基于Oracle数据库的分页显示中SQL复杂查询结果集总行与数据正确性控制

韩 强 周恕义

北京工业大学 北京 100124

摘 要: 本文首先介绍了分页显示技术及Oracle数据库的Rownum特性在分页显示中的应用，然后根据项目实例分析了含有聚合函数及分组关键字的复杂查询语句在分页显示查询中遇到的查询条目总数与数据正确性的问题，并提出了解决方案。

关键词: 分页显示 Oracle Rownum 聚合函数

查询数据库是众多基于Web的应用项目中经常使用到的一项功能。在很多的Web项目中，数据库中的数据量往往非常巨大，如何兼顾查询数据库时的效率问题以及查询后的数据的页面显示问题是必须考虑的重要问题。分页显示是当前显示查询后的数据结果普遍采用的显示方式，其查询操作所涉及到的查询条件多种多样，在分页显示时如何准确的定位记录，获得正确的总记录条数等问题在不同的查询条件下实现也不尽相同。本文以基于Struts架构北京工业大学教育在线信息反馈系统的开发为载体，介绍项目中巧妙解决分页显示中复杂查询带来的查询条目总数与控制查询数据准确性的问题。

一、分页显示方式技术简介

1.一次性读取所有查询结果

细分这种方式还可以分为两种，一种是每次查询都获得所有数据，然后根据游标定位，显示该页面的数据，丢弃不属于该页面的数据结果集。此方法实际上多次查询了数据库，并每次均获得完整的查询结果集，对内存浪费严重；另一种是，先将查询结果一次性获得，然后存入bean或者HttpSession中，翻页的时候从缓存中读取该页显示的数据，从而实际查询数据库只需要一次。这种方式是目前使用广泛的方式之一，实用性强，但是也有一定的弊端，比如在第一次查询数据库时如果数据量很大则往往会比较慢，而且对于数据实时性很高的系统来说，一次查询并缓存容易产生过期数据的问题。

2.分页多次查询数据库

分页多次查询数据库，实现思想为每次仅查询当前页面显示的数据，在翻页时再次连接并查询数据库，得到下一页的显示数据。这样查询虽然每次翻页都需要查询及关闭数据库，但是由于每次查询的数据较小，每次占用内存也较小，查询速度也比较快。

本文基于的项目是采用了分页查询数据库这种方式,这样可以充分的节省对内存以及数据库资源的占用,同时保证了每次查询的时间均衡且比较快速。

二、Oracle数据库Rownum解析

北京工业大学教育在线平台采用的后台数据库为Oracle10g，Oracle有一个先进且实用的特性：Rownum。本项目的分页查询的核心就是基于Oracle数据库中的Rownum特性。

Rownum是随着查询结果集产生的伪列，是Oracle数据库为查询返回行顺序的编号。它是先查到符合条件的结果集之后再加上去的一列，即对结果集进行排序，这个列必须是从1开始排列的。当Rownum的值大于限定条件中指定的值则不再返回更多的行。需要注意的是，当Rownum直接作用在where条件中时，不可以单独使用>，>=以及between and，Rownum只是返回小于（或等于）限定值的行。例如

收稿日期: 2009-01-19

作者简介: 韩强,研究生。周恕义，教授。

对于Rownum>5来说，Oracle读出的第一条记录序号为1，不符合Rownum>5的条件，所以被删除，接下来的数据就成为了新的第一条，序列号仍然为1，仍然不符合限定条件而被删除，最终将发现没有符合条件的结果集。同样道理，如果要使用 = 号，只能在Rownum=1时有效。

由于分页的实现是基于Rownum，所以必须要解决在一个范围内读取结果集的问题，并且这个范围是m<Rownum<n类型的一种范围，下文将用到SQL的嵌套查询解决此问题。

三、复杂查询中的问题及解决思路

信息反馈子模块中使用的查询含有聚合函数及分组关键字，使用分页多次查询数据库的方式实现分页显示，遇到的问题主要有两个。

1.m<Rownum<n范围的实现

限定Rownum的上限和下限需要通过嵌套查询实现。 Rownum直接作用于where条件中时不可以使用 > 号，但是可以通过给Rownum起别名的方式来实现。要实现Rownum的两端限定，可以将别名结合嵌套查询来实现。本项目中用到的SQL语句如下：

```
Select * from (select my_table.*,rownum as rnm from
 (select countSql) my_table where rownum <= (startRow+rows-1) ) where rnm>=startRow;
```

countSql 代表不同条件下的查询语句，其中包含按指定顺序的排序，从而解决按照Rownum排序有可能产生乱序的情况；startRow为开始显示的行序号，rows为当前页面显示多少行；rnm是子查询中Rownum的别名，可用于外查询where条件中。

2.聚合函数和分组关键字引起的条目总数和查询数据准确性的问题

(1)分页查询功能模块

北京工业大学教育在线平台是一个基于Struts架构的Web平台，分页显示中查询数据库的工作由单独的JavaBean实现，在每次不同的查询操作时，调用该模块实现对某一页数据的查询与读取。分页查询功能模块的主要控制类为：

```
OraclePageSQL.java //组合实现分页查询语句
Pagination.java    //获得当前页数据
```

Pagination类负责计算结果集的总行数（countRows函数），总页数（countPages函数），并将总行数、每页显示行数和查询语句三个参数传递给OraclePageSQL类，由OraclePageSQL类将参数组合成嵌套SQL查询语句pageSql并传递回Pagination类调用getPage函数得到当前页显示的结果集。

(2)一般查询计算结果集总行数的方法

简单的查询语句一般采取的方式为：截取源查询语句”from”后的字段与select count(*) count组成计算结果集总行数的查询语句，然后连接数据库获得总数。为保证通用性，本项目中的countRows函数中也采用了这种处理方式：

```
int fromPos = countSql.indexOf("from");
countSql = countSql.substring(fromPos); //countSql用于查询结果集总行数的sql语句
countSql = "select count(*) count  " + countSql;
```

组合成为新的查询语句 select count(*)from tablename where xxxx-condition;

(3)查询语句中聚合函数与分组引起的问题

项目中的信息反馈模块对查询的要求比较复杂，不仅需要统计任意学年和学期的数据，还涉及到了不同学年不同学期的统计求和。这就需要用到聚合函数sum() 以及group by 分组关键字。

源语句：

```
select course_name,course_id,sum(xkxss) sum_xkxss,sum(jkrs) sum_jkrs,sum(fwcs) sum_fwcs,sum(kjs) sum_kjs,sum(fbzys)sum_fbzys,sum(jxdgs) sum_jxdgs,sum(bjs) sum_bjs,sum(tjzys) from view_sys_course_info_course where xueqi=’上’ group by course_name,course_id;
```

view_sys_course_info_course是需要查询的表/视图，暂称为目的表/视图。

倘若直接套用上述的函数，源语句截取后成为

```
select count(*) count from view_sys_course_info_course where xueqi=’上’ group by course_name,course_id;
```

查询后会出现若干行值为1的结果集，原因就是group by 关键字对结果集进行了分组，而由于截取并重组后的语句没有了对分组的限制。

根据保证模块通用性同时能够处理含聚合函数查询语句的解决思路，实际操作中可以通过对原

Pagination类进行继承，在子函数中实现countRows函数以适应复杂的查询。

四、含聚合函数查询语句解决方法详细解析

m<Rownum<n的实现已经介绍过，思路清晰，通用性强。下面结合项目分四步介绍如何处理带有聚合函数的查询语句。

1.第一步 Group by分组关键字处理

由于截取SQL语句后，失去了必要的限定条件，group by分组关键字会导致对结果集不正确的分组，要避免出现多行数值为1的结果集,首先要去掉group by。

```
int groupPos=countSql.indexOf("group by");
if(groupPos!=-1)    //加入判断，是否语句中含有group by关键字
countSql = countSql.substring(fromPos, groupPos);
else
   countSql = countSql.substring(fromPos);
```

通过截取from 与 group by 之间的部分语句，可以得到唯一的一条统计数字。

源语句截取并重组后将会得到语句：

```
select count(*) count from view_sys_course_info_course where xueqi='上';
```

此时会得到所有满足条件xueqi='上'的结果集总行行数，但是由于sum()求和条件被舍弃所以此时计算出的总行数总是>=sum()之后应得的总行数。所以仅去掉group by限定词仍然无法保证获得准确的结果集总行数。

2.第二步 获得正确的结果集总行数

既然结果集大于需要的正确结果集数是由于忽略了聚合函数sum()的限定条件造成的，那么保留sum()函数即可得到正确的行数。又由于聚合函数必须和分函数group by 一起使用所以可以考虑利用建立视图的方式来解决。如上例可以建立一个视图view_course_sum:

```
Create view view_course_sum as
select xxxx（摘自源语句，省略）
from view_sys_course_info_course where xueqi='上'
group by course_name,course_id;
```

建立视图后，使用select * from view_course_sum 可以直接读出满足源查询语句条件的结果集，把源语句简化为一个一般的SQL查询语句，直接调用分页查询功能模块。

建立视图的思路解决了总行数准确性的问题，但是如上例针对每个不同条件的查询，如在对课程信息进行统计时，不同学年、不同学期以及两者的不同组合会带来若干种查询条件，如果对每个查询条件都做视图处理过于繁杂，而且其局限性不利于今后对系统的升级与维护。为了解决这个问题可以采取“曲线解决”的方式：

建立一个求和视图view_sys_course_info_sum:

```
Create view view_course_sum as
select course_name,course_id,sum(xkxss) sum_xkxss,sum(jkrs) sum_jkrs,sum(fwcs) sum_fwcs,sum(kjs) sum_kjs,sum(fbzys)sum_fbzys,sum(jxdgs) sum_jxdgs,sum(bjs) sum_bjs,sum(tjzys)
from view_sys_course_info_course
group by course_name,course_id;
```

此视图去掉了where条件限制，仅仅用求和分组条件限制查询数据。然后使用如下嵌套子查询语句：

```
Select * from view_sys_course_info_sum where course_id IN(select course_id from view_sys_course_info_course where xueqi='上');
```

在子查询中，先将满足xueqi='上'的course_id从目的表/视图中查出，然后通过course_id限定范围，使得外层查询只能在符合子查询条件的结果集中查找，这样就可以获得正确的结果集总行数。

3.第三步 控制结果集数据的准确性

上例中where限制条件在建立视图时先被去除，虽然结果集总行数正确，但是先求和再条件查询会导致查询出的结果值总是一个大于实际值和值而不是where限定条件下的和值。为了保证查询出结果是正确的，在查询结果时仍然使用源语句查询。源语句的查询、连接数据库方式使用常用的方式即可，此处不做详细介绍。

4.第四步 获得正确的总行数和结果集数据

为了既能保证结果集总行数的正确以及结果集数据的准确性，采取分开处理的方式。

首先，使用嵌套子查询得到正确的结果集数

（总行数）

以课程查询为例，在FindCourseAction类中（FindCoursePage继承自Pagination类）：

```
page=new FindCoursePage(sql,item,orderby,seq);
```

其中参数sql为源语句，含有sum及group by关键字，传递到FindCoursePage类中，在初始化时，通过this.setSql调用父类中的setSql函数，然后在父类的setSql函数中保存源语句并调用countRows函数和countPages函数计算总行数和总页数。

对Pagination类进行如下简单的修改：

(1)在setSql函数中加入判断：

```
int sumPos=sql.indexOf("sum(");
if(sumPos==-1)
        this.rowsCount = countRows();
else
this.rowsCount = countRows(sql);
```

使得当含有sum聚合函数时，调用countRows(sql)函数，不含有sum聚合函数时仍然调用原countRows()函数。

(2)在Pagination类中建立抽象函数countRows(sql)

```
protected abstract int countRows(String sql) throws Exception;
```

当调用countRows(sql)函数时，执行子类中的countRows(sql)函数。例如子类：课程查找类FindCoursePage继承自Pagination类，在该子类中通过如下方式获得正确的结果集总行数（总行数）：

```
String countSql=sql;                    //sql为传递的sql语句参数，即源语句
countSql = countSql.toLowerCase();
int fromPos = countSql.indexOf("from");
int groupPos=countSql.indexOf("group by");
countSql = countSql.substring(fromPos,groupPos); //截取from与 group by间的语句
countSql ="select count(*) count from view_sys_course_info_sum " +
                        "where course_id IN(select course_id "+countSql+")";
```

其中view_sys_course_info_sum是建立的求和视图，利用文章前面提到的“曲线解决”方法，上例中的嵌套子查询where条件为：where course_id IN(select course_id from view_sys_course_info_course where xueqi=’上’)

重新组合后的countSql语句可以得到正确的结果集总行数，这种处理方式通过子类的继承实现，没有破坏Pagination类的通用性。

其次，使用源SQL语句查找准确的结果集。

使用源SQL语句查找可以保证正确的查询结果。由于FindCoursePage类在执行setSql函数时保存了源查询语句，在FindCourseAction类中可以直接通过调用FindCoursePage类中的getPage函数得到当前页正确的结果集。

```
sql=…(省略源语句)
page=new FindCoursePage(sql,item,orderby,seq);
//得到总行数，总页数，同时存储源sql语句
result=page.getPage(pageNo,orderby+"-"+seq+"+2");
//调用getPage得到当前页正确的结果集
```

通过分成两步处理的思想，既解决了含有聚合和分组关键字的结果集总行数不对的问题，又保证了查询出来的结果的准确性。

五、结束语

本项目使用的数据库为Oracle10g，通过利用Oracle的特性简化了分页查询的实现过程。同时本文通过项目的实现，结合复杂的查询情况具体的分析了分页显示中按页查询读取数据库方式中遇到的问题，即含有聚合函数和分组关键字的复杂查询带来的结果集总行数与结果集准确性的问题，在以往简单的分页查询基础上，分析了复杂情况的处理，增加了分页显示处理的灵活性，对于需要经常对不同条件求和（或其他聚合函数）的数据库查询并分页显示处理上有一定的实际意义。

参考文献

[1]周作建，惠志婷，徐颖.JSP分页技术[J].电脑编程技巧与维护，2007，(8)11-15

[2]徐越人.Oracle Rownum的使用与JSP分页显示的实现[J].计算机与现代化，2007，10