BCSE302L - Database Systems

MODULE - 7

MODULE - 7

Module:7	NOSQL Database Management	3 hours
Introduction, Need of NoSQL, CAP Theorem, different NoSQL data bases: Key-value data stores, Columnar families, Document databases, Graph databases		

NOSQL Databases

NOSQL Databases

- A database is a collection of structured data or information which is stored in a computer system and can be accessed easily. A database is usually managed by a Database Management System (DBMS).
- NoSQL is a non-relational database that is used to store the data in the non-tabular form. NoSQL stands for Not only SQL.

NoSQL Database

- The relational databases were not designed to cope with the scale and agility of modern applications
- NoSQL are type of databases created in the late 90s to solve these problems
- **NoSQL → Not Only SQL**
- NoSQL databases : non-relational, distributed, open-source and horizontally scalable.

- NoSQL stands for
  - No Relational
  - No RDBMS
  - Not Only SQL

RDBMS Characteristics

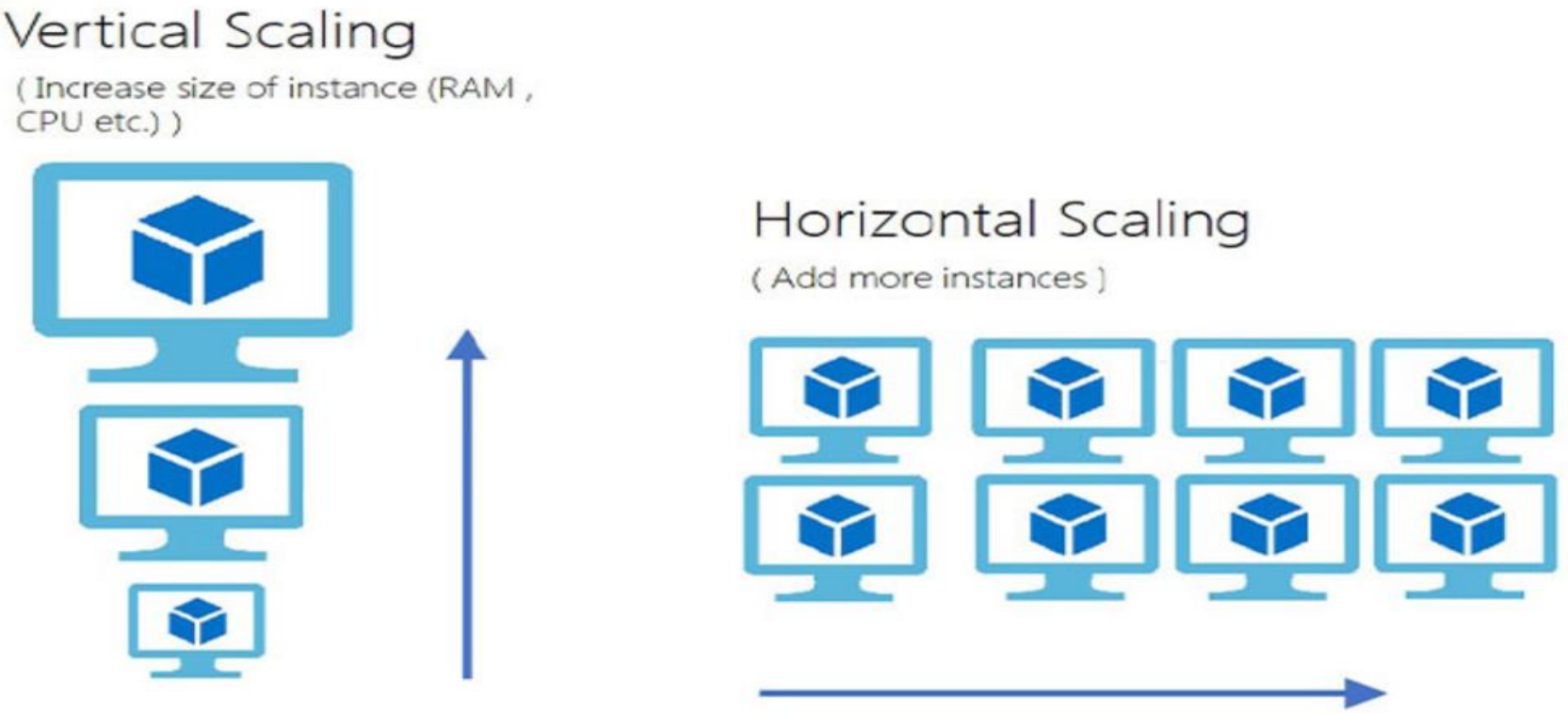
- Data stored in columns and tables
- Relationships represented by data
- Data Manipulation Language
- Data Definition Language
- Transactions
- Abstraction from physical layer
- Applications specify what, not how
- Physical layer can change without modifying applications
  - Create indexes to support queries
  - In Memory databases

Problems with RDBMS

- Should know the entire schema upfront
- Every record should have the same properties (rigid structure)
- Scalability is costly (transactions and joins are expensive when running on a distributed database)
- Many Relational Databases do not provide out of the box support for scaling
- Normalization
- Fixed schemas make it hard to adjust to application needs
- Altering schema on a running database is expensive
- Application changes for any change in schema structure
- SQL was designed for running on single server systems.
- Horizontal Scalability is a problem

NoSQL Database

- **Horizontal scaling** - adding more machines into the pool of resources
- **Vertical scaling** - adding more power (CPU, RAM) to an existing machine .



NOSQL Definition

Next Generation Databases mostly addressing some of the points

- Being non relational
- Distributed
- Open Source
- Horizontally Scalable

NoSQL Database

Features

- Distributed computing system.
- Higher Scalability.
- Reduced Costs.
- Flexible schema design.
- Process unstructured and semi-structured data.
- No complex relationship.
- Open-sourced.

NoSQL Database

- **Advantages:**
  - High scalability
    - Horizontal scaling
      - Partitioning data, placing it on multiple machines
      - Easy to implement
      - Ex: MongoDB,Cassandra
    - Vertical scaling
      - Adding more resources to existing machine
      - Difficult to implement
  - High availability
    - Auto replication feature
    - Data replicates itself to the previous consistent state in case of any failure

NoSQL Database

- **Disadvantages:**
  - Narrow focus
    - Mainly designed for storage
    - Provides little functionality
  - Open source
    - No reliable standard for NoSQL
  - Management challenge
    - Data management in NoSQL is more complex
    - Challenging to install
    - Difficult to manage on the daily basis
  - GUI is not available
  - Backup
    - No approach for backup of data in a consistent manner
  - Large document size

NoSQL Database

- **When should NoSQL be used**
  - When huge amount of data need to be stored and retrieved
  - The relationship between the stored data is not important
  - The data changing over time and is not structured
  - Support of constraints and joins are not required at database level
  - The data is growing continuously and you need to scale the database regular to handle

Schema-less Models

- NoSQL: schema-less model
- Donot have fixed data structure
- Can change the data structure as per our wish
- Supports
  - Donot require normalization
  - Donot require pre-setting of data types
  - Can store data with different characteristics
  - Can be easily transformed
  - More relaxed modelling constraints
  - Automatic distribution of data and elasticity with respect to the use of computing, storage and network bandwidth
  - Integrated data catching
  - Reduces the data access latency
  - Speed up the performance
  - Framework is optimized for different types of analysis
  - Benefit for both application developer and end-user analysts

NoSQL Database

Differences between SQL and NoSQL

- **SQL:**
  - Relational DBMS
  - Structured
  - Vertically scalable
  - Has fixed and pre-defined schema
  - Not suitable for hierarchical data storage
  - Applicable for complex queries

NoSQL Database

- **NoSQL :**
  - Non-relational database
  - Like documents
  - Horizontally scalable
  - Dynamic schema
  - Suitable for hierarchical data storage
  - Not suitable for complex queries

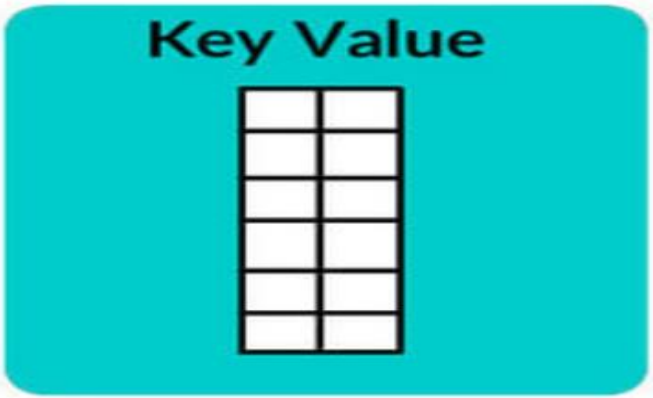
Benefits of NoSQL

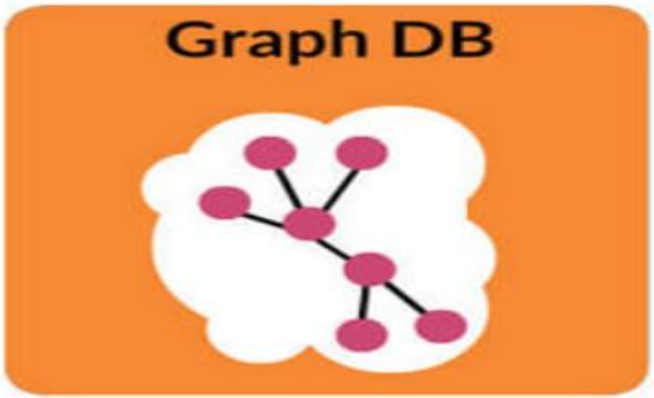
- When compared to relational databases, NoSQL databases are more scalable and provide superior performance, and their data model addresses several issues that the relational model is not designed to address
- Large volumes of rapidly changing structured, semi-structured, and unstructured data:[Schema-less]
- Mostly Open Source
- Object-oriented programming that is easy to use and flexible
- Running well on Clusters-Geographically distributed scale-out architecture instead of expensive, monolithic architecture

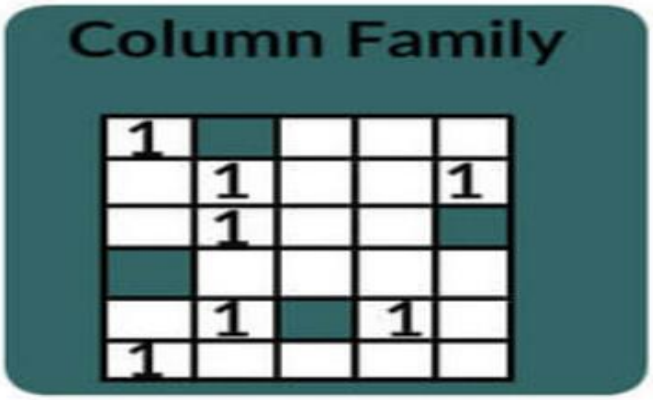
NOSQL Databases

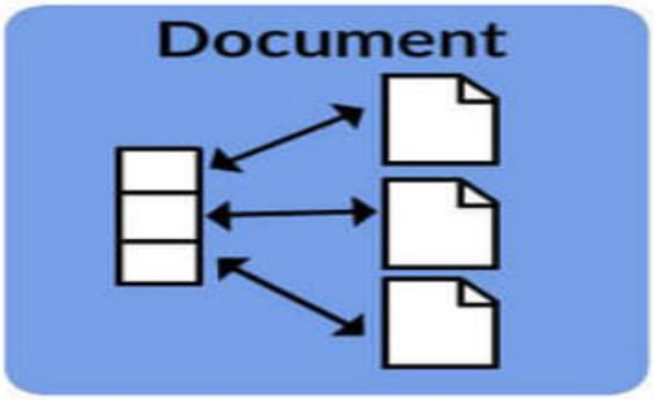
NoSQL Database

- **Types of NoSQL databases**









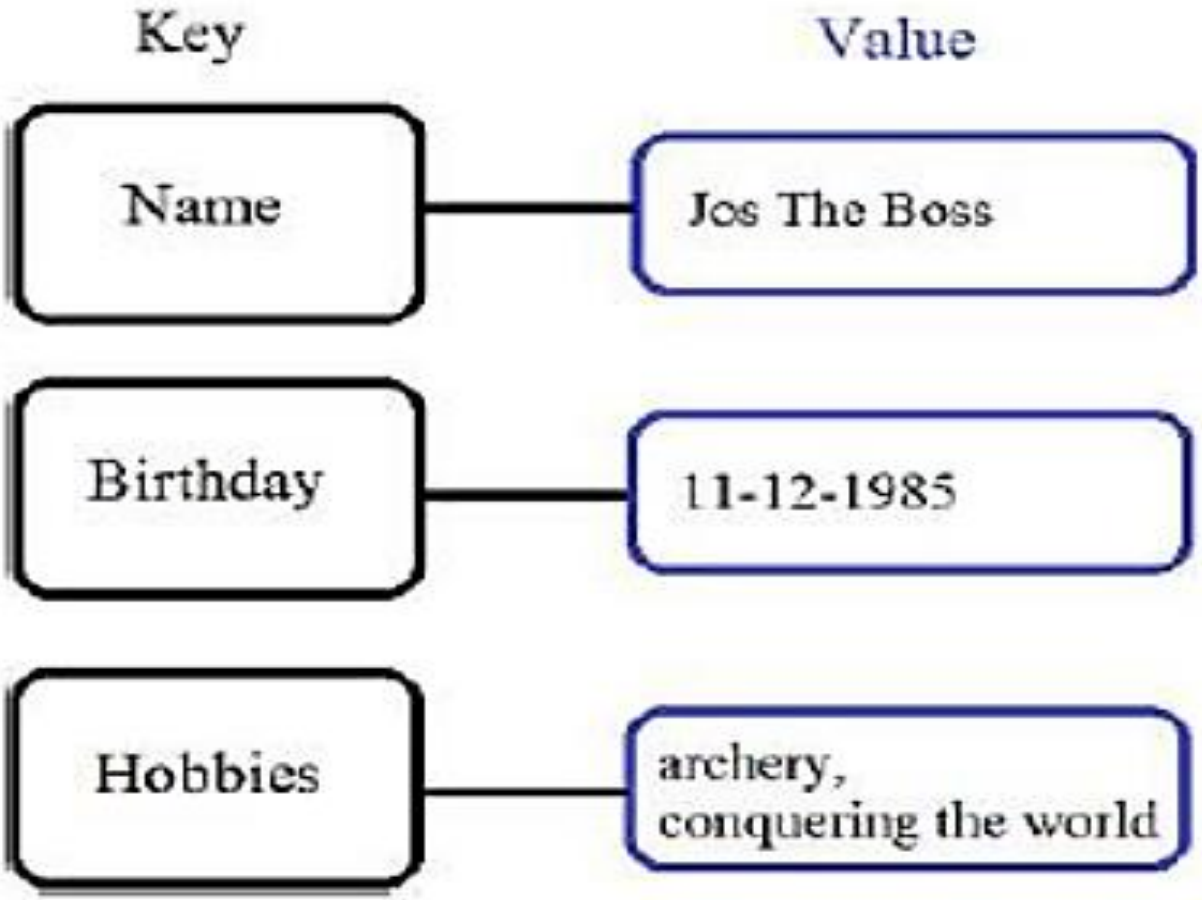
NOSQL Categories

- Most of the NOSQL products can be put into these categories
  1. Key /Value Stores
  2. Document Databases
  3. Graph Databases
  4. Column Databases

NOSQL Categories

- **Key-value stores** are the simplest NoSQL databases.
- Every single item in the database is stored as an attribute name (or 'key'), together with its value.
- Examples of key-value stores are Riak and Berkeley DB.
- Some key-value stores, such as Redis, allow each value to have a type, such as 'integer', which adds functionality.

Key Value Stores



Examples of Key Value Databases

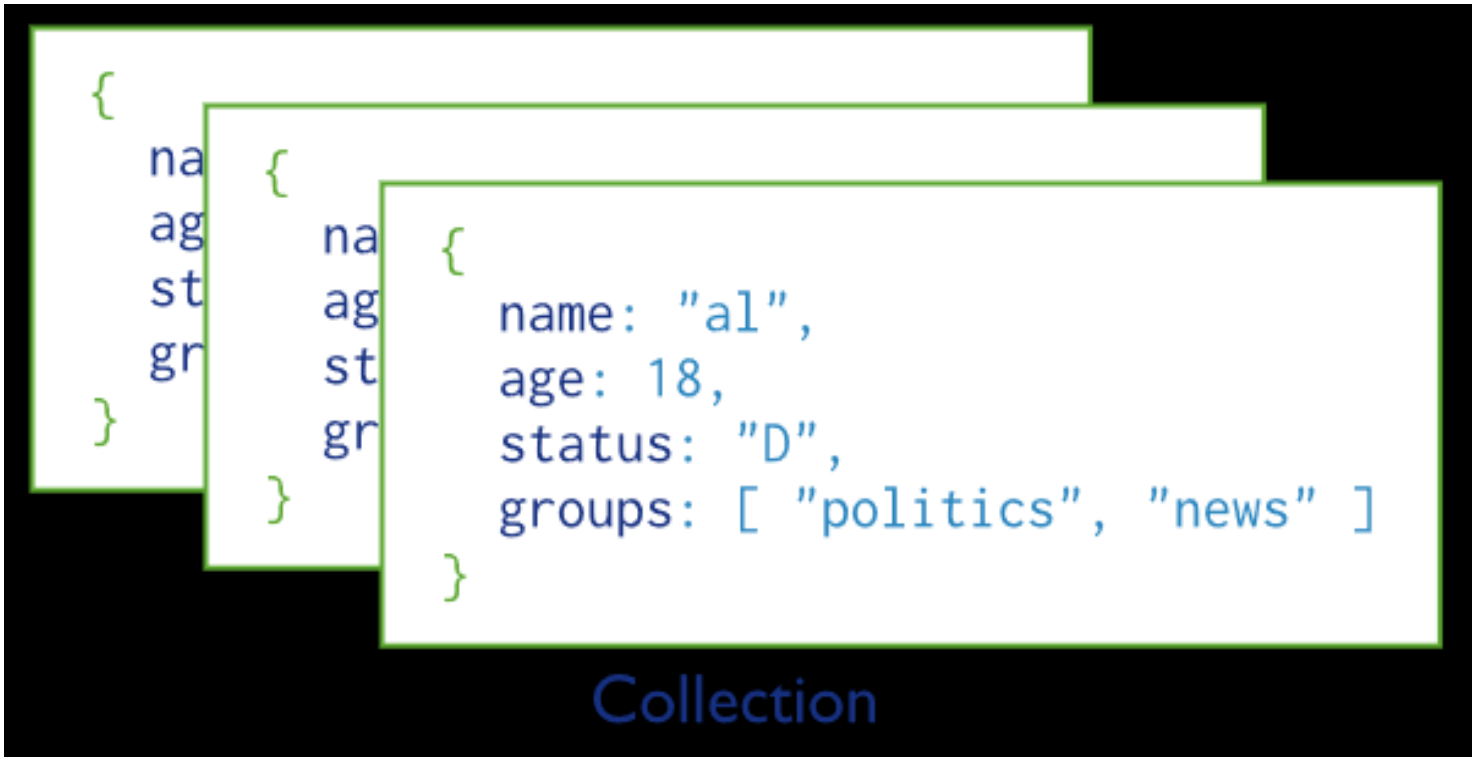
- Redis
- Amazon DynamoDB
- Riak
- ScyllaDB
- Aerospike
- Coughbase
- Amazon ElasticCache

Document Databases

- **Document databases** pair each key with a complex data structure known as a document.
- Documents are composed of field-and-value pairs and have the following structure:
- Documents can contain many different key-value pairs, or key-array pairs, or even nested documents

{ field1: value1, field2: value2, field3: value3, ...
fieldN: valueN }

Document Databases



MongoDB stores BSON documents, i.e. data records, in collections; the collections in databases.

Document Databases

```
{
    _id: ObjectId("51156a1e056d6f966f268f81"),
    type: "Article",
    author: "Derick Rethans",
    title: "Introduction to Document Databases with MongoDB",
    date: ISODate("2013-04-24T16:26:31.911Z"),
    body: "This arti…"
},
{
    _id: ObjectId("51156a1e056d6f966f268f82"),
    type: "Book",
    author: "Derick Rethans",
    title: "php|architect's Guide to Date and Time Programming with PHP",
    isbn: "978-0-9738621-5-7"
}
```

Examples of Document Databases

- MongoDB – Mongo DB inc
- Cosmos DB – Microsoft
- Couchbase
- ArangoDB
- Google cloud firestore
- MongoDB Atlas

NOSQL Categories

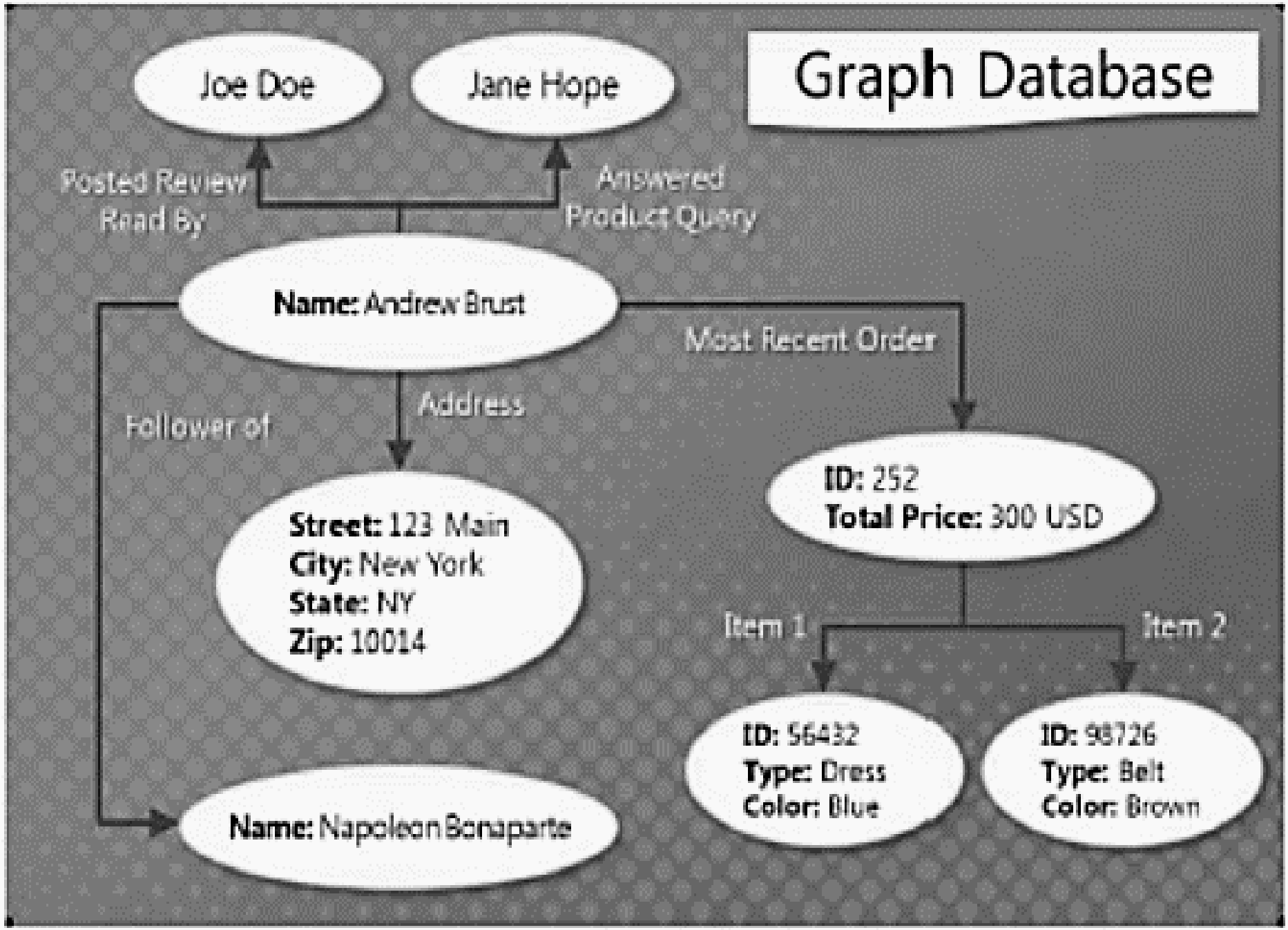
- **Graph stores** are used to store information about networks of data, such as social connections. Graph stores include Neo4J and Giraph.
- **Wide-column stores** such as Cassandra and HBase are optimized for queries over large datasets, and store columns of data together, instead of rows.

Graph Databases

- **Graph stores** are used to store information about networks of data, such as **social connections**.
- **Graph databases** are NoSQL **databases** which use the **graph** data model comprised of vertices, which is an entity such as a person, place, object or relevant piece of data and edges, which represent the relationship between two nodes.

Graph Databases

- Graph-oriented
- Everything is stored as an edge, a node or an attribute.
- Each node and edge can have any number of attributes.
- Both the nodes and edges can be labelled.
- Labels can be used to narrow searches.


Graph Database
Joe Doe
Jane Hope
Posted Review
Read By
Answered
Product Query
Name: Andrew Brust
Most Recent Order
Address
Follower of
Street: 123 Main
City: New York
State: NY
Zip: 10014
ID: 252
Total Price: 300 USD
Item 1
Item 2
ID: 56432
Type: Dress
Color: Blue
ID: 98726
Type: Belt
Color: Brown
Name: Napoleon Bonaparte

Graph Databases - Advantages

- Easy to represent connected data
- Very faster to retrieve, navigate and traverse connected data
- Can represent semi-structured data easily
- Not require complex or costly joins to retrieve connected data
- It supports full ACID (Atomicity, Consistency, Isolation and Durability) rules

Emp ID	Name	Salary	DNO
112	San	1000	1
113	Siva	2000	2

DID	DName
1	SCOPE
2	SBST

{id:112 Name: San Salary:1000 Dno:1}

{Did:1 Dname: SCOPE}

{id:113 Name: Siva Salary:2000 Dno:2}

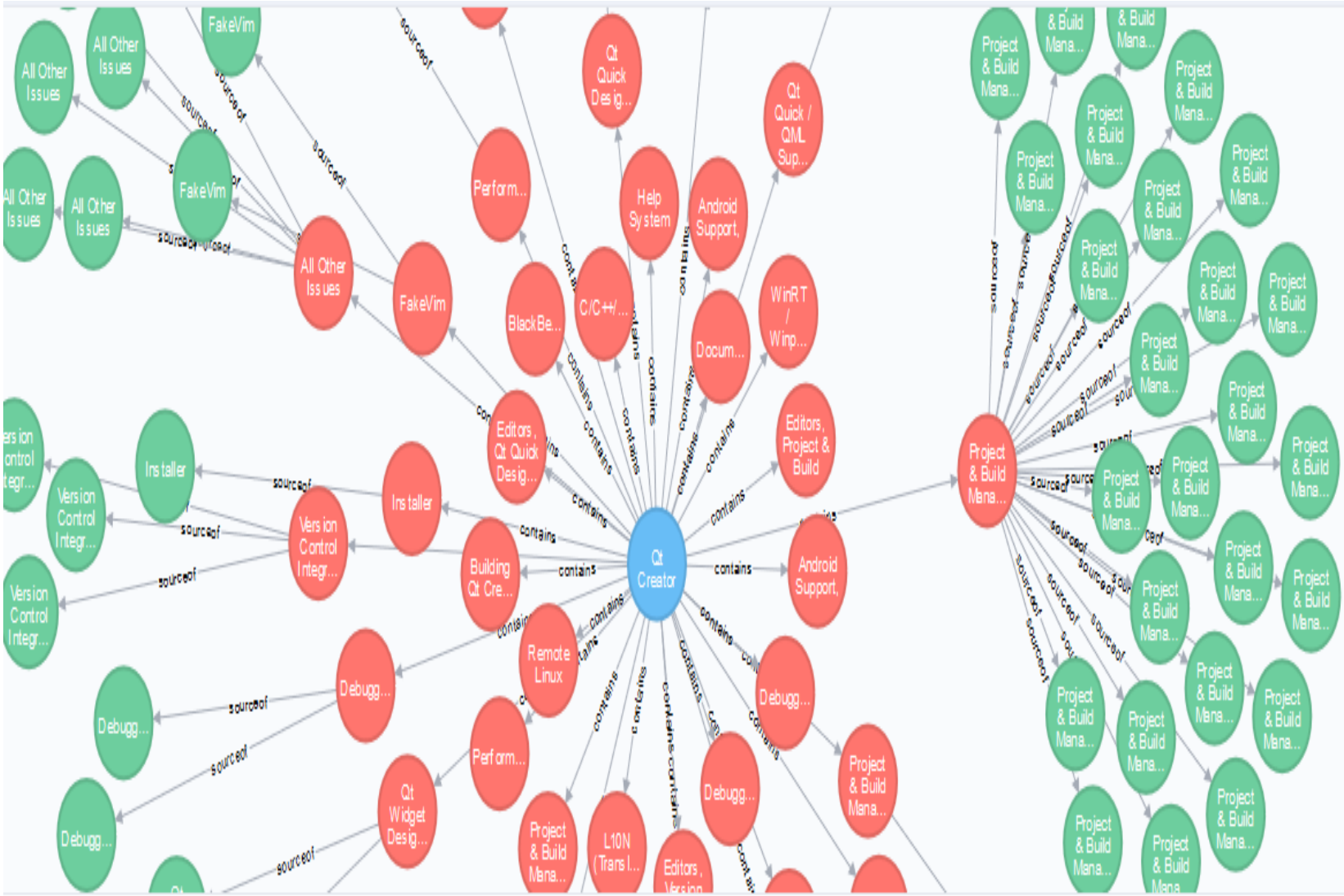
{Did:2 Dname: SBST}

Lets Convert a Relational Model to a Graph Data Model using an Example

Graph Databases

- Neo4j Graph Database
- ArangoDB
- OrientDB
- Dgraph
- Amazon Neptune
- DataStax

Sample Nodes from Neo4j



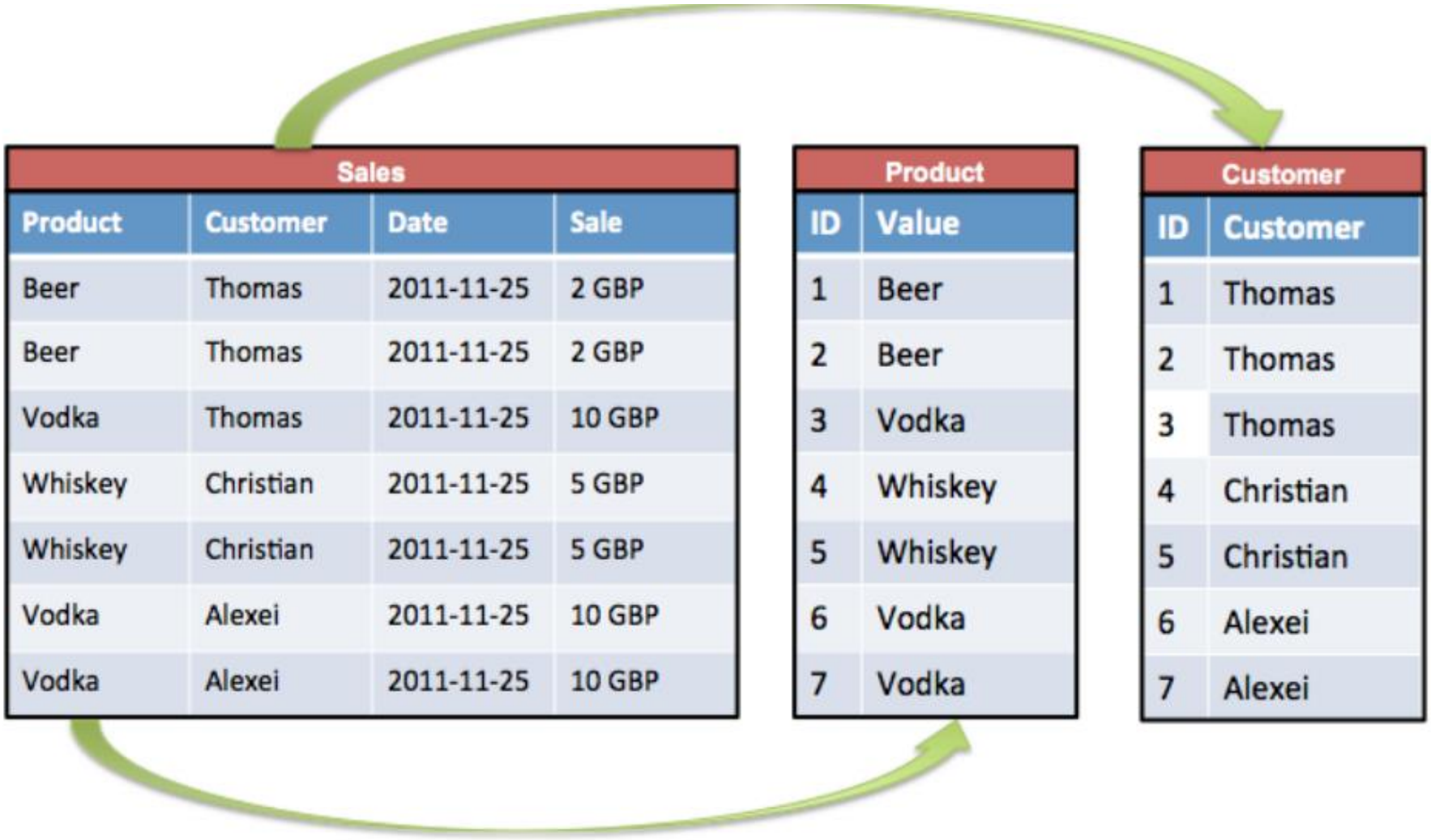
Column Family Data Model

- Column family databases are probably most known because of Google's BigTable implementation.
- They are very similar on the surface to relational database, but they are actually quite different beast.
- Some of the **difference is storing data by rows (relational) vs. storing data by columns (column family databases).**
- But a lot of the difference is conceptual in nature. You can't apply the same sort of solutions that you used in a relational form to a column database.

Sample

Row ID	Registration Number	Name	Mark
1	11BCE0001	Sam	4
2	11BCE0002	Satish	5
3	11BCE0003	Ram	3
4	11BCE0004	Tom	4
5	11BCE0005	Jeff	5
6	11BCE0006	Chris	2
7	11BCE0007	David	1

Sample



Sales

Product	Customer	Date	Sale
Beer	Thomas	2011-11-25	2 GBP
Beer	Thomas	2011-11-25	2 GBP
Vodka	Thomas	2011-11-25	10 GBP
Whiskey	Christian	2011-11-25	5 GBP
Whiskey	Christian	2011-11-25	5 GBP
Vodka	Alexei	2011-11-25	10 GBP
Vodka	Alexei	2011-11-25	10 GBP

Product

ID	Value
1	Beer
2	Beer
3	Vodka
4	Whiskey
5	Whiskey
6	Vodka
7	Vodka

Customer

ID	Customer
1	Thomas
2	Thomas
3	Thomas
4	Christian
5	Christian
6	Alexei
7	Alexei

Column Family Data Stores Example

- Bigtable
- Cassandra
- HBase
- Vertica
- Druid
- Accumulo
- Hypertable

NoSQL, No ACID

- RDBMSs are based on ACID (Atomicity, Consistency, Isolation, and Durability) properties
- NoSQL does not give importance to ACID properties. In some cases completely ignores them
- In distributed parallel systems it is difficult/impossible to ensure ACID properties
- Long-running transactions don't work because keeping resources blocked for a long time is not practical

BASE Property of Transaction

- Acronym contrived to be the opposite of ACID
  - Basically Available - Failure will not halt the system
  - Soft state - State of the system will change over time
  - Eventually Consistent - Will become consistent over time

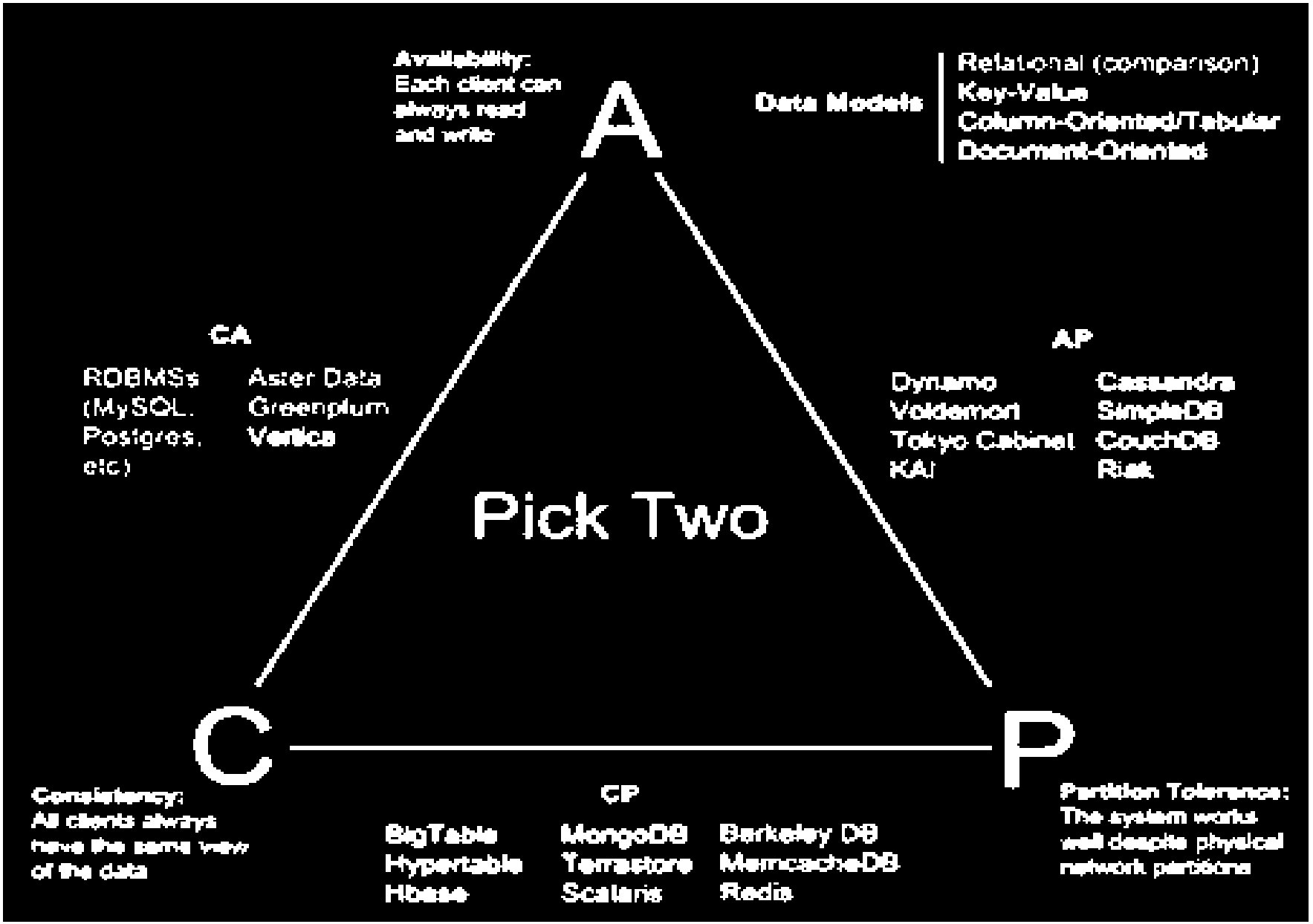
CAP Theorem

- A congruent and logical way for assessing the problems involved in assuring ACID-like guarantees in distributed systems is provided by the CAP theorem
- At most **two of the following three** can be **maximized at one time**
  - **Consistency**
    - Each client has the same view of the data
  - **Availability**
    - Each client can always read and write
  - **Partition tolerance**
    - System works well across distributed physical networks

CAP Theorem

- Eric Brewer's CAP theorem says that if you want consistency, availability, and partition tolerance, you have to settle for two out of three.
- For a distributed system, *partition tolerance* means the system will continue to work unless there is a total network failure.
- A few nodes can fail and the system keeps going.

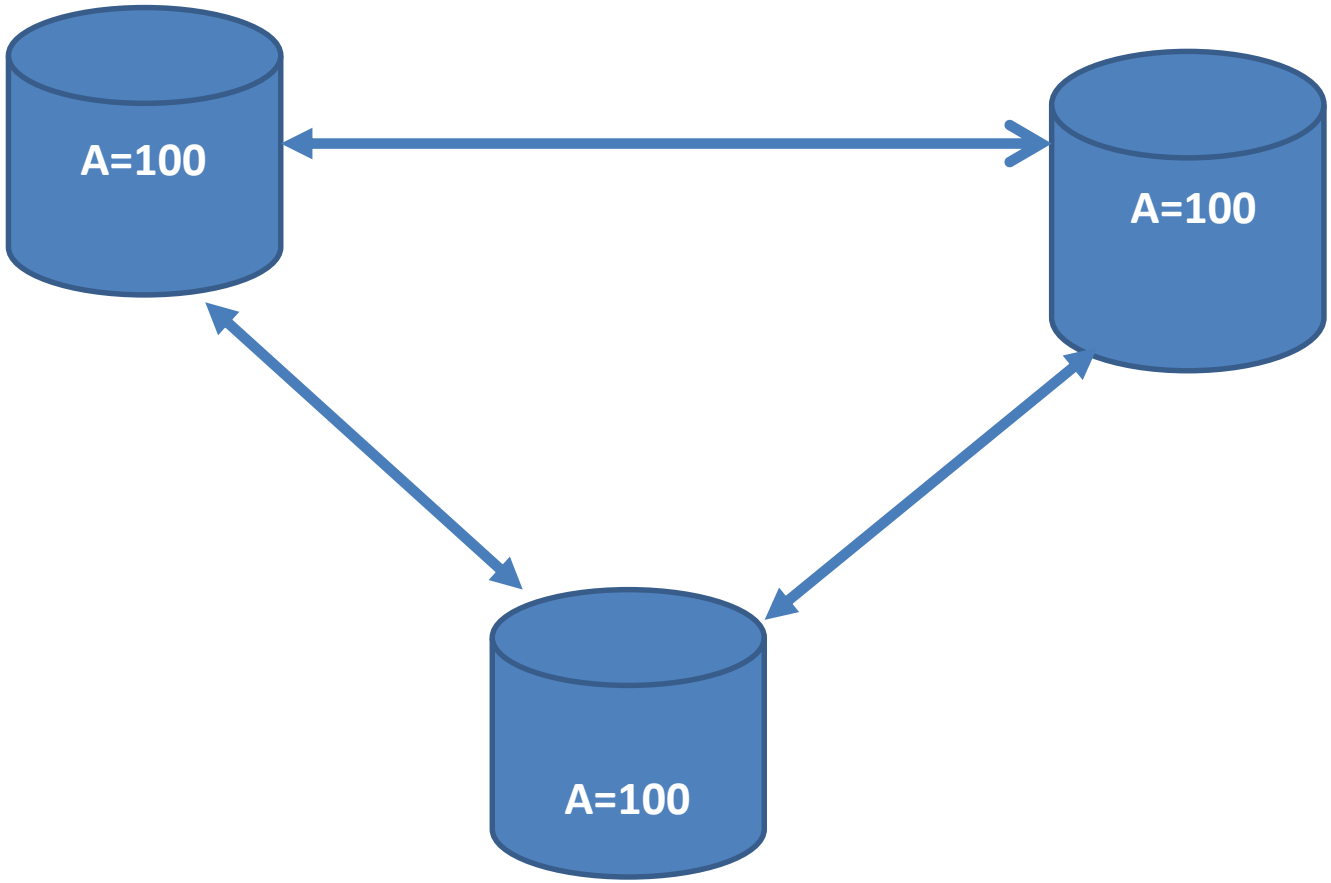
CAP Theorem


Availability:
Each client can always read and write
A
Data Models
Relational (comparison)
Key-Value
Column-Oriented/Tabular
Document-Oriented
CA
RDBMSs (MySQL, Postgres, etc)
Aster Data
Greenplum
Vertica
AP
Dynamo
Voldemort
Tokyo Cabinet
KAI
Cassandra
SimpleDB
CouchDB
Riak
Pick Two
C
P
Consistency:
All clients always have the same view of the data
CP
BigTable
Hypertable
Hbase
MongoDB
Terrastore
Scalaris
Berkeley DB
MemcacheDB
Redis
Partition Tolerance:
The system works well despite physical network partitions

According to CAPS Theorem a Distributed System can guarantee only two of the three properties

- Consistency
- availability
- Partition tolerance

Guaranteeing Consistency – User cant access the data until all servers are in sync
Guaranteeing Availability – User will not get the updated data all the time
Guaranteeing Partition Tolerance – Even when one site Fails other sites will support the user



NoSQL

Where would I use a NoSQL database?

- Do you have somewhere a large set of uncontrolled, unstructured, data that you are trying to fit into a RDBMS?
  - Log Analysis
  - Social Networking Feeds (many firms hooked in through Facebook or Twitter)
  - External feeds from partners
  - Data that is not easily analyzed in a RDBMS such as time-based data
  - Large data feeds that need to be massaged before entry into an RDBMS

NoSQL

Performance

- There is no perfect NoSQL database
- Every database has its advantages and disadvantages
  - Depending on the type of tasks (and preferences) to accomplish
- NoSQL is a set of concepts, ideas, technologies, and software dealing with
  - Big data
  - Sparse un/semi-structured data
  - High horizontal scalability
  - Massive parallel processing
- Different applications, goals, targets, approaches need different NoSQL solutions

NoSQL

Cassandra:

- Cassandra is a highly scalable NoSQL database designed to handle large amounts of data across multiple commodity servers.
- It is optimized for write-intensive workloads and can handle structured data efficiently.
- However, Cassandra is not as well-suited for handling complex queries as some other NoSQL databases, such as MongoDB.

HBase:

- HBase is an open-source, distributed NoSQL database that is designed to handle large amounts of structured data.
- It is built on top of Hadoop and is optimized for handling big data workloads.
- HBase can be used for storing customer records or product catalogs, but it may not be the best choice for use cases where the data structure is likely to change frequently.

NoSQL

MongoDB:

- MongoDB is a popular document-based NoSQL database that is well-suited for **handling large volumes of structured data**.
- It is highly scalable and provides rich querying capabilities that make it easy to query and analyze data.
- MongoDB can handle structured data efficiently and can also handle unstructured or semi-structured data.
- It is often used for **customer record management, product catalogs, and other use cases where structured data is the primary focus**.

Couchbase:

- Couchbase is a distributed NoSQL database that is designed for high performance and scalability.
- It is optimized for handling unstructured and semi-structured data, and it can also handle structured data efficiently.
- While Couchbase can be used for storing customer records or product catalogs, it may not be the best choice for use cases where the data structure is likely to change frequently, as it can be challenging to change the schema of a Couchbase database once it has been deployed.