ADO. NET 数据并发处理技术研究

ADO. NET Data Concurrent Processing Engineering Research

王英慧 曲国伟 （山东龙口 烟台南山学院 265713）

摘 要： 数据集作为 ADO. NET 对象模型的标志，作为一个不连接的数据源的副本提供服务。虽然使用数据集通过减少对数据库服务器的高花费的访问而提高了性能，但是它也带来了多个用户试图同时访问相同数据的可能性。现在，ADO. NET 通过“开放式并发”的方法灵活地解决了该问题。本文说明了该方法的工作原理以及如何使应用程序在具有高度可伸缩性的环境中变得更加健壮。

关键词： ADO. NET 并发处理

1 并发问题

并发成为 ADO. NET 的问题的原因是用户通常间接地对数据库进行更改。为了更好地提高性能，ADO. NET 采用了断开连接的方式。也就是要先把数据的副本读取到客户端，那么也容易引起多个用户同时更新一条记录产生数据并发异常。这个操作，很可能是其他用户删除了该行记录，也可能是修改了某个字段。这个问题一直是数据操作中的难点。下面以具体实例进行讨论和分析。

2 解决方案

通常处理并发冲突的方法有两个。一个是保守式并发处理，一个是开放式并发处理。开放式并发是我们所要讨论的重点，这个也是 ADO. NET 推荐使用的方法。使用开放式并发，行锁定仅在 ADO. NET 提交更改的瞬间应用。因此，在第一个用户向该行提交更改之前，另一个用户可以更改相同行。在这种情况下，第一个用户进行的更新引发 DBConcurrencyException 对象。ADO. NET 提供了有助于解决这类问题的功能。

3 开放式并发处理

3.1 更新方式的选择?

一般来说，更新方式有缓存更新和立即更新两种。缓存更新，就是使用 dataadapter. update( ) 方法来进行数据的更新。我们可能在 datagridview 中修改了很多记录，而只进行一次提交，那么这种操作如果从读取数据开始（或上次更新操作）算起，距离的时间越长，也就越容易引起并发冲突。相对来说，立即更新，也就是使用 command. executenoquery( ) 方法，直接来提交对一条记录的修改应该比较快捷，但仍然不能完全避免冲突的出现。

3.2 更新逻辑的选择？

就是按照哪种更新的方式来对数据进行更新操作。它可以是包含所有列；可以是只包含主键列；可以是包含主键列和被更新列；可以是包含主键列和时间戳列。我们来看这 4 种更新逻辑的差异。实例如下：有表 T1（学号，姓名，性别，籍贯，tamp）其中的 tamp 是时间戳列。那么对应的更新逻辑为：

（1）Update T1 set 学号 =?，姓名 =?，性别 =?，籍贯 =? Where 学号 =? And 姓名 =? and 籍贯 =?

（2）Update T1 set 学号 =?，姓名 =?，性别 =?，籍贯 =? Where 学号 =?

（3）Update T1 set 学号 =?，姓名 =?，性别 =?，籍贯 =? Where 学号 =? And 姓名 =?（假设只对姓名列进行修改，而其他列不变）

（4）Update T1 set 学号 =?，姓名 =?，性别 =?，籍贯 =? Where 学号 =? And tamp =?

我们来对这 4 种逻辑进行简要的说明：第 1 种也是默认地一种，把所有的字段都包含进去，那么当有用户 A，B 读取了数据后，A 成功更改了一行记录中的姓名，那么如果 B 再要更改同行记录时，由于 Where 条件中要求满足所有的字段，而这个时候姓名已经改变了，这样他将更新失败。第 2 种方法，它只包含主键，那么也就是

说如果不修改主键(或删除)(修改主键是危险的,应该避免!)的话,都会成功,但 A 更新成功了,B 也更新成功了,但 B 的更新覆盖了 A 的更新,这个时候 B 甚至不知道原来自己更新的时候记录已经有了变化,A 也不知道自己的记录已经被覆盖。这种方法也叫做"后来居上"的方式。也就是后面的覆盖前面的操作。第 3 种方法,既包含主键也包含更新的字段。假设 A 成功地更新了一条记录的姓名字段,B 更新该行记录的性别字段,那么 B 也会成功的实现自己的更新。但同样,如果没有刷新显示的话,他们都不知道该行记录的对应字段发生了变化。在.NET 里这种方式的构建相对来说比较麻烦,写的代码也比较多,是比较消耗时间和精力的。对于最后一种方式,它采用了时间戳列作为更新条件,时间戳能反映记录更新的变化。如果时间戳变化了,那么该记录自然已经被别人更新过了。

这种方法是推荐的方法。当然每种更新逻辑都有自己的好处,我们也不能特意地使用哪种,要根据自己的系统的情况来选择。一般来说,第 1 种方式是 dataadapter 默认生成的,而第 2 种,第 3 种,第 4 种都要我们自己来设置。手动地创建更新逻辑是个费时费力的过程,但更新效率也相对较高。

3.3 如果更新方式和更新逻辑都选择好了以后,那么就要考虑使用事务了。在事务中进行更新操作的结果是要么全部成功,要么都不成功。所以使用事务的程序代码是比较安全的。

类似如下的代码:

```
Dim Mytransaction As OledbTransaction
    Try
      Cn. Open ( )
      Mytransaction = cn. BeginTransaction
      cmd1. Transaction = mytransaction '这里假设已经建立了 cmd1 对象
      Cmd1. ExecuteNonQuery ( ) ?
      mytransaction. Commit( ) '事务提交
    Catch ex As Exception
      Mytransaction. Rollback( ) '事务回滚
      Return -1 '做一些其他处理
    Finally
      Cn. Close ( )
    End Try
```

说明:对于 dataadapter. update 方法,如何使用事务呢? 其实 dataadapter 本身并不进行数据更新,而是它的 insertcommand, updatecommand, deletecommand。那么就如上面一样设置这三个 command 对象的 Transaction 属性就可以了。

3.4 采用开放式并发处理,就要对异常进行相应的捕获,给出相应的提示信息和处理办法。对于那些可能引起异常的代码都要包含在 Try? …… end try 块之间。一般可以采用类似下面的操作:

```
Try
Catch Ex As Data. DBConcurrencyException '并发冲突的异常
  做相应的处理
End try
```

说明:这里的做相应的处理,可以是把最新的行从数据库中读出来更新现有的行(当然如果该行被删除例外),也可以重新填充数据(fill 操作)。这个时候我们可以做出判断,该行是被删除的情况,可以使用一个函数将 Ex. row("ID") 作为参数传递过去,使用一个 command. executeschar 方法来判断是否记录存在,使用 executereader 或 fill 来取得已经变化的记录或刷新全部记录。

ADO. NET 数据集对象的不连接特性提供了很高的性能,也给应用程序带来了数据并发性类型的错误。了解这些错误怎样和为什么出现将使我们很好地捕捉和处理这种错误,给应用程序增加另一个层次的支持,给用户维护下层数据的完整性提供更多的选择。

参考文献

1 (美) David Sceppa. ADO. NET 技术内幕,第 1 版,北京:清华大学出版社,2003 年.

2 微软公司. MSDN Library For Visual Studio 2005 . 2005 年.