PRESENTATION NoSQL

What is NoSQL

- Stands for Not Only SQL**. Term was redefined by Eric Evans after Carlo Strozzi.**
- Class of non-relational data storage systems.
- Do not require a fixed table schema nor do they use the concept of joins.
- Relaxation for one or more of the ACID properties (*Atomicity, Consistency, Isolation, Durability) using CAP theorem.*

Need of NoSQL

- Explosion of social media sites (Facebook, Twitter, Google etc.) with large data needs. (*Shardingis a problem)*
- Rise of cloud-based solutions such as Amazon S3 (simple storage solution).
- Just as moving to dynamically-typed languages (Ruby/Groovy), a shift to dynamically-typed data with frequent schema changes.
- Expansion of Open-source community.
- NoSQL solution is more acceptable to a client now than a year ago.

Characteristics of NoSQL

- It’s more than rows in tables—NoSQL systems store and retrieve data from many formats: key-value stores, graph databases, column-family (Bigtable) stores, document stores, and even rows in tables.
- It’s free of joins—NoSQL systems allow you to extract your data using simple interfaces without joins.
- It’s schema-free—NoSQL systems allow you to drag-and-drop your data into a folder and then query it without creating an entity-relational model.
- It works on many processors—NoSQL systems allow you to store your database on multiple processors and maintain high-speed performance.
- It uses shared-nothing commodity computers—Most (but not all) NoSQL systems leverage low-cost commodity processors that have separate RAM and disk.
- It supports linear scalability—When you add more processors, you get a consistent increase in performance.
- It’s innovative—NoSQL offers options to a single way of storing, retrieving, and manipulating data. NoSQL supporters (also known as NoSQLers) have an inclusive
- attitude about NoSQL and recognize SQL solutions as viable options. To the NoSQL community, NoSQL means “Not only SQL.”

NoSQL Types

- NoSQL data base are classified into four types:
  - **Key Value pair based**
  - **Column based**
  - **Document based**
  - **Graph based**

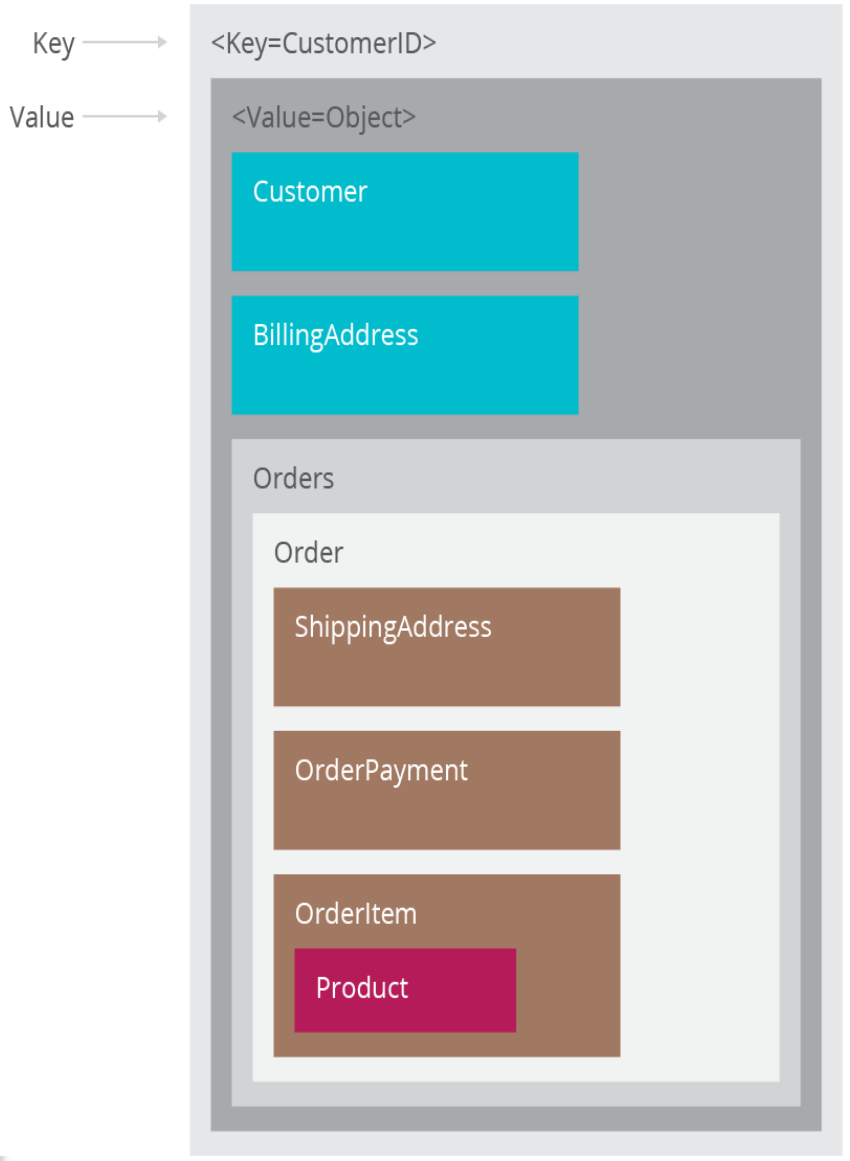
Key Value Pair Based

- Designed for processing dictionary. Dictionaries contain a collection of records having fields containing data.
- Records are stored and retrieved using a key that uniquely identifies the record, and is used to quickly find the data within the database.

Example: CouchDB, Oracle NoSQL Database, Riak etc.

We use it for storing session information, user profiles, preferences, shopping cart data.

We would avoid it when we need to query data having relationships between entities.



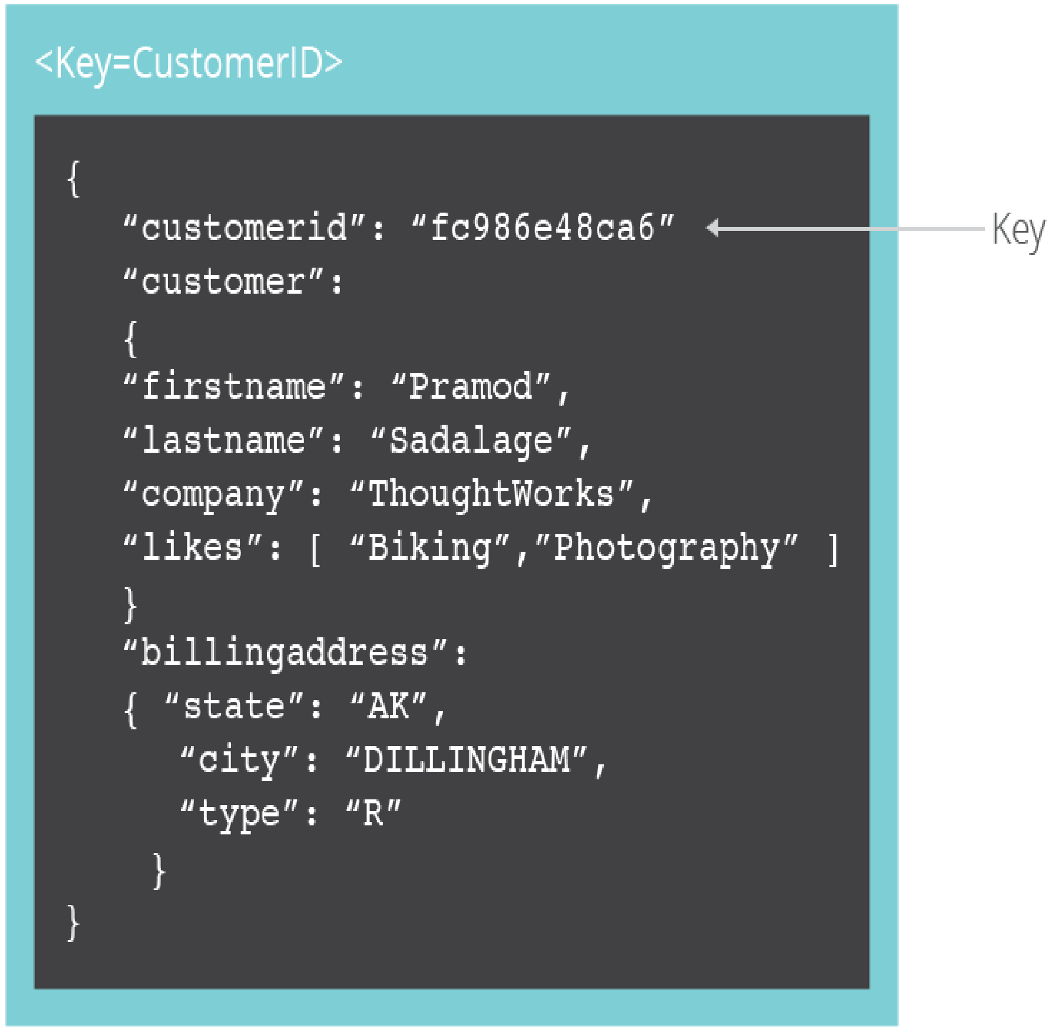
Document Based

The data base store sand retrieves documents. It stores documents in the value part of the key-value store.

Self-describing, hierarchical tree data structures consisting of maps, collections, and scalar values.

Example: LotusNotes, MongoDB, CouchDB, OrientDB, RavenDB.

- We use it for content management systems, blogging platforms, we banalytics, real-time analytics,e-commerce applications.
- We would avoid it for systems that need complex transactions spanning multiple operations orqueries against varying aggre gate structures



```
{
   "customerid": "fc986e48ca6"
   "customer":
   {
   "firstname": "Pramod",
   "lastname": "Sadalage",
   "company": "ThoughtWorks",
   "likes": [ "Biking","Photography" ]
   }
   "billingaddress":
   { "state": "AK",
     "city": "DILLINGHAM",
     "type": "R"
    }
}
```

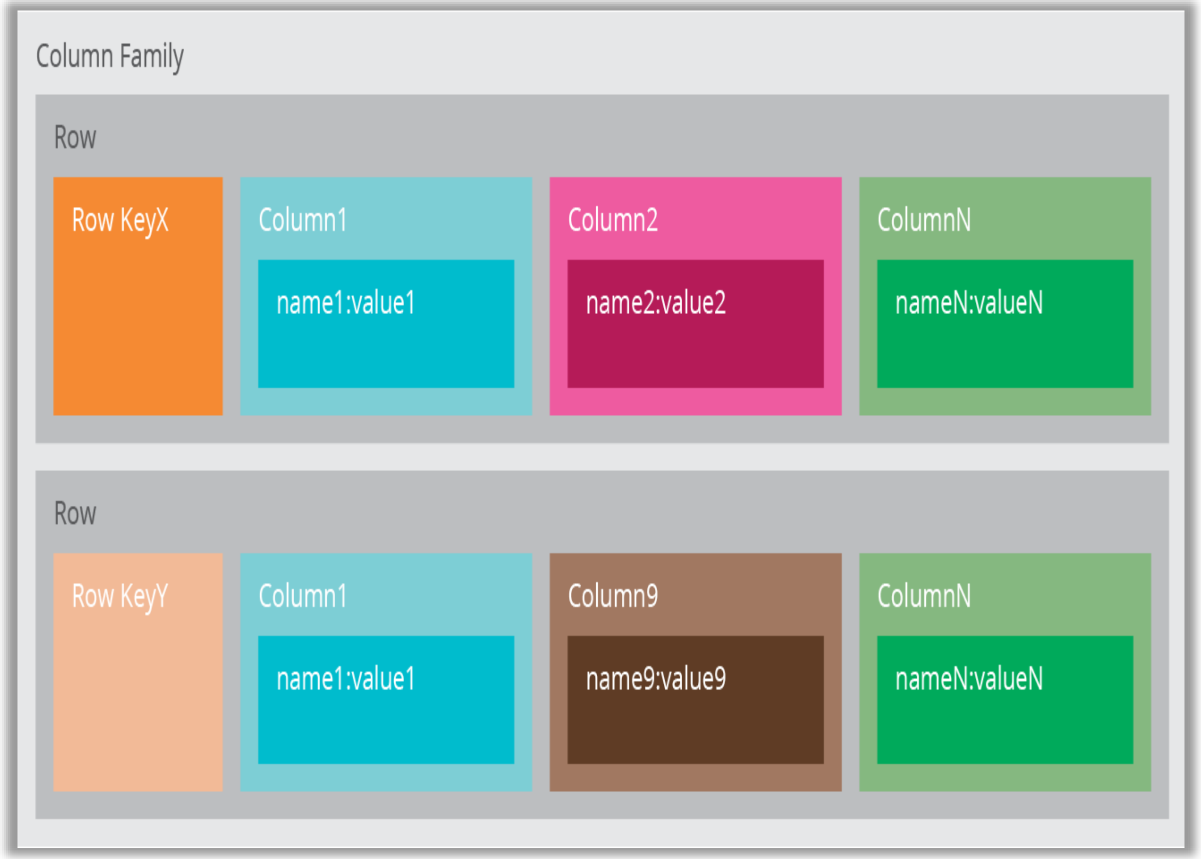
Column based

It store data as Column families containing rows that have many columns associated with arow key. Each row can have different columns.

Column families are groups offre lated data that is accessed together.

Example: Cassandra, HBase, Hypertable, and Amazon DynamoDB.

- We use it for content management systems, blogging platforms, log aggregation.
- We would avoid it for systems that are inearly development, changing query patterns.

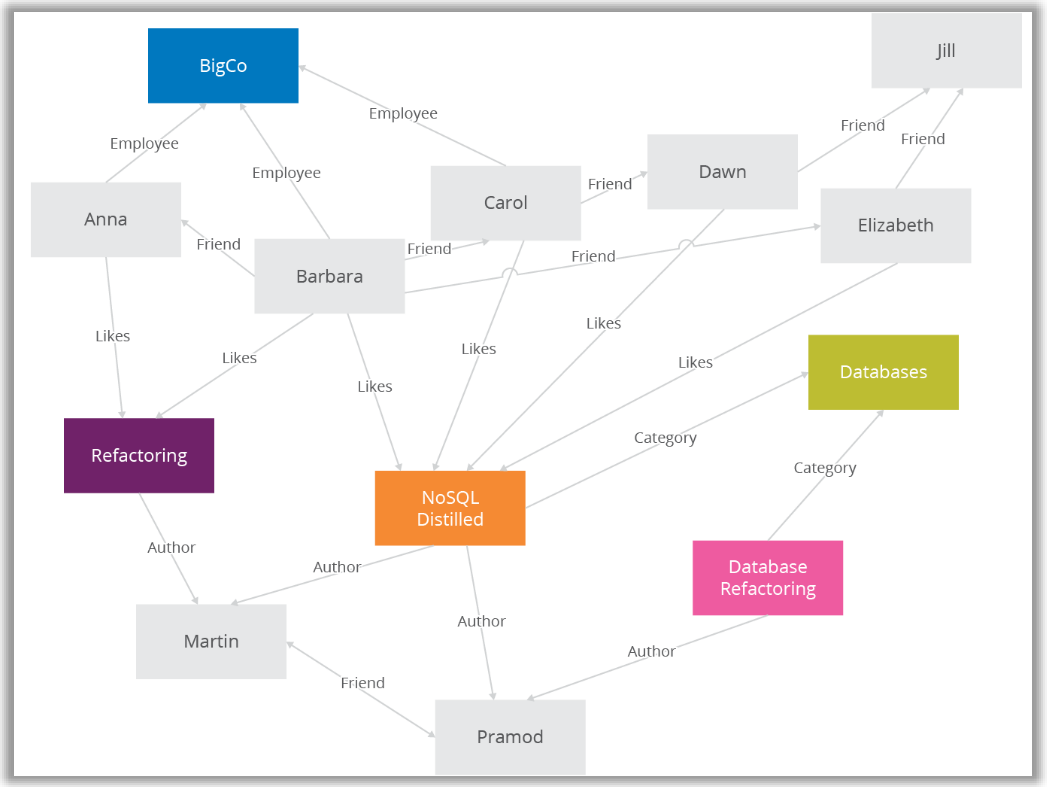


Graph Based

- Store entities and relationships between these entities as nodes and edges of agraph respectively. Entitie shave properties.
- Traversing the relationships is very fast as relationship between nodes is not calculated at query time but is actually persisted as a relationship.

Example: Neo4J, Infinite Graph, OrientDB, FlockDB.

It is well suited for connected data, such as social net works, spatial data, routing information for goods and supply.



CAP Theorem

- According to Eric Brewer a distributed system has 3 properties :
  - **Consistency :**different nodes respond with the same data to the same request
  - **Availability:** the system responds to a request, even if the system isn't working or its data is outdated
  - **Partitions:** the system detects, remains operational during, and heals a partition caused by the failure of one or many nodes
- We can have at most two of these three properties for any shared-data system
- To scale out, we have to partition. It leaves a choice between consistency and availability. *(In almost all cases, we would choose availability over consistency)*
- Everyone who builds big applications builds them on CAP : Google, Yahoo, Facebook, Amazon, eBay, etc.

ACID THEOREM

- According to Eric Brewer a distributed system has 3 properties :
  - **Atomicity:** either all of the operations in a transaction succeed, or none of them do
  - **Consistency:**the database enforces rules about its fields and the relationships between fields
  - **Isolation :**the extent to which rows that a transaction is affecting can be affected by other transactions
  - **Durability :**the database writes its data to a permanent medium (hard drive) so that data is not lost during a power failure or other system failure
- The idea of transactions, their semantics and guarantees, evolved with data management itself. As computers became more powerful, they were tasked with managing more data. Eventually, multiple users shared data on a machine. This led to problems where data could be changed or overwritten while other users were in the middle of a calculation. Something needed to be done; so the academics were called in.

Advantages of NoSQL

- ✓ Cheap and easy to implement (open source)
- ✓ Data are replicated to multiple nodes (therefore identical and fault-tolerant) and can be partitioned
- ✓ When data is written, the latest version is on at least one node and then replicated to other nodes
- ✓ No single point of failure
- ✓ Easy to distribute
- ✓ Don't require a schema

What is not provided by NoSQL

- Joins
- Group by
- **ACID transactions**
- SQL
- Integration with applications that are based on SQL