J2EE 中提高数据库应用性能的方法

On the Methods to Improve the Capacity of Database Application in J2EE

杨 瑞（淮海工学院电子工程系 222005）

摘 要：本文给出了大型J2EE数据库应用中改善性能和提高效率的方法，目的是为了让数据库应用在J2EE分布式环境中更有延展性并且具有更高的执行效率和性能，本文从开发角度阐述了提升性能的多种办法，并最终得出结论：通过使用恰当的开发技巧，可以达到提高系统性能改善执行效率的目的。

关键词：J2EE 数据库 性能

1 引言

J2EE架构是Sun公司定义的用来开发和运行企业级Web应用的标准，其最大的特点是可用于开发大型、多层次、分布式的电子商务及企业级应用。J2EE应用主要基于Java语言，同时也极大的丰富、扩充和发展了Java语言的功能，J2EE架构与技术为组件开发模型提供广泛的支持，同时也提供相应的开发工具和服务，以便开发模块化的、可重用并且平台无关的各种组件技术的业务逻辑，这就极大的提高了开发效率。数据库应用是J2EE中的一个重要方面，JDBC(Java Database Connectivity)是J2EE访问应用数据库资源的标准，JDBC标准定义了一组Java API，允许用户写出SQL语句，然后提交给数据库，完成相应的功能，也可以很容易的把企业级的Java应用从一个数据库移植到另一个数据库上。由于J2EE主要是面向大型的企业级应用，所以性能和效率非常重要，J2EE数据库应用中除了从硬件和服务器配置方面提高性能之外，本身开发的方法和技巧非常重要，本文就从开发的角度来讨论提高J2EE数据库应用性能的若干方法。

2 开发中提高性能的方法

2.1 使用数据库连接池技术

数据库连接池技术实际上就是建立一定数目的资源对象，例如数据库连接对象，企业对象等，在一个容器对象之中。然后当客户端应用程序需要使用这些资源的时候，从中取出资源对象让客户端应用程序使用，当客户端使用完毕后再把资源放回容器中，避免重新释放资源对象需花费的时间，并且等到其他客户端应用程序需要这些资源对象时再从容器对象中取出这些资源对象供客户端使用，免除重新建立资源对象的时间。其原理图如图1。

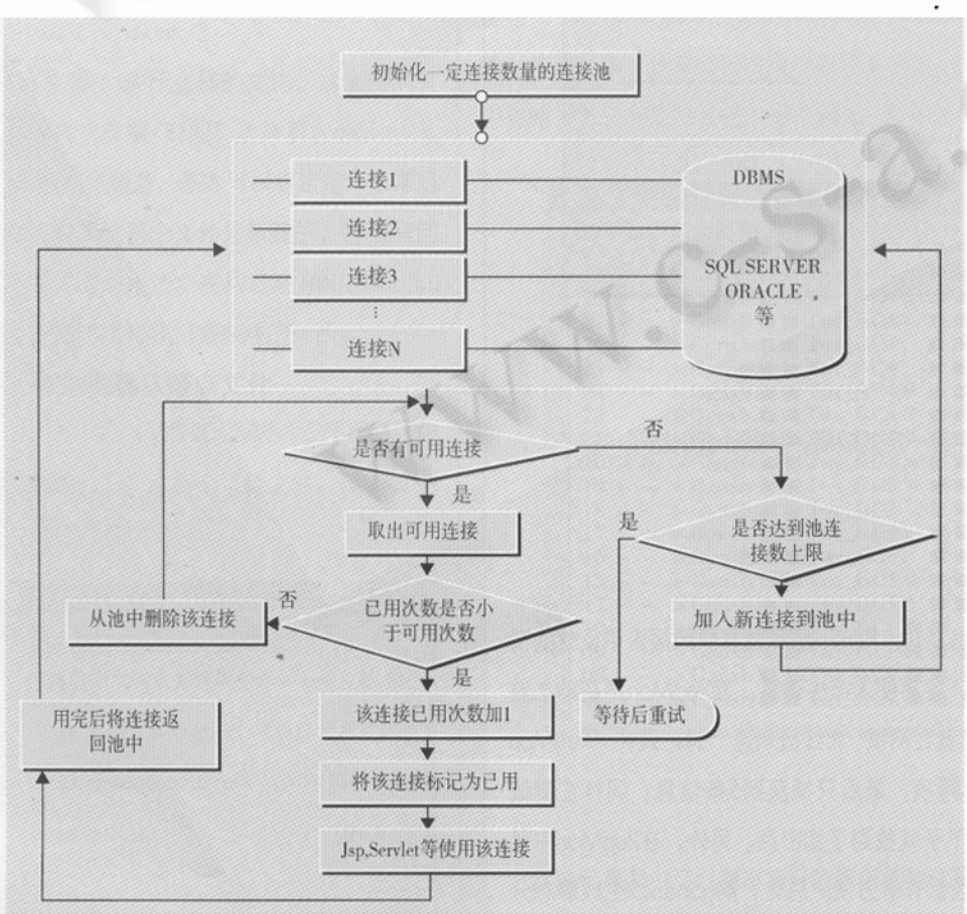

图 1 连接池工作原理图

首先应当初始化内含一定连接数量连接的连接池，然后当使用过程中池中连接数量不够时，再逐渐加入新的连接，但是兼顾到系统的资源情况，池中的连接数量也不能是无限的，

当达到最大数量时，池中就不能再加入新的连接，此时如果连接数量仍然不够时就只能等待别的连接被释放后使用，也即定义连接池中连接数量的上下界，定义上下界可根据客户端的情况动态的使用资源，提高系统的效率。另外，同一个连接如果被使用的次数太多，可能会导致该 连接的不稳定性，所以设定一个连接可以使用的最大次数，当使用次数达到最大次数后就将该连接关闭并从池中删除，在连接不够用的情况下可以生成一个新的连接加入到池中。使用连接池对于访问数据库的大多数应用是非常有利的，但不是对所有的应用都有利，并不需要在每种情况下都使用连接池技术，如果应用具备下面的特征，则可以考虑使用连接池：

（1）用户都通过通用的数据库帐户访问数据库，而不是每个用户使用各自的帐户；

（2）数据库连接只用于单个请求的持续工作期间，而不是来自相同客户的多个请求的联合持续工作期间。恰当的使用连接池将大大的提高大型企业级应用的效率，使用连接池的时候应有三个注意点：

① 不要长时间闲置某一个JDBC连接；

② 数据库连接池数量要恰当；

③ 某一个连接不能无限次的使用。

2.2 使用 JTA（Java Transaction API）的技巧

JTA即Java事务API，事务是数据库中保证一系列数据库操作能准确完成的重要手段，在JDBC的数据库操作中，一项事务是由一条或是多条表达式所组成的一个不可分割的工作单元，通过提交commit()或是回滚rollback()来结束事务的操作。事务在提交或回滚之前，Connection对象暂时处于中间过渡状态，数据库连接不会返回到连接池，即使用户此时关闭了数据库，数据库连接也不会返回连接池，这将极大的影响服务器的性能，所以，对于多用户网站尤其是大型应用来说，不该让事务长时间的停留在悬空打开状态直至超时。这样甚至有可能耗尽所有的连接资源，造成服务器的崩溃。应当慎重使用事务，避免长时间的事务，一个事务如果从加载浏览器开始，到用户点击提交按钮之后才提交，这将消耗大量资源，对于长时间的事务，应尽可能的将其拆成若干个时间短的事务。如果可能，在数据库中应该尽量使用显式提交事务的方法，这种方法可能会轻微加大数据库和网络通信的负荷，但对系统的总体性能有所提高。

2.3 尽量使用 Update 更新和批量更新

使用Update更新一个数据行比先删除数据再插入的方式更有效率，特别是对于成批数据的更新，所以在应用中更新数据应尽量使用Update而不使用Insert和Delete。另外对于多个更新的情况，应尽可能采用批量更新的方法，批量更新可以把一组数据库更新操作合并成一个批量的操作，所有的更新可以通过一个数据库调用完成，这样就不用对每一次更新都建立一次数据库连接，只需要一次数据库调用就可完成，这是个流水线式的过程，一次向数据库提交多个更新。因为在更新过程中DBMS会锁定有关数据行或表中的部分信息，用完后再释放，锁定就限制了其他客户访问相应的这些数据，造成系统性能的下降，而批量更新在一次连接中只发送一次更新命令就可以完成所有的更新动作，所以使用批量更新也可提高系统的效率。如更新语句：(注：stmt为Statement类实例)

```
stmt.executeUpdate("insert into tablename values('1',.....)");
stmt.executeUpdate("insert into tablename values('2',.....)");
stmt.executeUpdate("insert into tablename values('3',.....)");
```

可合并为一个批量操作：

```
stmt.addBatch("insert into tablename values('1',.....)");
stmt. addBatch("insert into tablename values('2',.....)");
stmt. addBatch("insert into tablename values('3',.....)");
try {
  stmt.executeBatch();
  }
 catch (Execption e)
   {
      System.out.println("更新出错！！，错误为："+e);
   }
```

2.4 注意关闭 JDBC 连接时的顺序

在用完数据库连接后应及时的关闭数据库的连接对象释放JDBC资源，关闭和释放资源的顺序是非常重要的，正确的关闭顺序应该与打开顺序是恰好相反的，即首先应该关闭ResultSet类对象，然后关闭Statement类对象，最后关闭Connection类对象，而且Connection对象应在finally{...}代码中关闭。如果在关闭一个ResultSet时收到一个异常错误，那么finally{...}中的余下代码包括其他close()语句都不会被执行，那么将会意外的闲置一个数据库连接，从而影响系统的性能。所以如果在finally{...}中使用多个close()方法时，应该放在try/catch代码块中调用，如果出现异常的话，用catch{...}中代码进行处理，使余下的代码能顺利执行。例如可用如下代码关闭JDBC连接：(注：其中rs为ResultSet类实例，stmt为Statement类实例，dbconn为Connection类实例)

```
try {
  ...
   }
catch  (Exception E) {
  ...
  }
finally
  {
  if (rs!=null) {
     try { rs.close(); } catch(Exception ignore) {};
   }
  if (stmt!=null) {
```

```
            try { stmt.close(); } catch(Exception
ignore) {};
        }
        if (dbconn!=null) {
            try { dbconn.close(); } catch
(Exception ignore) {};
        }
    }
```

2.5 初始代码放在 init（）方法中

在Applet或Servlet应用中，应该把初始代码放置在init()方法中，而不是start() 或 service()中，init()代码只是在系统初始的时候执行一次，start() 或 service()代码则会在每次执行Applet或Servlet时都执行一次，所以为了避免系统多次执行只需执行一次的代码，应该将初始代码放在init()方法中。

2.6 用好 SQL 语句

对于许多有性能问题的应用系统来说，常常是由于SQL语句的使用不当引起的，例如一次返回太多的数据行、查询太笼统等，正确的使用SQL可以优化服务器的性能，如SQL的连接和子查询。另外，充分利用DBMS提供的功能也是提升性能很重要的一方面，如使用宏语句和存储过程等。

3 结束语

以上提到的是为了提高系统的应用性能而在开发中使用的方法和技巧，在实际应用中还有很多其他方法可以改善系统的性能，如增强服务器的硬件性能、修改服务器配置等，但这些都并非是开发技巧，本文并不讨论。总之，在实际应用中恰当使用上述开发技巧，可以达到提高系统性能改善执行效率的目的。

参考文献

1 Duane K.Fields,Web development with Java Server page, GreenWish CD, Manning Publicaions co.,2000.4。

2 Michael Girdley，Rob Woollen 等，J2EE Applications and BEA WebLogic Server,电子工业出版社,2002.4。

3 林邦杰，Jsp交互网站务实经典，，中国青年出版社，2001.3。

4 Brett Spell,Java 专业编程指南,电子工业出版社，1999.6。