JDBC Batch insert update MySQL Oracle

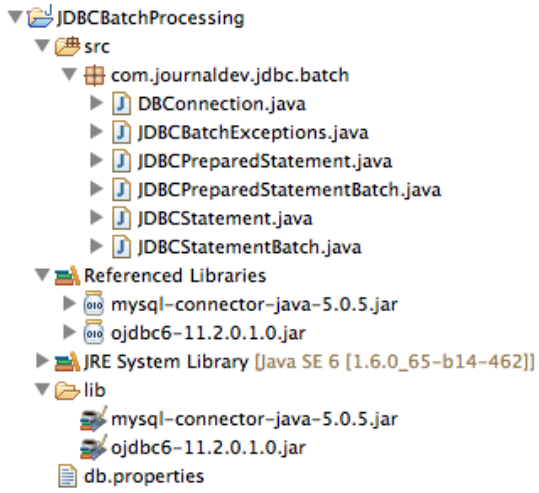
APRIL 6, 2018 BY PANKAJ — 17 COMMENTS

Today we will look into JDBC Batch insert and update examples in MySQL and Oracle databases. Sometimes we need to run bulk queries of similar kind for a database, for example loading data from CSV files to relational database tables. As we know that we have option to use `Statement` or `PreparedStatement` to execute queries. Apart from that JDBC provides Batch Processing feature through which we can execute bulk of queries in one go for a database.

JDBC Batch

JDBC batch statements are processed through Statement and PreparedStatement `addBatch()` and `executeBatch()` methods. This tutorial is aimed to provide details about JDBC Batch insert example for MySQL and Oracle database.

We will look into different programs so we have a project with structure as below image.



Notice that I have MySQL and Oracle DB JDBC Driver jars in the project build path, so that we can run our application across MySQL and Oracle DB both.

Let's first create a simple table for our test programs. We will run bulk of JDBC insert queries and look at the performance with different approaches.

```
--Oracle DB
CREATE TABLE Employee (
  empId NUMBER NOT NULL,
  name varchar2(10) DEFAULT NULL,
  PRIMARY KEY (empId)
);

--MySQL DB
CREATE TABLE `Employee` (
  `empId` int(10) unsigned NOT NULL,
  `name` varchar(10) DEFAULT NULL,
  PRIMARY KEY (`empId`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

We will read the Database configuration details from property file, so that switching from one database to another is quick and easy.

db.properties

```
#mysql DB properties
DB_DRIVER_CLASS=com.mysql.jdbc.Driver
DB_URL=jdbc:mysql://localhost:3306/UserDB
#DB_URL=jdbc:mysql://localhost:3306/UserDB?rewriteBatchedStatements=true
DB_USERNAME=pankaj
```

```
DB_PASSWORD=pankaj123

#Oracle DB Properties
#DB_DRIVER_CLASS=oracle.jdbc.driver.OracleDriver
#DB_URL=jdbc:oracle:thin:@localhost:1871:UserDB
#DB_USERNAME=scott
#DB_PASSWORD=tiger
```

Before we move into actual JDBC batch insert example to insert bulk data into Employee table, let's write a simple utility class to get the database connection.

DBConnection.java

```
package com.journaldev.jdbc.batch;

import java.io.FileInputStream;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Properties;

public class DBConnection {

	public static Connection getConnection() {
		Properties props = new Properties();
		FileInputStream fis = null;
		Connection con = null;
		try {
			fis = new FileInputStream("db.properties");
			props.load(fis);

			// load the Driver Class
			Class.forName(props.getProperty("DB_DRIVER_CLASS"));
```

Now let's look at different approach we can take for jdbc batch insert example.

1. Use Statement to execute one query at a time.

JDBCStatement.java

```
package com.journaldev.jdbc.batch;

import java.sql.Connection;
import java.sql.SQLException;
import java.sql.Statement;

public class JDBCStatement {

	public static void main(String[] args) {

		Connection con = null;
		Statement stmt = null;

		try {
			con = DBConnection.getConnection();
			stmt = con.createStatement();

			long start = System.currentTimeMillis();
			for(int i =0; i<10000;i++){
				String query = "insert into Employee values
("+i+",'Name"+i+"')";
				stmt.execute(query);
```

2. Use PreparedStatement to execute one query at a time.

JDBCPreparedStatement.java

```
package com.journaldev.jdbc.batch;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class JDBCPreparedStatement {

	public static void main(String[] args) {

		Connection con = null;
		PreparedStatement ps = null;
		String query = "insert into Employee (empId, name) values
(?,?)";
		try {
			con = DBConnection.getConnection();
			ps = con.prepareStatement(query);
```

```
                        long start = System.currentTimeMillis();
                        for(int i =0; i<10000;i++){
                                ps.setInt(1, i);
                                ps.setString(2, "Name"+i);
```

This approach is similar as using Statement but PreparedStatement provides performance benefits and avoid SQL injection attacks.

3. Using Statement Batch API for bulk processing.

JDBCStatementBatch.java

```
package com.journaldev.jdbc.batch;

import java.sql.Connection;
import java.sql.SQLException;
import java.sql.Statement;

public class JDBCStatementBatch {

        public static void main(String[] args) {

                Connection con = null;
                Statement stmt = null;

                try {
                        con = DBConnection.getConnection();
                        stmt = con.createStatement();

                        long start = System.currentTimeMillis();
                        for(int i =0; i<10000;i++){
                                String query = "insert into Employee values
("+i+",'Name"+i+"')";
                                stmt.addBatch(query);
```

We are processing 10,000 records with batch size of 1000 records. Once the batch size reaches, we are executing it and continue processing remaining queries.

4. Using PreparedStatement Batch Processing API for bulk queries.

JDBCPreparedStatementBatch.java

```
package com.journaldev.jdbc.batch;
```

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

public class JDBCPreparedStatementBatch {

        public static void main(String[] args) {

                Connection con = null;
                PreparedStatement ps = null;
                String query = "insert into Employee (empId, name) values
(?,?)";
                try {
                        con = DBConnection.getConnection();
                        ps = con.prepareStatement(query);

                        long start = System.currentTimeMillis();
                        for(int i =0; i<10000;i++){
                                ps.setInt(1, i);
                                ps.setString(2, "Name"+i);
```

Let's see how our programs work with MySQL database, I have executed them separately multiple times and below table contains the results.

MYSQL DB	STATEMENT	PREPAREDSTATEMENT	STATEMENT BATCH	PREPAREDSTATEMENT BATCH
Time Taken (ms)	8256	8130	7129	7019

When I looked at the response time, I was not sure whether it's right because I was expecting some good response time improvements with Batch Processing. So I looked online for some explanation and found out that by default MySQL batch processing works in similar way like running without batch. To get the actual benefits of Batch Processing in MySQL, we need to pass `rewriteBatchedStatements` as TRUE while creating the DB connection. Look at the MySQL URL above in **db.properties** file for this.

With `rewriteBatchedStatements` as `true`, below table provides the response time for the same programs.

MYSQL DB	STATEMENT	PREPAREDSTATEMENT	STATEMENT BATCH	PREPAREDSTATEMENT BATCH
Time Taken (ms)	5676	5570	3716	394

As you can see that PreparedStatement Batch Processing is very fast when rewriteBatchedStatements is true. So if you have a lot of batch processing involved, you should use this feature for faster processing.

Oracle Batch Insert

When I executed above programs for Oracle database, the results were in line with MySQL processing results and PreparedStatement Batch processing was much faster than any other approach.

JDBC Batch Processing Exceptions

Let's see how batch programs behave incase one of the queries throw exceptions.

JDBCBatchExceptions.java

```
package com.journaldev.jdbc.batch;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.Arrays;

public class JDBCBatchExceptions {

	public static void main(String[] args) {

		Connection con = null;
		PreparedStatement ps = null;
		String query = "insert into Employee (empId, name) values (?,?)";
		try {
			con = DBConnection.getConnection();

			ps = con.prepareStatement(query);

			String name1 = "Pankaj";
			String name2="Pankaj Kumar"; //longer than column length
			String name3="Kumar";
```

When I executed above program for MySQL database, I got below exception and none of the records were inserted in the table.

```
com.mysql.jdbc.MysqlDataTruncation: Data truncation: Data too long for column 'name'
at row 2
	at com.mysql.jdbc.MysqlIO.checkErrorPacket(MysqlIO.java:2939)
	at com.mysql.jdbc.MysqlIO.sendCommand(MysqlIO.java:1623)
	at com.mysql.jdbc.MysqlIO.sqlQueryDirect(MysqlIO.java:1715)
	at com.mysql.jdbc.Connection.execSQL(Connection.java:3249)
```

```
        at
com.mysql.jdbc.PreparedStatement.executeInternal(PreparedStatement.java:1268)
        at com.mysql.jdbc.PreparedStatement.executeUpdate(PreparedStatement.java:1541)
        at com.mysql.jdbc.PreparedStatement.executeUpdate(PreparedStatement.java:1455)
        at com.mysql.jdbc.PreparedStatement.executeUpdate(PreparedStatement.java:1440)
        at
com.mysql.jdbc.PreparedStatement.executeBatchedInserts(PreparedStatement.java:1008)
        at com.mysql.jdbc.PreparedStatement.executeBatch(PreparedStatement.java:908)
        at
com.journaldev.jdbc.batch.JDBCBatchExceptions.main(JDBCBatchExceptions.java:37)
```

When executed the same program for Oracle database, I got below exception.

```
java.sql.BatchUpdateException: ORA-12899: value too large for column
"SCOTT"."EMPLOYEE"."NAME" (actual: 12, maximum: 10)

        at
oracle.jdbc.driver.OraclePreparedStatement.executeBatch(OraclePreparedStatement.java:1007

        at
oracle.jdbc.driver.OracleStatementWrapper.executeBatch(OracleStatementWrapper.java:213)

        at
com.journaldev.jdbc.batch.JDBCBatchExceptions.main(JDBCBatchExceptions.java:38)
```

But the rows before exception were inserted into the database successfully. Although the exception clearly says what the error is but it doesn't tell us which query is causing the issue. So either we validate the data before adding them for batch processing or we should use JDBC Transaction Management to make sure all or none of the records are getting inserted incase of exceptions.

Same program with jdbc transaction management looks like below.

JDBCBatchExceptions.java

```
package com.journaldev.jdbc.batch;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.Arrays;
```

```
public class JDBCBatchExceptions {

	public static void main(String[] args) {

		Connection con = null;
		PreparedStatement ps = null;
		String query = "insert into Employee (empId, name) values (?,?)";
		try {
			con = DBConnection.getConnection();
			con.setAutoCommit(false);

			ps = con.prepareStatement(query);

			String name1 = "Pankaj";
```

As you can see that I am rolling back the transaction if any SQL exception comes. If the batch processing is successful, I am explicitly committing the transaction.

That's all for JDBC Batch insert update example, make sure to experiment with your data to get the optimal value of batch size for bulk queries.

One of the limitation of jdbc batch processing is that we can't execute different type of queries in the batch.

« PREVIOUS
JDBC Statement vs PreparedStatement – SQL

NEXT »
CallableStatement in Java Example

Injection Example

About Pankaj

If you have come this far, it means that you liked what you are reading. Why not reach little more and connect with me directly on **Google Plus**, **Facebook** or **Twitter**. I would love to hear your thoughts and opinions on my articles directly.

Recently I started creating video tutorials too, so do check out my videos on **Youtube**.

FILED UNDER: DATABASE, JAVA

Comments

latha says
JUNE 13, 2018 AT 10:27 PM
if we have 100 inserts added to batch and run execute batch it will create 100 Connections?
Reply

Pankaj says
JUNE 14, 2018 AT 10:44 AM
Obviously not, it will use the same connection to execute the batch of queries.
Reply

Shiva says
FEBRUARY 1, 2016 AT 8:23 AM
good one. Thanks for putting together great examples. Good Job!
Reply

menberbrains says
JANUARY 7, 2016 AT 11:50 AM
Very good article and very articulate. Thanks very much Pankaj

Reply

vu says

OCTOBER 27, 2015 AT 1:25 AM

Dear Pankaj,

I am 74 years old. After a long searching in many Web sites, suddenly your web site appears before my eyes with the article that you wrote and that I need to my study (the study is for my personal satisfaction), an article rich in details and clearly explicated if compared with many others Web sites which I have read before. Thank a lot.

Vu in France.

Reply

Mastanvali says

JULY 2, 2015 AT 11:12 PM

will this statement "rewriteBatchedStatements=true" work for Sybase database?

I could see batch statements are not working properly in Sybase database , Is sybase database is familiar with batch statements?

Reply

sushil says

APRIL 29, 2015 AT 8:19 AM

you are quite brilliant,thanks for posting such valuable things, your all interview question are fabulous like Struts 2,hibernate,spring… can you please post some cheat sheet of core spring and mvc.

At last Good work Hats Off

Reply

Deepti says

MARCH 12, 2015 AT 11:54 AM

Thanks for deep and clear cut explanation…….. cleared the concept ……..

Reply

Binh Thanh Nguyen says

DECEMBER 8, 2014 AT 10:26 PM

Thanks, nice post
Reply

satish says
OCTOBER 10, 2014 AT 10:41 PM
Good Job.. I Have a doubt that How to insert three Different table insertion in the same Batch Process to Execute it.
Reply

Ajijul says
JUNE 10, 2014 AT 4:57 AM
Nice suggestion. Great job.
Reply

Arpan says
MAY 7, 2014 AT 6:35 AM
Hi Pankaj, thanks for the detailed code examples. One quick question, I want to process 1 million records from an XML/Flat file in a batch of lets say 10000 each. And I want to log the not processed (inserted/modified) records i.e. which are failed. But I don't want to roll back, my batch program should continue to run and finish the XML/Flat file processing. So my requirement is to process all records and at the same time log the failed ones to report later.
Thanks
Arpan Ray
Reply

Bryan says
FEBRUARY 13, 2014 AT 8:24 PM
What about a comparison with INSERT INTO table (a, b) VALUES ('val', 'val2'), ('val', 'val2'), ('val', 'val2');
I've read that a single insert with multiple value sets are 10x faster than a batch. Have you tried or had any luck with that?
Reply

Denys says
JANUARY 6, 2017 AT 3:58 PM

http://stackoverflow.com/questions/26307760/mysql-and-jdbc-with-rewritebatchedstatements-true

Reply

ujwwala says

JANUARY 30, 2014 AT 7:41 AM

I have few queries as how I should set batch size like sometimes we get 1 recors or 100 records ,Please explain

Reply

Pankaj says

JANUARY 30, 2014 AT 10:55 AM

You can check the number of records and write logic for the batch size. Usually batch programming helps in bulk data processing so for 100 records it wont have much benefit though.

Reply

Siddu says

JANUARY 24, 2014 AT 6:46 AM

Can you please post article on DataSource class..

How to implement connection pooling in core java applications.

Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Save my name, email, and website in this browser for the next time I comment.

POST COMMENT

Search for tutorials...

DOWNLOAD ANDROID APP

CORE JAVA TUTORIAL

Java 10 Tutorials Java 9 Tutorials Java 8 Tutorials Java 7 Tutorials Core Java Basics OOPS Concepts Data Types and Operators String Manipulation Java Arrays Annotation and Enum Java Collections Java IO Operations Java Exception Handling MultiThreading and Concurrency Regular Expressions Advanced Java Concepts

RECOMMENDED TUTORIALS

Java Tutorials

› Java IO

- Java Regular Expressions
- Multithreading in Java
- Java Logging
- Java Annotations
- Java XML
- Collections in Java
- Java Generics
- Exception Handling in Java
- Java Reflection
- Java Design Patterns
- JDBC Tutorial

Java EE Tutorials

- Servlet JSP Tutorial
- Struts2 Tutorial
- Spring Tutorial
- Hibernate Tutorial
- Primefaces Tutorial
- Apache Axis 2
- JAX-RS
- Memcached Tutorial