NON Relational Databases

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.

Why NoSQL?

- ❑ Explosion of social media sites (Facebook, Twitter, Google etc.) with large data needs. (Sharding is 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.

NoSQL characteristics

•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.

NoSQL characteristics

•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 database are classified into four types:

- Key Value pair based
- Column based
- Document based
- Graph based

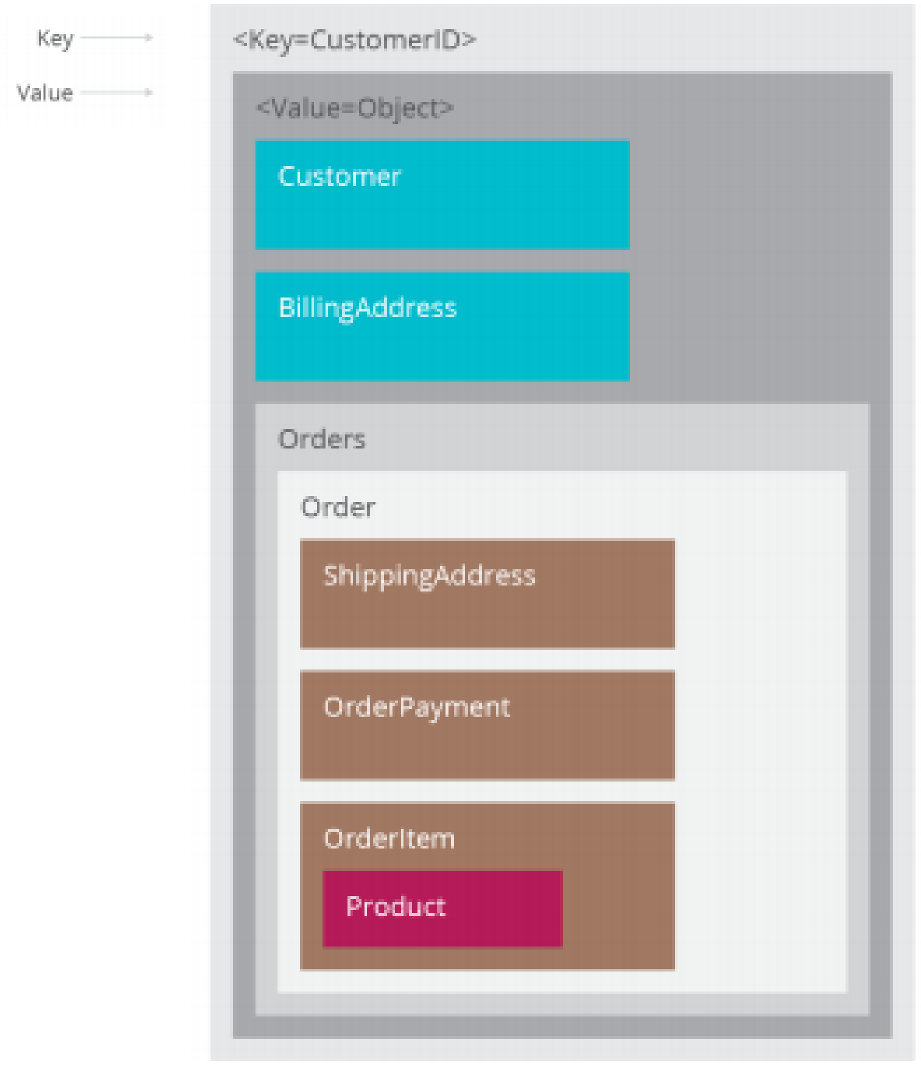
Key Value Pair Based

- 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.



Column based

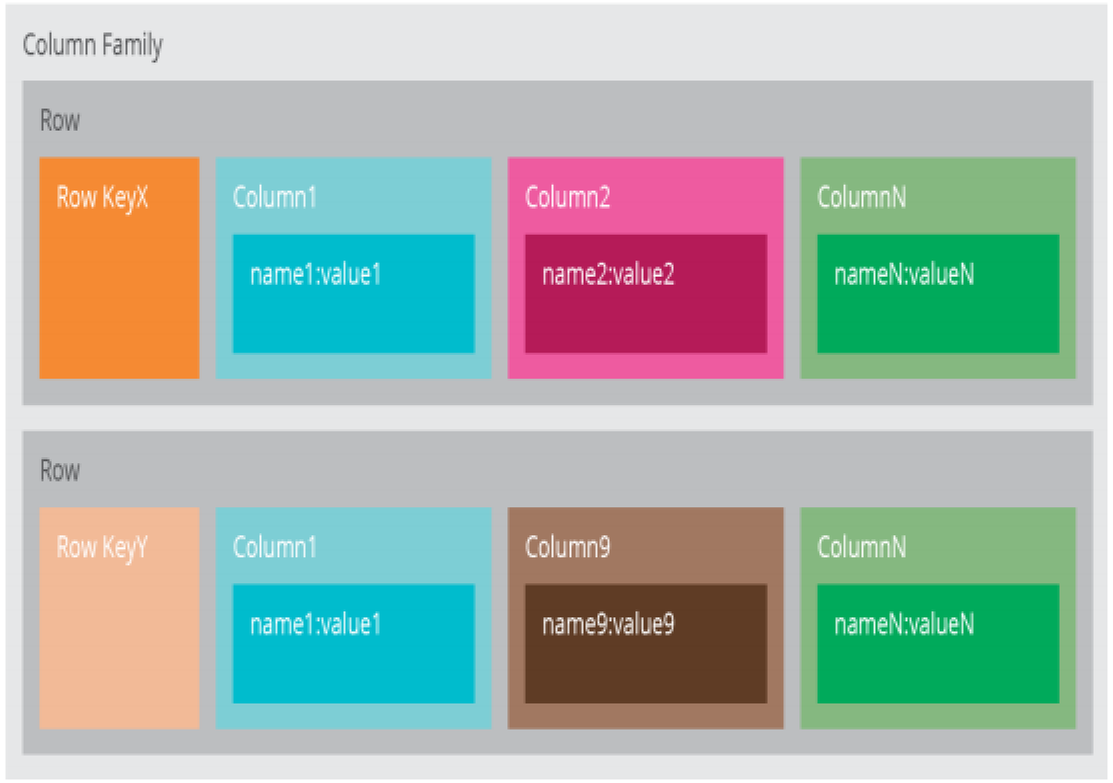
It store data as Column families containing rows that have many columns associated with a row key. Each row can have different columns.

Column families are groups of related 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 in early development, changing query patterns.



Document based

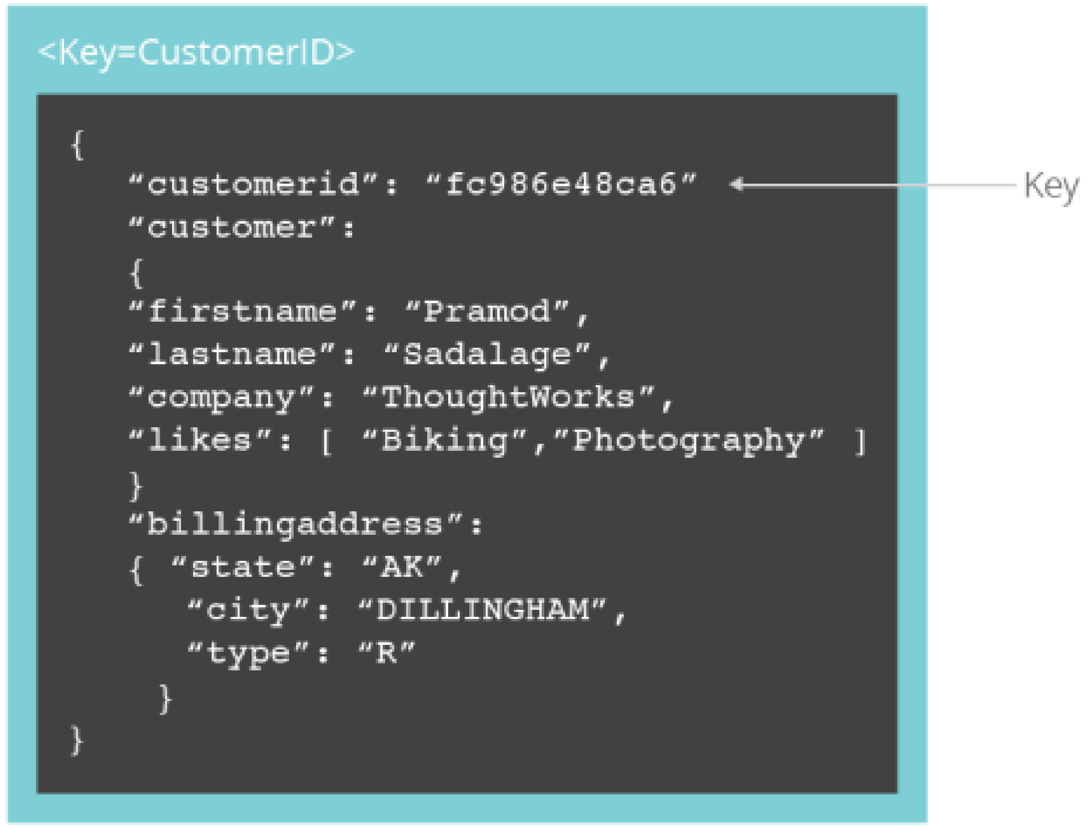
The database stores and retrieves documents. It stores documents in the value part of the key-value store.

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

Example: Lotus Notes, MongoDB, Couch DB, Orient DB, Raven DB.

We use it for content management systems, blogging platforms, web analytics, real-time analytics, e-commerce applications.

We would avoid it for systems that need complex transactions spanning multiple operations or queries against varying aggregate structures.



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

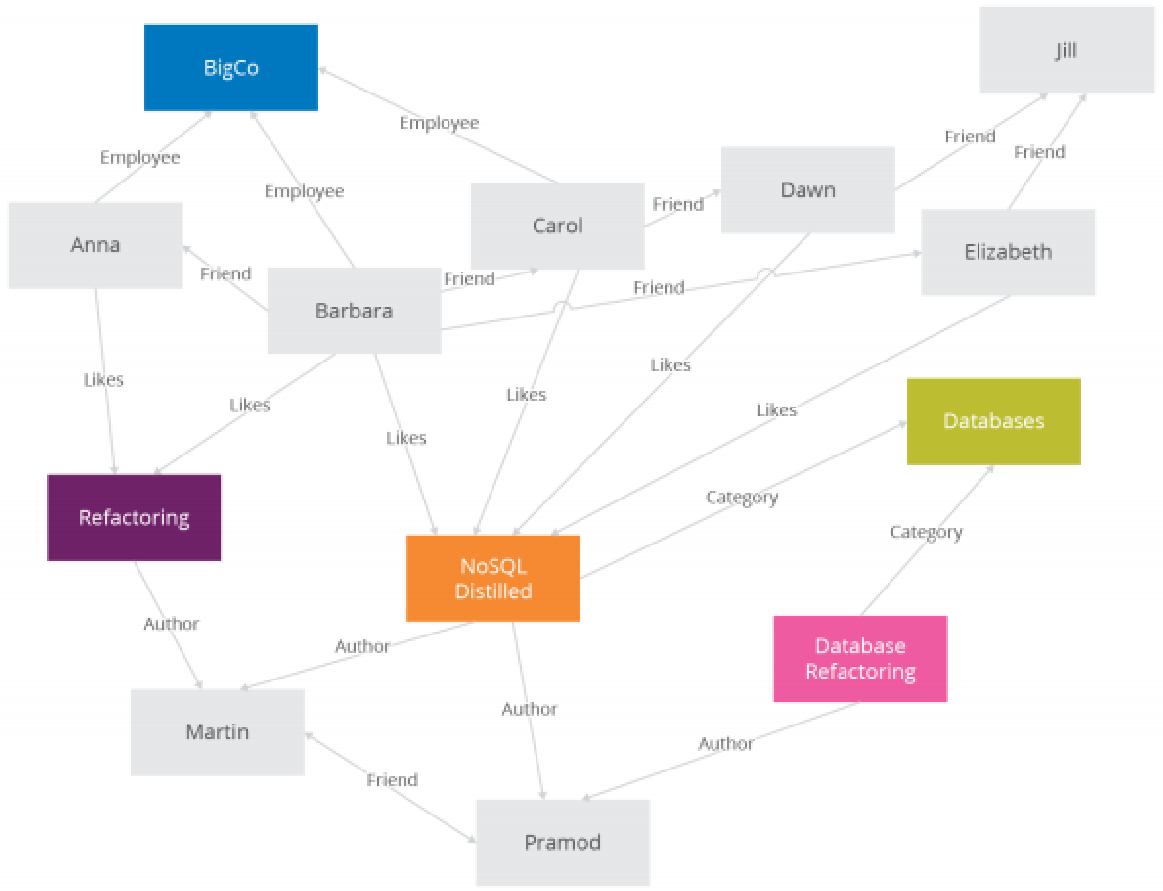
Graph Based

Store entities and relationships between these entities as nodes and edges of a graph respectively.

Entities have properties. Traversing the relationships is very fast as relationship between nodes is not calculated at query time but is actually persisted as a relationship.

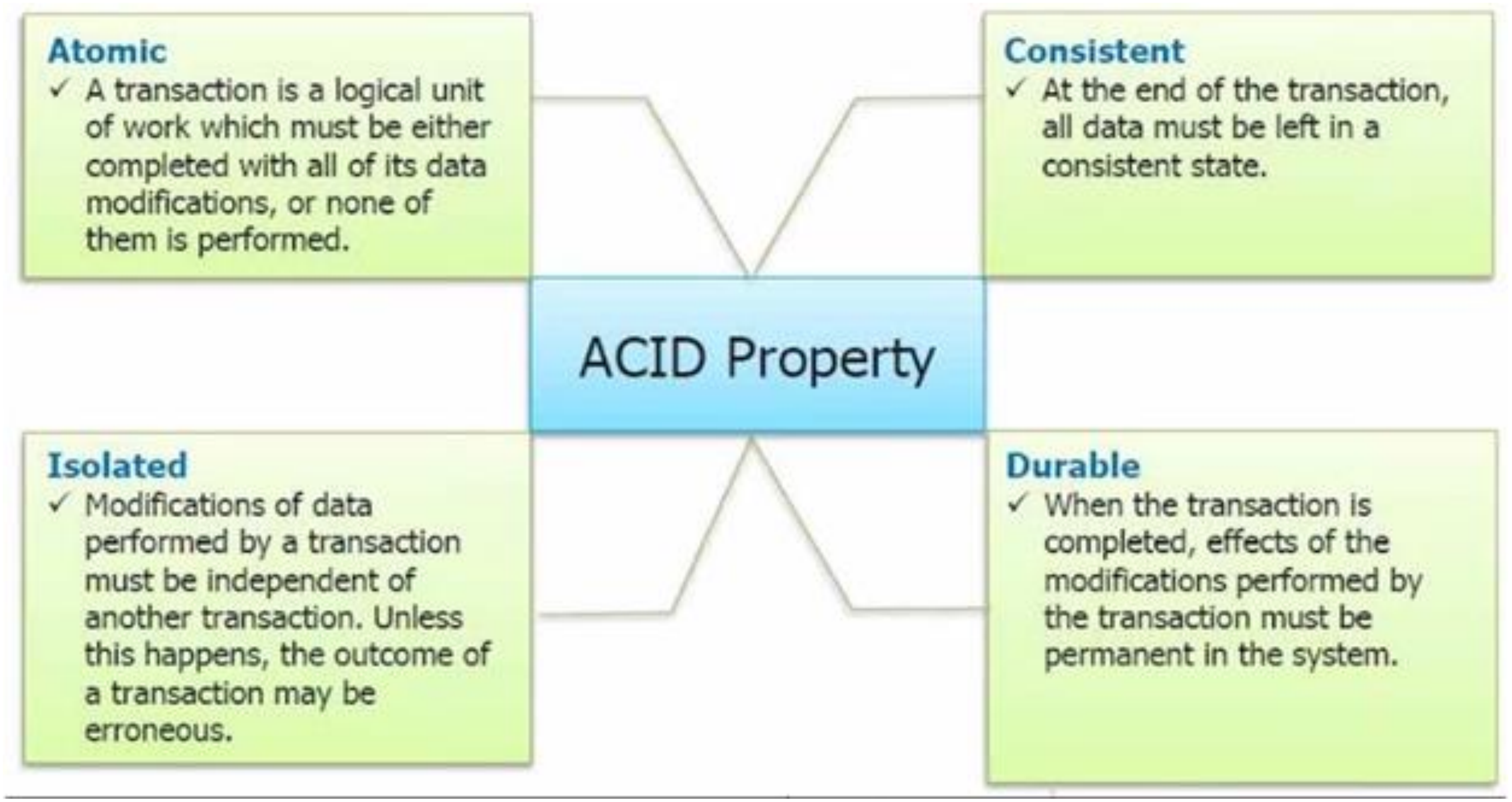
Example: Neo4J, Infinite Graph, OrientDB, FlockDB.

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



ACID Theorem

In computer science, ACID (atomicity,consistency ,isolation durability) is a set of properties of database transactions intended to guarantee data validity despite errors, power failures, and other mishaps.


ACID Property

Atomic
✓ A transaction is a logical unit of work which must be either completed with all of its data modifications, or none of them is performed.

Consistent
✓ At the end of the transaction, all data must be left in a consistent state.

Isolated
✓ Modifications of data performed by a transaction must be independent of another transaction. Unless this happens, the outcome of a transaction may be erroneous.

Durable
✓ When the transaction is completed, effects of the modifications performed by the transaction must be permanent in the system.

CAP Theorem

❑According to Eric Brewer a distributed system has 3 properties :

- Consistency
- Availability
- Partitions

❑ 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.

Advantages of NoSQL

- ❑ Cheap and easy to implement (open source).
- ❑ Data are replicated to multiple nodes (therefore. identical and faulttolerant) 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.