- Введение
- Установка JDK
- Основные шаги
- Данные
- Порядок операций
- IDE NetBeans
- ООП
- Инкапсуляция
- Наследование
- Пакеты
- Переопределение и перегрузка
- Полиморфизм
- Статические свойства и методы
- Отношения между классами
- Визуализация робота
- Пример — очередь объектов
- Массивы — знакомство
- Многомерные массивы
- Абстрактные классы
- Интерфейсы

JDBC — групповые операции

Много запросов с помощью Batch-метода

В реальных проектах часто возникает ситуация, когда вам необходимо сделать очень много однотипных запросов (наиболее частов в этом случае встречается PreparedStatement) — например надо вставить несколько десятков или сотен записей. Если вы будете выполнять каждый запрос отдельно, то это будет достаточно долго и производительность вашего приложения будет невысокой. Для повышения производительности вы можете использовать batch-режим вставки. Он заключается в том, что вы накапливаете некоторый буфер своими запросами, а потом выполняете их сразу. В качестве примера приведу кусочек кода:

- Расширенное описание классов
- Исключения
- Решения на основе классов
- Список контактов — начало
- Коллекции — базовые принципы
- Коллекции — продолжение
- Список контактов — GUI приложение
- Что такое JAR-файлы
- Многопоточность — первые шаги
- Многопоточность и синхронизация
- Работаем с XML
- Reflection — основы
- Установка СУБД PostgreSQL
- Базы данных на Java — первые шаги
- Возможности JDBC — второй этап
- JDBC — групповые операции
- Список контактов — работаем с БД
- Переезжаем на Maven
- Потоки ввода-вывода
- Сетевое взаимодействие
- С чего начинается Web

```
PreparedStatement stmt = con.prepareStatement(
            "INSERT INTO jc_contact (first_name, last_name, phone, email) VALUES (?, ?, ?, ?)");

for (int i = 0; i < 10; i++) {
    // Заполняем параметры запроса
    stmt.setString(1, "FirstName_" + i);
    stmt.setString(2, "LastNAme_" + i);
    stmt.setString(3, "phone_" + i);
    stmt.setString(4, "email_" + i);
    // Запрос не выполняется, а укладывается в буфер,
    //  который потом выполняется сразу для всех команд
    stmt.addBatch();
}
// Выполняем все запросы разом
stmt.executeBatch();
```

В принципе вот и все — ничего сверхсложного там нет. Могу только посоветовать не увлекаться и не вставлять больше нескольких сотен запросов за раз и обратить внимание, что executeBatch возвращает массив целых чисел (попробуйте сами догадаться, что он значит — в качестве подсказки

посмотрите, что возвращает executeUpdate).
Также есть смысл посмотреть на работу getGeneratedKeys() — если вы уже забыли, что это такое — посмотрите предыдущую статью.

Много запросов в одном Statement

Вторым достаточно любопытным способом повысить производительность может быть выполнение сразу нескольких разных SQL-запросов. В качестве примера можно посмотреть следующий код.

```
1  // Вы можете выполнить сразу несколько запросов внутри одного
2  PreparedStatement stmt = con.prepareStatement(
3              "SELECT * FROM jc_contact; DELETE FROM jc_contact");
4
5  // true - первый результат возвращает ResultSet
6  // false - первый результат выполнял update/delete/insert
7  boolean test = stmt.execute();
8  // Для проверки наличия еще одного выполненного SQL-запроса
9  // можно проверить stmt.getUpdateCount()
10 while (test || stmt.getUpdateCount() > -1) {
11     if (test) {
12         try (ResultSet rs = stmt.getResultSet()) {
13             while (rs.next()) {
14                 String str = rs.getString("contact_id") + ":" + rs.getString(2);
15                 System.out.println("Contact:" + str);
16             }
17         }
18     } else {
19         System.out.println("Update SQL is executed:" + stmt.getUpdateCount());
20     }
21     System.out.println("============================");
22     test = stmt.getMoreResults();
23 }
```

Как видите, мы в одну строку поместили ДВА SQL-запроса — один SELECT и один DELETE. После этого мы выполняем метод **execute()**, который возвращает **true**, если первое возвращаемое значение является **ResultSet** или **false** — если это был запрос на модификацию.
Для перехода к результату исполнения следующего запроса вызывается метод **getMoreResults()**он также возвращает **true** в случае, если следующий результат является типом **ResultSet** или **false** — это значит, что либо SQL-запрос выполнял модификацию, либо больше результатов нет. Чтобы выбрать между этими двумя вариантами надо обратиться к методу **getUpdateCount**. Если возвращается 0 и больше — значит выполнялся запрос на модификацию. Если результат равен **-1** — значит больше запросов нет.

И наконец, полный текст примера для обоих случаев — покопайтесь в нем самостоятельно. Обратите внмание на более лаконичную конструкцию для try с ресурсами. Она появилась в java 1.7 = думаю, самое время вам понять, что и как там происходит. (в общем ничего сложного — ресурс

автоматически закрывается)

```java
package edu.javacourse.database;

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

public class BatchExampleDb {

    public static void main(String[] args) {
        try {
            Class.forName("org.postgresql.Driver");
            String url = "jdbc:postgresql://localhost:5432/contactdb";

            try (Connection con = DriverManager.getConnection(url, "postgres", "postgres")) {
                addBatch(con);
                select(con);
            }
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace(System.out);
        }
    }

    private static void addBatch(Connection con) throws SQLException {
        try (PreparedStatement stmt = con.prepareStatement(
                "INSERT INTO jc_contact (first_name, last_name, phone, email) "
                        + "VALUES (?, ?, ?, ?)", new String[] {"contact_id"})) {

            for (int i = 0; i < 10; i++) {
                // Заполняем параметры запроса
                stmt.setString(1, "FirstName_" + i);
                stmt.setString(2, "LastNAme_" + i);
                stmt.setString(3, "phone_" + i);
                stmt.setString(4, "email_" + i);
                // Запрос не выполняется, а укладывается в буфер, кторый выполняется сразу для всех команд
                stmt.addBatch();
            }
            // Выполняем все запросы разом
            stmt.executeBatch();
            // Получить список сгенерированных contact_id
            ResultSet gk = stmt.getGeneratedKeys();
            while(gk.next()) {
```

```
                System.out.println("Inserted:" + gk.getString(1));
            }
        }
    }

    private static void select(Connection con) throws SQLException {
        // Вы можете выполнить сразу несколько запросов внутри одного
        try (PreparedStatement stmt = con.prepareStatement(
                "SELECT * FROM jc_contact; DELETE FROM jc_contact")) {

            // true - первый результат возвращает ResultSet
            // false - первый результат выполнял update/delete/insert
            boolean test = stmt.execute();
            // Для проверки наличия еще одного выдолненного SQL-запроса можно проверить stmt.getUpdateCount()
            while (test || stmt.getUpdateCount() > -1) {
                if (test) {
                    try (ResultSet rs = stmt.getResultSet()) {
                        while (rs.next()) {
                            String str = rs.getString("contact_id") + ":" + rs.getString(2);
                            System.out.println("Contact:" + str);
                        }
                    }
                } else {
                    System.out.println("Update SQL is executed:" + stmt.getUpdateCount());
                }
                System.out.println("==============================");
                test = stmt.getMoreResults();
            }
        }
    }
}
```

Сам проект можно скачать здесь — BatchExampleDb.zip

Что осталось за кадром

Сейчас не могу сказать, будет ли этот раздел по базам данных в Java заключительным, но пока планирую именно так — впереди только наш замечательный пример с контактами. Какие возможности и особенности есть у JDBC еще, вы можете посмотреть сами. Мы еще много чего не разбирали:

- работа с BLOB (binary large object)
- вызов процедур через интерфейс **CallableStatement**
- модификация **ResultSet**
- использование **RowSet** с его расширениями.

- создание и использование собственных типов данных
- работа с массивами
- работа с XML

В общем, я оставляю на вашу собственную голову достаточно широкий спектр вопросов. Весьма неплохое руководство можно посмотреть на сайте Oracle здесь: Trail: JDBC(TM) Database Access.

И теперь нас ждет следующая статья: Список контактов — работаем с БД

4 comments to *JDBC — групповые операции*

- 
  Март 28, 2018 at 17:45
  Shmel says:

  При выполнении последней программы «BatchExampleDb» такая вот ошибка:

  com.microsoft.sqlserver.jdbc.SQLServerException: Перед получением результатов необходимо выполнить инструкцию.
  at com.microsoft.sqlserver.jdbc.SQLServerException.makeFromDriverError(SQLServerException.java:227)
  at com.microsoft.sqlserver.jdbc.SQLServerStatement.getGeneratedKeys(SQLServerStatement.java:2108)
  at DB.BatchExampleDb.addBatch(BatchExampleDb.java:42)
  at DB.BatchExampleDb.main(BatchExampleDb.java:17)
  at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
  at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
  at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
  at java.lang.reflect.Method.invoke(Method.java:498)
  at com.intellij.rt.execution.application.AppMain.main(AppMain.java:140)

  Я использую MS SQL Server

  Reply

  - 
    Март 28, 2018 at 23:21
    admin says:

Вполне может быть такая ошибка — там при вставке генерируется автоматически идентификатор и это делается средствами именно PostgreSQL.
Для MS SQL идентификатор может генерироваться иначе. Вот и ошибка.

Reply

- 
  Август 3, 2018 at 11:56
  Александр says:

  Благодарю Вас за статью, очень полезная и лаконичная информация о том, что волнует.

  Написано просто замечательно.

  Вот только не хватает раздела о работе с XML.

  Может, Вы могли бы добавить немножко об этом? Цены Вашим материалам не будет.

  Reply

  - 
    Август 4, 2018 at 03:32
    admin says:

    Как я понял, речь идет о работе с XML именно для баз данных ? Надо подумать. На самом деле там много чего еще есть — надо как-то соблюсти баланс между простотой изложения и глубиной погружения.

    Reply

Leave a reply

Comment

You may use these HTML tags and attributes: <a href="" title=""> <abbr title=""> <acronym title=""> <b> <blockquote cite=""> <cite> <code class="" title="" data-url=""> <del datetime=""> <i> <q cite=""> <s> <strike> <pre class="" title="" data-url="">

Имя *

E-mail *

Сайт

7 − 5 =

Add comment