DATASTAX

DATASTAX

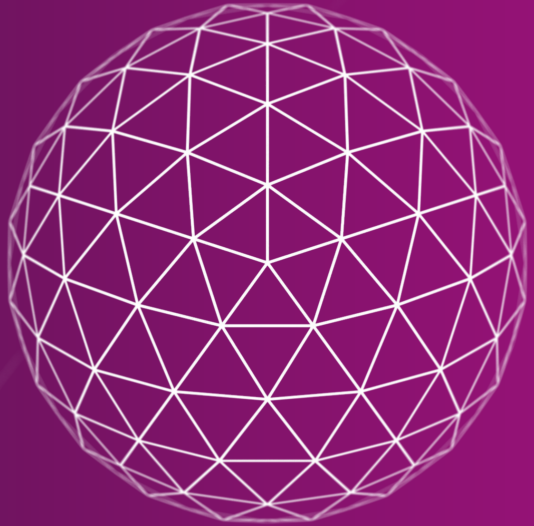
DATASTAX ACADEMY

# DataStax Enterprise

**Java Development with Apache Cassandra™**

**July 27th, 2023**

## ❯ #1 - Introduction



## Learning Objectives

- Building a Connection
- Simple Queries
- Simple Writes
- Prepared Statements
- BATCH updates

## ❯ Maven Dependencies

DATASTAX ACADEMY

**pom.xml:**

```xml
<dependency>
   <groupId>com.datastax.oss</groupId>
   <artifactId>java-driver-core</artifactId>
    <version>4.16.0</version>
</dependency>
```

## › Maven Dependencies

**pom.xml:**

```
<dependency>
    <groupId>com.datastax.astra</groupId>
     <artifactId>astra-spring-boot-starter</artifactId>
    <version>0.6</version>
</dependency>
```

## ❯ Building a connection

**Java:**

```
CqlSession session= CqlSession.builder()
                  .appContactPoints(endpointList)
                  .withKeyspace(keyspace)
                  .setAuthCredentials(user, pwd)
                  .withLocalDatacenter(datacenterName)
                  .build();
```

# Cassandra Connection

## Tips:

- Build and reuse one session object.
- Gracefully shutdown using finalize().
- Pass-in credentials as environment variables.
- Specify a default data center.
- Specify a default keyspace.

## › Simple Queries

DATASTAX ACADEMY

**Java:**

```java
String strCQL = "SELECT * FROM system.local LIMIT 1";
ResultSet rs = session.execute(strCQL);
Row localInfo = rs.one();

String address = localInfo.getString("rpc_address");
UUID hostId = localInfo.getUuid("host_id");
int port = localInfo.getInt("rpc_port");
```

## Prepared Statement Queries

**Java:**

```
String strCQL = "SELECT name, brand FROM product WHERE
product_group = ?";
PreparedStatement prepared = session.prepare(strCQL);
BoundStatement bound = prepared.bind(productGroup);
ResultSet rs = session.execute(bound);
List<Row> products = rs.all();
```

## ❯ Vector Search

Java:

```
PreparedStatement qvPrep = session.prepare(
   "SELECT * FROM product_vector ORDER BY
    product_vector ANN OF ? LIMIT 2;");
BoundStatement qvBound = qvPrep.bind(prd.getVector());
ResultSet rsV = session.execute(qvBound);
List<Row> ann = rsV.all();
```

## Cassandra Queries

**Tips:**

- **Name the columns in the SELECT clause.**
- **ALWAYS use a WHERE or a LIMIT clause.**
- **Never use ALLOW FILTERING.**
- **For repeated queries, use a prepared statement:**
    - **The prepared statement cache is your friend!**
    - **Prepare *outside* of the loop.**
    - **Execute *inside* of the loop.**

## ❯ Prepared Statement Writes

**Java:**

```
String strCQL = "INSERT INTO product (product_id,
        name, brand) VALUES (?,?,?)";
PreparedStatement prepared = session.prepare(strCQL);
BoundStatement bound = prepared.bind(productId,
        name, brand);
session.execute(bound);
```

## › Cassandra Batch – correct use

### Java:

```
UUID productId = UUID.randomUUID();


BoundStatement productBoundStatement1 = this.getProductInsertStatement(product, productId,
PRODUCT_BY_ID_TABLE_NAME);
BoundStatement productBoundStatement2 = this.getProductInsertStatement(product, productId,
PRODUCT_BY_NAME_TABLE_NAME);


BatchStatement batch = BatchStatement.newInstance(DefaultBatchType.LOGGED,
            productBoundStatement1,productBoundStatement2);


session.execute(batch);
```

## › Cassandra Batch – incorrect use

### CQL:

```
BEGIN BATCH
    INSERT INTO cycling.cyclist_name (id, lastname, firstname) VALUES
    (6d5f1663-89c0-45fc-8cfd-60a373b01622,'HOSKINS', 'Melissa');
    INSERT INTO cycling.cyclist_name (id, lastname, firstname) VALUES
    (38ab64b6-26cc-4de9-ab28-c257cf011659,'FERNANDES', 'Marcia');
    INSERT INTO cycling.cyclist_name (id, lastname, firstname) VALUES
    (9011d3be-d35c-4a8d-83f7-a3c543789ee7,'NIEWIADOMA', 'Katarzyna');
    INSERT INTO cycling.cyclist_name (id, lastname, firstname) VALUES
    (95addc4c-459e-4ed7-b4b5-472f19a67995,'ADRIAN', 'Vera');
 APPLY BATCH;
```

## › Cassandra Writes

**Tips:**

- **Don’t INSERT nulls!  Those are “tombstones.”**
- **Don’t run in-place updates.**
- **Don’t run many deletes.**
- **BATCH for one update across multiple tables - Good!**
- **BATCH for multiple updates across one table - Bad!**

❯ Cassandra Bulk Writes – correct

DATASTAX ACADEMY

## Java:

```
for (BoundStatement boundStatement : cqlInserts) {
    ResultSetFuture future = session.executeAsync(boundStatement);
    futures.add(future);
    threadCount++;
    if (threadCount > 19) {
        futures.forEach(ResultSetFuture::getUniterruptibly);
        futures = new ArrayList<>();
        threadCount = 0;
    }
}
```

**Iterate through a List<BoundStatement**

**Execute each statement**

**Add to List<ResultSetFuture>**

**If too many threads are in-flight, wait to ensure completion, restart threadCount and futures.**

## › Java Exercise

## Hands-on Exercise
Java

- Build a simple, restful product/recommendation service
- Support query by product_id
- Support query by vector ANN

## Exercises

Gitpod

https://github.com/aar0np/DS-Java-DSE