

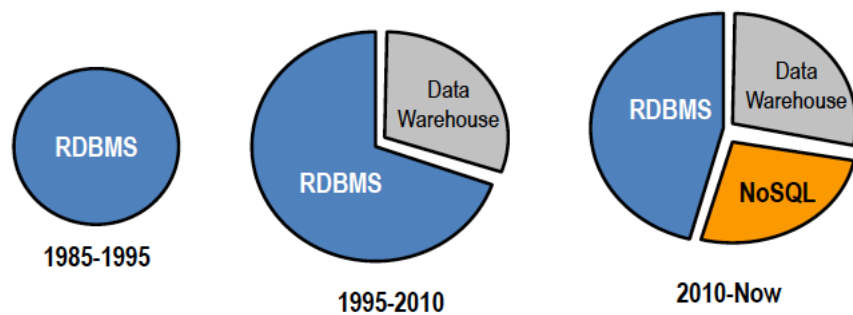


**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**  
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

# NoSQL

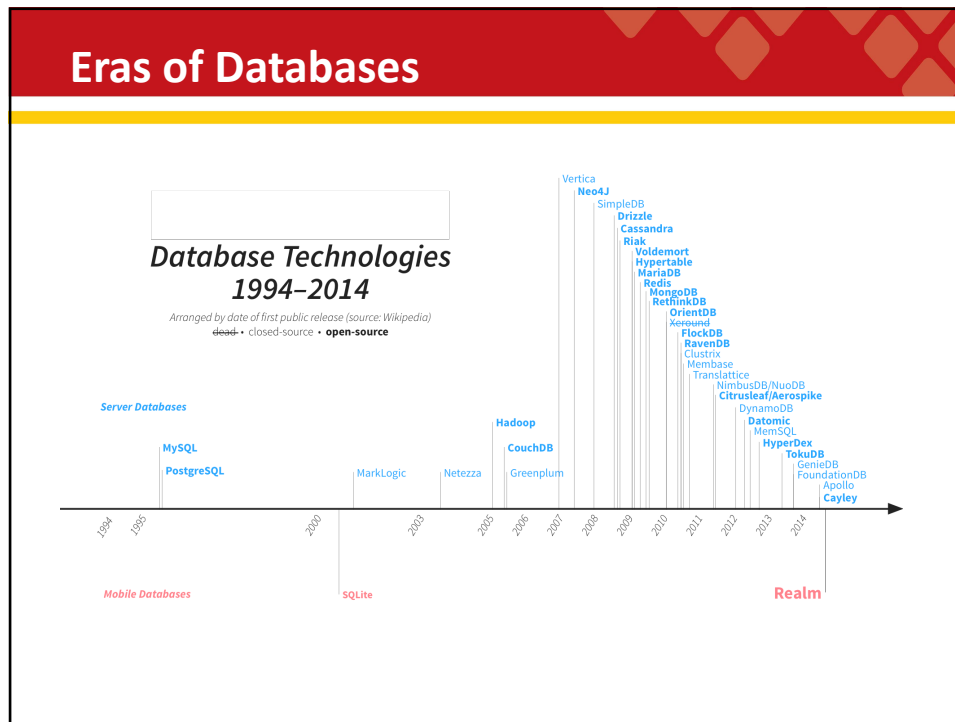
1

## Eras of Databases

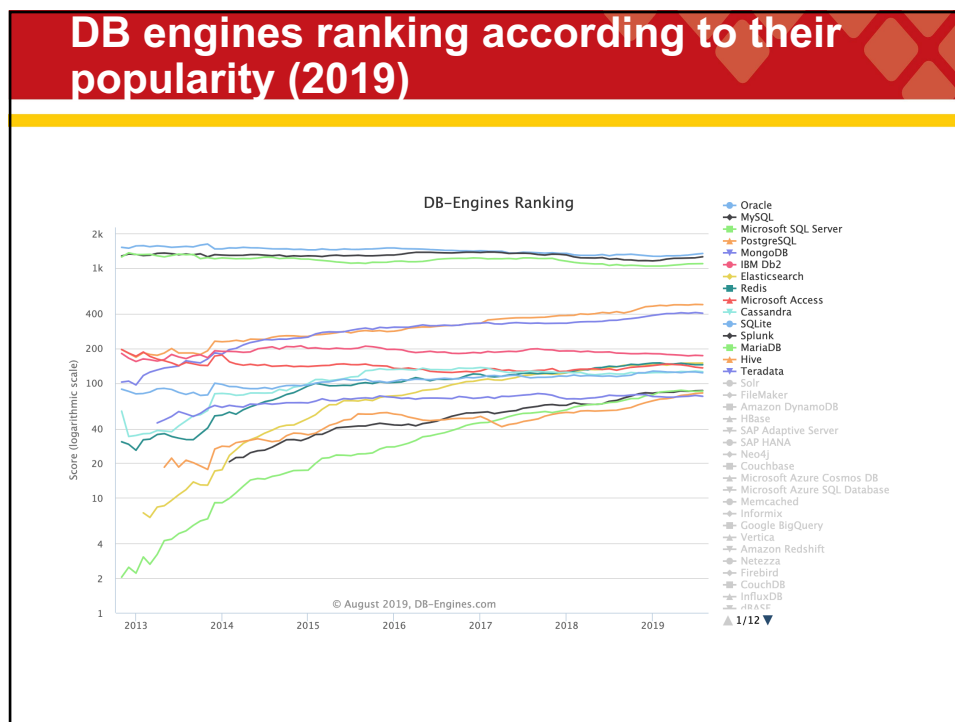


2

2

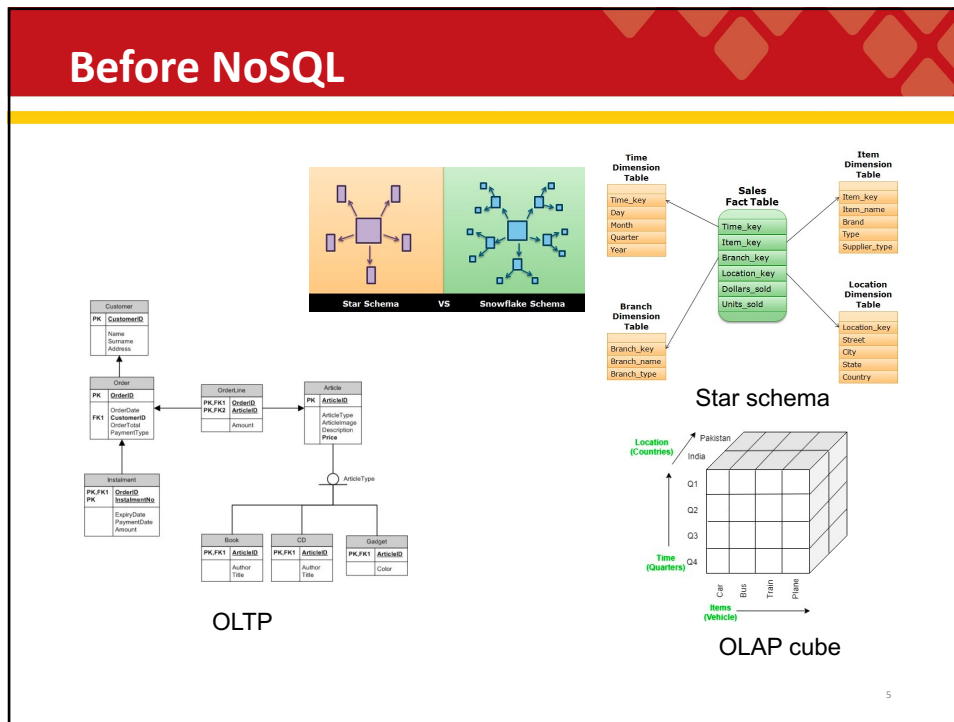


3



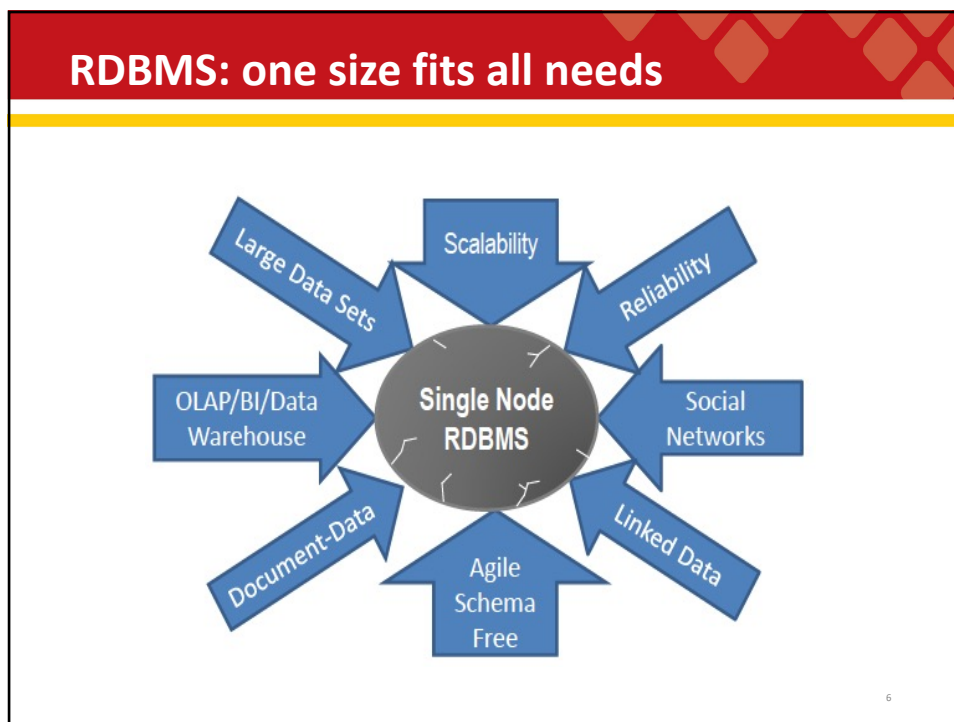
4

## Before NoSQL



5

## RDBMS: one size fits all needs



6

## ICDE 2005 conference

### "One Size Fits All": An Idea Whose Time Has Come and Gone

Authors: [Michael Stonebraker](#) StreamBase Systems, Inc.  
[Ugur Cetintemel](#) Brown University and StreamBase Systems, Inc.



2005 Article

Published in:

· Proceeding  
 ICDE '05 Proceedings of the 21st International Conference on Data Engineering  
 Pages 2-11

April 05 - 08, 2005

IEEE Computer Society Washington, DC, USA ©2005

[table of contents](#) ISBN:0-7695-2285-8 doi>[10.1109/ICDE.2005.1](#)

[Bibliometrics](#)

· Citation Count: 73  
 · Downloads (cumulative): 0  
 · Downloads (12 Months): 0  
 · Downloads (6 Weeks): 0

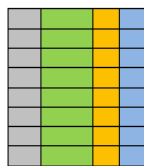
The last 25 years of commercial DBMS development can be summed up in a single phrase: "one size fits all". This phrase refers to the fact that **the traditional DBMS architecture (originally designed and optimized for business data processing) has been used to support many data-centric applications** with widely varying characteristics and requirements. In this paper, we argue that this concept is no longer applicable to the database market, and that the commercial world will fracture into a collection of independent database engines ...

7

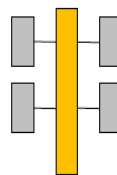
7

## After is NoSQL

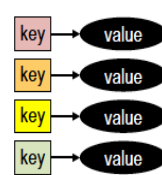
### Relational



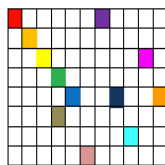
### Analytical (OLAP)



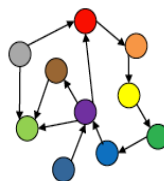
### Key-Value



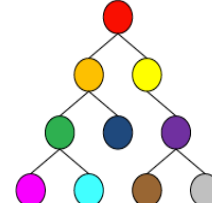
### Column-Family



### Graph



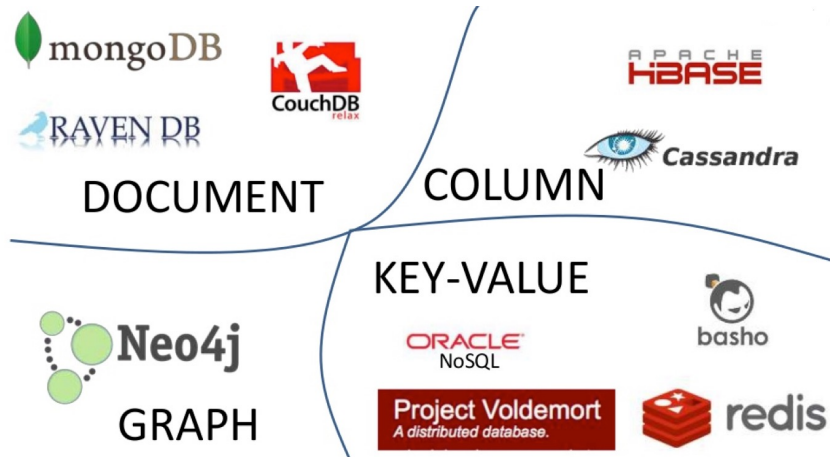
### Document



8

8

## NoSQL landscape

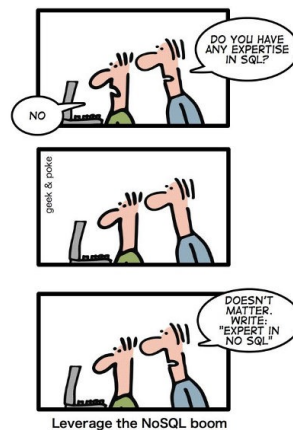


9

9

## How to write a CV

### HOW TO WRITE A CV



Leverage the NoSQL boom

10

10

## Why NoSQL

- Web applications have different needs
  - Horizontal scalability – lowers cost
  - Geographically distributed
  - Elasticity
  - Schema less, flexible schema for semi-structured data
  - Easier for developers
  - Heterogeneous data storage
  - High Availability/Disaster Recovery
- Web applications do not always need
  - Transaction
  - Strong consistency
  - Complex queries

11

11

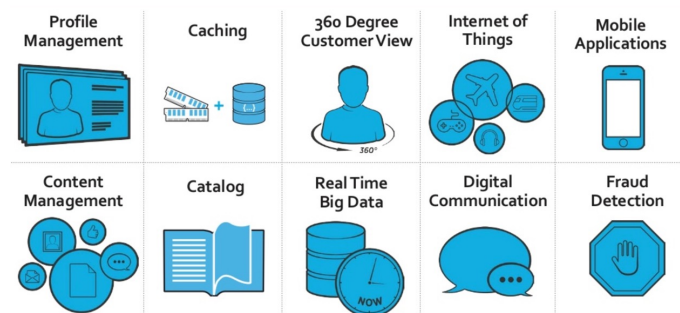
## SQL vs NoSQL

SQL	NoSQL
Gigabytes to Terabytes	Petabytes(1kTB) to Exabytes(1kPB) to Zetabytes(1kEB)
Centralized	Distributed
Structured	Semi structured and Unstructured
Structured Query Language	No declarative query language
Stable Data Model	Schema less
Complex Relationships	Less complex relationships
ACID Property	Eventual Consistency
Transaction is priority	High Availability, High Scalability
Joins Tables	Embedded structures

12

## NoSQL use cases

- Massive data volume at scale (Big volume)
  - Google, Amazon, Yahoo, Facebook – 10-100K servers
- Extreme query workload (Big velocity)
- High availability
- Flexible, schema evolution

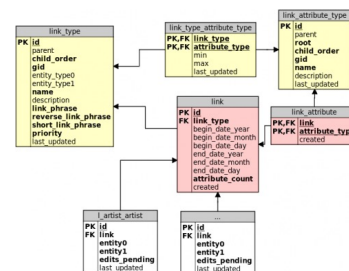


13

13

## Relational data model revisited

- Data is usually stored in row by row manner (row store)
- Standardized query language (SQL)
- **Data model defined before you add data**
- Joins merge data from multiple tables
  - Results are tables
- **Pros:** Mature ACID transactions with fine-grain security controls, widely used
- **Cons:** Requires up front data modeling, does not scale well



Oracle, MySQL, PostgreSQL,  
Microsoft SQL Server, IBM  
DB/2

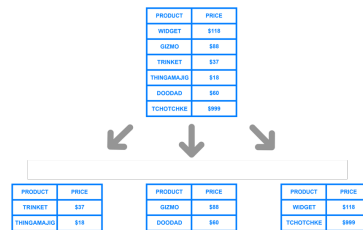
14

14

## Key/value data model

- Simple key/value interface
  - GET, PUT, DELETE
- Value can contain any kind of data
- Super fast and easy to scale (no joins)
- Examples
  - Berkley DB, Memcache, DynamoDB, Redis, Riak

key	value
firstName	Bugs
lastName	Bunny
location	Earth



15

15

## Key/value vs. table

- A table with two columns and a simple interface
  - Add a key-value
  - For this key, give me the value
  - Delete a key



Key	Value

↑ string datatype      ↑ Blob datatype

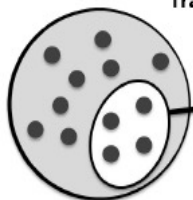
16

16



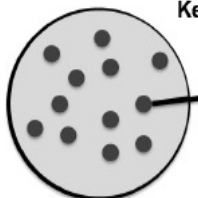
## Key/value vs. Relational data model

### Traditional Relational Model



- Result set based on row values
- Value of rows for large data sets must be indexed
- Values of columns must all have the same data type

### Key-Value Store Model

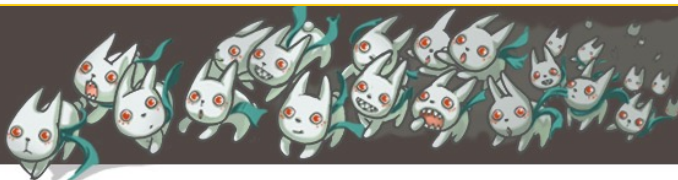


- All queries return a single item
- No indexes on values
- Values may contain any data type

17

17

## Memcached



- Open source in-memory key-value caching system
- Make effective use of RAM on many distributed web servers
- Designed to speed up dynamic web applications by alleviating database load
  - Simple interface for highly distributed RAM caches
  - 30ms read times typical
- Designed for quick deployment, ease of development
- APIs in many languages

18

18

## Redis

- Open source in-memory key-value store with optional durability
- Focus on high speed reads and writes of common data structures to RAM
- Allows simple lists, sets and hashes to be stored within the value and manipulated
- Many features that developers like expiration, transactions, pub/sub, partitioning



19

19

## Amazon DynamoDB

- Scalable key-value store
- Fastest growing product in Amazon's history
- Focus on throughput on storage and predictable read and write times
- Strong integration with S3 and Elastic MapReduce

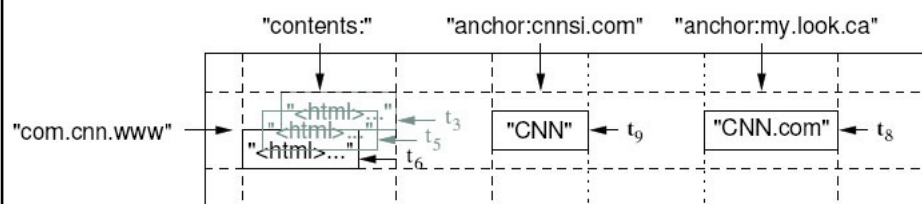


20

20

## Column family store

- Dynamic schema, column-oriented data model
- Sparse, distributed persistent multi-dimensional sorted map
- (row, column (family), timestamp) -> cell contents

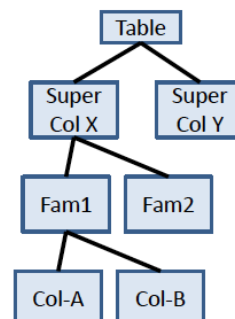


22

22

## Column families

- Group columns into "Column families"
- Group column families into "Super-Columns"
- Be able to query all columns with a family or super family
- Similar data grouped together to improve speed

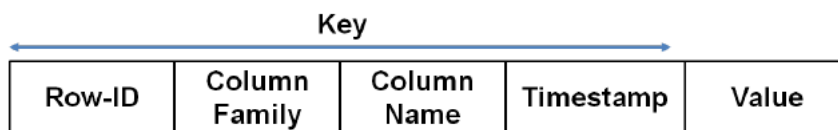


23

23

## Column family data model vs. relational

- Sparse matrix, preserve table structure
  - One row could have millions of columns but can be very sparse
- Hybrid row/column stores
- Number of columns is extendible
  - New columns to be inserted without doing an "alter table"



24

24

## Bigtable

- ACM TOCS 2008
- Fault-tolerant, persistent
- Scalable
  - Thousands of servers
  - Terabytes of in-memory data
  - Petabyte of disk-based data
  - Millions of reads/writes per second, efficient scans
- Self-managing
  - Servers can be added/removed dynamically
  - Servers adjust to load imbalance

### Bigtable: A Distributed Storage System for Structured Data

Full Text: PDF Get this Article

Authors: [Fay Chang](#) [Google, Inc.](#)  
[Jeffrey Dean](#) [Google, Inc.](#)  
[Sanjay Ghemawat](#) [Google, Inc.](#)  
[Wilson C. Hsieh](#) [Google, Inc.](#)  
[Deborah A. Wallach](#) [Google, Inc.](#)  
[Mike Burrows](#) [Google, Inc.](#)  
[Tushar Chandra](#) [Google, Inc.](#)  
[Andrew Fikes](#) [Google, Inc.](#)  
[Robert E. Gruber](#) [Google, Inc.](#)



#### Published in:

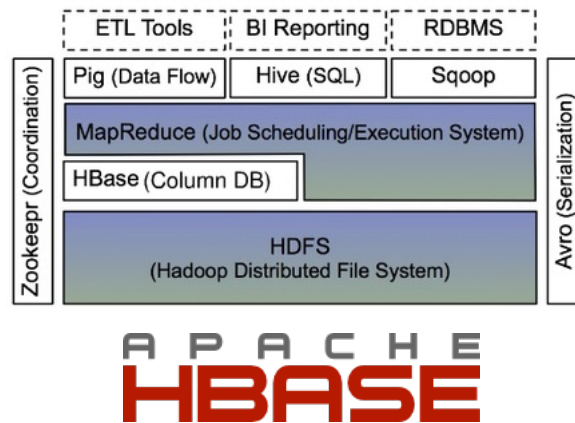
Journal  
 ACM Transactions on Computer Systems (TOCS) [TOCS Homepage](#) [Article](#)  
 Volume 26 Issue 2, June 2008  
 Article No. 4  
 ACM New York, NY, USA  
[table of contents](#) [doi>10.1145/1365815.1365816](#)

25

25

## Apache Hbase

- Open-source Bigtable, written in JAVA
- Part of Apache Hadoop project



26

26

## Apache Cassandra

- Apache open source column family database
- Supported by DataStax
- Peer-to-peer distribution model
- Strong reputation for linear scale out (millions of writes/second)
- Written in Java and works well with HDFS and MapReduce

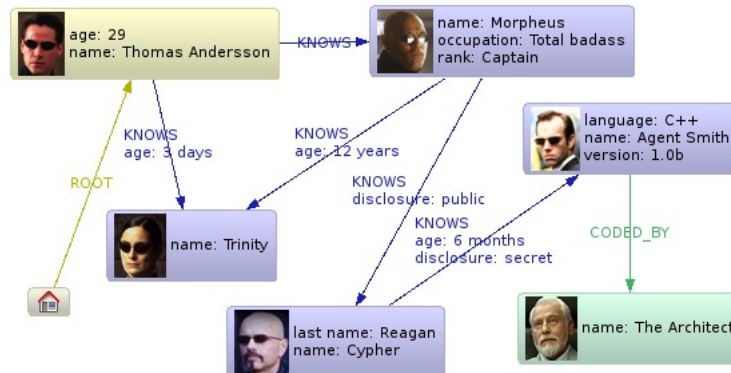


27

27

## Graph data model

- Core abstractions: Nodes, Relationships, Properties on both



28

28

## Graph database store

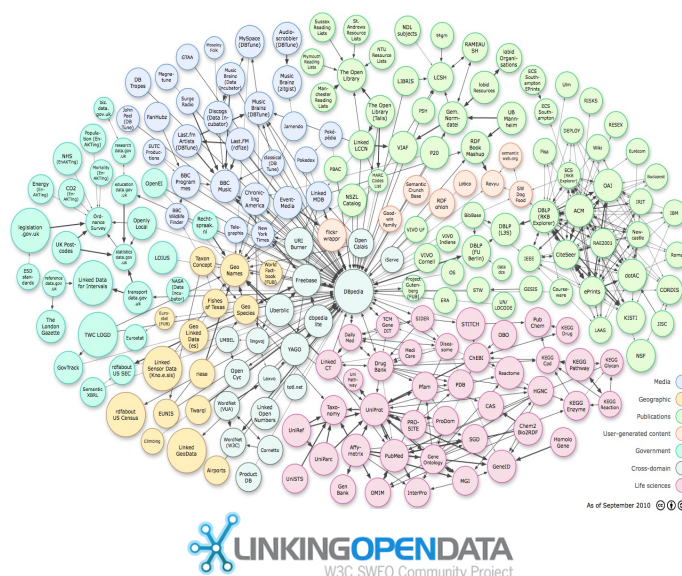
- A database stored data in an explicitly graph structure
- Each node knows its adjacent nodes
- Queries are really graph traversals



29

29

## Linking open data



33

33

## Neo4j

- Graph database designed to be easy to use by Java developers
- Disk-based (not just RAM)
- Full ACID
- High Availability (with Enterprise Edition)
- 32 Billion Nodes, 32 Billion Relationships, 64 Billion Properties
- Embedded java library
- REST API



34

34

## Document store

- Documents, not value, not tables
- JSON or XML formats
- Document is identified by ID
- Allow indexing on properties

```
{
  person: {
    first_name: "Peter",
    last_name: "Peterson",
    addresses: [
      {street: "123 Peter St"},
      {street: "504 Not Peter St"}
    ],
  }
}
```

35

35

## MongoDB

- Open Source JSON data store created by 10gen
- Master-slave scale out model
- Strong developer community
- Sharding built-in, automatic
- Implemented in C++ with many APIs (C++, JavaScript, Java, Perl, Python etc.)



40

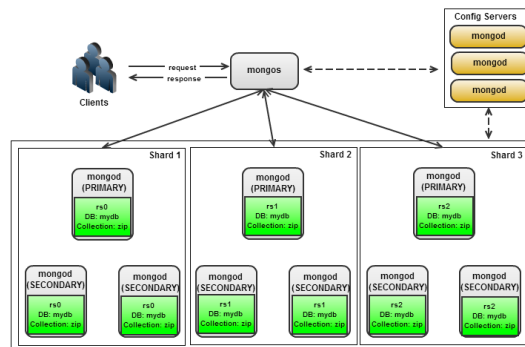
40



## MongoDB architecture

- Replica set
  - Copies of the data on each node
  - Data safety
  - High availability
  - Disaster recovery
  - Maintenance
  - Read scaling

- Sharding
  - “Partitions” of the data
  - Horizontal scale



41

## Apache CouchDB


- Apache project
- Open source JSON data store
- Written in ERLANG
- RESTful JSON API
- B-Tree based indexing, shadowing b-tree versioning
- ACID fully supported
- View model
- Data compaction
- Security



Apache CouchDB™ is a database that uses **JSON** for documents, **JavaScript** for **MapReduce** indexes, and regular **HTTP** for its **API**

42

42



**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**  
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

**Thank you for your attention!**  
**Q&A**