COMPUTER ENGINEERING DEPARTMENT

BDA Assignment 1

COURSE: **B.E.**   YEAR: **2020-2021**   SEMESTER:
VII DEPT: **Computer Engineering**
SUBJECT CODE: **CSDLO7032**   DATE OF ASSIGNMENT: **24-08-2021**
==
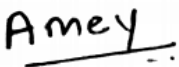
NAME: **AMEY MAHENDRA THAKUR**   ROLL NO.: **50**

CLASS: **COMPS BE B**   DATE OF SUBMISSION: **24-08-2021**

Sr. No.	Questions
1	How databases use replication to make backup copies of data in real time?
2	How can load balancers work with the application layer to distribute queries to the correct database server?
3	How using replication allows you to horizontally scale read requests?

Amey

Signature of Student

Q.1 How databases use replication to make backup copies of data in real time?
Ans:

- With the number of users/visitors trying to access the data more frequently and parallelly, ensuring a high data availability is one of the biggest challenges organisations face today. Irrespective of the data type, be it blogs, linked media, etc., organisations have an ever-growing need to scale up their systems to provide robust and speedier access to data. While fitter data transfer and storage technologies have facilitated this cause, data replication triggers an organised mechanism to optimise data availability.

- **Real-Time Data Replication** refers to the process of synchronising data across a source and a destination database as and when a change occurs in the data source. Most databases provide robust inherent support for replicating data in real-time, thereby allowing users to generate numerous copies of complex datasets and storing them across various locations of the same kind of database to facilitate seamless access. It plays a crucial role in ensuring the high availability of data for individual systems and numerous servers.

- **Strategies to Achieve Real Time Data Replication:**
1. Using Built-In Replication Mechanisms:
   - Most enterprise-grade transaction databases such as Oracle, MariaDB, PostgreSQL, etc. house robust support for in-built data replication mechanisms that help users backup their data to a similar replica database.

2. Using Continuous Polling Methods:
   - Using the continuous polling mechanism to achieve Real Time Data Replication, requires you to write custom-code snippets to replicate/copy data from your source database to your destination database.

3. Using Trigger-Based Custom Solution:
   - Databases such as Oracle, MariaDB, etc. house an intuitive functionality of built-in triggers that execute under specific scenarios to carry out an operation such as data replication. These operate as call-back functions and insert data-based changes into a queuing mechanism such as Kafka, RabbitMQ, etc., and then the consumer helps transfer data to the destination database.

4. Using Transaction Logs:
   - Most databases maintain transaction logs to monitor operations that are occurring in the database. Such data logs contain information associated with operations/tasks such as data definition commands, inserts, updates, deletes, etc. and even keep track of the specific points where these tasks take place. Such databases allow users to access the transaction logs non-synchronously to fetch the changes and then leverage a queuing mechanism such as Kafka to replicate

changes in the destination database.

5. <u>Using Cloud-Based Mechanisms</u>:
    - In case you leverage a cloud-based database to manage & store your crucial business data; then you might already have robust mechanisms in place for your cloud computing platform. Various cloud service providers house the support for carrying out Real Time Data Replication.

Q.2 How can load balancers work with the application layer to distribute queries to the correct database server?
Ans:

- As strain increases on a website or business application, eventually, a single server cannot support the full workload. To meet demand, organizations spread the workload over multiple servers. Called "load balancing," this practice prevents a single server from becoming overworked, which could cause it to slow down, drop requests, and even crash.

- Load balancing lets you evenly distribute network traffic to prevent failure caused by overloading a particular resource. This strategy improves the performance and availability of applications, websites, databases, and other computing resources. It also helps process user requests quickly and accurately.

- From a user perspective, load balancing acts as an invisible facilitator that sits between a client and a group of servers, ensuring connection requests don't get lost. Without load balancing, applications, websites, databases, and online services would likely fail when demand gets too high. A single high-traffic website may field hundreds or thousands of user requests at the same time. It needs multiple servers to accurately populate web pages with the requested information, including text, photos, video, and audio streaming.

- The load balancer uses a predetermined pattern, known as a load balancing algorithm or method. This ensures no one server has to handle more traffic than it can process. Different algorithms manage the process using different techniques. You, therefore, have multiple options to choose from when making a decision on what type of load balancer to use.

- **Here are the basics of how a load balancer works:**
    - A client, such as an application or browser, receives a request and tries to connect with a server.
    - A load balancer receives the request, and, based on the preset patterns of the algorithm, it routes the request to one of the servers in a server group (or farm).
    - The server receives the connection request and responds to the client via the load balancer.
    - The load balancer receives the response and matches the IP of the client with that of the selected server. It then forwards the packet with

the response.
- Where applicable, the load balancer handles SSL offload, which is the process of decrypting data using the Security Socket Layer encryption protocol, so that servers don't have to do it.
- The process repeats until the session is over.

Q.3 How using replication allows you to horizontally scale read requests?
Ans:

- Horizontal scaling, also known as scale-out, refers to bringing on additional nodes to share the load. This is difficult with relational databases due to the difficulty in spreading out related data across nodes. With non-relational databases, this is made simpler since collections are self-contained and not coupled relationally. This allows them to be distributed across nodes more simply, as queries do not have to "join" them together across nodes.

- There are a variety of scaling techniques which depend on the database system and what components are used. However, they all use the concept of a node, which is an individual machine storing some or all of the data. A group of nodes that work together is called a cluster.

- There are two commonly used horizontal database scaling techniques: replication and horizontal partitioning (or sharding).

- Replication refers to creating copies of a database or database node. Replication adds fault-tolerance to a system. Each node in a cluster contains a copy of the data. If one of the nodes goes down, the cluster is still able to serve client requests because the other nodes in the cluster can respond to the requests.

- Replication is also a form of scaling because client requests can be spread across all the nodes in the cluster instead of overwhelming a single node. This increases the capacity of the system to handle more database read requests.