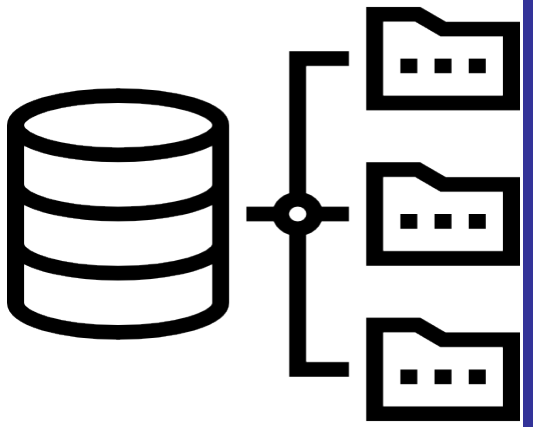
Advanced Database- IS411

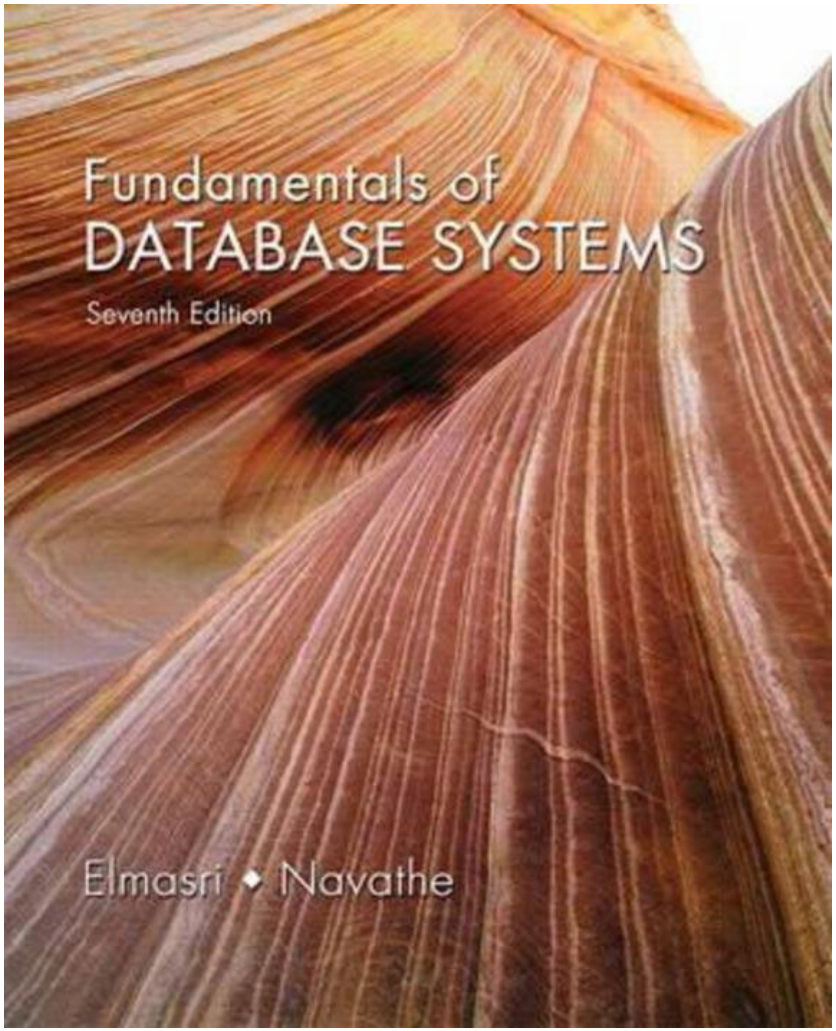


Introduced by

Dr. Ebtsam Adel

Lecturer of Information Systems,
Information Systems department,
Faculty of computers and information,
Damanhour university

Materials




QUIZ!

1. Which of the following properties does a transaction not have?

a. Atomicity
b. Consistency
c. Isolation
d. Scalability

2. What does the 'Durability' property ensure in a transaction?

a. Transactions are completed without errors
b. Once a transaction is committed, it remains so even in case of a failure
c. Transactions can be rolled back
d. Transactions are isolated from each other

3. **Which property of transactions guarantees that concurrent transactions do not interfere with each other?**

a. Atomicity
b. Consistency
c. Isolation
d. Durability

4. **What is the primary purpose of the commit operation in a transaction?**

a. To start a transaction
b. To save changes made during the transaction
c. To undo changes made
d. To check for errors in the transaction

Topics

- ✓ **Chapter 20** Introduction to Transaction Processing Concepts and Theory
- ✓ **chapter 24** NOSQL Databases and Big Data Storage Systems
- ✓ **chapter 25** Big Data Technologies Based on MapReduce and Hadoop
- ✓ **chapter 27** Introduction to Information Retrieval and Web Search
- ✓ **chapter 29** Overview of Data Warehousing and OLAP
- ✓ **chapter 30** Database Security

CHAPTER 24

NOSQL Databases and Big Data Storage Systems

Data types

- Data could be presented in three different types of data: **structured**, semi-structured, and unstructured data.
- Structured data as (database), semi-structured data (XML files, JSON documents), and unstructured data such as (documents, video, e-mails, and reports).

Unstructured data

The university has 5600 students.
John's ID is number 1, he is 18 years old and already holds a B.Sc. degree.
David's ID is number 2, he is 31 years old and holds a Ph.D. degree. Robert's ID is number 3, he is 51 years old and also holds the same degree as David, a Ph.D. degree.

Semi-structured data

```
<University>
 <Student ID="1">
  <Name>John</Name>
  <Age>18</Age>
  <Degree>B.Sc.</Degree>
 </Student>
 <Student ID="2">
  <Name>David</Name>
  <Age>31</Age>
  <Degree>Ph.D. </Degree>
 </Student>
....
</University>
```

Structured data

ID	Name	Age	Degree
1	John	18	B.Sc.
2	David	31	Ph.D.
3	Robert	51	Ph.D.
4	Rick	26	M.Sc.
5	Michael	19	B.Sc.

Introduction

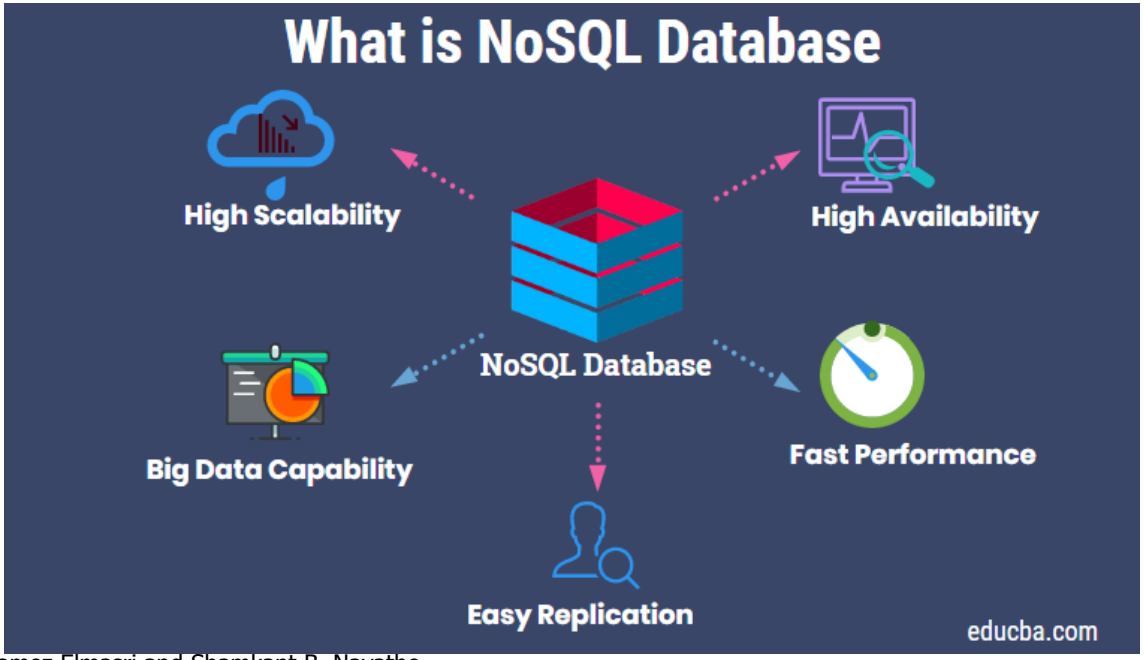
- **Most SQL databases** are relational. Relational Databases are tabular and have a pre-defined schema that organizes the data logically.

- **NoSQL** is a non-relational database management system for certain data models. These data models don't need a schema and are scalable.

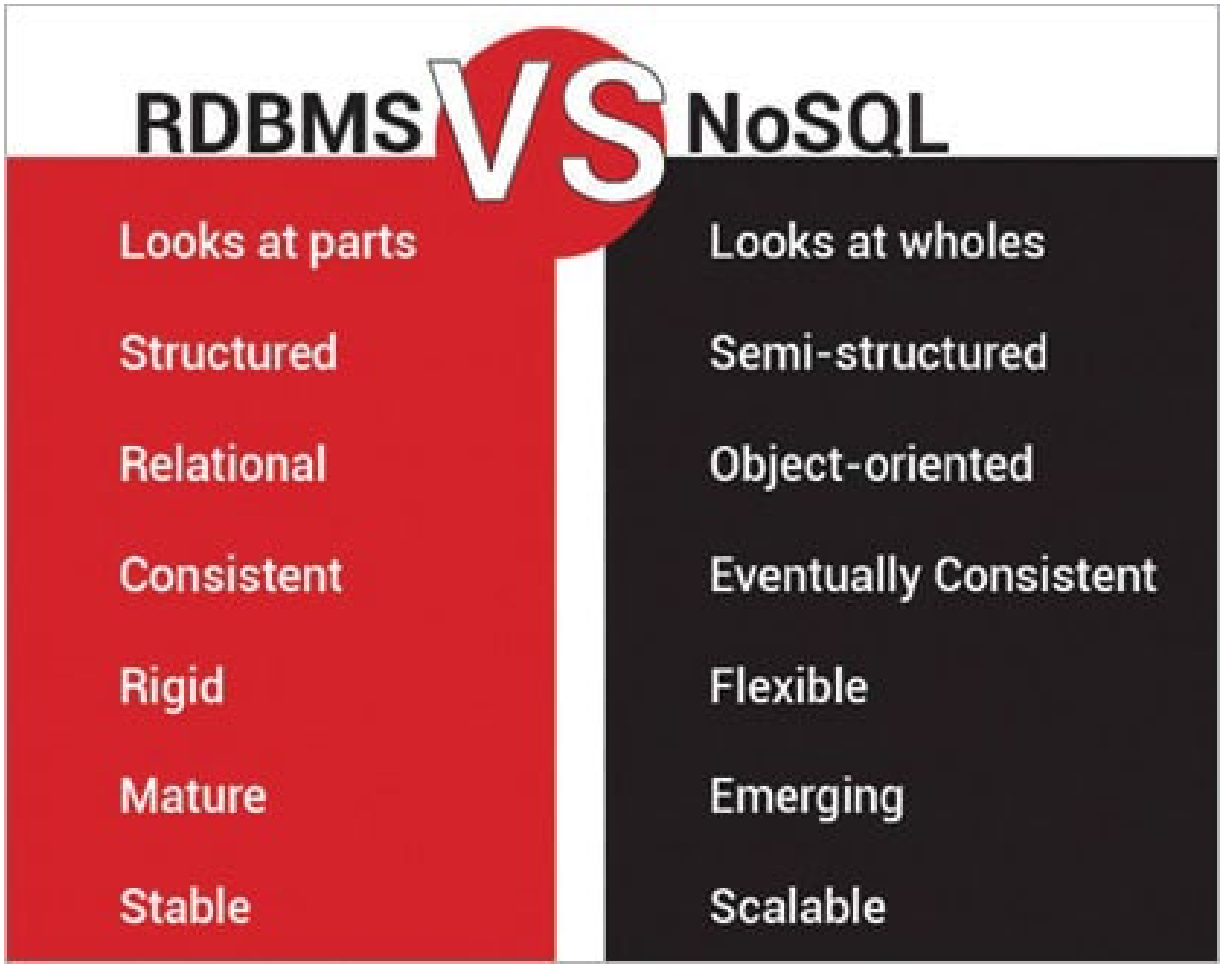
- **NoSQL databases** don't require a certain schema. They prioritize speed and flexibility in data storage. Amazon, Facebook, and Google.

Introduction

- NOSQL: Not only SQL
- Most NOSQL systems are distributed databases or distributed storage systems.
  - Focus on semi-structured data storage, high performance, **availability**, **data replication**, and **scalability**.



Introduction

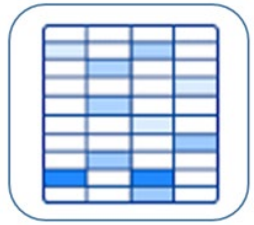


Introduction (cont'd.)

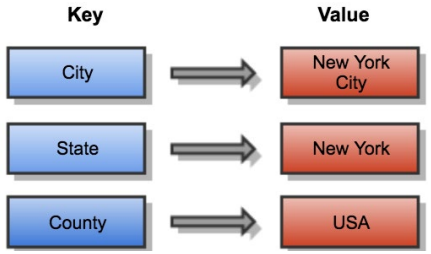
- NOSQL systems focus on storage of "big data".
- Typical applications that use NOSQL
  - Social media
  - Web links
  - User profiles
  - Marketing and sales
  - Posts and tweets
  - Road maps and spatial data
  - Email
  - ……..

24.1 Introduction to NOSQL Systems

- BigTable
  - Google's proprietary NOSQL system
  - Column-based or wide column store
- DynamoDB (Amazon)
  - Key-value data store
- Cassandra (Facebook)
  - Uses concepts **from both** key-value store and column-based systems.



Wide-Column Store



Introduction to NOSQL Systems (cont'd.)

- MongoDB and CouchDB
  - Document stores
- Neo4J and GraphBase
  - Graph-based NOSQL systems
- OrientDB
  - Combines several concepts
- Database systems classified on the object model
  - Or native XML model

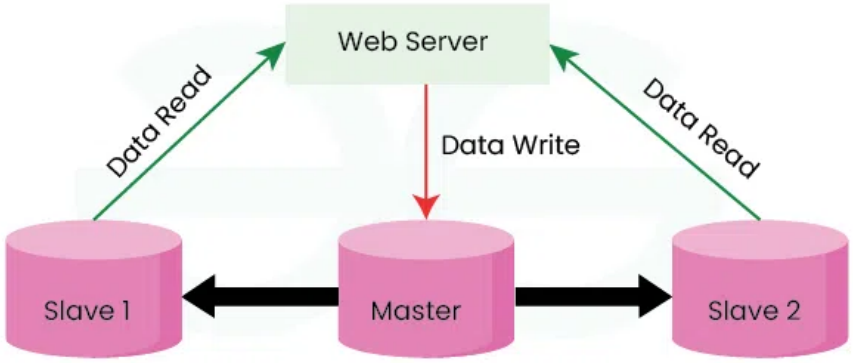
NOSQL characteristics related to *distributed databases*

- NOSQL characteristics related to **distributed databases** and distributed systems
  - Scalability
  - Availability, replication, and eventual consistency
  - Replication models
    - Master-slave
    - Master-master
  - Sharding of files
  - High performance data access

Master-Slave Replication

- The process of copying and synchronizing data from a primary database (the **master**) to one or more secondary databases (the **slaves**) is known as master-slave replication.

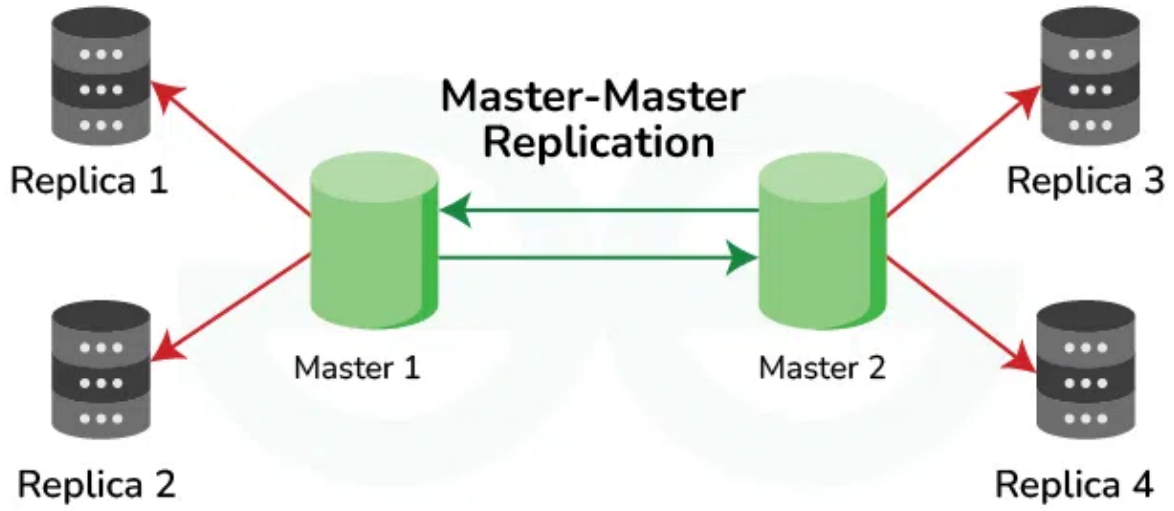
- In this configuration, all write operations, including inserts, updates, and deletions, must be received by the master database.
- After modifications are made to the master database, a copy of the data is kept in the slave databases.



Master-Slave Replication

Master-Master Replication

- is a configuration where two or more databases are set up as master databases, each of which is able to accept write operations.
- In other words, any modifications made to one master database are reflected in all other master databases within the setup.



Master-Master Replication

NOSQL characteristics related to *data models* and *query languages*

NOSQL characteristics related to **data models** and **query languages**

- ✓ Less powerful query languages
- ✓ Versioning
- ✓ Schema not required

Enter MySQL Query:

```
1 SELECT Type FROM Places
2 WHERE Type IN('Type1','Type 2')
3 ORDER BY Type;
```

MongoDB Syntax:

```
1 db.Places.find({
2     "Type": {
3         "$in": ["Type1", "Type 2"]
4     }
5 }, {
6     "Type": 1
7 }).sort({
8     "Type": 1
9 });
```

Introduction to NOSQL Systems (cont'd.)

- **Versioning**: multiple versions of the data items, with the timestamps of when the data version was created.

Introduction to NOSQL Systems (cont'd.)

Categories of NOSQL systems

- Document-based NOSQL systems
- NOSQL key-value stores
- Column-based or wide column NOSQL systems
- Graph-based NOSQL systems
- Hybrid NOSQL systems
- Object databases
- XML databases

The CAP Theorem

24.2 The CAP Theorem

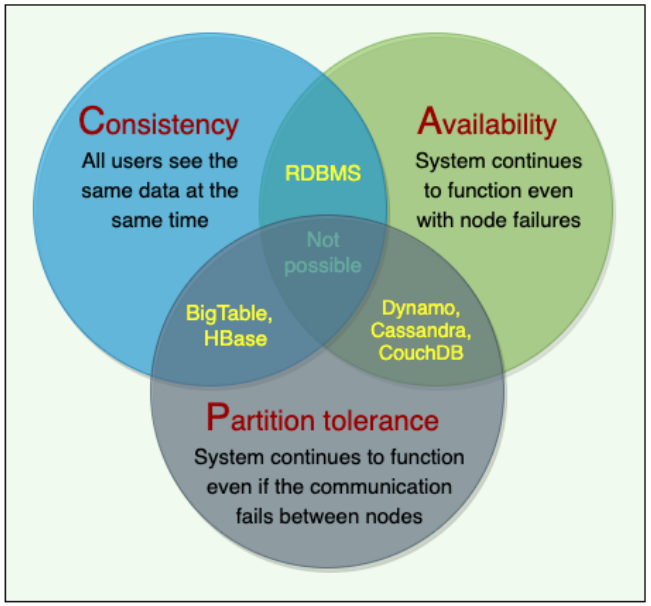
- **Consistency** means that the nodes will have the same copies of a replicated data item visible for various transactions.
- **Availability** means that each read or write request for a data item will either be processed successfully or will receive a message that the operation cannot be completed.
- **partition tolerance** (in the face of the nodes in the system being partitioned by a network fault).

24.2 The CAP Theorem

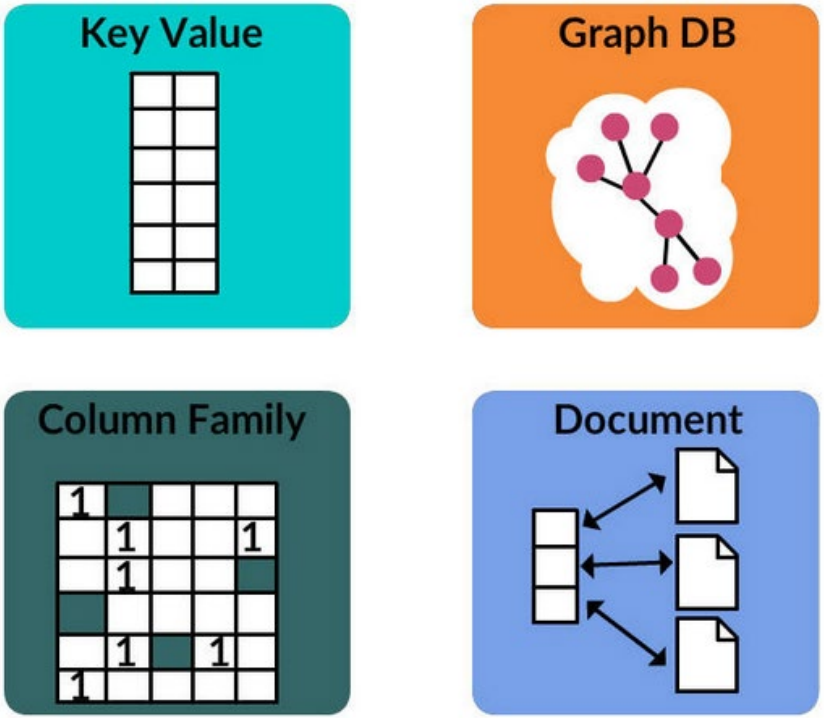
- Various levels of consistency among replicated data items
  - Enforcing serializabilty the strongest form of consistency
    - High overhead – can reduce read/write operation performance
- CAP theorem
  - Consistency, availability, and partition tolerance
  - Not possible to guarantee all three simultaneously
    - In distributed system with data replication

The CAP Theorem (cont'd.)

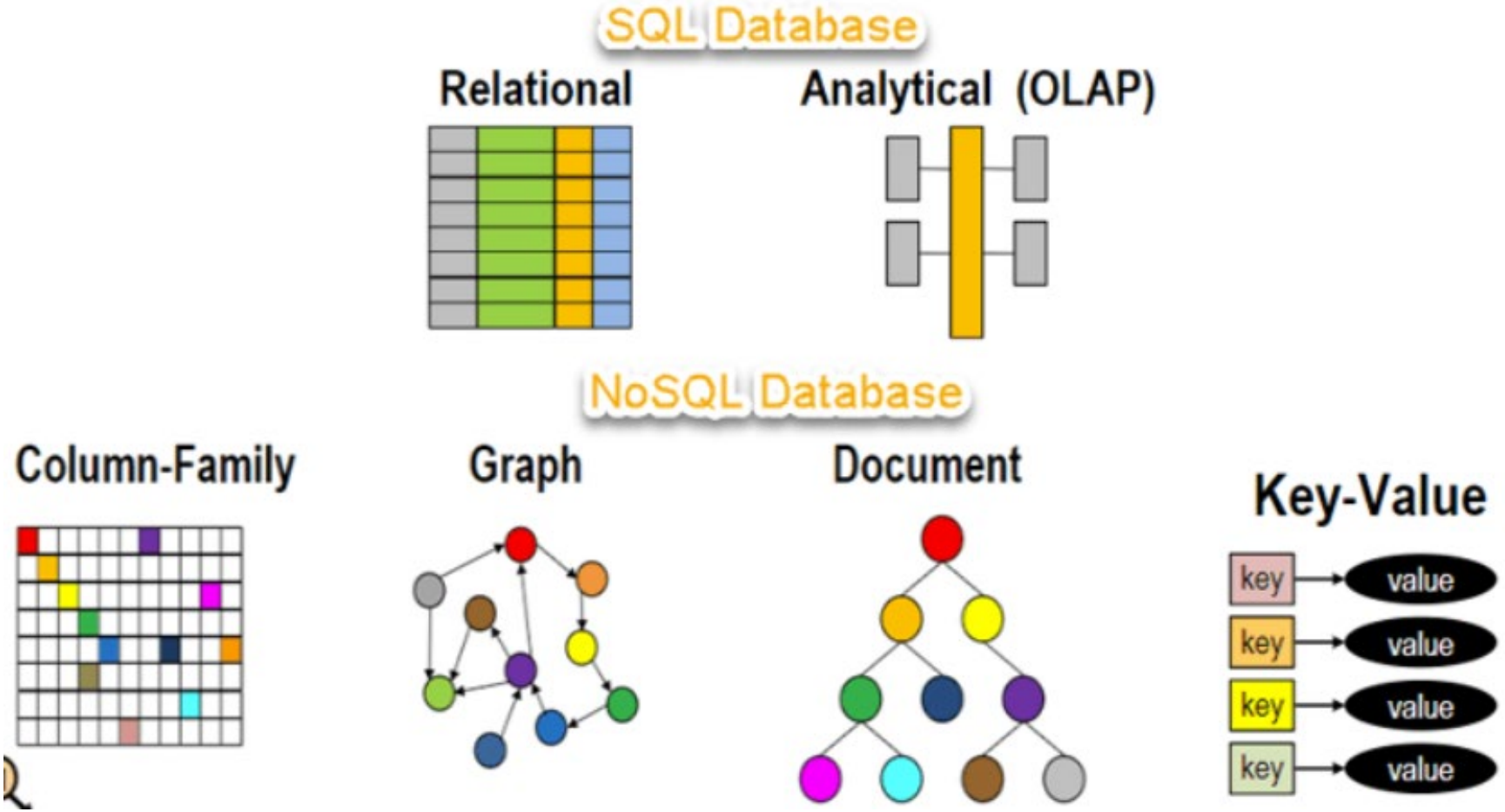
- Designer can choose two of three to guarantee
  - Weaker consistency level is often acceptable in NOSQL distributed data store
  - Guaranteeing availability and partition tolerance more important
  - Eventual consistency often adopted.



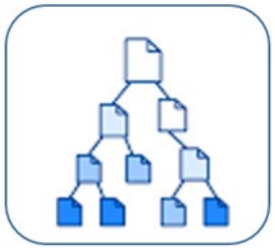
Types of NoSQL Databases



Types of NoSQL Databases



Document-Based NOSQL



Document
Store

24.3 Document-Based NOSQL Systems and MongoDB

- Document stores
  - Collections of similar documents
- Individual documents resemble complex objects or XML documents
  - Documents are self-describing
  - Can have different data elements
- Documents can be specified in various formats
  - XML
  - **JSON**
  - **BSON**

MongoDB Data Model

- Documents stored in binary JSON (BSON) format
- Individual documents stored in a collection
- Example command
  - First parameter specifies name of the collection
  - Collection options include limits on size and number of documents

```
db.createCollection("project", { capped : true, size : 1310720, max : 500 } )
```

- Each document in collection has unique ObjectID field called **_id**

MongoDB Data Model (cont'd.)

- A collection does not have a schema
  - Structure of the data fields in documents chosen based on how documents will be accessed
  - User can choose normalized or denormalized design
- Document creation using insert operation

```
db.<collection_name>.insert(<document(s)>)
```

- Document deletion using remove operation

```
db.<collection_name>.remove(<condition>)
```

MongoDB Distributed Systems Characteristics

- **Two-phase commit** method
  - Used to ensure atomicity and consistency of multidocument transactions
- Replication in MongoDB
  - Concept of replica set to create multiple copies on different nodes
  - Variation of master-slave approach
  - Primary copy, secondary copy, and arbiter
    - Arbiter participates in elections to select new primary if needed

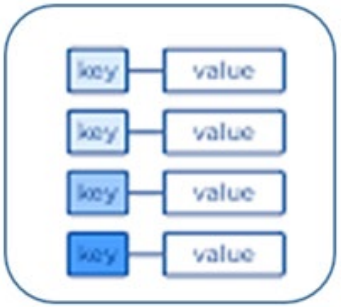
MongoDB Distributed Systems Characteristics (cont'd.)

- **Replication** in MongoDB (cont'd.)
  - All write operations applied to the primary copy and propagated to the secondaries
  - User can choose read preference
    - Read requests can be processed at any replica
- **Sharding** in MongoDB
  - Horizontal partitioning divides the documents into disjoint partitions (shards)
  - Allows adding more nodes as needed
  - Shards stored on different nodes to achieve load balancing

MongoDB Distributed Systems Characteristics (cont'd.)

- Sharding in MongoDB (cont'd.)
  - Partitioning field (shard key) must exist in every document in the collection
    - Must have an index
  - Range partitioning
    - Creates chunks by specifying a range of key values
    - Works best with range queries
  - Hash partitioning
    - Partitioning based on the hash values of each shard key

NOSQL Key-Value



Key-Value
Store

24.4 NOSQL Key-Value Stores

- Key-value stores focus on high performance, availability, and scalability.
  - Can store structured, unstructured, or semi-structured data
- **Key**: unique identifier associated with a data item
  - Used for fast retrieval
- **Value**: the data item itself
  - Can be string or array of bytes
  - Application interprets the structure
- Not support complex queries.

DynamoDB Overview

- DynamoDB part of Amazon's Web Services/SDK platforms
- Table holds a collection of self-describing items
- Item consists of attribute-value pairs
  - Attribute values can be single or multi-valued
- Primary key used to locate items within a table
  - Can be single attribute or pair of attributes

Voldemort Key-Value Distributed Data Store

- Voldemort: open source key-value system similar to DynamoDB
- Voldemort features
  - Simple basic operations (get, put, and delete)
  - High-level formatted data values
  - Consistent **hashing** for distributing (key, value) pairs
  - Consistency and versioning
    - Concurrent writes allowed
    - Each write associated with a vector clock

Examples of Other Key-Value Stores

- Oracle key-value store
  - Oracle NOSQL Database

Ex. Redis

- Caches data in main memory to improve performance
  - Offers master-slave replication and high availability
  - Offers persistence by backing up cache to disk.

Column-Based NoSQL databases

Wide-Column Store

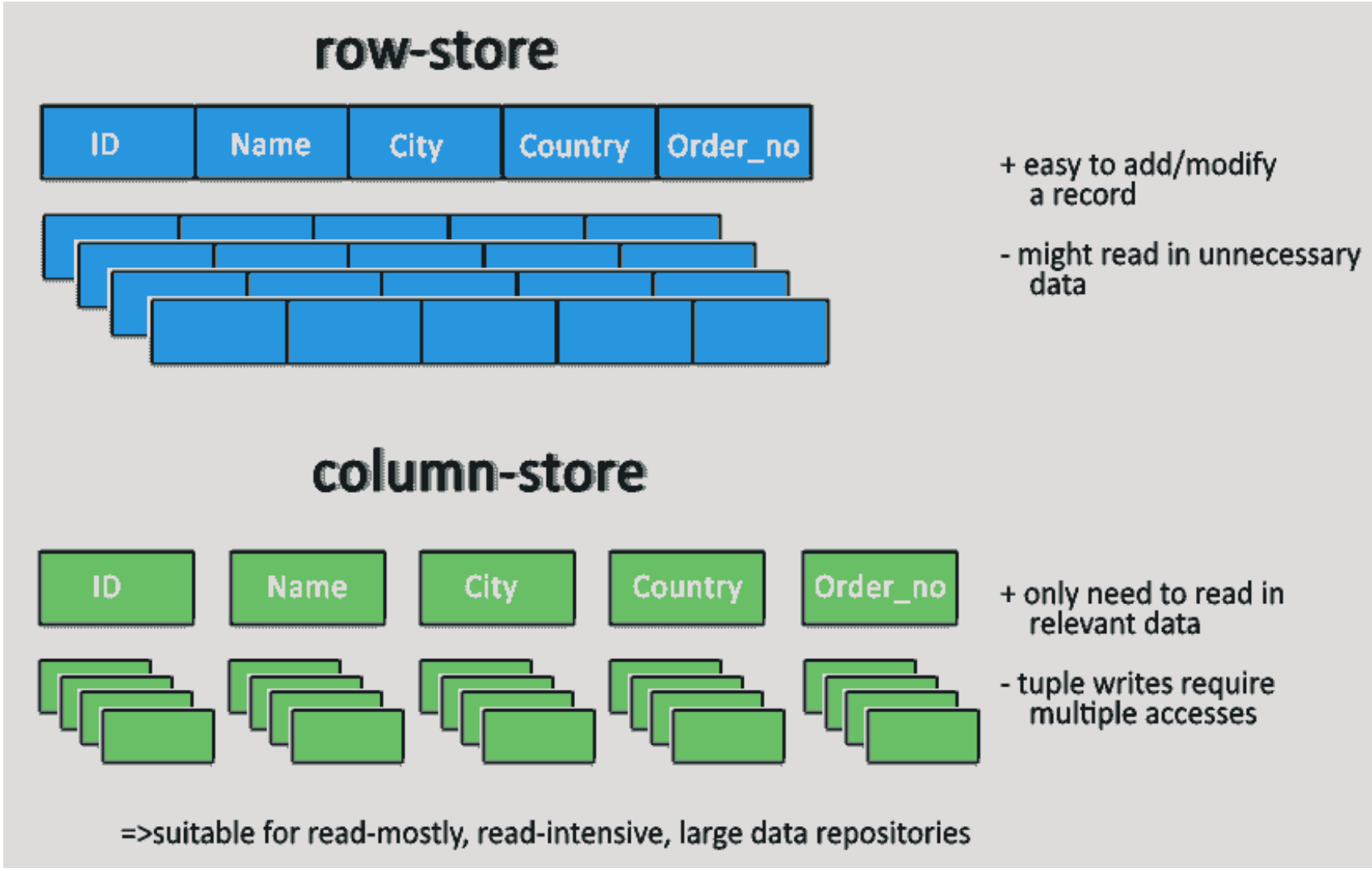
24.5 Column-Based or Wide Column NOSQL Systems

- Data is stored in **columns** **rather than** **rows**, enabling high-speed analytics and distributed computing.

- Efficient for **handling large-scale data** with high write/read demands.
- Great for **time-series data, IoT applications**, and big data analytics.

- **Examples**: **Apache Cassandra, HBase, Google Bigtable.**

24.5 Column-Based or Wide Column NOSQL Systems



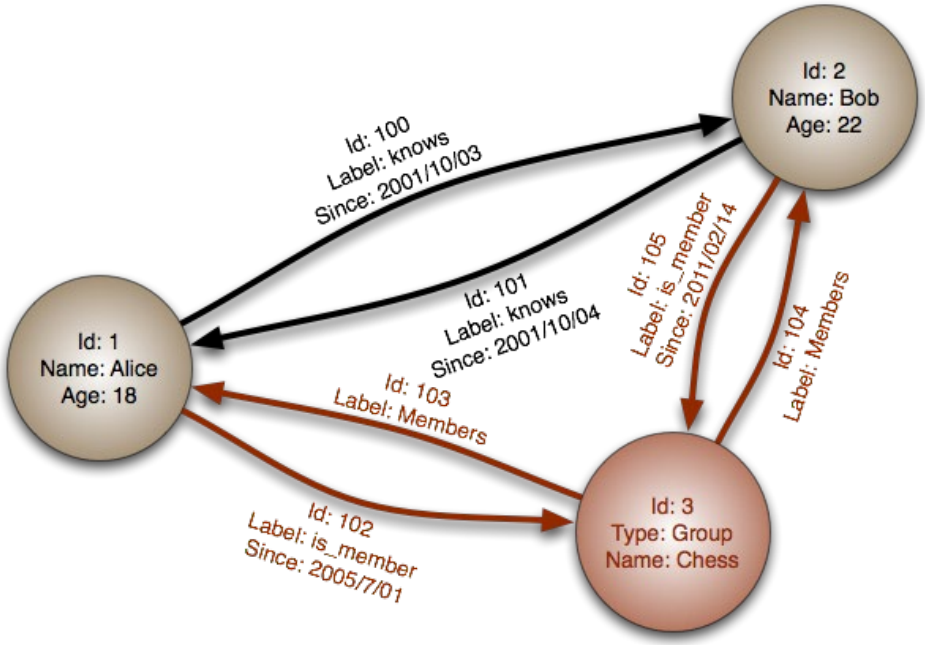
Hbase Data Model and Versioning (cont'd.)

- HBase stores multiple versions of data items
  - Timestamp associated with each version
- Each row in a table has a unique row key
- Table associated with one or more column families
- Column qualifiers can be dynamically specified as new table rows are created and inserted
- Namespace is collection of tables
- Cell holds a basic data item

Hbase Crud Operations

- Provides only low-level **CRUD** (create, read, update, delete) operations.
- Application programs implement more complex operations
- Create
  - Creates a new table and specifies one or more column families associated with the table
- Put
  - Inserts new data or new versions of existing data items

NOSQL Graph



24.6 NOSQL Graph Databases and Neo4j

- **Graph databases**
  - Data represented as a **graph**
  - Collection of vertices (**nodes**) and **edges**
  - Possible to store data associated with both individual nodes and individual edges.
  - Can perform relations faster than SQL

- Example: Neo4j, VertexDB

Neo4j (cont'd.)

- Nodes can have labels
  - Zero, one, or several
- Both nodes and relationships can have properties
- Each relationship has a start node, end node, and a relationship type
- Properties specified using a map pattern
- Somewhat similar to ER/EER concepts

Neo4j (cont'd.)

- Creating nodes in Neo4j
  - CREATE command
  - Part of high-level declarative query language
  - Node label can be specified when node is created
  - Properties are enclosed in curly brackets

24.7 **Summary**

- NOSQL systems focus on storage of "big data"
- General categories
  - Document-based
  - Key-value stores
  - Column-based
  - Graph-based
  - Some systems use techniques spanning two or more categories
- Consistency paradigms
- CAP theorem