UNIVERSIDAD TECNOLÓGICA
DE TAMAULIPAS NORTE

TECNIQUE INFORM:

“BDCN UNIT ACTIVITY 1”

HIGHER UNIVERSITY TECHNICIAN IN INFORMATION TECHNOLOGY DEVELOPMENT OF MULTI-PLATFORM SOFTWARE AREA.

PRESENT BY:

JOEL ALEJANDRO REYES VICENTE

PRESENT BY:

ING. FRANCISCO EMANUEL PORTILLA SALAZAR

CT. REYNOSA, TAMAULIPAS **AUGUST 2023.**

Introduction.

NoSQL databases are designed for a number of data access patterns that include low-latency applications. NoSQL search databases are designed for analytics over semi-structured data.

NoSQL databases provide a variety of data models such as key-value, document, and graph, which are optimized for performance and scale.

NoSQL databases often make tradeoffs by relaxing some of the ACID properties of relational databases for a more flexible data model that can scale horizontally. This makes NoSQL databases an excellent choice for high throughput, low-latency use cases that need to scale horizontally beyond the limitations of a single instance.

NoSQL databases typically are partitionable because access patterns are able to scale out by using distributed architecture to increase throughput that provides consistent performance at near boundless scale.

Characteristics of the NoSQL DMS.

MongoDB.

-Ad-hoc queries for optimized, real-time analytics.

When designing the schema of a database, it is impossible to know in advance all the queries that will be performed by end users. An ad hoc query is a short-lived command whose value depends on a variable. Each time an ad hoc query is executed, the result may be different, depending on the variables in question.

MongoDB supports field queries, range queries, and regular expression searches. Queries can return specific fields and also account for user-defined functions. This is made possible because MongoDB indexes BSON documents and uses the MongoDB Query Language (MQL).

-Indexing appropriately for better query executions.

The number one issue that many technical support teams fail to address with their users is indexing. Done right, indexes are intended to improve search speed and performance. A failure to properly define appropriate indices can and usually will lead to a myriad of accessibility issues, such as problems with query execution and load balancing.

-Replication for better data availability and stability.

When your data only resides in a single database, it is exposed to multiple potential points of failure, such as a server crash, service interruptions, or even good old hardware failure. Any of these events would make accessing your data nearly impossible.

Replication allows you to sidestep these vulnerabilities by deploying multiple servers for disaster recovery and backup. Horizontal scaling across multiple servers that house the same data (or shards of that same data) means greatly increased data availability and stability. Naturally, replication also helps with load balancing. When multiple users access the same data, the load can be distributed evenly across servers.

-Sharding.

When dealing with particularly large datasets, sharding—the process of splitting larger datasets across multiple distributed collections, or “shards”—helps the database distribute and better execute what might otherwise be problematic and cumbersome queries. Without sharding, scaling a growing web application with millions of daily users is nearly impossible.

-Load balancing.

At the end of the day, optimal load balancing remains one of the holy grails of large-scale database management for growing enterprise applications. Properly distributing millions of client requests to hundreds or thousands of servers can lead to a noticeable (and much appreciated) difference in performance.

Cassandra.

-Open-source availability

Nothing is more exciting than getting a handy product for free. This is probably one of the significant factors behind Cassandra’s far-reaching popularity and acceptance. Cassandra is among the open-source products hosted by Apache and is free for anyone who wants to utilize it.

-Distributed footprint.

Another feature of Cassandra is that it is well distributed and meant to run over multiple nodes as opposed to a central system. All the nodes are equal in significance, and without a master node, no bottleneck slows the process down. This is very important because the companies that utilize Cassandra need to constantly run on accurate data and can not tolerate data loss. The equal and wide distribution of Cassandra data across nodes means that losing one node does not significantly affect the system’s general performance.

-Scalability.

Cassandra has elastic scalability. This means that it can be scaled up or down without much difficulty or resistance. Cassandra’s scalability once again is due to the nodal architecture. It is intended to grow horizontally as your needs as a developer or company grow. Scaling-up in Cassandra is very easy and not limited to location. Adding or removing extra nodes can adjust your database system to suit your dynamic needs.

-Cassandra Query Language.

Cassandra is not a relational database and does not use the standard query language or SQL. It uses the Cassandra query language (CQL). This would have posed a problem for admins as they would have to master a whole new language – but the good thing about Cassandra Query language is that it is very similar to SQL. It is structured to operate with rows and columns, i.e., table-based data.

-Fault tolerance.

Cassandra is fault-tolerant primarily because of its data replicative ability. Data replication denotes the ability of the system to store the same information at multiple locations or nodes. This makes it highly available and tolerant of faults in the system. Failure of a single node or data center does not bring the system to a halt as data has been replicated and stored across other nodes in the cluster. Data replication leads to a high level of backup and recovery.

Oracle NoSQL.

Simple APIs: Oracle NoSQL Database Cloud Service provides easy-to-use CRUD (Create Read Update Delete) APIs that allow developers to easily create tables and maintain data in them.

Data Modeling: Oracle NoSQL Database Cloud Service supports both schema-based and schema-less (JSON) modeling.

Data Safety in Redundancy: The Oracle NoSQL Database Cloud Service stores data across multiple Availability Domains (ADs) or Fault Domains (FDs) in single AD regions. If an AD or FD becomes unavailable, user data is still accessible from another AD or FD.

Data Security: Data is encrypted at rest (on disk) with Advanced Encryption Standard (AES 256). Data is encrypted in motion (transferring data between the application and Oracle NoSQL Database Cloud Service) with HTTPS.

ACID-Compliant Transactions: ACID (Atomicity, Consistency, Isolation, Durability) transactions are fully supported for the data you store in Oracle NoSQL Database Cloud Service. If required, consistency can be relaxed in favor of lower latency.

JSON Data Support: Oracle NoSQL Database Cloud Service allows developers to query schema-less JSON data by using the familiar SQL syntax.

Partial JSON Updates: Oracle NoSQL Database Cloud Service allows developers to update (change, add, and remove) parts of a JSON document. Because these updates occur on the server, the need for a read-modify-write cycle is eliminated, which would consume throughput capacity.

Time-To-Live: Oracle NoSQL Database Cloud Service lets developers set a time frame on table rows, after which the rows expire automatically, and are no longer available. This feature is a critical requirement when capturing sensor data for Internet Of Things (IoT) services.

SQL Queries: Oracle NoSQL Database Cloud Service lets developers access data with SQL queries.