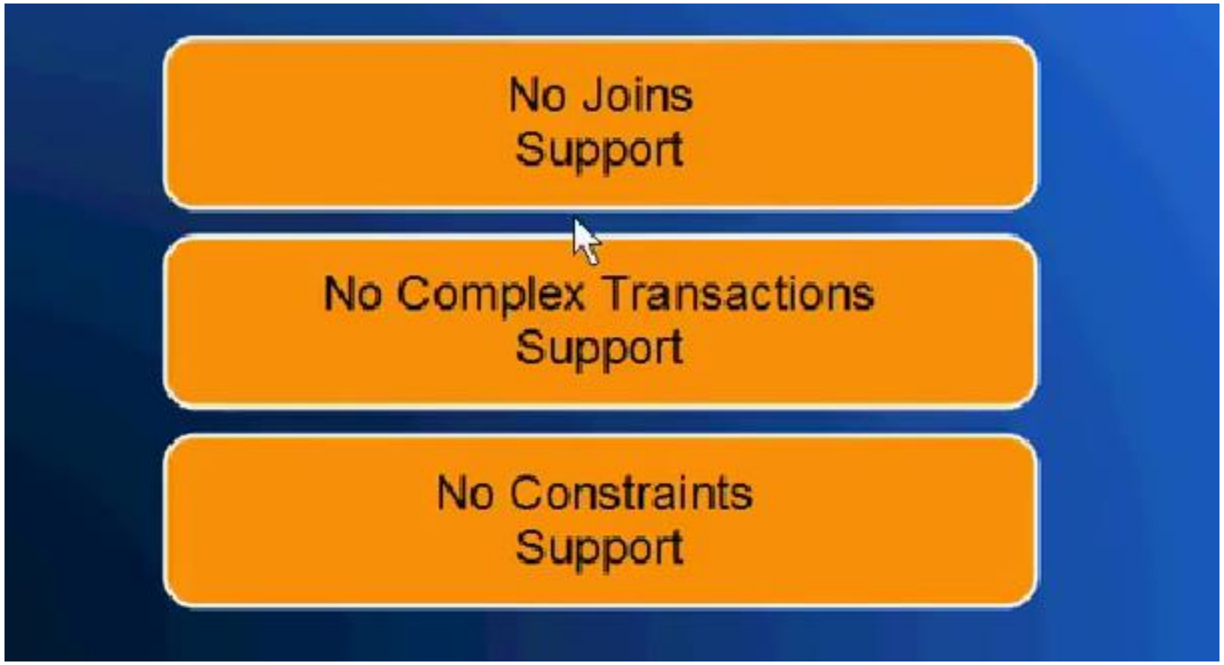


Examples:
Memcached
Coherence
Redis

Examples:
BigTable
Hbase
Accumulo

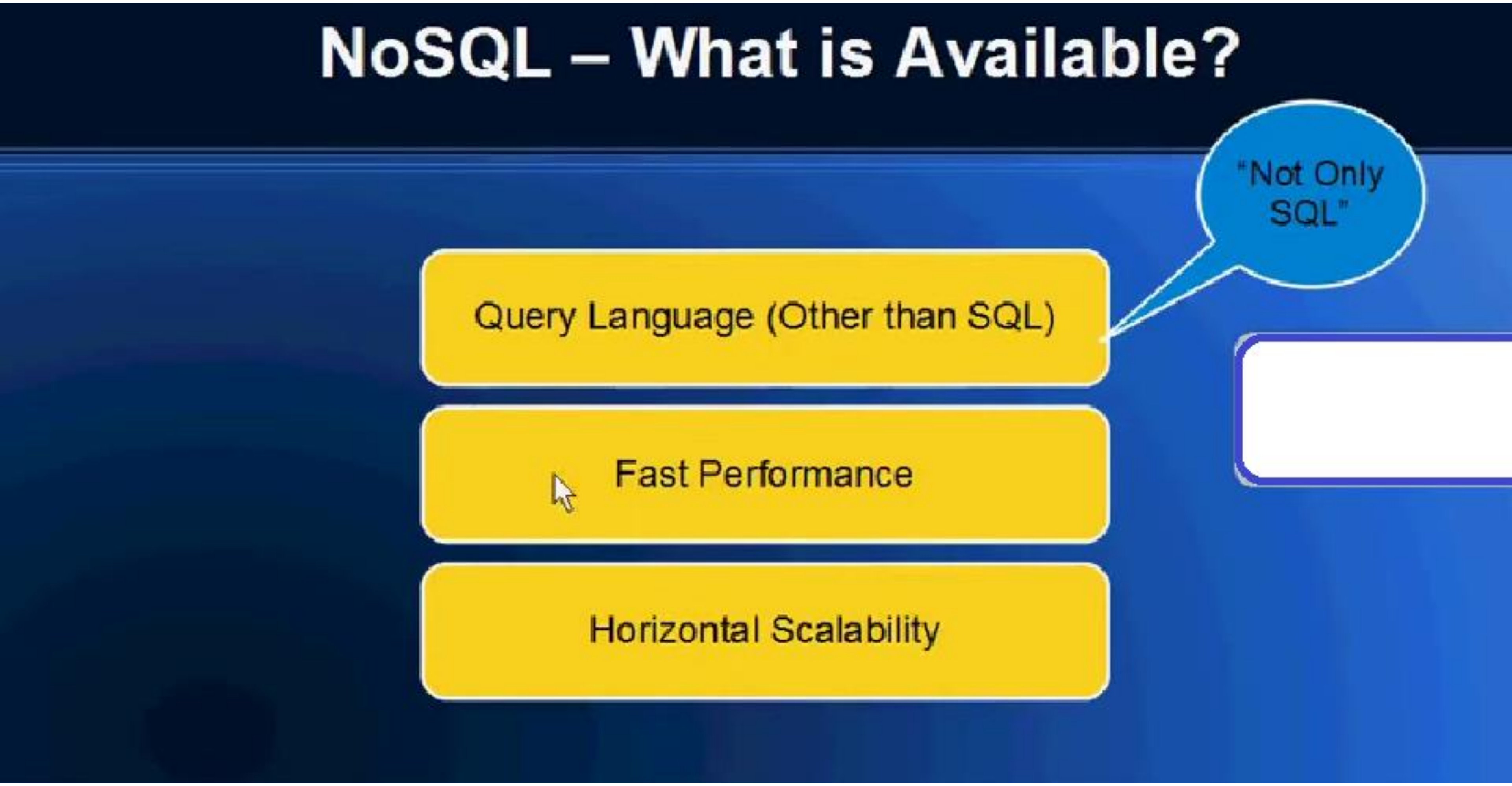
Examples:
MongoDB
Couch DB
Cloudant

NoSQL – What is Missing


No Joins
Support

No Complex Transactions
Support

No Constraints
Support

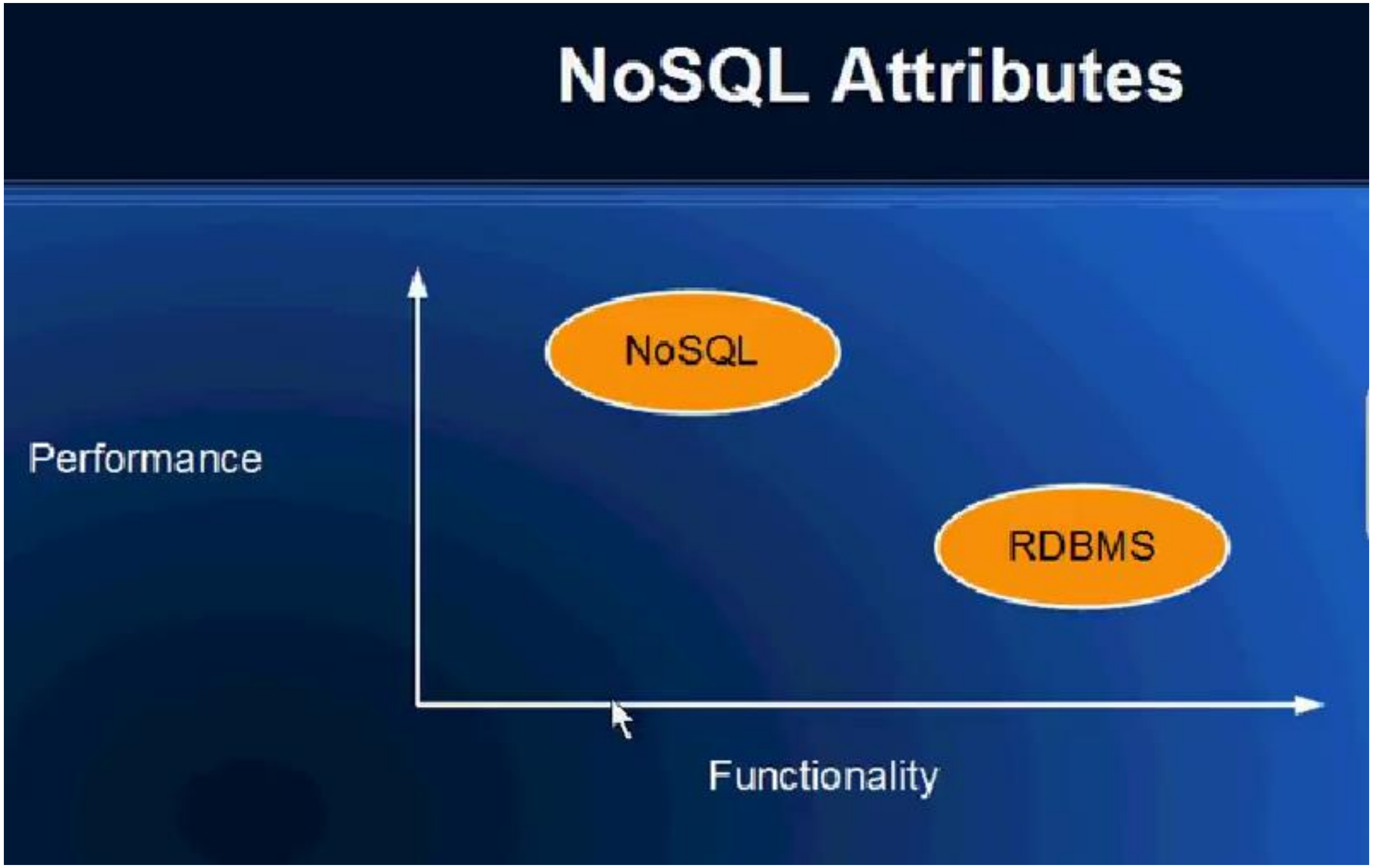


NoSQL – What is Available?

Query Language (Other than SQL)

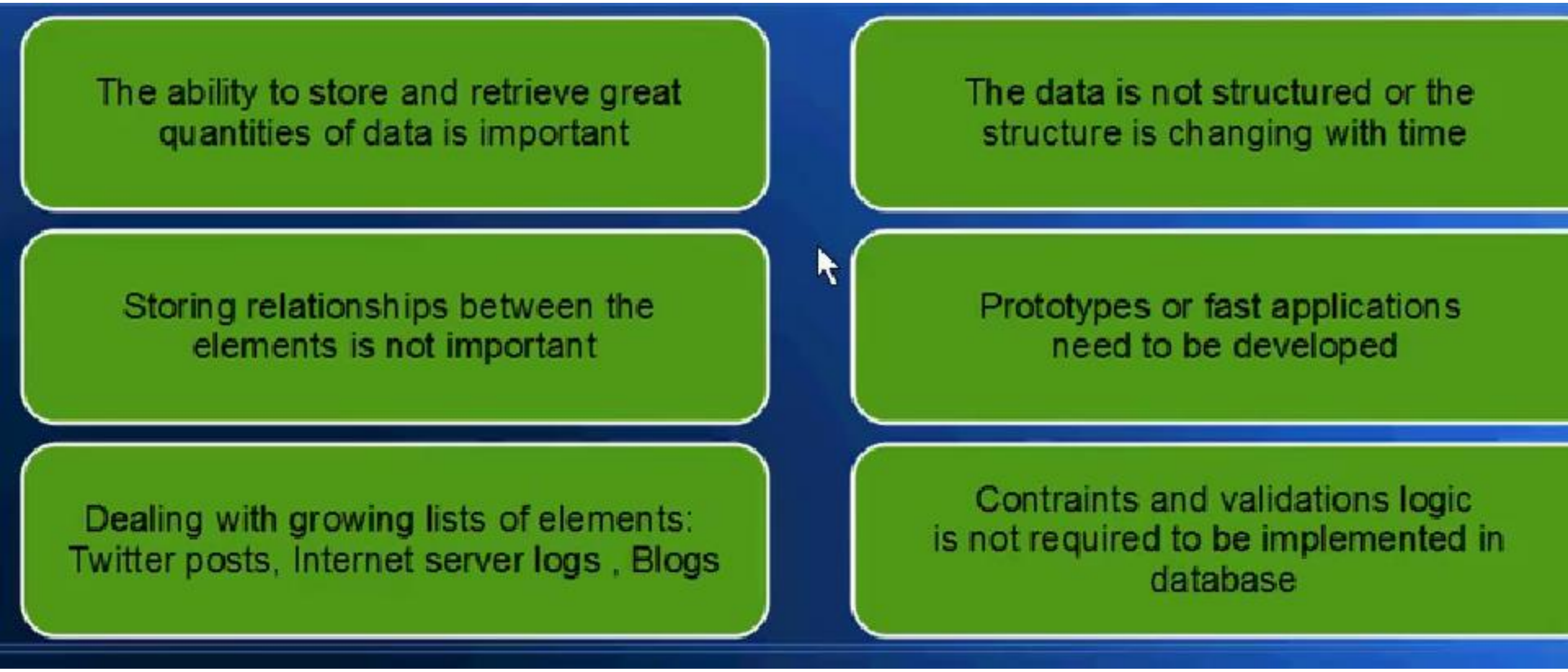
Fast Performance

Horizontal Scalability



NoSQL Attributes

When to use NoSQL



The ability to store and retrieve great quantities of data is important

The data is not structured or the structure is changing with time

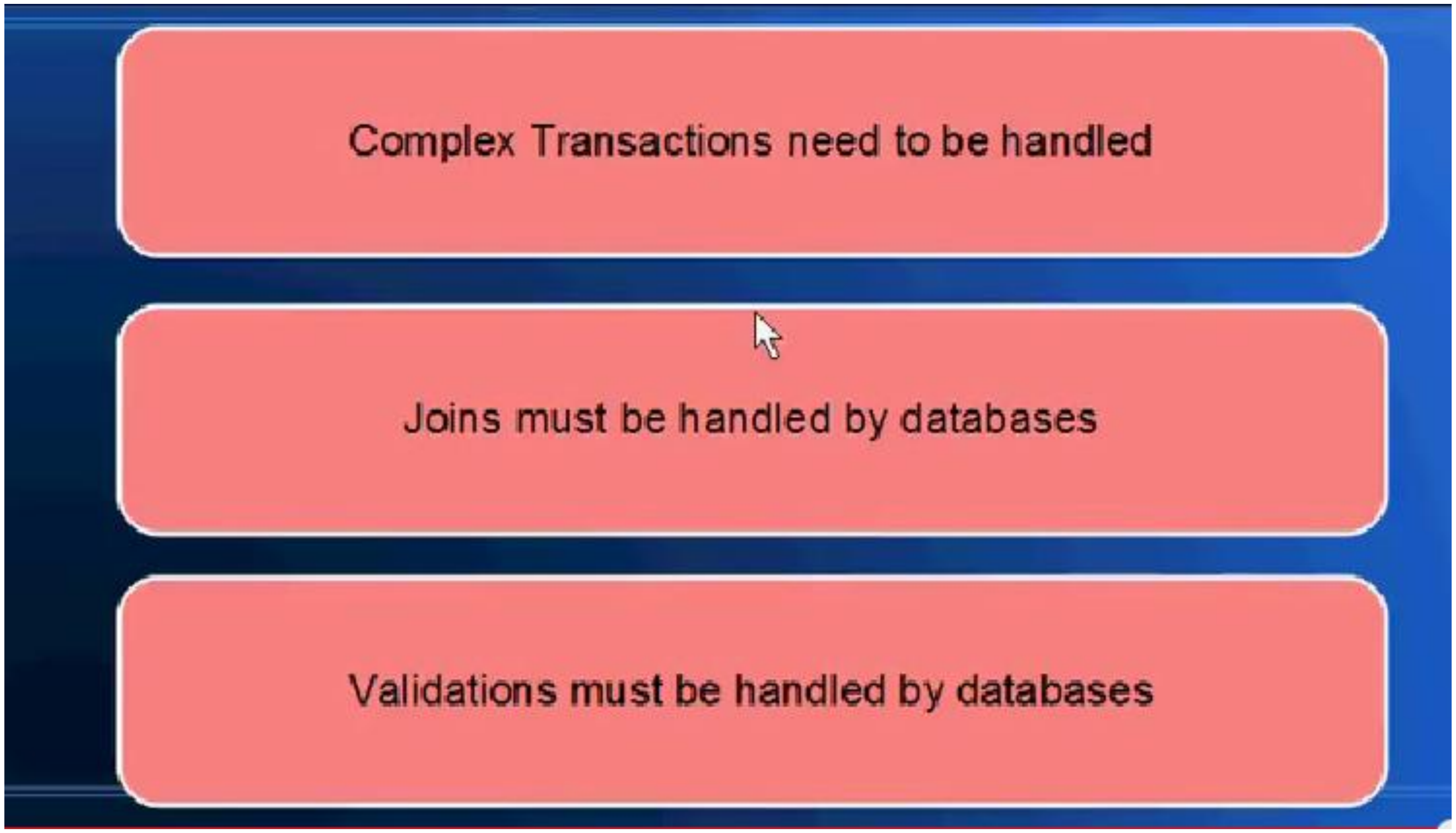
Storing relationships between the elements is not important

Prototypes or fast applications need to be developed

Dealing with growing lists of elements: Twitter posts, Internet server logs , Blogs

Contraints and validations logic is not required to be implemented in database

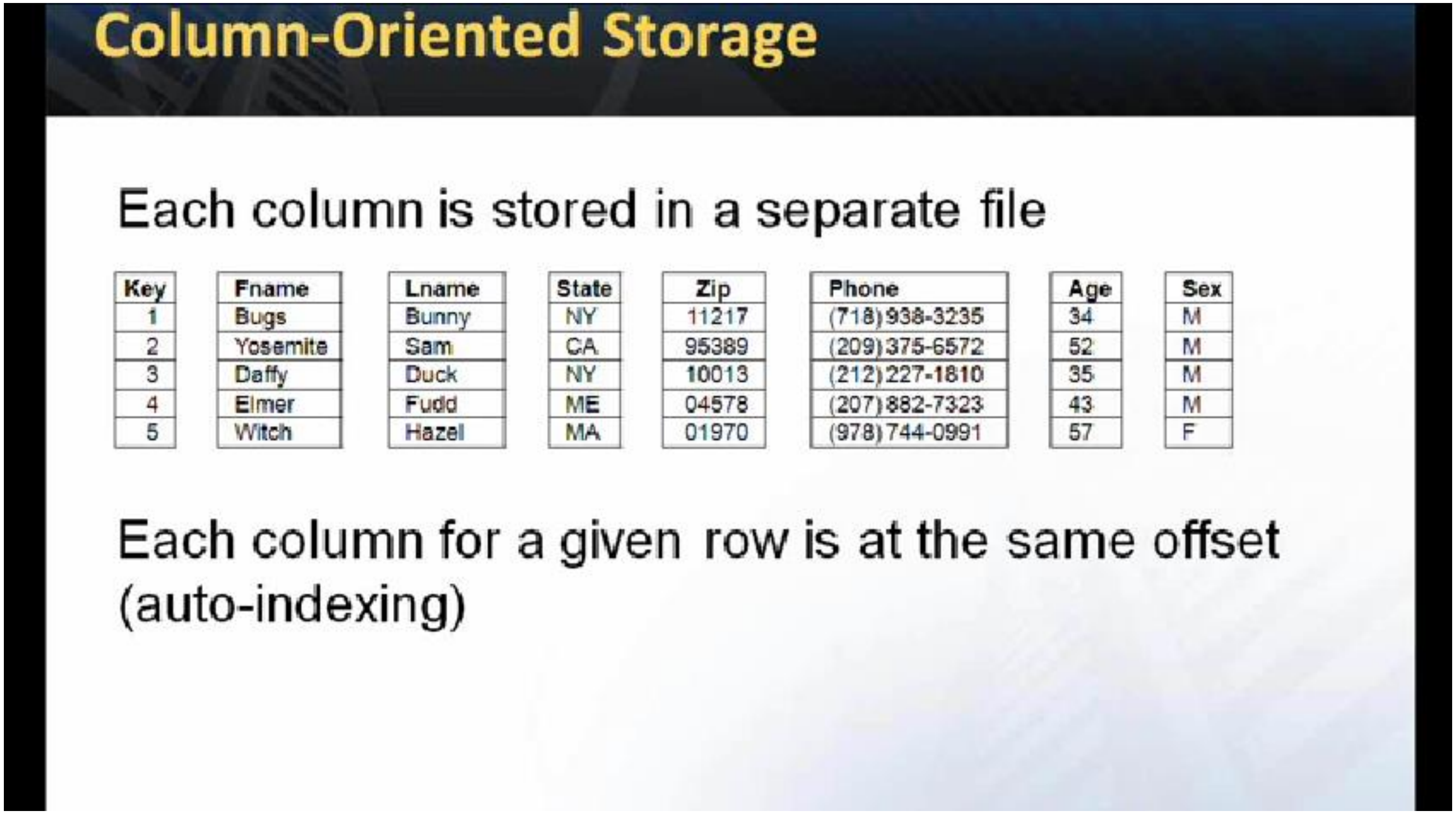
When not to use NoSQL



Complex Transactions need to be handled

Joins must be handled by databases

Validations must be handled by databases



Column-Oriented Storage

Each column is stored in a separate file

Key
1
2
3
4
5

Fname
Bugs
Yosemite
Daffy
Elmer
Witch

Lname
Bunny
Sam
Duck
Fudd
Hazel

State
NY
CA
NY
ME
MA

Zip
11217
95389
10013
04578
01970

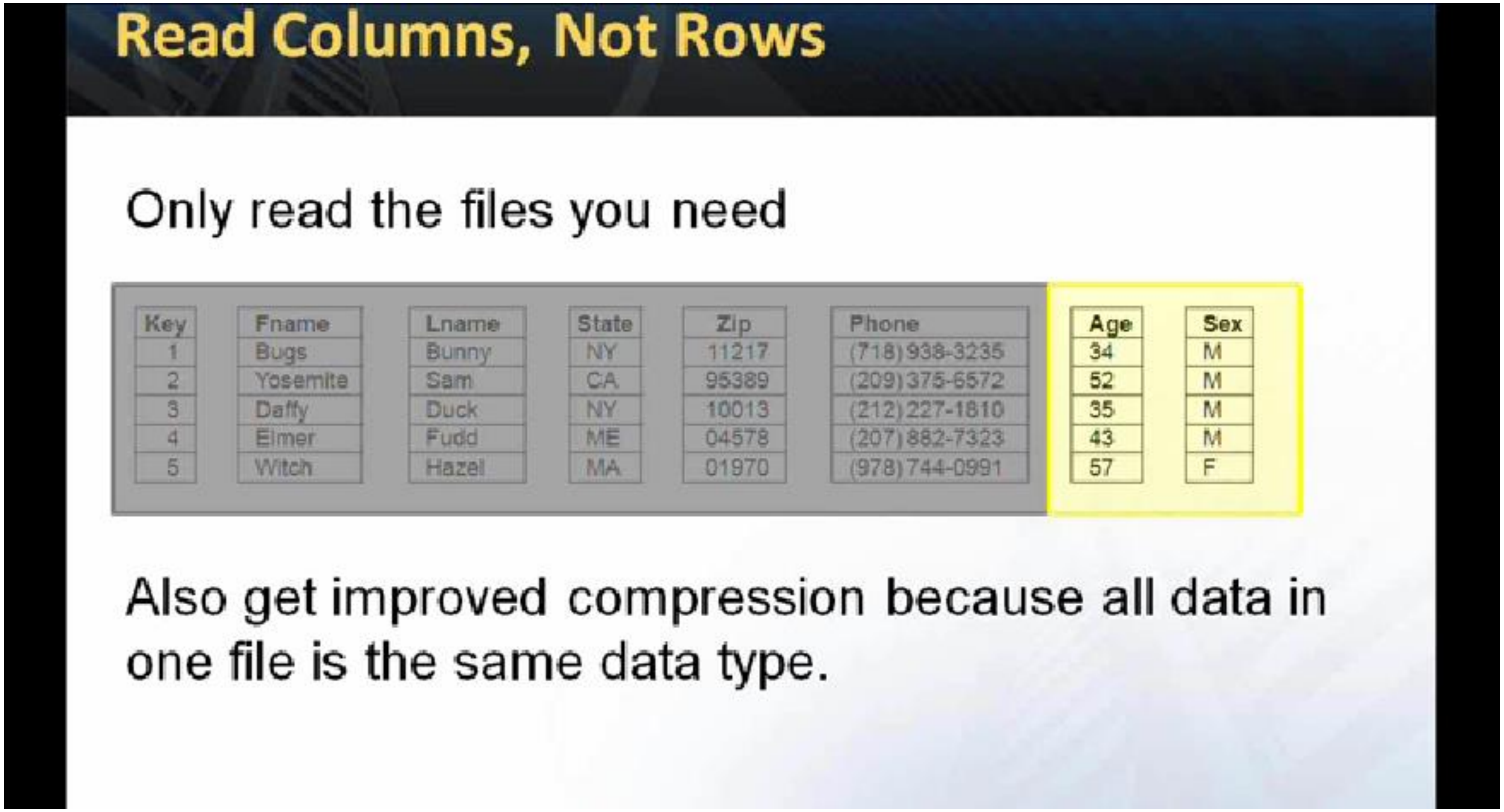
Phone
(718) 938-3235
(209) 375-6572
(212) 227-1810
(207) 882-7323
(978) 744-0991

Age
34
52
35
43
57

Sex
M
M
M
M
F

Each column for a given row is at the same offset (auto-indexing)

e.g. InfiniDB (e.g. find average age of all Male customers)

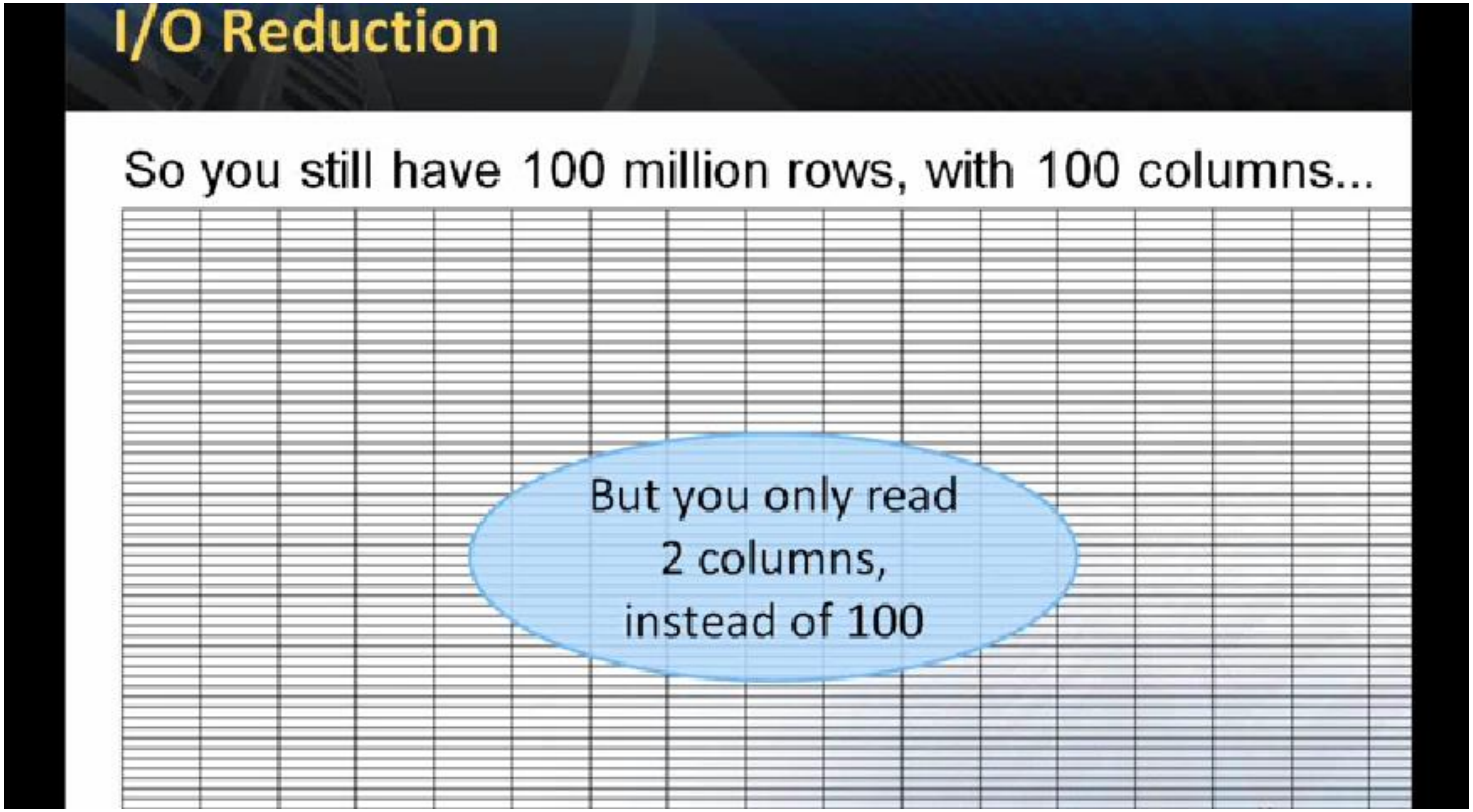


Read Columns, Not Rows

Only read the files you need

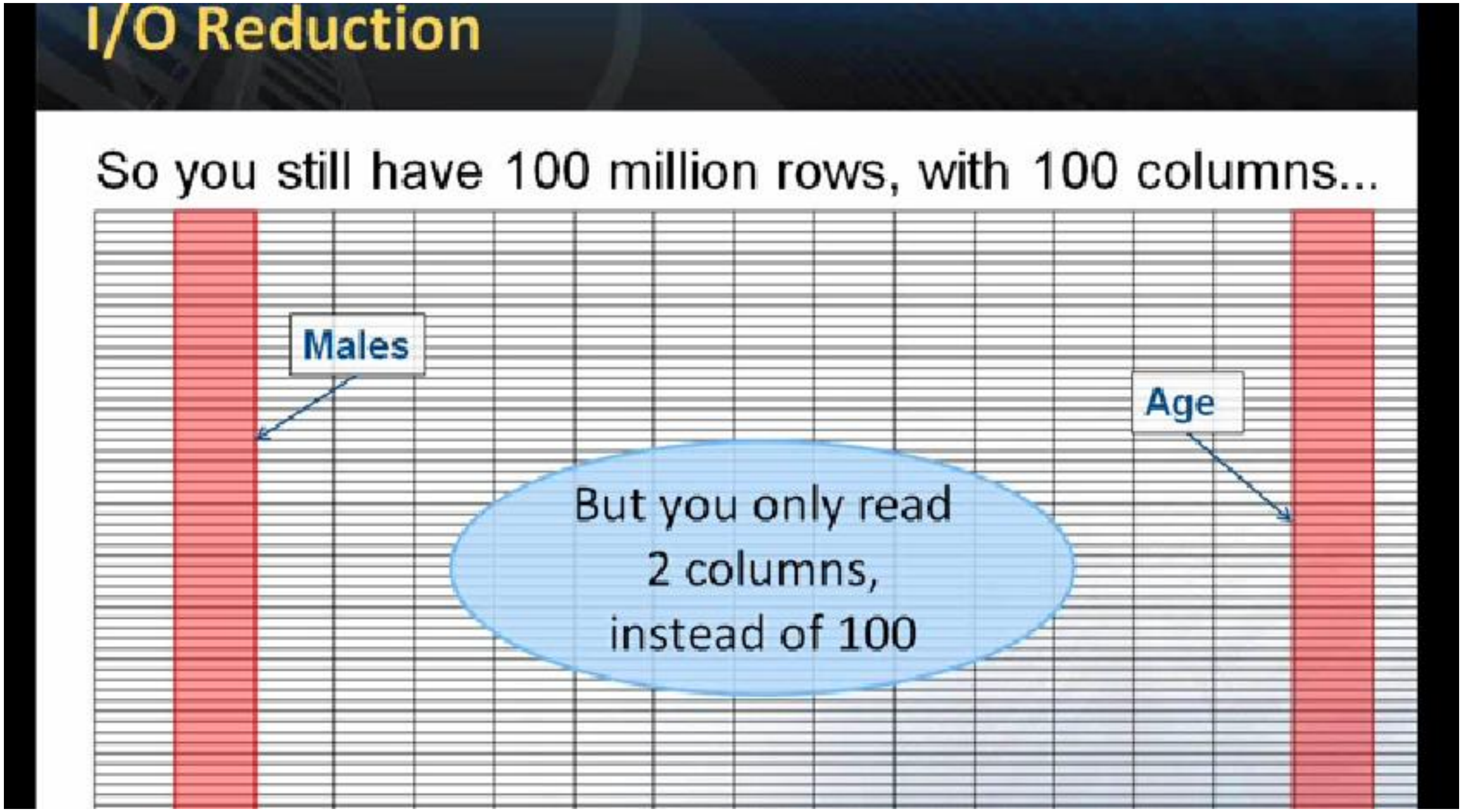
Key	Fname	Lname	State	Zip	Phone	Age	Sex
1	Bugs	Bunny	NY	11217	(718) 938-3235	34	M
2	Yosemite	Sam	CA	95389	(209) 375-6572	52	M
3	Daffy	Duck	NY	10013	(212) 227-1810	35	M
4	Elmer	Fudd	ME	04578	(207) 882-7323	43	M
5	Witch	Hazel	MA	01970	(978) 744-0991	57	F

Also get improved compression because all data in one file is the same data type.



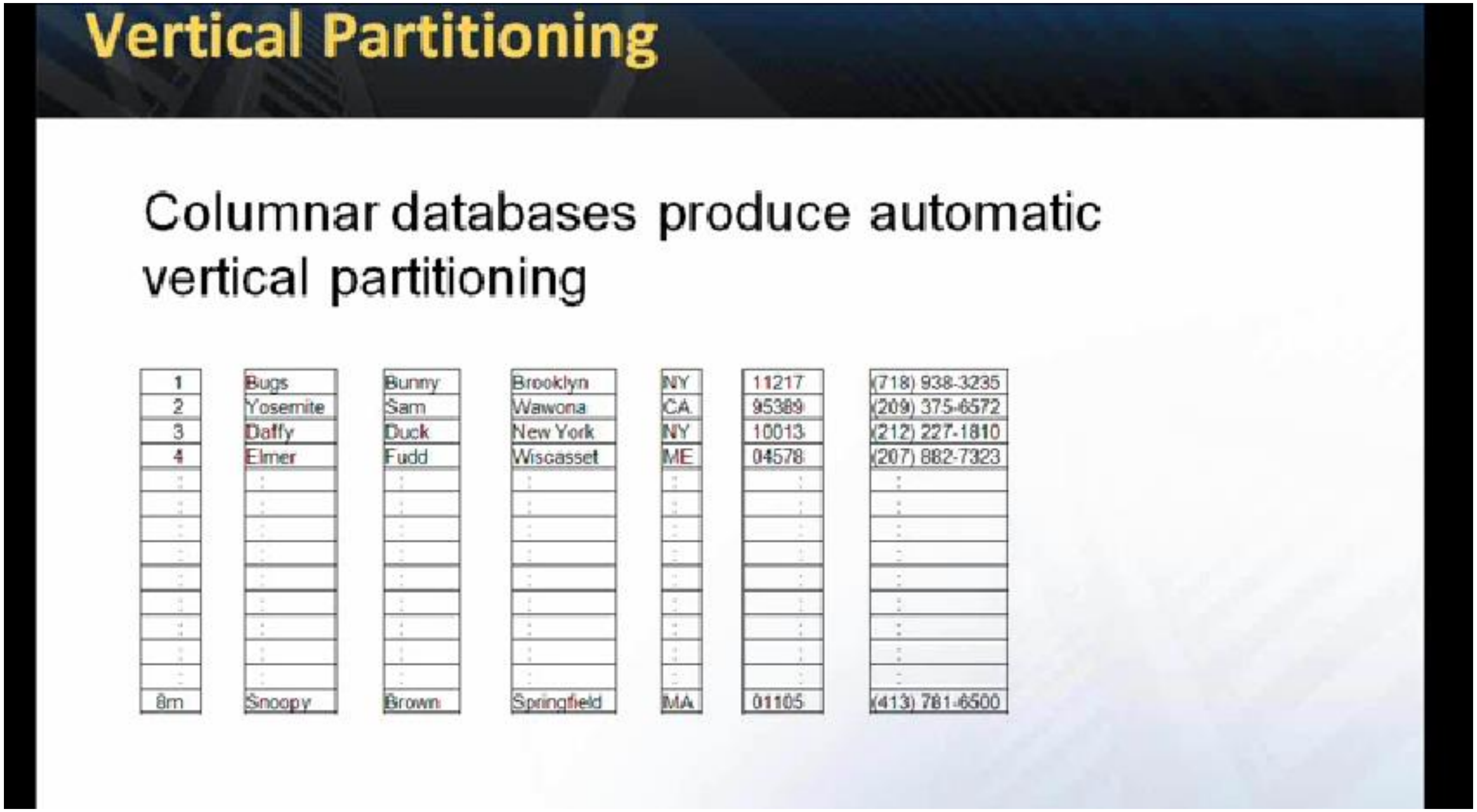
I/O Reduction

So you still have 100 million rows, with 100 columns...



I/O Reduction

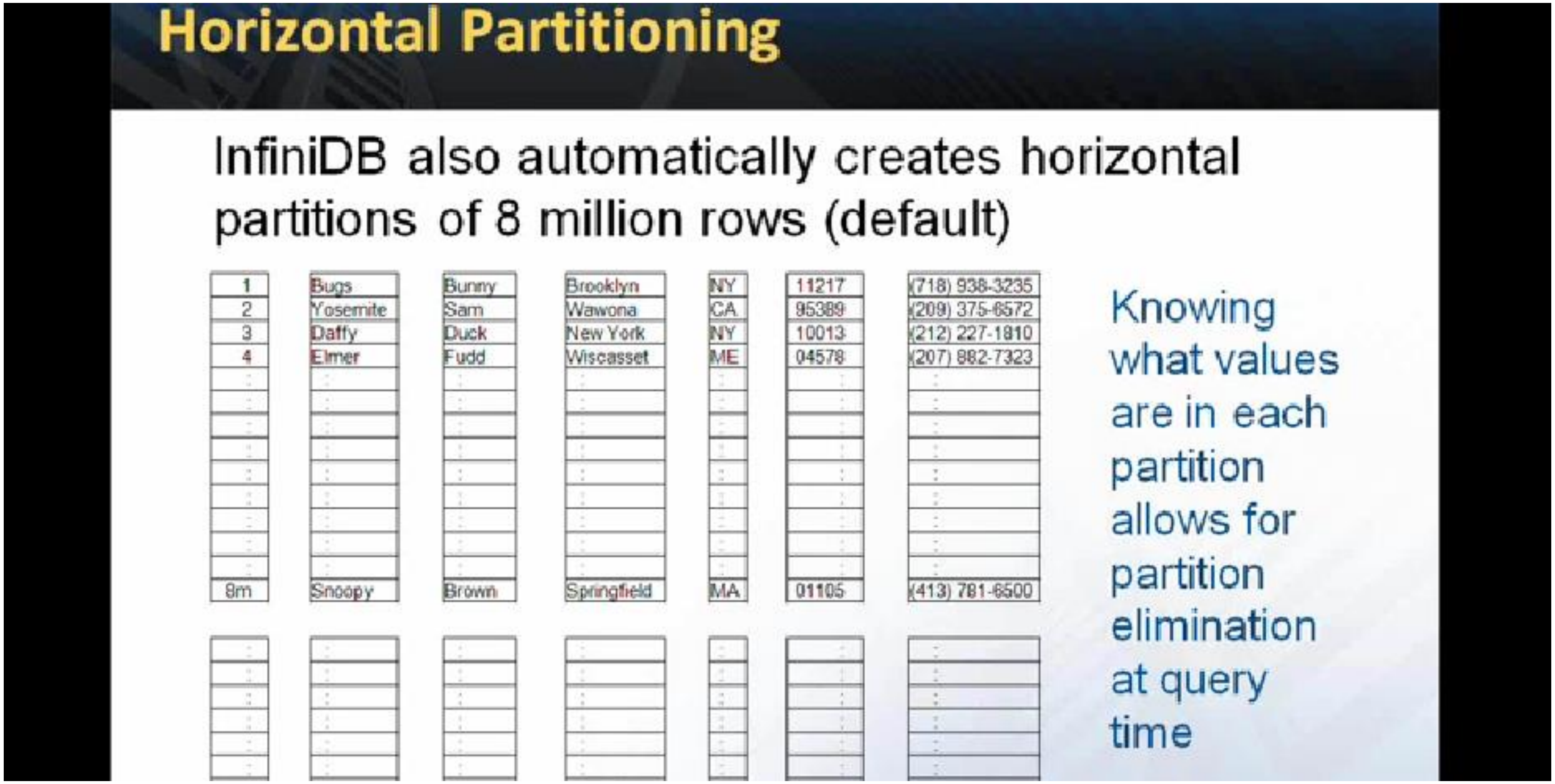
So you still have 100 million rows, with 100 columns...



Vertical Partitioning

Columnar databases produce automatic vertical partitioning

1	Bugs	Bunny	Brooklyn	NY	11217	(718) 938-3235
2	Yosemite	Sam	Wawona	CA	95389	(209) 375-6572
3	Daffy	Duck	New York	NY	10013	(212) 227-1810
4	Elmer	Fudd	Wiscasset	ME	04578	(207) 882-7323
⋮	⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮	⋮
8m	Snoopy	Brown	Springfield	MA	01105	(413) 781-6500

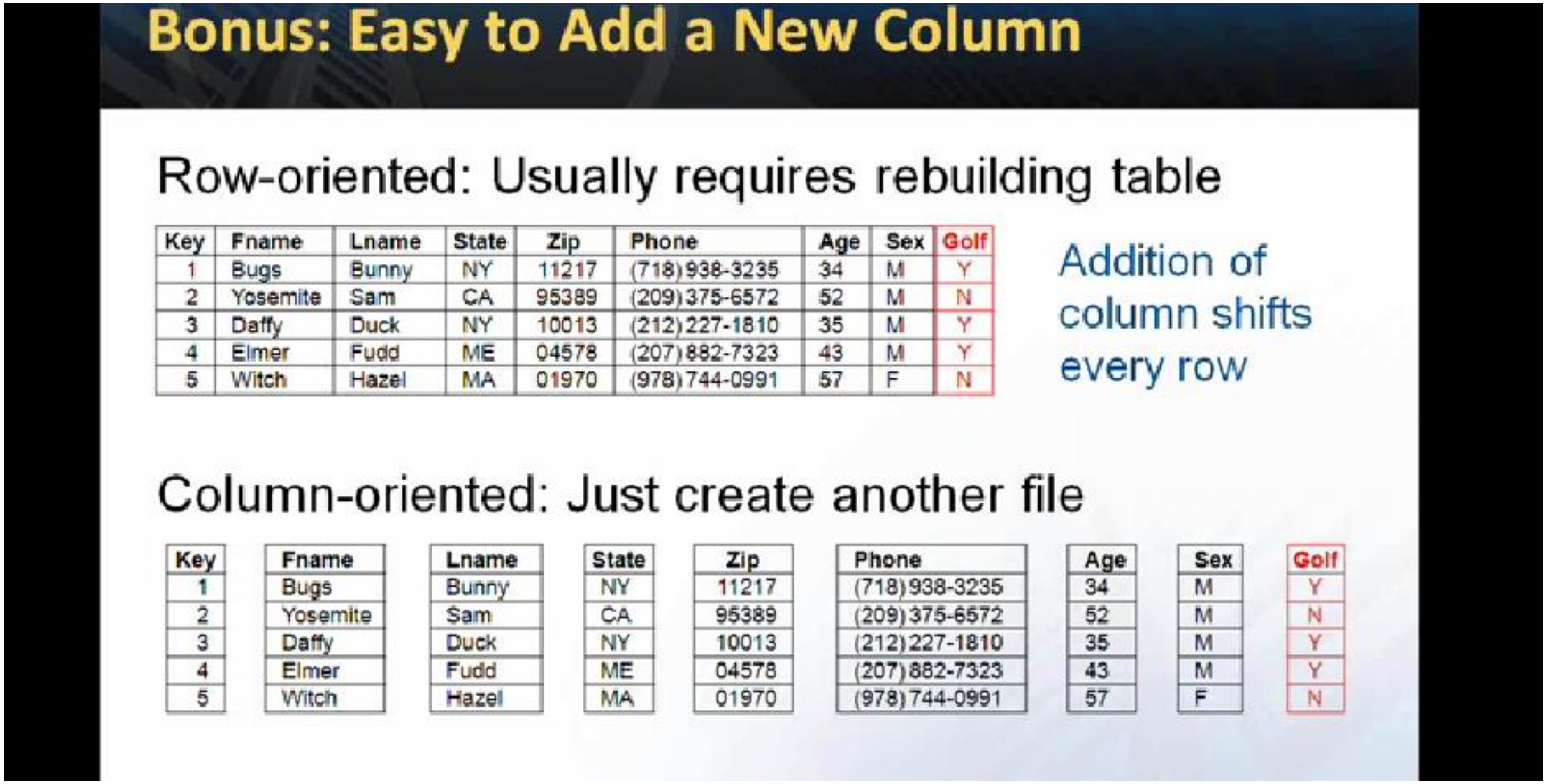


Horizontal Partitioning

InfiniDB also automatically creates horizontal partitions of 8 million rows (default)

1	Bugs	Bunny	Brooklyn	NY	11217	(718) 938-3235
2	Yosemite	Sam	Wawona	CA	95389	(209) 375-6572
3	Daffy	Duck	New York	NY	10013	(212) 227-1810
4	Elmer	Fudd	Wiscasset	ME	04578	(207) 882-7323
8m	Snoopy	Brown	Springfield	MA	01105	(413) 781-6500

Knowing what values are in each partition allows for partition elimination at query time



Bonus: Easy to Add a New Column

Row-oriented: Usually requires rebuilding table

Key	Fname	Lname	State	Zip	Phone	Age	Sex	Golf
1	Bugs	Bunny	NY	11217	(718)938-3235	34	M	Y
2	Yosemite	Sam	CA	95389	(209)375-6572	52	M	N
3	Daffy	Duck	NY	10013	(212)227-1810	35	M	Y
4	Elmer	Fudd	ME	04578	(207)882-7323	43	M	Y
5	Witch	Hazel	MA	01970	(978)744-0991	57	F	N

Addition of column shifts every row

Column-oriented: Just create another file

Key
1
2
3
4
5

Fname
Bugs
Yosemite
Daffy
Elmer
Witch

Lname
Bunny
Sam
Duck
Fudd
Hazel

State
NY
CA
NY
ME
MA

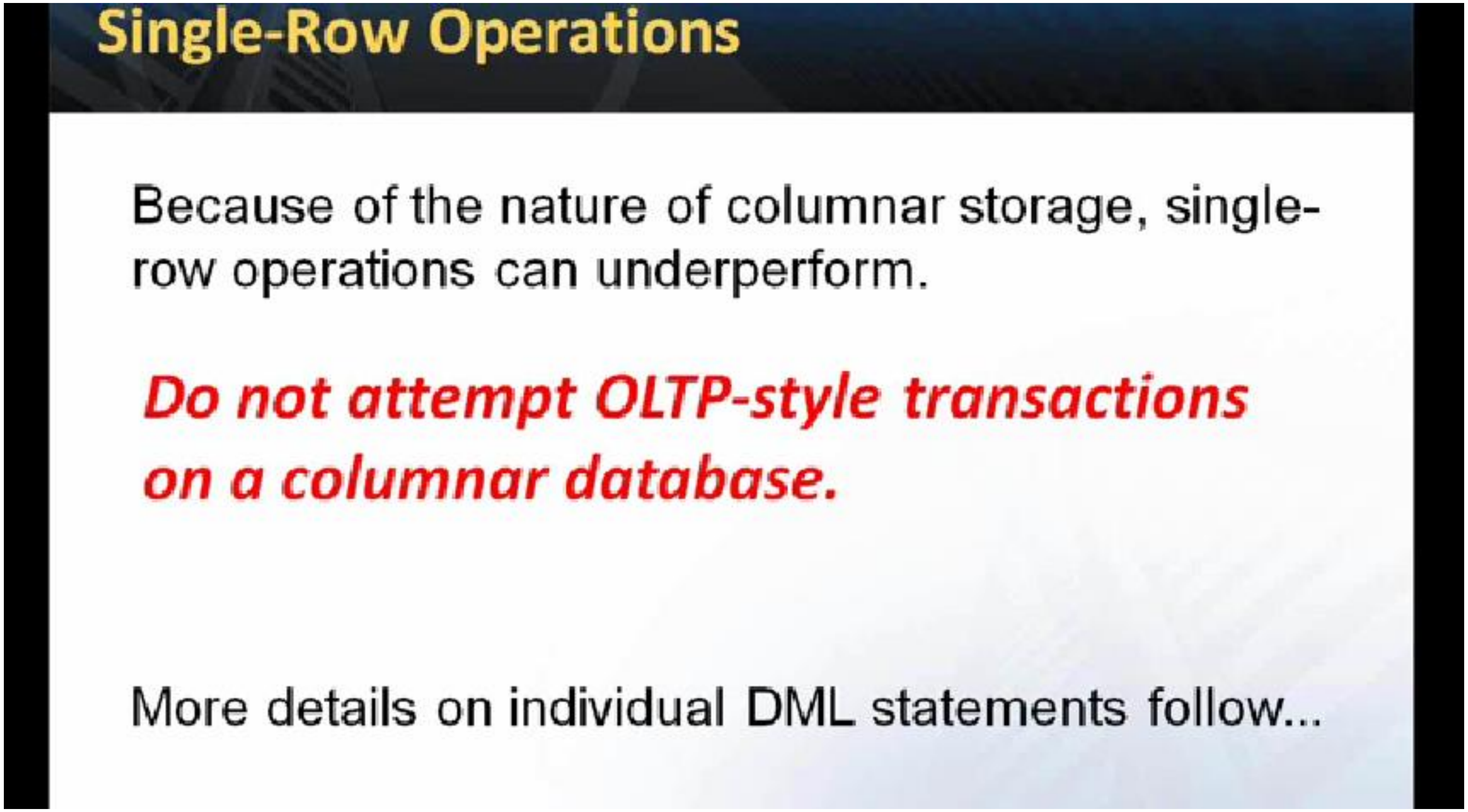
Zip
11217
95389
10013
04578
01970

Phone
(718)938-3235
(209)375-6572
(212)227-1810
(207)882-7323
(978)744-0991

Age
34
52
35
43
57

Sex
M
M
M
M
F

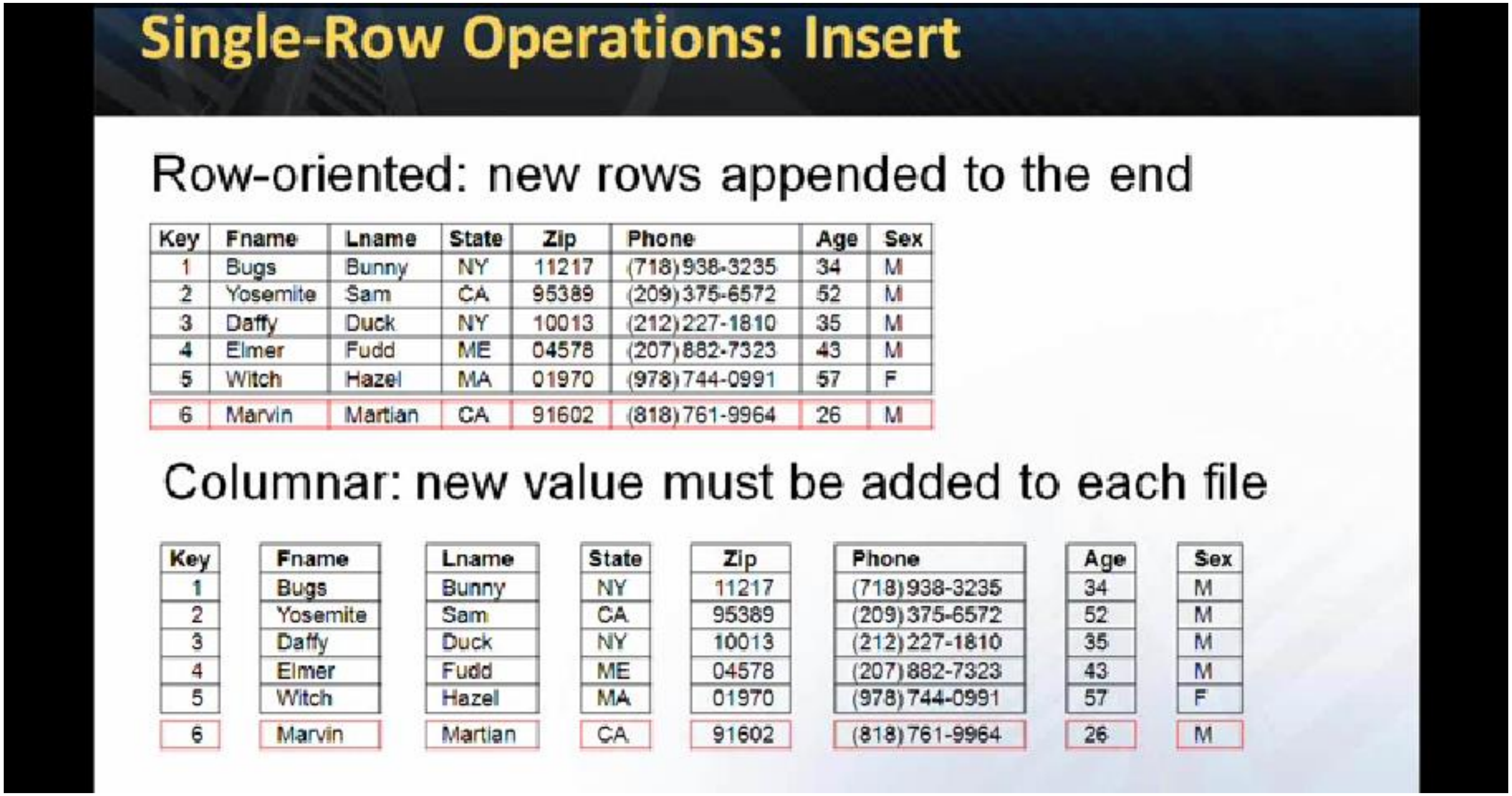
Golf
Y
N
Y
Y
N


Single-Row Operations

Because of the nature of columnar storage, single-row operations can underperform.

Do not attempt OLTP-style transactions on a columnar database.

More details on individual DML statements follow...



Single-Row Operations: Insert

Row-oriented: new rows appended to the end

Key	Fname	Lname	State	Zip	Phone	Age	Sex
1	Bugs	Bunny	NY	11217	(718) 938-3235	34	M
2	Yosemite	Sam	CA	95389	(209) 375-6572	52	M
3	Daffy	Duck	NY	10013	(212) 227-1810	35	M
4	Elmer	Fudd	ME	04578	(207) 882-7323	43	M
5	Witch	Hazel	MA	01970	(978) 744-0991	57	F
6	Marvin	Martian	CA	91602	(818) 761-9964	26	M

Columnar: new value must be added to each file

Key
1
2
3
4
5
6

Fname
Bugs
Yosemite
Daffy
Elmer
Witch
Marvin

Lname
Bunny
Sam
Duck
Fudd
Hazel
Martian

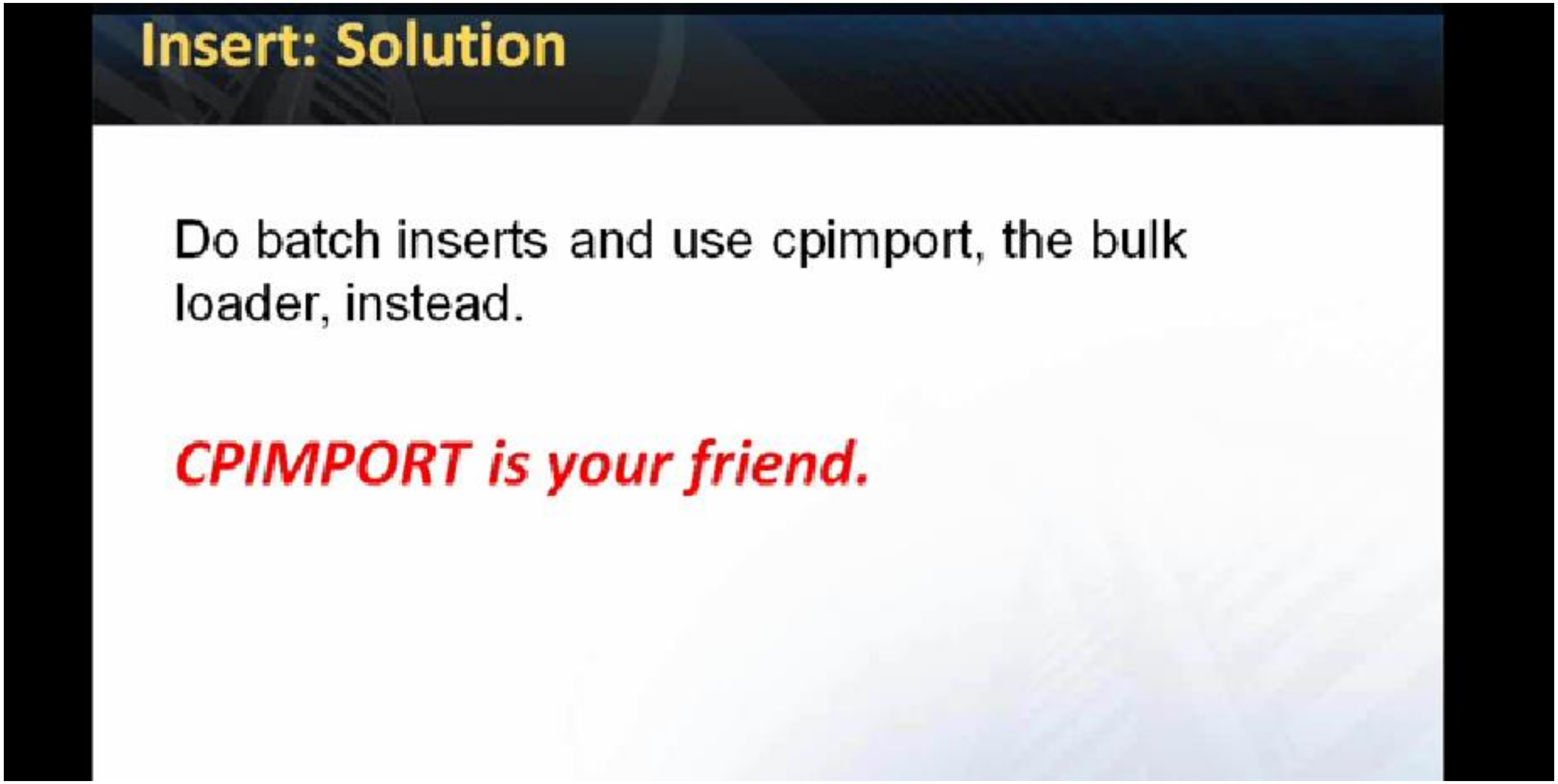
State
NY
CA
NY
ME
MA
CA

Zip
11217
95389
10013
04578
01970
91602

Phone
(718) 938-3235
(209) 375-6572
(212) 227-1810
(207) 882-7323
(978) 744-0991
(818) 761-9964

Age
34
52
35
43
57
26

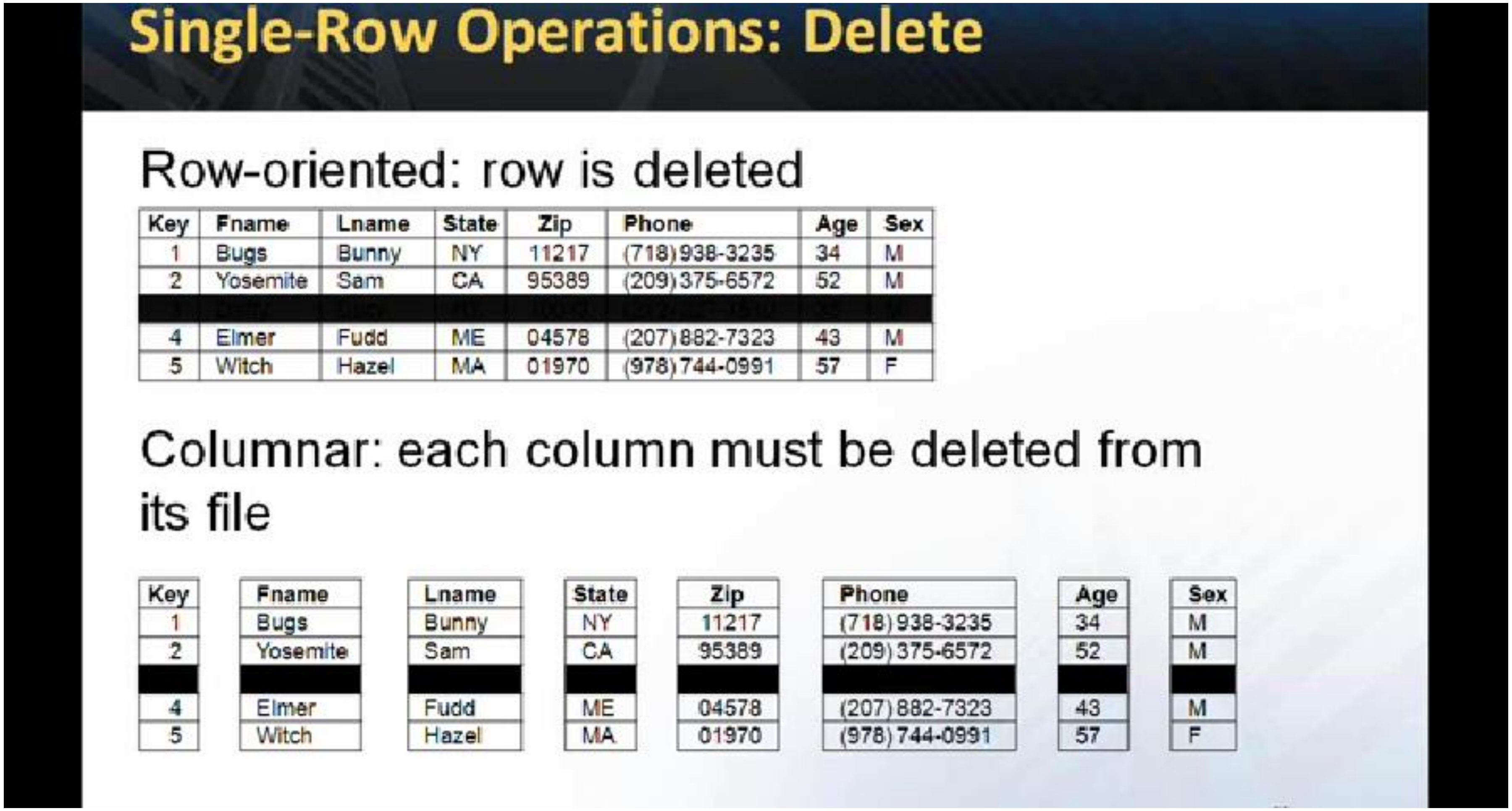
Sex
M
M
M
M
F
M



Insert: Solution

Do batch inserts and use cpimport, the bulk loader, instead.

CPIMPORT is your friend.



Single-Row Operations: Delete

Row-oriented: row is deleted

Key	Fname	Lname	State	Zip	Phone	Age	Sex
1	Bugs	Bunny	NY	11217	(718)938-3235	34	M
2	Yosemite	Sam	CA	95389	(209)375-6572	52	M
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
4	Elmer	Fudd	ME	04578	(207)882-7323	43	M
5	Witch	Hazel	MA	01970	(978)744-0991	57	F

Columnar: each column must be deleted from its file

Key
1
2
[redacted]
4
5

Fname
Bugs
Yosemite
[redacted]
Elmer
Witch

Lname
Bunny
Sam
[redacted]
Fudd
Hazel

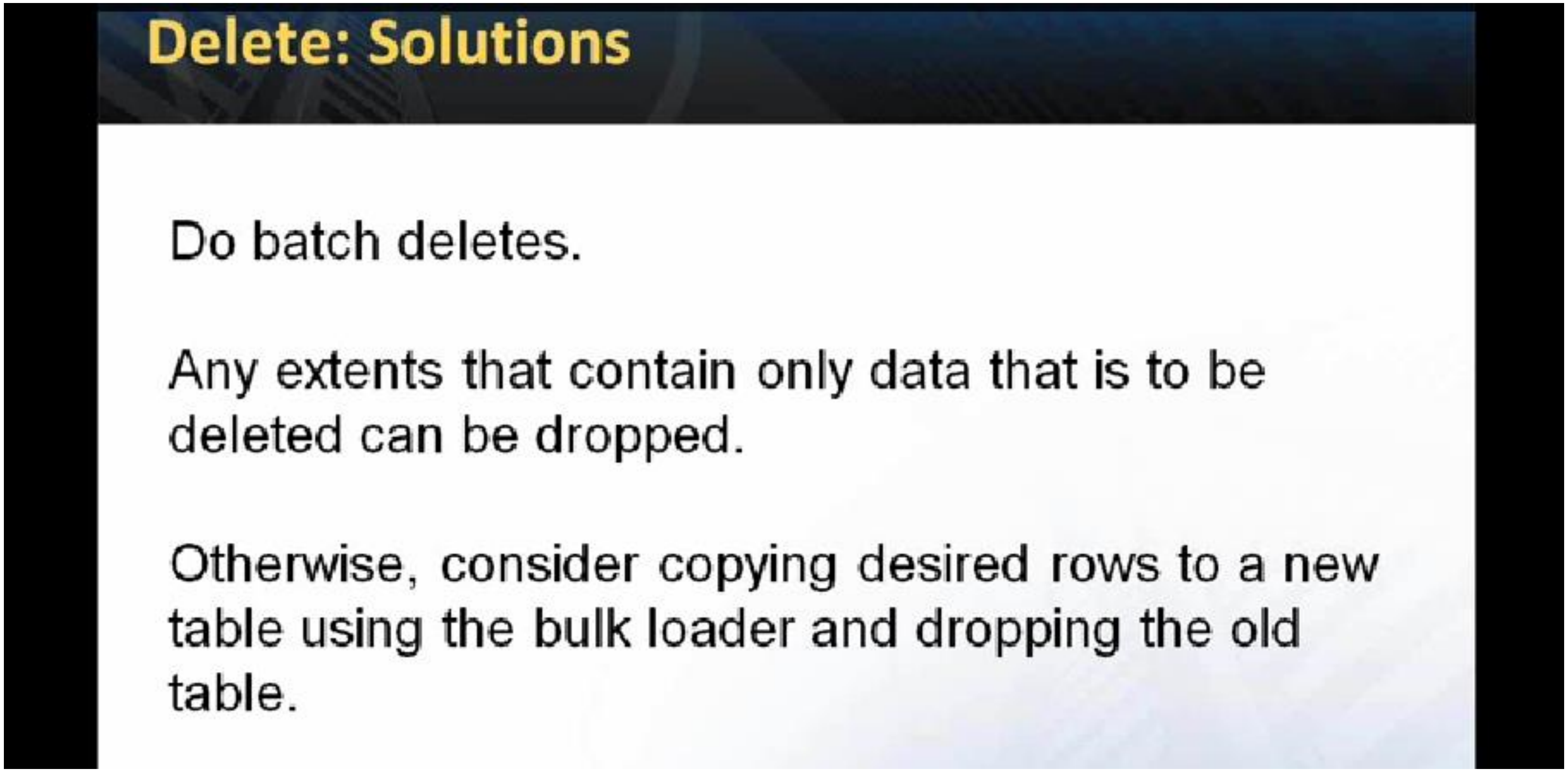
State
NY
CA
[redacted]
ME
MA

Zip
11217
95389
[redacted]
04578
01970

Phone
(718) 938-3235
(209) 375-6572
[redacted]
(207) 882-7323
(978) 744-0991

Age
34
52
[redacted]
43
57

Sex
M
M
[redacted]
M
F

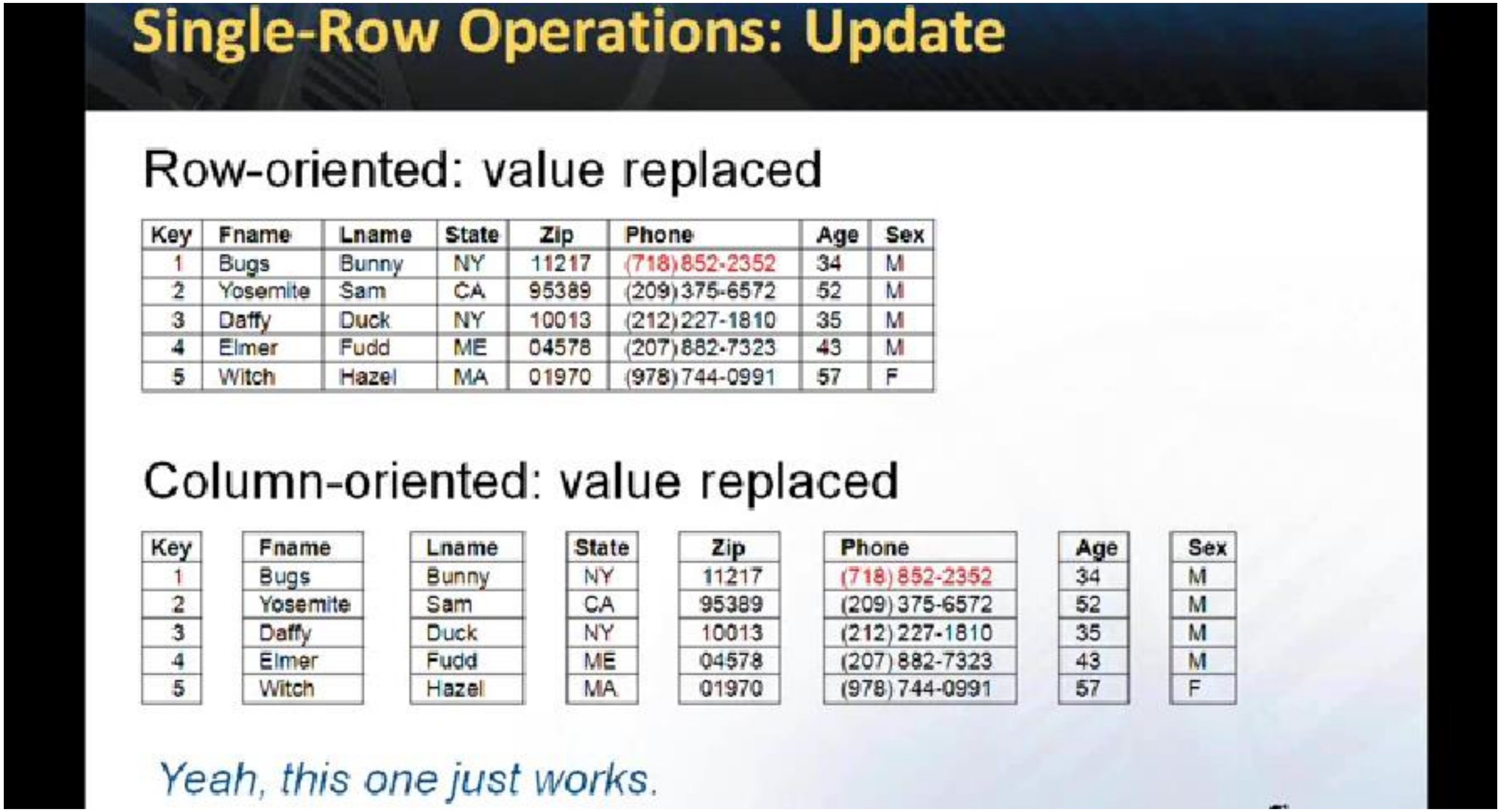


Delete: Solutions

Do batch deletes.

Any extents that contain only data that is to be deleted can be dropped.

Otherwise, consider copying desired rows to a new table using the bulk loader and dropping the old table.



Single-Row Operations: Update

Row-oriented: value replaced

Key	Fname	Lname	State	Zip	Phone	Age	Sex
1	Bugs	Bunny	NY	11217	(718) 852-2352	34	M
2	Yosemite	Sam	CA	95389	(209) 375-6572	52	M
3	Daffy	Duck	NY	10013	(212) 227-1810	35	M
4	Elmer	Fudd	ME	04578	(207) 882-7323	43	M
5	Witch	Hazel	MA	01970	(978) 744-0991	57	F

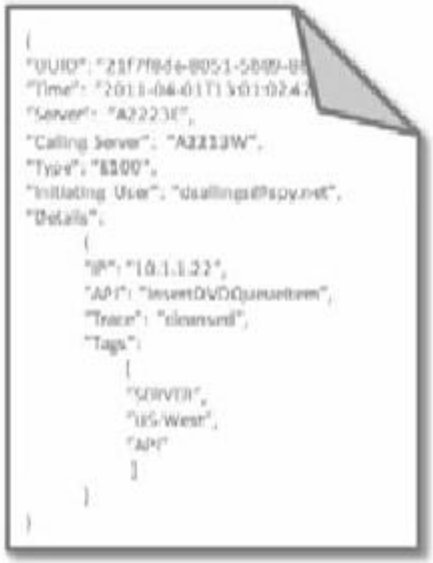
Column-oriented: value replaced

Key	Fname	Lname	State	Zip	Phone	Age	Sex
1	Bugs	Bunny	NY	11217	(718) 852-2352	34	M
2	Yosemite	Sam	CA	95389	(209) 375-6572	52	M
3	Daffy	Duck	NY	10013	(212) 227-1810	35	M
4	Elmer	Fudd	ME	04578	(207) 882-7323	43	M
5	Witch	Hazel	MA	01970	(978) 744-0991	57	F

Yeah, this one just works.

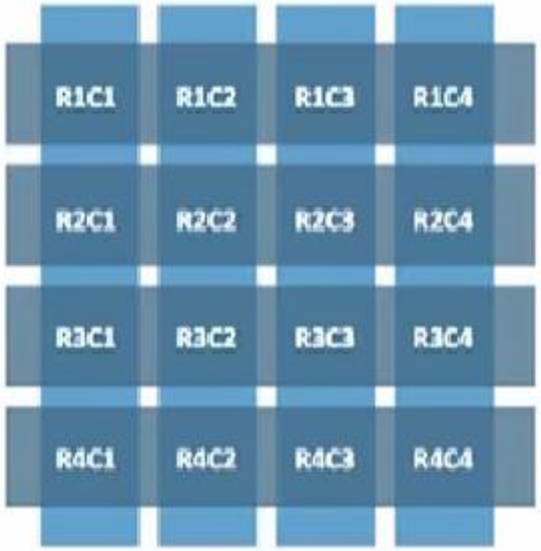
Document Databases

- Each record in the database is a self-describing document
- Each document has an independent structure
- Documents can be complex
- All databases require a unique key
- Documents are stored using JSON or XML or their derivatives
- Content can be indexed and queried
- Offer auto-sharding for scaling and replication for high-availability



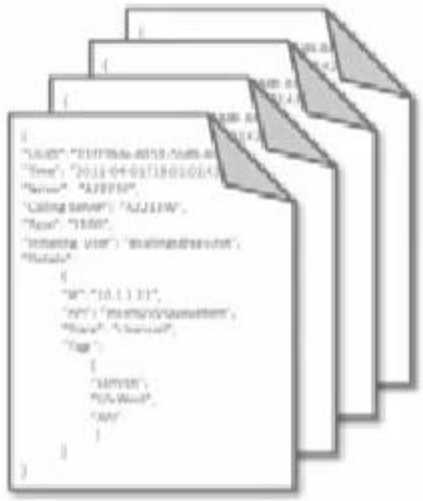
```
{
"UUID": "21f7f8de-8051-5b89-8
"Time": "2011-04-01T13:01:02.4
"Server": "A2223E",
"Calling Server": "A2213W",
"Type": "E100",
"Initiating User": "dsallings@spy.net",
"Details",
        {
        "IP": "10.1.1.22",
        "API": "InsertDVDQueueItem",
        "Trace": "cleansed",
        "Tags":
            [
            "SERVER",
            "US-West",
            "API"
            ]
        }
}
```

Relational vs Document data model



Relational data model

Highly-structured table organization with rigidly-defined data formats and record structure.



Document data model

Collection of complex documents with arbitrary, nested data formats and varying "record" format.

Example: Error Logging Use case

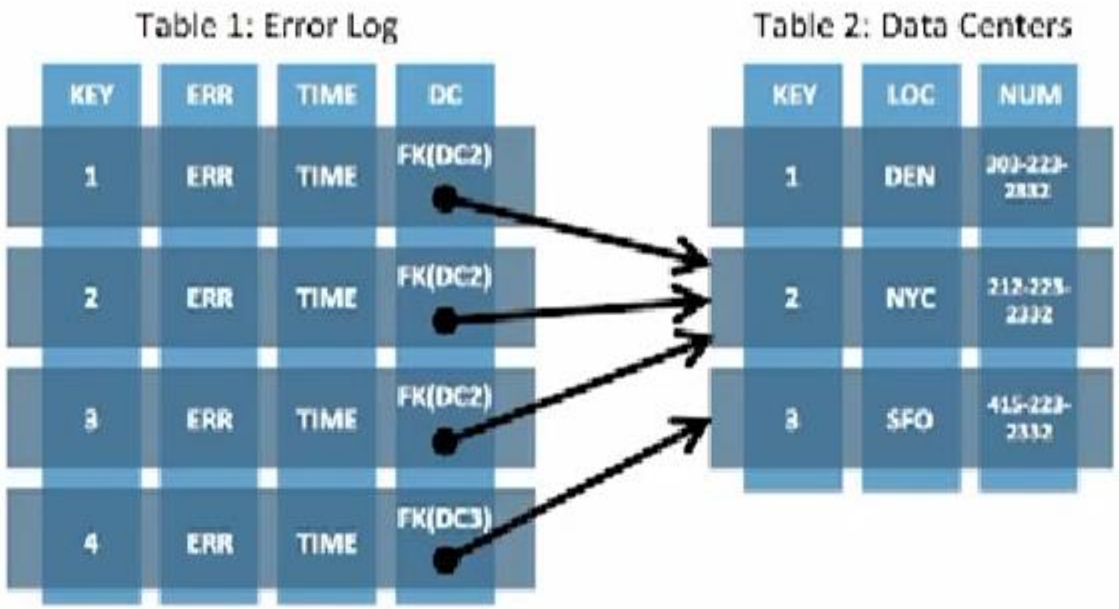


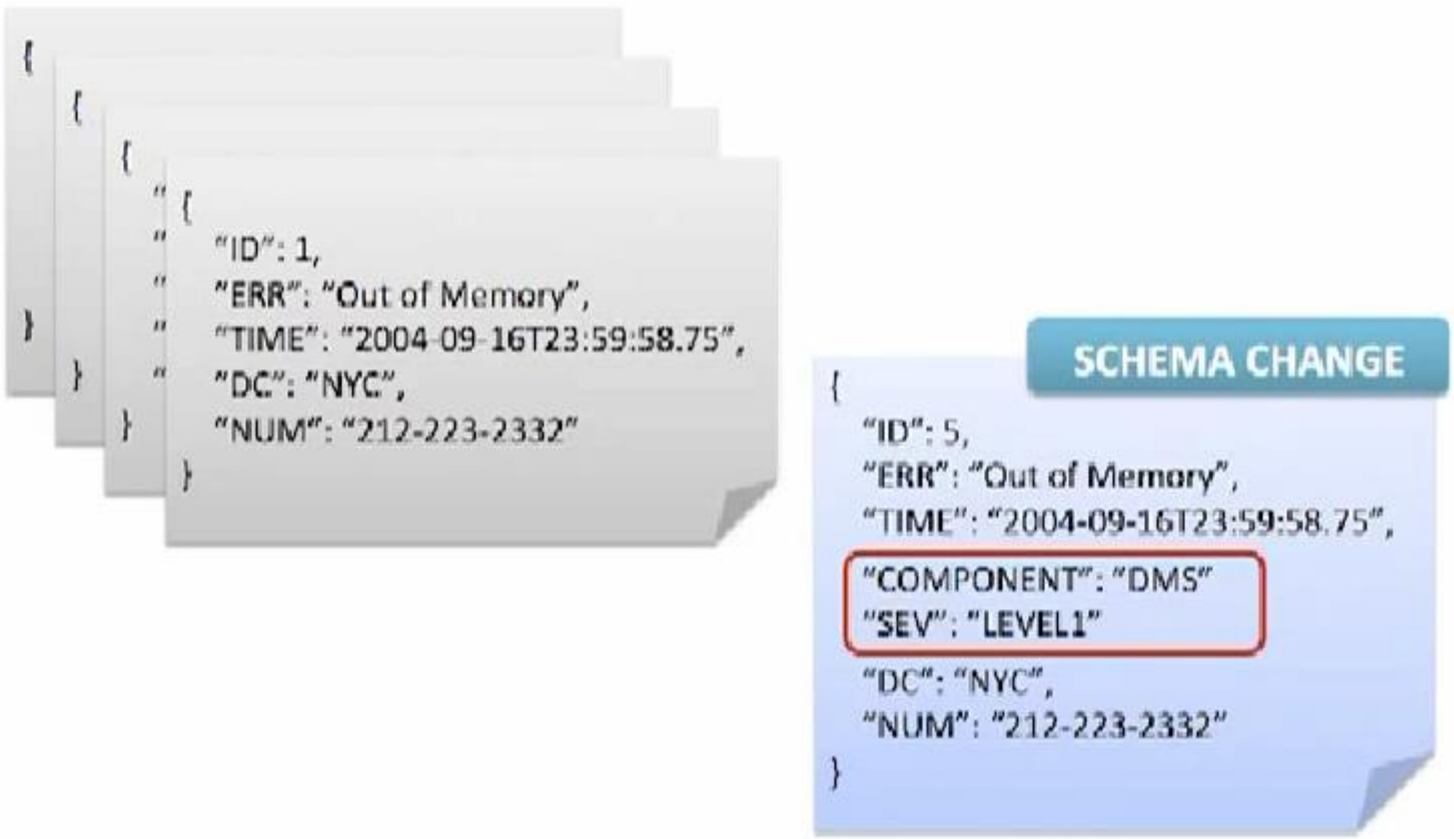
Table 1: Error Log

KEY	ERR	TIME	DC
1	ERR	TIME	FK(DC2)
2	ERR	TIME	FK(DC2)
3	ERR	TIME	FK(DC2)
4	ERR	TIME	FK(DC3)

Table 2: Data Centers

KEY	LOC	NUM
1	DEN	303-223-2332
2	NYC	212-223-2332
3	SFO	415-223-2332

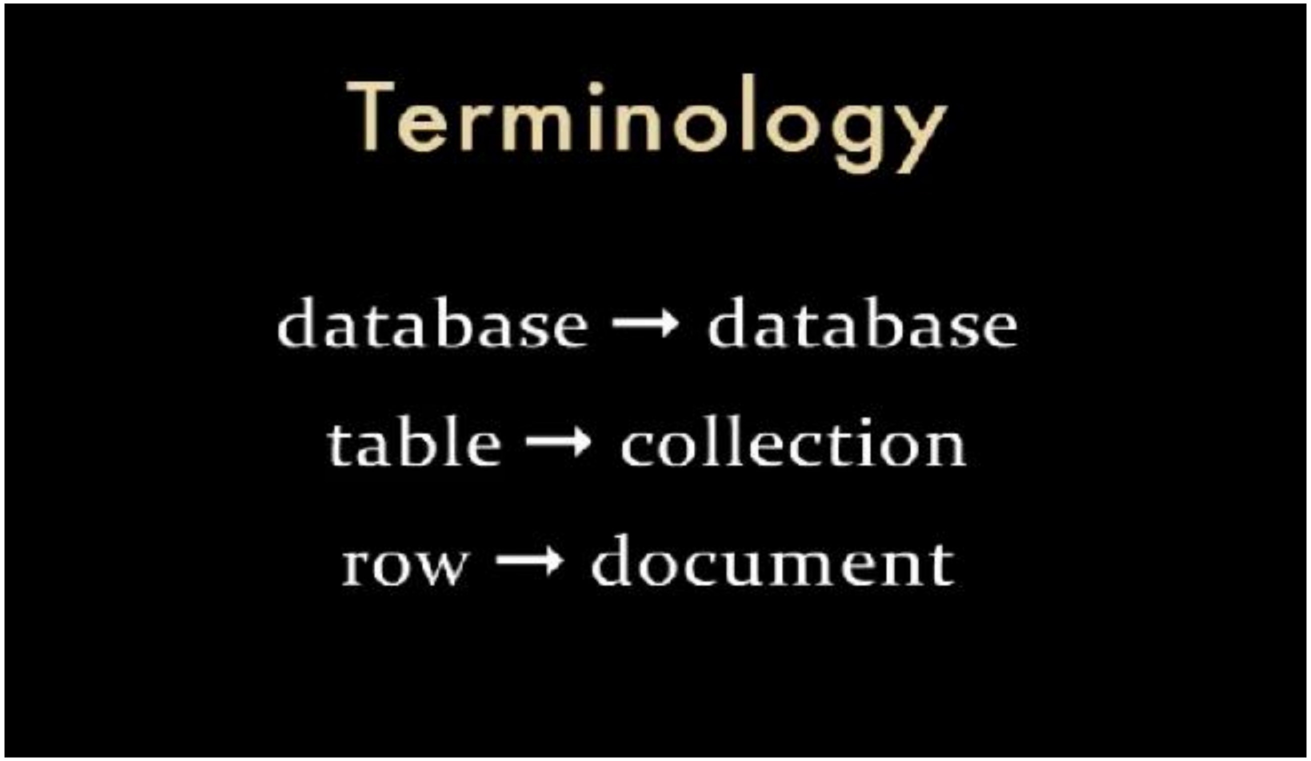
Document design with flexible schema



```
{
  "ID": 1,
  "ERR": "Out of Memory",
  "TIME": "2004-09-16T23:59:58.75",
  "DC": "NYC",
  "NUM": "212-223-2332"
}
```

```
{
  "ID": 5,
  "ERR": "Out of Memory",
  "TIME": "2004-09-16T23:59:58.75",
  "COMPONENT": "DMS"
  "SEV": "LEVEL1"
  "DC": "NYC",
  "NUM": "212-223-2332"
}
```

MongoDB sponsored by 10gen



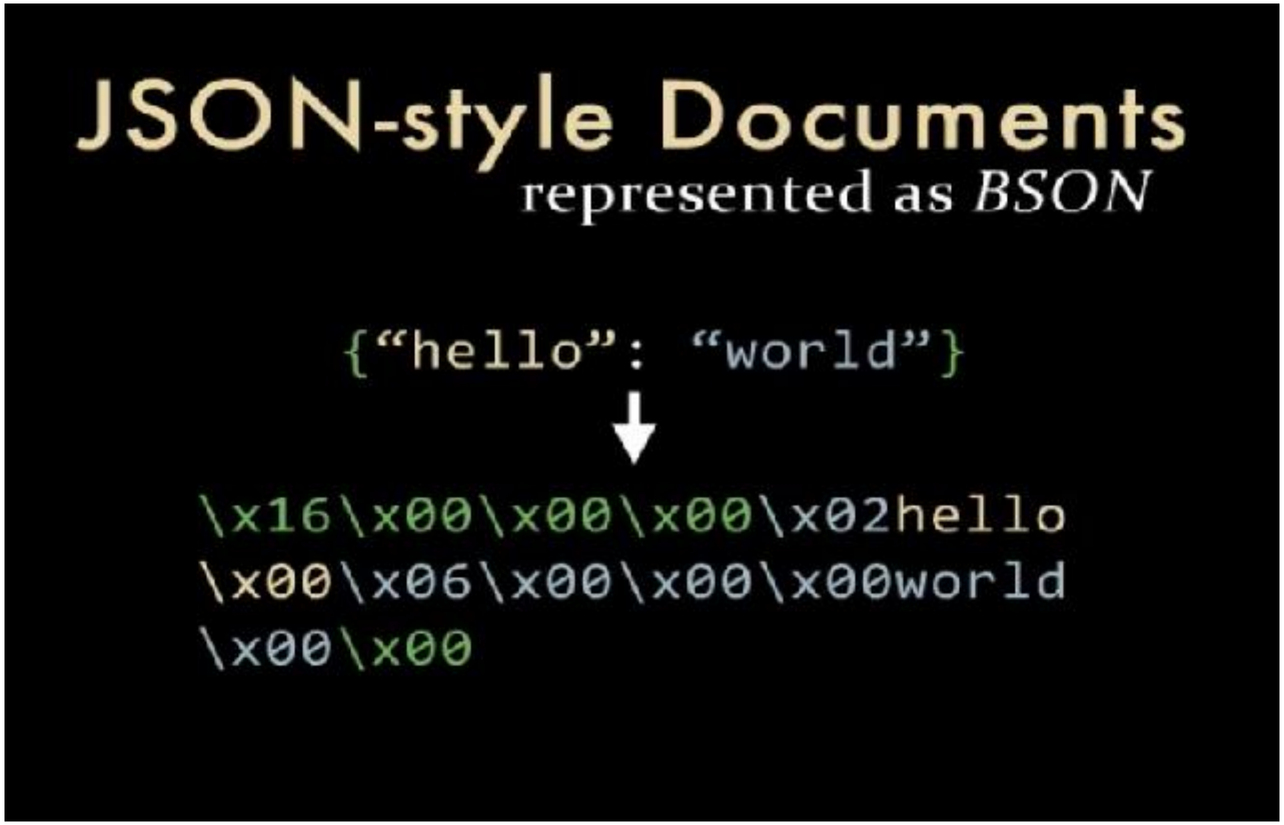
Terminology

database → database

table → collection

row → document

4 MB limit on document size
No limit on nesting depths

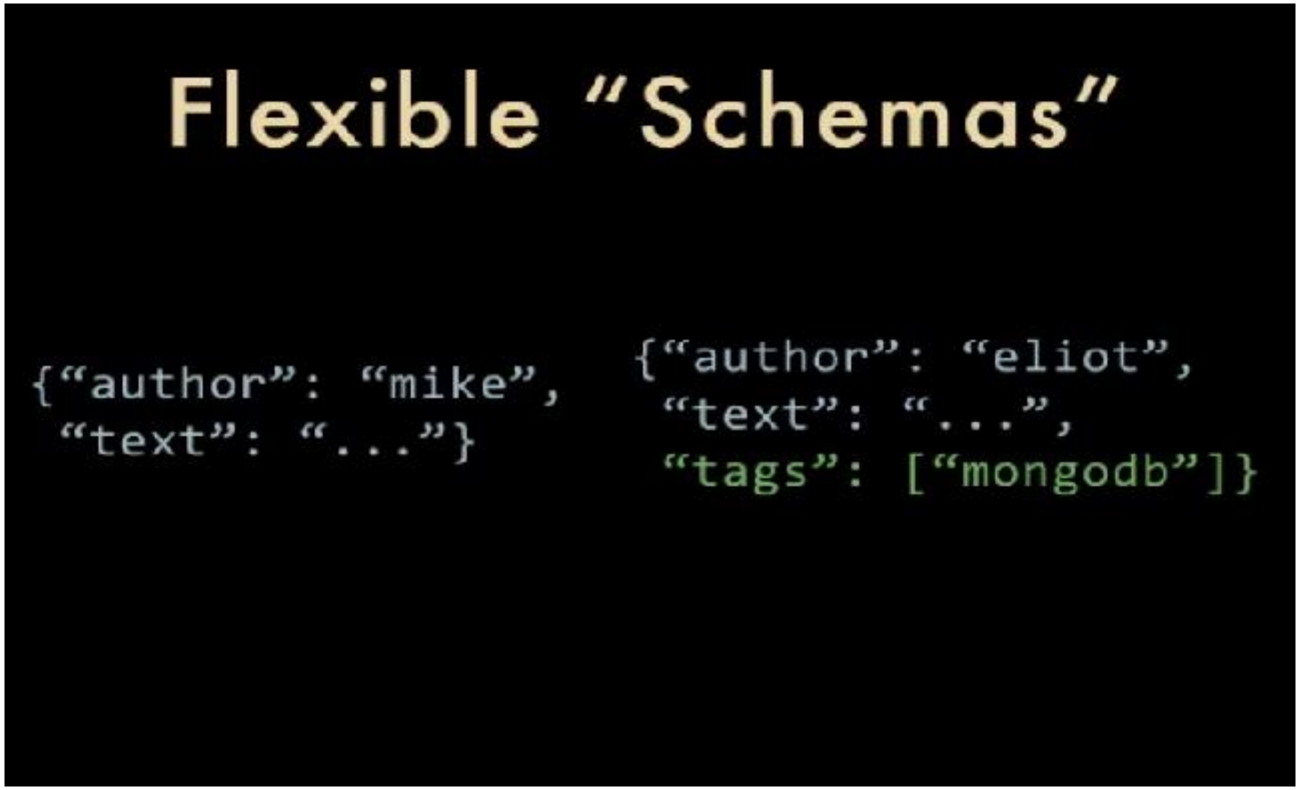


JSON-style Documents represented as BSON

```
{"hello": "world"}
```

```
\x16\x00\x00\x00\x02hello
\x00\x06\x00\x00\x00world
\x00\x00
```

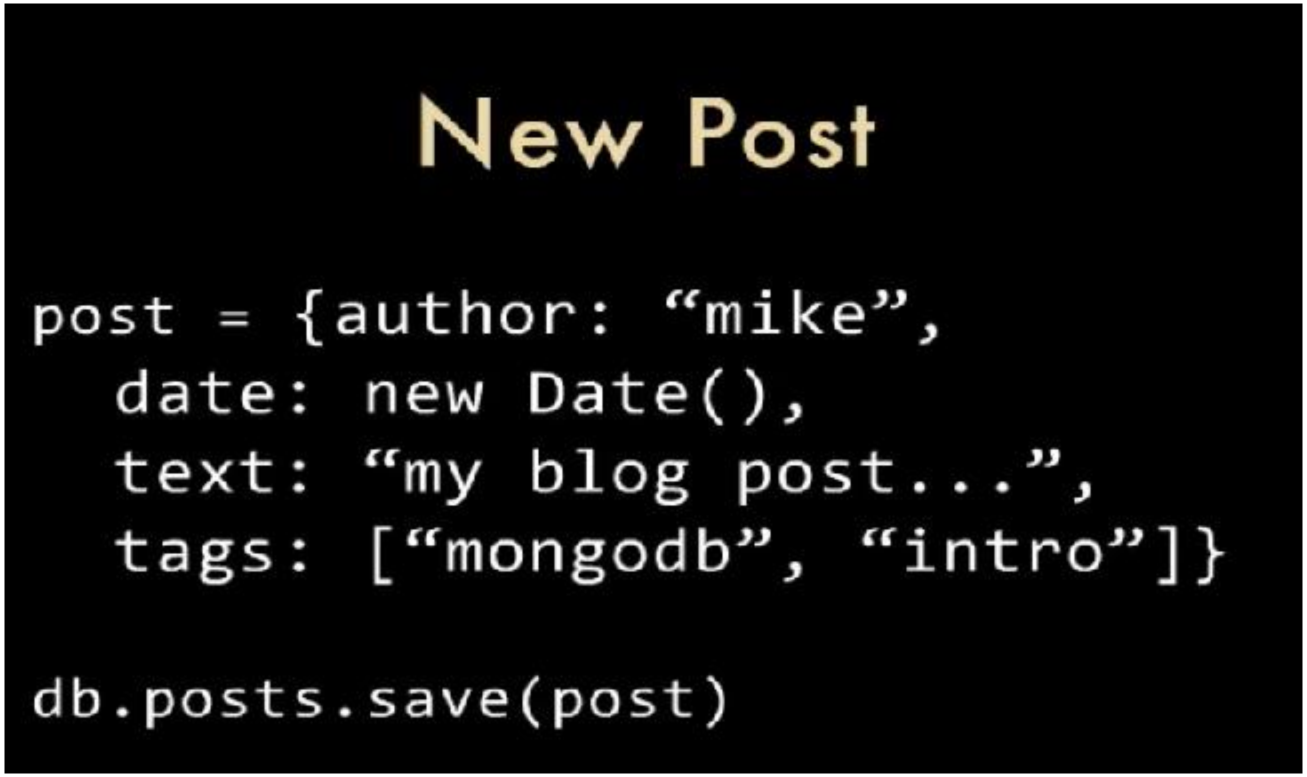
JavaScript Object Notation Json.org/Bsonspec.org



Flexible "Schemas"

```
{"author": "mike",
 "text": "..."}
```

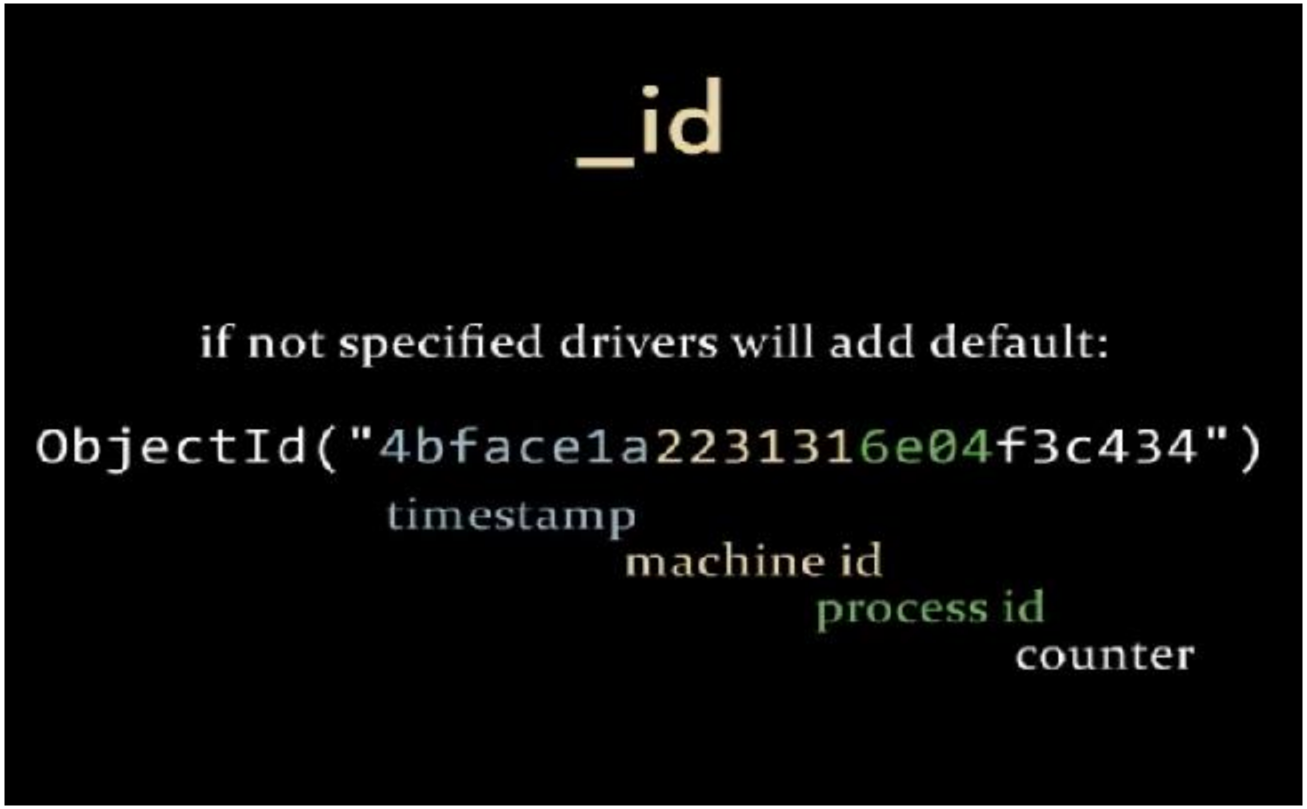
```
{"author": "eliot",
 "text": "...",
 "tags": ["mongodb"]}
```



New Post

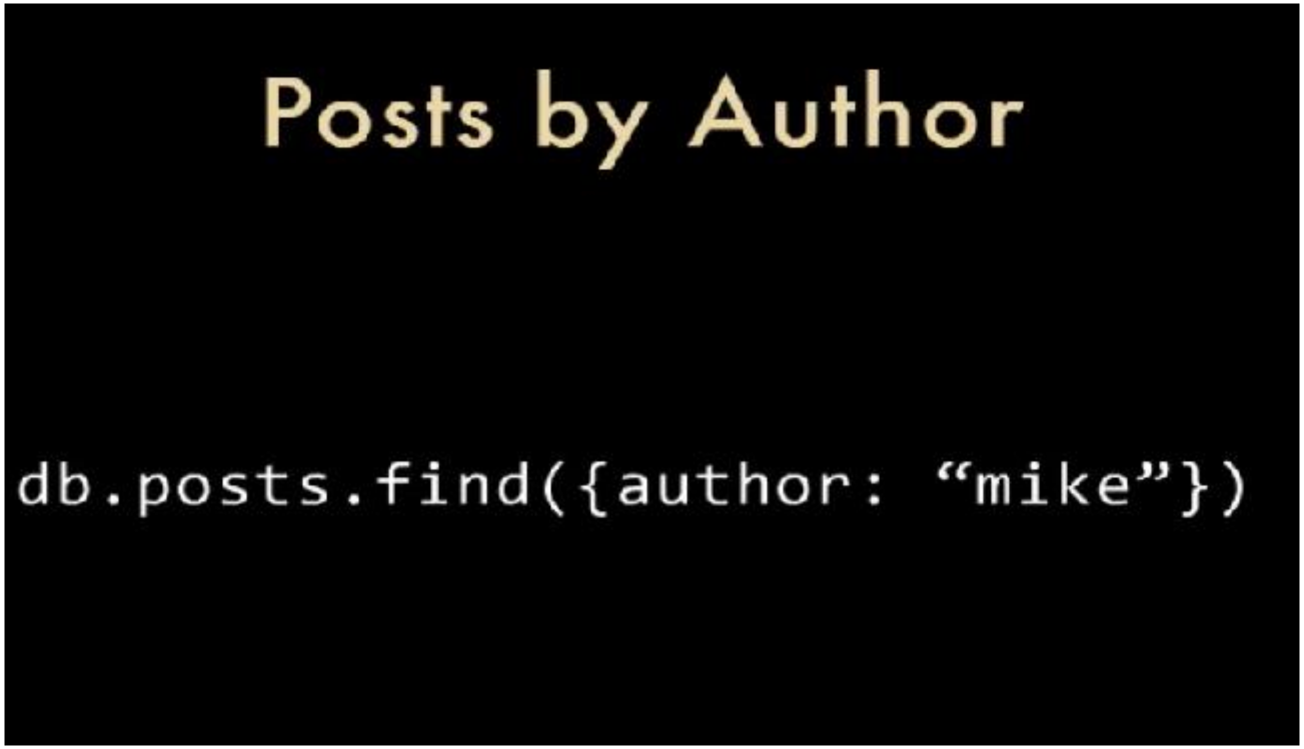
```
post = {author: “mike”,
  date: new Date(),
  text: “my blog post...”,
  tags: [“mongodb”, “intro”]}

db.posts.save(post)
```

Db.posts is a collection (if it does not exist it will create it)
It is going to save that post into that collection


_id
if not specified drivers will add default:
ObjectId("4bface1a2231316e04f3c434")
timestamp
machine id
process id
counter

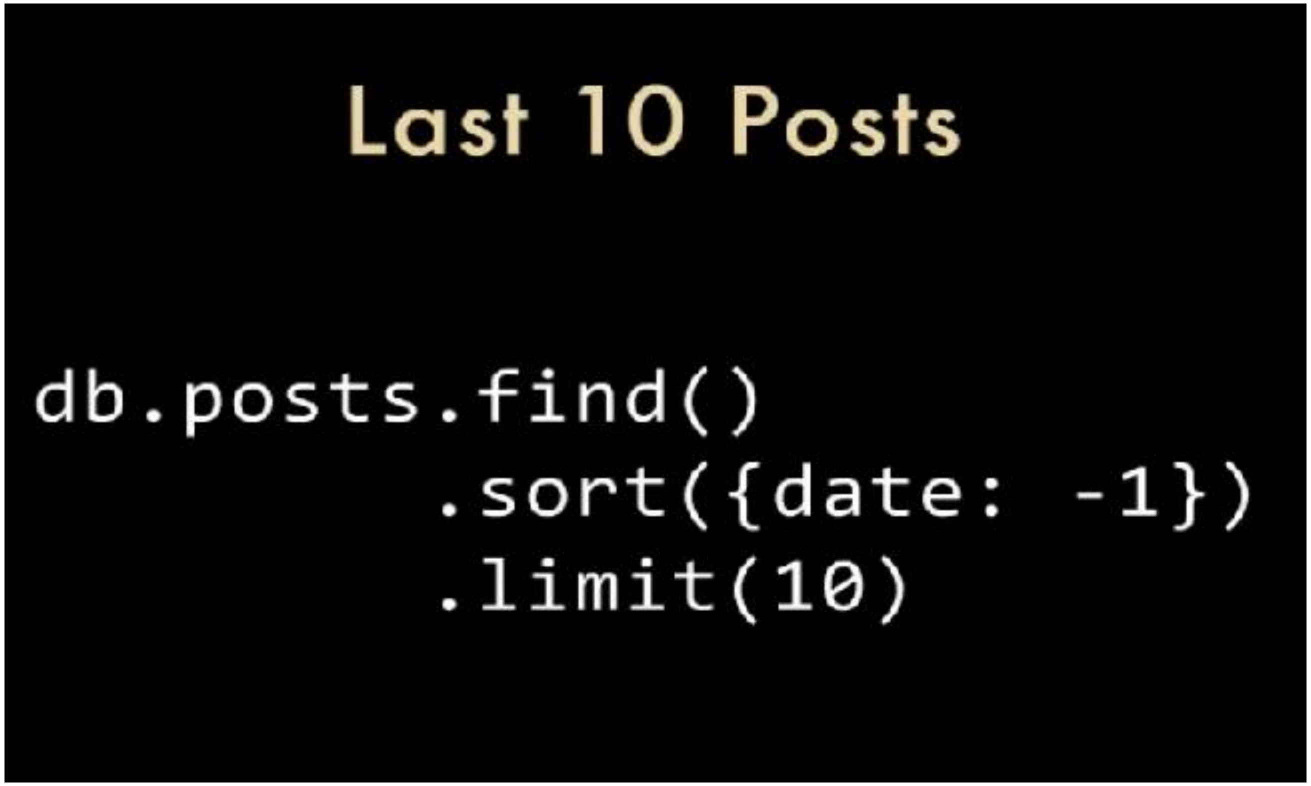
_id is lightweight occupying 12 bytes of storage
Generated on client side to reduce load on database server

Dynamic Queries



Posts by Author

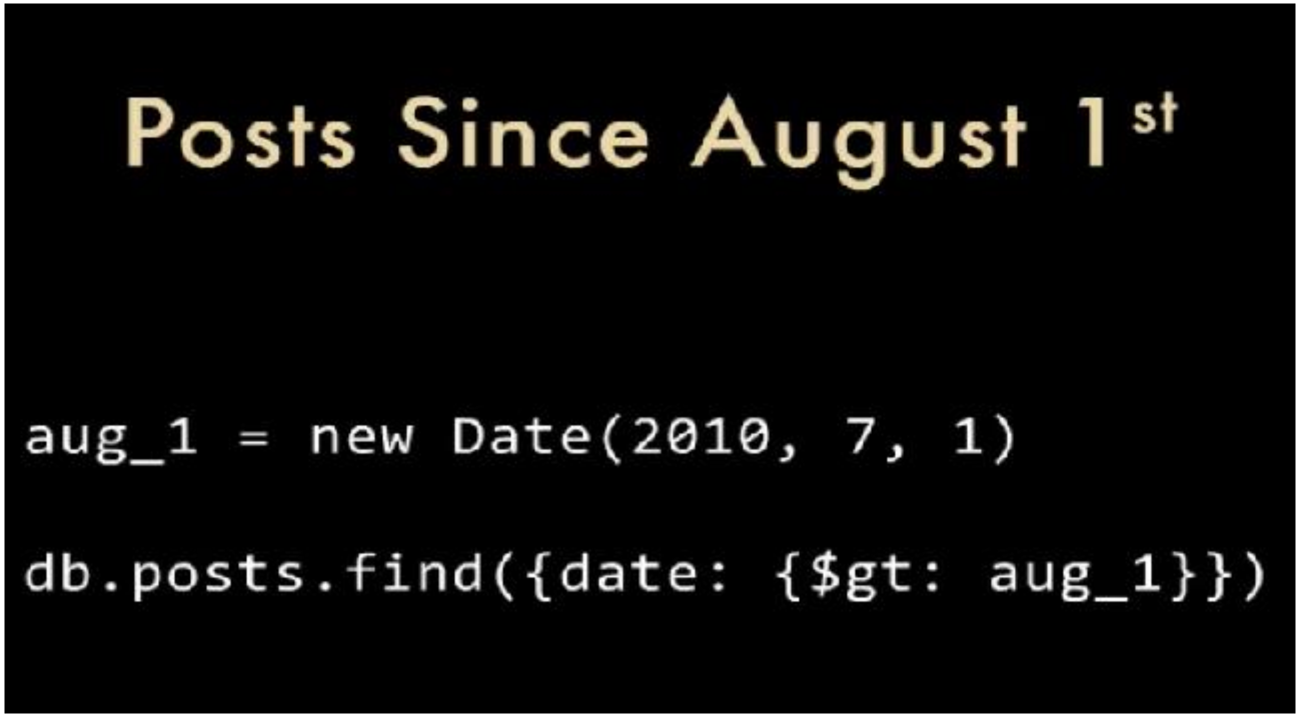
```
db.posts.find({author: “mike”})
```



Last 10 Posts

```
db.posts.find()
          .sort({date: -1})
          .limit(10)
```

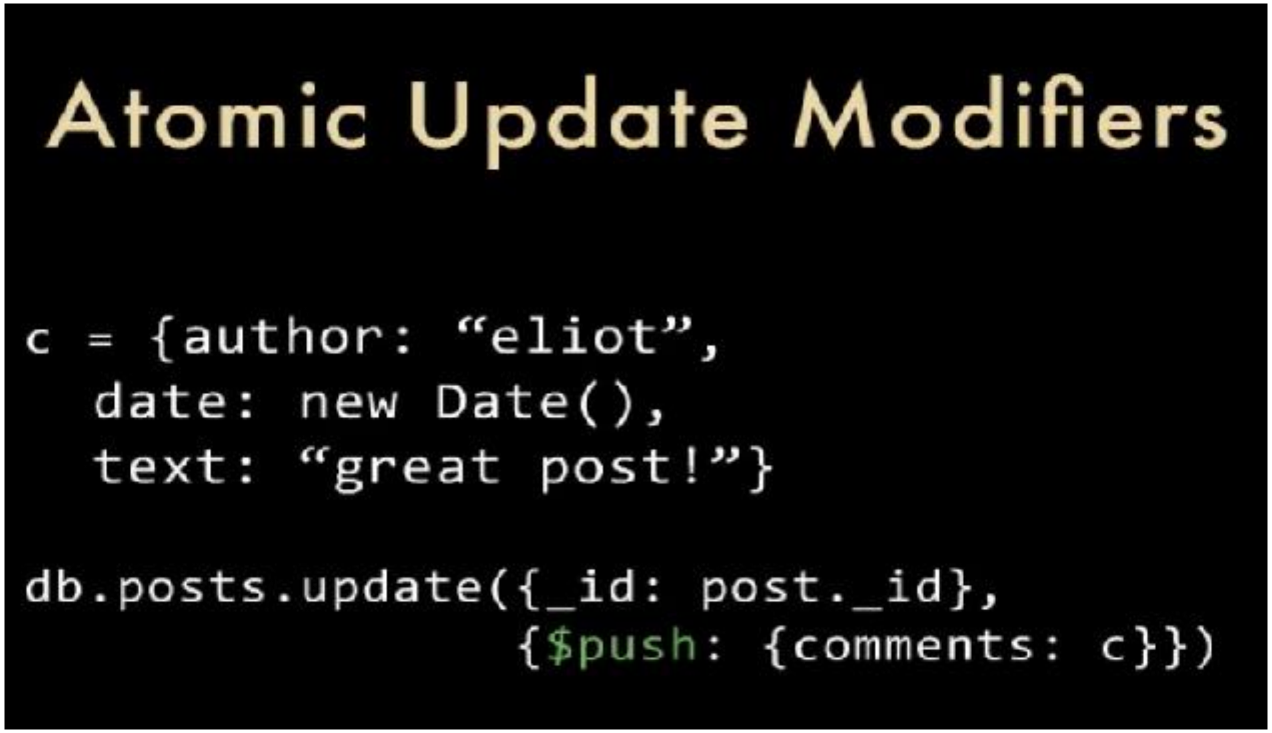
Find is a method



Posts Since August 1st

```
aug_1 = new Date(2010, 7, 1)

db.posts.find({date: {$gt: aug_1}})
```

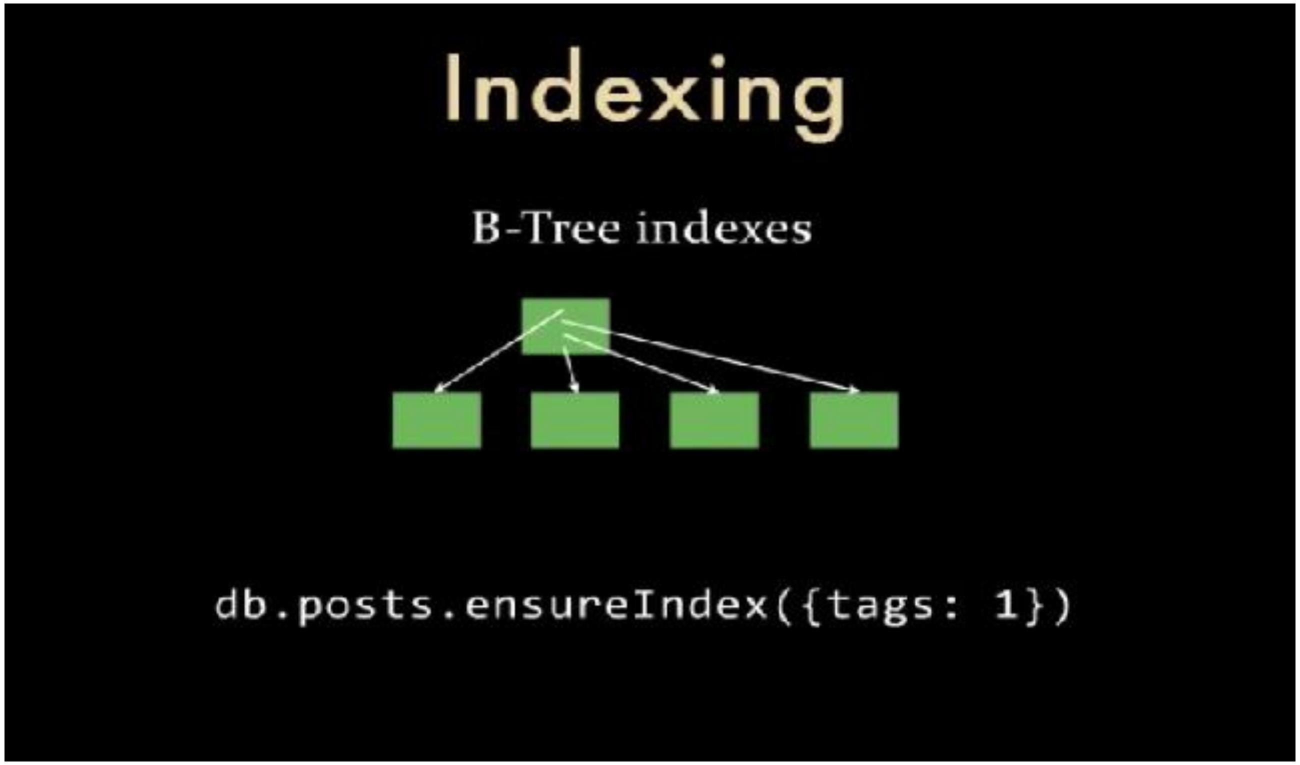
$gt operator stands for greater than
Javascript month starts with 0



Atomic Update Modifiers

```
c = {author: “eliot”,
  date: new Date(),
  text: “great post!”}

db.posts.update({_id: post._id},
                {$push: {comments: c}})
```

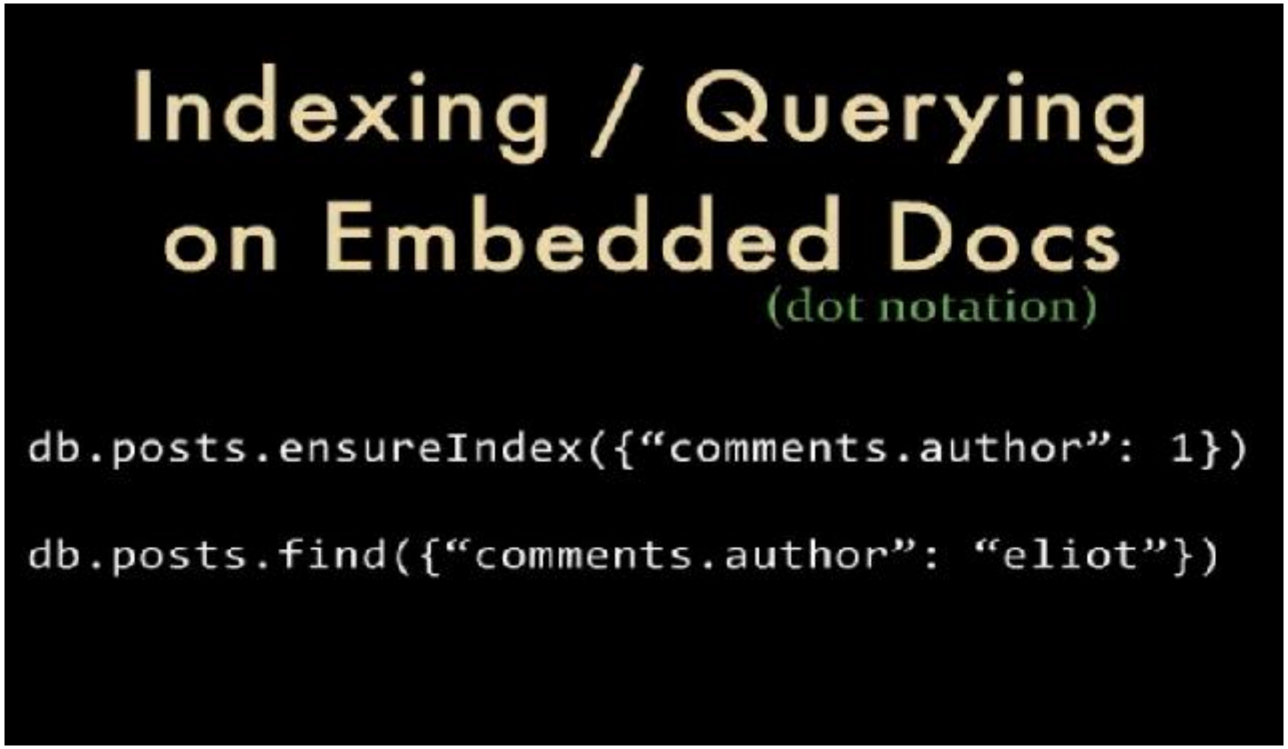
$ sign update modifier
New comments will keep on appending so no worry about locking



Indexing

B-Tree indexes

```
db.posts.ensureIndex({tags: 1})
```



Indexing / Querying on Embedded Docs

(dot notation)

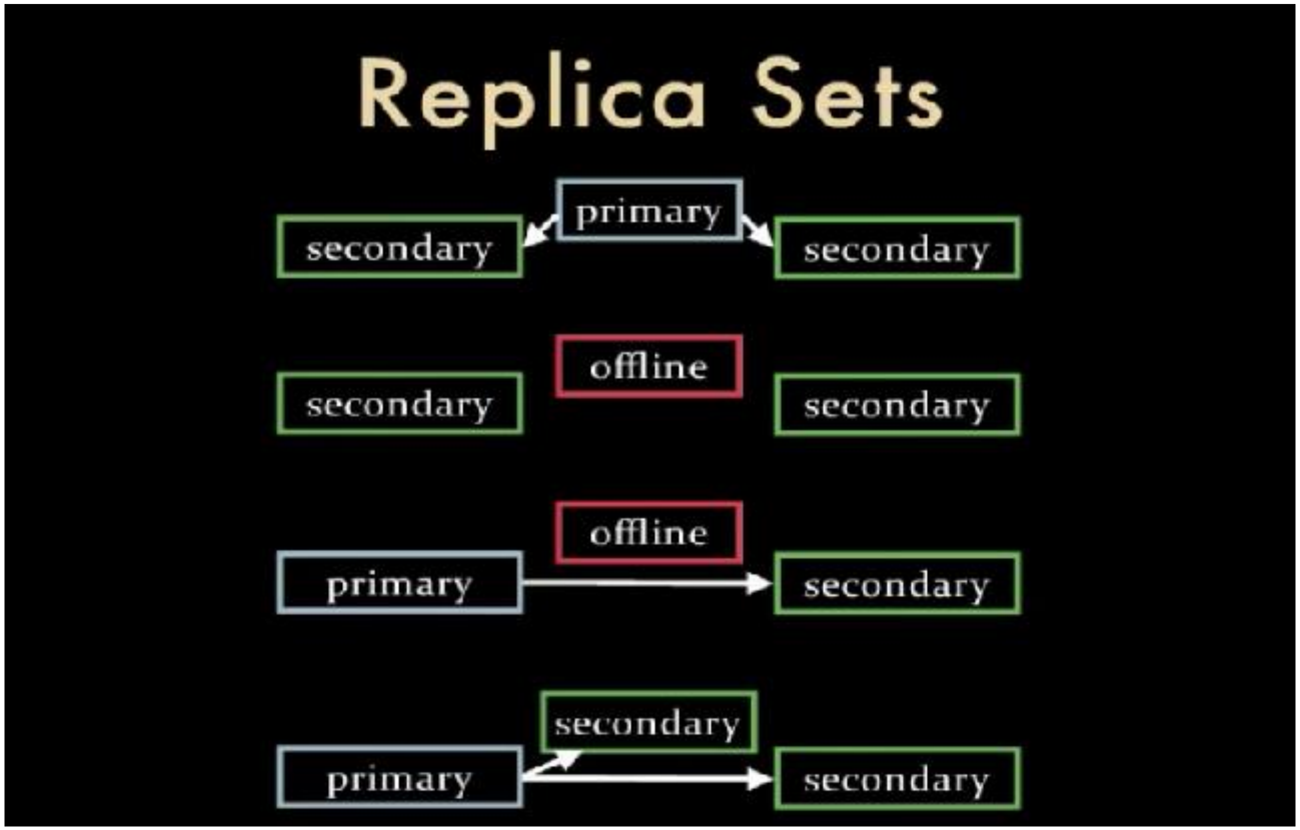
```
db.posts.ensureIndex({“comments.author”: 1})

db.posts.find({“comments.author”: “eliot”})
```

Many Supported Platforms / Languages

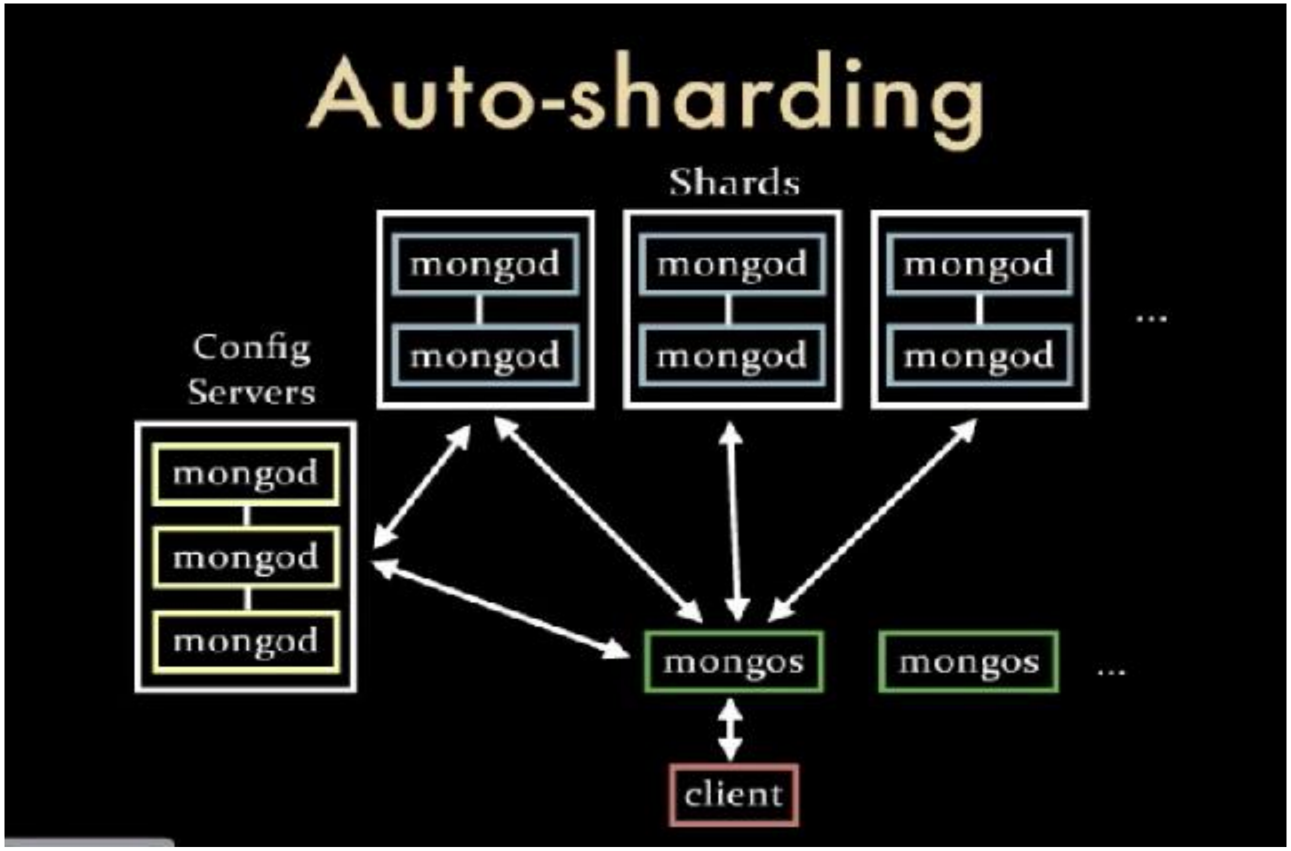
Drivers for PHP, Perl, C++, C#, .Net, Python, Ruby, etc.
All OS also 32/64 bit

Focus on Performance

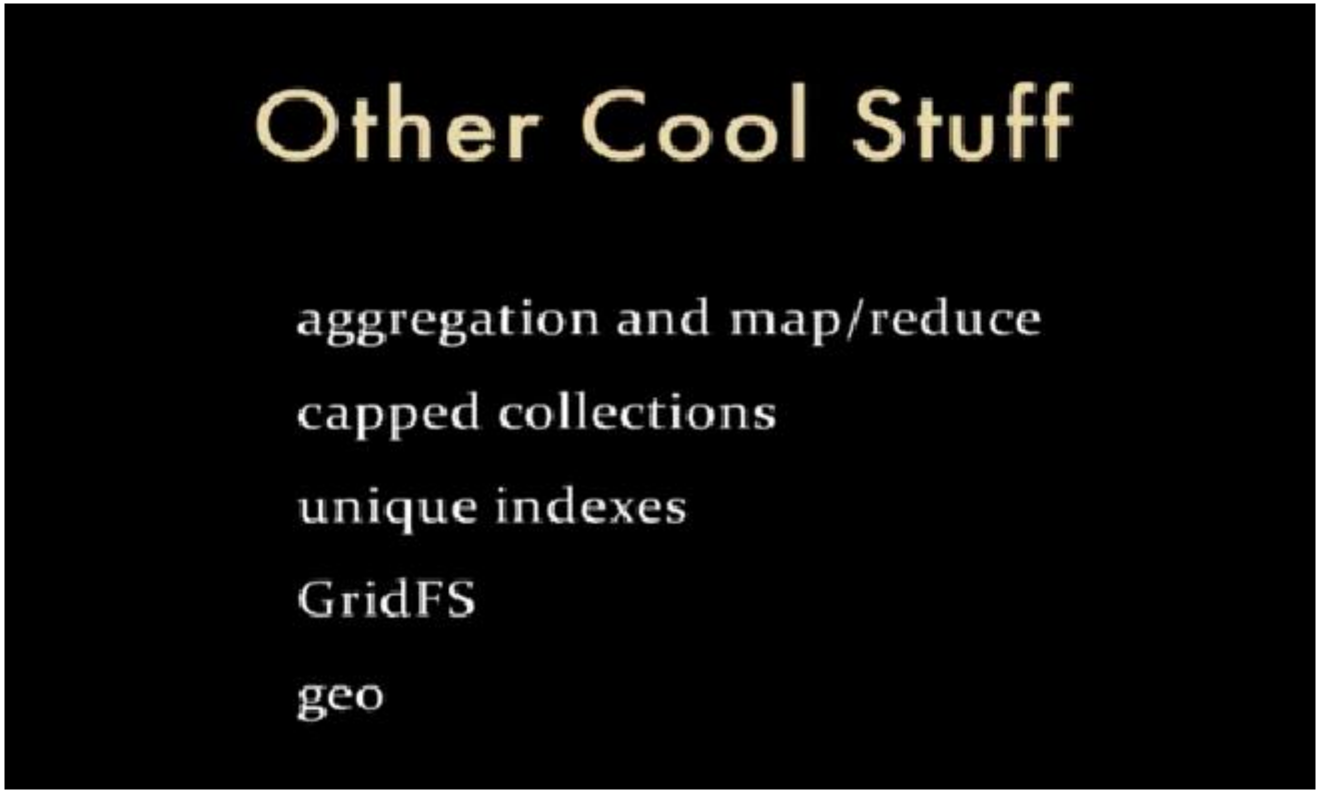
Does not use REST (http protocol)
Binary tcp wired protocol to communicate betwween client and server
Does not wait for Success Response to operations, proceeds to next one



Master/Slave
Primary is not fixed (Automatic failover)
If Primary goes offline then other nodes elect a new Primary



Split up the data, each shard has a subset of the data
Mongos is the process that distributes queries and writes to the shards
Scale out when needed not necessarily in advance



Other Cool Stuff

aggregation and map/reduce

capped collections

unique indexes

GridFS

geo

Download MongoDB

http://www.mongodb.org