Introduction to NoSQL

Mrihan Mohamed Ahmed

Teaching Assistant – ITI

mrihan.m.ahmed@gmail.com

What is NoSQL?

- Next Generation Database Management Systems mostly addressing some of the points: being **non-relational, distributed, open-source** and **horizontally scalable**.
- the community now translates it mostly with “**not only sql**”.
- The movement began early 2009 and is growing rapidly.
- Often more characteristics apply such as: **schema-free, easy replication support, simple API, eventually consistent** / **BASE** (not ACID), a **huge amount of data** and more.

Why to Develop and use NoSQL?

- **Avoidance of Unneeded Complexity**: RDB rich feature set and the ACID properties might be more than necessary.
- **High Throughput**.
- **Horizontal Scalability:** the volume of data is getting so huge that people are looking at other technologies.
- **Avoidance of Expensive Object-Relational Mapping**: all you really need is a key, value that supports some level of query functionality and has decent persistence semantics.

Why to Develop and use NoSQL?

- **The Current “One size fit’s it all” Databases Thinking Was and Is Wrong**.
- **Requirements of Cloud Computing**: High until almost ultimate scalability especially in the horizontal direction and Low administration overhead.
- **Yesterday’s vs. Today’s Needs**.

NoSQL vs. RDBMS

- RDBMS assumes a well-defined structure of data and assumes that the data is largely uniform.
- It needs the schema of your application and its properties (columns, types, etc.) to be defined up-front before building the application. This does not match well with the agile development approaches for highly dynamic applications.
- As the data starts to grow larger, you have to scale your database vertically, i.e. adding more capacity to the existing servers.

NoSQL vs. RDBMS

- **Schema Less:** NoSQL databases being schema-less do not define any strict data structure.
- **Dynamic and Agile:** NoSQL databases have good tendency to grow dynamically with changing requirements. It can handle structured, semi-structured and unstructured data.
- **Scales Horizontally:** In contrast to SQL databases which scale vertically, NoSQL scales horizontally by adding more servers and using concepts of sharding and replication.
- **Better Performance:** All the NoSQL databases claim to deliver better and faster performance as compared to traditional RDBMS implementations.

Limitations of NoSQL

- Since NoSQL is an entire set of databases (and not a single database), the limitations differ from database to database.
- Some of these databases do not support ACID transactions while some of them might be lacking in reliability.
- Each one of them has their own strengths due to which they are well suited for specific requirements.
- No standardization.
- Limited query capabilities (so far).

ACID vs. BASE

The key ACID guarantee is that it provides a safe environment in which to operate on your data. The ACID acronym stands for:

- ***Atomic***: All operations in a transaction succeed or every operation is rolled back.
- ***Consistent***: On the completion of a transaction, the database is structurally sound.
- ***Isolated***: Transactions do not contend with one another. Contentious access to data is moderated by the database so that transactions appear to run sequentially.
- ***Durable***: The results of applying a transaction are permanent, even in the presence of failures.

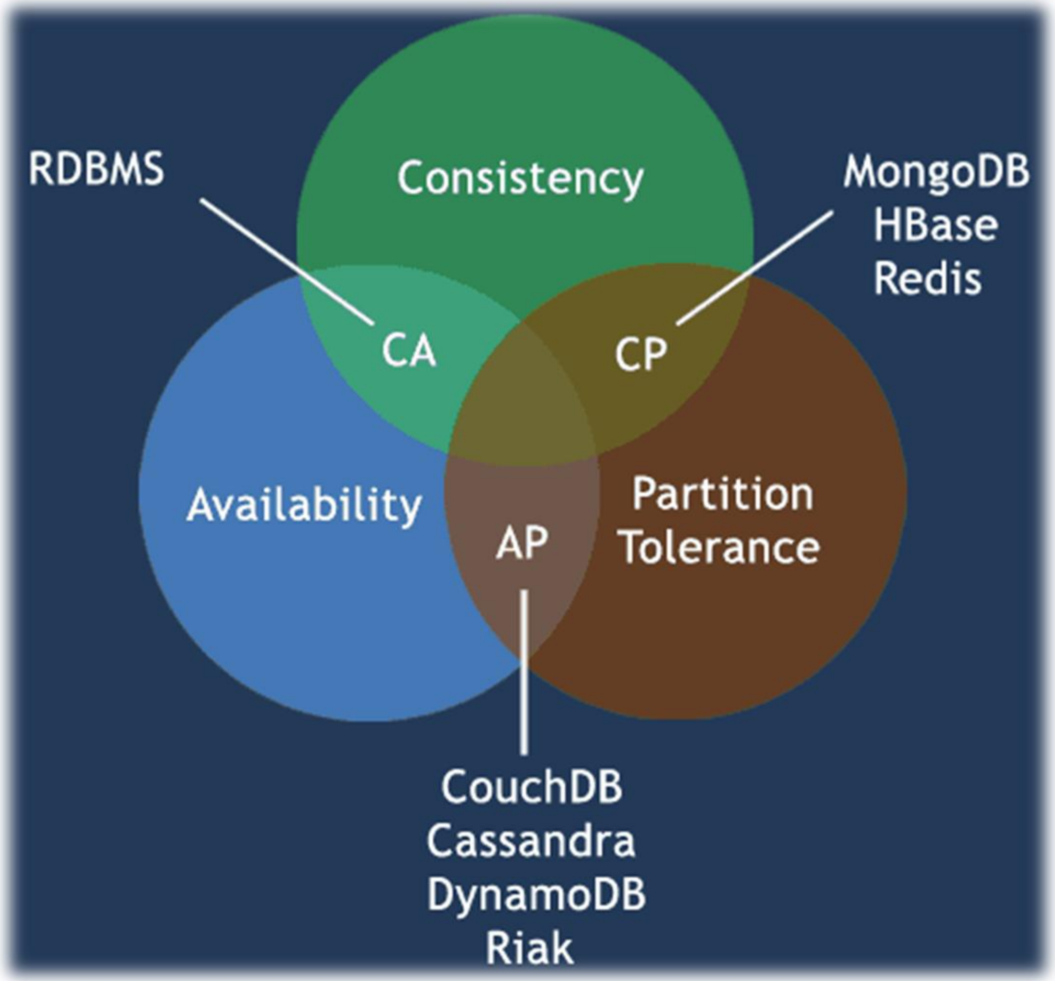
ACID vs. BASE

BASE acronym breaks down:

- ***Basic Availability***: The database appears to work most of the time.
- ***Soft-state***: Stores don't have to be write-consistent, nor do different replicas have to be mutually consistent all the time.
- ***Eventual consistency***: Stores exhibit consistency at some later point.

A BASE data store values availability, but it doesn't offer guaranteed consistency of replicated data at write time. Overall, the BASE consistency model provides a less strict assurance than ACID: data will be consistent in the future.

CAP Theorem


RDBMS
Consistency
MongoDB
HBase
Redis
CA
CP
Availability
AP
Partition
Tolerance
CouchDB
Cassandra
DynamoDB
Riak

NoSQL Types

Key – Value Store	redis	riak
Wide Column Store	APACHE HBASE	cassandra
Document Store	mongoDB®	Apache CouchDB relax
Graph Store	neo4j	INFINITEGRAPH®

Categorization Based on Customer Needs

- **Features-First**:
  - This class of databases provides a (large) number of high level features that make the programmer’s job easier. On the downside, they are difficult to scale.
  - Oracle, Microsoft SQL Server, MySQL, PostgreSQL, Amazon RDS12.

- **Scale-First:**
  - This sort of databases has to scale from the start. On the downside, they lack particular features and put responsibility back to the programmer.
  - Project Voldemort, Ringo, Amazon SimpleDB, Kai, Dynomite, Yahoo PNUTS, ThruDB, Hypertable, CouchDB, Cassandra, MemcacheDB.

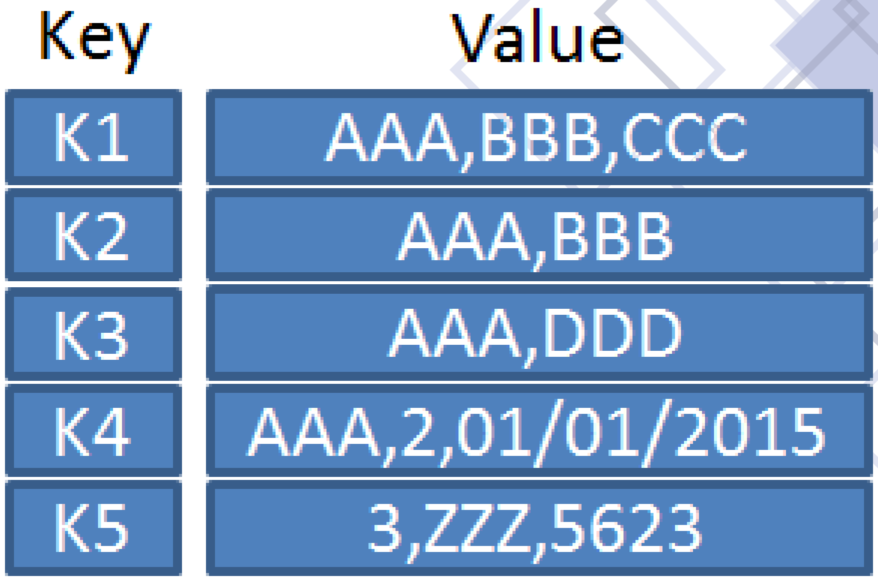
Categorization Based on Customer Needs

- **Simple Structure Storage:**
  - This class subsumes key/value-stores with an emphasis on storing and retrieving sets of arbitrary structure. The downside is that they generally don’t have the features or the scalability of other systems.
  - file systems, Cassandra, BerkelyDB, Amazon SimpleDB.

- **Purpose-Optimized Storage:**
  - These are databases which are designed and built to be good at one thing, e.g. data warehousing or stream processing.
  - StreamBase, Vertica, VoltDB, Aster Data, Netezza, Greenplum.

Key-Value Type

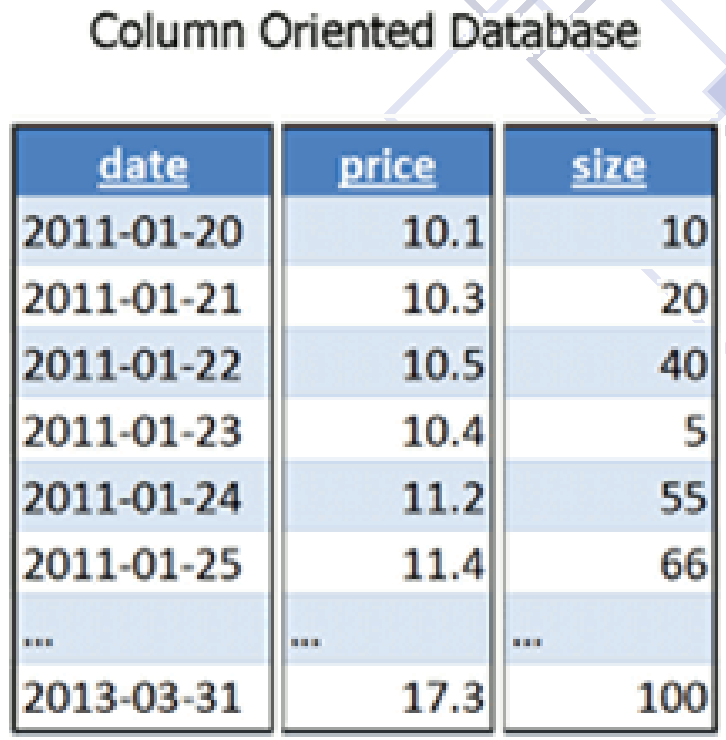
- The key of a key/value pair is a unique value in the set and can be easily looked up to access the data.
- The main idea here is using a hash table where there is a unique key and a pointer to a particular item of data.
- Examples: Tokyo Cabinet/Tyrant, Redis, Voldemort, Oracle BDB, Amazon SimpleDB, Riak.



Key	Value
K1	AAA,BBB,CCC
K2	AAA,BBB
K3	AAA,DDD
K4	AAA,2,01/01/2015
K5	3,ZZZ,5623

Column-Based Type

- The column-oriented storage allows data to be stored effectively.
- Were created to store and process very large amounts of data distributed over many machines.
- The columns are arranged by column family.
- Examples: Cassandra, HBase.



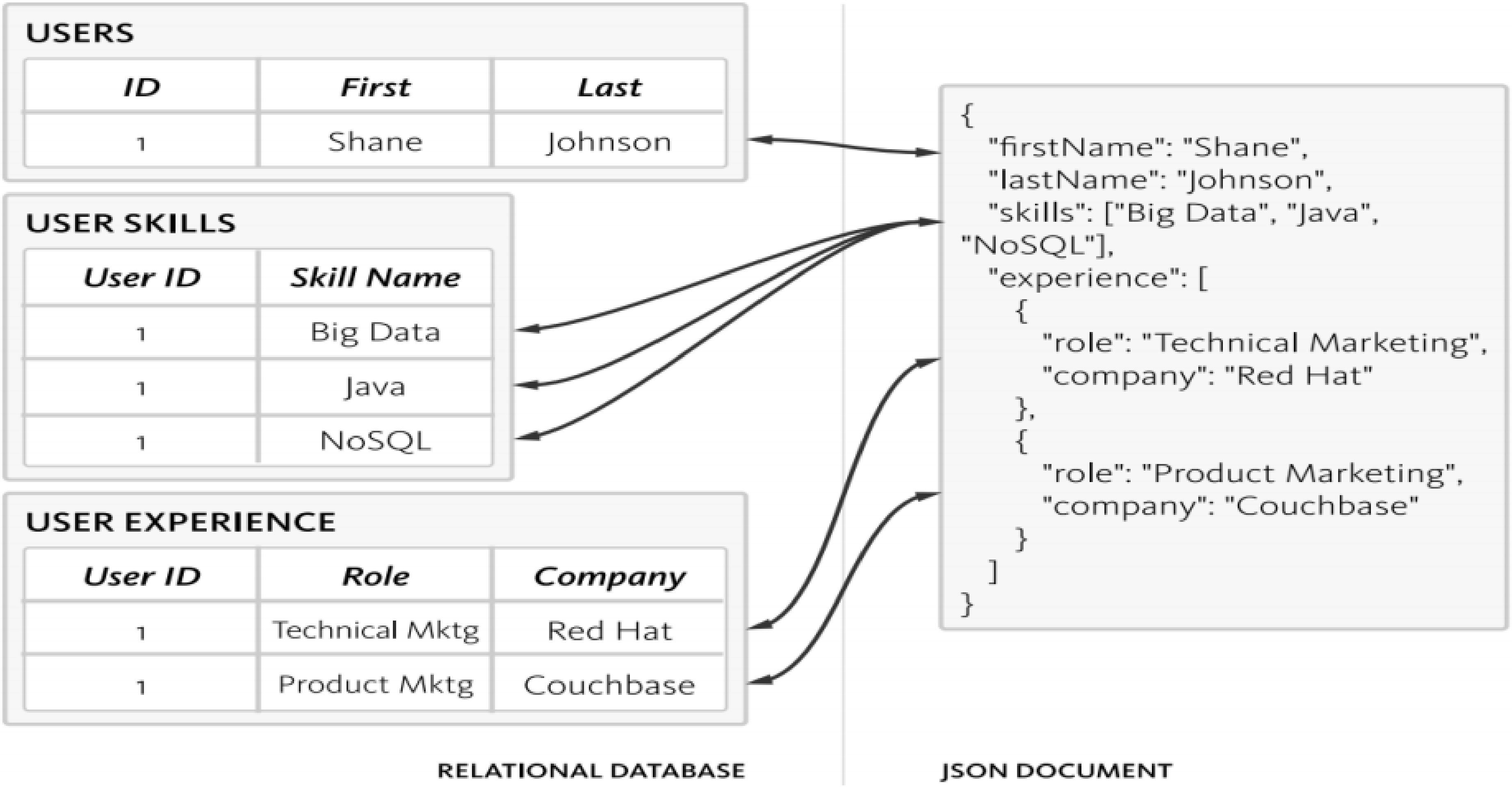
Column Oriented Database

date	price	size
2011-01-20	10.1	10
2011-01-21	10.3	20
2011-01-22	10.5	40
2011-01-23	10.4	5
2011-01-24	11.2	55
2011-01-25	11.4	66
...	...	...
2013-03-31	17.3	100

Document Oriented Type

- These are similar to key-value stores.
- The model is basically collections of other key-value collections.
- The semi-structured documents are stored in formats like JSON.
- Document databases are essentially the next level of Key/value, allowing nested values associated with each key.
- Document databases support querying more efficiently.
- Examples: MongoDb.
- In fact, **MongoDB** has become one of the most popular NoSQL databases.

Document Oriented Type



USERS

ID	First	Last
1	Shane	Johnson

USER SKILLS

User ID	Skill Name
1	Big Data
1	Java
1	NoSQL

USER EXPERIENCE

User ID	Role	Company
1	Technical Mktg	Red Hat
1	Product Mktg	Couchbase

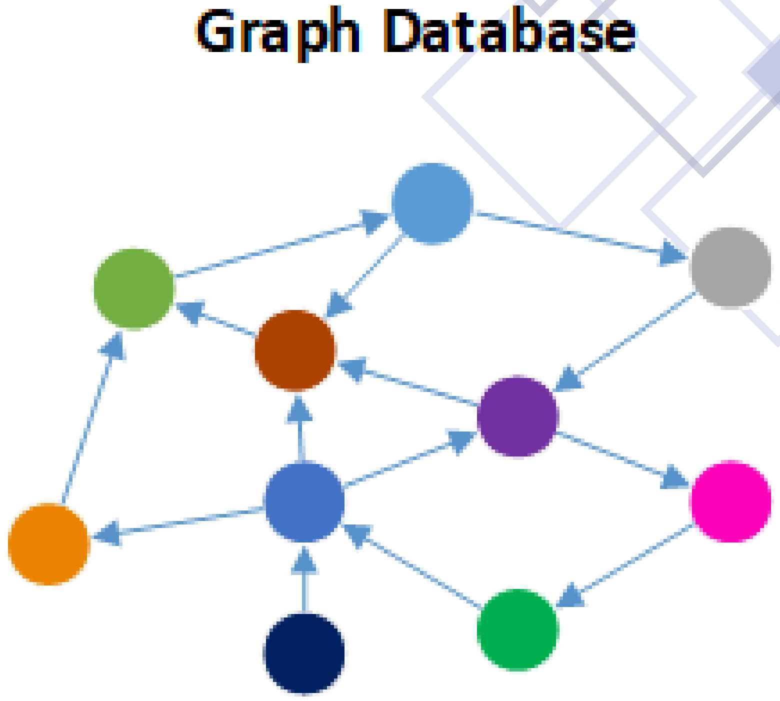
```
{
  "firstName": "Shane",
  "lastName": "Johnson",
  "skills": ["Big Data", "Java",
"NoSQL"],
  "experience": [
    {
      "role": "Technical Marketing",
      "company": "Red Hat"
    },
    {
      "role": "Product Marketing",
      "company": "Couchbase"
    }
  ]
}
```

RELATIONAL DATABASE

JSON DOCUMENT

Graph-Based Type

- Instead of tables of rows and columns and the rigid structure of SQL, a flexible graph model is used.
- A graph database uses graph structures with nodes, edges, and properties to represent and store data.
- This means that every element contains a direct pointer to its adjacent element and no index lookups are necessary.
- Examples: Neo4J, InfoGrid, Infinite Graph.



Thanks