基于 JAVA 的多种数据库分页方法研究

Research of Web Database Paging Based on Java

陈远平　及俊川　（中国科学院计算机网络信息中心 100864）

摘要：Web 应用中分页显示是一种常见的浏览和显示大量数据的方法，同时也是 Web 编程中开发人员最常处理的事件之一。主要讨论并分析了在服务器端上利用 Java 技术对几种主要的关系数据库数据进行分页显示的方法，并且给出了相应的实例。

关键词：Java Web 关系数据库 分页

在网上经常进行信息的发布与查询。但是，如果要发布或查询的信息量比较大，简单的将数据一次性的全部发往客户端，不仅不方便终端用户的浏览，而且影响了系统的性能：首先，将大量的数据密密麻麻的显示在终端用户的浏览器窗口上使得用户只能通过窗口上的滚动条来帮助查看数据，很不方便，同时容易错过自己想要的信息；其次，一次性将大量的数据发往客户端，增加了网络带宽的负担，延缓了系统的反应速度，降低了系统的性能。因此，对网络应用中数据的分页显示是十分有必要的。本文讨论了利用 Java 技术进行 Web 应用开发中针对不同的数据库进行分页显示的方法。

1　相关 Java 技术介绍

1.1　Java Servlet

Servlets 是 Java2. 0 中新增的一个功能。Servlets 是一种采用 Java 技术实现 CGI 功能的技术。Servlet 和 CGI 一样都是运行在服务器端上，用来生成 Web 页面。得益于 Java 的跨平台的特性，Servlet 也是与平台无关的，实际上，只要符合 Java Servlet 的规范，Servlet 是完全和平台无关并且和 Web 服务器无关的。Java Servlet 内部是以线程方式提供服务，不必对于每个请求都启动一个进程，利用多线程机制可以同时为多个请求服务，因此 Java Servlet 效率非常高。

1.2　Java Server Page

为了解决 Java Servlet 没有把应用的逻辑和页面的输出分开而导致整个 Servlet 代码混乱不堪的缺点，SUN 推出了 Java Server Page – JSP。JSP 技术与 ASP 技术类似，是一种在服务器端进行解析，动态的生成网页输出到客户端浏览器上的 Web 技术；它提供了一种是内容和显示逻辑分开的简单方式。同 ASP 和 PHP3 等语言不同的是，JSP 在执行的时候是编译式的而不是解释式的。本质上说，JSP 是一种高层的 Servlet。执行的时候，JSP 被翻译为 Servlet 的 Java 源文件，再经过 Java 编译器编译为 Servlet 的 Class 文件，最后由支持 Java 虚拟机的服务器来执行，输出结果。JSP 的强大之处主要体现在能和 JavaBeans、Enterprise JavaBeans 一起工作，从而能够处理各种不同的业务逻辑，满足用户的需求。

1.3　JavaBeans

JavaBeans 是 Java 的可重用组件技术。JSP 通过 JavaBeans 来实现复杂的功能，如文件上传、发送 Email、负责与数据库交互并提取以及将业务处理或将复杂计算分离出来成为独立可重复利用的模块。JavaBeans 的使用给 JSP 应用带来了更多的灵活性与可伸缩性。

1.4　JDBC

JDBC 是用于执行 SQL 语句的 Java 应用程序接口，由一组 Java 语言编写的类与接口组成，它实现了一个独立与特定数据库管理系统 DBMS 的通用的 SQL 数据访问和存储结构，从而使得独立于 DBMS 的 Java 应用程序的开发工具和产品成为可能。JDBC 定义了四种不同的驱动程序，这里不再详细赘述了。大多数数据库供应商既为它们的数据库提供了类型 3 驱动程序，

也提供了类型 4 驱动程序。这两种类型的驱动程序都是纯粹的 Java 库,与具体的平台无关。

2 具体实现涉及的问题

2.1 JDBC 驱动的选择

不同的数据库厂商都提供了针对自己数据库产品的 JDBC 驱动程序,同时许多的第三方公司也针对目前流行的数据库生产了一些驱动程序,在具体实现的时候可以做出适当的选择以获取更好的性能,更高的可靠性。

2.2 数据库连接的获取和操作

利用 JDBC 对数据库进行操作时,要首先获得一个对数据库的连接。获取数据库连接的方式主要有以下两种:

(1)直接获取连接。即每个单独的 Servlet 或者 JavaBean 实例都直接获得数据库连接。实现的时候首先加载相应的数据库驱动程序类,然后通过 JDBC 提供的驱动程序管理器(DriverManager)对这个类进行注册并生成一个实例,通过给此实例传递一个 URL 参数以指定所要连接的数据库地址,最后通过 DriverManager 的 getConnection( ) 方法得到数据库的一个连接。

(2)使用 Web 容器的数据库连接池连接。通过使用 Web 容器的数据源——数据库连接池来获得数据库连接。理论上,这种方式能提供最佳的性能。Servlet 或者 JavaBeans 直接从 Web 容器配置中取得数据源及其连接对象,然后通过此连接对象来操作数据库。对于数据库连接对象的管理则交由 Web 容器来负责。利用此方法需要对 Web 容器做相应的配置,具体可查阅相关资料。

(3)数据库连接的操作方式。在获得数据库的连接后,其操作方式也有不同,具体方式有:

① 在 Servlet 实例的每个处理方法中每次都调用数据库连接,然后用此连接进行数据库的查询等操作,最后关闭并释放此连接。

② 在 Servlet 实例的初始化操作时建立一个数据库连接,直到 Servlet 实例在销毁时关闭并释放此数据库连接。

2.3 数据库请求结果的处理方式

(1) 直接使用 ResultSet 。直接使用 JDBC 中提供的 ResultSet 对象来处理数据。ResultSet 可以看作是一张表,它包含了符合 SQL 查询语句条件的所有行以及查询的列标题和值。ResultSet 直接在数据库上建立游标,并维护指向其当前数据行的光标。ResultSet 预先定义了一系列的方法,这些方法包括了对数据集的读取、添加、删除、修改等,大大的方便了开发人员对数据库的操作。

实现时,当用户第一次请求数据查询时,就执行 SQL 语句查询,获得的 ResultSet 对象及其要使用的连接对象都保存到其对应的会话对象中。以后的分页查询操作都通过第一次执行 SQL 获得的 ResultSet 对象定位要显示记录的范围。最后在用户不再进行分页查询时或会话关闭时,释放数据库连接和 ResultSet 对象等数据库访问资源。

(2) 定位结果集。提取用户在分页面查询请求中传递的参数,如页码,每页显示的记录数等,利用这些参数可以确定要显示的分页面查询的数据集的行范围,并且动态的生成指定范围的 SQL 查询语句,然后每次请求获得一个数据库连接对象并执行 SQL 查询,把查询的结果利用 Vector 对象或者以 Bean 封装后返回给用户,最后释放所有的数据库访问资源。

这种方法对于特定的数据库的处理方式有所不同。某些数据库提供了对查询的结果集进行行范围定位的 SQL 接口技术,我们可以利用此技术方便的实现结果集的定位。这样的数据库有 Oracle、DB2、PostgreSQL、MySQL 等。另外象 MS SQL Server 这样的数据库没有提供此类的 SQL 接口技术,在实现的时候我们需要通过辅助的方法实现,比如增加一个可以唯一标识记录的字段等,下文会有详细的说明。

(3) 使用变量缓存全部结果。使用中间变量来缓存查询结果的全部数据并将其保存到会话中,用户以后的分页查询操作直接从此缓存中取得。

3 解决方法的比较选择

前面已经讨论过,相对于让 Servlet 或 Bean 实例直接获得数据库连接的方式而言,通过 Web 容器的数据库连接池来获得并管理数据库连接能够得到更好的性能。结合实际应用情况对相应参数进行设置后,数据库连接池能对数据库连接对象的最大活动连接(Active Connection)个数、最大的空闲(Idle Connection)连

接个数、最大的等待连接数以及连接等待时间等进行有效的管理,合理的分配和回收资源,从而使系统性能能得到很好的管理、改善和提高。

因此以下主要针对通过 Web 容器获得数据库连接后对数据库连接的操作方式以及对查询结果集的操作方式作进一步的讨论。

(1) 如果采用使用 ResultSet 对象来直接操作请求结果,那么我们就需要在 Bean 或者 Servlet 实例初始化的时候建立一个数据库连接,并且维护这个连接直到实例销毁或者会话结束。这种情况下,其优点是:减少了数据库连接对象的多次分配获取,减少了对数据库的 SQL 查询执行等操作。但是,其缺点也是明显的。由于每个客户对应一个会话,每个会话对应一个数据库连接和一个结果集(游标),数据库连接和游标是在会话终止时才释放的,因此,一个用户的分页查询就要占用一个数据库连接对象和结果集的游标,在多个客户的请求过程中,对数据库的访问资源如数据库连接和用于数据操作的游标等资源的占用会比较大,利用率不高。同时由于应用中代码的复杂,如包含有较多的 IF,ELSE 等嵌套语句,可能导致开发人员程序设计上出现疏忽:会话对应的数据库连接和游标在会话终止时还没有释放。因此,在这种方法下数据库访问及处理的资源占用可能是系统性能的一个瓶颈。

(2) 如果采用的是定位结果集或者使用变量缓存查询结果的方法,那么可以采用在每个业务处理的时候建立连接,业务处理结束的时候关闭并释放连接,然后将结果返回给客户端的方法。这种情况下,其缺点是用户每次分页面查询请求都要从 Web 容器中获得数据库访问资源(数据库连接对象和数据库游标),要根据相应的结果集范围的参数多次执行 SQL 语句的数据库查询操作,这对数据库有一定的影响。系统较频繁的做创建和释放数据库连接的操作,也会对系统的性能有所影响。但是如果能够对 Web 容器的数据库连接池的配置根据具体情况进行合理的调整(如数据库连接对象个数和 Web 容器配额的线程个数的关系等),能将这方面的负面影响有效的降低,提高性能。采用这种方法的优点是对数据库的访问资源使用完了就立即释放,不白占用数据库访问资源,从而提高了资源的利用率,同时这种方法对于提供了对查询的结果集进行行范围定位的 SQL 接口技术的数据库而言,充分利用了数据库自己的 SQL 查询功能对分页查询性能做了优化。另外这种方法对网络带宽的占用也相应的较小,传输速度较快。

(3) 对于缓存一次 SQL 查询的结果集的情况,其优点是减少了对数据库访问的次数,对数据库连接以及游标等的访问资源占用也很少。但是其缺点也是显而易见的:首先在这种方式下,对 Web 容器内存的要求很高,当有大量用户的查询操作,并且查询的结果集较大时,系统内存需求将会是整个系统性能的一个很大的瓶颈;其次,既然查询使用的是缓存结果,那么用户就可能会看到过期的数据。

综上所述,采用定位结果集的方法进行分页显示是较好的选择。

4 具体实现

通过前面的讨论,我们采用定位结果集的方法进行分页显示。针对不同的数据库,具体实现也有所不同。以下将数据库类型分为两类做进一步的讨论。

4.1 支持结果集行定位 SQL 技术的数据库

这类数据库中主要的有 Oracle、MySQL、DB2 以及 PostgreSQL 等。以下给出一个 Oracle 数据库分页实例代码的主要部分:

```
/**
 * @param pageSize—每页显示的记录条数;
 * @param pageNo 当前显示页的页码
 * @param _condition 查询条件
 */
public List getPersonList(int pageSize,int pageNo,
String _condition){
    DBConnection DBConn = new DBConnection
    ();//数据库连接管理类;
    Connection conn = null;String sql = null;
    Vector result = new Vector();//存放结果列
    表
    //根据 Oracle 提供的 rownum 功能进行分页
    int startRowNum = pageSize*(pageNo-1)
    +1;//本页起始行
    int endRowNum = pageSize*pageNo;//本页
    末行
    String sqlStart = " select id,name,tele,address
```

```
        from ( select rownum num , person. * from per-
        son
      where 1 = 1 " ;
        String sqlEnd = " order by id asc ) t " ;
        String sqlPaging = " where t. num > = " + Inte-
ger. toString ( startRowNum ) + " and t. num < = " + In-
teger. toString ( endRowNum ) ; //定位所显示页的记录
        if ( _condition. equals ( " " ) | | _condition = =
null )
          sql = sqlStart + sqlEnd + sqlPaging ; //组合
查询语句
        else
          sql = sqlStart + _condition + sqlEnd + sql-
Paging ; //组合查询语句
        try {
          conn = DBConn. getConnection ( ) ; //获取数
据库连接
          DBAccess dba = new DBAccess ( conn ) ;
          ResultSet rs = dba. openSelect ( sql ) ; //执行
sql 语句
          while ( rs. next ( ) ) { //将每行记录映射成 Per-
sonBean , 存放入 Vector 中
            PersonBean person = new PersonBean
( ) ;
            person. setId ( rs. getInt ( " id" ) ) ;
              person. setName ( rs. getString ( "
name" ) ) ;
            person. setTele ( rs. getString ( " tele" ) ) ;
              person. setAddress ( rs. getString ( " ad-
dress" ) ) ;
            result. add ( person ) ;
          }
          rs. close ( ) ; //关闭 ResultSet
          dba. closeSelect ( ) ; //关闭 Statemnet 等资源
          DBConn. freeConnection ( conn ) ; //释放数据
库连接
          return result ; //返回结果列表
        } catch ( Exception ex ) {
          System. out. println ( ex. getMessage ( ) ) ;
          DBConn. freeConnection ( conn ) ;
          return null ;
        }
      }
```

对于其他数据库可把上面代码中的黑体字部分表示的 SQL 语句替换成下面的相应的语句即可。

MySQL 数据库采用 Limit 关键字来实现结果集的定位：select select_list from TableName limit pageSize * (pageNo - 1) , pageSize * pageNo order by ColumnName。

DB2 数据库采用 rownumber ( ) 函数来实现：select * from (select select_list rownumber ( ) over (order by ColumnName) as row_scope from TableName) as table_temp where row_scope between (pageSize * (pageNo - 1) + 1) and (pageSize * pageNo) 。

PostgreSQL 数据库采用 LIMIT 和 OFFSET 两个关键字组合的方式来实现：select select_list from TableName order by ColumnName LIMIT pageSize OFFSET pageSize * (pageNo - 1) 。

具体的 JSP 调用代码这里就不再赘述。

4.2 不支持结果集行定位的数据库

这类数据库中的代表是 MS SQL Server 数据库。下面就以 MS SQL Server 为例来说明如何实现这类数据库上结果集的行定位。

(1) 通过关键字段或唯一 (unique) **值字段实现。对于拥有此类可唯一标识记录行字段的表可以通过嵌套查询的方法对结果集进行行范围的定位。例如，若 A 表有唯一字段 ID，那么可以用如下的 SQL 查询语句定位某一页数据** (**其中 pagesize 为单页显示的条数，** curpage **为当前要显示的页码**) :

select top " + pageSize + " * from A where ID not in (select top " + ((curpage - 1) * pageSize) + " ID from A order by ID) order by ID "

(2) 用存储过程动态产生具有关键字表的临时表。少数情况下，对于不拥有关键字或者唯一值字段的表，我们可以通过存储过程在原有表的基础上插入一列作为标识，生成一个临时表，在临时表的基础上定位结果集的范围，然后在前台页面上调用存储过程得到相应的分页数据。

(**下转第 65 页**)

（上接第 69 页）

5 结论

数据库分页显示是一项在 Web 开发中经常遇到的看似简单但却是非常重要的工作，合理的选择分页显示的方法，对于提高系统性能有着很大的帮助。同时通过以上的讨论，我们也能够编写出通用的数据库分页类方便以后的应用开发。

参考文献

1 黄理、洪亮、曹林有、张勇等，Jsp 高级编程[M]，希望电子出版社，2001。

2 Cay S. Horstmann, Gary Cornell. Java 核心技术卷[M]，机械工业出版社，2002。

3 微软公司，MicrosoftSQLServer2000 使用，Transact－SQL 进行数据库查询[M]，希望电子出版社，2001。

4 George Reese. Database Programming with JDBC and JAVA，Second[M]，中国电力出版社，2002。