NoSQL : Databases at web scale

Relational DBs solve real problems

Relational DBs are a very good solution to these problems:

- □ How to support concurrent access to the data by thousands of users?
- □ How to keep data consistent, and hide temporary inconsistency?
- □ How to avoid redundancy?
- □ How to let multiple applications share data?

New requirements

Trends since the ‘90s have led to new requirements:

- □ Need to support millions or 10s of millions of users
- □ Need to support rapid feature development and schema changes
- □ Need to access data from OO languages
- □ Need for very high availability

Can relational DBs scale?

- vertical scaling (with relational DBs)
  - get bigger, faster machines, special hardware
  - very expensive; may not provide enough scaling
- horizontal scaling
  - distribute data over many cheap machines
  - sharding

Schema changes

- Schema changes are painful with relational DBs
- “Data migration”
- Data migration might involve:
  - moving data off database
  - modifying schema
  - modifying data
  - putting data back in database
- Problem is worse when database shared by many applications

Availability and Consistency

Many applications want “availability, at the cost of consistency”

Examples:

- Amazon cart

When databases get distributed, and failures can occur, you can’t have both availability and consistency.

Cost factor

Commercial databases are expensive.

- proof: Larry Ellison (Oracle) is worth $52 billion

The main NoSQL databases are open source.

What is the alternative?

- commodity hardware
- large clusters
- open source
- no schema
- simple API
- eventual consistency

NoSQL

NoSQL databases

There are different kinds of NoSQL databases.

Main types:

type	examples
key-value	Redis, DynamoDB
document	MongoDB, Couchbase
column-family	Cassandra, HBase
graph	Neo4J, OrientDB

Key-value

- super simple API
- if you want to store something, give it a name and store it
  - database does not "know" the structure of the data
- core data structure is the associative array
- core technology is distributed hash table

In an associative array:

- different types of values can be stored
- any type of value can be used as an index

Document

- Similar to key-value, but now values have standard structure
- Typically values are XML or JSON documents
- Closer to relational DB; concepts like “collections”
- Supports a query language
- No pre-defined schema

Column-family

- Another variant of key-value store
- Data actually stored in columns, not rows
  - or “families of columns”
- Reads and writes are done on columns, not rows
- Rows can have different number of columns
- Search through a column becomes very fast; high performance for many queries

Graph

- Data is modeled as a graph
- Key-value store extended to allow relationships between the values
- Data stored in a key-value store
- Supports efficient traversal between values

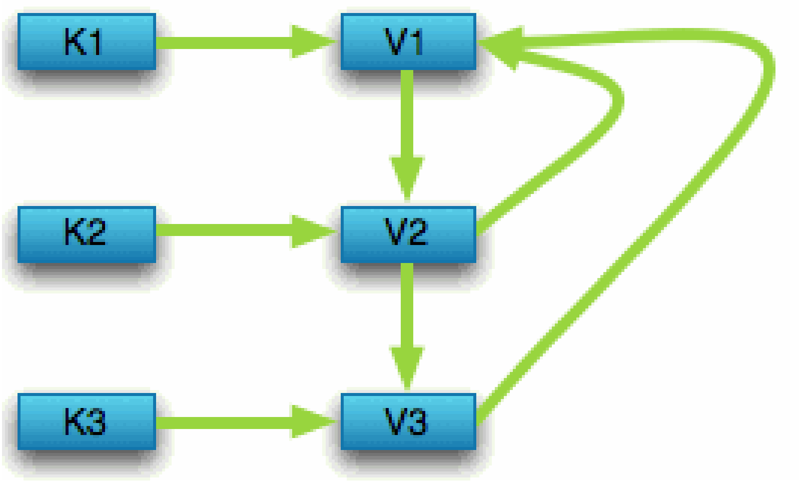

figure from: neo4j.com/developer/graph-db-vs-nosql/

When to use NoSQL

Factors in favor of NoSQL:

- data is so big that a cluster is required
- data is non-uniform
- data is in the form of aggregates
- availability is important

Factors in favor of relational DBs:

- consistency is important
- a large and stable set of tools are needed
- many applications will use the data
- security is important

Summary

NoSQL arose because of new requirements:

- millions of users
- high availability
- rapid app development

Main types of NoSQL databases:

- key-value
- document
- column family
- graph