CSA3423-NoSQL

Demonstrate partitioning by inserting data with different partition keys

Cassandra Exp.4

Demonstrating Partitioning in Cassandra

1. Create a keyspace
2. Create a table with a partition key
3. Insert records with different partition keys
4. Observe how data groups by partitions
5. Query data by partition key

1. Create Keyspace

```
CREATE KEYSPACE IF NOT EXISTS partition_demo
WITH REPLICATION = {
   'class': 'SimpleStrategy',
   'replication_factor': 1
};
USE partition_demo;
```

```
cqlsh> CREATE KEYSPACE IF NOT EXISTS partition_demo
   ... WITH REPLICATION = {
   ...     'class': 'SimpleStrategy',
   ...     'replication_factor': 1
   ... };
cqlsh> USE partition_demo;
```

2. Create a Table with a Clear Partition Key

The table below uses **department** as the partition key.
All students in the same department will be stored in the same partition.

```
CREATE TABLE IF NOT EXISTS students_by_department (
   department text,
   student_id UUID,
   name text,
   year int,
   PRIMARY KEY (department, student_id)
);
```

```
cqlsh:partition_demo> CREATE TABLE IF NOT EXISTS students_by_department (
                  ...     department text,
                  ...     student_id UUID,
                  ...     name text,
                  ...     year int,
                  ...     PRIMARY KEY (department, student_id)
                  ... );
```

Explanation:

- **Partition Key:** department
- **Clustering Key:** student_id

Thus:

- A partition is created per department.
- Each partition stores multiple students belonging to that department.

3. Insert Data with Different Partition Keys

Department: Computer Science (Partition 1)

INSERT INTO students_by_department (department, student_id, name, year)

VALUES ('CS', uuid(), 'Arjun', 3);

```
cqlsh:partition_demo> INSERT INTO students_by_department (department, student_id, name, year)
                  ... VALUES ('CS', uuid(), 'Arjun', 3);
```

INSERT INTO students_by_department (department, student_id, name, year)

VALUES ('CS', uuid(), 'Meghana', 2);

```
cqlsh:partition_demo> INSERT INTO students_by_department (department, student_id, name, year)
                  ... VALUES ('CS', uuid(), 'Meghana', 2);
```

Department: Electronics (Partition 2)

INSERT INTO students_by_department (department, student_id, name, year)

VALUES ('ECE', uuid(), 'Shinchana', 1);

```
cqlsh:partition_demo> INSERT INTO students_by_department (department, student_id, name, year)
                  ... VALUES ('ECE', uuid(), 'Shinchana', 1);
```

INSERT INTO students_by_department (department, student_id, name, year)

VALUES ('ECE', uuid(), 'Rohit', 3);

```
cqlsh:partition_demo> INSERT INTO students_by_department (department, student_id, name, year)
                  ... VALUES ('ECE', uuid(), 'Rohit', 3);
```

Department: Mechanical (Partition 3)

INSERT INTO students_by_department (department, student_id, name, year)

VALUES ('ME', uuid(), 'Vishal', 4);

```
cqlsh:partition_demo> INSERT INTO students_by_department (department, student_id, name, year)
                  ... VALUES ('ME', uuid(), 'Vishal', 4);
```

INSERT INTO students_by_department (department, student_id, name, year)

VALUES ('ME', uuid(), 'Sahana', 2);

```
cqlsh:partition_demo> INSERT INTO students_by_department (department, student_id, name, year)
                   ... VALUES ('ME', uuid(), 'Sahana', 2);
```

These inserts create **three separate partitions**:

- Partition for key **CS**
- Partition for key **ECE**
- Partition for key **ME**

Each partition holds only its department's rows.

4. Query Data by Partition Key

Retrieve only CS students

SELECT * FROM students_by_department

WHERE department = 'CS';

```
cqlsh:partition_demo> SELECT * FROM students_by_department
                   ... WHERE department = 'CS';

 department | student_id                           | name    | year
------------+--------------------------------------+---------+------
         CS | e13ad99b-c559-40ef-aad5-ec9b4a82797e |   Arjun |
 3
         CS | f751db68-0d72-4197-90fc-c6a202300419 | Meghana |
 2
```

Retrieve only ECE students

SELECT * FROM students_by_department

WHERE department = 'ECE';

```
cqlsh:partition_demo> SELECT * FROM students_by_department
                   ... WHERE department = 'ECE';

 department | student_id                           | name      | year
------------+--------------------------------------+-----------+------
        ECE | 6d76831e-ff78-43e3-9999-84b175eac5c8 |     Rohit |
   3
        ECE | ea3731a7-b108-4066-b91f-746766ec95fd | Shinchana |
   1
```

Retrieve only ME students

SELECT * FROM students_by_department

WHERE department = 'ME';

```
cqlsh:partition_demo> SELECT * FROM students_by_department
                   ... WHERE department = 'ME';

 department | student_id                           | name    | year
------------+--------------------------------------+---------+------
         ME | 2c0d85a5-2d6f-4165-b3c8-faf4c0d31a20 | Vishal |    4
         ME | bbb19c89-3340-4cf8-9ae8-1b2753ce8e68 | Sahana |    2
```

Each query reads from a **single partition**, demonstrating how Cassandra organizes data based on the partition key.

5. Understanding the Partitioning Behavior

1. Partition Key Determines Location

Cassandra hashes the partition key:

hash(department)

This decides which node stores that partition.

2. All Rows in the Same Partition Are Stored Together

Example:
All students in **ECE** stay in one physical partition on a node.

3. Efficient Reads

Querying by partition key is efficient because Cassandra directly finds the partition without scanning the whole table.

4. Inserts Scale Automatically

Adding a new department automatically creates a new partition.

6. Verification Using System Tables (Optional)

List partitions for the table

SELECT * FROM system_schema.tables

WHERE keyspace_name='partition_demo';

```
cqlsh:partition_demo> SELECT * FROM system_schema.tables
                   ... WHERE keyspace_name='partition_demo';

 keyspace_name  | table_name            | additional_write_policy | allow_auto_snapshot | bloom_filter_fp_chance | caching
                                | cdc  | comment | compaction

                                                  | compression

   | crc_check_chance | dclocal_read_repair_chance | default_time_to_live | extensions | flags        | gc_grace_seconds | id
                               | incremental_backups | max_index_interval | memtable | memtable_flush_period_in_ms | min_index_in
terval | read_repair | read_repair_chance | speculative_retry

 partition_demo | students_by_department |                  99p |               null |                  0.01 | {'keys': 'ALL
', 'rows_per_partition': 'NONE'} | null |        | {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy',
'max_threshold': '32', 'min_threshold': '4'} | {'chunk_length_in_kb': '16', 'class': 'org.apache.cassandra.io.compress.LZ4Compres
sor'} |               1 |                          0 |
       0 |          {} | {'compound'} |          864000 | 00f088d0-0739-11f1-9122-cdf73f26f65f |               null |

     2048 |     null |                          0 |
   128 |    BLOCKING |                0 |              99p
```

```
(1 rows)
 partition_demo | students_by_department |        name |            none |             0x6e616d65 |       regular |       -1 | t
ext
 partition_demo | students_by_department |  student_id |             asc | 0x73747564656e745f6964 |    clustering |        0 | u
uid
 partition_demo | students_by_department |        year |            none |             0x79656172 |       regular |       -1 |
int
```

Inspect partition columns

SELECT * FROM system_schema.columns

WHERE keyspace_name='partition_demo'

AND table_name='students_by_department';

This confirms that **department** is indeed the partition key.