Notes - Session 4 - Databases

Sharding and Partitioning

What is Sharding?

Process of distributing data across multiple DB instances

What is Partitioning?

Process of splitting a subset of data within the same DB instance

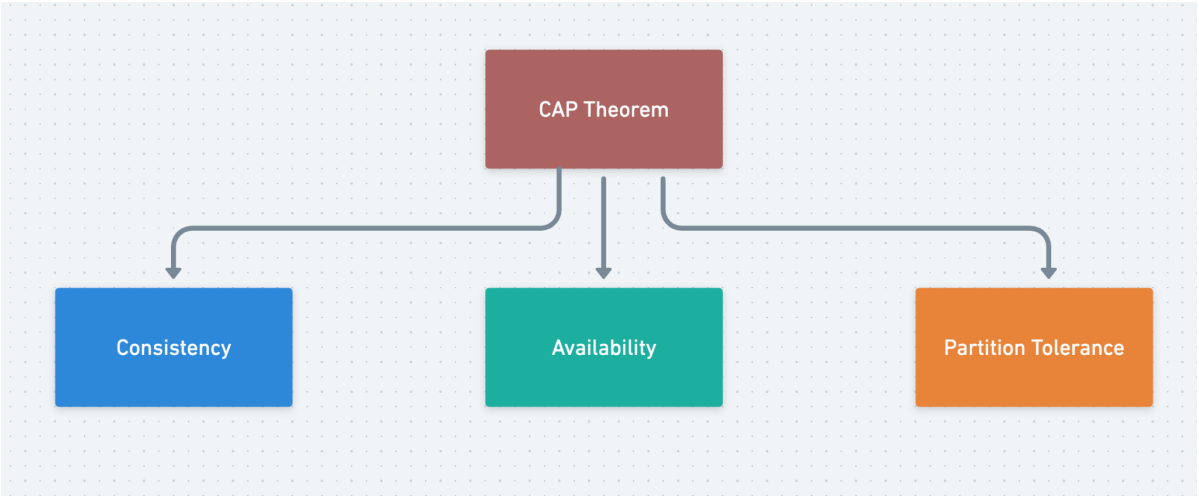
Advantages of Sharding:

- Higher availability
- Increase overall storage capacity
- Higher reads and writes

Disadvantages of Sharding:

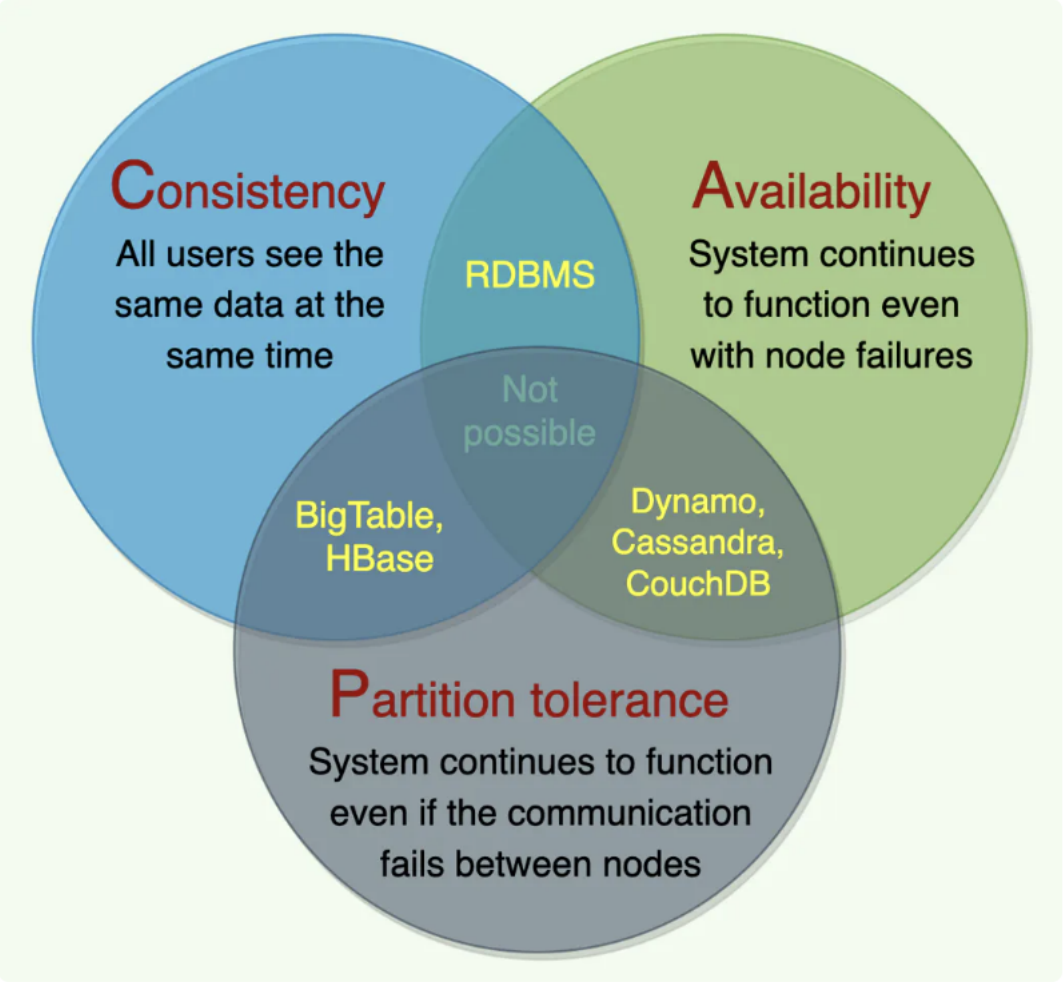
- Cross-shard queries are very expensive
- Complex to implement

CAP Theorem



One can only select 2 out of the 3 options i.e. ‘Consistency’, ‘Availability’, ‘Partition Tolerance’

Assuming that all databases need partition tolerance in the current world, you have to make a trade-off selection between ‘Consistency’ and ‘Availability’ while picking the right DB



CAP Theorem

NoSQL DB

- Schema-less design
- Horizontal scalability
- Performs well under high write loads

- Suited for large-scale data storage and retrieval

Types of NoSQL

1. Document DB:
    - Mostly JSON based
    - Supports complex queries
    - Don’t require a uniform schema
    - Flexibility in data storage

Use-case:

User Profiles, Product Catalog, Content Management, In-App Notifications

Examples:

MongoDB, Amazon DocumentDB, CouchDB.

Example Data:

```
Unset
{
  "_id": "123456",
  "name": "John Doe",
  "email": "john@example.com",
  "age": 29,
  "address": {
    "street": "123 Main St",
    "city": "Anytown",
    "state": "Anystate"
  }
}
```

Example Query:

```
Unset
db.users.find({ "age": { $gte: 30 } }) // finds all users aged 30 and above
db.users.find({ "address.city": "Anytown" }) // finds all users in "Anytown"
```

2. Key-Value Databases:
    - Extremely simple DB
    - High write and read performance for simple data models
    - Limited functionalities like GET, PUT, DEL
    - No complex queries supported like aggregations

Use-Case:

- Session Management
- User Preferences
- Product Recommendations

Examples:

Amazon DynamoDB, Azure Cosmos DB, Riak.

Example Data:

```
Unset
key: "user:1000", value: "{ 'name': 'Jane Doe', 'email': 'jane@example.com',
'password': 's3cret' }"
key: "user:1001", value: "{ 'name': 'John Smith', 'email': 'john@example.com',
'password': 'p@ssword' }"
```

Example Query:

```
Unset
GET user:1000 # retrieves the data associated with the key 'user:1000'
SET user:1002 "{ 'name': 'Emily Johnson', 'email': 'emily@example.com',
'password': '12345' }" # sets a record in the database
```

3. In-memory Key-Value Database:
    - The data is primarily stored in memory, unlike disk-based databases.
    - Minimal response times due to no disk access
    - Risk of losing data in case of server failure
    - Can persist data by storing it in log files or by taking snapshots

Examples:

Redis, Memcached, Amazon Elasticache.

4. Graph DB

- Maps relationship between data using nodes and edges
- Nodes are the values
- Edges are the relationships between values
- Best suited for running complex graph algorithms

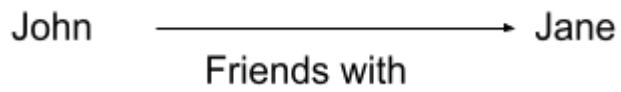
Use-Case:

- Session Management
- User Preferences
- Product Recommendations

Examples:

Amazon DynamoDB, Azure Cosmos DB, Riak.

Example Data:



Nodes:

```
Unset
CREATE (Person {name: 'John', age: 27})
CREATE (Person {name: 'Jane', age: 25})
```

Relationships:

```
Unset
CREATE (John)-[:FRIENDS_WITH]->(Jane)
```

Example Query:

Neo4j uses a query language called Cypher. Here's how you might find someone's friends:

```
Unset
MATCH (p:Person)-[:FRIENDS_WITH]->(friend) WHERE p.name = 'John' RETURN
friend
```

5. Time-Series DB (TSDB)

- Designed for time-stamped or time-series data
- Used for measuring changes in data over time

Use-Case:

- IoT Devices: Sensors sending data at regular intervals.
- Stock Market: Financial data tracking the fluctuation in stock prices.
- Performance Metrics: System health checks or performance of an application over time.
- Environmental Data: Tracking weather conditions.

Examples:

Graphite, Prometheus, Amazon Timestream, InfluxDB

Example Data:

Nodes:

In this table, every row represents a new data point, with its respective timestamp, source of the data, and the recorded value.

```
Unset
Timestamp                 |  Data Source  |  Value
-----------------------------------------------------------
2023-10-17T08:00:00Z      |  sensor_1     |  65.2
2023-10-17T08:01:00Z      |  sensor_1     |  67.1
2023-10-17T08:02:00Z      |  sensor_1     |  66.9
...                       |  ...          |  ...
```

Example Query:

This query calculates the average temperature from "sensor_1" in "building1" between 8:00 and 9:00 on October 17, 2023.

```
Unset
SELECT mean("temperature") FROM "temperature_readings"
WHERE "location" = 'building1' AND "sensor_id" = 'sensor_1'
AND time >= '2023-10-17T08:00:00Z' AND time <= '2023-10-17T09:00:00Z'
```

Selecting the right DB

- No one size fits all
- Important to select the DB based on the product use-case, and the user-journeys

Ask the right questions such as:

- What data are you storing?
- How much data storage is required?
- How will the data be accessed?
- What type of queries will we be firing?
- Any customization such as data expiry, etc?