DB INTERVIEW QUESTIONAIRE

INTERSWITCH

You are a Database Administrator for a large financial institution, which uses SQL Server as its the primary database. The company's database server is experiencing performance issues, and the users are reporting slow response times when querying the database.

Task: Identify the root cause of the performance issues and provide recommendations to optimize the database server's performance.

SOLUTION:
Based on the above scenario, the performance would be due to long-running queries, CPU and memory usage, or excessive requests from users at a time leading to the server slowing in response. This can be done by reviewing server logs and checking out the performance metrics and database queries.

The SQL Server has many features which include log shipping, database mirroring, failover clustering, and always-on availability groups.
Each feature above is briefly explained below;
Log shipping: This feature allows the creation of a standby database on another server as a backup from the primary database. It helps in minimizing data loss in case the primary database fails.
Database mirroring: It works as a standby database on a separate server and continuously updates changes made on the primary database. It provides high availability solution on a single database.
Failover clustering: This feature allows to creation n clusters that work as SQL server instances. In case of any server failures, the clusters will failover to the secondary server.
Always on Availability Groups (AG): This is a collection of databases that are replicated over multiple SQL server instances in case of failover on the server. This feature provides high availability and disaster recovery.

I would recommend **always on availability groups** to improve availability since in the scenario the database server is experiencing slow response time from the users. The always-on availability group has both high availability and disaster recovery capabilities which can minimize the ito mpact of slow response time in regard the performance of the server thus ensuring the server is available and can quickly recover from any failures. This is the best though a costly approach
I would recommend scaling the server to improve on the performance of the SQL Server since it would give the server more resources to perform better and efficiently hence giving users a great experience and load balancing since the access by users is on the network, it would help evenly distribution of traffic to prevent load traffic since the SQL Server is getting a slow response on the user's side hence would enable the server to perform better and also process user request quickly.

Table partitioning is a database design approach for dividing large tables into smaller manageable pieces of data called partitions. Data is divided into smaller pieces based on the partition key. When data is inserted in the table it is assigned/sent to a partition based on the partition key. When the table is queried, it scans through the necessary partitions based on the partition key.
This approach is best when the table is expected to grow to a large size and queries become slower over time.

Let me consider having a transaction table as an example;
Transaction Table

Transct ID	Date	Type	Amount	AccountNumber

In this case can query the transaction table with the transaction date by viewing transactions for a particular month. This can will the scan through the month's partition (given by date)

Query:

```
SELECT *
FROM transaction_table
WHERE date BETWEEN '2023-02-01' AND '2023-02-31';
```

Run a query for the total value of transactions for a specific user

Query:

```
SELECT *
FROM transaction_table
WHERE date BETWEEN '2023-02-01' AND '2023-02-31';
AND account_number = '0123456789'
GROUPBY account_number;
```

With the example above it's a demonstration of how table portioning can improve performance since it helps to limiting the amount of data being scanned and also helps in distributing workload across several partions to improve data retrieval speed.

From the above scenario, the following are queries that can cause performance issues as below;

Use of SELECT * it can cause performance issues. I recommend query optimization by using joins or filters to limit the amount of data being processed and also indexing by ensuring all tables are having appropriate indexes.

Stored procedures like SelectAllTransactions. I recommend avoiding stored procedures and use inline queries whenever possible.

I recommend monitoring and analyzing query performance regularly and the other tools besides SQL Server Profile Trace to analyze queries.

I recommend partitioning of tables to improve query performance with large table especially dates.

Some of the SQL server engine performance configuration that can improve the overall performance from the issues leading to performance are as follows:

CPU and Memory Configurations: Ensuring the SQL server has sufficient CPU resources and properly configured to use them and also allocating appropriate amount of memory to be used by the SQL Server instance.

Query optimization: Ensuring that queries are properly optimized to reduce resource usage and improve performance with appropriate indexing.

Database Maintenance: Regularly perform database maintenance tasks like

statistics updates and backups to ensure optimal database performance.

Database backups help in restoring data in the database in a previous state incase of data loss or any unplanned disasters. With a large financial institution, its very essential to have robust backup and recovery plan.
There are various backup types in SQL Server as below;
Full backup: Copies the entire database including all the data, database objects etc.
Differential backup: Copies all changes made since the last full backup.
Transactional log backup: It copies all transaction log changes since the last transaction log backup or full backup. Requires that the database is in full recovery mode.
Copy-only backup: This is a standalone backup that does not affect the normal backup and restore procedures.

Recommendations for no data loss and retention periods of 10 years:
Perform regular backups: I would recommend full backup at least once a week, differential once a day and transaction log backup every hour.
I would recommend backup monitoring to verify that backups are being done as scheduled.
I would recommend that backups are stored in multiple locations at least two separate locations like offsite storage to ensure there is availability of the data in case of any disaster.

As far as the over-growing database logs that consume server diskspace I would do the following;
Log file management: I would set up automated log files such truncating transaction log files or shrinking to control log file growth.
Increasing the frequency of transaction log backups in order to keep the log size in control.
I would enable instant file initialization to allow SQL Server allocate disk space which can save time and reduce log file growth during new database file creations.

From the above scenario, routine database maintenance is a set of tasks performed to keep the database running optimally. After analyzing the route cause of the

performance of the SQL server can either be as a result of long running queries, CPU and memory usage or excess requests from users, several routines can be done to ensure performance is optimized are follows;

Index maintenance: Defragmenting and rebuilding indexes to optimize query performance.
Database backups: Carrying out regular backups to ensure data is recovered in case of a disaster.
DB Integrity checks: Checking for data corruption for verifying the integrity of the database like checksum errors
Database cleanup: Removing unnecessary data from the database to reduce the size of the data.

The SQL Server provides SQL Server Agent with a built-in too for scheduling jobs. This allows to schedule maintenance jobs such as backups and index maintenance to run either on a regular basis or specified time.

Procedures to consider when conducting DB maintenance drills;
Monitor the health of the database by carrying out error checks and database corruption
Perform index maintenance to ensure queries are running efficiently.
Test backups regularly to ensure that are restored successfully.
Regularly review database usage patterns to identify areas that require optimization

Recommendations:
Monitoring server performance regularly include CPU Usage, Memory Usage, Disk I/O performance.
Considering portioning large tables for example considering a table for payments partitioning the amount by year/month could help on reducing on the querying time from the server.
Database configuration reviews regularly like memory allocation and file placement to ensure optimal performance.