Introduction to NoSQL

By Kedar Gore

Agenda

What is NoSQL

Why NoSQL

SQL: vs. NoSQL

Types of NoSQL

Where is NoSQL Used

Presentation by: Kedar Gore

Email: gkedar@gmail.com

1

What is NoSQL

What is NoSQL?

- NoSQL is a set of database technologies designed to store non-relational data at large or very large scale.
- Stands for Not Only SQL
- Does not use SQL as querying language
- Distributed, fault-tolerant architecture
- Flexible Schema / No fixed table schema
- No Joins
- No multi-document transactions
- It's not a replacement for a RDBMS but compliments it.
- It is open-source
- NoSQL systems leverage low-cost commodity processors that have separate ram and disk.
- Supports linear scalability - add more processors you get consistent increase in performance.

2

Why NoSQL

Why NoSQL?

- We are storing more data now than we ever have before.
- Connection between our data are growing all the time.
- We do not make things knowing the structure from day 1.
- Server architecture is at a stage where we can take advantage of it.
- Schemaless data representation
- Development time
- Speed
- Manageability and administration
- Lower cost (than competitive solution at that scale)
- High availability

3

SQL vs. NoSQL

SQL vs. NoSQL

	SQL Databases	*NoSQL Databases*
Types & History	All types support SQL standard. Developed in 1970.	Multiple types exists, such as document stores, key value stores, column databases, etc. Developed in 2000s.
Data Storage Model	Data is stored in rows and columns in a table, where each column is of a specific type. The tables generally are created on principles of normalization. Joins are used to retrieve data from multiple tables.	The data model depends on the databasetype. Say data is stored as a key-value pair for key-value stores. In document based databases, the data is stored as documents. The data model is flexible, in contrast to the rigid table model of the RDBMS.
Schemas	Fixed structure and schema, so any change to schema involves altering the database.	Dynamic schema, new data types, or structures can be accommodated by expanding or altering the current schema. New fields can be added dynamically.
Scalability	Scale up approach is used; this means as the load increases, bigger, expensive servers are bought to accommodate the data.	Scale out approach is used; this means distributing the data load across inexpensive commodity servers
Supports Transactions	Supports ACID and transactions	Supports partitioning and availability,and compromises on transactions. Transactions exist at certain level, such as the database level or document level.
Support	High level of enterprise support is provided.	Open source model. Support through third parties or companies building the open source products.

SQL vs. NoSQL

	SQL Databases	*NoSQL Databases*
Maturity	Have been around for a long time.	Some of them are mature; others are evolving.
Consistency	Strong consistency.	Dependent on the product. Few chose to provide strong consistency whereas few provide eventual consistency.
Querying Capabilities	Available through easy-to-use GUI interfaces.	Querying may require programming expertise and knowledge. Rather than an UI, focus is on functionality and programming interfaces.
Expertise	Large community of developers who have been leveraging the SQL language and RDBMS concepts to architect and develop applications	Small community of developers working on these open source tools.
Example	Oracle, SQLServer, MySQL, PostgresQL	Memcached, Cassandra, MongoDB, Neo4j

ACID vs. BASE

ACID ***(Atomicity, Consistency,Isolation,Durability)***	***BASE*** ***(Basic availability, Soft-state,Eventual consistency)***
Atomicity—Systems that claim they have atomic transactions must consider all failure modes: disk crashes, network failures, hardware failures, or simple software errors. Testing atomic transactions even on a single CPU is difficult.	Basic availability allows systems to be temporarily inconsistent so that transactions are manageable. In BASE systems, the information and service capability are "basically available."
Consistency—It's the responsibility of the database to block all reports during atomic operations.	Soft-state recognizes that some inaccuracy is temporarily allowed and data may change while being used to reduce the amount of consumed resources.
Isolation—Isolation refers to the concept that each part of a transaction occurs without knowledge of any other transaction.	Eventual consistency means eventually, when all service logic is executed, the system is left in a consistent state.
Durability—Durability refers to the fact that once all aspects of a transaction are complete, it's permanent.	
• Get transaction details right • Block any reports while you are working • Be pessimistic anything might go wrong • Detailed testing and failure mode analysis • Lots of locks and unlocks	• Never blocks a write • Focus on throughput, not consistency • Be optimistic: if one service fails it eventually get caught up • Some reports may be inconsistent for a while but do not worry • Keep things simple and void locks

4

Type of NoSQL

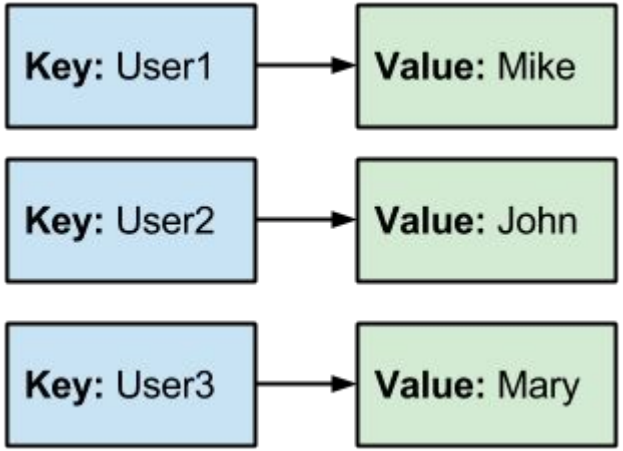
Types of NoSQL

Key-Value Store	*Column-Family Store*	*Graph Store*	*Document Store*
A simple data storage system that uses a key to access a value	A sparse matrix system that uses a row and a column as keys	For relationship intensive problems	Hierarchical data structures directly in the database
• Image Stores • Key-based filesystems • Object cache • System designed to scale	• Web crawler results • Big data problems that can relax consistency rules	• Social networks • Fraud detection • Relationship-heavy data	• High-variability data • Document search • Integration hubs • Web content mgt • Publishing
• Memcache • Redis	• Apache HBase • Apache Cassandra	• Neo4J • AllegroGraph	• MongoDB • CouchDB

Key-Value Store

- Data is stored in key-value pairs.
- Designed to handle massive data and heavy load.
- Inspired by Amazon's Dynamo and Distributed Hashtables.
- Each key can have different types of data.
- Basic Operations
  - get(key): extract the value given a key
  - Put(key,value): create or update the value given its key
  - delete(key): remove the key and its associated value.
- Multiple Types
  - In Memory(Memcache), OnDisk(Redis,SimpleDB), Eventually Consistent(Dynamo, VoldeMort)
- Advantages: efficiency, scalability, fault-tolerance

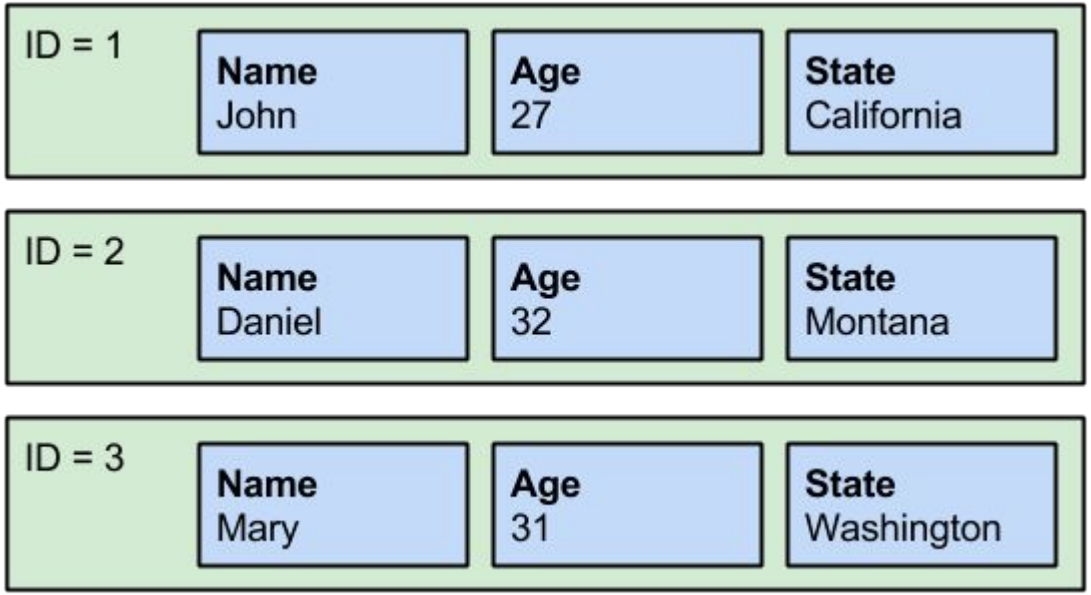
Key-Value Store Example



Column Family Store

- Data is stored in columnar format.
- Each storage block contains data from only one column.
- Inspired by Google Bigtable.
- Allow key-value pairs to be stored in a massively parallel system
  - Data model: families of attributes defined in schema, new attribute can be added online
  - Storing principle: big hashed distributed tables
- Example
  - Google Bigtable, Apache Hbase, Apache Cassandra
- Advantages: Partitioning (Horizontal/Vertical), High availability, Completely transparent to application

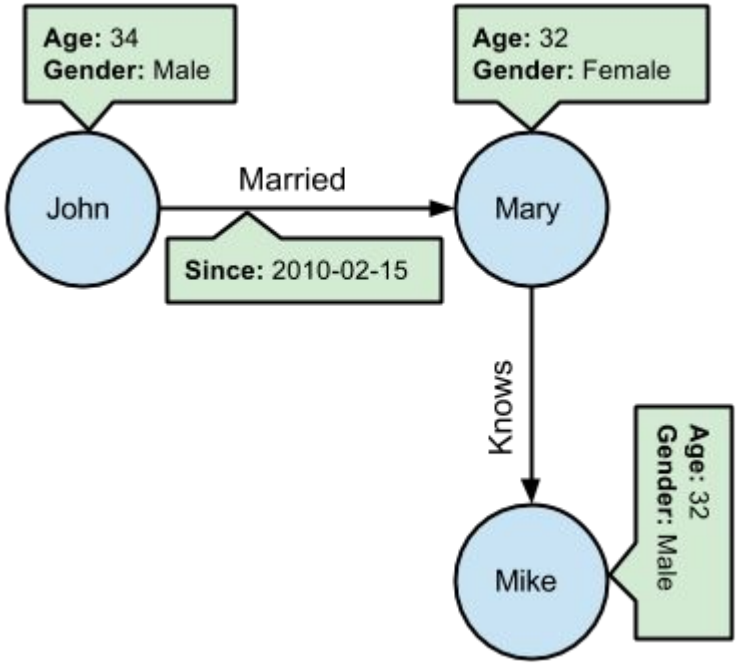
Column-family Store Example



Graph Store

- Focuses on modelling of data and associated connections.
- Are based on concept of Vertex and edges
- Relational DB can model graphs, but an edge does not require a join which is expensive.
- Inspired by Mathematical graph theory.
- Data model: nodes and edges
  - Nodes may have properties (including ID)
  - Edges may have labels or roles
- Example
  - Neo4j, AllegroGraph, FlockDB
- Advantages: Scales to the complexity of data

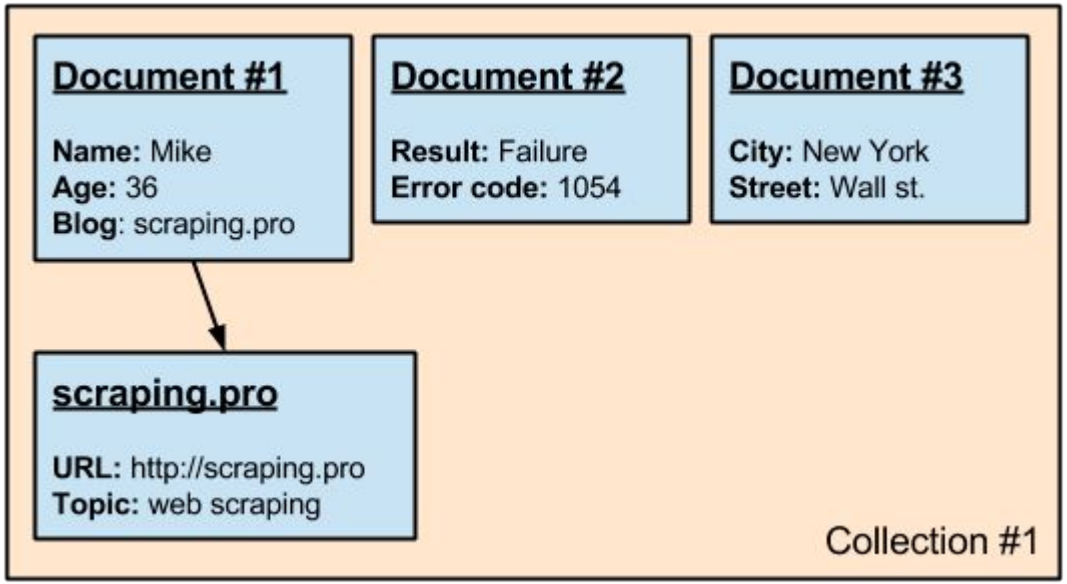
Graph Store - Example



Document Store

- Data stored as a whole document.
- Document stored in JSON or BSON formats.
- Maps well to an object oriented programming model.
- Inspired by Lotus Notes.
- Data model: Similar to key-value store but with a major difference, value is a document
- Query Model: Javascripts or custom
- Indexes are done via B-trees.
- Example
    - MongoDB, CouchDB, RethinkDB
- Advantages: Flexible schema, Easier Application logic,

Document Store Example



Feature Comparison

Feature	Column-Oriented	Document Store	Key-Value Store	Graph
Table-like schema support (columns)	Yes	No	No	Yes
Complete update/fetch	Yes	Yes	Yes	Yes
Partial update/fetch	Yes	Yes	Yes	No
Query/filter on value	Yes	Yes	No	Yes
Aggregate across rows	Yes	No	No	No
Relationship between entities	No	No	No	Yes
Cross-entity view support	No	Yes	No	No
Batch fetch	Yes	Yes	Yes	Yes
Batch update	Yes	Yes	Yes	No

5 Where is NoSQL used

Where is NoSQL used?

Key-Value Store	*Column-Family Store*	*Graph Store*	*Document Store*
• Image Stores • Object Stores • File Storage • Rest API calls • Shopping Cart	• Web crawler results • Structured Data • Map data • Analytics data	• Social networks • Fraud detection • Relationship-heavy data	• Real time operational intelligence • Product Data management • User Data Management • Web content management • High Volume Data Feeds

DB-Engines Ranking Top 10

310 systems in ranking, November 2016

Rank Nov 2016	Rank Oct 2016	Rank Nov 2015	DBMS	Database Model	Score Nov 2016	Score Oct 2016	Score Nov 2015
1.	1.	1.	Oracle	Relational DBMS	1413.01	-4.09	-67.94
2.	2.	2.	MySQL	Relational DBMS	1373.56	+10.91	+86.71
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1213.80	-0.38	+91.48
4.	↑ 5.	↑ 5.	PostgreSQL	Relational DBMS	325.82	+7.12	+40.13
5.	↓ 4.	↓ 4.	MongoDB	Document store	325.48	+6.67	+20.87
6.	6.	6.	DB2	Relational DBMS	181.46	+0.90	-21.07
7.	7.	↑ 8.	Cassandra	Wide column store	133.97	-1.09	+1.05
8.	8.	↓ 7.	Microsoft Access	Relational DBMS	125.97	+1.30	-14.99
9.	9.	↑ 10.	Redis	Key-value store	115.54	+6.00	+13.13
10.	10.	↓ 9.	SQLite	Relational DBMS	112.00	+3.43	+8.55

Thanks!

Any questions ?

You can find me at

- @geekduo
- gkedar@gmail.com

Credits

Special thanks to all the people who made and released these awesome resources for free:

- Presentation template by SlidesCarnival
- Slideshare Community
- http://db-engines.com/en/