NoSQL Databases

Joshua Eckroth

http://www.cse.ohio-state.edu/~eckroth/nosql-guide.pdf

Outline

CAP theorem

From Eric Brewer, 2000.

- **Consistency**: all nodes see the same data at the same time
- **Availability**: every query returns a response
- **Partition tolerance**: continues to operate if nodes fail or message loss, or nodes are added/removed
- **Pick 2.**

E.g., if you want to scale up (A+P), must give up on consistency (C).

ACID

Regarding a transaction,

- **Atomicity**: all or nothing
- **Consistency**: transaction does not violate foreign key checks or other constraints
- **Isolation**: executing many transactions concurrently is same as serially; they don't interact
- **Durability**: completed transaction remains after power loss, etc. (i.e., written to disk)

ACID

- hard to achieve if transaction spans nodes
- hard to achieve without locking (which is detrimental to performance)
- give up C+I for availability, graceful degradation, and performance
- maybe even give up D for extra performance

RDBMS's are typically ACID-compliant; NoSQL systems typically aren't.

BASE

- **Basically Available**: usually works
- **Soft state**: not always consistent
- **Eventually consistent**: reads across all nodes will eventually agree (if no updates happen in the meantime)

. . . as opposed to ACID. (Obviously a "backronym.")

Key-value database

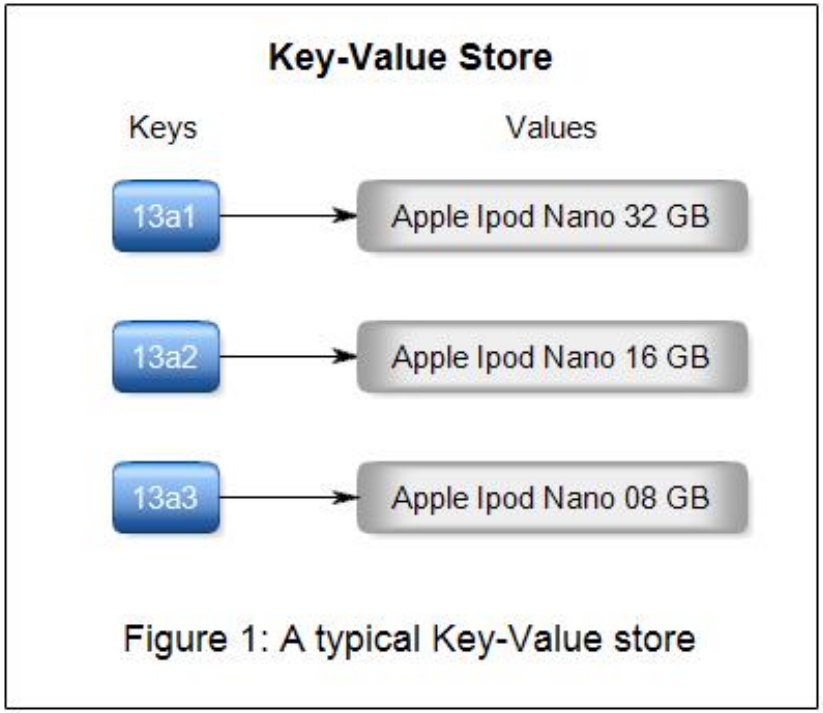


Figure 1: A typical Key-Value store

- can only retrieve by a unique key
- just get back a value; your client code interprets the value (db sees it as a blob)
- performance is uniform and fast
- horizontal scaling (seems to me) straightforward

`nosql.rishabhagrawal.com/2012/07/types-of-nosql-databases.html`

Key-value database

Project Voldemort

Auto-replication, auto-partitioning, "tunable" consistency, transparent failure handling.

API:

- `get(key)` — returns a value
- `put(key, value)`
- `delete(key)`

`http://www.project-voldemort.com`

Key-value database

Redis

- master-slave replication; a slave can become a master if the master dies
- supports sets, lists, dictionaries

`http://redis.io/`

Memcached

- in-memory cache, never saved to disk
- when full, purges by LRU
- Facebook apparently has terabytes of "in-memory cache"

`http://memcached.org/`

Document database

Storage of arbitrary dictionaries that represent documents.

- also retrieved by key, or simple queries on fields
- data structures are stored (dictionaries & lists, each can be nested in the other)
- can store a list of keys, and retrieve those recursively, in one request
- good for syncing (copy newer revisions; propagate deletes)

Document database

Example “document”:

```
{
 '_id':       '29a8f708e',
 '_rev':      12,
 'author':    'Josh',
 'title':     'My first blog post',
 'tags':      ['foo', 'bar'],
 'content':   'Welcome to my blog! ...',
 'backlinks': ['37dd04387', '883bc2ccd']
}
```

Document database

MongoDB

- most popular NoSQL db
- easy sharding
  - sharding: data subsets stored in separate machines; not replicated; no joins; painful in RDBMS
- has some query support (find based on field values, plus some operators like $<$ $>$ etc.)
- can add indexes for faster queries

`http://www.mongodb.org/`

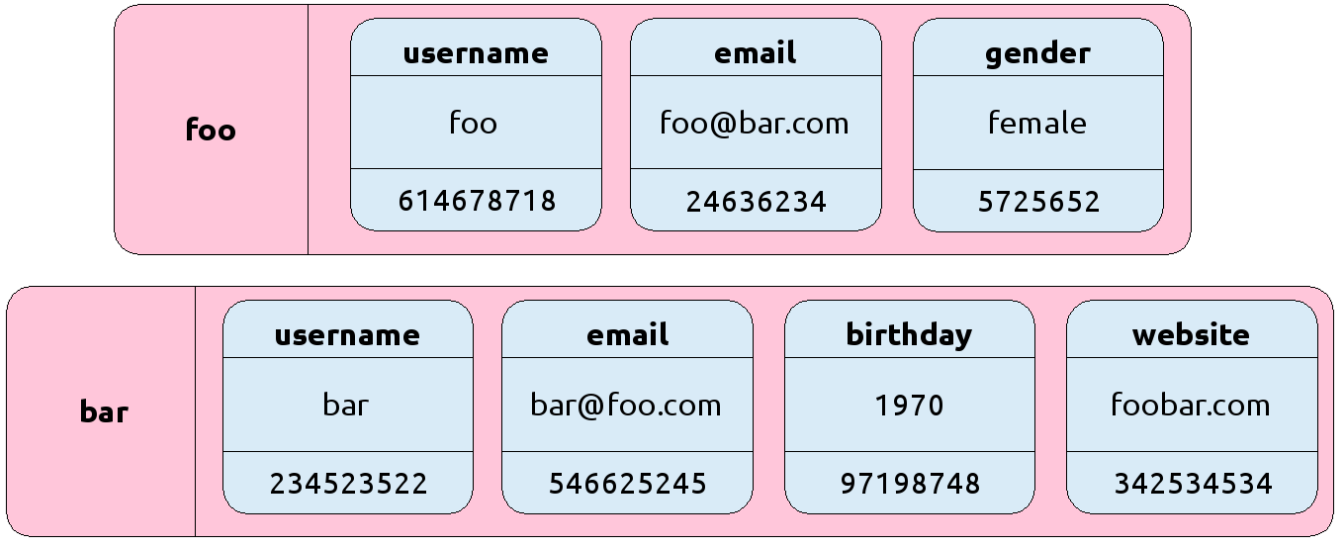
Document database

CouchDB

- create “views” of the data; these are updated when docs are updated
- if one doc is changed by two clients, two revisions are saved; merging is left to the client
- good for offline usage; changes are sync’ed later (again, no default merging)

`http://couchdb.apache.org/`

Column-family store



- key identifies a row in a table, which is part of 1+ column families
- each column family can have multiple columns
- values are timestamped (multiple versions of a value can be kept)

`http://wikis.gm.fh-koeln.de/wiki_db/Datenbanken/Cassandra`

Column-family store

Cassandra

Very high performance. Tunable consistency. Decentralized (no masters). No joins or subqueries.

API:

- `get(table, key, columnName)`
- `insert(table, key, rowMutation)`
- `delete(table, key, columnName)`

`http://cassandra.apache.org`

Graph database

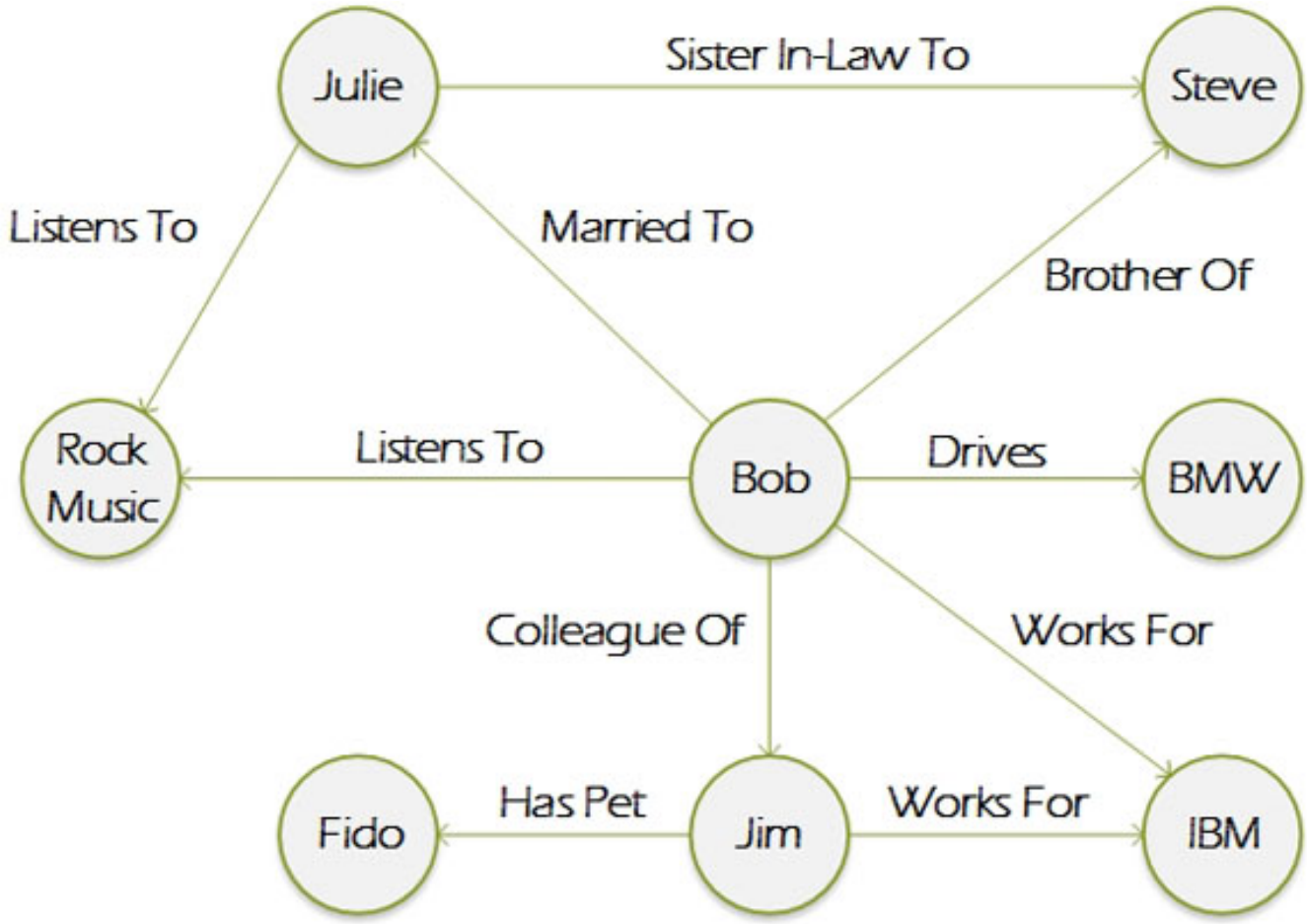
Nodes + edges that connect nodes.

- each node has arbitrary relations with others
- each node / edge has arbitrary properties
- can "walk" or query the graph according to these relations

Implementations:

- **Neo4J**: `http://www.neo4j.org`
- **HyperGraphDB**: `http://www.hypergraphdb.org`

Graph database



`http://www.computerweekly.com/feature/Whiteboard-it-the-power-of-graph-databases`

Object database

```
Foo f = new Foo();
db.persist(f);
for (Foo g : db.query("SELECT g FROM Foo")) {
  // do something with g...
}
```

- essentially, persisting live objects
- basic query support, e.g., find objects of this class, etc.
- sometimes suffers from poor indexing, poor search, memory/disk fragmentation

Implementations:

- **db4o**: `http://www.db4o.com`
- **Caché**: `http://www.intersystems.com/cache`

Choosing a DB

Do you need ACID compliance?

RDBMS	Key-value	Document	Column-family	Graph	Object
+1	-1	-1	-1	+1	+1

Do you expect your schema to change often?

RDBMS	Key-value	Document	Column-family	Graph	Object
-1	+1	+1	-1		+1

Do you expect to store terabytes / petabytes of data?

RDBMS	Key-value	Document	Column-family	Graph	Object
-1		-1	+1	+1	

Choosing a DB

Do you require syncing from mobile devices?

RDBMS	Key-value	Document	Column-family	Graph	Object
-1		+1	-1	-1	-1

Do you require horizontal scaling?

RDBMS	Key-value	Document	Column-family	Graph	Object
-1	+1	+1	+1		-1

Do you require extreme performance?

RDBMS	Key-value	Document	Column-family	Graph	Object
-1	+1		+1		

Choosing a DB

Do you require queries for arbitrary relations among data?

RDBMS	Key-value	Document	Column-family	Graph	Object
+1	-1	-1	-1	+1	

Do you want the DB to take care of complex constraints?

RDBMS	Key-value	Document	Column-family	Graph	Object
+1	-1	-1	-1	-1	

Do you want to save internal object state?

RDBMS	Key-value	Document	Column-family	Graph	Object
+1					+1

[RDBMS with Object-relational mapping (ORM)]

Trends

Search trends

http://www.google.com/trends/explore?q=nosql#q=nosql&geo=US&date=1%2F2009%2049m&cmpt=q

Job trends

http://www.indeed.com/jobtrends?q=sql%2C+nosql&l=

Job trends growth

http://www.indeed.com/jobtrends?q=sql%2C+nosql&l=&relative=1

Resources

Cassandra vs MongoDB vs CouchDB vs Redis vs Riak vs HBase vs Couchbase vs Neo4j vs Hypertable vs ElasticSearch vs Accumulo vs VoltDB vs Scalaris comparison
by Kristof Kovacs
`http://kkovacs.eu/cassandra-vs-mongodb-vs-couchdb-vs-redis/`

NoSQL Databases, a free book by Christof Strauch
`http://www.christof-strauch.de/nosqldbs.pdf`

The NoSQL Ecosystem, free book chapter by Adam Marcus
`http://www.aosabook.org/en/nosql.html`