

BIG DATA MANAGEMENT

Assignment IV – Google Bigtable Integration Using Java

Submitted To:

Dr. Dip Shankar Banerjee

Associate Professor

School of AI and Data Engineering

Indian Institute of Technology Jodhpur – Rajasthan

Submitted By:

Name: Sree Charan

Roll Number: G24AI1095

Email: G24AI1095@iij.ac.in

Course: PGD-Trimester 3

Date: 02/07/2025

Index

1. Introduction
2. Environment Setup and Tools
3. Connecting to Google Bigtable
4. Schema Design and Table Creation
5. Data Loading Strategy
6. Query Implementations
7. Challenges Faced
8. Conclusion
9. GitHub Link

1. Introduction

Google Bigtable is a fully managed, high-performance NoSQL database service suitable for massive-scale workloads. This assignment aimed to give hands-on experience in integrating Java with Google Bigtable to design a weather-monitoring data system for three different cities. The objective included loading large datasets, organizing data by row keys and column families, and executing structured queries to retrieve meaningful insights such as temperature patterns and extreme weather observations.

2. Environment Setup and Tools

We used Java 11 and Eclipse IDE for application development, along with Maven to handle dependencies. Google Cloud SDK was configured for project authentication and deployment. The Bigtable instance was set up on Google Cloud Platform, where we enabled the Bigtable Admin and Data APIs, and authenticated the local environment using `gcloud auth application-default login`. This setup allowed our Java code to communicate securely with the Bigtable instance.

3. Connecting to Google Bigtable

We initialized a connection using the Bigtable client libraries provided by Google. Two main clients were used: `BigtableDataClient` for performing read/write operations and `BigtableTableAdminClient` for managing tables. Proper exception handling and logging were implemented to verify successful authentication and connectivity.

Code :

```
public void connect() throws Exception {
    BigtableDataSettings dataSettings = BigtableDataSettings.newBuilder()
        .setProjectId("bigdata-429910")
        .setInstanceId("bigtable")
        .build();
    BigtableDataClient dataClient = BigtableDataClient.create(dataSettings);
    System.out.println("Data client connected.");

    BigtableAdminSettings adminSettings = BigtableAdminSettings.newBuilder()
        .setProjectId("bigdata-429910")
        .setInstanceId("bigtable")
        .build();
    BigtableAdminClient adminClient =
        BigtableAdminClient.create(adminSettings);
    System.out.println("Admin client connected.");
}
```

Image:

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "BigDataAssignmentFour". It contains a "src" directory with a "main" package containing "Main.java" and "Bigtable.java".
- Main.java:** Contains the provided Java code for connecting to Bigtable and creating a table.
- Pom.xml:** Configuration file for the project.
- Run Tab:** Shows the output of running the application. It includes a warning about problematic credentials, a "Data client connected" message, a "Creating table: weather" message, and a "Table 'weather' created successfully." message. The process finished with exit code 0.

4. Schema Design and Table Creation

We created a table named `weather` with a column family `sensor`, which stores attributes such as temperature, pressure, humidity, and wind speed. The row key followed the format `station#YYYY-MM-DD-HH` to support fast retrieval by station and timestamp. This schema is designed to optimize scanning and filtering of time-based weather data.

Code:

```
public void createTable() {
    String tableName = "weather"; // Or pass as parameter
    if (adminClient.tableExists(tableName)) {
        System.out.println("Table " + tableName + " already exists.");
        return;
    }
    System.out.println("Creating table: " + tableName);
    CreateTableRequest createTableRequest = CreateTableRequest.of(tableName)
        .addFamily("data");
    adminClient.createTable(createTableRequest);
    System.out.println("Table " + tableName + " created successfully.");
}
```

Image:

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "BigDataAssignmentFour". It contains a "src" directory with "main" and "java" sub-directories. The "java" directory contains a package "org.example" with a class "Bigtable". There are also CSV files ("portland.csv", "seatac.csv", "vancouver.csv") and a "pom.xml" file.
- Code Editor:** The code editor shows the "Bigtable.java" file with the provided Java code for creating a table.
- Terminal:** A terminal window at the bottom shows the execution of the Java code. The output includes a warning about problematic credentials, a message about connecting to a data client, and a confirmation that the table "weather" already exists. It also shows the loading of data from three CSV files: seatac.csv, vancouver.csv, and portland.csv.

5. Data Loading Strategy

Data from three CSV files (SeaTac, Vancouver, Portland) was read using buffered file streams. Each line of data was parsed into relevant weather attributes, and mutations were created for each row. Bulk writing was performed to optimize throughput using `BulkMutation`. The loading logic ensured that malformed lines were skipped and exceptions were logged.

Code:

```
public void loadData() throws Exception {
    loadCSV("seatac.csv", "SEA");
    loadCSV("vancouver.csv", "YVR");
    loadCSV("portland.csv", "PDX");
}

private void loadCSV(String fileName, String stationCode) throws IOException {
    System.out.println("Loading data from:" + fileName)
    BufferedReader reader = new BufferedReader(new FileReader("/bin/data/" +
fileName));
    String line = reader.readLine(); // Skip header
    HashSet < String > seen = new HashSet < > ();
    int count = 0;
    while ((line = reader.readLine()) != null) {
        String[] parts = line.split(",");
        if (parts.length < 8) continue;
        String date = parts[0].trim();
        String time = parts[1].trim();
        String hour = time.split(":")[0];
        String rowKey = stationCode + "#" + date + "#" + hour;
        if (seen.contains(rowKey)) continue; // First row per hour only
        seen.add(rowKey);
        try {
            RowMutation mutation = RowMutation.create(tableId, rowKey)
                .setCell(COLUMN_FAMILY, "temperature", parts[2].trim())
                .setCell(COLUMN_FAMILY, "dewpoint", parts[3].trim())
                .setCell(COLUMN_FAMILY, "humidity", parts[4].trim())
                .setCell(COLUMN_FAMILY, "windspeed", parts[5].trim())
                .setCell(COLUMN_FAMILY, "pressure", parts[6].trim());
            dataClient.mutateRow(mutation);
        } catch (Exception e) {
            System.out.println("Error processing row: " + rowKey);
        }
    }
}
```

```

        count++;
        if (count % 100 == 0) {
            System.out.println("Inserted " + count + " rows so far...");
        }
    } catch (Exception e) {
        System.err.println("Failed to insert row: " + rowKey + " → " +
e.getMessage());
    }
}
reader.close();
System.out.println("Finished loading: " + fileName + " | Total rows: " + count);
}

```

Image:

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "BigDataAssignmentFour". It contains a "src" directory with "main" and "test" packages. The "main" package has a "java" directory containing three CSV files: "portland.csv", "seatac.csv", and "vancouver.csv". It also contains a "Bigtable" class and a "resources" directory.
- Code Editor:** The "Main.java" file is open. The code defines a `Bigtable` class with a static main method that creates a `Bigtable` object, connects to it, creates a table, loads data from three CSV files, and then closes the connection.
- Run Tab:** The "Bigtable" run configuration is selected. The output window shows the command being run: `/usr/lib/jvm/default-java/bin/java ...` and the application's log output.
- Output Window:** The log output shows the application connecting to Google Cloud Bigtable, creating a table named "weather", and loading data from three CSV files: "seatac.csv", "vancouver.csv", and "portland.csv". It also indicates that the table already exists.
- Status Bar:** The status bar at the bottom right shows the file count (108), line count (LF), character encoding (UTF-8), and indentation settings (4 spaces).

6. Query Implementations

Query 1

Retrieve the temperature recorded in Vancouver on 2022-10-01 at 10 a.m. using direct row key access.

Code:

```
public int query1() throws Exception {
    String rowKey = "YVR#2022-10-01#10";
    Row row = dataClient.readRow(tableId, rowKey);
    if (row != null) {
        List < RowCell > cells = row.getCells(COLUMN_FAMILY, "temperature");
        if (!cells.isEmpty()) {
            String tempStr = cells.get(0).getValue().toStringUtf8();
            return Integer.parseInt(tempStr);
        }
    }
    return -999;
}
```

Image:

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "BigDataAssignmentFour". It contains a "src" directory with "main" and "java" sub-directories. Inside "java", there is a "bin.data" folder containing three CSV files: "portland.csv", "seatac.csv", and "vancouver.csv". There is also an "org.example" package containing a "Bigtable" class.
- Main.java:** A Java file with the following code:

```
public class Bigtable {
    public static void main(String[] args) throws Exception {
        Bigtable bt = new Bigtable();
        bt.connect();
        bt.createTable();
        bt.loadData();
        int temp = bt.query();
        System.out.println("Temperature at YVR on 2022-10-01 10AM: " + temp);
        bt.close();
    }

    public void connect() throws Exception {
        BigtableDataSettings dataSettings = BigtableDataSettings.newBuilder()
            .setProjectId(projectId)
            .setInstanceId(instanceId)
            .build();
        dataClient = BigtableDataClient.create(dataSettings);
        System.out.println("Data client connected.");
    }
}
```
- Run Output:** The terminal window shows the execution of the code. It prints the path to the Java executable, a warning about problematic credentials from Google Cloud SDK, and the temperature at YVR on 2022-10-01 10AM, which is 52.

Query 2

Find the highest wind speed recorded in Portland during the month of September 2022 using prefix scan and comparison.

Code:

```
public int query2() throws Exception {
    int maxWind = -1;
    String start = "PDX#2022-09-01";
    String end = "PDX#2022-09-30z"; // 'z' ensures inclusive end of month
    Query query = Query.create(tableId).range(start, end);
    for (Row row: dataClient.readRows(query)) {
        List < RowCell > windCells = row.getCells(COLUMN_FAMILY, "windspeed");
        if (!windCells.isEmpty()) {
            try {
                int val = Integer.parseInt(windCells.get(0).getValue().toStringUtf8());
                if (val > maxWind) maxWind = val;
            } catch (NumberFormatException ignored) {}
        }
    }
    return maxWind;
}
```

Image:

The screenshot shows the IntelliJ IDEA interface. The code editor displays the Java code for the `query2()` method. The terminal window at the bottom shows the execution of the code and its output. The output includes a warning about problematic credentials, a message indicating a data client connection, and the result of the query: "Max Windspeed in Portland Sept 2022: 25".

```
Jun 25 00:18 • 56% 2.0 3 KB/s 14% 50°C
Activities jetbrains-idea
Project BigDataAssignmentFour Version control
Main.java pom.xml (BigDataAssignmentFour) Bigtable.java portland.csv seatac.csv vancouver.csv
18 public class Bigtable {
19     public static void main(String[] args) throws Exception {
20         Bigtable bt = new Bigtable();
21         bt.connect();
22         bt.createTable();
23         bt.loadTable();
24         int temp = bt.query();
25         System.out.println("Temperature at VVR on 2022-10-01 10AM: " + temp);
26         int wind = bt.query2();
27         System.out.println("Max Windspeed in Portland Sept 2022: " + wind);
28         bt.close();
29     }
30
31     public void connect() throws Exception {
32         BigtableDataSettings dataSettings = BigtableDataSettings.newBuilder()
33             .setProjectId(projectId)
34             .setInstanceId(instanceId)
35             .setTableId(tableId)
36             .setRegionCode(regionCode)
37             .build();
38         BigtableDataClient dataClient = BigtableDataClient.create(dataSettings);
39         BigtableTableAdminClient adminClient = BigtableTableAdminClient.create(dataSettings);
40         bt.setDataClient(dataClient);
41         bt.setTableAdminClient(adminClient);
42     }
}
Jun 25, 2025 12:18:27 AM com.google.auth.oauth2.DefaultCredentialsProvider warnAboutProblematicCredentials
WARNING: Your application has authenticated using end user credentials from Google Cloud SDK. We recommend that most server applications use service accounts instead. If your application continues to use end user credentials, you may experience intermittent connectivity issues.
Data client connected.
Admin client connected.
Max Windspeed in Portland Sept 2022: 25
Connections closed.

Process finished with exit code 0
```

Query 3

Display all readings from SeaTac station on 2022-10-02 by scanning for the date-prefixed row keys.

Code:

```
public ArrayList < Object[] > query3() throws Exception {  
    System.out.println("Executing query #3.");  
    ArrayList < Object[] > data = new ArrayList < > ();  
    String prefix = "SEA#2022-10-02";  
    Query query = Query.create(tableId).prefix(prefix);  
    for (Row row: dataClient.readRows(query)) {  
        String[] parts = row.getKey().toStringUtf8().split("#");  
        String date = parts[1];  
        String hour = parts[2];  
        String temp = "", dew = "", hum = "", wind = "", press = "";  
        for (RowCell cell: row.getCells()) {  
            String col = cell.getQualifier().toStringUtf8();  
            String val = cell.getValue().toStringUtf8();  
            switch (col) {  
                case "temperature":  
                    temp = val;  
                    break;  
                case "dewpoint":  
                    dew = val;  
                    break;  
                case "humidity":  
                    hum = val;  
                    break;  
                case "windspeed":  
                    wind = val;  
                    break;  
                case "pressure":  
                    press = val;  
                    break;  
            }  
        }  
        Object[] record = new Object[] {  
            date,  
            hour,  
            Integer.parseInt(temp),  
            Integer.parseInt(dew),  
            Integer.parseInt(hum),  
            Integer.parseInt(wind),  
            Double.parseDouble(press)  
        };  
        data.add(record);  
    }  
    return data;  
}
```

```

        Integer.parseInt(dew),
        hum,
        wind,
        press
    };
    data.add(record);
}
return data;
}

```

Image:

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "BigDataAssignmentFour". It contains a "src" directory with "main" and "bin.data" packages. "bin.data" contains three CSV files: "portland.csv", "seatac.csv", and "vancouver.csv".
- Main.java:** This file contains the main method for the application.
- Bigtable.java:** This file contains the logic for connecting to a Bigtable database and querying it.
- Run Tab:** The "Bigtable" configuration is selected for running the application.
- Output Tab:** The output window shows the results of the application's execution, including logs and the output of the "readings" query.

```

public class Bigtable {
    public static void main(String[] args) throws Exception {
        Bigtable bt = new Bigtable();
        bt.connect();
        bt.createTable();
        bt.loadData();
        int temp = bt.query1();
        System.out.println("Temperature at YVR on 2022-10-01 10AM: " + temp);
        int wind = bt.query2();
        System.out.println("Max Windspeed in Portland Sept 2022: " + wind);
        ArrayList<Object> readings = bt.query3();
        System.out.println("Seatac Readings for 2022-10-02:");
        for (Object row : readings) {
            for (Object val : row) System.out.print(val + " ");
            System.out.println();
        }
        bt.close();
    }
}

```

```

Jun 25, 2025 12:34:10 AM com.google.auth.oauth2.DefaultCredentialsProvider warnAboutProblematicCredentials
WARNING: Your application has authenticated using end user credentials from Google Cloud SDK. We recommend that most server applications use service accounts instead. If your application continues to use end user credentials, please see https://cloud.google.com/docs/authentication/end-user-authentication#problematic_end_user_credentials.
data client connected.
Admin client connected.
Seatac Readings for 2022-10-02:
2022-10-02 00 74 53 47.8 9 1014.1
2022-10-02 01 69 53 56.7 7 1014.1
2022-10-02 02 67 53 68.7 7 1014.3
2022-10-02 03 66 53 62.9 7 1014.4
2022-10-02 04 64 53 67.4 7 1014.2
2022-10-02 05 63 52 67.3 7 1014.1
2022-10-02 06 61 52 72.2 8 1014.3
2022-10-02 07 65 51 64.8 9 1014.2
2022-10-02 08 61 53 74.9 4 1014.0
2022-10-02 09 59 52 77.5 8 1014.2
2022-10-02 10 58 52 80.4 0 1014.3
2022-10-02 11 55 51 86.3 3 1014.3
2022-10-02 12 57 52 83.3 4 1014.7
2022-10-02 13 56 52 84.3 1015.9

```

Query 4

Identify the highest temperature recorded in July and August 2022 to evaluate peak summer heat.

Code:

```
public Object[] query4() throws Exception {
    System.out.println("Executing query #4: Highest temperature in July or August 2022.");
    Query query = Query.create(tableId);
    int maxTemp = Integer.MIN_VALUE;
    String bestStation = "", bestDate = "", bestHour = "";
    for (Row row: dataClient.readRows(query)) {
        String rowKey = row.getKey().toStringUtf8(); // e.g., SEA#2022-07-12#14
        String[] parts = rowKey.split("#");
        if (parts.length < 3) continue;
        String station = parts[0];
        String date = parts[1];
        String hour = parts[2];
        // Check if month is July or August
        String[] dateParts = date.split("-");
        if (dateParts.length != 3) continue;
        int year = Integer.parseInt(dateParts[0]);
        int month = Integer.parseInt(dateParts[1]);
        if (year != 2022 || (month != 7 && month != 8)) continue;
        List<RowCell> tempCells = row.getCells(COLUMN_FAMILY,
        "temperature");
        if (!tempCells.isEmpty()) {
            String tempStr = tempCells.get(0).getValue().toStringUtf8();
            try {
                int temp = Integer.parseInt(tempStr);
                if (temp > maxTemp) {
                    maxTemp = temp;
                    bestStation = station;
                    bestDate = date;
                    bestHour = hour;
                }
            } catch (NumberFormatException e) {
                System.err.printf("Skipping invalid temperature '%s' at row %s\n", tempStr,
```

```

rowKey);
    }
}
}
Object[] result = new Object[] {
    bestStation,
    bestDate,
    bestHour,
    maxTemp
};
return result;
}

```

Image:

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "BigDataAssignmentFour". It contains a "src" directory with "main" and "java" sub-directories. The "java" directory contains "Main.java" and "Bigtable.java". There are also "bin" and "data" directories under "main", which contain "portland.csv", "seatac.csv", and "vancouver.csv".
- Main.java:** This file contains the main application logic. It imports `java.util.List` and `org.example.Bigtable`. The `main` method creates a `Bigtable` object, connects to it, loads data, queries for the highest temperature and windspeed, and prints the results.
- Bigtable.java:** This is a generated Java class representing a Bigtable client. It includes methods for connecting, querying, and closing the connection.
- Run Tab:** The "Run" tab is active, showing the command line output of the application's execution. The output shows the application connecting to Google Cloud Bigtable and printing the highest reading from each of the three CSV files.
- Bottom Status Bar:** The status bar at the bottom indicates the file is 45.38 KB, uses LF line endings, has 4 spaces indentation, and is in UTF-8 encoding.

```

Activities jetbrains-idea
Jun 25 00:46 • 55% 2.0 8 KB/s 13% 48°C
BigDataAssignmentFour Version control
Project
BigDataAssignmentFour - /Users/proj_intel/BigData_Projects/BigDataAssignmentFour
  Idea
  .mvn
  src
    main
      java
        bin.data
          portland.csv
          seatac.csv
          vancouver.csv
        org.example
          Bigtable
        resources
    test
  target
    .gitignore
    pom.xml
External Libraries
Scratches and Consoles

Main.java (BigDataAssignmentFour) Bigtable.java portland.csv seatac.csv vancouver.csv
public class Bigtable {
    public static void main(String[] args) throws Exception {
        Bigtable bt = new Bigtable();
        bt.connect();
        bt.createTable();
        bt.loadData();
        int temp = bt.query1();
        System.out.println("Temperature at VVR on 2022-07-01 10AM: " + temp);
        int wind = bt.query2();
        System.out.println("Max Windspeed in Portland Sept 2022: " + wind);
        ArrayList<Object[]> readings = bt.query3();
        System.out.println("Seatac Readings for 2022-07-02:");
        for (Object[] row : readings) {
            for (Object val : row) System.out.print(val + " ");
            System.out.println();
        }
        Object[] temps = bt.query4();
        System.out.println("Hottest reading: " + Arrays.toString(temps));
        bt.close();
    }

    public void connect() throws Exception {
        BigtableDataSettings dataSettings = BigtableDataSettings.newBuilder()
            .setProjectId("projetcid")
    }
}

Run Bigtable
Jun 25, 2025 12:45:37 AM com.google.auth.oauth2.DefaultCredentialsProvider warnAboutProblematicCredentials
WARNING: Your application has authenticated using end user credentials from Google Cloud SDK. We recommend that most server applications use service accounts instead. If your application continues to use end user credentials, Data client connected.
✓ Admin client connected.
■ Hottest reading: [SEA, 2022-07-01, 00:53, 81]
Connections closed.

Process finished with exit code 0
45.38 LF UTF-8 4 spaces ⌂

```

8. Challenges Faced

Some of the challenges faced during this assignment included properly authenticating the Bigtable client on the first run, ensuring that the CSV data was clean and in a consistent format, and tuning the schema design for optimal performance. Additionally, handling exceptions gracefully and implementing efficient scans required thoughtful coding and testing.

9. Conclusion

This assignment offered a deep dive into integrating cloud-based NoSQL solutions like Google Bigtable with Java. We gained practical skills in handling bulk data, designing scalable schemas, performing conditional queries, and debugging cloud-native applications. This experience builds a foundation for real-world big data applications involving high-velocity time-series data.

10. GitHub Link

GitHub Link: <https://github.com/Sreecharan162/Big-Data.git>