NoSQL

Systems Analysis & Design

Learning Objectives

By the end of this session, you will have acquired the following information:

- ACID vs BASE
- Volume, Velocity, and Variety in Big Data
- Types of NoSQL Databases

Relational Database

- It organizes data into rows and columns, which collectively form a table.
- Data is typically structured across multiple tables, which can be joined together.
- Transactions are a key concept in relational databases.

Transaction

- A transaction is a unit of program execution that accesses and potentially updates various data items.
- This set of steps must appear to the user as a single, indivisible unit.
- Since a transaction is indivisible, it either executes completely or not at all.
- If a transaction begins to execute but fails for any reason, any changes that the transaction may have made to the database must be undone.

ACID

Atomicity

- Either the entire transaction occurs at once or it doesn't happen at all.

Consistency

- All data will remain valid according to all defined rules, both before and after the transaction.

Isolation

- Even though multiple transactions may execute concurrently, the system guarantees that, for every pair of transactions Ti and Tj, it appears to Ti that either Tj completed execution before Ti started, or Tj began execution after Ti finished.

Durability

- After a transaction successfully completes, the changes it has made to the database persist, even in the event of system failures.

Big Data

- Modern data management applications often need to handle data that is not necessarily in relational form.
- The volume of such data quickly grew well beyond the scale that traditional database systems could handle.
- This data is often referred to as Big Data.

Volume

- The volume of data to be stored and processed has significantly exceeded the capacity of traditional databases.

Velocity

- In today’s networked world, data arrives at a much higher rate than before.

Variety

- Not all data are relational.

BASE

Basically Available

- The data is available for both reading and writing operations.

Soft State

- The state of the system may change over time, even without user interaction, due to the synchronization of replicated values.

Eventual Consistency

- The system will eventually reach a consistent state after a certain period of time.

CAP Theorem

Any distributed system can simultaneously provide only two of three guarantees:

Consistency

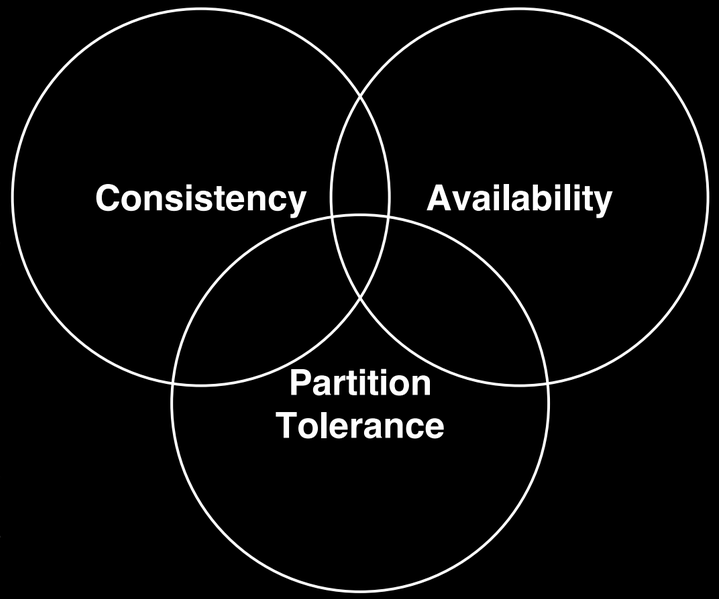
- Every read receives the most recent write or an error.

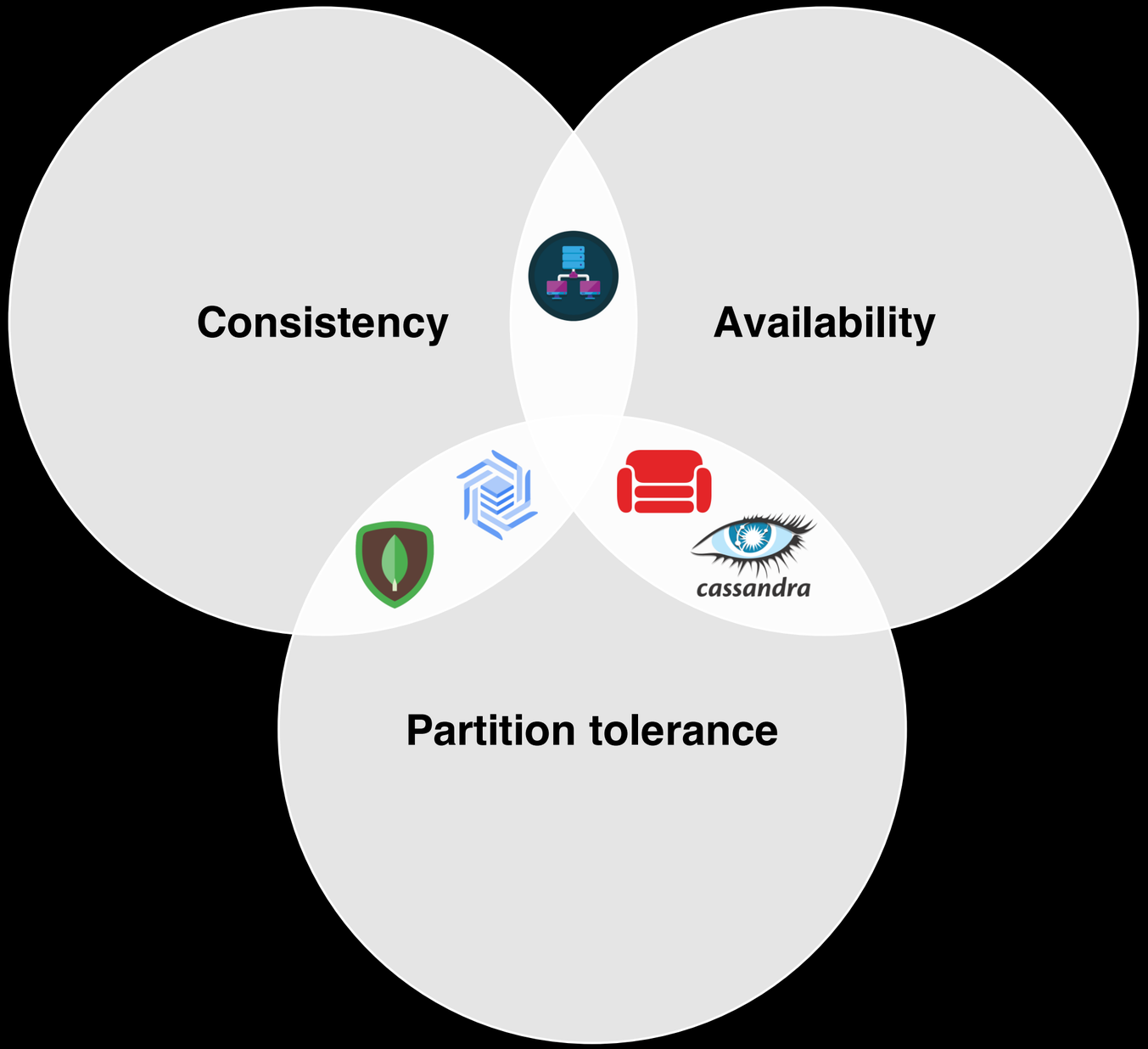
Availability

- Every request receives a non-error response, without the guarantee that it contains the most recent write.

Partition Tolerance

- The system continues to operate despite an arbitrary number of messages being dropped (or delayed) by the network between nodes.




Consistency
Availability
Partition tolerance
cassandra

Types of NOSQL Databases

Key-Value Databases

- Key-value databases store elements in the form of key-value pairs (k,v).
- “k” is a key. “v” is a value, which is a string of bytes.
- The value “v” can have a self-describing structure like JSON or XML. In this case, the data structure should be serialized and stored as a string of bytes.
- They are suitable for storing user session information and weather data.
- Amazon DynamoDB and Redis are examples of key-value databases.

```
# Set a key-value pair
127.0.0.1:6379> SET user:aaghamohammadi "Alireza Aghamohammadi"
OK

# Get the value of the key we just set
127.0.0.1:6379> GET user:aaghamohammadi
"Alireza Aghamohammadi"
```

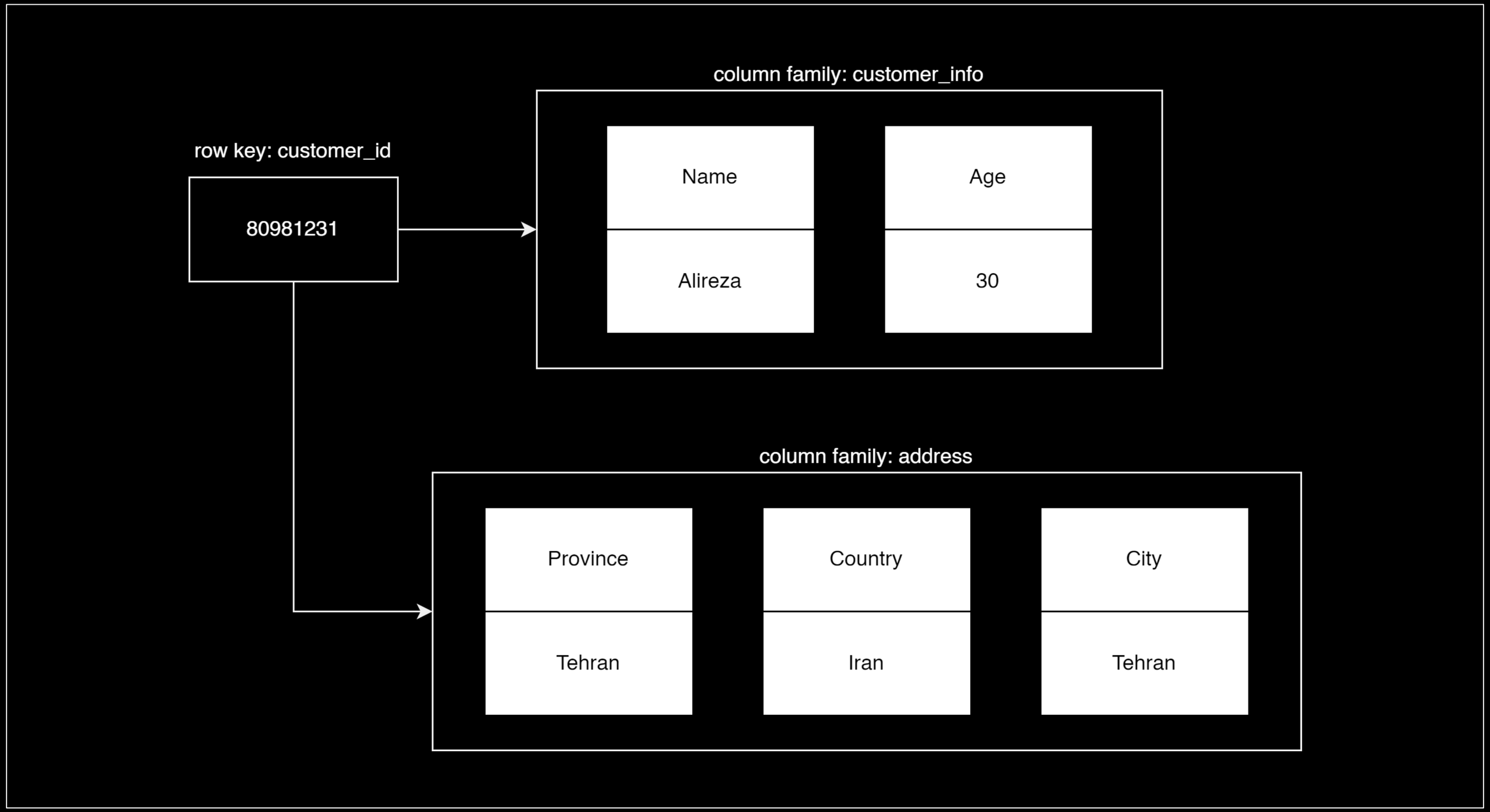
Document-Oriented Databases

- Document-oriented databases store data in a self-descriptive document format.
- They lack a predefined schema, providing flexibility.
- They reduce storage space by storing non-empty and important parts.
- The main difference with key-value databases is the ability to search on values.
- MongoDB is an example of a document-oriented database.

```
> db.profiles.insertMany([
    {name: "Alireza", age: 30, email: "al.aghamohammadi@gmail.com"},
    {name: "Cina", occupation: "R&D"}])
> db.profiles.find({age: {$gt: 25}})
```

Column-Family Databases

- Each table consists of rows that each have a unique key.
- These tables are composed of a fixed number of column families.
- Within each column family, there is a variable number of columns.
- Apache Cassandra is an example of a column-family database.
- It is suitable for analytical and statistical tasks.


column family: customer_info
row key: customer_id
80981231
Name
Age
Alireza
30
column family: address
Province
Country
City
Tehran
Iran
Tehran

Graph Databases

- Graph databases are composed of nodes and edges.
- Each node represents an entity (e.g., a person), and each edge represents a relationship between two nodes.
- They are suitable for modeling relationships in social networks.
- Neo4j is an example of a graph database.

```
# Create a node
CREATE (n:Person { name: 'Alireza', age: 30 })

# Create a relationship between nodes
MATCH (a:Person), (b:Person)
WHERE a.name = 'Alireza' AND b.name = 'Cina'
CREATE (a)-[r:FRIEND]->(b)
```

Further Resources

- Seven Databases in Seven Weeks, MongoDB (pages 106-146)