J2EE 架构下数据库访问的性能优化

Performance Optimization of Database Access Based on J2EE

成典勤 （安康学院计算机科学系 725000）
崔杜武 （西安理工大学计算机科学与工程学院 710048）

摘要：J2EE 架构下 B/S 结构的应用系统大多需要数据库的支持，数据库访问的性能极大地影响着应用系统的性能，本文从数据库设计的优化、数据库连接方法的选择和代码优化三个方面提出了应用程序连接数据库的性能优化方法。

关键词：J2EE 数据库 连接方法 性能优化

1 引言

随着网络技术的发展，应用程序由 C/S 模式向 B/S 模式的转变是大势所趋。在 J2EE 架构下，无论是建设企业内部信息系统还是开发基于 Web 的应用软件，都离不开数据库的支持，而经常困扰开发人员和设计人员的问题之一是应用程序的响应速度太慢。响应速度的快慢在很大程序上取决于应用程序对数据库的访问速度。为此可以从数据库的优化设计、数据库连接方法的选择、应用程序编码的优化三个方面来提高数据库的访问速度，从而达到提高整个应用系统的响应速度。以下将从这三个方面来论述如何提高和优化数据库访问的性能。

2 数据库设计的优化

作为一个应用系统的后端数据库，要想有较好的数据库访问性能，首先是从数据库设计的角度建立一个有良好库结构的数据库[2]。良好的数据库设计以减少数据冗余、保持数据的完整性和一致性为基本原则。过多的冗余数据量不仅浪费存储空间而且使数据库变得复杂低效，数据库维护也较困难，但从另一方面来说，适当的冗余数据可减少数据库连接的操作，从而提高数据库访问效率。

（1）数据模型的优化。关系数据模型的优化以规范化理论为指导，在确定数据依赖的基础上对其进行极小化处理，消除冗余的联系，考察是否有部分函数依赖、传递函数依赖、多值依赖等，对某些模式进行合并或分解，最终使数据模型尽可能优化。

（2）数据规范化 。在数据库的设计中，把数据元素组织为相关的组，尽可能减少冗余并确保数据完整性。

（3）引用完整性。关系数据库在创建表时向其中添加主键和外键，使这些数据组建立起关系，引用完整性就是在此基础上向数据库添加约束而形成的。而 DBMS 能阻止主键和外键被删除或修改，也阻止在不能保持完整性的情况下向表中插入新行。

（4）适当的索引。在一个表上建立索引，能提高信息检索的效率，但索引也不是越多越好，因为 DBMS 要在数据表的插入与删除时对与此表关联的所有索引进行修改，有时因索引过多而产生较长延迟，导致性能下降。故而索引一定要适宜而建。

（5）数据库设置。对应用程序的后端数据库要根据数据库并发访问量的预估峰值和均值，采用适当的数据库缓冲策略，并根据数据库服务器的具体情况调整数据库的参数，比如分配给 SGA（System Global Area）和 PGA（Program Global Area）的内存大小比例等。

3 数据库连接方法的选择

在 J2EE 架构下，数据库的连接有三种方法[1]，分别阐述如下：

3.1 基于 JDBC 直接连接的数据库访问方法

JDBC（Java Database Connection）是 SUN 公司制定的一个基于 Java 数据接口的规范，也是 J2EE 架构下

©《计算机系统应用》编辑部 http://www.c-s-a.org.cn

应用程序开发的一组 API。它主要提供三项功能：连接数据库、向数据库发送 SQL 语句；处理返回的结果。采用这种技术是在需要访问数据库时通过 JDBC 驱动程序建立与数据库的物理连接[4]，访问结束又断开连接。即用即连，用完即断。JDBC 驱动分为四种类型：

（1）JDBC－ODBC 桥。用于在 JDBC 规范和 ODBC 规范之间转换 DBMS 调用，符合 JDBC 规范的 J2EE 组件接收到消息后由该桥驱动转换为 ODBC 消息格式，再转换为 DBMS 可以理解的消息格式。

（2）JDBC－Native 桥。是由具体的 DBMS 制造商提供桥驱动和 API 类，J2EE 组件产生被本地特定 DBMS 所理解的代码。这使代码失去了一定的可移植性。

（3）JDBC－Network 桥。这种驱动程序将 JDBC 转换为与 DBMS 无关的网络协议，之后又被某个服务器转换为一种 DBMS 协议。该类型的驱动与平台无关，客户端无需安装，很适合于 Internet 应用。

（4）Pure Java JDBC Driver。该驱动将 JDBC 调用直接转换为 DBMS 所使用的网络协议，允许从客户机直接访问 DBMS 服务器，是本地协议完全 Java 化的驱动程序。

从目前来看，前两种属过渡方案，现已逐渐淘汰，而后两种是 JDBC 访问数据库的主流驱动。第四种驱动程序由于避开了本地代码的问题，减少了开发的复杂性，也减少了冲突和出错的机会。随着 JDK1.4 的出现，Type4 驱动应该是最好的选择，因为它不要移植任何本地代码，由纯 Java 实现可以直接访问数据库，容易进行控制与部署，也不需要安装另外的中间件。

部分代码实例如下：

```
import Java.sql.*;
…….
Try{ Class.forName("Oracle.JDBC.Driver.OracleDriver");
Connection Con = null;
Con = DriverManager.getConnection("JDBC:Oracle:Thin:@[host]:[port]:[SID",[user],[password]];
……..//数据库操作代码部分
}
catch(exception e){ //处理 exception 的代码部分}
…….
}
```

3.2 基于连接池的数据库访问方法

在数据库处理中，资源开销最大的是建立数据库连接。比如要建立一个数据库连接，数据库服务器必须分配通信和内存资源以验证用户，并且为每个连接建立安全环境。若每一个用户访问时都重新建立连接，不仅用户要长时间等待，而且系统有可能会由于资源消耗过大而停止响应。为此 JDBC2.0 中的连接池技术能在客户之间共享一组连接，提高应用程序的响应能力[3]。

连接池（Connection Pool）就是众多连接对象的“缓冲存储池”，也就是连接对象的集合体，连接池是一个类，用它生成一个对象时，即在系统初始化时连接池从数据库一次性的获取预设数目的连接对象并由 J2EE 服务器打开和维护，使用连接时从连接池中获取，用完后逻辑断开，这样就可以避免连接随意建立、关闭造成的开销。连接池可以使用静态和动态两种策略来管理池中的连接。

通过连接池完成数据库访问的部分代码如下：

```
ConnectionPool con = ConnectionManager.getConnectionPool("testPool")
……
Statement stmt = dbCon.createStatement();
ResultSet rs = stmt.executeQuery("select field1, field2 from tablename");
…… //关闭 rs stmt
```

3.3 基于实体 Bean 的数据库访问方法

实体 Bean 是 EJB（Enterprise Java Bean）组件技术之一，代表处理底层持久性数据的组件模型，是能够存放在永久性存储空间中的持久对象。实体 Bean 的持久性可以被 Bean 管理，即 BMP；也可以让 EJB 容器管理，即 CMP。数据库的操作根据实体 Bean 是 BMP 还是 CMP 有所不同：BMP 要求开发人员在实体 Bean 中提供数据存取代码完成对数据库的访问；CMP 持久性则意味着 EJB 容器自动处理数据存取的调用，没有数据库访问代码写入 Bean 类。实体 Bean 和数据库的连接不是通过数据库的绝对名来连接数据库，而是用逻辑名连接到数据库，即用 JNDI 来查找数据源，并进一步实现连接。

©《计算机系统应用》编辑部 http://www.c-s-a.org.cn

通过实体 Bean 访问数据库要先设计映射数据库表的实体 Bean，对应于 testtable 表设计的实体 Bean 为 TestTableBean。在建立了连接的基础上，进行数据库操作。

(1) BMP 中实体 Bean 对数据库的访问。在 BMP 中的实体 Bean 里，数据库操作的代码写入了实体 Bean 的相应方法中，调用该方法就可以实现对数据库的操作。在 TestTableBean. java 的 ejbLoad( ) 方法中取得数据库的连接，并实现对数据库的操作，代码如下：

```
public void ejbLoad( ) {
…….
PrepareStatement pstmt = null;
Connection con = null;
try{
con = getConnection( ) ;
pstmt = con. prepareStatement ( " select field1, field2 from tablename" ) ;
pstmt. setString (1,id) ;
ResultSet rs = pstmt. executeQuery( ) ;
……..}
…….} //处理异常并关闭 pstmt,con
```

(2) CMP 中实体 Bean 对数据库的访问。CMP 能够分离一个实体 Bean 和它的持久化表示，即分离数据逻辑和 JDBC 连接，由于不再控制常规的 Bean 的持久化操作，所以，CMP 实体 Bean 的许多方法可以是空的，由容器做这些事情。例如 JDBC 代码就是通过继承实体 Bean 类来生成的。由于 CMP 对数据库的操作完全由容器管理，所以在 Bean 中不需要写入操作数据库的代码，而是在生成实体 Bean 时增加 finder 方法，方法的内容是用 EJB 查询语言 EJB－QL 表示的。

3.4 三种数据库访问方法的比较

方法一每次要建立连接与关闭，在多用户并发访问数据库时的系统开销是不可忽视；方法二中数据库连接的建立、断开都由连接池自身来管理，还可以通过设置连接池的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，并可以通过其自身的管理机制来监视数据库连接的数量和使用情况；方法三是由 EJB 容器维护数据库的连接池，连接池对实体 Bean 透明，当实体 Bean 申请一个连接时，EJB 容器从连接池中提取一个连接并分配给组件，实体 Bean 很快就可以完成连接的获得、使用和释放，从而在 EJB 容器中一个连接可以被多个组件使用。

从系统开发复杂度来考虑，前两种方法要编写 JDBC 代码，而方法三中 JDBC 代码的编写简单甚至没有(CMP 中)，开发效率高，在 J2EE 应用系统中，方法三是首选。

从应用环境考虑，实体 Bean 可以连接到不同的数据库，可以在具有不同数据库名的不同环境中部署，并可重复使用，也能集成到分布式环境的应用系统中去。因此，方法三更适合分布式环境中多个用户同时更新数据库的情况；而前两种方法更适合于 Web 应用。

4 编程中的优化

在编程中运用以下方法优化程序代码：

(1) Servlet 中的 init( ) 方法。仅需执行一次的代码应放入 init( ) 方法中而不是 service( ) 方法中。

(2) 多线程的 DataSource 技术。DataSource 对象是 Connection 对象的工厂，用一个连接池来提供和存储 DataSource 的连接，通过调用 DataSource 类实例的有关方法可访问数据库。而 DriverManager 是一个同步执行类，其效率不及 DataSource，应当尽量避免使用。部分代码如下：

```
try{ Context ctxt = new InitialContext( ) ;
javax. sql. DataSource ds = ( javax. sql. DataSource) ctxt. lookup( "my_datasource_pool" ) ;
javax. sql. Connection con = ds. getConnection( ) ;
}catch( SQLException e) {
……. //处理 exception 的代码部分
}
```

(3) PreparedStatement 允许一个语句使用不同的参数重复执行，对支持预编译的数据库能提升数据缓冲命中率，对大量的数据库访问可以提高执行效率。

(4) 尽快释放连接池中的连接。由于连接池中的连接被多个用户所共享，在并发请求数大于连接池的容量时，尽快释放暂不使用的数据库连接就更为重要。

(5) 在编程中应尽量少使用事务，或把一个时间长的事务拆分成若干个时间短的事务。

(下转第 70 页)

©《计算机系统应用》编辑部 http://www.c-s-a.org.cn

(上接第 66 页)

(6) 应充分利用数据库的触发器和存储过程[5]。

5 结束语

在 J2EE 架构下,有数据库支持的应用系统要想提高系统的可用性和整体性能,首要的是从数据库的物理设计、逻辑设计和关系规范化等方面进行优化以得到性能良好的数据库系统;最重要的是依据应用系统的运行环境和数据库的并发访问量来选择连接数据库的方法,使系统具有较快的响应速度;最不可忽视的是在编码中运用优化的编程方法,使得代码尽可能最优。当然,在这样的系统中,开发人员也应和有资深经验的数据库管理员密切合作才能使整个系统性能达到最优。

参考文献

1 [美]Jim Keogh 著,宁建平等译,J2EE 参考大全[M],电子工业出版社,2003。

2 尹萍, SQLServer 数据库性能优化[J],计算机应用与软件,2005,51-53。

3 孙叶枫、宋中山,JSP 中基于连接池的数据库访问技术[J],计算机应用,2004,(6),80-82。

4 宗平、阙凌燕、张艳丽等,基于 J2EE 的 Oracle 数据库连接研究与实现[J],计算机应用研究,2002,(8),73-74。

5 蒋智宁、林凌峰、袁平波,利用 Oracle8i 开发 Java 应用的研究[J],计算机应用研究,2000,(4),43-45。

©《计算机系统应用》编辑部 http://www.c-s-a.org.cn