



Tajamul Khan

# NoSQL Cheat Sheet



@Tajamulkhan



# Mongo DB

## Document-Based NoSQL Database

### Filtering Data

- show dbs // List all databases
- use myDatabase // Switch to database
- db.createCollection("users") // Create a collection
- collection show collections

### Insert Data

- db.users.insertOne({ name: "John", age: 30 })
- db.users.insertMany([{ name: "Alice" }, { name: "Bob" }])

### Find Data

- db.users.find()
- db.users.find({ age: { \$gt: 25 } })
- db.users.findOne({ name: "John" })

### Update Data

- db.users.updateOne({ name: "John" }, { \$set: { age: 35 } })
- db.users.updateMany({}, { \$set: { active: true } })

### Delete Data

- db.users.deleteOne({ name: "John" })
- db.users.deleteMany({ age: { \$lt: 18 } })

### Indexes

- db.users.createIndex({ name: 1 })
- db.users.createIndex({ email: 1 }, { unique: true })
- db.users.dropIndex("name\_1")

### Aggregation (Like GROUP BY in SQL)

- db.users.aggregate([ { \$group: { \_id: "\$age", count: { \$sum: 1 } } } ])

### Sorting & Limiting

- db.users.find().sort({ age: -1 })
- db.users.find().limit(5)

### Replication & Sharding

- rs.initiate()
- sh.enableSharding("myDatabase")
- sh.shardCollection("myDatabase.users", { \_id: "hashed" })



@Tajamulkhan



# Cassandra

## Column-Family Based NoSQL Database

### Basic Commands

- DESCRIBE KEYSPACES; -- Show all keyspaces  
USE myKeyspace; -- Switch to keyspace  
CREATE KEYSPACE myKeyspace WITH replication = {'class': 'SimpleStrategy', 'replication\_factor': 3};

### Create Table

- CREATE TABLE users ( id UUID PRIMARY KEY, name text, age int );

### Insert Data

- INSERT INTO users (id, name, age) VALUES (uuid(), 'Alice', 25);

### Select Data

- SELECT \* FROM users;
- SELECT name, age FROM users WHERE age > 20;

### Update Data

- UPDATE users SET age = 30 WHERE id = <some-uuid>;

### Delete Data

- DELETE FROM users WHERE id = <some-uuid>;

### Indexes

- CREATE INDEX ON users (age);

### Batch Operations

BEGIN BATCH

    INSERT INTO users (id, name, age) VALUES (uuid(), 'Bob', 28);

    INSERT INTO users (id, name, age) VALUES (uuid(), 'Eve', 32);

APPLY BATCH;

### Sorting & Limiting

```
CREATE TABLE orders (
    order_id UUID PRIMARY KEY,
    user_id UUID,
    product text
) WITH CLUSTERING ORDER BY (user_id ASC);
```



@Tajamulkhan



# Redis

## Key-Value NoSQL Database

### Basic Commands

- redis-cli # Start Redis CLI
- SET name "John" # Store a key-value pair
- GET name # Retrieve the value
- DEL name # Delete a key
- EXISTS name # Check if key exists

### Lists

```
PUSH queue "task1" # Push left  
RPUSH queue "task2" # Push right  
LPOP queue # Pop left  
RPOP queue # Pop right  
LRANGE queue 0 -1 # Get all elements
```

### Sets

```
SADD myset "apple" "banana" "cherry"  
SMEMBERS myset # View set items  
SISMEMBER myset "apple" # Check membership  
SREM myset "banana" # Remove from set
```

### Hashes

```
HSET user:1 name "Alice" age "30"  
HGET user:1 name  
HGETALL user:1
```

### Sorted Sets

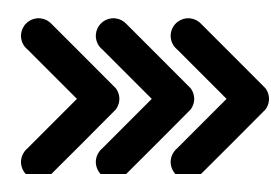
```
ZADD leaderboard 100 "Player1" 200 "Player2"  
ZRANGE leaderboard 0 -1 WITHSCORES  
ZREVRANGE leaderboard 0 -1 WITHSCORES
```

### Transactions

```
MULTI  
SET balance 100  
INCR balance  
EXEC
```



@Tajamulkhan



# DynamoDB

AWS NoSQL Database

## Create Table

- aws dynamodb create-table \  
  --table-name Users \  
  --attribute-definitions AttributeName=UserID,AttributeType=S \  
  --key-schema AttributeName=UserID,KeyType=HASH \  
  --billing-mode PAY\_PER\_REQUEST

## Insert Item

```
aws dynamodb put-item --table-name Users --item '{"UserID": {"S": "123"}, "Name": {"S": "Alice"}}'
```

## Query Items

```
aws dynamodb scan --table-name Users
```

```
aws dynamodb get-item --table-name Users --key '{"UserID": {"S": "123"}}'
```

## Update Items

```
aws dynamodb update-item \  
  --table-name Users \  
  --key '{"UserID": {"S": "123"}}' \  
  --update-expression "SET Age = :age" \  
  --expression-attribute-values '{":age": {"N": "30"}}'
```

## Delete Item

```
aws dynamodb delete-item --table-name Users --key '{"UserID": {"S": "123"}}'
```

## Create Secondary Index

- aws dynamodb update-table \  
  --table-name Users \  
  --attribute-definitions AttributeName=Age,AttributeType=N \  
  --global-secondary-index-updates "[{\\"Create\\":{\\\"IndexName\\": \\\"AgeIndex\\\", \\\"KeySchema\\\": [{\\\"AttributeName\\\": \\\"Age\\\", \\\"KeyType\\\": \\\"HASH\\\"}], \\\"Projection\\\": {\\\"ProjectionType\\\": \\\"ALL\\\"}, \\\"ProvisionedThroughput\\\": {\\\"ReadCapacityUnits\\\": 5, \\\"WriteCapacityUnits\\\": 5}}}]"



@Tajamulkhan



# Found Helpful?

---

# Repost



## Follow for more!

