NoSQL Databases

Tejas Parikh

CS 5200, March 2017

a bit about me

- Graduated in 2009 from NEU with M.S in Computer Science
- Past
  - Advisory Software Engineer & Team Lead at IBM Watson
- Current
  - Lead Software Engineer at Nielsen
  - Part-time Lecture at NEU teaching graduate course on Cloud Computing (CSYE 6225)

Relational Databases

What is a relational database?

Relational databases are what most of us are all used to. They have been around since late 70s.

The term "relational database" was invented by E. F. Codd at IBM in 1970.

An RDBMS at minimum

- Present the data to the user as relations (a presentation in tabular form, i.e. as a collection of tables with each table consisting of a set of rows and columns);
- Provide relational operators to manipulate the data in tabular form.

Popular RDBMS

- Oracle
- MySQL
- Microsoft SQL Server
- PostgreSQL
- IBM DB2
- Microsoft Access
- SQLite
- Amazon Aurora
- MariaDB

Value of Relation Database

- Store large amounts of persistent data
- Concurrency
- Standard Model (Relational Algebra & SQL)

Not so Good side of RDBMS

- Impedance mismatch - The difference between relational model and in-memory data structure.
- Mediocre horizontal scaling support

Transactions

In order for a database management system (DBMS) to operate efficiently and accurately, it must have ACID transactions. ACID is short for Atomicity, Consistency, Isolation, Durability.

Atomicity

Atomicity requires that each transaction be "all or nothing": if one part of the transaction fails, then the entire transaction fails, and the database state is left unchanged. An atomic system must guarantee atomicity in each and every situation, including power failures, errors, and crashes. To the outside world, a committed transaction appears (by its effects on the database) to be indivisible ("atomic"), and an aborted transaction does not happen.

Consistency

The consistency property ensures that any transaction will bring the database from one valid state to another.

Isolation

The isolation property ensures that the concurrent execution of transactions results in a system state that would be obtained if transactions were executed sequentially, i.e., one after the other.

Durability

The durability property ensures that once a transaction has been committed, it will remain so, even in the event of power loss, crashes, or errors. In a relational database, for instance, once a group of SQL statements execute, the results need to be stored permanently (even if the database crashes immediately thereafter).

NoSQL Databases

What is NoSQL?

The term NoSQL may refer to "non SQL", "non relational" or "not only SQL".

A NoSQL database provides a mechanism for storage and retrieval of data which is modeled in means other than the tabular relations used in relational databases.

Why is it called “NoSQL”?

Johan Oskarsson, then a developer at Last.fm, wanted a name for meetup in 2009. Something that would make a good hashtag: short, memorable, and without too many Google hits so that search of the term would quickly find the meetup.

The name attempted to label the emergence of an increasing number of non-relational, distributed data stores.

Characteristics of NoSQL Database

- Schema-less
- Does not use relational model
- Runs well on cluster

NoSQL is more than rows

NoSQL systems store and retrieve data from many formats such as

- Key-Value (DynamoDB)
- Graph
- Column-family (Cassandra)
- Document (MongoDB)

NoSQL is free of Joins

NoSQL systems allow you to extract your data using simple interfaces. It does not support “joins”.

NoSQL databases are Schema Free

A NoSQL system can handle data in various formats. You do not have to tell NoSQL system in advance about the format of data.

Scaling NoSQL

NoSQL systems scale linearly. More processors you add, more performant the system gets.

CAP Theorem

The CAP theorem states that it is impossible for a distributed computer system to simultaneously provide more than two out of three of the following guarantees:

1. **Consistency** - Every read receives the most recent write or an error
2. **Availability** - Every request receives a (non-error) response – without guarantee that it contains the most recent write
3. **Partition tolerance** - The system continues to operate despite an arbitrary number of messages being dropped (or delayed) by the network between nodes

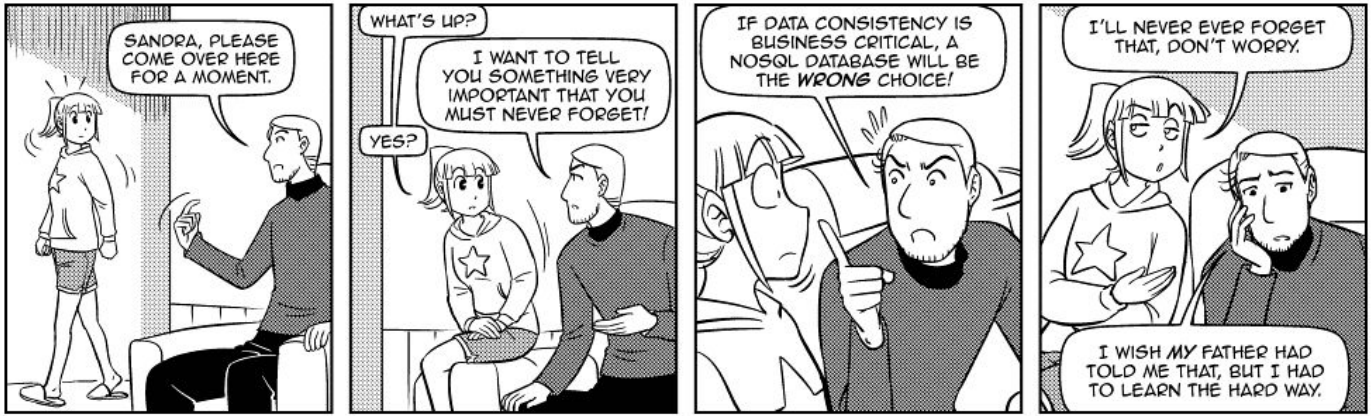
In other words, the CAP Theorem states that in the presence of a network partition, one has to choose between consistency and availability. Note that consistency as defined in the CAP Theorem is quite different from the consistency guaranteed in ACID database transactions.

Transactions in NoSQL

Most NoSQL stores lack true ACID transactions.

NoSQL Databases & Consistency

NoSQL databases offer a concept of "eventual consistency" in which database changes are propagated to all nodes "eventually" (typically within milliseconds) so queries for data might not return updated data immediately or might result in reading data that is not accurate, a problem known as stale reads.



Sandra and Woo by Oliver Knörzer (writer) and Powree (artist) – www.sandraandwoo.com

What is key-value store?

A key-value store is a simple database that when presented with a simple string (the key) returns an arbitrary large BLOB of data (the value).

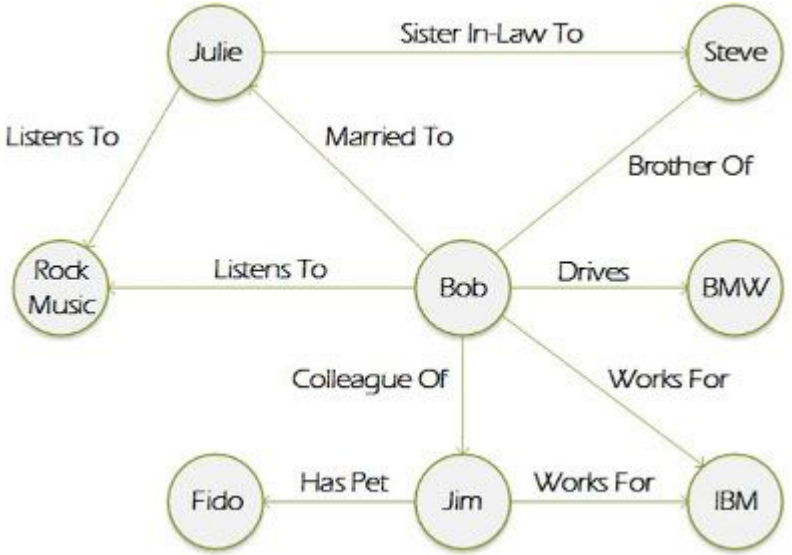
Key-value stores usually have no query language.

In a key-value store, key is of String data type and value can be of any type. One does not have to specify the type for value.

Example: DynamoDB & Redis

Graph Databases

This kind of database is designed for data whose relations are well represented as a graph consisting of elements interconnected with a finite number of relations between them. The type of data could be social relations, public transport links, road maps or network topologies.



Column Family (Bigtable) Stores

Bigtable maps two arbitrary string values (row key and column key) and timestamp (hence three-dimensional mapping) into an associated arbitrary byte array.

Think of spreadsheets. You can use combination of row number and column letter as an address to “look up” the value of any cell.

Examples: Google Bigtable & Apache Cassandra

Document Database

Document Databases differ from other NoSQL databases we have looked at in the sense that they Key may be simple id which is never used or seen but you can get almost any item out of document store by querying any value or content within the document.

The central concept of a document-oriented database is the notion of a document. While each document-oriented database implementation differs on the details of this definition, in general, they all assume documents encapsulate and encode data (or information) in some standard formats or encodings such as JSON, BSON, XML, etc.

Example Documents

```
{
    "FirstName": "Bob",
    "Address": "5 Oak St.",
    "Hobby": "sailing"
}
```

```
<contact>
  <firstname>Bob</firstname>
  <lastname>Smith</lastname>
  <phone type="Cell">(123) 555-0178</phone>
  <phone type="Work">(890) 555-0133</phone>
  <address>
    <type>Home</type>
    <street1>123 Back St.</street1>
    <city>Boys</city>
    <state>AR</state>
    <zip>32225</zip>
    <country>US</country>
  </address>
</contact>
```

Normalization & Denormalization

Database normalization is a technique for designing relational database schemas that ensures that the data is optimal for ad-hoc querying and that modifications such as deletion or insertion of data does not lead to data inconsistency.

Database denormalization is the process of optimizing your database for reads by creating redundant data. A consequence of denormalization is that insertions or deletions could cause data inconsistency if not uniformly applied to all redundant copies of the data within the database.

Polyglot Persistence

- Polyglot Persistence, like polyglot programming, is all about choosing the right persistence option for the task at hand.
- NoSQL has result in widespread use of Polyglot Persistence.

Consistency

Concurrency Control Mechanisms

- **Optimistic** - Delay the checking of whether a transaction meets the isolation and other integrity rules (e.g., serializability and recoverability) until its end, without blocking any of its (read, write) operations ("...and be optimistic about the rules being met..."), and then abort a transaction to prevent the violation, if the desired rules are to be violated upon its commit. An aborted transaction is immediately restarted and re-executed, which incurs an obvious overhead (versus executing it to the end only once). If not too many transactions are aborted, then being optimistic is usually a good strategy.
- **Pessimistic** - Block an operation of a transaction, if it may cause violation of the rules, until the possibility of violation disappears. Blocking operations is typically involved with performance reduction.

Write-Write Conflict

Two users updating same data item with different values.

Optimistic Approaches:

- Conditional Update: See if value has changed since last read before update
- Process both updates and record that there is a conflict

Read-Write Conflict

Alice and Bob are using a website to book tickets for a specific show. Only one ticket is left for the specific show. Alice signs on first to see that only one ticket is left, and finds it expensive. Alice takes time to decide. Bob signs on and also finds one ticket left, and orders it instantly. Bob purchases and logs off. Alice decides to buy a ticket, to find there are no tickets. This is a typical read-write conflict situation.

Version Stamps

A field that changes every time the underlying data in the record changes. When you read the data you keep note of the version stamp, so that when you write data you can check to see if the version has changed.

Distribution Models

Single Server

- One machine handles all reads and writes to the database.
- Simplest distribution model as there is no cluster.
- Suitable for development setup.

Sharding

- Store different parts of data on different servers.
- Spread the load across the cluster as different users ideally will be accessing data from different nodes.
- Sharding can improve both read and write performance.

Note: If the data is sharded using wrong key, you can end up with highly unbalanced cluster and all traffic will end up going to same server nodes.

Master Slave Replication

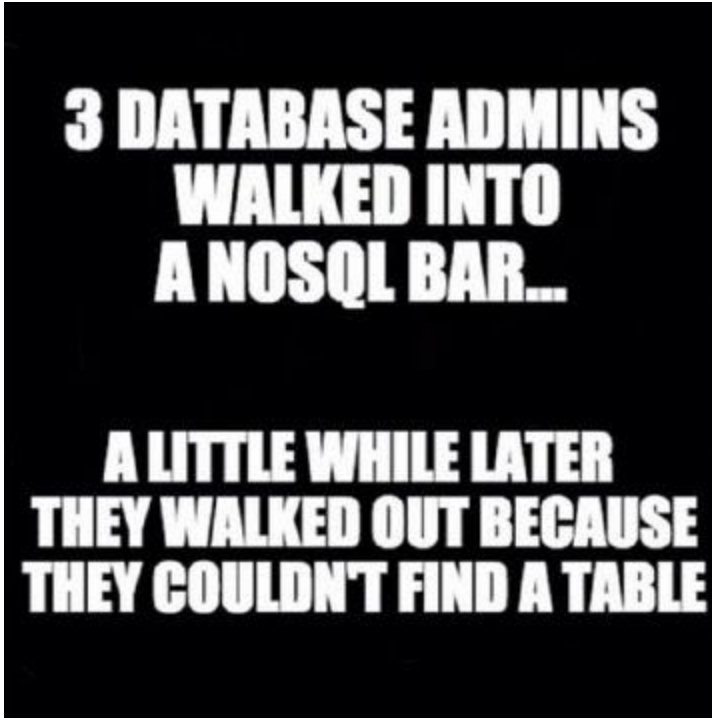
- In this distribution model you replicate data across multiple nodes. One node is designated as “primary” or “master” and others are designated as “secondary” or “slave”. This “master” node is authoritative source of data and is usually responsible for processing any updates to that data.
- Master node can be a bottleneck as all writes must go to it.
- If master node dies, slave node can still answer read queries while another node is elected master either automatically or manually.
- Master node can be single point of failure.

Peer-to-Peer Replication

- There is no master node in peer-to-peer replication
- All replicas have equal weight, they can all accept writes, and loss of any of them doesn’t prevent access to data store.

Combining Sharding & Replication

Peer-to-peer replication combined with sharding is common strategy used by column-family databases.


3 DATABASE ADMINS
WALKED INTO
A NOSQL BAR...
A LITTLE WHILE LATER
THEY WALKED OUT BECAUSE
THEY COULDN'T FIND A TABLE