JDBC 编程中批处理管理策略的应用

Application of Batch – handle Management Strategy in JDBC Programming

吴　伟　（西安石油大学计算机学院　西安　710065）
史云鹏　（西安石油大学计算机学院　西安　710065）
杨兆进　（深圳发展银行青岛分行信息科技部　青岛　266071）

摘要：数据库的连接要针对实际需求选用不同的连接策略。对现有的两种数据库连接技术在管理策略和参数配置等方面进行了分析，提出了一种基于批处理封装模式的 JDBC 数据库连接管理策略。在某些实际应用中能够简化系统结构，提高软件可重用性，降低 JDBC 与应用程序的耦合度，减少应用程序与数据库的频繁连接。

关键词：批处理　模式　JDBC　数据库　连接池

Java 语言的跨平台性、可移植性以及安全性使其成为开发数据库的一种优秀语言。其提供的与数据库管理系统连接的接口标准 JDBC 是有一组用 Java 语言编写的类和接口组成，为数据库开发人员提供了一组标准的 API，使他们能够用纯 Java API 来编写数据库应用程序。但 JDBC 是个较为底层的接口，其工作模式：(1)在主程序中建立 JDBC 数据库连接。(2)进行 SQL 操作访问数据库。(3)断开连接。这样频繁的建立与关闭连接，极大的降低了系统性能，因此在实际应用中普遍采用基入 JDBC 的数据库连接池复用技术。但无论是基本的 JDBC，还是连接池技术，随着应用规模的不同，都有其特定的适应性和局限性。本文通过对几种现有数据库连接技术的分析比较，探讨了一种在实际应用中较为可行的基于批处理模式封装的 JDBC 数据库连接管理策略，并给出了实例性能分析。

1　几种常用的数据库连接技术分析

1.1　JDBC 直接连接数据库访问方法

工作机制：

JDBC 是一个基入 Java 数据接口的规范，通过 JDBC 提供的 API，应用程序能完成与数据库的连接和通信，需经过下面几个步骤[1]：建立数据源；装入 JDBC 驱动程序；建立连接；执行 SQL 语句；检索结果；关闭连接。采用这种技术访问数据库时要通过 JDBC 驱动程序建立与数据库的物理连接[2]（如图 1 所示），其采用的驱动程序有四种：JDBC – ODBC Bridge；JDBC – Native API Bridge；JDBC – MiddleWare；Pure JDBC Driver。

图 1

适用性与局限性：

在系统中简单地利用 JDBC 提供的 API 频繁地与数据库建立和关闭连接在多用户并发访问数据库时系统开销量不可忽视，导致整体性能下降，因此在实际应用中普遍采用建立一个数据库连接池使连接多次复用。

1.2　基入连接池的数据库访问技术

工作机制：

连接池就是由应用程序服务器维持和打开连接的集[3]。当程序在 DataSource 对象上调用 getConnection 方法时服务器将池内一个连接分配给它。当程序关闭连接，服务器将该连接返回连接池但并不关闭。用这种方法各个程序或单个程序的多个线程可以共享连接。连接池的调度策略如图 2[4] 所示。

适应性与局限性：

数据库连接的建立、断开都由连接池自身来管理，用户可以通过设置连接池参数来控制连接池中的连接数、每个连接的最大使用次数等。通过使用连接池将大大提高程序效率，并可以通过其自身管理机制来监视数据库连接的数量和使用情况。但通过对其调度策略的分析和在实践应用中发现现有的连接池技术也存在一定的适用环境的局限性：首先，连接池的参数配置大多是静态的，不能根据应用需求的变化而实时地调整参数。系统运行时，处理问题的规模是在变化的，对数据库的访问的程序是数目以及线程的数目是在变化的，即需要建立的连接是在变化的，因此连接池这种先初始化一定数量的参数配置，再连接访问的机制有其一定程度的滞后性。其次，在应用程序模块与数据库模块引入连接池管理层，必定会占用一定规模的系统资源。因此，在问题规模不是很复杂应用环境中，我们可以使用基入批处理模式策略的 JDBC 数据库连接技术。实践证明，它具有简单有效的特点，并能克服 JDBC 直接连接所带来的频繁连接关闭的缺点。

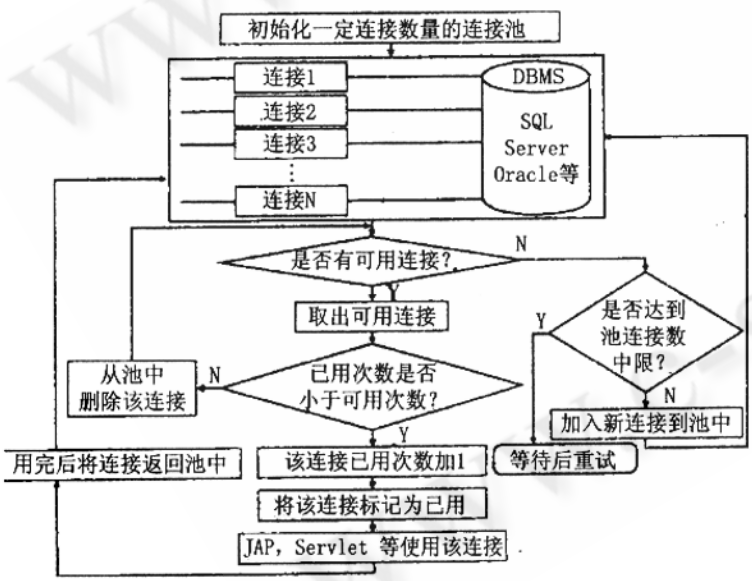

图 2　现有连接池调度策略

2　批处理管理策略在 JDBC 数据库连接技术中的应用

2.1　批处理管理策略在 JDBC 数据库连接程序中的模式分析

在与数据库连接更新数据时，批处理管理策略是一个非常有效的方法模式，这样做的目的是可以将多个单独的事务成批处理，避免反复的连接和关闭，提高效率。正如我们所知，在数据库中每次进行插入时都需要获得和释放一个锁，批处理可以减少这种开销。另外，在数据库中自动提交每个事务也是非常昂贵的，每当提交一个事务时都要在回退日志中做一些工作，如果我们在某些情况下，进行 JDBC 数据库连接时采用批处理策略，将会对系统性能产生积极的影响。这种批处理管理策略在 JDBC 连接程序中的应用模式如图 3 所示。其基本思路是：(1) 当客户需要多次访问数据库，进行多个事务处理时，用户程序只是多次调用批处理程序模块添加更新数据，避免频繁地与数据库连接、SQL 操作、关闭连接。(2) 当一组事务请求批处理完成后，客户程序调用连接程序，与数据库进行一次性联系交换数据。下面我们通过一个实例程序进行分析。

2.2　批处理管理策略在 JDBC 数据库连接程序中的实例分析

在实际的 JDBC 数据库连接应用程序中，对数据库的主要操作有：获取数据库连接；SQL 查询、插入、修改、删除；关闭连接。批处理管理策略的目的就是设置批处理程序模式，使客户程序对数据库多次访问、对多个事务处理的查询、插入修改等操作能够在一个数据库的（连接 – SQL 操作 – 关闭）周期内完成。或在尽量少的周期内完成。以此减少应用程序调用数据库的次数，提高系统性能。

以下是我们开发的某旅游公司客户信息管理系统的一个 JDBC 数据库连接程序片段。程序中，我们依据 2.1 所述的批处理策略，引入了批处理封装程序模块来管理应用程序与数据库的连接操作，来简化软件系统结构。由于降低了 JDBC 程序与数据库的连接访问次数，从而简化了连接程序的编程复杂度。同时，也降低了 JDBC 与应用程序的耦合度[6]，提高了软件的可重用性。程序运行表明，系统性能明显提高。

```
Package peng. customer009. register;
  import java. sql. *;
  ……
//获取数据库连接类 SQLServer
public class SQLServer{
/ * * create new SQLServer * /
```

```
public SQLServer( ) { ……}
public void connecte( ) {
……
try {
```

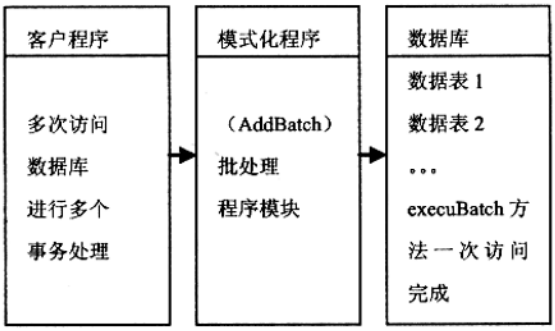

图 3

```
//load the driver class 加载 jdbc 驱动程序类
class. forName( "sun. jdbc. odbc. JdbcodbcDriver") ;
//specify the ODBC date source 定义 jdbc 的 URL 对象
string sourceURL = "jdbc:odbc:cruiseTicket" ;
//get a connection to the database 连接数据库
connection dbConn = DriverManager. getConnection (sourceURL) ;
} catch( java. lang. ClassNotFoundExceptione)
{ System. out. println ( forname:" + e. getMessage ( )) ;}
//add the customer information 批处理程序模块
public class Batcher{
string insertstatement;
statement statement1;
Batcher( ) {……}
public void batch( ) {
//建立 Statement 类对象
statement statement1 = dbConn. createStatement ( ) ;
insertSatement = "INSERT INTO CruiseCustomer VALUES(……)" ;
//使用 addBatch 方法添加一个插入操作
statement1. addBatch( insertStatement) ;
updateSatement = …
//添加一个更新操作
statement1. addBatch( updateSatement) ;
deleteSatement = …
//添加一个删除操作
statement1. addBatch( deleteSatement) ;
……}
……
}
public static void main( string[ ] args) {
string createstatement;
// * * create the tables in the database * /生成一个数据表
tableCreate( ) {……}
// * * add the customer data * /添加客户数据
AddCustomer( ) {……}
……
try{
//使用 executeBatch 执行批量更新
Batcher. statement1. executebatch( ) ;
System. out. println ( "the travelagent batch has been executed") ;
//flush and close 关闭数据库连接
dbConn. close( ) ;
}
Catch ( SQLException sqle)
{System. err. println( sqle) ;}
Catch( Exception e) {
System. err. println( e) ;}
}//main
}
```

我们可以看到，此程序也是以传统的 connecte( ) 方法获得一个连接 dbConn 并且创建一个语句。但是我们并不使用 statement 对象多次与数据库发生联系更新数据，而是将要更新的数据添加进一个批中。

Statement1. addBatch(insertStatement) ; 语句的反复使用。

当我们将所有的要更新的数据放入批中之后，调用 executeBatch( ) 方法执行实际的数据处理工作，与数据库进行一次性联系交换数据。

（下转第 123 页）

(上接第 113 页)

3 结论

在实际的 Java 数据库开发实践中,数据库的连接要针对实际需求选用不同的连接策略。本文所探讨的批处理模式在 JDBC 数据库连接应用中可以简化系统结构,提高软件的可重用性,降低 JDBC 与应用程序的耦合度。特别是,能在保证数据正确更新的前提下,减少与数据库的频繁连接。

参考文献

1 David J. Gallardo. Java Oracle Database Development,北京 清华大学出版社,2003.

2 李芝兴,Java 程序设计之网络编程,北京 清华大学出版社,2006.

3 Joe Wigglesworth. Java Programming: Advanced Topics,北京 清华大学出版社,2005.

4 马海燕,基入 JDBC 数据库连接池的自适用管理策略研究[J],计算机应用研究,2006,(2):57 -59.

5 成典勤,J2EE 架构下数据库访问的性能优化,计算机系统应用,2006,(4):64 -70.

6 Stephen. Potts Java 2 unleashed. 北京 机械工业出版社,2003.