SQL vs NOSQL Discussion

SQL vs NOSQL

OUTLINE

- What is NOSQL?
- How does NOSQL compare to SQL?
- Data structures and performance considerations
- Why should OSEHRA care about SQL vs NOSQL?

SQL Characteristics

Data stored in columns and tables

Relationships represented by data

Data Manipulation Language

Data Definition Language

Transactions

Abstraction from physical layer

SQL Physical Layer Abstraction

Applications specify what, not how

Query optimization engine

Physical layer can change without modifying applications

Create indexes to support queries

In Memory databases

Data Manipulation Language (DML)

Data manipulated with Select, Insert, Update, & Delete statements

Select T1.Column1, T2.Column2 …
From Table1, Table2 …
Where T1.Column1 = T2.Column1 …

Data Aggregation

Compound statements

Functions and Procedures

Explicit transaction control

Data Definition Language

Schema defined at the start

Create Table (Column1 Datatype1, Column2 Datatype 2, …)

Constraints to define and enforce relationships
Primary Key
Foreign Key
Etc.

Triggers to respond to Insert, Update , & Delete

Stored Modules

Alter …

Drop …

Security and Access Control

Transactions – ACID Properties

Atomic – All of the work in a transaction completes (commit) or none of it completes

Consistent – A transaction transforms the database from one consistent state to another consistent state. Consistency is defined in terms of constraints.

Isolated – The results of any changes made during a transaction are not visible until the transaction has committed.

Durable – The results of a committed transaction survive failures

NewSQL: more OLTP throughput, real-time analytics

1) SQL as the primary mechanism for application interaction

2) ACID support for transactions

3) A non-locking concurrency control mechanism so real-time reads will not conflict with writes, and thereby cause them to stall.

4) An architecture providing much higher per-node performance than available from the traditional "elephants”

5) A scale-out, shared-nothing architecture, capable of running on a large number of nodes without bottlenecking

Challenges

- **Big data** is a term for data sets that are so large or complex that traditional data processing applications are inadequate.
  - Challenges include analysis, capture, data curation, search, sharing, storage, transfer, visualization, querying and information privacy.
  - The term often refers simply to the use of predictive analytics or certain other advanced methods to extract value from data, and seldom to a particular size of data set.
  - Causes: Cloud computing, social media, Internet of Things (IoT), etc

What is NoSQL?

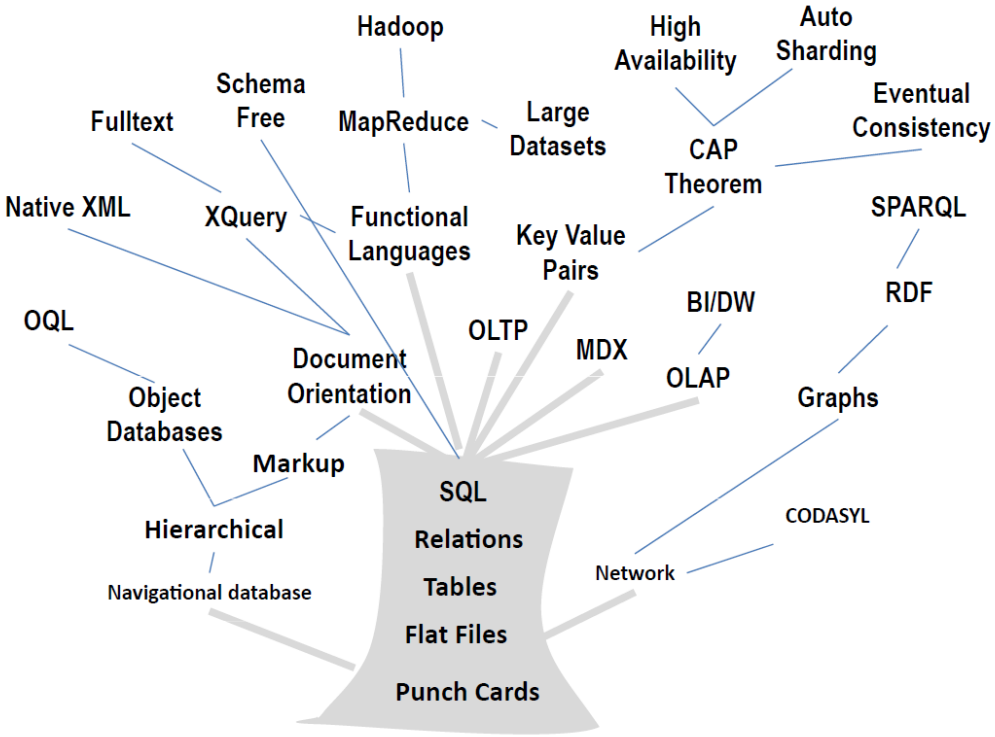
- Stands for No-SQL or **N**ot **O**nly **SQL??**
- Class of non-relational data storage systems
  - E.g. BigTable, Dynamo, PNUTS/Sherpa, ..
- Usually do not require a fixed table schema nor do they use the concept of joins
  - Distributed data storage systems
- All NoSQL offerings relax one or more of the ACID properties (will talk about the CAP theorem)

What is NOSQL?

- NoSQL is a class of database management system identified by its non-adherence to the widely used relational database management system (RDBMS) model with its structured query language (SQL).
- NOSQL has evolved to mean “Not Only” SQL
- NOSQL has become prominent with the advent of web scale data and systems created by Google, Facebook, Amazon, Twitter and others to manage data for which SQL was not the best fit.

What is NOSQL?

- Definitions for NOSQL vary greatly from newer systems using document stores, key value stores, XML databases, graph databases, column stores, object stores, etc. (like MongoDB, Cassandra, Couchbase, Hadoop, etc.) to older Hierarchical systems that had many similar characteristics (like Cache and GT.M)
- The NOSQL concept tree illustrates the variety of concepts related to NOSQL.



NOSQL Concept Tree

Source: CIO's Guide to NOSQL, Dan McCreary, June 2012

What is NOSQL?

NoSQL Definition

From www.nosql-database.org:

> **Next Generation Databases mostly addressing some of the points: being non-relational, distributed, open-source and horizontal scalable. The original intention has been modern web-scale databases. The movement began early 2009 and is growing rapidly. Often more characteristics apply as: schema-free, easy replication support, simple API, eventually consistent / BASE (not ACID), a huge data amount, and more.**

What is NOSQL?
Large diversity of NOSQL databases

- Document Store
  - BaseX, Clusterpoint, Apache Couchbase, eXist, Jackrabbit, Lotus Notes and IBM Lotus Domino LotusScript, MarkLogic Server, MongoDB, OpenLink Virtuoso, OrientDB, RavenDB, SimpleDB, Terrastore
- Column Based
  - **Cassandra**
- Graph
  - AllegroGraph, DEX, FlockDB, InfiniteGraph, Neo4j, OpenLink Virtuoso, OrientDB, Pregel, Sones GraphDB, OWLIM
- Key Value
  - BigTable, CDB, Keyspace, LevelDB, membase, MemcacheDB, MongoDB, OpenLink Virtuoso, Tarantool, Tokyo Cabinet, TreapDB, Tuple space
  - Eventually-consistent - Apache Cassandra, Dynamo, Hibari, OpenLink Virtuoso, Project Voldemort, Riak
  - Hierarchical - GT.M, InterSystems Caché
  - Tabular – BigTable, Apache Hadoop, Apache Hbase, Hypertable, Mnesia, OpenLink Virtuoso
  - Object Database - db4o, Eloquera, GemStone/S, InterSystems Caché, JADE, NeoDatis ODB, ObjectDB, Objectivity/DB, ObjectStore, OpenLink Virtuoso, Versant Object Database, Wakanda, ZODB
  - Multivalue databases - Extensible Storage Engine (ESE/NT), jBASE, OpenQM, OpenInsight , Rocket U2, D3 Pick database, InterSystems Caché, InfinityDB
  - Tuple store- Apache River, OpenLink Virtuoso, Tarantool

How Does NOSQL compare to SQL?

While there are numerous characteristics that differentiate SQL and NOSQL the two most significant are Scaling and Modeling.

- **Scaling** – Traditionally SQL does not lend itself to massively parallel processing, which lead to larger computers (scale up) vs. distribution to numerous commodity servers, virtual machines or cloud instances (scale out).
- **Modeling** – SQL databases are highly normalized and require pre-defined data models prior to inserting data into the system. In contrast NOSQL databases do not require (although they support) pre-defined data model(s).

Data structures and performance considerations

- Structured vs. Unstructured
  - Tables, fields, pairs vs. unstructured text
- Transactions vs. Analytics
- Federated vs. Persisted
- Big Data: Volume, Variety and Velocity
- Retrieval
  - Indexing, MapReduce, Search, Query
- Precision vs. Discovery

Why should an Organization take care about SQL vs NOSQL?

- Cloud based architecture
- Standardized or flexible data models
- Different or same application and analytic data stores?

NoSQL Distinguishing Characteristics

Large data volumes

Google's "big data"

Scalable replication and distribution

Potentially thousands of machines

Potentially distributed around the world

Queries need to return answers quickly

Mostly query, few updates

Asynchronous Inserts & Updates

Schema-less

ACID transaction properties are not needed – BASE

CAP Theorem

Open source development

BASE Transactions

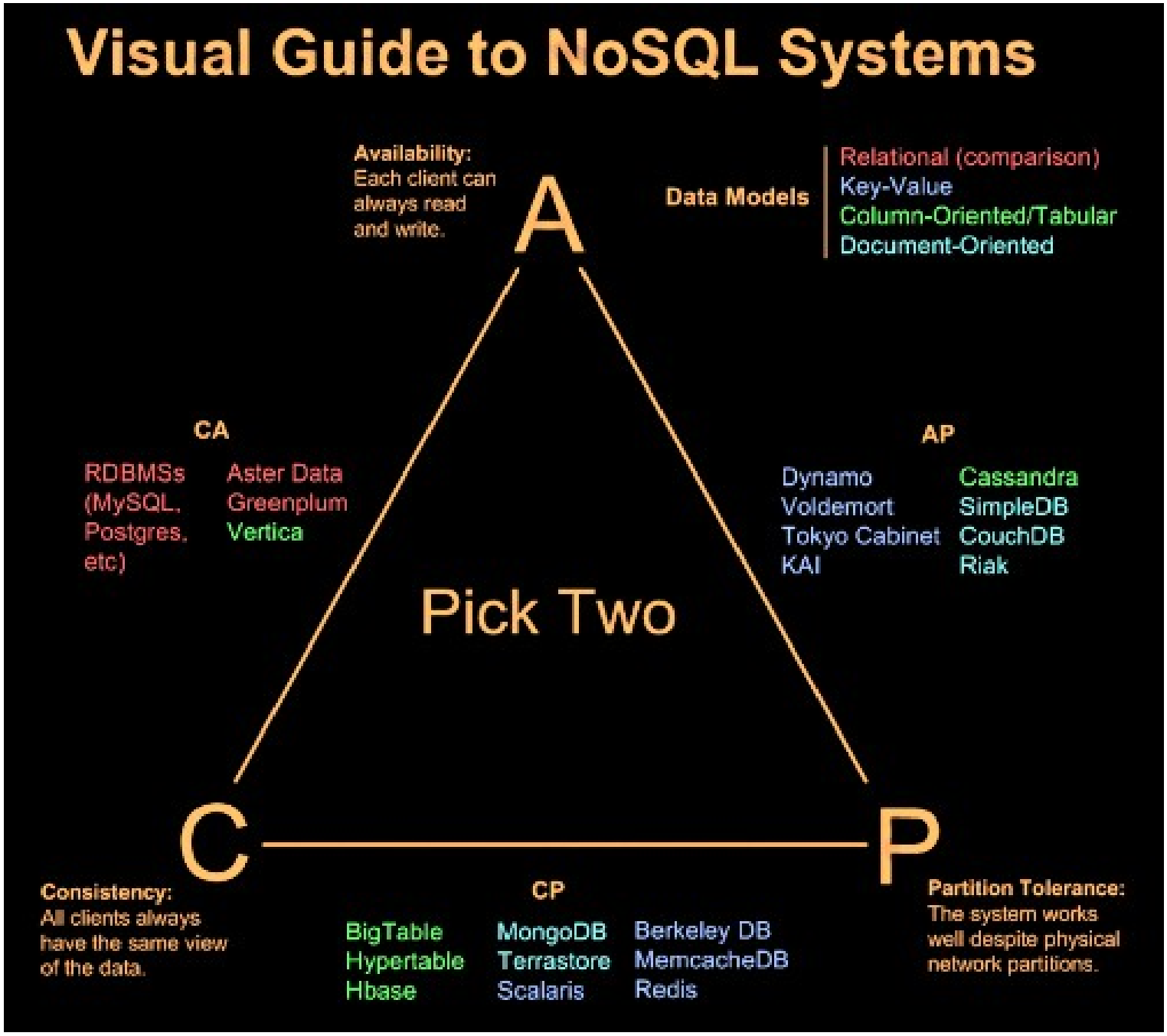
- **Acronym contrived to be the opposite of ACID**
  - **Basically Available,**
  - **Soft state,**
  - **Eventually Consistent**
- **Characteristics**
  - **Weak consistency – stale data OK**
  - **Availability first**
  - **Best effort**
  - **Approximate answers OK**
  - **Aggressive (optimistic)**
  - **Simpler and faster**

Brewer’s CAP Theorem

A distributed system can support only two of the following characteristics:

- **Consistency**
- **Availability**
- **Partition tolerance**

Visual Guide to NoSQL Systems



Other Non-SQL Databases

XML Databases

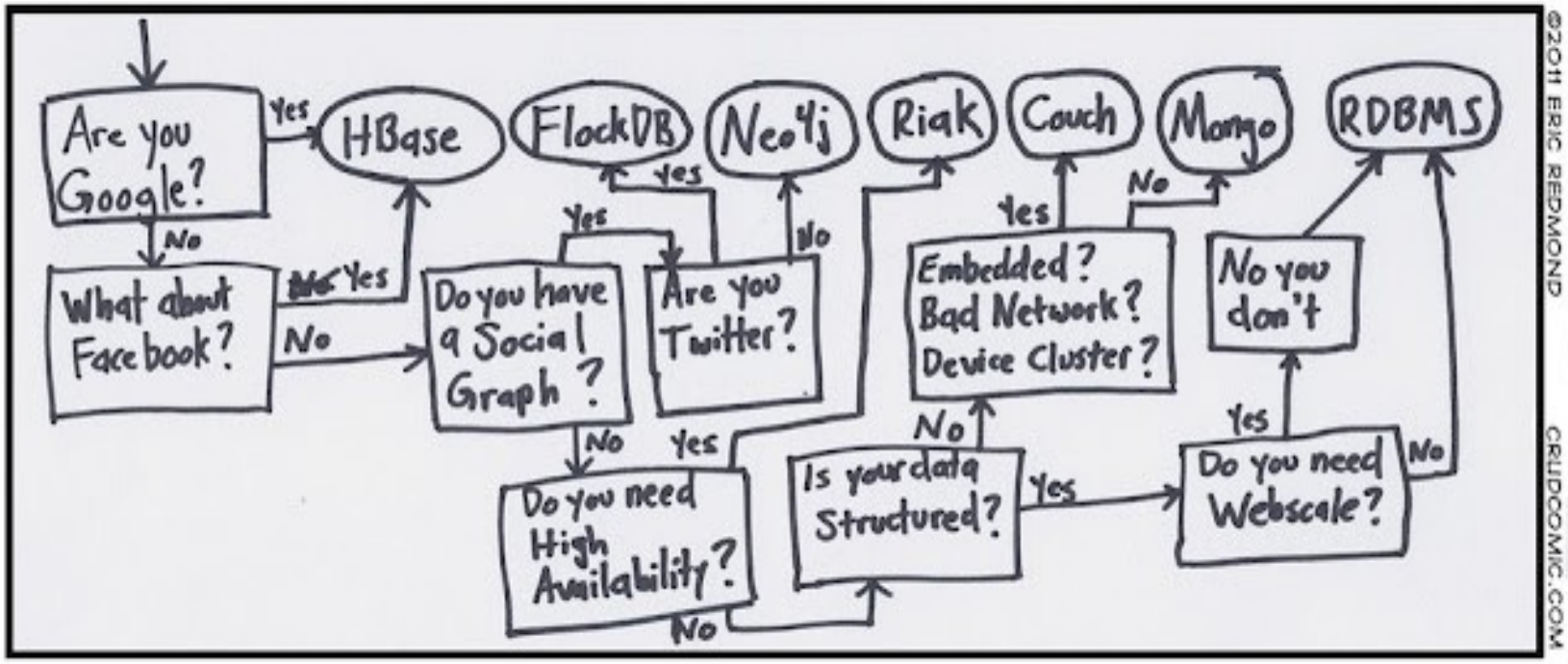
Graph Databases

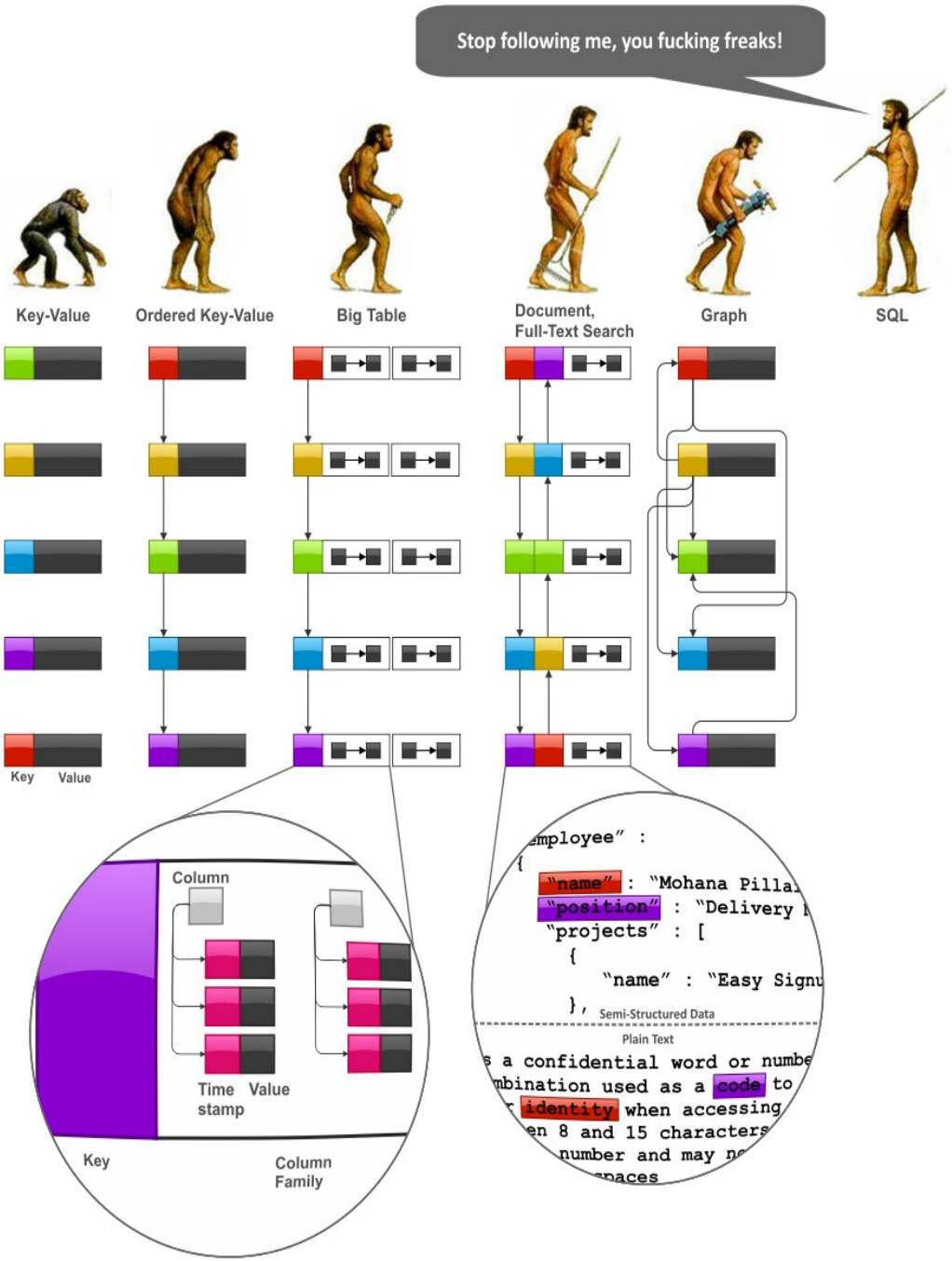
Codasyl Databases

Object Oriented Databases

Etc…

Will not address these today


Are you Google?
Yes
HBase
FlockDB
Neo4j
Riak
Couch
Mongo
RDBMS
No
What about Facebook?
Yes
No
Do you have a Social Graph?
Yes
Are you Twitter?
Yes
No
Embedded? Bad Network? Device Cluster?
Yes
No
No you don't
Yes
No
Do you need High Availability?
No
Yes
Is your data Structured?
No
Yes
Do you need Webscale?
Yes
No
©2011 ERIC REDMOND
CRUDCOMIC.COM


Stop following me, you fucking freaks!
Key-Value
Ordered Key-Value
Big Table
Document,
Full-Text Search
Graph
SQL
Key
Value
Column
Time stamp
Value
Key
Column Family
"name" : "Mohana Pilla
"position" : "Delivery
"projects" : [
{
"name" : "Easy Signu
},
Semi-Structured Data
Plain Text
a confidential word or numbe
bination used as a code to
identity when accessing
en 8 and 15 characters
number and may n
paces

Storing and Modifying Data

Syntax varies

HTML

Java Script

Etc.

Asynchronous – Inserts and updates do not wait for confirmation

Versioned

Optimistic Concurrency

Retrieving Data

Syntax Varies

No set-based query language

Procedural program languages such as Java, C, etc.

Application specifies retrieval path

No query optimizer

Quick answer is important

May not be a single “right” answer

Open Source

Small upfront software costs

Suitable for large scale distribution on commodity hardware

Typical NoSQL API

- Basic API access:
  - get(key) -- Extract the value given a key
  - put(key, value) -- Create or update the value given its key
  - delete(key) -- Remove the key and its associated value
  - execute(key, operation, parameters) -- Invoke an operation to the value (given its key) which is a special data structure (e.g. List, Set, Map .... etc).

Flexible Data Model

ColumnFamily: Rockets

Key	Value
1	**Name** / **Value** name: Rocket-Powered Roller Skates toon: Ready, Set, Zoom inventoryQty: 5 brakes: false
2	**Name** / **Value** name: Little Giant Do-It-Yourself Rocket-Sled Kit toon: Beep Prepared inventoryQty: 4 brakes: false
3	**Name** / **Value** name: Acme Jet Propelled Unicycle toon: Hot Rod and Reel inventoryQty: 1 wheels: 1

NoSQL Data Storage: Classification

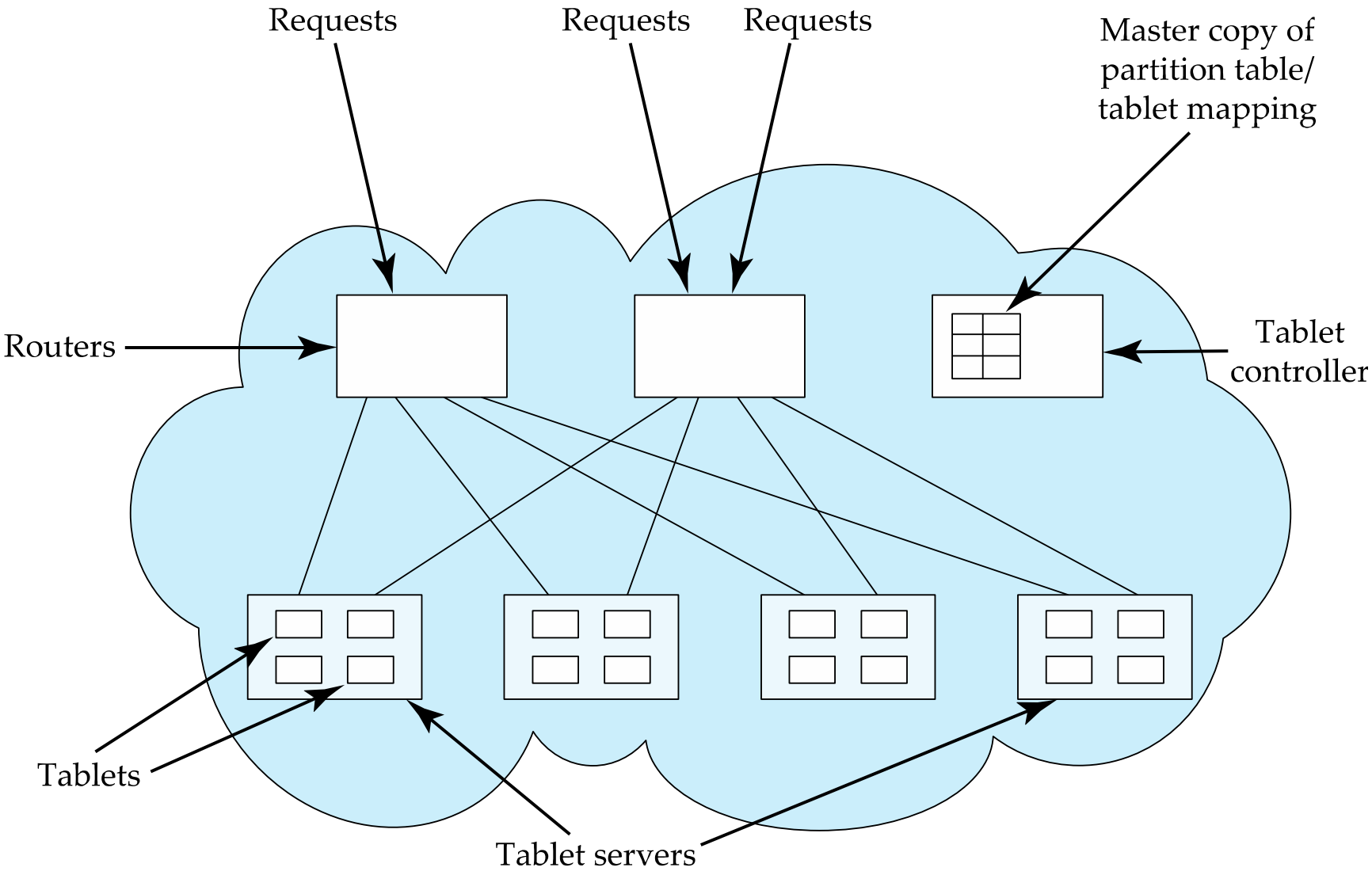
Uninterpreted key/value or ‘the big hash table’.

- Amazon S3 (Dynamo)

Flexible schema

- BigTable, Cassandra, HBase (ordered keys, semi-structured data),
- Sherpa/PNuts (unordered keys, JSON)
- MongoDB (based on JSON)
- CouchDB (name/value in text)

PNUTS Data Storage Architecture



What does NoSQL Not Provide?

- Joins
- Group by
  - But PNUTS provides interesting materialized view approach to joins/aggregation.
- ACID transactions
- SQL
- Integration with applications that are based on SQL

Should I be using NoSQL Databases?

- NoSQL Data storage systems makes sense for applications that need to deal with very very large semi-structured data
  - Log Analysis
  - Social Networking Feeds
- Most of us work on organizational databases, which are not that large and have low update/query rates
  - regular relational databases are the correct solution for such applications

Summary

SQL Databases

- Predefined Schema
- Standard definition and interface language
- Tight consistency
- Well defined semantics

Summary

- NoSQL databases reject:
  - Overhead of ACID transactions
  - “Complexity” of SQL
  - Burden of up-front schema design
  - Declarative query expression
  - Yesterday’s technology
  - Programmer responsible for
  - Step-by-step procedural language
  - Navigating access path
  - No predefined Schema
  - Per-product definition and interface language
  - Getting an answer quickly is more important than getting a correct answer

Web References

“NoSQL -- Your Ultimate Guide to the Non - Relational Universe!”
http://nosql-database.org/links.html

“NoSQL (RDBMS)”
http://en.wikipedia.org/wiki/NoSQL

PODC Keynote, July 19, 2000. *Towards Robust*. *Distributed Systems*. Dr. Eric A. *Brewer*. Professor, UC Berkeley. Co-Founder & Chief Scientist, Inktomi **. *www.eecs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf***

“Brewer's CAP Theorem” posted by Julian Browne, January 11, 2009.
http://www.julianbrowne.com/article/viewer/brewers-cap-theorem

“How to write a CV” Geek & Poke Cartoon
http://geekandpoke.typepad.com/geekandpoke/2011/01/nosql.html