Theory Questions:

**What do you consider as Big Data?**

Big data challenges include capturing data, data storage, data analysis, search, sharing, transfer, visualization, querying, updating, information privacy and data source

**What are the differences between SQL and NoSQL databases?**

|  |  |  |
| --- | --- | --- |
|  | SQL | NOSQL |
| Also known as | Non-relational or distributed database | Relational Databases (RDBMS) |
| Language | Queries are focused on collection of documents. Sometimes it is also called as UnQL (Unstructured Query Language). | Use SQL (Structured Query Language) for defining and manipulating the data |
| Basis | Databases are document based, key-value pairs, graph databases or wide-column stores | Table based databases |
| Data storage | Hierarchical data storage | No hierarchical data storage |
| Schema | Have dynamic schema for unstructured data | Have predefined schema |
| Schema | Database is schema-agnostic and is dictated by the application. It allows for agility and highly iterative development | Schema must be maintained and kept in sync between application and database |
| Data Properties | Offers flexibility as not every record needs to store the same properties | Great for solutions where every record has the same properties |
| New data | New data can be added without any reservation | Adding new data might require altering schemas or backfilling data |
| Types of Data | Good for semi-structured, complex, or nested data | Good for structured data |
| Relationships | Relationships between the data are often captured by denormalizing data and representing all data for an object in a single record | Relationships are often captured in normalized models using joins to resolve references across tables. |
| Scalability | Are horizontally scalable. Are scaled by increasing the databases servers in the pool of resources to reduce the load. | Are vertically scalable. Are scaled by increasing the horse-power of the hardware. |
| Complex Queries | Are not good fit for complex queries | Are good fit for the complex query intensive environment |
| Standard interfaces | Don’t have standard interfaces to perform complex queries | Have standard interfaces to perform complex queries |
| Size of data set | Highly preferred for large data set as can store hierarchical data | Not ideal for large data set |
| Complex transactional applications | Not comparable and stable enough in high load and for complex transactional applications. | Better suited for heavy duty transactional type applications, as it is more stable and promises the atomicity as well as integrity of the data. |
| Adoption and Support | Not widely adopted and local community support only | Widely adopted and easier support available |
| Properties | Follows the Brewers CAP theorem (Consistency, Availability and Partition tolerance). Some databases may follow ACID properties. | Emphasizes on ACID properties (Atomicity, Consistency, Isolation and Durability) |
| Examples | MongoDB, BigTable, Redis, RavenDb, Cassandra, Hbase, Neo4j and CouchDb | MySql, Oracle, Sqlite, Postgres and MS-SQL |

**What is Hadoop?**

Hadoop is an open source distributed processing framework that manages data processing and storage for big data applications running in clustered systems. It is at the center of a growing ecosystem of big data technologies that are primarily used to support advanced analytics initiatives, including predictive analytics, data mining and machine learning applications. Hadoop can handle various forms of structured and unstructured data, giving users more flexibility for collecting, processing and analysing data than relational databases and data warehouses provide.

**What is parquet file?**

A parquet file is a columnar file format that provides optimizations under the hood to speed up queries and is a far more efficient file format than CSV or JSON.

**Performance Testing**

How do you proceed and execute those tests?

**Load Testing:**

Load Testing is a type of performance testing that simulates real-world load on any software, application, or website. It examines how the system behaves during normal and high loads and determines if a system, piece of software, or computing device can handle high loads given a high demand of end users.

1. Identifying business goals. A strong understanding of future goals for scope and volume will draw clear guidelines to inform the process.
2. Determining key measures for the application and web performance. Agree on criteria to track. Some criteria include response times, throughput, resource utilization, maximum user load, and business performance metrics.
3. Choosing a suitable tool like LoadRunner or Jmeter
4. Creating test cases. In writing a test case, make sure both positive and negative scenarios are taken into account. [Test cases](https://stackify.com/constructing-good-test-cases/) must be accurate and capable of being traced to requirements.
5. Understanding our environment. Considering different types of deployments. Creating configurations similar to typical production
6. Run tests incrementally. During these tests, the system will ultimately fail. One key goal is determining what volume results in failure, and spotlighting what fails first.

**Endurance Testing:**

Endurance Testing is testing of an application or software with a significant load extended over a long period of time, to evaluate how the system behaves under sustained use.

Below is the basic testing approach for Endurance Test

Testing Environment – Identifying the hardware, software, operating system required for the endurance testing, assigning roles and responsibilities within the team, etc. The environment should be ready before the test execution. We also need to estimate common database production size and yearly growth

Creating the Test Plan, Scenarios - Based on nature of testing - manual or automation or combination of both, Test Case design, reviews, and execution should be planned. Testing to stress the system, break point testing, etc. should also be part of the test plan. Testing to stress the system determines the break point in the application.

When testing using a performance tool, we should consider those points:

Will performance remain consistent over time?

Are there other minor issues that have not yet been detected?

Is there external interference that was not addressed?

Analysing reports and retesting

**Volume testing:**

Volume testing is done to analyze the system performance by increasing the volume of data in the database. With the help of Volume testing, the impact on response time and system behavior can be studied when exposed to a high volume of data.

In volume testing, following things need to be tested

Test to check if there is any data loss

Check the system's response time

Check if the data is stored correctly or not

Verify if the data is overwritten without any notification

Check for warning and error messages, whether it comes at all for volume problems

Check whether high volume data affects the speed of processing

Approach for executing Volume test cases:

Stoping all servers and check all logs

Before the load test, we manually execute the application scenario

For most useful results we stagger the number of users

Analyzing the use case for improvement once a baseline has been established

A repetition of particular parts of volume testing becomes inevitable in case there is a performance bottleneck

**Capacity Testing:**

Capacity testing is targeted at testing whether the application and environment can handle the amount of traffic that it was designed to handle. It can be used during the design phase to benchmark the numbers of users or transaction the system can handle under a given set of circumstances.

Capacity testing can also help with strategic planning, so we can proactively increase infrastructure before limits are reached and the user experience is impacted.

The approach is the same as the load testing.

**Stress testing:**

The purpose of web server stress testing is to find the target application’s crash point.

The crash point is not always an error message or access violation. It can be a perceptible slowdown in the request processing.

Approach for testing on a website:

1. Recording one or more user scenarios.
2. Creating a load test that will simulate several virtual users.
3. Specifying the browser, connection speed and start delay for each user.
4. Specifying the workstation for each user.
5. Run the test.
6. Analyzing the load test results and check whether the application “crashes”.
7. If the test results contain warnings and errors, this may mean that you we have found the crash point.
8. Once again, a crash is not always an error, this may be unacceptable performance.
9. If the application did not crash, we increase the number of virtual users and perform the test again.

**The approach to ensure that tests could be rerun in the future:**

1. Design test cases that take into account the impact on the potential hardware growth (i.e data base size, file size..).
2. Designing test cases that can adapt to potential load increase on requirements (i.e: more user connections, more transactions
3. Tescases should be reusable as much as possible, in case of automation, the implementation of procedures and functions are recommended practices.
4. Writing good test cases and document them using an adequate tool, avoiding keeping fragile test cases in the test suit.
5. Storing testdata and testcases, with reports in an adequate tool.
6. Keeping reports, could help in terms of result comparison in the future.
7. Giving continuous support to the team and transfer knowledge upon new joiners so they can continue the design and execution without any human issues due to lack f knowledge.

**The most important KPI to store are:**

* Number of Users
* Requests per second
* Errors per Second
* Response Time
* Latency
* Connect time
* Bandwidth consumption that is generated by the test per second - Bytes/s (Throughput)

**What will determine the status of the tests?**

The test reports is the way of reporting the status of the test. Those reports are related to each other’s:

* Test plan status, given a oveall all idea of the plan to all the team,
* Test documentation status: where the designing of the tests, data gathering and other activities have begun and also when they are finished. This report will not only let them know about the progress of the task but also signal the teams that need to review and provide signoff on the artifacts, that they are up next.
* Test execution status and/or defect status