USN | | | | | | | | | | |

Internal Assessment Test 2 – November 2024

Sub:	NoSQL database					Sub Code:	21CS745	Branch:	CSE
Date:	/11/2024	Duration:	90 mins	Max Marks:	50	Sem/Sec:	VII/A,B,C		OBE

Answer any 5 FIVE FULL Questions

		MARKS	CO	RBT
1	Imagine you're designing a data processing system for a company that frequently updates its datasets. How would you apply the incremental MapReduce process to efficiently handle these updates, and what strategies would you use to scale a key-value store for managing large, growing data volumes? Provide detailed explanations and examples to support your approach. Incremental Map-Reduce process -5 marks scaling in key-value store-5 marks	[10]	CO2	L2
2	Describe Map-reduce process to compare the sales of products for each month in 2011 to the prior year. Use suitable diagrams. Map-reduce process with description -8 marks Diagrams-2 marks	[10]	CO2	L3
3	Suppose we want to return all the documents in an order collection (all rows in the order table. Write down **SQL and MongoDB queries** for: i) Selecting the orders for a single customerId of 883c2c5b4e5b(3 marks) ii) Selecting orderId and orderDate for one customer (3 marks) iii) Query for all the orders where one of the items ordered has a name like Refactoring (2+2 marks)	[10]	CO3	L3
4	Describe how key-value stores handle consistency and transactions, and explain the process involved in creating buckets within these stores. Consistency in key-value store, bucket creation – 3+3 marks Transaction processing in key-value store- 4 marks	[10]	CO3	L2
5	Identify the situations where document databases are i) applicable ii) not advisable. Justify your answer. document databases are applicable with justification - 5 marks Not advisable with justification -5 marks	[10]	CO2	L3
6	Define a key-value store and illustrate it with an example. Also, identify and explain two important features of key-value stores. key-value store with example- 3+3 marks two features of key-value store - 4 marks	[10]	CO3	L2

1. Incremental MapReduce and Scaling in Key-Value Store

a) Incremental MapReduce Process

Incremental MapReduce processes efficiently handle updates to datasets by minimizing recomputation. Here's how it can be applied:

1. **Identify Updated Data:** Divide the dataset into incremental and non-incremental parts. Only process new or modified data since the last computation.
2. **Reuse Intermediate Results:** Store results from previous computations. Use these precomputed results to avoid reprocessing unchanged data.
3. **Combine New Results:** Merge the results from new data with stored intermediate outputs.

Example:

- In a sales database, if new sales records are added daily, only the new data is processed, and the existing aggregated results are reused.

---

b) Scaling in Key-Value Store
Key-value stores are scalable by design, but the following strategies enhance their scalability for large data volumes:

1. **Partitioning (Sharding):**
    - Distribute keys across multiple servers. Each server manages a subset of keys.
    - Example: Hash-based partitioning divides the keys based on a hash function.
2. **Replication:**
    - Create copies of data across multiple nodes to ensure high availability.
    - Example: Redis and Cassandra replicate data to handle node failures.
3. **Caching:**
    - Use in-memory caching to speed up data retrieval.
    - Example: Use tools like Redis as a cache for frequently accessed keys.
4. **Consistent Hashing:**
    - Ensures even distribution of data when nodes are added or removed.
    - Example: Systems like DynamoDB use consistent hashing to manage partitions.
5. **Write-Ahead Logs:**
    - Log changes before applying them to maintain consistency during scaling operations.

---

2. MapReduce Process for Comparing Sales Data

MapReduce Process

1. **Map Phase:**
    - Read the sales records for each month in 2011 and 2010.
    - Emit (productID, salesAmount) pairs for each record.
2. **Reduce Phase:**
    - Aggregate sales amounts for each productID by year.
    - Compare sales for 2011 and 2010. Emit the difference or percentage change.

Example Pseudocode:

python
Copy code
```python
# Mapper
for record in sales_data:
    emit(productID, (year, salesAmount))

# Reducer
for productID, values in grouped_data:
```

```
    total_2011 = sum(value[1] for value in values if value[0] == 2011)
    total_2010 = sum(value[1] for value in values if value[0] == 2010)
    emit(productID, total_2011 - total_2010)
```

---

3. SQL and MongoDB Queries for Orders Collection

i) Selecting Orders by CustomerID:

SQL:
sql
Copy code
```sql
SELECT * FROM orders WHERE customerID = '883c2c5b4e5b';
```

-

MongoDB:
javascript
Copy code
```javascript
db.orders.find({ customerID: "883c2c5b4e5b" });
```

-

ii) Selecting OrderID and OrderDate for One Customer:

SQL:
sql
Copy code
```sql
SELECT orderID, orderDate FROM orders WHERE customerID = '883c2c5b4e5b';
```

-

MongoDB:
javascript
Copy code
```javascript
db.orders.find(
  { customerID: "883c2c5b4e5b" },
  { orderID: 1, orderDate: 1, _id: 0 }
);
```

-

iii) Query for All Orders Where One Item Has a Specific Name:

SQL:
sql
Copy code
```sql
SELECT * FROM orders WHERE itemName = 'SpecificName';
```

-

MongoDB:
javascript

Copy code

```
db.orders.find({ "items.name": "SpecificName" });
```

-

---

4. Consistency and Transactions in Key-Value Stores

a) Consistency in Key-Value Store:

- **Eventual Consistency:**
  Changes are propagated across replicas asynchronously. Ensures high availability. Example: Amazon DynamoDB.
- **Strong Consistency:**
  Ensures that all reads return the most recent write. Example: Redis in specific configurations.

b) Bucket Creation Process:

- Buckets are logical groupings of key-value pairs.
- Process:
  1. Define the bucket.
  2. Assign replication and partitioning policies.
  3. Store and retrieve objects using unique keys.

c) Transaction Processing:

- Use atomic operations for read-modify-write cycles.
- Example in Redis: Multi commands ensure atomic execution.

---

5. Applicability of Document Databases

a) Situations Where Document Databases Are Applicable:

1. **Schema Flexibility:**
   - Example: Applications with dynamic or evolving data models.
2. **Nested Data:**
   - Example: E-commerce platforms storing order and customer data.
3. **Real-Time Analytics:**
   - Example: Log analysis systems like MongoDB.

b) Situations Where They Are Not Advisable:

1. **Complex Transactions:**
   - Use relational databases for multi-row or multi-table operations.
2. **Strict Schema Requirements:**
   - Example: Financial systems requiring fixed schemas.
3. **High Consistency Needs:**
   - Use databases like PostgreSQL for strict consistency.

---

6. Key-Value Store and Its Features

a) Definition and Example:

- **Definition:**
  A key-value store stores data as key-value pairs where keys are unique identifiers.

Example:
Redis storing user sessions:
plaintext
Copy code

```
Key: "user:1234"
Value: "{name: 'John', age: 30}"
```

-

b) Features of Key-Value Stores:

1. **Scalability:**
   - Horizontal scaling with sharding.
2. **Performance:**
   - Optimized for fast reads and writes.