Why NO-SQL ?

- Three interrelated megatrends

  - Big Data
  - Big Users
  - Cloud Computing

are driving the adoption of NoSQL technology.

Why NO-SQL ?

- *Google, Amazon, Facebook, and LinkedIn were among the first companies to discover the serious limitations of relational database technology for supporting these new application requirements.*

- *Commercial alternatives didn’t exist, so they invented new data management approaches themselves. Their pioneering work generated tremendous interest because a growing number of companies faced similar problems.*

- *Open source NoSQL database projects formed to leverage the work of the pioneers, and commercial companies associated with these projects soon followed.*

Why NO-SQL ?

- *1,000 daily users of an application was a lot and 10,000 was an extreme case.*
- *Today, most new applications are hosted in the cloud and available over the Internet, where they must support global users 24 hours a day, 365 days a year.*
- *More than 2 billion people are connected to the Internet worldwide – and the amount time they spend online each day is steadily growing – creating an explosion in the number of concurrent users.*
- *Today, it's not uncommon for apps to have millions of different users a day.*

Introduction

- New Generation Databases mostly addressing some of the points
  - being **non-relational**
  - **distributed**
  - **Opensource**
  - and **horizontal scalable.**
  - **Multi-dimensional rather than 2-D (relational)**

➢ **The** movement began early 2009 and is growing rapidly.

▪ **characteristics :**

- ▪ schema-free
- ▪ Decentralized Storage System
- ▪ easy replication support
- ▪simple API, etc.

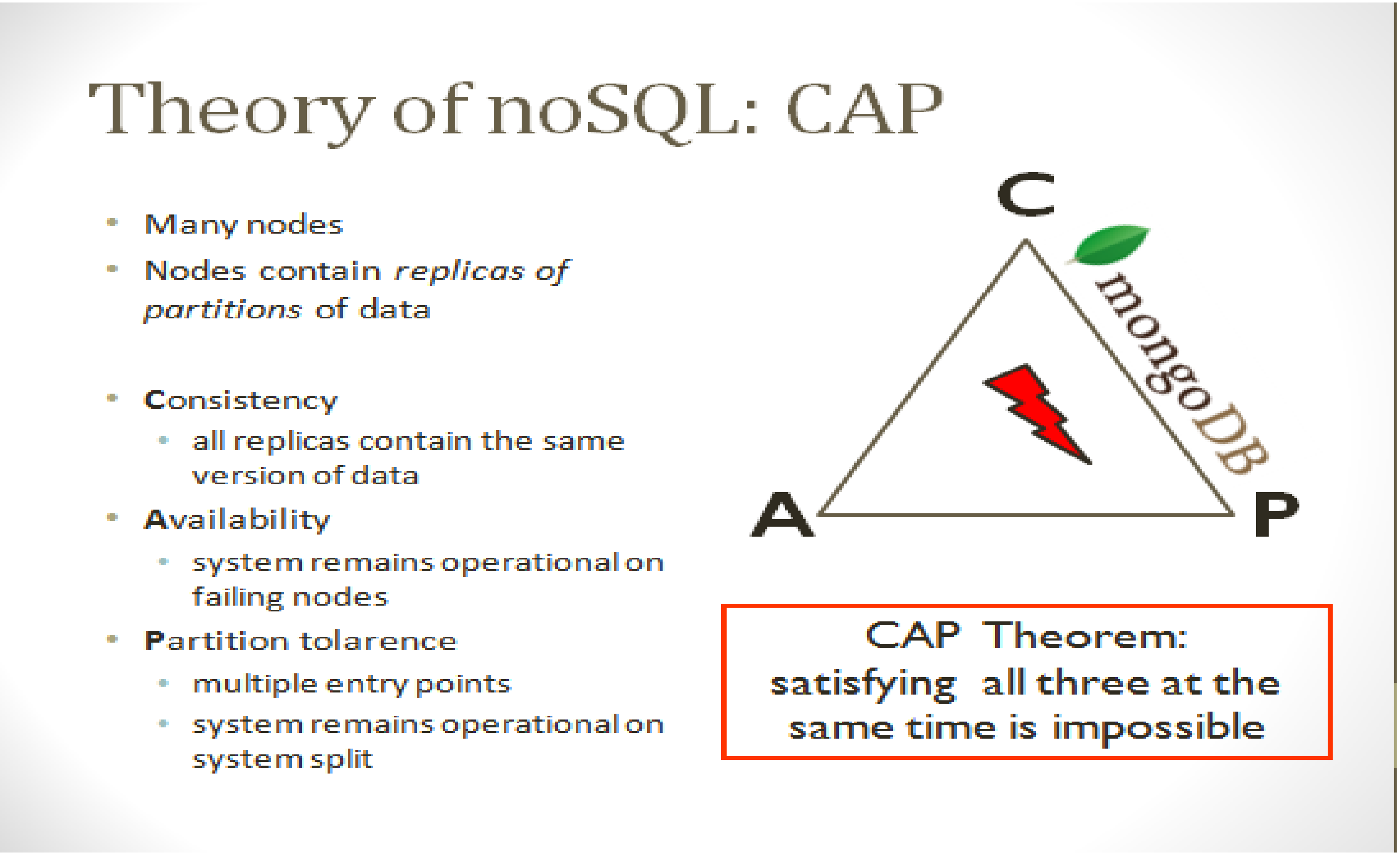
Introduction

- Application needs have been changing dramatically, due in large part to three trends: growing numbers of users that applications must support growth in the volume and variety of data that developers must work with; and the rise of cloud computing.
- NoSQL technology is rising rapidly among Internet companies and the enterprise because it offers data management capabilities that meet the needs of modern application:
- Greater ability to scale dynamically to support more users and data
- Improved performance to satisfy expectations of users wanting highly responsive applications and to allow more complex processing of data.
- NoSQL is increasingly considered aviable alternative to relational databases, and should be considered particularly for interactive web and mobile applications.

No-SQL (Examples)

- *Cassandra*
- *MongoDB*
- *Ealsticsearch*
- *Hbase*
- *CouchDB*
- *StupidDB*
- *Etc.*

No-SQL



Theory of noSQL: CAP

- Many nodes
- Nodes contain *replicas of partitions* of data

- Consistency
  - all replicas contain the same version of data
- Availability
  - system remains operational on failing nodes
- Partition tolarence
  - multiple entry points
  - system remains operational on system split

C
mongoDB
A
P

CAP Theorem:
satisfying all three at the same time is impossible

Cassandra

- *Cassandra is a distributed storage system for managing very large amounts of data spread out across many commodity servers.*

- *while providing highly available service with no single point of failure.*

- *Cassandra aims to run on top of an infrastructure of hundreds of nodes At this scale, small and large components fail continuously.*

- *The way Cassandra manages the persistent state in the face of these failures drives the reliability and scalability of the software systems relying on this service.*

- *Cassandra system was designed to run on cheap commodity hardware and handle high write throughput while not sacrificing read efficiency.*

Cassandra

- *Facebook runs the largest social networking platform that serves hundreds of millions users at peak times using tens of thousands of servers located in many data centers around the world.*

- *There are strict operational requirements on Facebook's platform in terms of performance, reliability and efficiency, and to support continuous growth the platform needs to be highly scalable.*

- *Dealing with failures in an infrastructure comprised of thousands of components is standard mode of operation*

- *There are always a small but significant number of server and network components that are failing at any given time. As such, the software systems need to be constructed in a manner that treats failures as the norm rather than the exception.*

Cassandra

- *Cassandra is now deployed as the backend storage system for multiple services within Facebook.*

- *To meet the reliability and scalability needs described above Facebook has developed Cassandra.*

- *Cassandra was designed to full the storage needs of the Search problem.*

Data Model (Cassandra)

- *Cassandra is a distributed key-value store.*
- *A table in Cassandra is a distributed multi dimensional map indexed by a key. The value is an object which is highly structured.*

- *The row key in a table is a string with no size restrictions, although typically 16 to 36 bytes long.*

- *Every operation under a single row key is atomic per replica no matter how many columns are being read or written into.*

- *Columns are grouped together into sets called column families.*
- *Cassandra exposes two kinds of columns families, Simple and Super column families.*

- *Super column families can be visualized as a column family within a column family.*

Architecture (Cassandra)

- *Cassandra is designed to handle big data workloads across multiple nodes with no single point of failure.*

- *Its architecture is based in the understanding that system and hardware failure can and do occur.*

- *Cassandra addresses the problem of failures by employing a peer-to-peer distributed system where all nodes are the same and data is distributed among all nodes in the cluster.*

- *Each node exchanges information across the cluster every second.*
- *A commit log on each node captures write activity to ensure data durability.*

Architecture (Cassandra)

- *Data is also written to an in-memory structure, called a **memtable**, and then written to a data file called an **SStable** on disk once the memory structure is full.*

- *All writes are automatically partitioned and replicated throughout the cluster.*

- *Client read or write requests can go to any node in the cluster.*

- *When a client connects to a node with a request, that node serves as the coordinator for that particular client operation.*

- *The coordinator acts as a proxy between the client application and the nodes that own the data being requested.*

- *The coordinator determines which nodes in the ring should get the request based on how the cluster is configured.*

Cassandra (Properties)

- **Written in:** Java
- **Main point:** Best of BigTable and Dynamo
- **License:** Apache
- Querying by column
- BigTable-like features: columns, column families
- Writes are much faster than reads
- Map/reduce possible with Apache Hadoop.

MangoDB

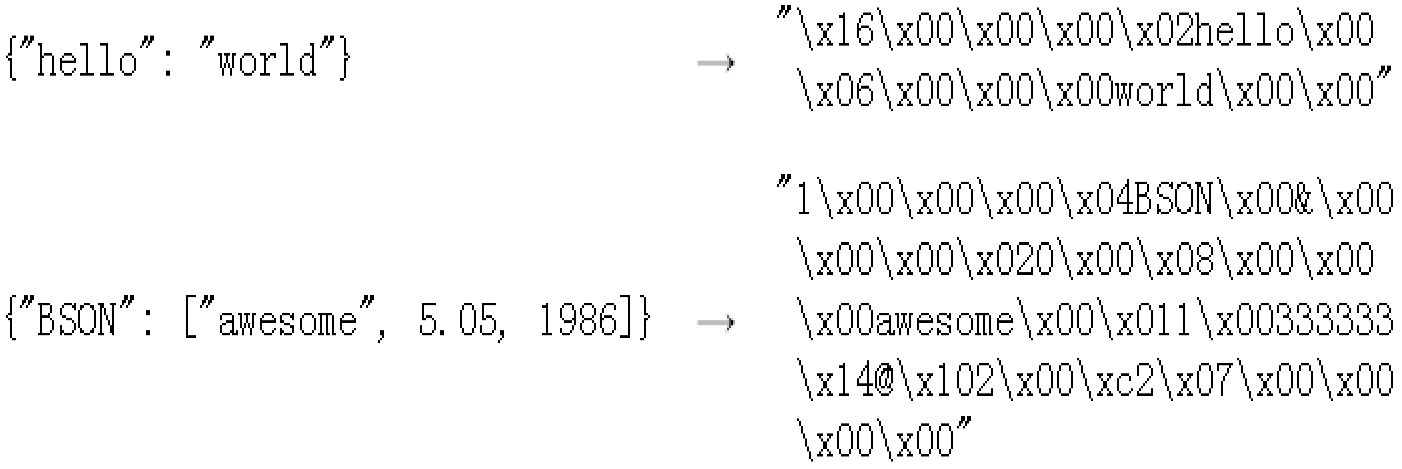
- ***What is?***
- *MongoDB is an open source, document-oriented database designed with both scalability and developer agility in mind.*
- *Instead of storing data in tables and rows like as relational database, in MongoDB store JSON-like documents with dynamic schemas(schema-free, schemaless).*

MangoDB (Introduction)

- ***Written in:*** *C++*
- ***Main point:*** *Retains some friendly properties of SQL*
- ***License:*** *AGPL (Drivers: Apache)*
- *Master/slave replication (auto failover with replica sets)*
- *Sharding built-in*
- *Queries are javascript expressions*
- *Run arbitrary javascript functions server-side*
- *Better update-in-place than CouchDB*
- *Uses memory mapped files for data storage*
- *An empty database takes up 192Mb*
- *GridFS to store big data + metadata (not actually an FS)*

Data Model

- **Data model:** Using BSON (binary JSON), developers can easily map to modern object-oriented languages without a complicated ORM layer.
- BSON is a binary format in which zero or more key/value pairs are stored as a single entity.
- lightweight, traversable, efficient.



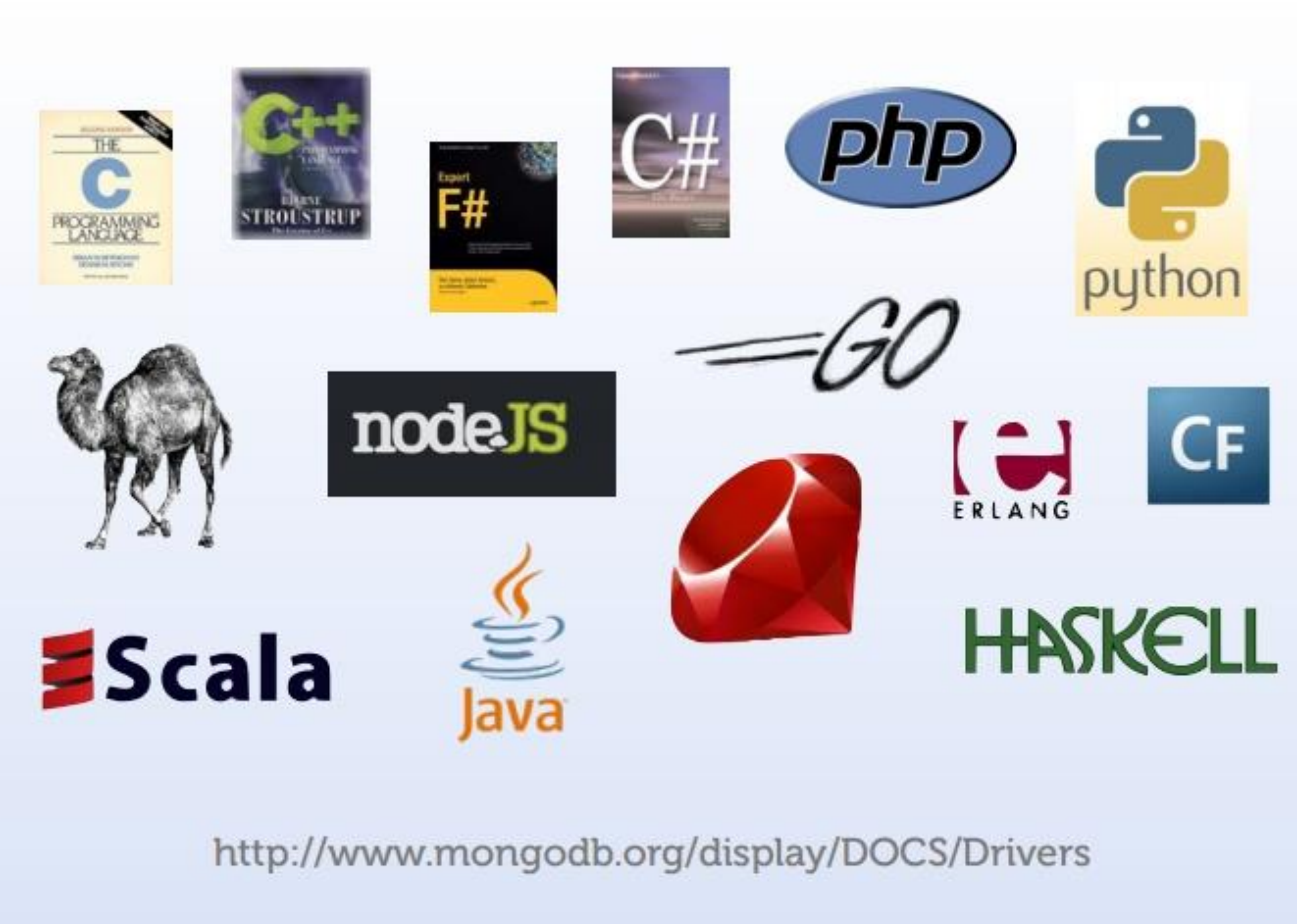
```
                                         "\x16\x00\x00\x00\x02hello\x00
{"hello": "world"}                    →   \x06\x00\x00\x00world\x00\x00"

                                         "1\x00\x00\x00\x04BSON\x00&\x00
                                          \x00\x00\x020\x00\x08\x00\x00
{"BSON": ["awesome", 5.05, 1986]}     →   \x00awesome\x00\x011\x00333333
                                          \x14@\x102\x00\xc2\x07\x00\x00
                                          \x00\x00"
```

Schema Design

```
• {
•         "_id" : ObjectId("5114e0bd42…"),
•         "first" : "John",
•         "last" : "Doe",
•         "age" : 39,
•        "interests" : [
•                "Reading",
•                "Mountain Biking ]
•        "favorites": {
•               "color": "Blue",
•               "sport": "Soccer"}
• }
```

Supported Languages

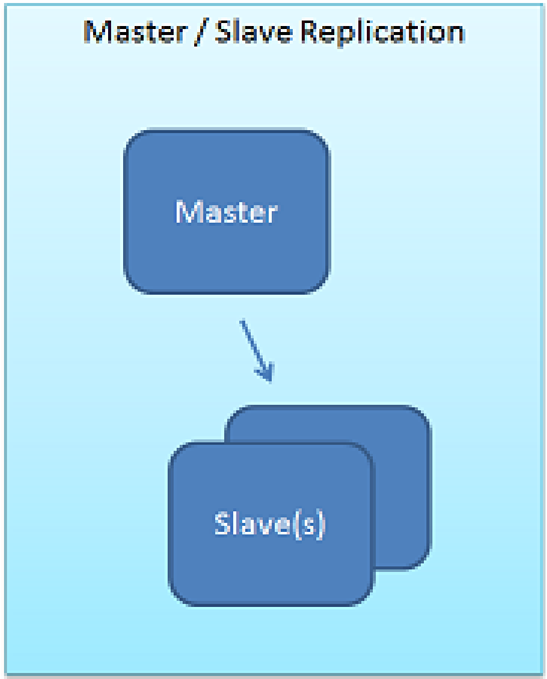


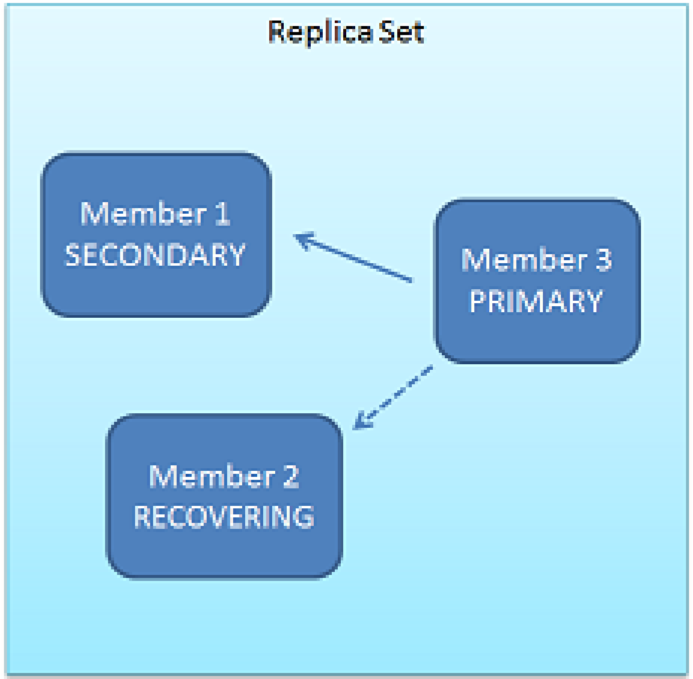
http://www.mongodb.org/display/DOCS/Drivers

Architecture (MangoDB)

1. Replication

- Replica Sets and Master-Slave
- replica sets are a functional superset of master/slave.





Architecture (Write process)

- *All write operation go through primary, which applies the write operation.*
- *write operation than records the operations on primary's operation log "oplog"*
- *Secondary are continuously replicating the oplog and applying the operations to themselves in a asyn0chronous process.*

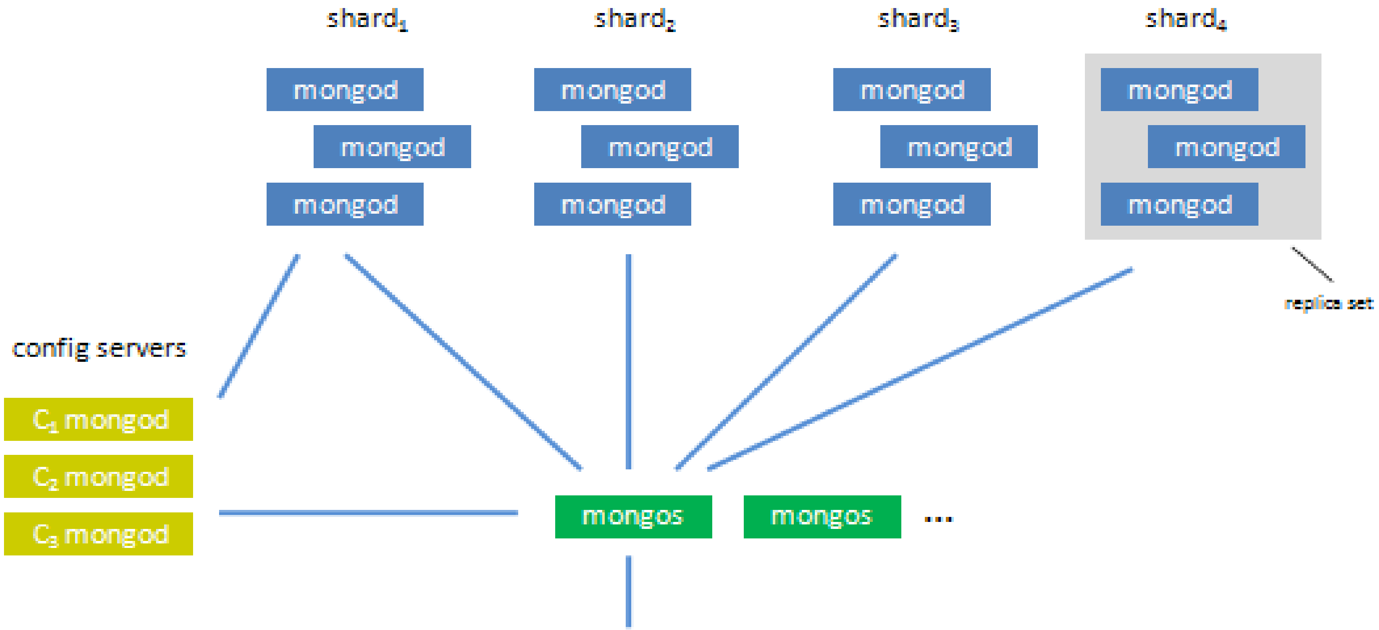
Why replica sets

- Data Redundancy
- Automated Failover
- Read Scaling
- Maintenance
- Disaster Recovery(delayed secondary)

Architecture

2. Sharding

- Sharding is the partitioning of data among multiple machines in an order-preserving manner.(horizontal scaling)



Features

- Document-Oriented storege
- Full Index Support
- Replication & High Availability
- Auto-Sharding
- Querying
- Fast In-Place Updates
- Map/Reduce

HBase

- *HBase was created in 2007 at Powerset and was initially part of the contributions in Hadoop.*
- *Since then, it has become its own top-level project under the Apache Software Foundation umbrella.*
- *It is available under the Apache Software License, version 2.0.*

Hbase

Features

- – **non-relational**
- – **distributed**
- – **Opensource**
- – and **horizontal scalable.**
- – **Multi-dimensional rather than 2-D (relational)**

- ▪ schema-free
- ▪ Decentralized Storage System
- ▪ easy replication support
- ▪simple API, etc.

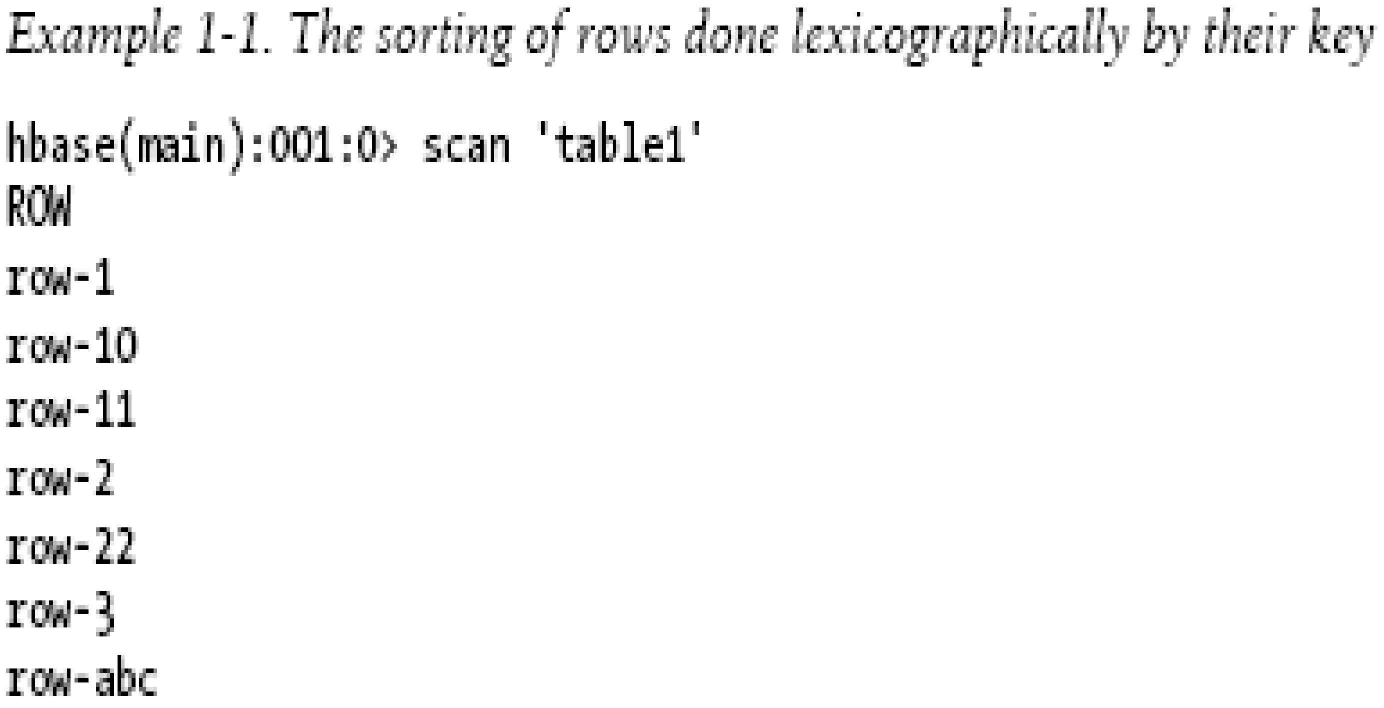
Basic Architecture

Tables, Rows, Columns, and Cells

- *the most basic unit is a column.*
- *One or more columns form a row that is addressed uniquely by a row key.*
- *A number of rows, in turn, form a table, and there can be many of them.*
- *Each column may have distinct value contained in a separate cell.*

Architecture

- All *rows are always sorted lexicographically by their row key.*



Example 1-1. The sorting of rows done lexicographically by their key

```
hbase(main):001:0> scan 'table1'
ROW
row-1
row-10
row-11
row-2
row-22
row-3
row-abc
```

Architecture

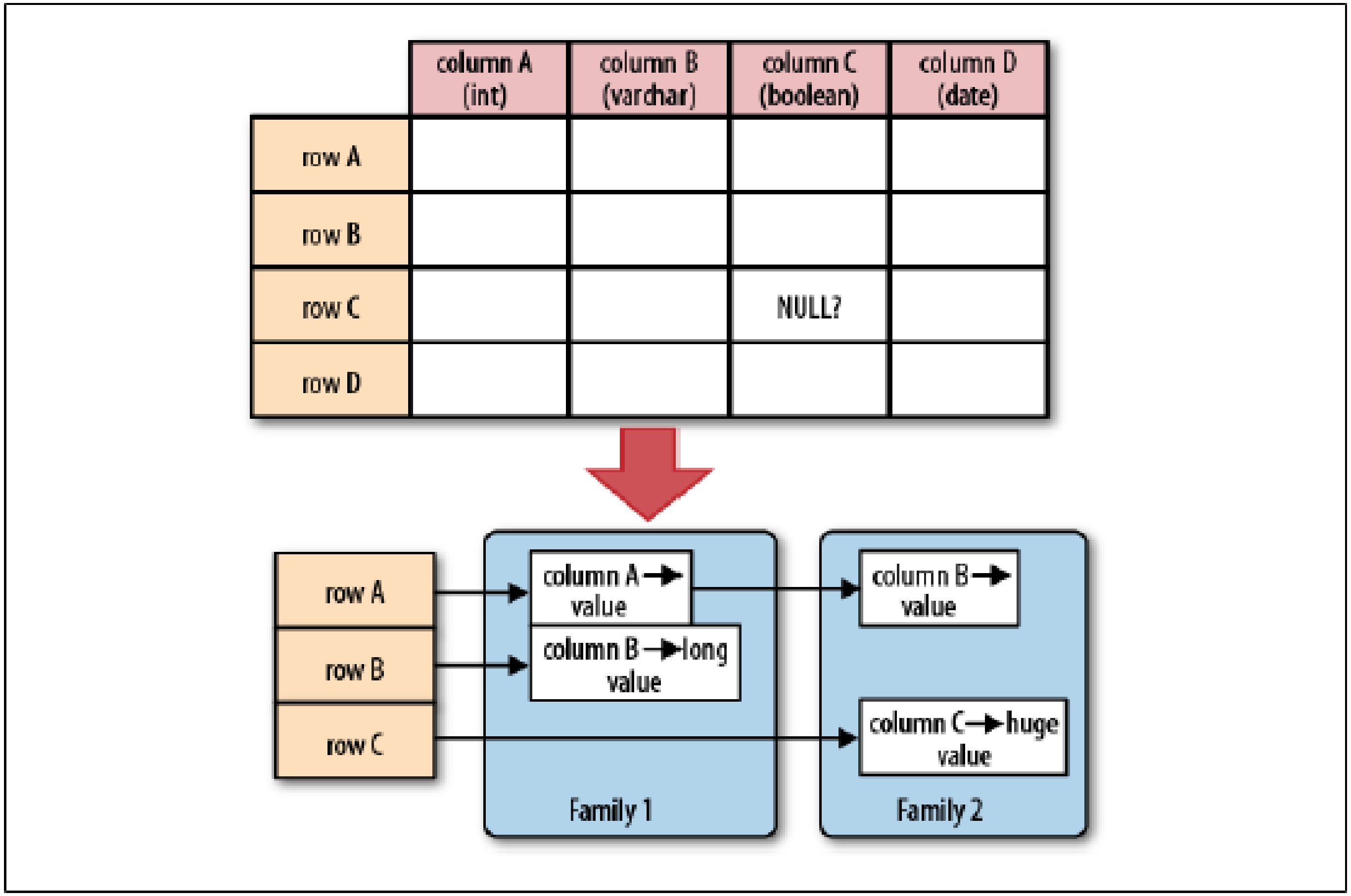
- *Rows are composed of columns, and those, in turn, are grouped into column families.*

- *All columns in a column family are stored together in the same low level storage file, called an **HFile**.*

- *millions of columns in a particular column family.*

- *There is also no type nor length boundary on the column values.*

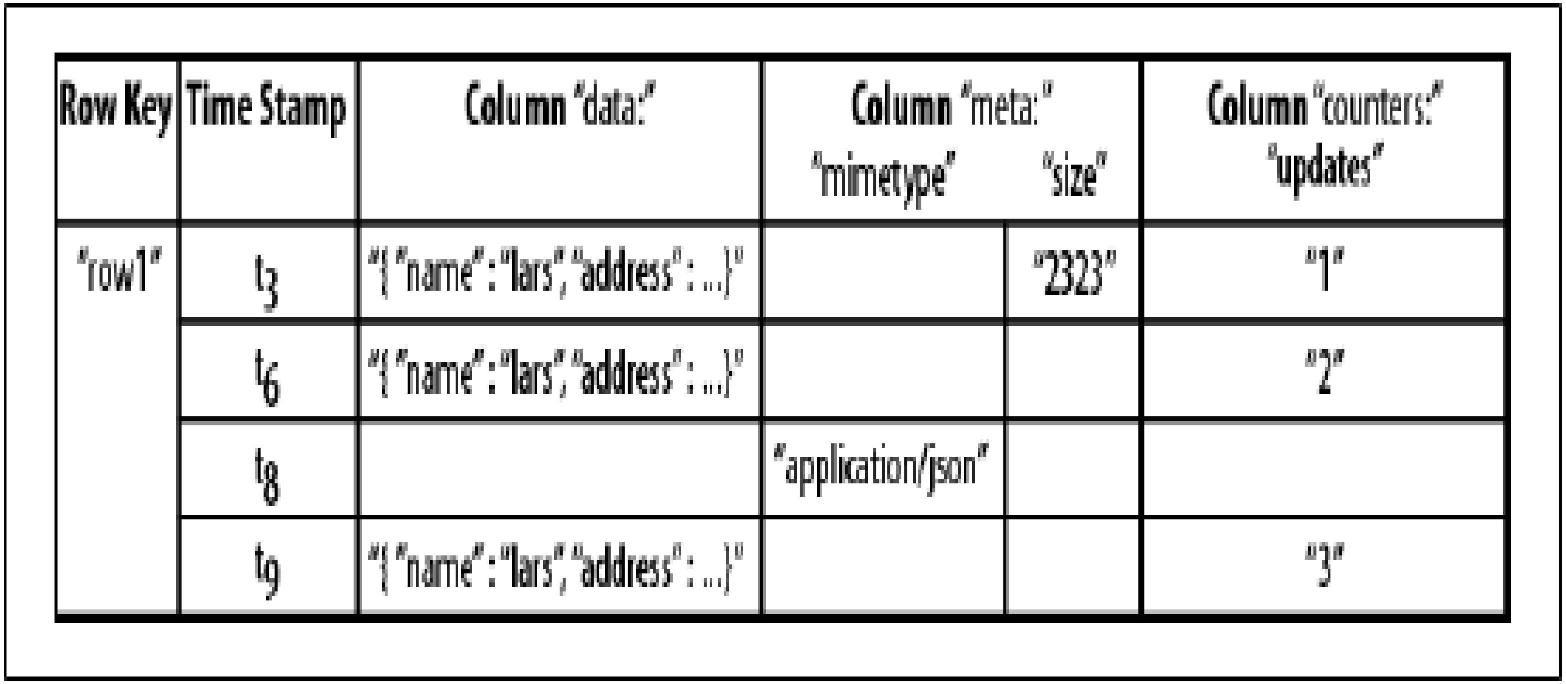
Rows and Columns in HBase



	column A (int)	column B (varchar)	column C (boolean)	column D (date)
row A				
row B				
row C			NULL?	
row D				

Figure 1-4. Rows and columns in HBase

Rows and Columns in HBase



Row Key	Time Stamp	Column "data:"	Column "meta:"		Column "counters:"
			"mimetype"	"size"	"updates"
"row1"	t_3	"{ "name" : "lars", "address" : ...}"		"2323"	"1"
	t_6	"{ "name" : "lars", "address" : ...}"			"2"
	t_8		"application/json"		
	t_9	"{ "name" : "lars", "address" : ...}"			"3"

Figure 1-6. The same parts of the row rendered as a spreadsheet

Assignment

- *1. Storage structure of Google's BigTable.*
- *2. How NO-SQL deals with Structured and unstructured data*