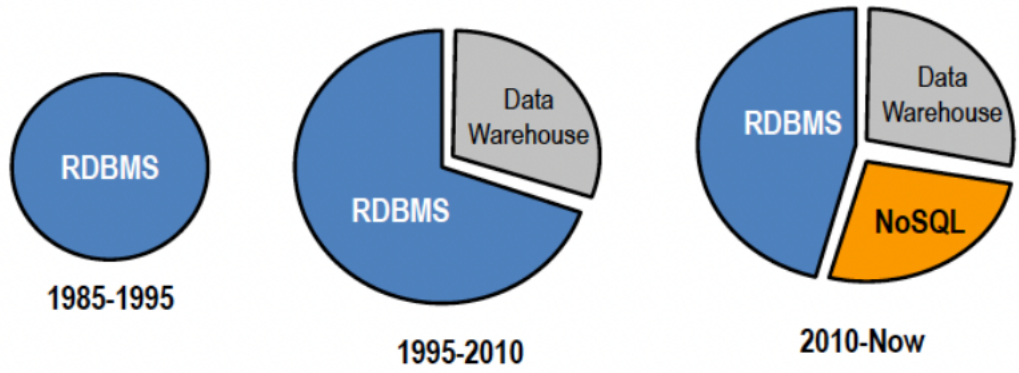
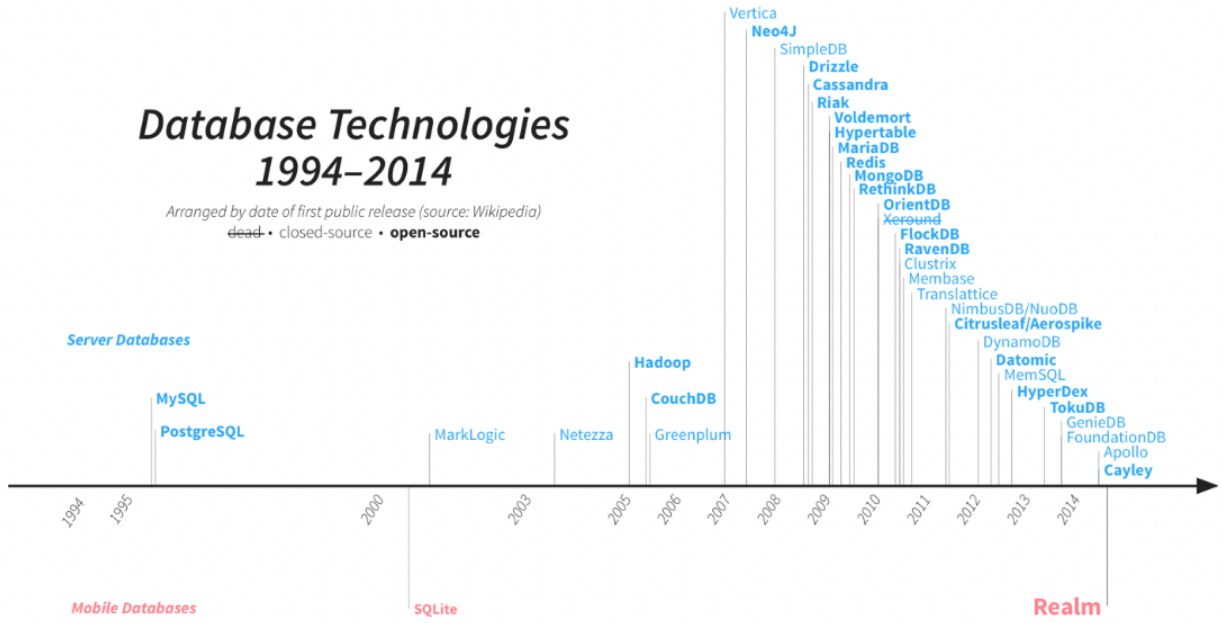
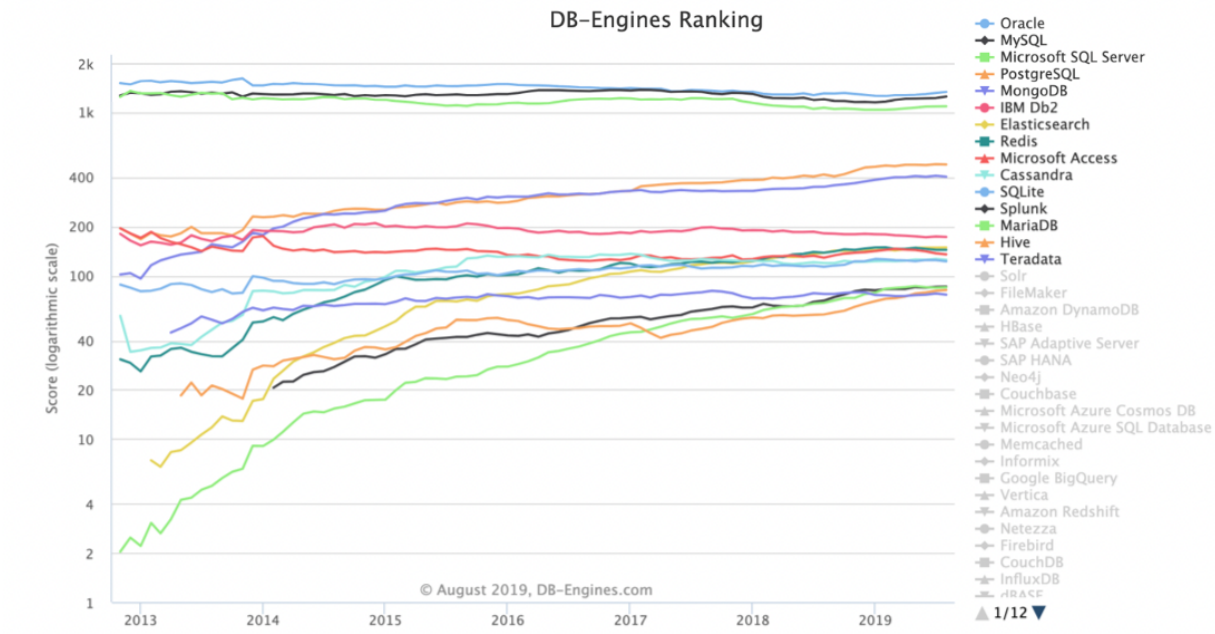
NoSQL

Eras of Databases

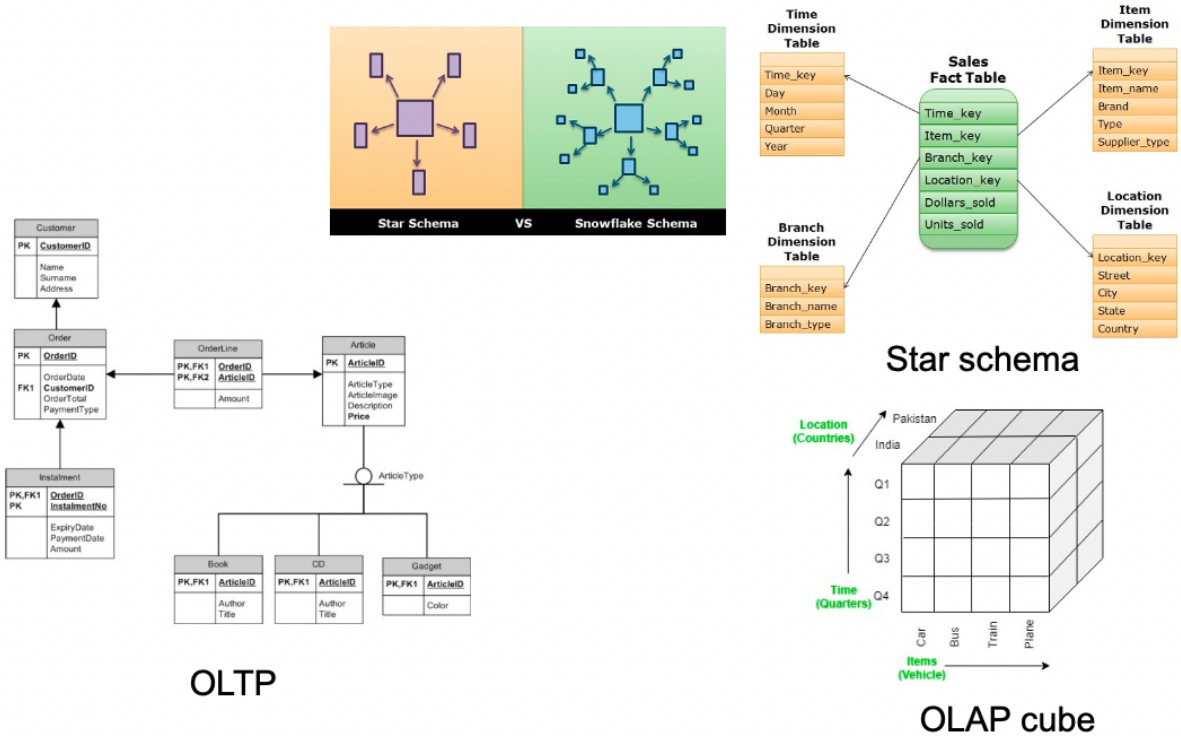




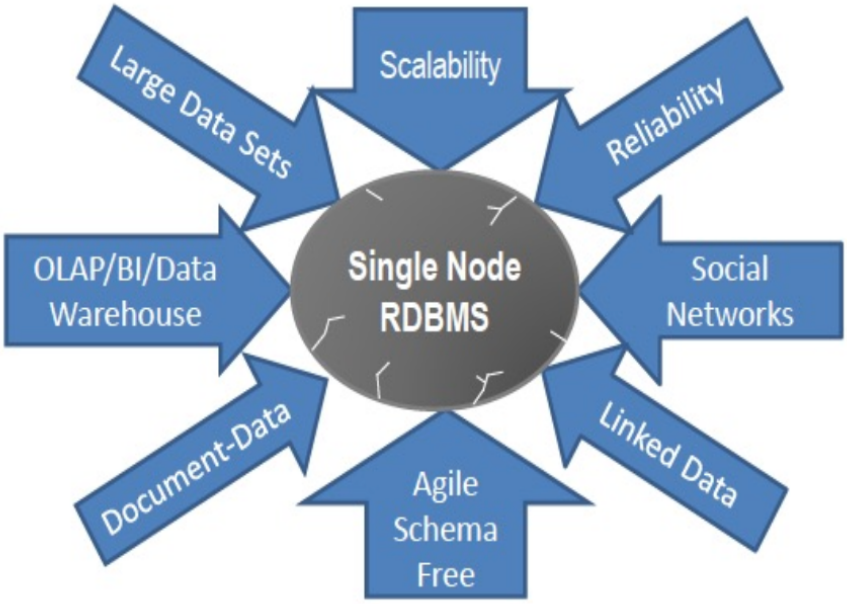
DB engines ranking according to their popularity (2019)



Before NoSQL



RDBMS: One Size Fits all Needs



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

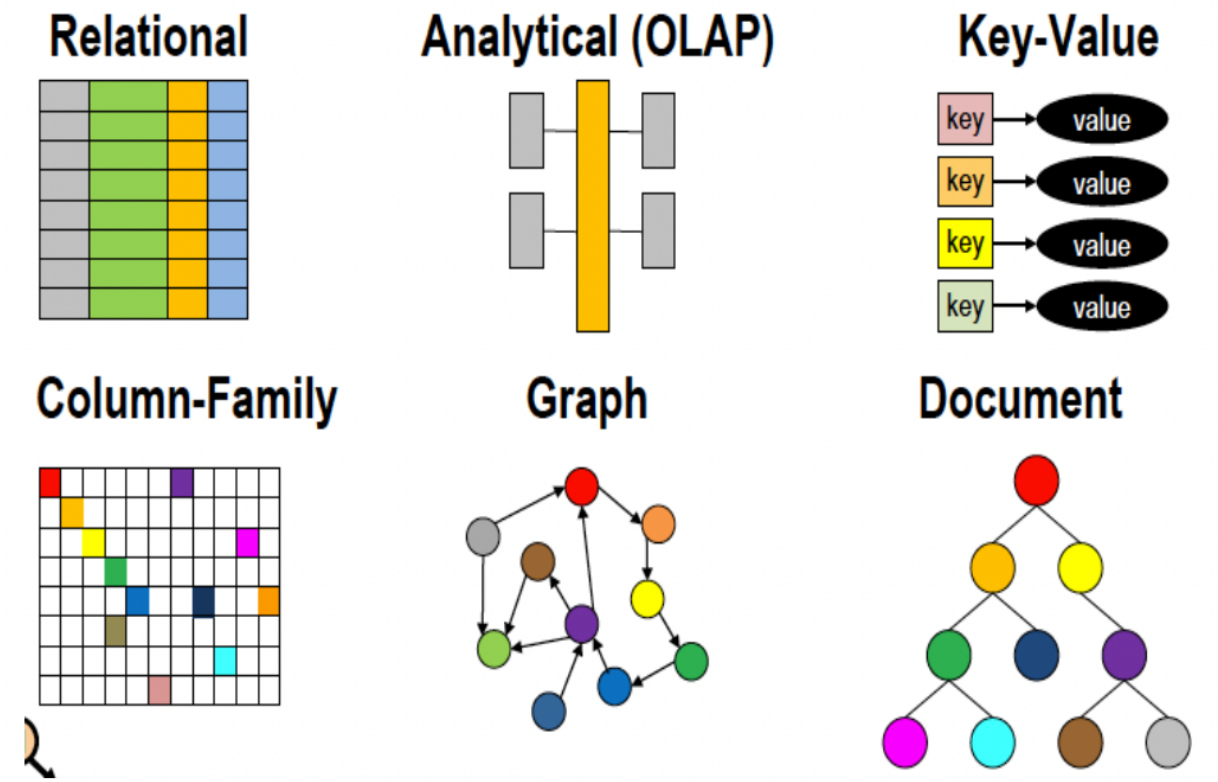
Published in:
· Proceeding
ICDE '05 Proceedings of the 21st International Conference on Data Engineering
Pages 2-11

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

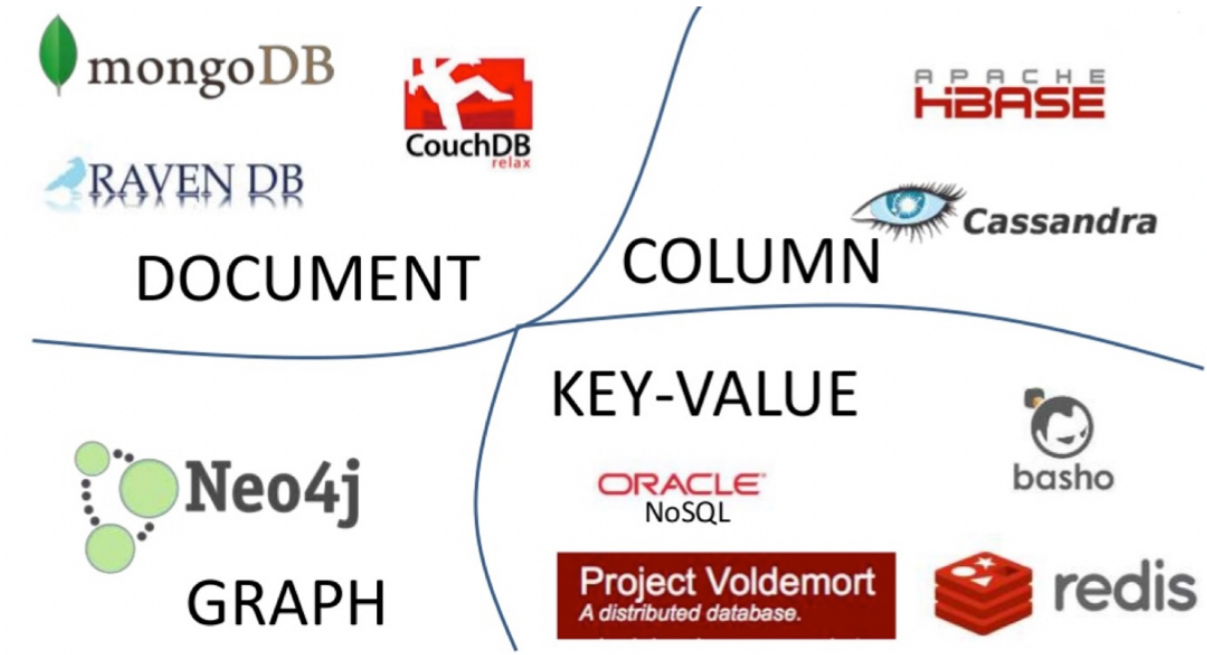
Bibliometrics
· Citation Count: 73
· Downloads (cumulative): 0
· Downloads (12 Months): 0
· Downloads (6 Weeks): 0

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 application**s 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 ...

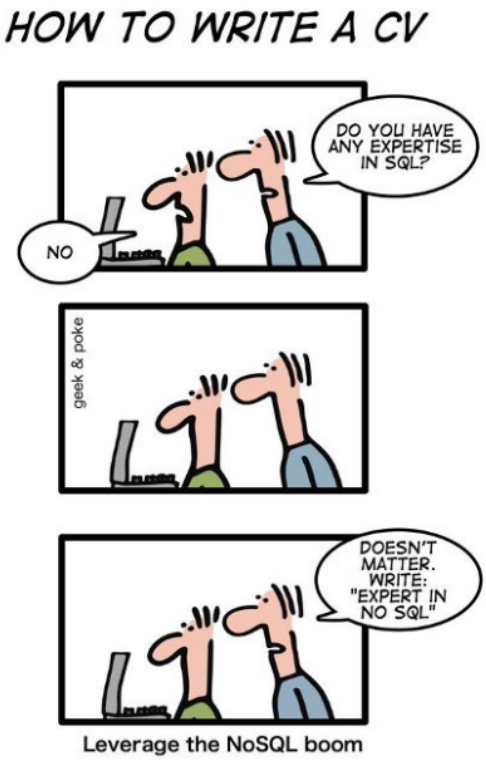
After is NoSQL



NoSQL landscape



How to write a CV



Why NoSQL

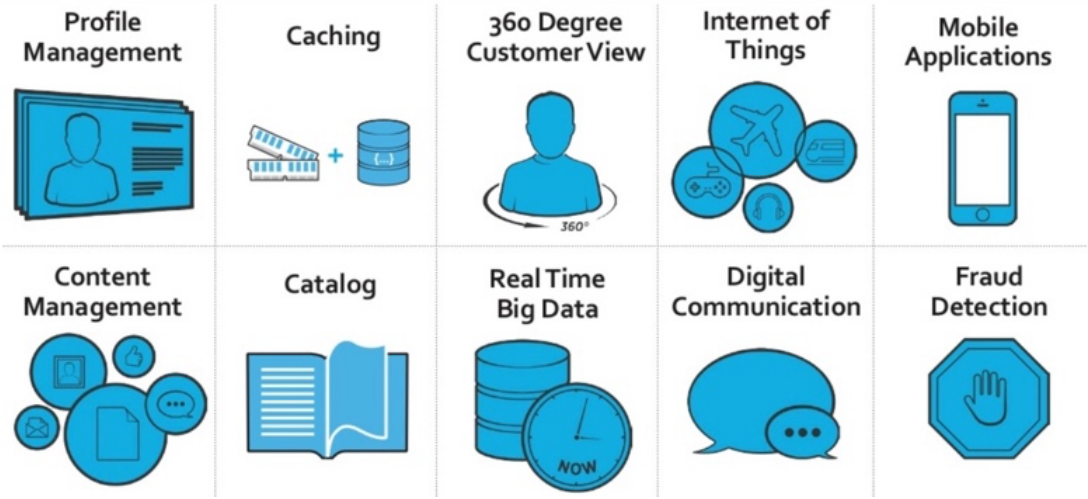
- Web application have different needs:
  - Horizontal scalability – lower cost.
  - Geographically distributed.
  - Elasticity.
  - Scheme less, flexible schema for semi-constructed data.
  - Easier for developers.
  - Heterogeneous data storage.
  - High Availability/Disaster Recovery.
- Web applications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

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

NoSQL Use Cases

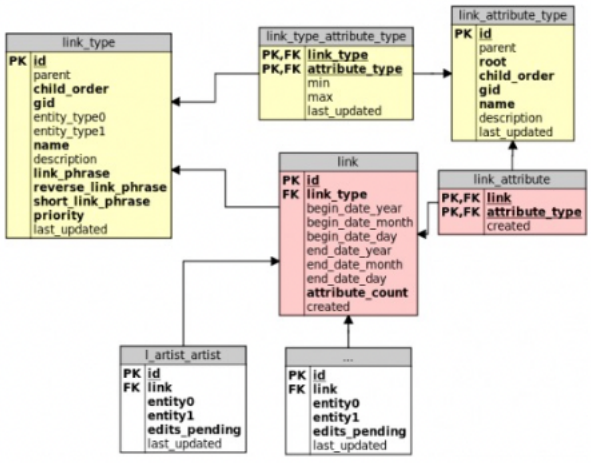
- Massive data volume at scale (Big volume).
    - Google, Amazon, Yahoo, Facebook – 10-100k servers.
- Extreme query workload (Big Velocity).
- High availability.
- Flexible, schema evolution.



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.

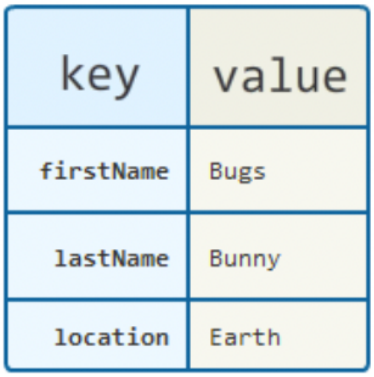
- Joins merge data from multiple tables.
    - Results are tables.
- **Pros**: Mature ACID transactions with fine-grain security controls, widely used.
- **Cons**: Requires up front data modeling, does not scale well.

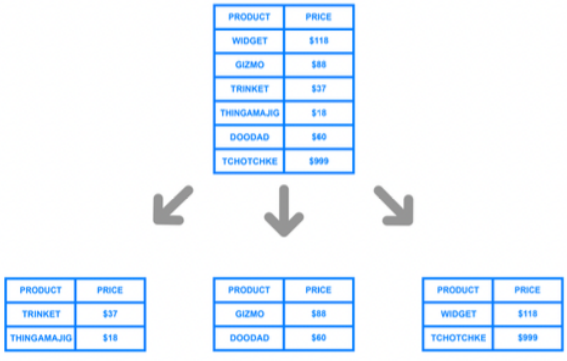


Oracle, MySQL, PostgreSQL, Microsoft SQL Server, IBM DB/2

Key/value data model

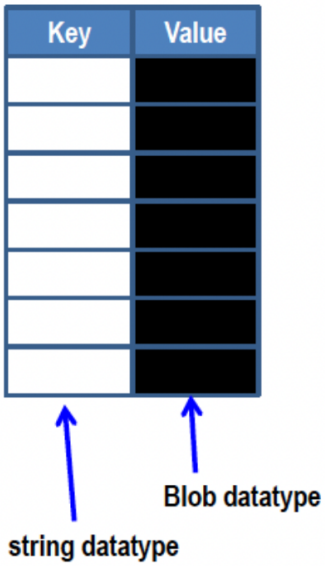
- Simple key/value interface.
    - GET/PUT/DELETE.
- Value can contain any kind of data.
- Super-fast and easy to scale (no joins).
- Examples:
    - Berkley DB, Memcached, DynamoDB, Redis, Riak.



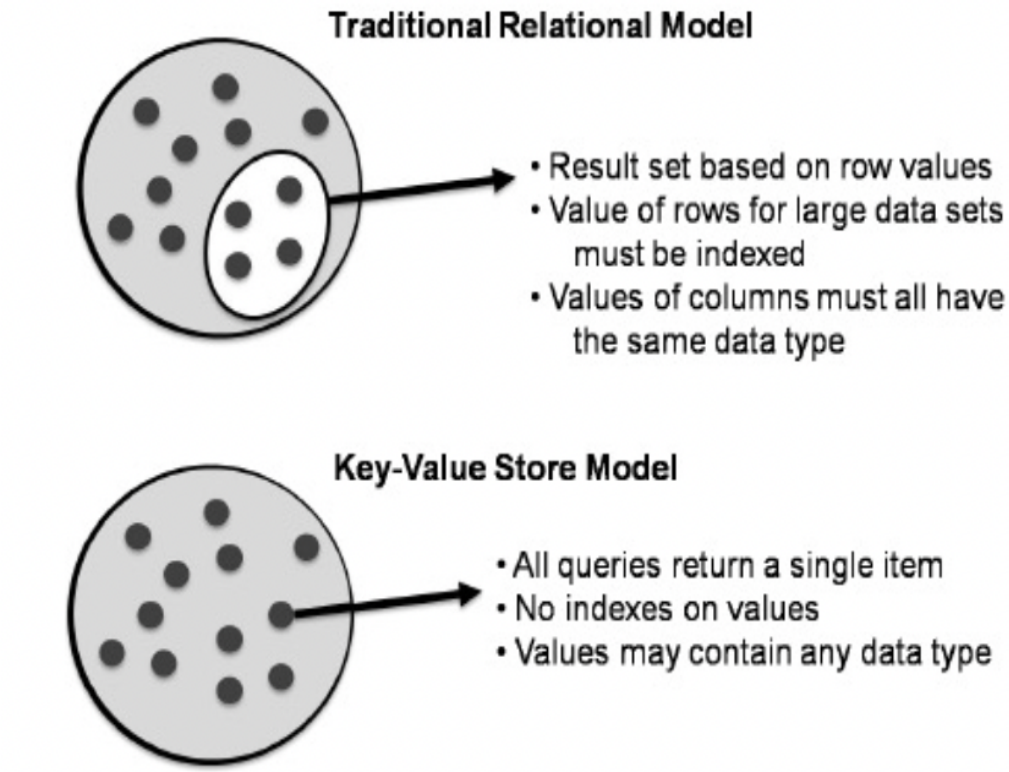


Key/value vs. table

- 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 vs. Relational data model



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.

Redis

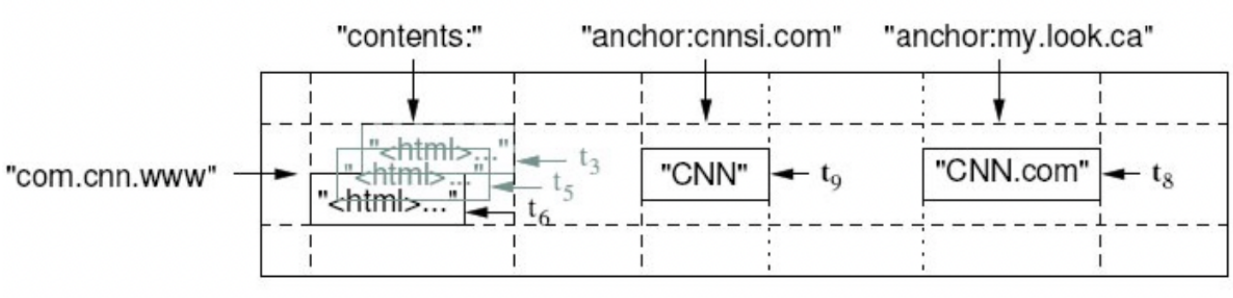
- Open source in-memory key-value store with optional durability.
- Focus on high speed reads and writes of common data structures to RAM.
- Allows simple lists, sets and hashes to be stored within the value and manipulated.
- Many features that developers like expiration, transactions, pub/sub, partitioning.

Amazon DynamoDB

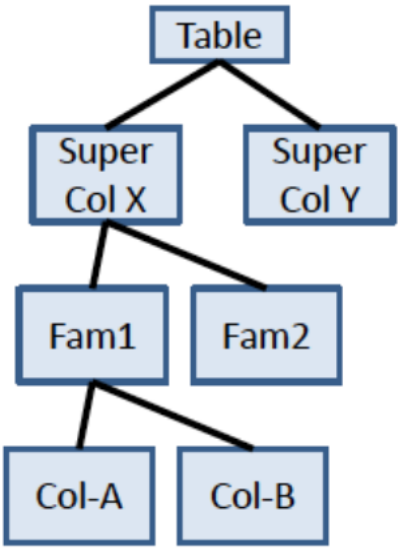
- Scalable key-value store.
- Fastest growing product in Amazon's history.
- Focus on throughput on storage and predictable read and write times.
- Strong integration with S3 and Elastic MapReduce.

Column family store

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

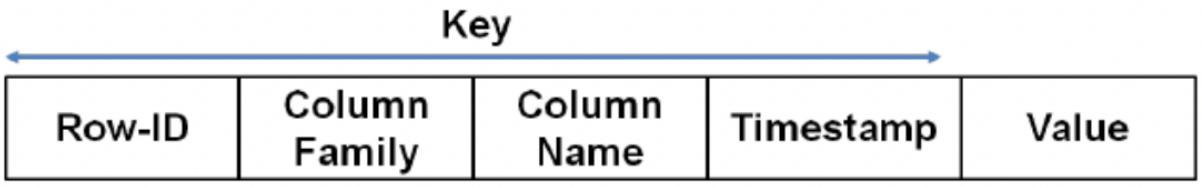


- Group columns into “Column families”.
- Group column families into “Super-Columns”.
- Be able to query all columns with a family or super family.
- Similar data grouped together to improve speed.



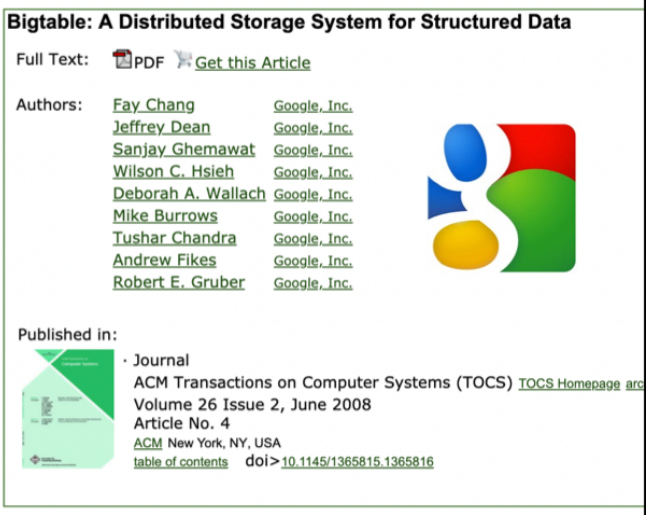
Column family data model vs. relational

- Sparse matrix, preserver 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”.



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:

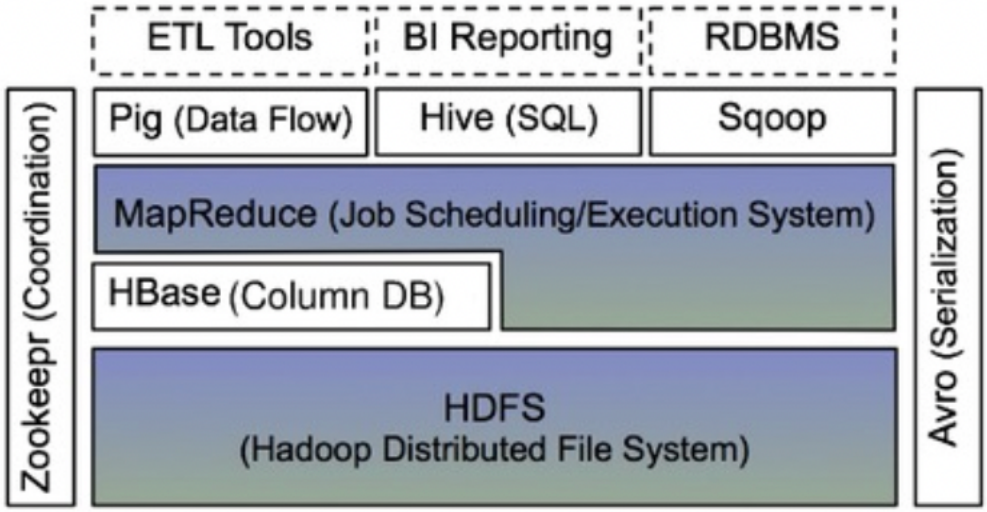
Fay Chang	Google, Inc.
Jeffrey Dean	Google, Inc.
Sanjay Ghemawat	Google, Inc.
Wilson C. Hsieh	Google, Inc.
Deborah A. Wallach	Google, Inc.
Mike Burrows	Google, Inc.
Tushar Chandra	Google, Inc.
Andrew Fikes	Google, Inc.
Robert E. Gruber	Google, Inc.

Published in:

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

Apache HBase

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



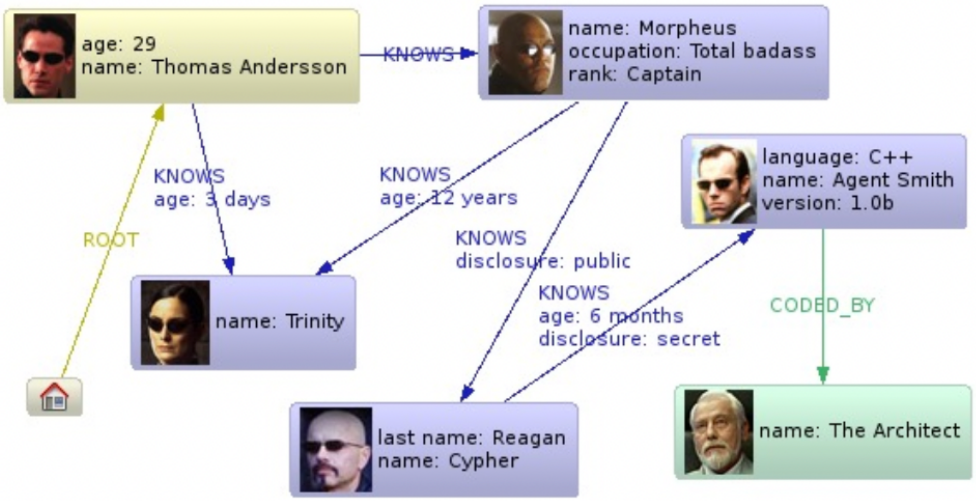
APACHE HBASE

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

Graph data model

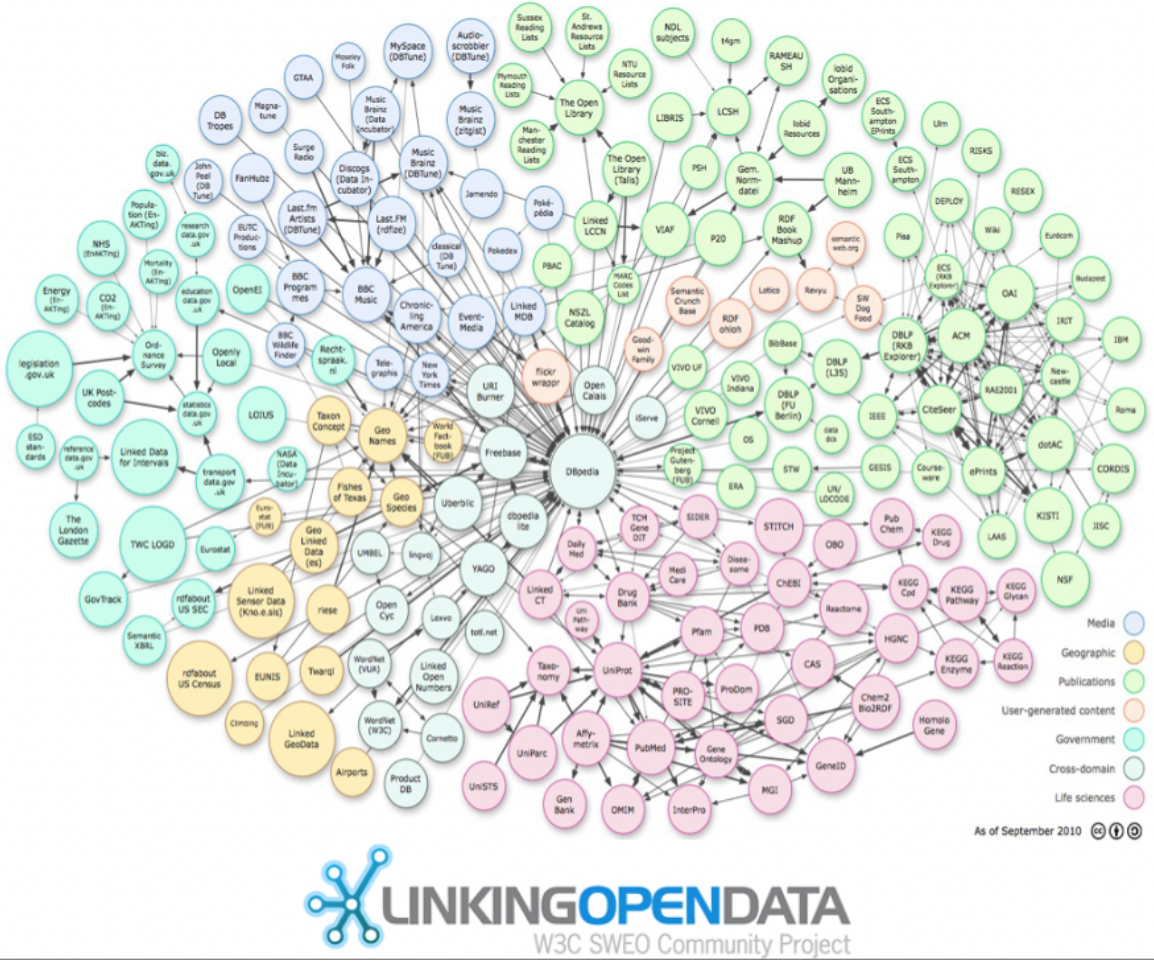
- Core abstractions: Nodes, Relationships, Properties, or both.



Graph database store

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

Linked open data



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

Document store

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

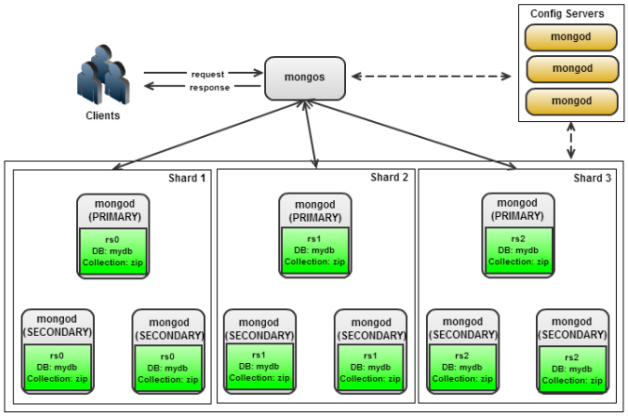
```
{
  person: {
    first_name: "Peter",
    last_name: "Peterson",
    addresses: [
      {street: "123 Peter St"},
      {street: "504 Not Peter St"}
    ],
  }
}
```

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.)

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



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

Table of Contents