TY B.Tech. (CSE) – II [2022-23]

5CS372 : Advanced Database System Lab.

Assignment No. 08

Name: Khushi Nitinkumar Patel

PRN: 2020BTECS00037

Batch: T2

Title: Distributed databases

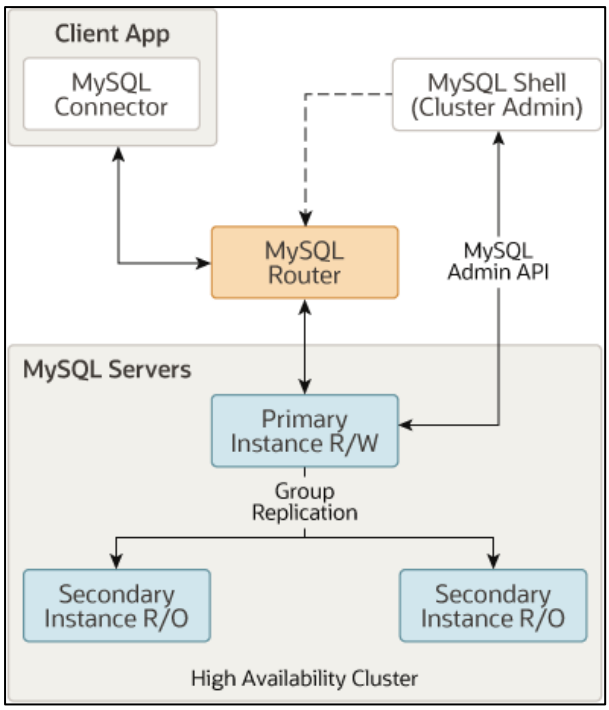
Aim: Installation and configuration of MYSQL InnoDB cluster and demonstrate the working by connecting portals to the different nodes.

Introduction and Related Theory:

An InnoDB Cluster consists of at least three MySQL Server instances, and it provides high-availability and scaling features. InnoDB Cluster uses the following MySQL technologies:

- MySQL Shell, which is an advanced client and code editor for MySQL.
- MySQL Server, and Group Replication, which enables a set of MySQL instances to provide high-availability. InnoDB Cluster provides an alternative, easy to use programmatic way to work with Group Replication.
- MySQL Router, a lightweight middleware that provides transparent routing between your application and InnoDB Cluster.

The following diagram shows an overview of how these technologies work together:



You work with InnoDB Cluster using the AdminAPI, provided as part of MySQL Shell. AdminAPI is available in JavaScript and Python, and is well suited to scripting and automation of deployments of MySQL to achieve high-availability and scalability. By using MySQL Shell's AdminAPI, you can avoid the need to configure many instances manually. Instead, AdminAPI provides an effective modern interface to sets of MySQL instances and enables you to provision, administer, and monitor your deployment from one central tool.

To get started with InnoDB Cluster you need to download and install MySQL Shell. You need some hosts with MySQL Server instances installed, and you can also install MySQL Router.

InnoDB Cluster supports MySQL Clone, which enables you to provision instances simply. In the past, to provision a new instance before it joins a set of MySQL instances you would need to somehow manually transfer the transactions to the joining instance. This could involve making file copies, manually copying them, and so on. Using InnoDB Cluster, you can simply add an instance to the cluster and it is automatically provisioned.

Similarly, InnoDB Cluster is tightly integrated with MySQL Router, and you can use AdminAPI to work with them together. MySQL Router can automatically configure itself based on an InnoDB Cluster, in a process called bootstrapping, which removes the need for you to configure routing manually. MySQL Router then transparently connects client applications to the InnoDB Cluster, providing routing and load-balancing for client connections. This integration also enables you to administer some aspects of a MySQL Router bootstrapped against an InnoDB Cluster using AdminAPI. InnoDB Cluster status information includes details about MySQL Routers bootstrapped against the cluster. Operations enable you to create MySQL Router users at the cluster level, to work with the MySQL Routers bootstrapped against the cluster, and so on.

Installing Required Packages:

We will use the new MySQL Shell to manage 3 MySQL Server instances and create a 3 member InnoDB cluster (running Group Replication). Then we will set up the new MySQL Router as to hide the multiple MySQL instances behind a single TCP port. Client applications can then connect to the ports MySQL Router provides, without any need to be aware of the InnoDB cluster topology. In the event of an unexpected failure, the InnoDB cluster adjusts itself automatically and MySQL Router detects the change. This removes the need for your client application to handle failover.

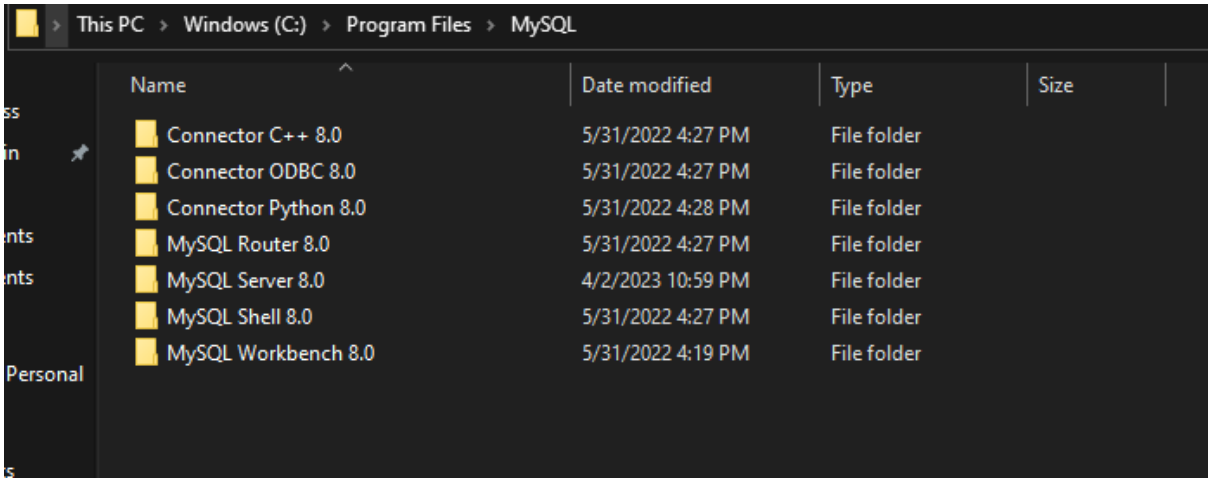
In addition to MySQL Server 8.0.11 GA, you will need to download the two other components of MySQL InnoDB Cluster:

- MySQL Shell 8.0.11 GA with X AdminAPI
- MySQL Router 8.0.11 GA

Download and install the packages, through the usual platform-specific methods:

- **MySQL Server**
- **MySQL Shell**
- **MySQL Router**

Default installation path is C:program files/mysql



Configuring InnoDB Cluster

We need a minimum of three instances of MySQL in the cluster to make it tolerant to the failure of one instance (Group Replication fault-tolerance). We'll start by three instances running in different hosts, so the next step is to prepare instances for InnoDB cluster. Before moving to the auto-configuration of the instances we want to check whether instances are already ready for InnoDB cluster usage or not.

Note: For testing purposes only, sandbox deployment is available on the Shell. Sandbox deployment can be done using dba.deploySandboxInstance().

Example: **dba.deploySandboxInstance(3306);**

```
MySQL  JS > dba.deploySandboxInstance(3307);
A new MySQL sandbox instance will be created on this host in
C:\Users\DBE2740\MySQL\mysql-sandboxes\3307

Warning: Sandbox instances are only suitable for deploying and
running on your local machine for testing purposes and are not
accessible from external networks.

Please enter a MySQL root password for the new instance: *******

Deploying new MySQL instance...

Instance localhost:3307 successfully deployed and started.
Use shell.connect('root@localhost:3307') to connect to the instance.
```

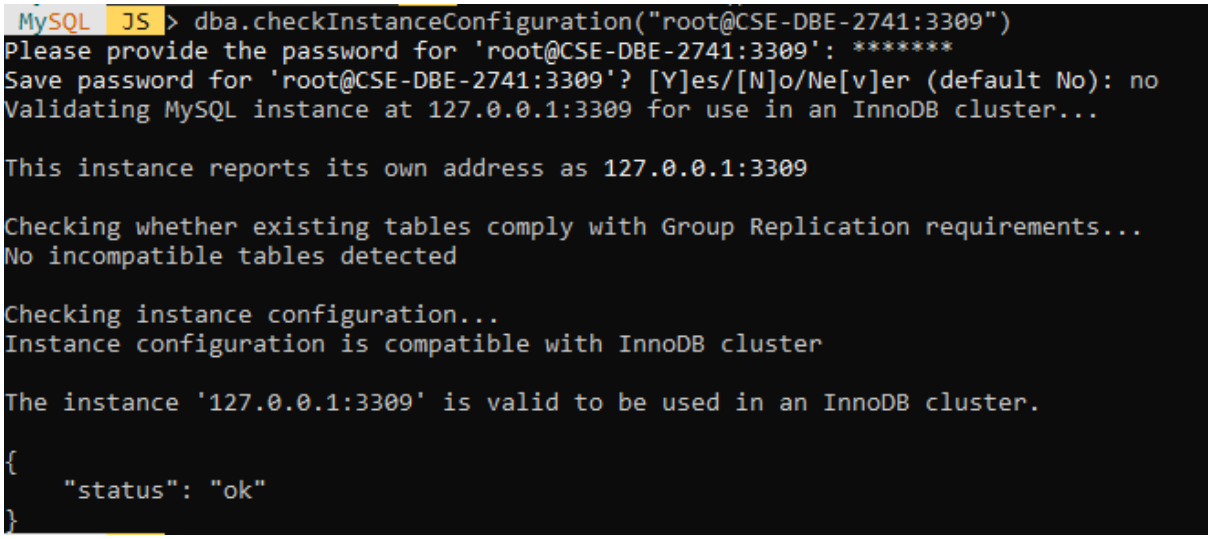
```
MySQL  JS > shell.connect('root@localhost:3307')
Creating a session to 'root@localhost:3307'
Please provide the password for 'root@localhost:3307': *******
Save password for 'root@localhost:3307'? [Y]es/[N]o/Ne[v]er (default No): no
Fetching schema names for auto-completion... Press ^C to stop.
Your MySQL connection id is 12
Server version: 8.0.33 MySQL Community Server - GPL
No default schema selected; type \use <schema> to set one.
<ClassicSession:root@localhost:3307>
```

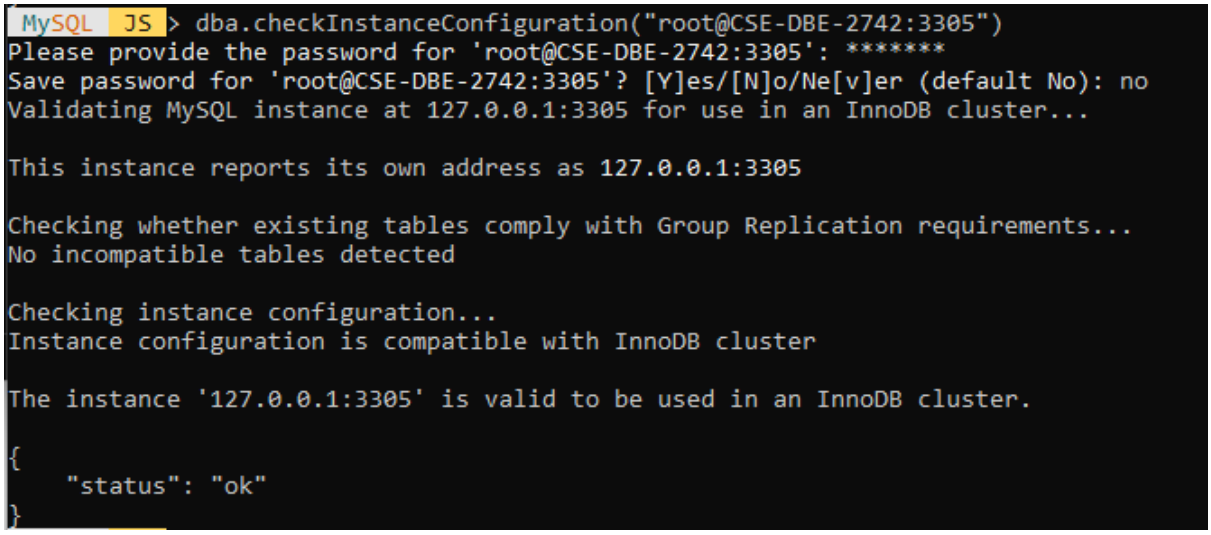
```
MySQL  localhost:3307 ssl  JS > shell.disconnect()
```

Start MySQL Shell:- **mysqlsh**

The argument to **checkInstanceConfiguration()** is the connection data to a MySQL Server instance. The connection data may be specified in the following formats:

- A URI string
- A dictionary with the connection options





```
MySQL  JS > dba.checkInstanceConfiguration("root@localhost:3307")
Please provide the password for 'root@localhost:3307': *******
Save password for 'root@localhost:3307'? [Y]es/[N]o/Ne[v]er (default No): no
Validating local MySQL instance listening at port 3307 for use in an InnoDB cluster...
NOTE: Instance detected as a sandbox.
Please note that sandbox instances are only suitable for deploying test clusters for use within the same host.

This instance reports its own address as 127.0.0.1:3307

Checking whether existing tables comply with Group Replication requirements...
No incompatible tables detected

Checking instance configuration...
Instance configuration is compatible with InnoDB cluster

The instance '127.0.0.1:3307' is valid to be used in an InnoDB cluster.

{
    "status": "ok"
}
```

The AdminAPI provides a command to automatically and remotely configure an instance for InnoDB cluster usage: dba.configureInstance(). So the next step is to use dba.configureInstance() on each target instance.

Note: dba.configureInstance() does not need to be executed locally on the target instance. It can be executed remotely and supports remote restart of the target instance (if required).

Repeat the steps above on all the instances before initializing your cluster.

Note: The user account used to administer an instance does not have to be the root account, however it needs to be an account with a specific set of priveliges for InnoDB cluster management. The preferred method to create users to administer the cluster is using the clusterAdmin option with the dba.configureInstance().

Example: **dba.configureInstance("root@ic-1:3306", {clusterAdmin: "myAdmin", clusterAdminPassword: "myAdminPwd"};**

```
MySQL  JS > dba.configureInstance("root@CSE-DBE-2741:3309",{clusterAdmin:"sample",clusterAdminPassword:"sample@123"})
Please provide the password for 'root@CSE-DBE-2741:3309': *******
Save password for 'root@CSE-DBE-2741:3309'? [Y]es/[N]o/Ne[v]er (default No): no
Configuring MySQL instance at 127.0.0.1:3309 for use in an InnoDB cluster...

This instance reports its own address as 127.0.0.1:3309
Assuming full account name 'sample'@'%' for sample

applierWorkerThreads will be set to the default value of 4.

The instance '127.0.0.1:3309' is valid to be used in an InnoDB cluster.

Creating user sample@%.
Account sample@% was successfully created.

The instance '127.0.0.1:3309' is already ready to be used in an InnoDB cluster.

Successfully enabled parallel appliers.
```

```
 MySQL  JS > dba.configureInstance("root@CSE-DBE-2742:3305",{clusterAdmin:"sample",clusterAdminPassword:"sample@123"})
Please provide the password for 'root@CSE-DBE-2742:3305': *******
Save password for 'root@CSE-DBE-2742:3305'? [Y]es/[N]o/Ne[v]er (default No): no
Configuring MySQL instance at 127.0.0.1:3305 for use in an InnoDB cluster...

This instance reports its own address as 127.0.0.1:3305
Assuming full account name 'sample'@'%' for sample

applierWorkerThreads will be set to the default value of 4.

The instance '127.0.0.1:3305' is valid to be used in an InnoDB cluster.

Creating user sample@%.
Account sample@% was successfully created.

The instance '127.0.0.1:3305' is already ready to be used in an InnoDB cluster.

Successfully enabled parallel appliers.
```

```
 MySQL  JS > dba.configureInstance("root@localhost:3307",{clusterAdmin:"sample",clusterAdminPassword:"sample@123"})
Please provide the password for 'root@localhost:3307': *******
Save password for 'root@localhost:3307'? [Y]es/[N]o/Ne[v]er (default No): no
Configuring local MySQL instance listening at port 3307 for use in an InnoDB cluster...
NOTE: Instance detected as a sandbox.
Please note that sandbox instances are only suitable for deploying test clusters for use within the same host.

This instance reports its own address as 127.0.0.1:3307
Assuming full account name 'sample'@'%' for sample

applierWorkerThreads will be set to the default value of 4.

The instance '127.0.0.1:3307' is valid to be used in an InnoDB cluster.

Creating user sample@%.
Account sample@% was successfully created.

The instance '127.0.0.1:3307' is already ready to be used in an InnoDB cluster.

Successfully enabled parallel appliers.
```

Initializing the InnoDB Cluster

Next, we connect the Shell to one of the instances we just configured, which will be the seed instance. The seed instance is the one that would hold the initial state of the database, which will be replicated to the other instances as they're added to the cluster.

```
 MySQL  JS > shell.connect("root@CSE-DBE-2741:3309")
Creating a session to 'root@CSE-DBE-2741:3309'
Please provide the password for 'root@CSE-DBE-2741:3309': *******
Save password for 'root@CSE-DBE-2741:3309'? [Y]es/[N]o/Ne[v]er (default No): no
Fetching schema names for auto-completion... Press ^C to stop.
Your MySQL connection id is 15
Server version: 8.0.28 MySQL Community Server - GPL
No default schema selected; type \use <schema> to set one.
<ClassicSession:root@CSE-DBE-2741:3309>
```

Next, create the InnoDB cluster:

```
MySQL  CSE-DBE-2741:3309 ssl  JS > cluster = dba.createCluster("myCluster");
A new InnoDB Cluster will be created on instance '127.0.0.1:3309'.

Validating instance configuration at CSE-DBE-2741:3309...

This instance reports its own address as 127.0.0.1:3309

Instance configuration is suitable.
NOTE: Group Replication will communicate with other members using '127.0.0.1:3309'. Use the localAddress option to overr
ide.

* Checking connectivity and SSL configuration...

Creating InnoDB Cluster 'myCluster' on '127.0.0.1:3309'...

Adding Seed Instance...
Cluster successfully created. Use Cluster.addInstance() to add MySQL instances.
At least 3 instances are needed for the cluster to be able to withstand up to
one server failure.

<Cluster:myCluster>
```

Now check the whether cluster is created or not on machine CSE-DBE-2741 on port 3309

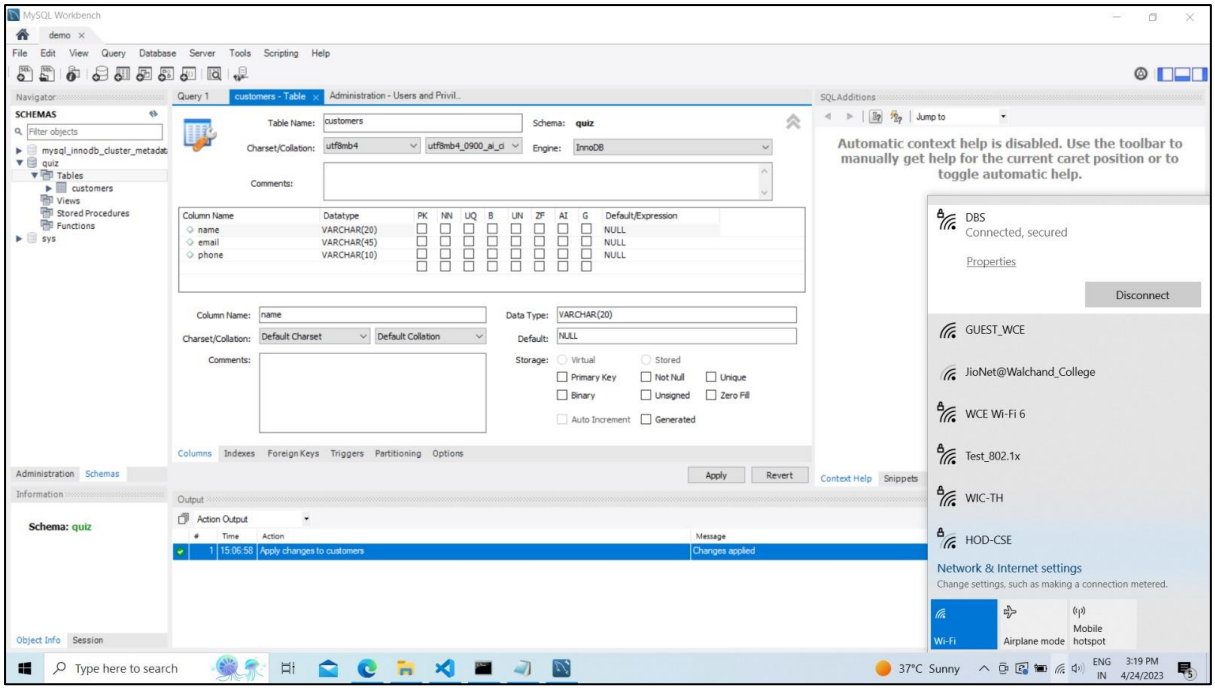
```
MySQL  JS > shell.connect("root@localhost:3309")
Creating a session to 'root@localhost:3309'
Please provide the password for 'root@localhost:3309': *******
Save password for 'root@localhost:3309'? [Y]es/[N]o/Ne[v]er (default No): no
Fetching schema names for autocompletion... Press ^C to stop.
Your MySQL connection id is 45
Server version: 8.0.28 MySQL Community Server - GPL
No default schema selected; type \use <schema> to set one.
<ClassicSession:root@localhost:3309>
```

```
MySQL  localhost:3309 ssl  JS > dba.getCluster()
<Cluster:myCluster>
```

We have successfully configured and the instance remotely and executed the new cluster

Now, next step is to start MYSQL server, create connection with seed node and cluster password and create databases and schemas.

Then, connect the cluster to application using connection string



Now connect any node to quiz database and check replication are happened correctly or not

```
C:\Users\DBE2740>mysqlsh root@10.4.2.151:3309/quiz
Please provide the password for 'root@10.4.2.151:3309': *******
Save password for 'root@10.4.2.151:3309'? [Y]es/[N]o/Ne[v]er (default No): no
MySQL Shell 8.0.33

Copyright (c) 2016, 2023, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Type '\help' or '\?' for help; '\quit' to exit.
Creating a session to 'root@10.4.2.151:3309/quiz'
Fetching schema names for auto-completion... Press ^C to stop.
Your MySQL connection id is 84
Server version: 8.0.28 MySQL Community Server - GPL
Default schema set to `quiz`.
```

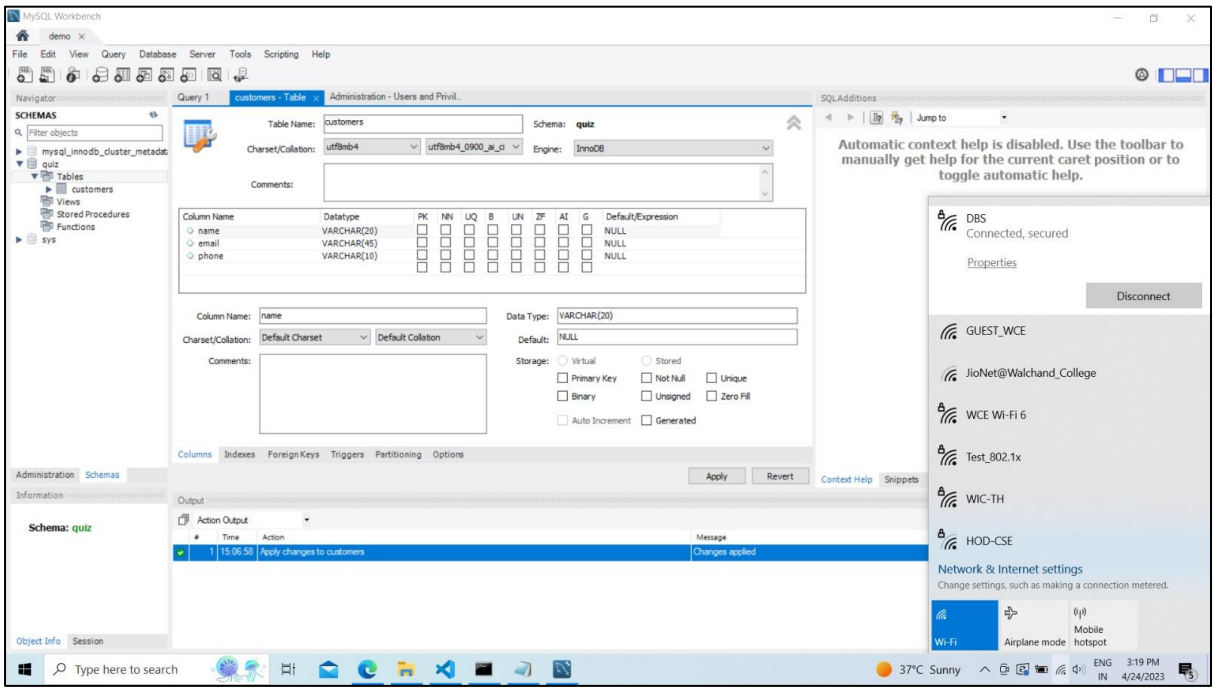
```
 MySQL  10.4.2.151:3309 ssl  quiz  JS > \sql
Switching to SQL mode... Commands end with ;
Fetching global names, object names from `quiz` for auto-completion... Press ^C to stop.
 MySQL  10.4.2.151:3309 ssl  quiz  SQL > select * from customers;
Empty set (0.0040 sec)
 MySQL  10.4.2.151:3309 ssl  quiz  SQL > desc customers;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| name  | varchar(20) | YES  |     | NULL    |       |
| email | varchar(45) | YES  |     | NULL    |       |
| phone | varchar(10) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
3 rows in set (0.0047 sec)
```

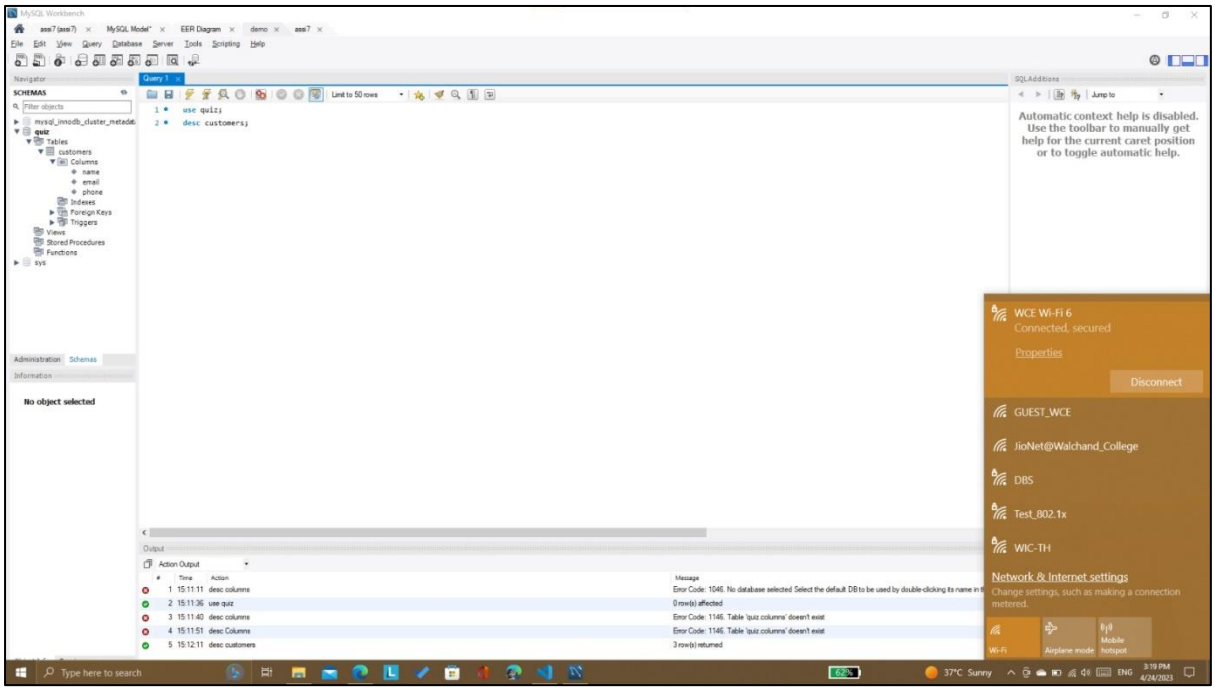
You can access this cluster setup over network to network.

Here under whole WCE network there are 2 networks which are

- WIFI6
- DBE

The InnoDB cluster created under WIFI6 network can be accessible over DBE network that is it provides high-availability and scaling features





Conclusion:

We have configured MYSQL InnoDB cluster by using three separate machines and demonstrated the actual working.