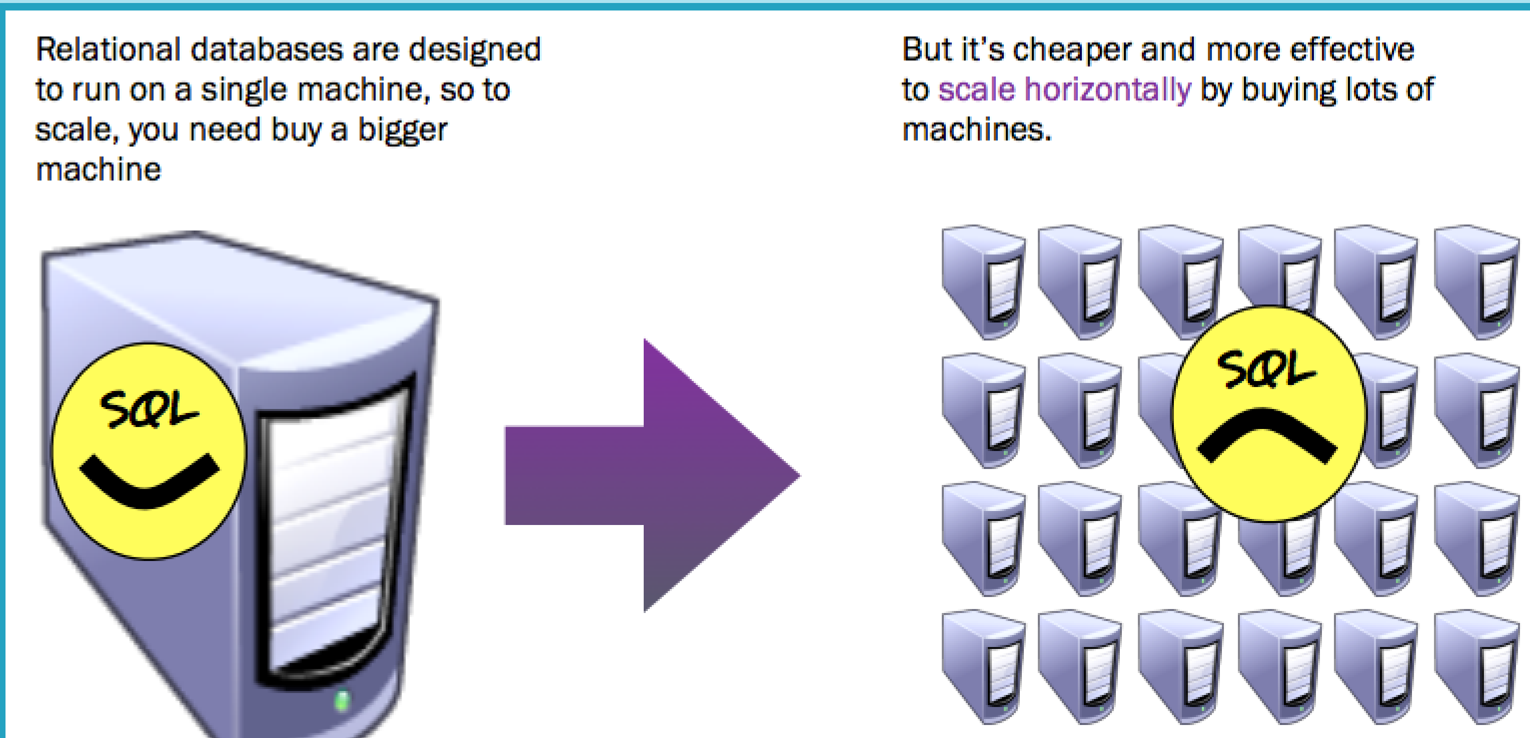
NoSQL

By Dinesh Amatya

NoSQL


Relational databases are designed to run on a single machine, so to scale, you need buy a bigger machine
SQL
But it's cheaper and more effective to scale horizontally by buying lots of machines.
SQL

NoSQL

There is no full agreement but nowadays we can summarize NoSQL definition as follows

- Next generation databases addressing some of the points:
  - non relational
  - schema-free
  - no Join
  - distributed
  - horizontally scalable with easy replication support
  - eventually consistent
  - open source

NoSQL

NoSQL databases first started out as in-house solutions to real problems:

- Amazon’s Dynamo
- Google’s BigTable
- LinkedIn’s Voldemort
- Facebook’s Cassandra
- Yahoo!’s PNUTS

NoSQL

The listed companies didn't start off by rejecting relational technologies. They tried them and found that they didn't meet their requirements:

- Huge concurrent transactions volume
- Expectations of low-latency access to massive datasets
- Expectations of nearly perfect service availability while operating in an unreliable environment

NoSQL

They tried the traditional approach

- Adding more HW
- Upgrading to faster HW as available

...and when it didn't work they tried to scale existing relational solutions:

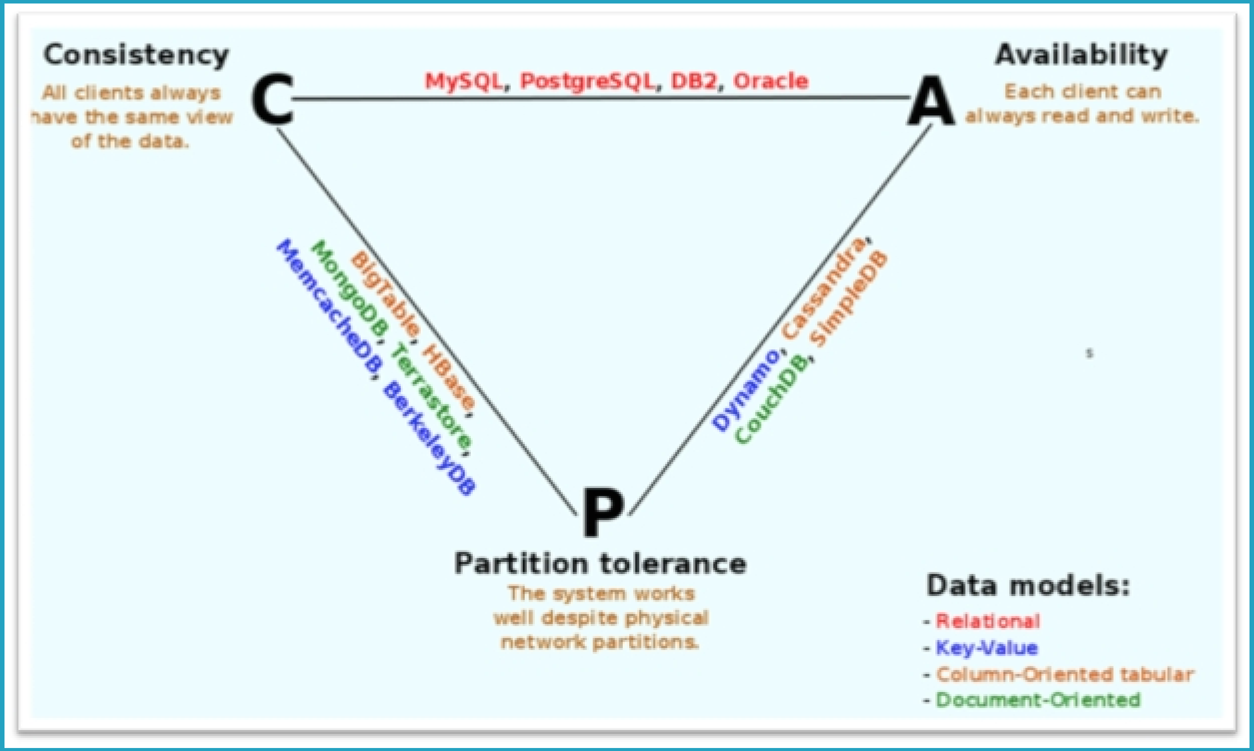
- Simplifying DB schema
- De-normalization
- Introducing numerous query caching layers
- Separating read-only from write-dedicated replicas
- Data partitioning

CAP Theorem

Formulated in 2000 by Eric Brewer

- It is impossible for a distributed computer system to simultaneously provide all three of the following guarantees:
  - Consistency (all nodes always see the same data at the same time)
  - Availability (every request always receives a response about whether it was successful or failed)
  - Partition Tolerance (the system continues to operate despite arbitrary message loss or failure of part of the system)

CAP Theorem



Consistency
All clients always have the same view of the data.
C
MySQL, PostgreSQL, DB2, Oracle
A
Availability
Each client can always read and write.
BigTable, HBase,
MongoDB, Terrastore,
MemcacheDB, BerkeleyDB
Dynamo, Cassandra,
CouchDB, SimpleDB
P
Partition tolerance
The system works well despite physical network partitions.
Data models:
- Relational
- Key-Value
- Column-Oriented tabular
- Document-Oriented

CAP Theorem And NoSQL

Most NoSQL database system architectures favor partition tolerance and availability over strong consistency

- Eventual Consistency: inconsistencies between data held by different nodes are transitory. Eventually all nodes in the system will receive the latest consistent updates.

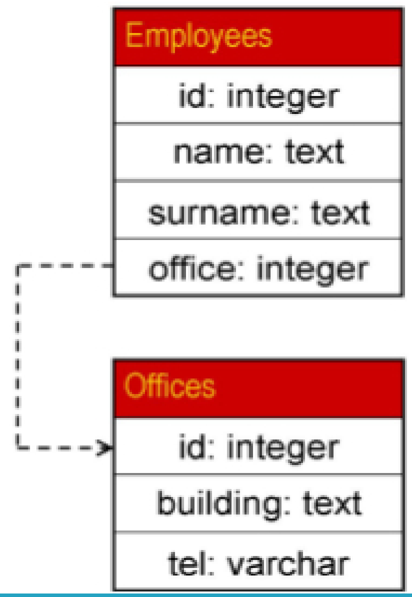
RDBMS vs NoSQL

- RDBMSs enforce global ACID properties thus allowing multiple arbitrary operations in the context of a single transaction.
- NoSQL databases enforce only local BASE properties
  - Basically Available (data is always perceived as available by the user)
  - Soft State (data at some node could change without any explicit user intervention. This follows from eventual consistency)
  - Eventually Consistent (NoSQL guarantees consistency only at some undefined future time)

NoSQL Taxonomy

- Key/Value Store
  - Amazon’s Dynamo, LinkedIn’s Voldemort, MemCached, Redis . . .
- Document Store
  - MongoDB, CouchDB, . . .
- Column Store
  - Google’s Bigtable, Apache’s HBase, Facebook’s Cassandra, . . .
- Graph Store
  - Neo4J, InfiniteGraph, . . .

RDMS Data



id	name	surname	office
1	Tom	Smith	41
2	John	Doe	42
3	Ann	Smith	41

id	building	tel
41	A4	45798
42	B7	12349

Key/Value Store

- Global collection of Key/Value pairs. Every item in the database is stored as an attribute name (key) together with its associated value
- Every key associated to exactly one value. No duplicates
- The value is simply a binary object. The DB does not associate any structure to stored values
- Designed to handle massive load of data
- Inspired by Distributed Hash Tables

Key/Value Store

Car	
Key	Attributes
1	Make: Nissan Model: Pathfinder Color: Green Year: 2003
2	Make: Nissan Model: Pathfinder Color: Blue Color: Green Year:2005 Transmission: Auto

Example of a Typical Key/Value Domain

JSON

- Stands for JavaScript Object Notation
- Syntax for storing and exchanging text information
- Uses JavaScript syntax but it is language and platform independent
- Much like XML but smaller, faster and easier to parse than XML (and human readable)
- Basic data types(Number, String, Boolean) and supports data structures as objects and arrays

JSON

```
{
     "firstName": "John",
     "lastName": "Smith",
     "age": 25,
     "address": {
         "streetAddress": "21 2nd Street",
         "city": "New York",
         "state": "NY",
         "postalCode": "10021"
     },
     "phoneNumber": [
         { "type": "home", "number": "212 555-1234" },
         { "type": "fax", "number": "646 555-4567" }
     ]
 }
```

Document Store

- *Same as Key/Value Store but pair each key with a arbitrarily complex data structure known as a document.*
- *Documents may contain many different key-value pairs or key-array pairs or even nested documents (like a JSON object).*
- *Data in documents can be understood by the DB: querying data is possible by other means than just a key (selection and projection over results are possible).*

Document Store

```
{
  "firstname": "Pramod",
  "citiesvisited": [ "Chicago", "London", "Pune", "Bangalore" ],
  "addresses": [
    { "state": "AK",
      "city": "DILLINGHAM",
      "type": "R"
    },
    { "state": "MH",
      "city": "PUNE",
      "type": "R" }
  ],
  "lastcity": "Chicago"
}
```

```
{ "firstname": "Martin",
"likes": [ "Biking","Photography" ],
  "lastcity": "Boston",
  "lastVisited":
}
```

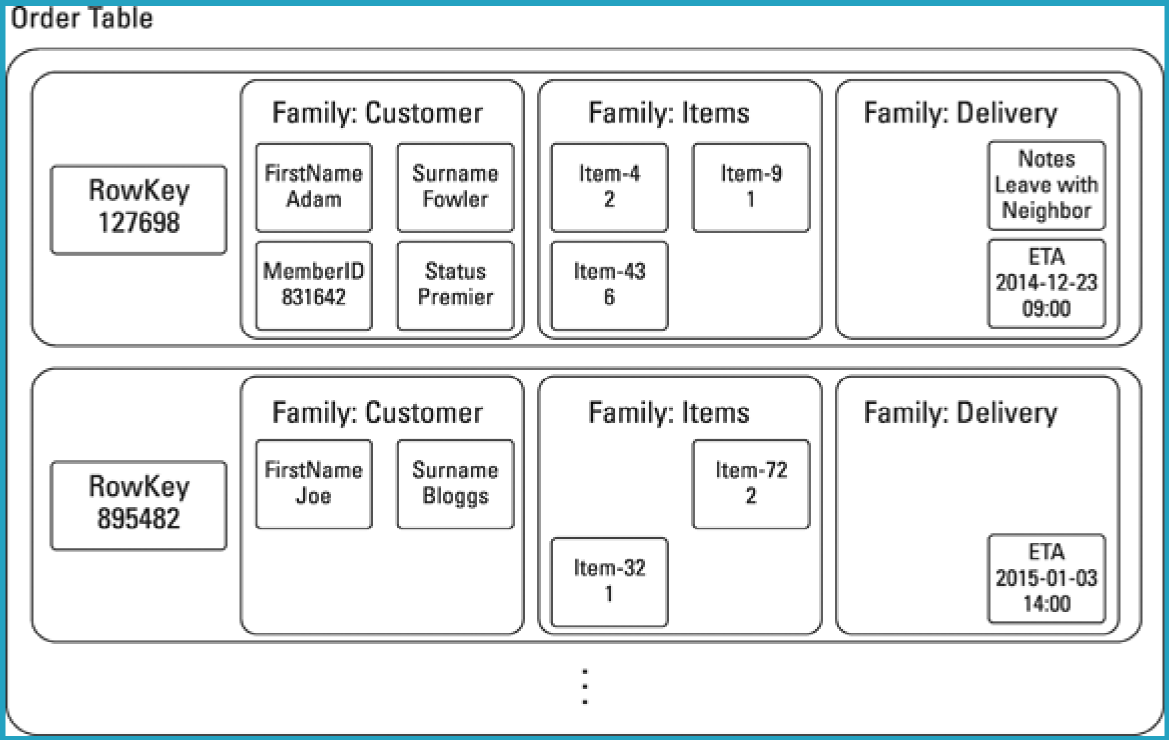
Column Store

- *”A sparse, distributed multi-dimensional sorted map”*
- *Store rows of data in similar fashion as typical RDBMSs do*
- *Rows are contained within a Column Families. Column Families can be considered as tables in RDBMSes*
- *Unlike table in RDBMSes, a Column Family can have different columns for each row it contains*
- *Each row is identified by a key that is unique in the context of a single Column Family. The same key can be however re-used within other Column Families, so it is possible to store unrelated data about the same key in different Column Families*

Column Store

- *Usually data from the same Column Family are stored contiguously on disk (and consequently on the same node of the network)*
- *Each column is simply a key/value couple*

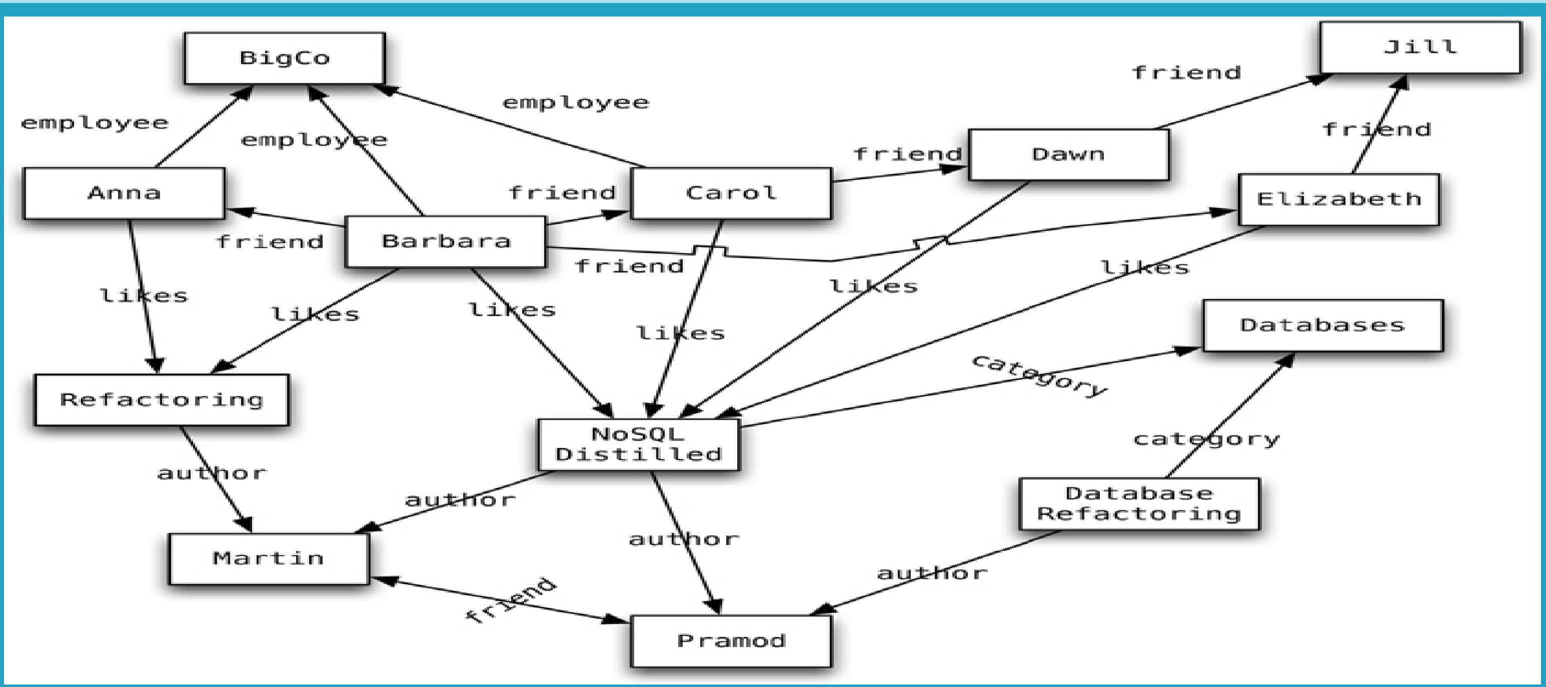
Column Store



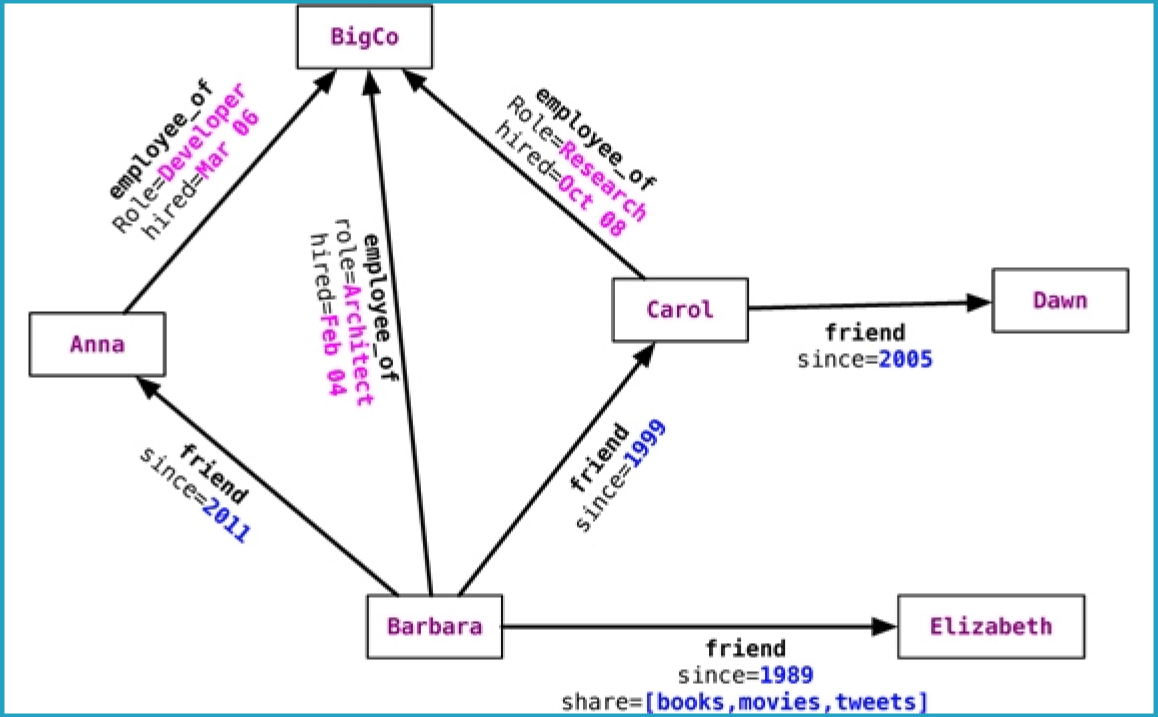
Graph Store

- Use graph structures with nodes, edges and properties to store *pieces of data and relations between them*
- *Every element contains direct pointers to its adjacent elements.*
- *Computing answers to queries over the DB corresponds to finding suitable paths on the graph structure*

Graph Store


BigCo
Jill
employee
friend
employee
employee
friend
Dawn
friend
Anna
friend
Carol
Elizabeth
friend
Barbara
friend
likes
likes
likes
likes
likes
likes
Databases
Refactoring
category
NoSQL
Distilled
category
author
Database
Refactoring
author
Martin
author
author
friend
Pramod

Graph Store



BigCo
employee_of
Role=Developer
hired=Mar 06
employee_of
role=Architect
hired=Feb 04
employee_of
Role=Research
hired=Oct 08
Anna
Carol
Dawn
friend
since=2005
friend
since=2011
friend
since=1999
Barbara
Elizabeth
friend
since=1989
share=[books,movies,tweets]

References

- http://db.cs.berkeley.edu/cs286/papers/errors-cacmblog2010.pdf
- http://www.quora.com/Can-someone-provide-an-intuitive-proof-explanation-of-CAP-theorem
- http://www.slideshare.net/yoavaa/introduction-to-the-cap-theorem
- http://netwovenblogs.com/2013/10/10/hbase-overview-of-architecture-and-data-model/
- NoSQL Distilled A Brief Guide to the Emerging World of Polyglot
- NoSQL For Dummies by Adam Fowler
- http://www.dataversity.net/acid-vs-base-the-shifting-ph-of-database-transaction-processing/
- http://databases.about.com/od/otherdatabases/a/Abandoning-Acid-In-Favor-Of-Base.htm