

Group B

Assignment 11

PROBLEM STATEMENT:

Create databases and tables, insert small amounts of data, and run simple queries using Impala

OBJECTIVE:

- To Learn and understand the steps to create databases and tables in Impala, insert data into these tables, and execute simple queries.
- To learn and understand how to use Impala to analyze data.

Theory:

Impalas

- A tool which we use to overcome the slowness of Hive Queries is what we call Impala.
- Syntactically Impala queries run very faster than Hive Queries even after they are more or less same as Hive Queries.
- It offers high-performance, low-latency SQL queries.
- Impala is the best option while we are dealing with medium-sized datasets and we expect the real-time response from our queries. However, make sure Impala is available only in Hadoop distribution.
- Since MapReduce store intermediate results in the file system, Impala is not built on MapReduce. Hence, it is very slow for real-time query processing.

In addition, Impala has its own execution engine. Basically, that stores the intermediate results in In-memory. Therefore, when compared to other tools which use MapReduce its query execution is very fast.

Some Key Points

- It offers high-performance, low-latency SQL queries.
- Moreover, to share databases and tables between both Impala and Hive it integrates very well with the Hive Metastore.
- Also, it is Compatible with HiveQL Syntax

Impala is an open-source SQL query engine allows you to query data stored in Apache Hadoop clusters. It's a fast, distributed, and highly scalable system that can run complex

queries on large datasets in near real-time. In this manual, we will guide you through the process of creating databases and tables, inserting small amounts of data, and running simple queries using Impala.

Impala uses a distributed architecture, which means that data is stored across multiple nodes in a cluster. Impala also uses a query engine that allows you to write SQL queries that are executed in parallel across the nodes in the cluster. This allows for fast query execution, even on large datasets.

Impala uses a metadata store to keep track of the databases and tables that are created. The metadata store is used to store information about the location of the data, the schema of the tables, and other metadata. Impala supports a wide range of data formats, including Parquet, Avro, and RCFile.

Data Types:

BIGINT :- This datatype stores numerical values and the range of this data type is - 9223372036854775808 to 9223372036854775807. This datatype is used in create table and alter table statements.

BOOLEAN:-This data type stores only true or false values and it is used in the column definition of create table statement.

CHAR:- This data type is a fixed length storage, it is padded with spaces, you can store up to the maximum length of 255.

DECIMAL:- This data type is used to store decimal values and it is used in create table and alter table statements.

DOUBLE:- This data type is used to store the floating point values in the range of positive or negative 4.94065645841246544e-324d -1.79769313486231570e+308.

FLOAT :- This data type is used to store single precision floating value datatypes in the range of positive or negative 1.40129846432481707e-45 .. 3.40282346638528860e+38.

Example:

To create a database in Impala, you can use the following SQL command:

```
CREATE DATABASE my_database;
```

This will create a new database called "my_database". To create a table within this database, you can use the following SQL command:

```
CREATE TABLE my_table ( id INT, name STRING, age INT );
```

This will create a new table called "my_table" with three columns: id, name, and age. To insert data into this table, you can use the following SQL command:

```
INSERT INTO my_table VALUES (1, 'John', 25), (2, 'Jane', 30), (3, 'Bob', 40);
```


This will insert three rows into the table, each with a unique id, name, and age. To run a simple query on this table, you can use the following SQL command:

```
SELECT * FROM my_table;
```

This will return all the rows in the table. You can also use more complex queries, such as aggregations and joins, to analyze the data in the table.

Conclusion:

Impala is a powerful SQL query engine that can be used to analyze large datasets stored in Apache Hadoop clusters. We have seen how to create databases and tables, insert data, and execute simple queries in Impala.

```
1 import java.sql.DriverManager
2
3 object MySQLExample {
4   def main(args: Array[String]): Unit = {
5     val url = "jdbc:mysql://localhost:3306/"
6     val username = "root"
7     val password = "Mrp@2004" //  Replace with your MySQL root password
8
9     // Register the driver
10    Class.forName("com.mysql.cj.jdbc.Driver")
11
12    val connection = DriverManager.getConnection(url, username, password)
13    val statement = connection.createStatement()
14
15    // Create database
16    statement.execute("CREATE DATABASE IF NOT EXISTS student_db")
17    statement.execute("USE student_db")
18
19    // Create table
20    statement.execute(
21      """CREATE TABLE IF NOT EXISTS students (
22        | id INT PRIMARY KEY,
23        | name VARCHAR(50),
24        | age INT,
25        | department VARCHAR(50)
26        |)""").stripMargin
27    )
28
29    statement.execute("DELETE FROM students")
30
31
32    // Insert 10 Indian student records
33    statement.execute("INSERT INTO students VALUES (1, 'Mangesh Patil',
34      21, 'AI&DS')")
35    statement.execute("INSERT INTO students VALUES (2, 'Sneha Iyer', 22, '
36      CSE')")
37    statement.execute("INSERT INTO students VALUES (3, 'Raj Verma', 20, '
38      ECE')")
39    statement.execute("INSERT INTO students VALUES (4, 'Anjali Sharma',
40      23, 'IT')")
41    statement.execute("INSERT INTO students VALUES (5, 'Arjun Reddy', 21
42      , 'ME')")
43    statement.execute("INSERT INTO students VALUES (6, 'Priya Deshmukh
44      ', 22, 'CSE')")
```

```

39 statement.execute("INSERT INTO students VALUES (7, 'Rohan Gupta', 24
, 'EE')")
40 statement.execute("INSERT INTO students VALUES (8, 'Neha Joshi', 20, '
AI&DS')")
41 statement.execute("INSERT INTO students VALUES (9, 'Aman Singh', 23, '
ECE')")
42 statement.execute("INSERT INTO students VALUES (10, 'Kavya Nair', 21, '
IT')")
43
44 // Query data
45 val rs = statement.executeQuery("SELECT * FROM students")
46 println("ID\tName\t\tAge\tDepartment")
47 while (rs.next()) {
48     println(s"${rs.getInt("id")}\t${rs.getString("name")}\t${rs.getInt("age")}\t${rs.
getString("department")}")
49 }
50
51 // Clean up
52 rs.close()
53 statement.close()
54 connection.close()
55 }
56 }
57
58 // Output:
59
60 PS C:\Users\91705\Desktop\TE LABS\SL-3\scala-mysql-jdbc> sbt run
61 [info] welcome to sbt 1.9.7 (Oracle Corporation Java 21.0.5)
62 [info] loading project definition from C:\Users\91705\Desktop\TE LABS\SL-3\
scala-mysql-jdbc\project
63 [info] loading settings for project scala-mysql-jdbc from build.sbt ...
64 [info] set current project to ScalaMySQL (in build file:/C:/Users/91705/Desktop
/TE%20LABS/SL-3/scala-mysql-jdbc/)
65 [info] compiling 1 Scala source to C:\Users\91705\Desktop\TE LABS\SL-3\
scala-mysql-jdbc\target\scala-2.13\classes ...
66 [info] running MySQLExample
67 ID    Name           Age  Department
68 1    Mangesh Patil  21   AI&DS
69 2    Sneha Iyer    22   CSE
70 3    Raj Verma     20   ECE
71 4    Anjali Sharma  23   IT
72 5    Arjun Reddy   21   ME
73 6    Priya Deshmukh 22   CSE
74 7    Rohan Gupta   24   EE

```

```
75 8    Neha Joshi    20    AI&DS
76 9    Aman Singh    23    ECE
77 10   Kavya Nair     21    IT
78 [success] Total time: 4 s, completed 11-Apr-2025, 7:07:42ΓÇ»pm
79
```