NoSQL Data management

Contents

- What is SQL
- Features of SQL
- Popular SQL Databases
- No SQL
- Features of NoSQL
- Difference between SQL and NoSQL
- Interactions of layers in NoSQL
- Why NoSQL
- Characteristics of NoSQL
- Advantages and Disadvantages of NoSQL
- Types of NoSQL data models

NoSQL Data management

- **SQL (Structured Query language)** is the standard language for communicating with relational, table-based databases.
- Typically, these databases are called SQL databases.
- With SQL programming, you can easily search, insert, update, and delete database data.
- SQL is an incredibly powerful and versatile language, which makes it a very safe choice.
- With SQL, you are required to pre-define schemas that structure the way your data will be organized.

Features:

- **Scalabality:** SQL databases can scale vertically by adding CPU, RAM, or SSD. By adding more resources, the database can handle more load.
- **Community:** SQL databases have been around for a long time, which means that they have a wide community with great documentation and support.
- **Versatile:** SQL databases can be applied to a ton of data models, which makes it suitable for all types of applications. SQL also has a large collection of tools and functions that makes it very powerful.

Popular SQL databases:

- MySQL
- PostgreSQL
- Microsoft SQL Server
- Oracle Express Edition

NoSQL

- **NoSQL** are non-relational databases, that varies from traditional database.
- Nosql is provided for distributed data stores where there is a need for large scale of data storing.
- For example
- Google and FACE book
- In Nosql tables are stored as ASCII files with each tuple represented by a row and fields separated with tabs.
- The database is manipulated through shell scripts that can be combined to UNIX pipeline
- These databases can be structured in a more flexible form since we do not need to predefine a schema.
- However, there's a common misconception that NoSQL databases do not hold relationship data well.
- In reality, they simply create relationships differently from SQL databases.
- Nosql doesn't use SQL as query language

Features:

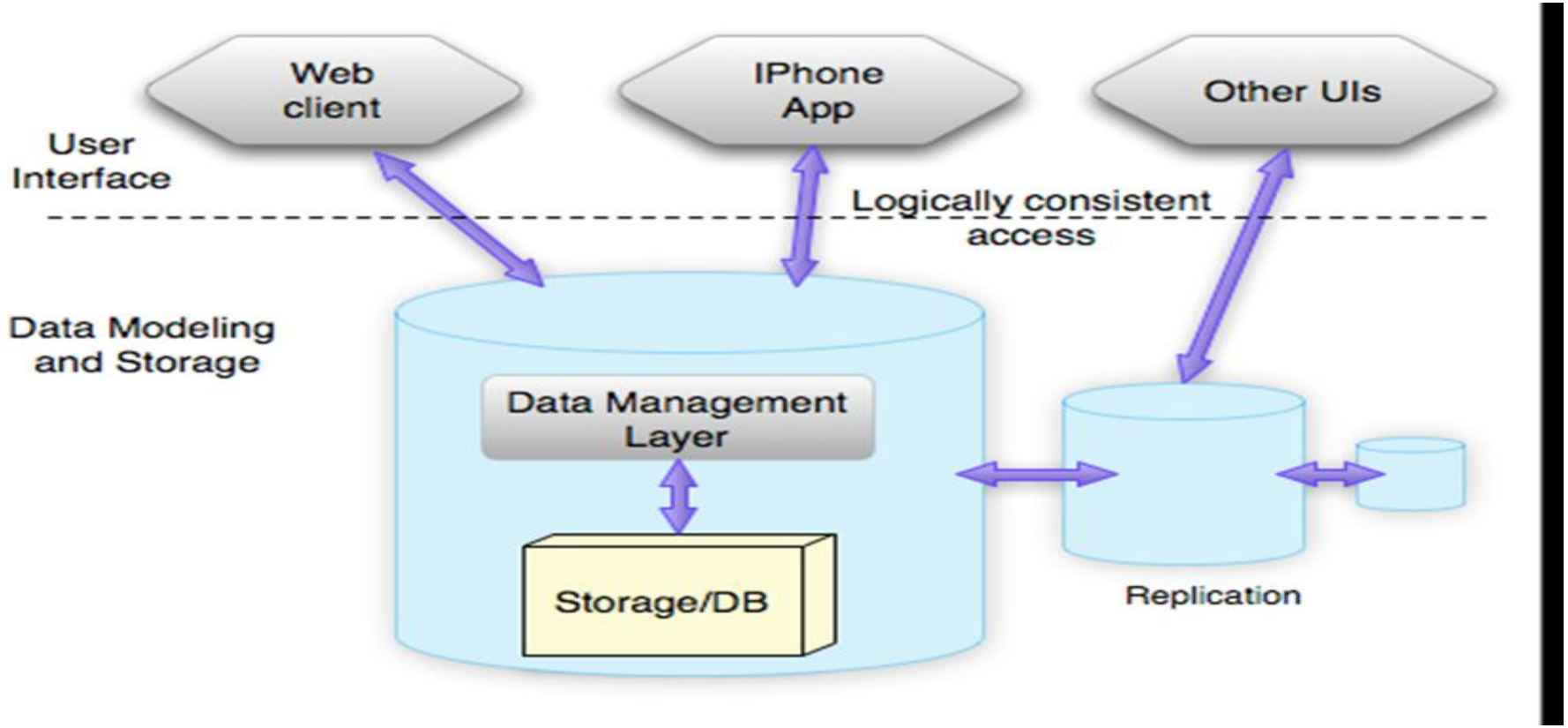
Scalability: Similar to SQL databases, NoSQL databases are also easily scalable.

- However, they scale horizontally meaning that you add more servers to your NoSQL database. Ultimately, NoSQL scales better for larger and more powerful applications.
- **Community:** NoSQL is relatively new compared to SQL databases meaning that sometimes there will be less documented support for utilizing the database.
- However, the popularity of NoSQL is rapidly growing in the industry.
- **Flexibility:** With a NoSQL database, you are given more flexibility to store your data without a pre-defined structure, which is useful depending on the application you are building.

Differences between sql and NoSQL

- **SQL**
- SQL databases are table based databases
- Have *predefined schema*
- Are vertically scalable
- Use SQL (Structured Query Language) for defining and manipulating the data
- A good fit for the complex query intensive environment
- Emphasize **ACID** properties (Atomicity, Consistency, Isolation, and Durability)
- Examples include: MySQL, Oracle, Sqlite, Postgres, and MS-SQL
- **NoSQL**
- NoSQL databases are document-based, key-value pairs, and graph databases
- Have *dynamic schema*
- Are horizontally scalable
- Focused on the collection of documents
- Not ideal for complex queries
- Follow the **Brewers CAP theorem** (Consistency, Availability, and Partition tolerance)
- Examples include: MongoDB, BigTable, Redis, RavenDb, Neo4j, and CouchDb

Interaction of Layers in NoSQL



Why NoSql

- In present day, as we are handling s humongous amount of data, data being organized and well-structured actually creates a problem, especially at extremely large volumes.
- The structured approach of RDBMS database like SQL slows down performance as data volume or size gets bigger and it is also not scalable to meet the needs of Big Data.
- NoSQL was conceived as a completely different framework of databases that allows for high-performance, processing of information at a much bigger scale.
- This is the database well-adapted to the high demands of big data.
- The new version of NoSQL runs the database MongoDB, which stores unstructured data.
- This means that you don't need to know in advance exactly what kind of data you'll be collecting and storing. You can collect a lot more data of different kinds and can access and analyze data much faster
- NoSQL is centered on the concept of distributed databases, where unstructured data may be stored across multiple processing nodes, and often across multiple servers.
- This distributed architecture allows NoSQL databases to be horizontally scalable as data continues to explode, just add more hardware to keep up, with no slowdown in performance.

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 f 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.Nosql offers range of options for consistency and distribution
- 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.
- Nosql is open source project

Advantages of NoSQL

- Can be used as Primary or Analytic Data Source
- Big Data Capability
- No Single Point of Failure
- Easy Replication
- No Need for Separate Caching Layer
- It provides fast performance and horizontal scalability.
- Can handle structured, semi-structured, and unstructured data with equal effect
- Object-oriented programming which is easy to use and flexible
- NoSQL databases don’t need a dedicated high-performance server
- Support Key Developer Languages and Platforms
- Simple to implement than using RDBMS
- It can serve as the primary data source for online applications.
- Handles big data which manages data velocity, variety, volume, and complexity
- Excels at distributed database and multi-data center operations
- Eliminates the need for a specific caching layer to store data
- Offers a flexible schema design which can easily be altered without downtime or service disruption

Disadvantages of NoSQL

- No standardization rules
- Limited query capabilities
- RDBMS databases and tools are comparatively mature
- It does not offer any traditional database capabilities, like consistency when multiple transactions are performed simultaneously.
- When the volume of data increases it is difficult to maintain unique values as keys become difficult
- Doesn’t work as well with relational data
- The learning curve is stiff for new developers
- Open-source options so not so popular for enterprises.

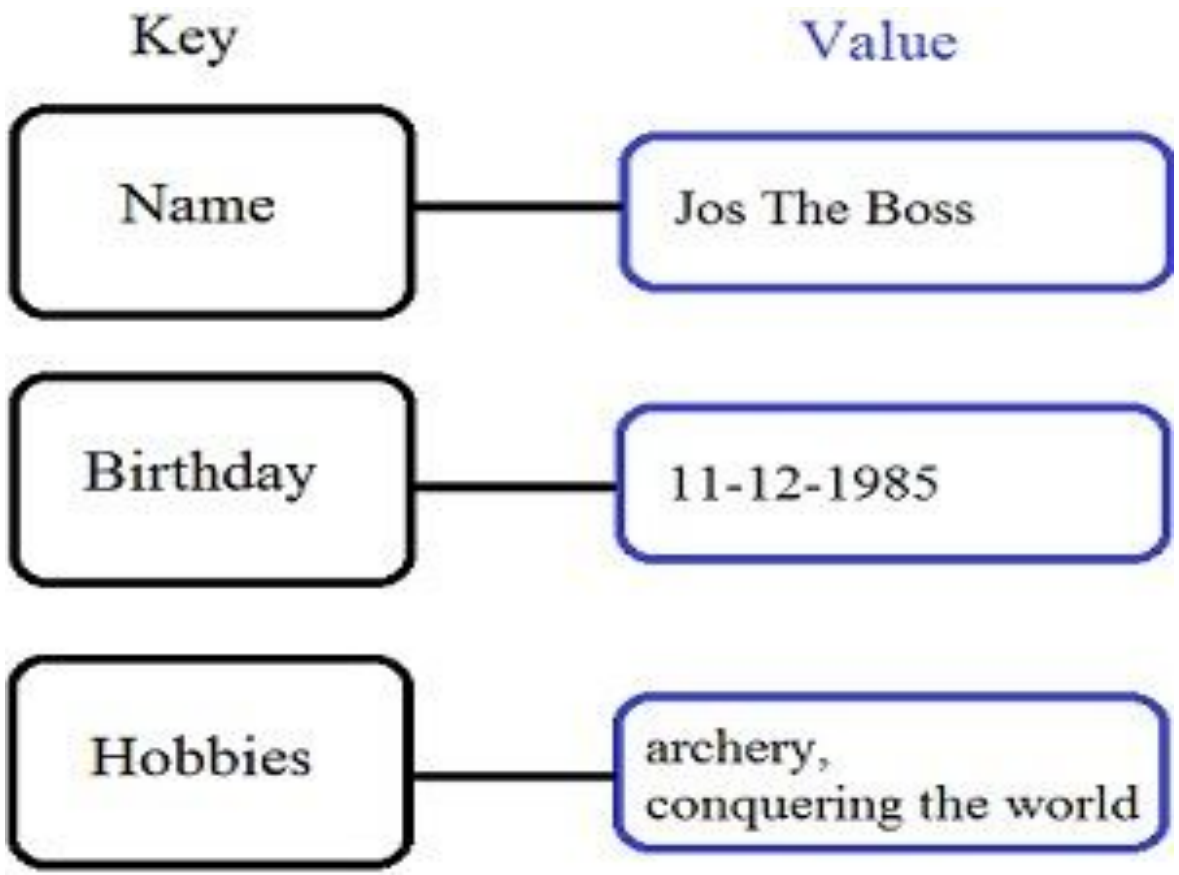
Types of NoSQL Data Models

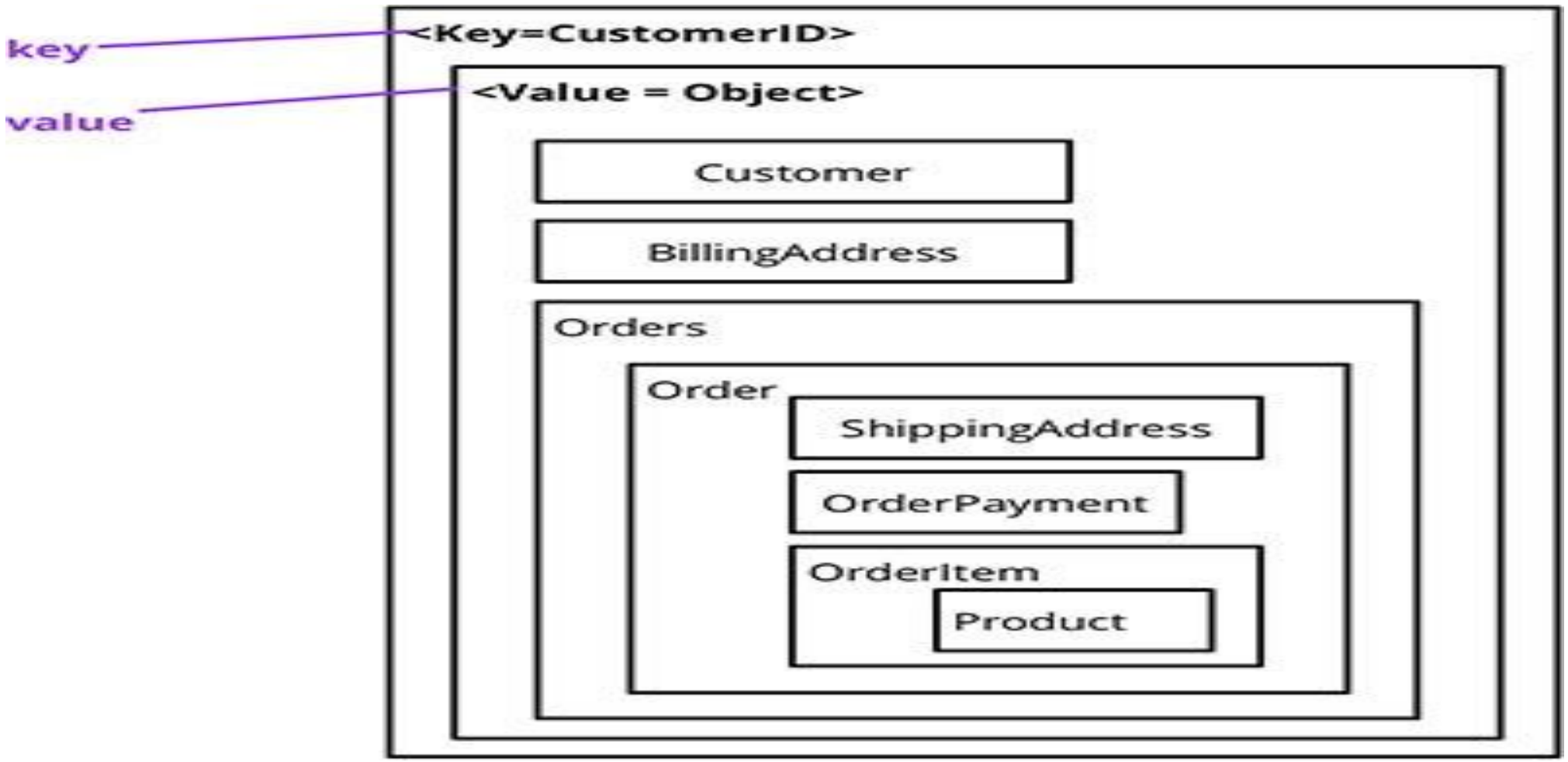
- Key value data model
- Column oriented data model
- Document data model
- Graph databases

Key value data model

- The simplest type of NoSQL database is a key-value store .
- Every data element in the database is stored as a key value pair consisting of an attribute name (or "key") and a value.
- In a sense, a key-value store is like a relational database with only two columns: the key or attribute name (such as state) and the value (such as Alaska).
- Use cases include shopping carts, user preferences, and user profiles.
- In a key value data model the client can get the value for a key, put a value for key or delete key from data store.
- Here value is BLOB(binary large object) that is only concerned about data and not what is inside. It is responsibility of application to understand what exactly is stored
- Blob is also scalable because it uses primary key access for different purposes
- Examples of key-value stores are **Redis, Voldemort, Riak, and Amazon's DynamoDB.**

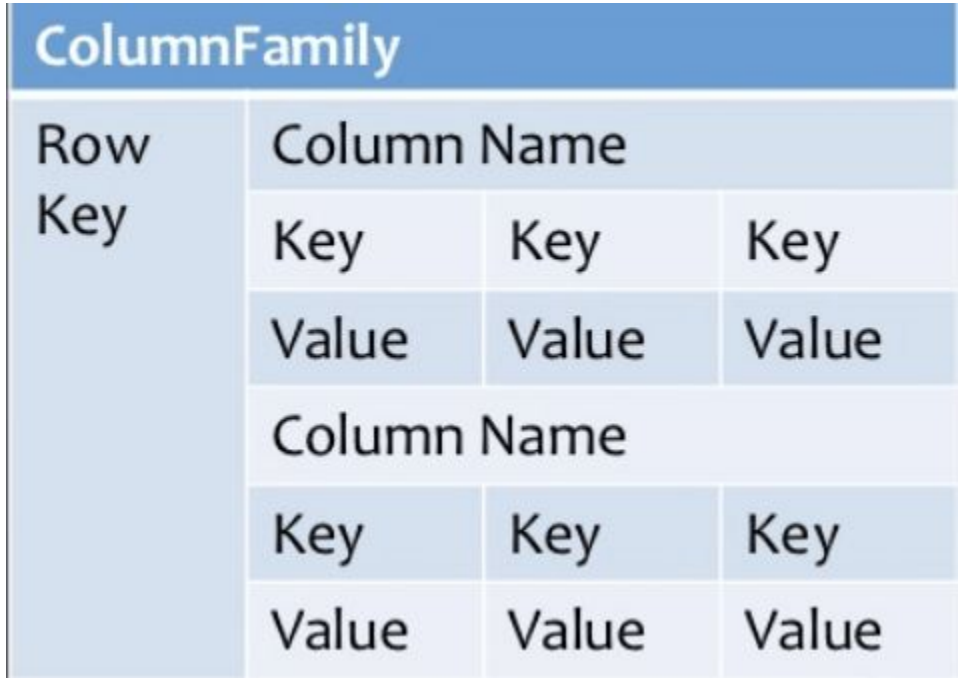
Key value data model

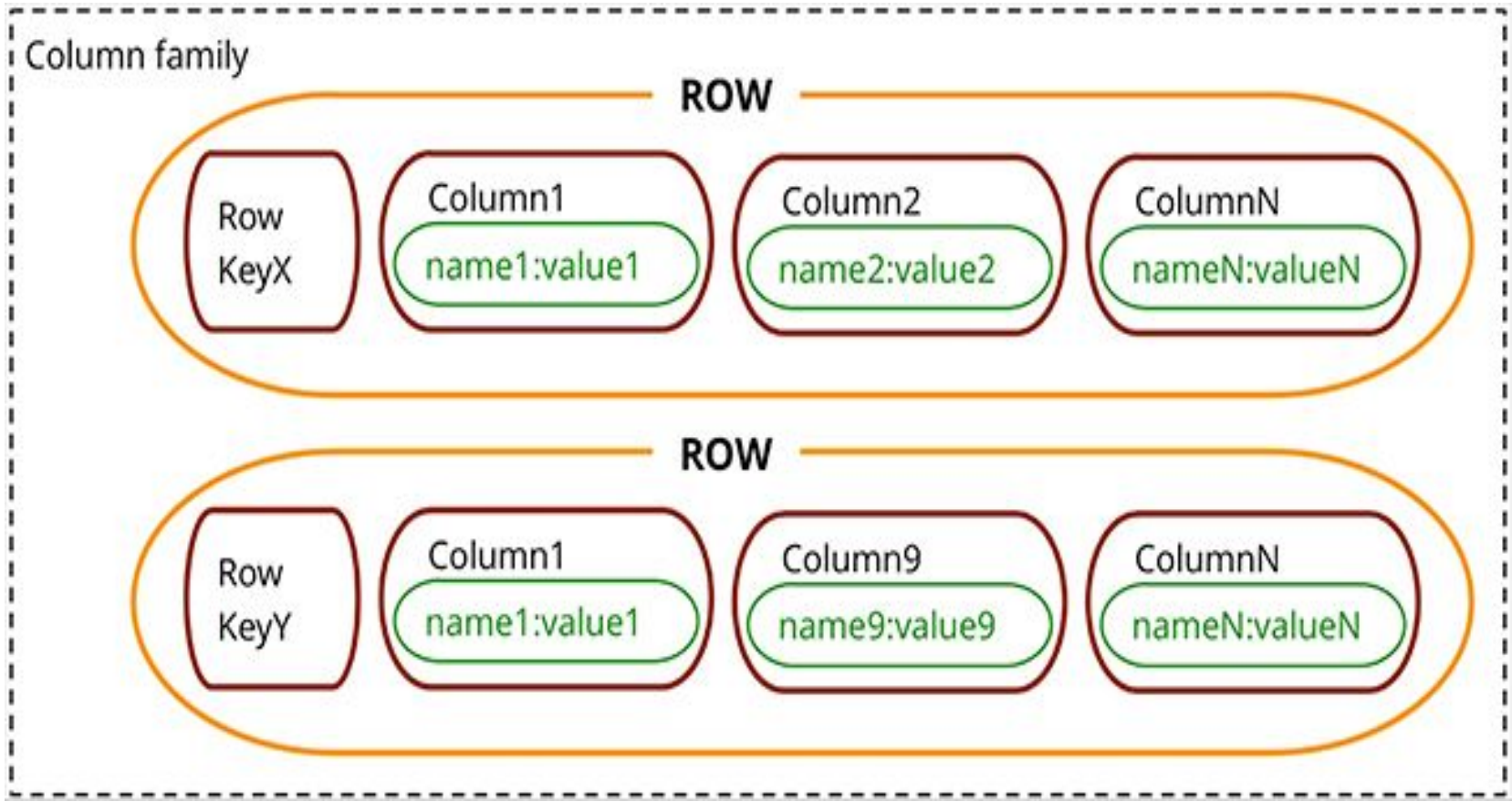



key
value
<Key=CustomerID>
<Value = Object>
Customer
BillingAddress
Orders
Order
ShippingAddress
OrderPayment
OrderItem
Product

Column oriented data model

- Column-oriented databases work on columns and are based on Big Table paper by Google. Every column is treated separately.
- The values of single column databases are stored contiguously.




Column family
ROW
Row
KeyX
Column1
name1:value1
Column2
name2:value2
ColumnN
nameN:valueN
ROW
Row
KeyY
Column1
name1:value1
Column9
name9:value9
ColumnN
nameN:valueN

Column oriented data model

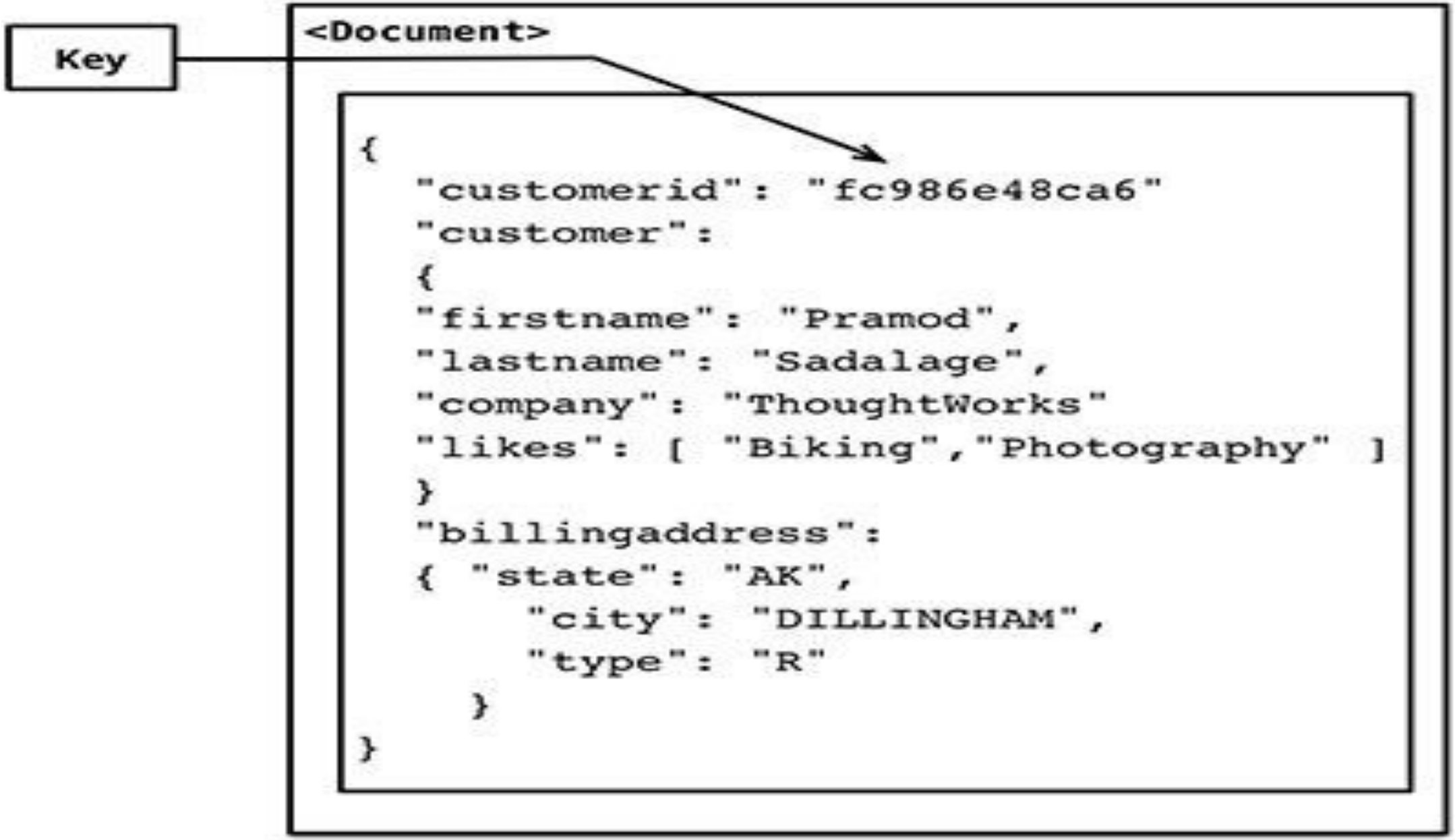
- They deliver high performance on aggregation queries like SUM, COUNT, AVG, MIN, etc. as the data is readily available in a column.
- Column-based NoSQL databases are widely used to manage **data warehouses**, **business intelligence**, **Library card catalogs**,
- **HBase**, **Cassandra**, **HBase**, **Hypertable** are examples of a column-based database.

Document data model

- There are many types of document databases such as xml,json,bson
- These document databases are self describing hierarchical tree data structures that contain maps ,collections and scalar values.
- Mainly stores documents in the value part of the key /value store
- Provides indexing and searching
- Document database have a different structure to store the information related to an entity unlike relational database , which uses tabular format

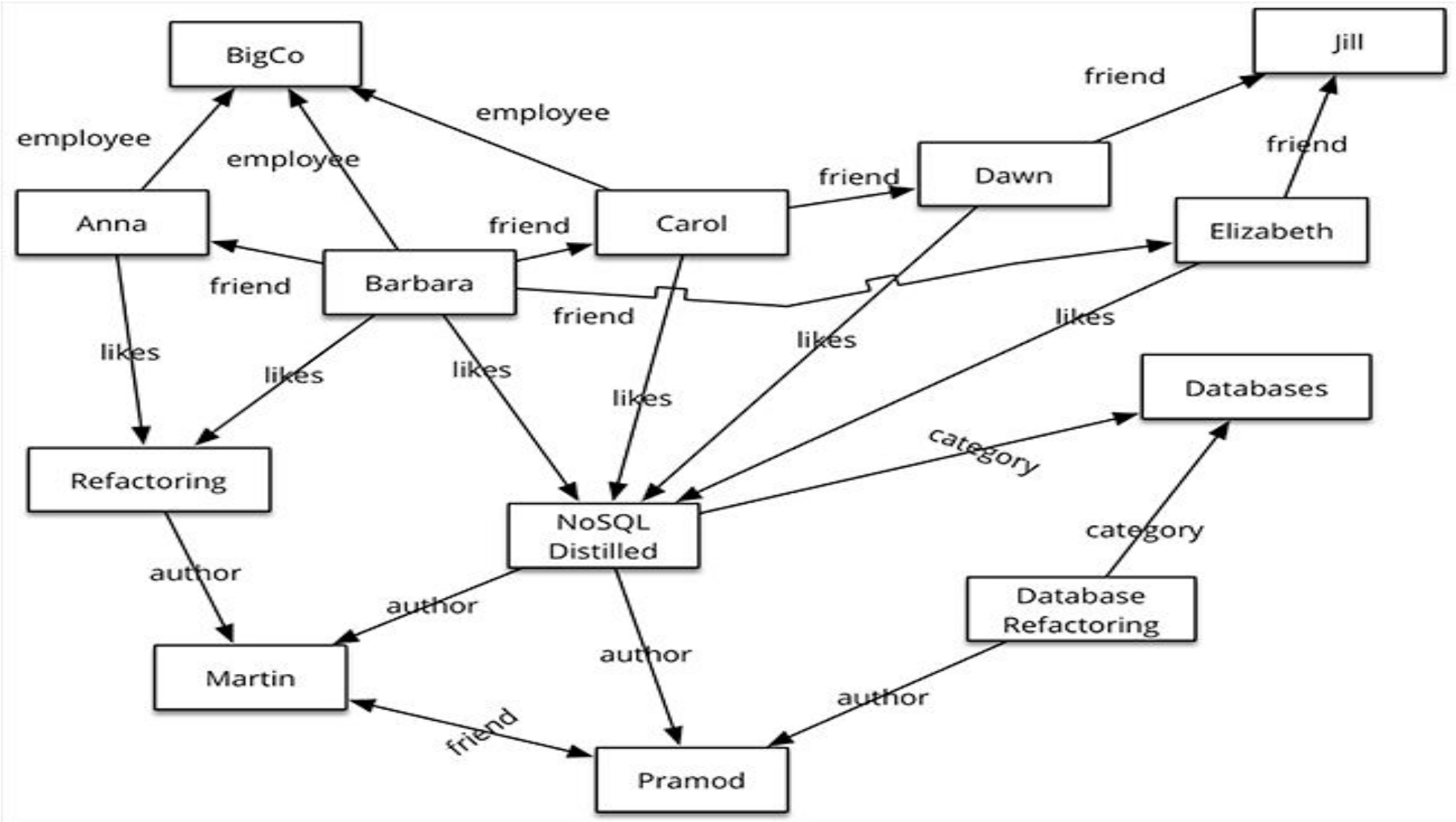
Document data model

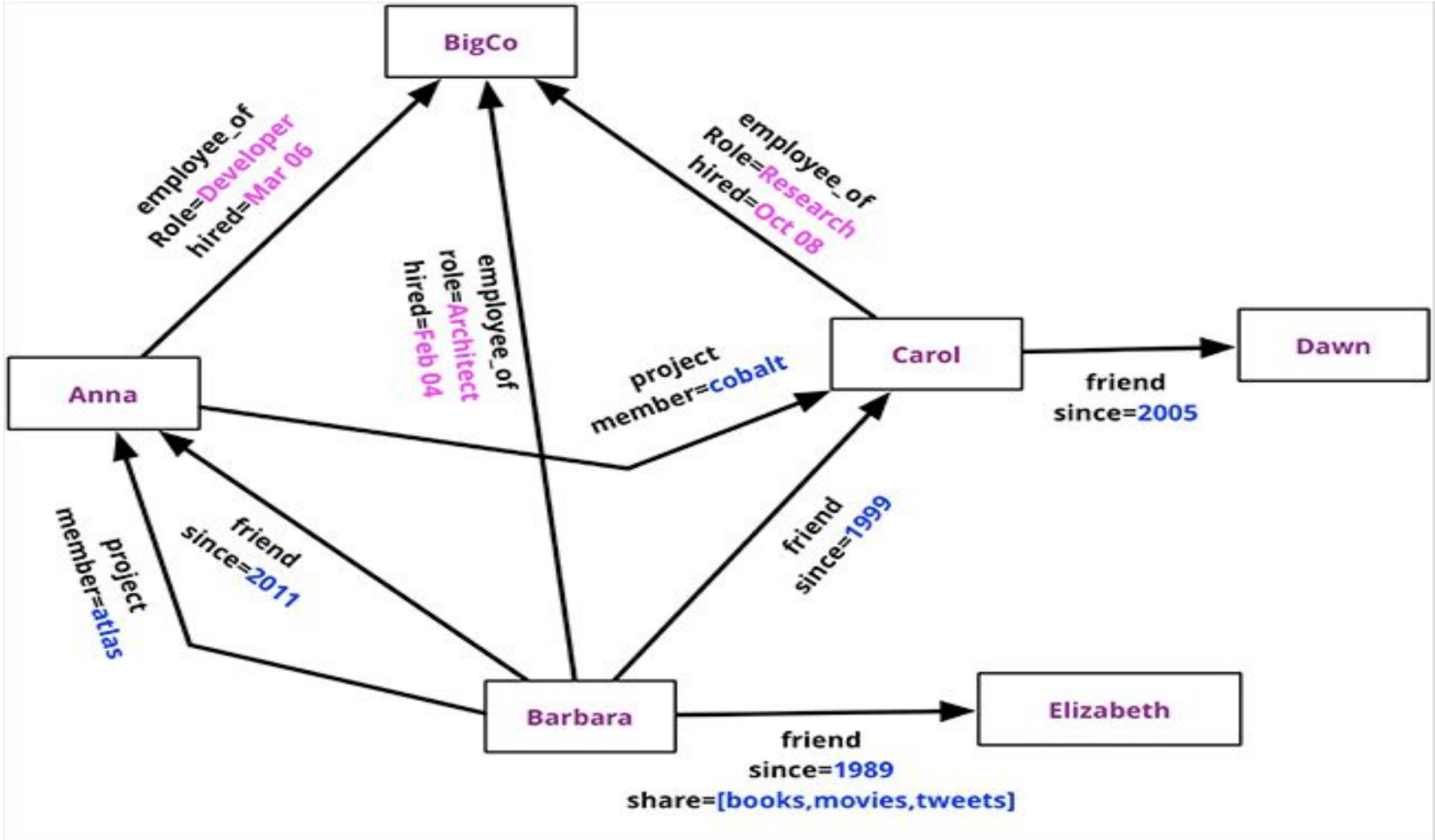
- Though the document oriented model is rich in performance and scalability it doesn’t provide ACID and data integrity
- Therefore data integrity, transactions and locks are no longer there
- Document database and relational databases are no replacements for each other
- For example open sky corporation uses both Mysql(relation) and mongo dB(document)
- Popular Document databases are MongoDb,CouchDb,Terrastore,OrientDb,RavenDb ,lotus notes


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

Graph Databases

- Graph databases allow you to store entities and relationships between these entities.
- Entities are also known as nodes, which have properties. Think of a node as an instance of an object in the application.
- Relations are known as edges that can have properties.
- Edges have directional significance nodes are organized by relationships which allow you to find interesting patterns between the nodes.
- The organization of the graph lets the data to be stored once and then interpreted in different ways based on relationships.


BigCo
Jill
employee
employee
employee
friend
Dawn
friend
Anna
Carol
friend
Elizabeth
friend
Barbara
friend
friend
likes
likes
likes
likes
likes
likes
Databases
Refactoring
category
NoSQL
Distilled
author
category
author
Database
Refactoring
author
author
Martin
friend
author
Pramod


BigCo
employee_of
Role=Developer
hired=Mar 06
employee_of
role=Architect
hired=Feb 04
employee_of
Role=Research
hired=Oct 08
Anna
project
member=cobalt
Carol
friend
since=2005
Dawn
project
member=atlas
friend
since=2011
friend
since=1999
Barbara
Elizabeth
friend
since=1989
share=[books,movies,tweets]

- A graph database is optimized to capture and search the connections between data elements, overcoming the overhead associated with JOINing multiple tables in SQL.
- Very few real-world business systems can survive solely on graph queries.
- As a result graph databases are usually run alongside other more traditional databases.
- Use cases include fraud detection, social networks, and knowledge graphs.
- As you can see, despite a common umbrella, NoSQL databases are diverse in their data structures and their applications.

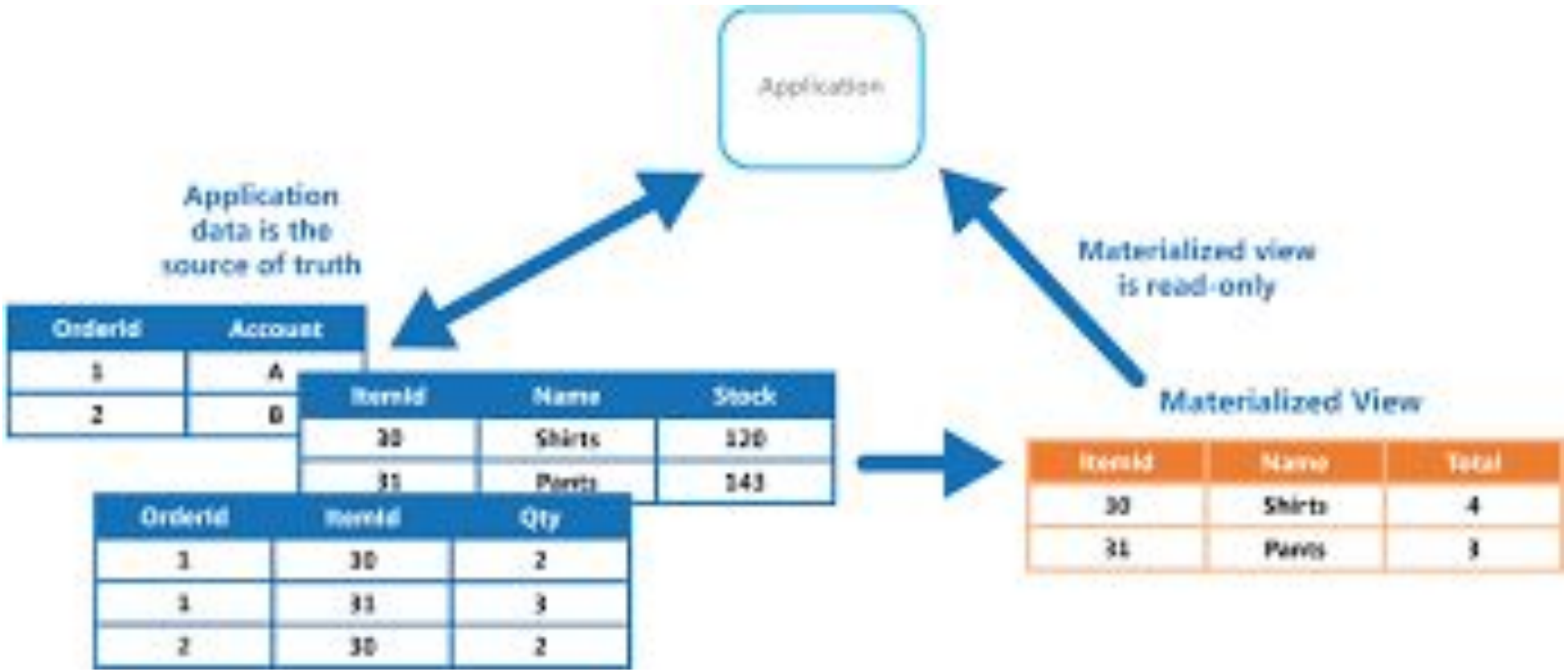
- Usually, when we store a graph-like structure in RDBMS, it's for a single type of relationship
- Adding another relationship to the mix usually means a lot of schema changes and data movement, which is not the case when we are using graph databases.
- Similarly, in relational databases we model the graph beforehand based on the Traversal we want if the Traversal changes, the data will have to change.
- graph databases, traversing the joins or relationships is very fast.
- We can have nodes with different types of relationship between them
- Relationship are of major importance in graph databases.
- They help in deriving most of the values in these databases
- Popular graph databases are Neo4J,infinite Graph,FlockDB

Schema Less Data bases

- As the name suggests a database without any schema is known as Schema less database.
- A well defined schema of a database describes the tables columns and data types of the values in the columns of the database.
- Storing data in NoSQL is easier as compared to sql..
- In case of key value database you can store any type of data under particular key
- In document database no restriction on type of document you want to store
- Under column database you can store any type of data according to your requirement in graph database no restrictions in adding edges or properties to nodes
- In case of schema defined database u need plan for the collection of data to be stored in database
- No such planning is required for schema less database
- Schema less database also makes it easier to arrange non uniform information where each record has an alternate set of fields
- Schema less database permits each record to contain exactly what it needs. This type of database eliminates many conditions found in fixed schema databases

Materialized Views

- The priority for developers and data administrators in data storage is how the data is stored and how it is read
- The chosen storage format is usually closely related to the format of the data requirements for managing data size and data integrity and kind of store in use.
- For example: when using document model data is often represented as series of aggregates each of which contains all information about entity
- This may have negative effect on queries.
- When a query requires only a subset of the data from some entities, such as summary of orders of several customers without all the order details, it needs to extract all of the data for the relevant entities in order to obtain the required information
- To support efficient querying a common solution is to generate in advance a view that contains the data in a format most suited to generate the required dataset
- This view is known as materialized view
- The materialized view can be defined as a pattern that generates prepoluted views of data in environments where the source data is in a format that is not suitable for querying


Application
Application data is the source of truth
Materialized view is read-only
Materialized View
Orderid
Account
1
A
2
B
Itemid
Name
Stock
30
Shirts
120
31
Pants
143
Orderid
Itemid
Qty
1
30
2
1
31
3
2
30
2
Itemid
Name
Total
30
Shirts
4
31
Pants
3

- These materialized views which contain only data required by a query ,allow applications to quickly obtain the information they need.
- In addition to joining tables or combining data entities, materialized views may include the current values of the calculated column or data items the result of combining values or executing transformation
- On the data items and values specified as part of the query
- A materialized view may even be optimized for just single query
- Materialized view and data it contains is completely disposable
- Never updated directly by the application so it is effectively specialized cache
- When there s change in the source data while creating view the view must be updated to include new information

Distributed Models

- Aggregate oriented databases allow easy distribution of data.
- In such databases the distribution mechanism is just concerned abt movement of aggregate data and not the related data, as all the related data is stored in the aggregate
- Distribution of data can be done in two ways
- **Through sharding**
- Sharding is one of the major technique of data distribution.
- It is used to distribute various types of data across multiple servers.
- Therefore each server acts as a single source for a subset of data

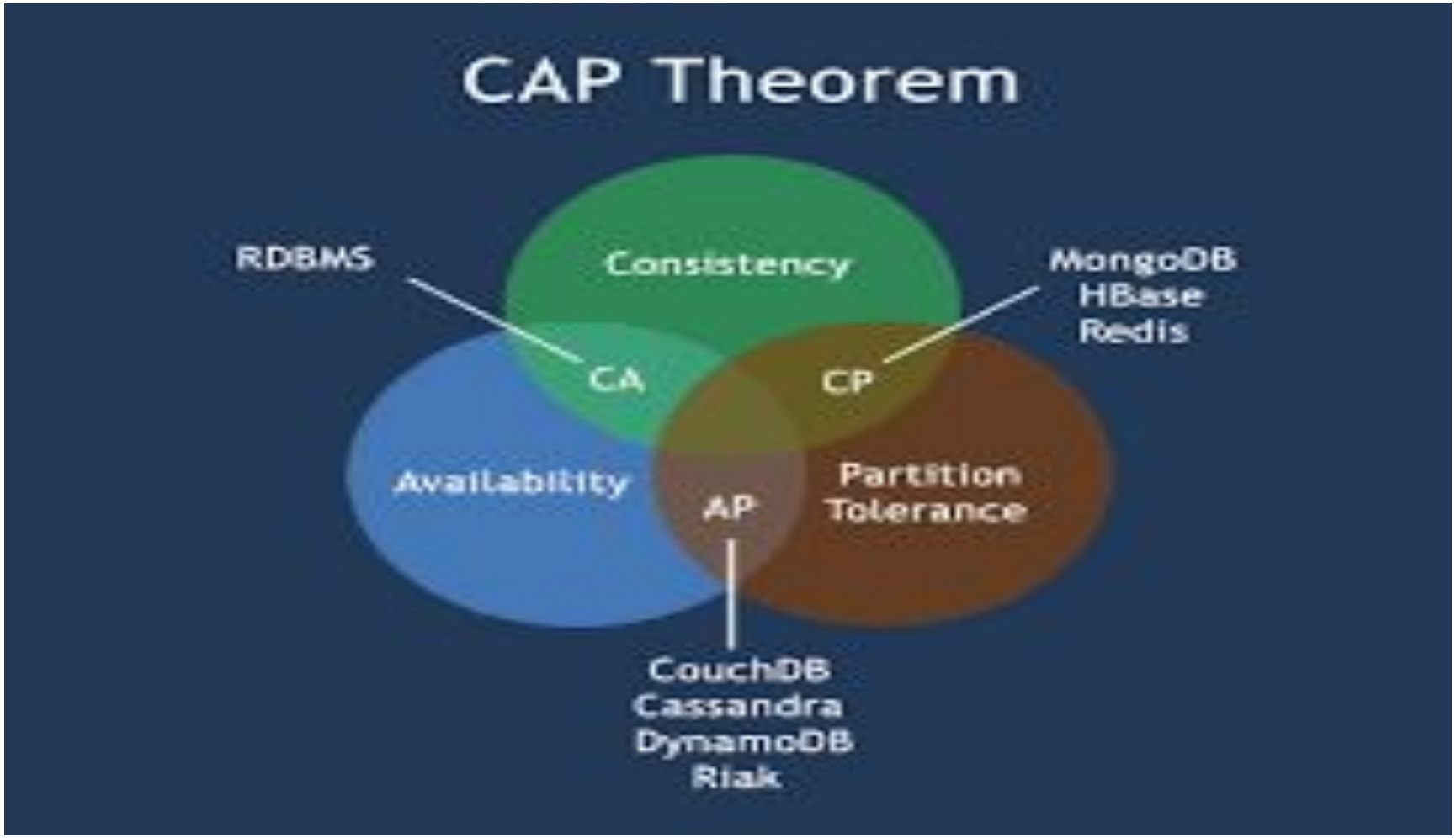
Through Replication

- Replication is one of the major technique for fault tolerance.
- The idea is to copy data across multiple servers so that each bit of data can be found in multiple places
- Replication occurs in two forms
- Master Slave replication
- Peer to peer replication

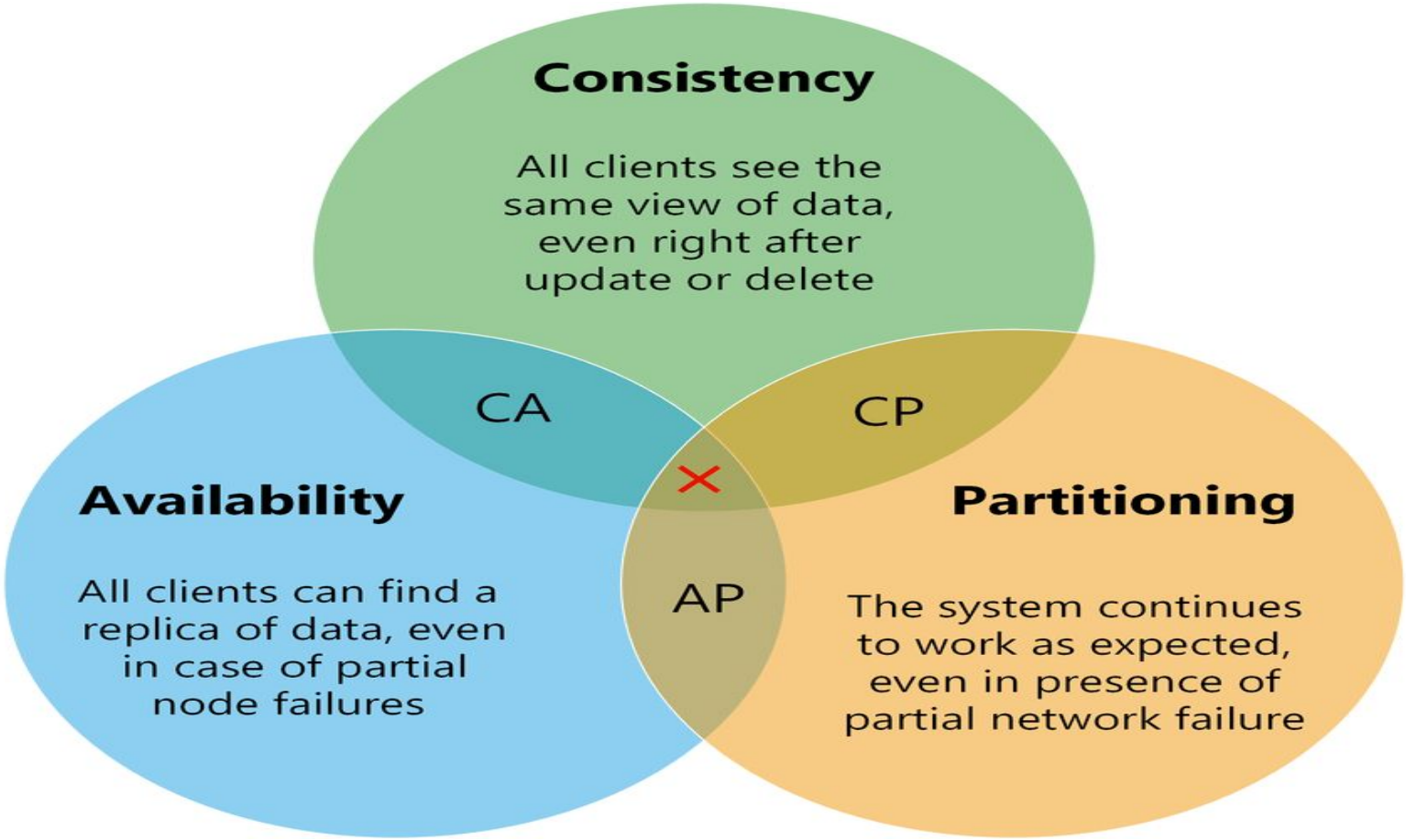
CAP Theorem

- In distributed database the three important aspects of CAP thm are
- Consistency
- Availability
- Partition tolerance
- The CAP thm states that in any distributed systems we can select only two aspects
- It is desirable to have Consistency, Availability, Partition tolerance in every system, unfortunately no system can achieve all three at the same time

CAP Theorem


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

CAP Theorem


Consistency
All clients see the same view of data, even right after update or delete
CA
CP
Availability
All clients can find a replica of data, even in case of partial node failures
AP
Partitioning
The system continues to work as expected, even in presence of partial network failure

- Consistency: Implies consistent data in the database even after the execution of an operation
- For example: after an insert operation all the clients will see the same data
- Availability: Implies that the system is always available in other words there is no downtime
- Partition Tolerance: Implies that the system continues to work even though the communication among the servers is unreliable
- It is desirable to have Consistency, Availability, Partition tolerance in every system, unfortunately no system can achieve all three at the same time
- Therefore according to CAP thm two out of three requirement should be followed

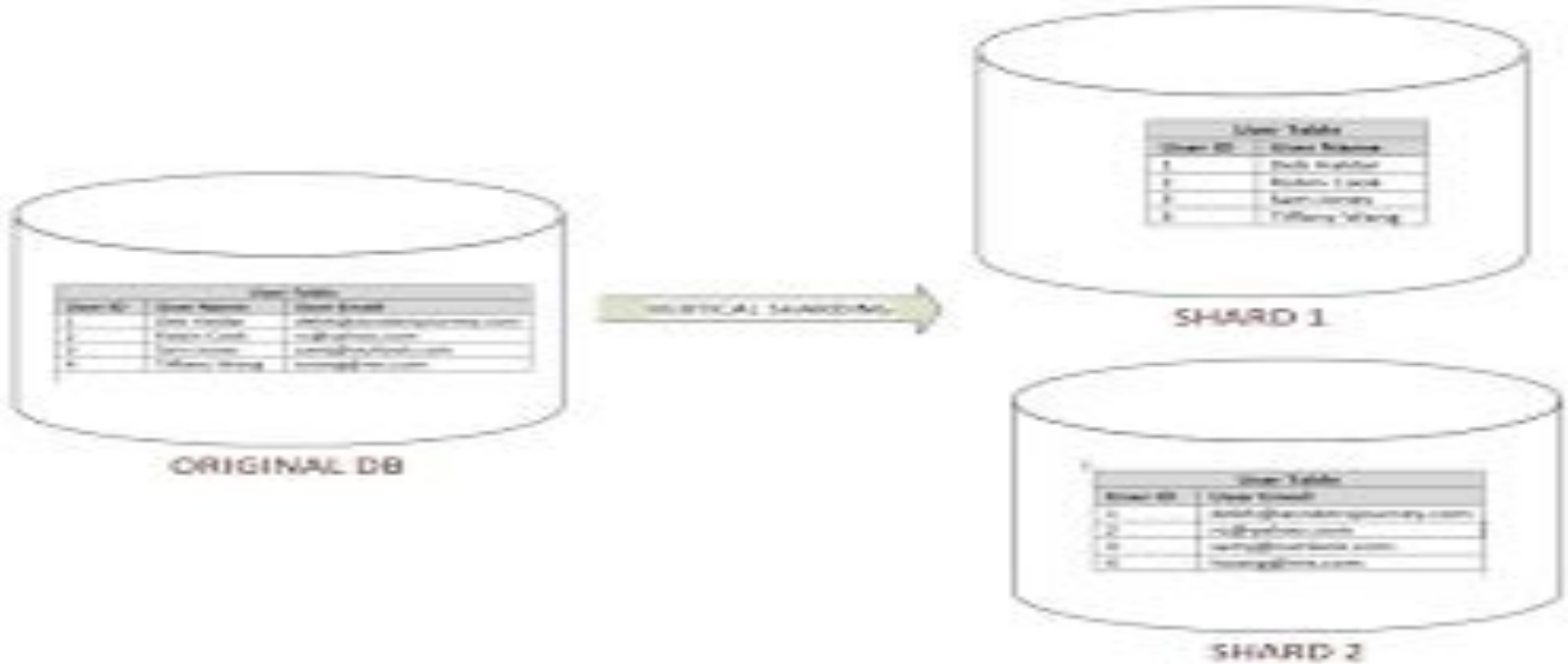
- CA: Implies single site cluster in which all nodes communicate with each other
- CP:Implies all the available data will be consistent or accurate, provided some data may not be available
- AP:Implies some data returned may be inconsistent

ACID Property

- Atomicity: In this either all the operations in transaction will complete or not a single one. If any part of transaction fails ,the entire transaction will fail.
- Consistency: A transaction must leave the database in an inconsistent state.
- This ensures that any transaction which is done will change the database to another valid state.
- Isolation: Transactions will not interfere with each other
- Durability: The successful completion of transaction will not be reversed.
- An alternate to ACID is BASE
- Basic Availability: System does not guarantee availability
- Soft state: System changes over time, even without input
- Eventual consistency: The system will become consistent over time

Sharding

- Database sharding can be defined as a
- partitioning scheme for large database distributed across various servers and is responsible for new levels of database performance and scalability.
- It divides a database into smaller parts called shards and replicates those across a number of distributed servers
- Sharding was coined by Google engineers and gained popularity through their publication of Big Table architecture


ORIGINAL DB
SHARD 1
SHARD 2

- Database sharding has been gaining popularity due to the extensive growth in transactional volume and size of business application databases.
- It is popular among successful e-service providers, software as service companies and social networking websites
- Database sharding employs a scalable approach for improving the throughput and overall performance of high transaction large dataset centric business application
- As the size and transaction volume of database increase in a linear pattern the response time also grows logarithmically

Growth in database transaction and volume