ASP. NET 中数据列表分页方法研究

On the Methods of Data List Pagination in ASP. NET

卢成均　（重庆文理学院 数学与计算机科学系 重庆永川　402160）

摘要：在以数据为中心的动态 Web 应用开发过程中，数据列表的分页显示是必须具备的功能。本文结合 ASP. NET 与 SQL（结构化查询语言）编程特性，深入研究了多种分页显示方法，并进行了相关性能的分析与比较，从而为不同应用环境而选用不同的分页技术，提供了较为完整的技术参考。

关键词：ASP. NET SQL 数据列表分页显示

1 引言

由于网络带宽及阅读时的视觉局限，人们总是希望数据列表能进行分页显示（比如每页显示 30 条），因此，对于 Web 应用系统来说，数据列表的分页显示是必备的功能，实现的方法多种多样，但如何根据网络环境、数据记录的规模、以及访问流量来灵活地使用不同的分页技术，对于提高 Web 应用程序的运行时性能至关重要。本文探讨了在 ASP. NET 编程环境下实现数据分页的多种方法，并分析其性能指标及应用特点。

为叙述方便，我们不妨作如下假定：

- PageNo 表示当前页，其初值为 1；
- PageSize 表示每页大小（记录条数）；
- RecordCount 表示记录集的总记录数；
- PageCount 表示总页数；
- StartIndex 表示所在页的起始记录索引号，初值为 0；
- 数据库环境为 Microsoft Sql Server 2000；

2 各种分页技术上的实现原理

数据列表的分页技术按具体实现办法可分为应用客户端分页与数据库服务器端分页。如果分页逻辑主要使用应用程序编码语言（如 VB. NET、C#等）进行描述，即为应用客户端分页；如果分页逻辑主要使用数据库服务器端 Sql 脚本（如 Microsoft Sql Server、Sybase 等）进行描述，即为数据库服务器端分页。前者依赖于编程语言本身的强大逻辑描述功能，开发效率较高；而后者由数据库服务器来完成整个计算过程，能有效减少网络流量，执行效率更高，且更具广普适应性（对数据库类型没有特别要求，如 Sql Server 、Oracle、Sybase 等；对编程环境也没有具体限制，如 Asp、Asp. net、JSP 等都可以）。显而易见，在下述几种方法中，一、二属于应用客户端分页；三、四、五、六属于数据库服务器端分页。

2.1 利用 DataGrid 的内置分页功能实现分页（方法一）

DataGrid[1] 是 ASP. NET 中内置功能最强的数据列表控件，直接支持记录集分页是其特色应用之一，只需简单地设置几个属性，如 AllowPaging、PageSize，CurrentPageIndex，即能实现 DataGrid 内置自动分页（示例代码如下）。

```
SqlDataAdapter1 = New SqlDataAdapter( " Select * _
From TblName Order By Id " , SqlConnection1 )
DataSet1 = New DataSet( )
SqlDataAdapter1. Fill( DataSet1 , " TblName" )
DataGrid1. DataSource = DataSet1
DataGrid1. PageSize = 30 ' 每页显示 30 条
DataGrid1. CurrentPageIndex = 0 ; ' 第一页
DataGrid1. DataBind( )
```

此法的优点是复杂的分页逻辑对开发人员完全透明，实现较为简捷，从而提高了开发效率；缺点是每当导航到新的一页时，都必须从数据源读取所有记录，所以不适合数据记录较多（如总数超过 10 万条）或用户访问量较大的应用环境。

2.2 利用数据适配器中 Fill 方法实现分页（方法二）

通过 DataAdapter[2] 的 Fill 方法能十分容易地实现

记录集的分页显示(示例代码如下,以 Repeater 列表控件为例)。

本例通过视图状态保存当前页号,动态计算出当前页的开始记录索引号。

```
StartIndex = (ViewState("PageNo") - 1) * PageSize
'在 Fill 方法中使用 StartIndex、PageSize 参数获取当前页
SqlDataAdapter1.Fill(DataSet1, StartIndex, _
PageSize, "UserTable")
Repeater.DataSource = DataSet1.Tables(0).DefaultView()
Repeater.DataBind()
```

通过插入分页导航按钮(一般为链接按钮)并添加相关代码改变 StartIndex 的值实现分页。如:"下一页"导航按钮的代码应为:

```
ViewState("PageNo") = Math.Min _
(ViewState("PageCount"), ViewState("PageNo") + 1)
DataBind1() '调用数据绑定过程重新绑定数据
```

这种方法采用 SqlDataAdapter 的 Fill 方法每次选取相应页的记录子集,从而实现分页计算。分页功能的实现不依赖于任何一种界面控件,因而移植性更好;由于分页逻辑由编程语言本身的功能进行描述,因此属于应用客户端的分页实现,其性能指标与"方法一"类似。

2.3 充分利用索引列的实现分页(方法三)

自定义分页的思想是不必为每一页获取所有的记录,而只是获取当前页所需要的记录子集。但要求数据库表必须具有索引列或标识列,而且索引列在数值上连续,否则对当前页的记录列表计算会出现错误。其关键实现原理如下(以 DataGrid 列表控件为例):

在前述自动分页基础上,将 DataGrid 控件的 AllowCustomPaging 属性值设为 True。并将通过查询获取的记录总数 RecordCount 赋给 DataGrid 控件的 VirtualItemCount 属性。

当前页起始记录与终止记录索引号算法为:

```
StartIndex = (PageNo - 1) * DataGrd.PageSize
EndIndex = StartIndex + DataGrd.PageSize
```

形成 Sql 语句算法为:

```
strSql = " Select * From TableName Where (Id >"
strSql + = StartIndex.ToString() + ") And (Id <="
strSql + = EndIndex.ToString() + ") Order By Id"
```

这种分页方法的优点是只从数据库获取当前页所需的记录子集,可以用于非常大的数据表(如总数在 10 万条以上);分页列表的计算过程主要由数据库服务器来完成,与界面控件无关,其思想可用于 Repeater、DataList 等其他列表控件。不足之处在于数据表的索引列必须在数值上保持连续且有序,实际应用中常因删除无效或过期数据而导致不连续,故其应用范围受到极大的限制。

2.4 改进的利用索引列实现分页(方法四)

为了弥补"方法三"对索引列的苛刻要求,一个变通的方法是利用临时数据表重新生成连续的索引列(示例代码如下)。

```
Dim cmd1 As SqlCommand, strSql As String
'判断若临时表 tmp_table 存在,就删除它
strSql = " if exists (select * from dbo.sysobjects "
strSql + = " where id = object_Id(N'[dbo].[tmp_table]') "
strSql + = " and OBJECTPROPERTY(id,N'IsUserTable') = 1) "
strSql + = " drop table tmp_table"
cmd1 = New SqlCommand(strSql, SqlConnection1)
'打开数据库连接,执行 SQL
SqlConnection1.Open()
cmd1.ExecuteNonQuery()
'建立临时表
strSql = " create table tmp_table(Id int identity(1,1),"
strSql + = " u_id int not null,Primary Key(Id))"
cmd1 = New SqlCommand(strSql, SqlConnection1)
cmd1.ExecuteNonQuery()
'将原表 TableName 中标识列 u_id 复制到临时表
cmd1 = New SqlCommand(" insert into tmp_table(u_id) _
  select u_id from TableName", SqlConnection1)
cmd1.ExecuteNonQuery()
```

```
'统计临时表中记录数赋给 VirtualItemCount 属性
strSql = " select count( * ) from tmp_table"
cmd1 = New SqlCommand( strSql, SqlConnection1)
DataGrid1. VirtualItemCount = cmd1. ExecuteScalar( )
SqlConnection1. Close( )
'执行数据绑定过程,后述
DataBind1( )
```

以上代码片断在数据库中建立一个临时表,其中只有自增标识列 ID 和同原表标识列同名的 u_id 列两个字段,然后将原表的 u_id 列复制进临时表的同名列。为了提高程序性能,最好将以上多个 SQL 语句编制为存储过程。为避免多个用户建立临时表时发生冲突,应确保表名唯一(可随机产生)。两表联合查询实现数据列表分页计算(示例代码如下)。

```
Sub DataBind1( )
  '省略变量定义及起始记录与终止记录索引号算法
   strSql = " Select * From tmp_table, TableName
Where "
  strSql + =" ( Id > @ StartIndex) And ( Id < = @ EndIn-
dex) "
  strSql + =" and ( TableName. u_id = tmp_table. u_
Id) "
  strSql + =" Order By TableName. u_id"
  SqlDataAdapter1 = New SqlDataAdapter( strSql, _
  SqlConnection1)
  '添加参数变量及数据绑定代码较简或与前同,略去
End Sub
```

起始记录与终止记录索引号算法可参考"方法三"。

此方法的优点是标识索引列可以不连续;其缺点是每次新的请求都要将原表标识列复制到临时表中,但由于标识列已建立了索引,在数据量不是太大(10 万条记录以内)的情况下所带来的性能下降基本可忽略。其性能指标基本与"方法三"同。

2.5 利用 Select 语句的 Top 子句实现分页(方法五)

示例代码如下[3]:

```
If PageNo = 1 Then
  strSql = " Select top PageSize * From TableName "
  strSql + =" Order By Id"
Else
  strSql = " select top PageSize * from TableName _
  where ( Id > ( Select Max( Id) from ( Select top _
  ( PageNo - 1) * PageSize Id from TableName _
  Order by Id ) as t) ) Order by Id"
End if
```

此方法的基本思想是先将记录集按索引列排序,然后取出前 PageSize 条记录,并且保证 Id 号大于"当前页"以前的最大索引号。这种方法不要求索引列 Id 的连续性,但要有惟一性。其优点在于分页计算全部由数据库服务器来执行完成,因而可用于任何界面元素的列表分页实现;其缺点是必须依赖于排序列的惟一性。

2.6 改进的利用 Select 语句实现分页(方法六)

实现代码如下:

```
If PageNo = 1 Then
  strSql = " Select top PageSize * From TableName _
  Where ( u_name like '% 内容%') Order By u_name"
Else
  strSql = " Select top PageSize * From TableName _
  Where ( Id Not In ( Select top _
  ( PageNo -1) * PageSize Id From TableName Where
_
  ( u_name like '% 内容%') Order By u_name) ) _
  and ( u_name like '% 内容%') Order By u_name"
End if
```

此方法的基本思想与"方法五"类似,但不再要求排序列具有惟一性。其优越性在于不但实现了从数据库表获取当前页数据记录的目的,而且还实现了按任意字段排序及模糊查询(如只显示某种姓氏的所有记录)功能;检索效率高,能满足大数据量的要求(如 10 万条记录以上)。

3 六种分页方法综合性能比较(见表 1)

对以上各分页方法在实际应用中的分析比较,我们总结出以下综合性能比较表,以便于开发人员能根据不同的数据规模及网络环境选取不同的分页方案。总之,每一种分页方案都有其优点与不足之处,但就某具体应用而言,分页办法可能会有多种选择,开发者可灵活掌握,合理运用;对某一具体算法而言,其性能不可能无限制地提高,如记录集特别巨大(如总数在 100

万条以上),那么在采用较优化算法的同时,应增加硬件投入、改善网络性能来最终提高应用系统的承载能力。

表 1　分页方法综合性能比较表

分页方法	支持记录量	对索引列的要求	对 Repeater、DataList 的支持	实现复杂度	排序支持
方法一	小规模	无要求	不支持	简单	任意列
方法二	小规模	无要求	支持	较简单	任意列
方法三	中大规模	唯一性、连续性	支持	较复杂	索引列
方法四	中规模	唯一性	支持	复杂	索引列
方法五	中大规模	唯一性	支持	较复杂	索引列
方法六	中大规模	无要求	支持	较复杂	任意列

4　结束语

本文讲述了数据列表分页实现技术的基本思想,并进行了总体性能的分析与比较。在实际的 Web 开发中应反复体会这些方法的异同,根据不同的应用需求选用不同的分页方法,对提高应用程序的性能,减轻服务器的资源消耗具有非常重要的意义。(完整的示例源代码下载 Url:

http://test.hrm.cn/SrcCode/DataPage.rar)

参考文献

1　(美)Stephen Walther. ASP.NET 技术内幕[M],马朝晖译,北京:机械工业出版社,2002。

2　微软官方支持网站:
http://msdn.microsoft.com/library

3　朱涛、李云云,基于 ASP.NET 技术的 Web 数据库分页显示[J],电脑与学习,2005.4,2 期。

4　廖望、何俊等,SQLServer2000 案例教程[M],北京:冶金工业出版社,2004。

5　(美)Julian Templeman 等著,Visual Studio.NET Framework 技术内幕[M],邓劲生等译,北京:中国水利水电出版社,2003。