Batch Insert In Java – JDBC

viralpatel.net/blogs/batch-insert-in-java-jdbc/

By Viral Patel

Let's see how we can perform batch insert in Java using JDBC APIs. Although you might already knew this, I will try to explain the basic to a bit complex scenarios.

In this note, we will see how we can use JDBC APIs like `Statement` and `PreparedStatement` to insert data in any database in batches. Also we will try to explore scenarios where we can run out of memory and how to optimize the batch operation.

So first, the basic API to Insert data in database in batches using Java JDBC.

Simple Batch

I am calling this a simple batch. The requirement is simple. Execute a list of inserts in batch. Instead of hitting database once for each insert statement, we will using JDBC batch operation and optimize the performance.

Consider the following code:

Bad Code

```
String [] queries = {
 "insert into employee (name, city, phone) values ('A', 'X', '123')",
 "insert into employee (name, city, phone) values ('B', 'Y', '234')",
 "insert into employee (name, city, phone) values ('C', 'Z', '345')",
};

Connection connection = new getConnection();
Statement statement = connection.createStatement();

for (String query : queries) {
 statement.execute(query);
}
statement.close();
connection.close();
```

This is the BAD code. You are executing each query separately. This hits the database for each insert statement. Consider if you want to insert 1000 records. This is not a good idea.

We'll below is the basic code to perform batch insert. Check it out:

Good Code

```
Connection connection = new getConnection();
Statement statement = connection.createStatement();

for (String query : queries) {
 statement.addBatch(query);
}
statement.executeBatch();
```

```
statement.close();
connection.close();
```

Note how we used `addBatch()` method of Statement, instead of directly executing the query. And after adding all the queries we executed them in one go using `statement.executeBatch()` method. Nothing fancy, just a simple batch insert.

Note that we have taken the queries from a String array. Instead you may want to make it dynamically. For example:

```
import java.sql.Connection;
import java.sql.Statement;

//...

Connection connection = new getConnection();
Statement statement = connection.createStatement();

for (Employee employee: employees) {
 String query = "insert into employee (name, city) values('"
   + employee.getName() + "','" + employee.getCity + "')";
 statement.addBatch(query);
}
statement.executeBatch();
statement.close();
connection.close();
```

Note how we are creating query dynamically using data from Employee object and adding it in batch to insert in one go. Perfect! isn't it?

wait.. You must be thinking what about SQL Injection? Creating queries like this dynamically is very prone to SQL injection. And also the insert query has to be compiled each time.

Why not to use `PreparedStatement` instead of simple `Statement`. Yes, that can be the solution. Check out the below SQL Injection Safe Batch.

SQL Injection Safe Batch

Consider the following code:

```
import java.sql.Connection;
import java.sql.PreparedStatement;

//...

String sql = "insert into employee (name, city, phone) values (?, ?, ?)";
Connection connection = new getConnection();
PreparedStatement ps = connection.prepareStatement(sql);

for (Employee employee: employees) {

 ps.setString(1, employee.getName());
```

```
 ps.setString(2, employee.getCity());
 ps.setString(3, employee.getPhone());
 ps.addBatch();
}
ps.executeBatch();
ps.close();
connection.close();
```

Checkout the above code. Beautiful. We used `java.sql.PreparedStatement` and added insert query in the batch. This is the solution you must implement in your batch insert logic, instead of above `Statement` one.

Still there is one problem with this solution. Consider a scenario where you want to insert half million records into database using batch. Well, that may generate **OutOfMemoryError**:

```
java.lang.OutOfMemoryError: Java heap space

com.mysql.jdbc.ServerPreparedStatement$BatchedBindValues.<init>(ServerPreparedStatement.j
ava:72)
    com.mysql.jdbc.ServerPreparedStatement.addBatch(ServerPreparedStatement.java:330)

org.apache.commons.dbcp.DelegatingPreparedStatement.addBatch(DelegatingPreparedStatement.
a:171)
```

This is because you are trying to add everything in one batch and inserting once. Best idea would be to execute batch itself in batch. Check out the below solution.

Smart Insert: Batch within Batch

This is a simplest solution. Consider a batch size like 1000 and insert queries in the batches of 1000 queries at a time.

```
String sql = "insert into employee (name, city, phone) values (?, ?, ?)";
Connection connection = new getConnection();
PreparedStatement ps = connection.prepareStatement(sql);

final int batchSize = 1000;
int count = 0;

for (Employee employee: employees) {

 ps.setString(1, employee.getName());
 ps.setString(2, employee.getCity());
 ps.setString(3, employee.getPhone());
 ps.addBatch();

 if(++count % batchSize == 0) {
  ps.executeBatch();
 }
}
ps.executeBatch(); // insert remaining records
```

```
ps.close();
connection.close();
```

This would be the ideal solution. This avoids SQL Injection and also takes care of out of memory issue. Check how we have incremented a counter `count` and once it reaches `batchSize` which is 1000, we call `executeBatch()`.

Hope this helps.