* The Batch operation helps to bundle together a group of operations and execute them as a single unit.
* This avoids repeated database calls reducing the number of network calls to be made to the database.
* When executing a bunch of operations in a batch one or more operations could fail leading to an unstable state of the database; hence we are going to run the batch operations in **transaction units**. Think of it as atomic units. This would ensure that if any one of the operations in the batch fails, the whole batch fails. And if all operations in the batch succeed, the whole batch succeeds. To achieve this, the *autocommit* property of the connection object would be turned off and an explicit commit/rollback of the entire batch would be done finally.

**Below code has 1000 persons and commit every 50 persons in a batch**

String sqlQuery = "insert into PSERSONS values (?,?,?,?)";

int count = 0;

int batchSize = 50;

try{

connection.setAutoCommit(false);

PreparedStatement pstmt = connection.prepareStatement(SQL);

for(int i=1;i<=1000;i++){

pstmt.setString(1,"Java");

pstmt.setString(2,"CodeGeeks");

pstmt.setInt(3,i);

pstmt.setInt(4, i);

**pstmt.addBatch();**

count++;

if(count % batchSize == 0){

System.out.println("Commit the batch");

result = pstmt.executeBatch();

System.out.println("Number of rows inserted: "+ result.length);

**connection.commit();**

}

}

}catch(Exception e){

e.printStackTrace();

**connection.rollBack();**

} finally{

if(pstmt!=null)

pstmt.close();

if(connection!=null)

connection.close();

}

**Hibernate code for Batch Update**

Session session = factory.openSession();

Transaction tx = null;

Integer employeeID = null;

try{

tx = session.beginTransaction();

for ( int i=0; i<100000; i++ ) {

String fname = "First Name " + i;

String lname = "Last Name " + i;

Integer salary = i;

Employee employee = new Employee(fname, lname, salary);

session.save(employee);

if( i % 50 == 0 ) {

session.flush();

session.clear();

}

}

tx.commit();

}catch (HibernateException e) {

if (tx!=null) tx.rollback();

e.printStackTrace();

}finally {

session.close();

}