基于 ADO. NET 的 ERP 系统性能优化

ERP Capability Optimal Method Based on ADO. NET

杨小平 陆 军 焦有章 （中国人民大学信息学院 100872）

摘要：ERP 系统是频繁访问数据库的信息系统，Microsoft. NET 是最新的软件开发技术，ADO. NET 作为 Microsoft. NET 应用程序的数据访问模型很好的解决了 ERP 与数据的交互问题。本文就以 ADO. NET 数据访问模型为背景讨论了 ERP 数据访问中的优化问题。

关键词：ADO. NET 公共语言运行环境 . NET 数据供应器 XML 数据集

1 引言

ERP 是为了建立规范先进的管理体系，实现企业内部控制的透明化；实现业务活动和经营活动的短流程、高效率，增强企业对外部市场的反应能力而产生的一个企业信息系统. 理所当然 ERP 要和企业各式各样的数据打交道。一般而言，影响数据库访问性能的主要因素有三个：数据库服务器的硬件性能、网络性能以及数据库访问程序的设计。前两个是" 硬" 因素，后一个是" 软" 因素. 目前，前两项的性能普遍得到提高和增强，所以数据库访问的程序设计将直接影响到应用程序的最终性能。而数据库访问程序的设计又可以分为三个部分：选择托管程序 应用 ADO. NET 编写的程序的优化以及后台 DBMS 的优化，后文就从这三个方面来讨论基于 ADO. NET 的 ERP 系统的性能优化，并着重讨论了使用 ADO. NET 进行程序开发的优化。

2 ADO. NET 的概述及其数据结构

ADO. NET 是 Microsoft . NET 应用程序的数据访问模型。它由 ADO 技术发展而成，在某种程度上，ADO. NET 代表了最新版本的 ADO 技术。由于 ADO. NET 建构于. NET 框架之内，它的建立和管理都是基于 CLR（Common Language Runtime ，公共语言运行环境），所以直接或间接地得益于. NET 框架在内存管理、类型转换、对象池等方面的技术改善和优化。ADO. NET 引入了一些重大变化和革新，它对 XML 提供全面的支持，提供了新的非连接数据缓冲模型，使其在构建结构松散的、非链接的应用程序上有着得天独厚的优势 。ADO. NET 有以下两个核心组件：

2.1 ADO. NET DataSet

DataSet 是数据的内存驻留表示形式，也就是我们通俗所说的 IMDB（In Memory DataBase）。DataSet 的设计目的是为了实现独立于任何数据源的数据访问，在本地内存中实现一个数据缓存。它代表一套包含关系表、约束和表间关联信息的完整数据结合（如图）DataSet 对底层的数据源一无所知，它提供了独立于任何数据源的数据访问，因此它可以用于多种不同的数据源，用于 XML 数据，或用于管理应用程序本地的数据。DataSet 可通过多层应用程序的不同层由一个组件传递到另一组件，也可以作为 XML 数据流被序列化，因而非常适合于不同类型平台间的数据传输。ADO. NET 使用 DataAdapter 对象为发送到和来自 DataSet 及底层数据源的数据建立通道。

如图所示，DataSet 主要由两部分组成：DataTableCollection 和 DataRelationCollection，DataTableCollection 包含零个或多 DataTable 对象。一个 DataTable 对象代表驻留内存的数据表。它包含 DataColumn 所表示的列和 Constraint 所表示的约束的集合，这些列和约束一起定义了该表的结构. DataTable 还包含 DataRow 所表示的行的集合，每个 DataRow 对象代表表中的一行数据。DataRelationCollection 代表 DataSet 对象中表之间的关系集合，关系由 DataRelation 对象来表示（如图 1 所示）。

2.2 NET 数据供应器（Data Provider）

ADO. NET 依靠数据. net 数据供应器提供给它的

服务,它们提供了对底层数据源的访问。数据供应器是一组包括 Connection ,Command ,DataReader 和 DataAdapter 对象在内的组件。它实现了对数据库的连接、操作和快速、只读的访问。Connection 对象提供与数据源的连接;Command 对象用来执行数据库命令;DataReader 从数据源中获得高性能的数据流;DataAdapter 对象是连接数据源和 DataSet 对象的桥梁。DataAdapter 使用 Command 对象在数据源中执行 SQL 命令和调用存储过程,以便将数据加载到 DataSet 中,并保持 DataSet 中数据的更改与数据源中的数据一致。System. Data 包含了独立于供应器的类型,如 DataSet 以及 DataTable。

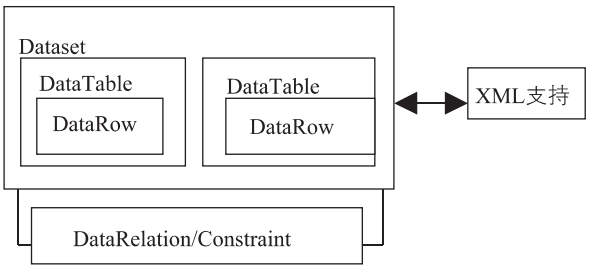

图 1

在数据供应器中 Microsoft 发行了如下两个供应器:

A:SQL Server. NET 数据提供程序。应该说这种供应器针对 SQL Server 的性能是最高的。

B:OLE DB. NET 数据提供程序。这是一个用于管理 OLE DB 数据源的数据提供程序。

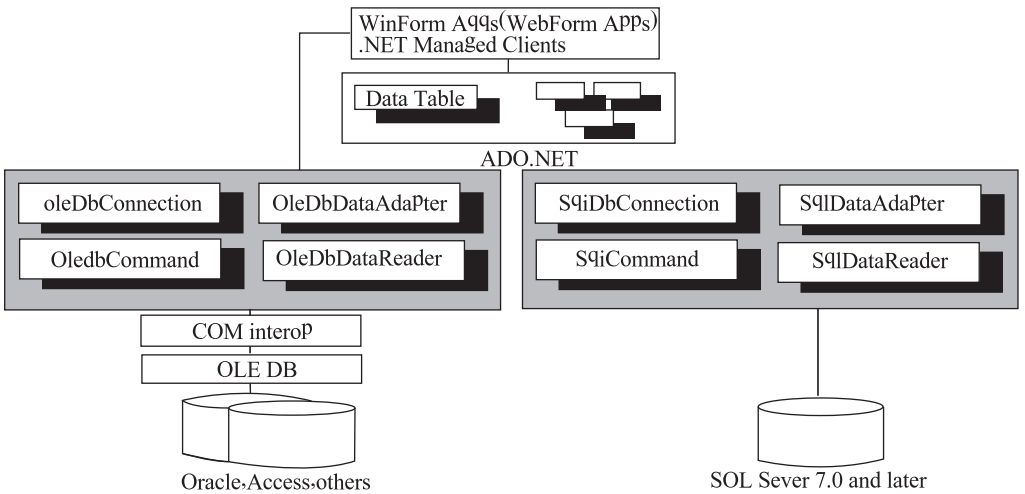

图 2

3 使用 ADO. NET 访问数据库的性能优化

3.1 在选择托管程序方面

选择合适的. NET 数据供应器

ADO. NET 的数据库访问基础是. NET 数据供应器(Data Provider)。其中最常用的是上面介绍的 OLE DB 和 SQL Server 两种,前者是一个用于管理 OLE DB 数据源的数据提供程序。它的效率要低于 SQL Server. NET 数据提供程序,因为在与数据库通信时,它需要通过 OLE DB 层进行呼叫。但它提供了对 Oracle 、DB2、Sybase 等数据库的通用性支持。而后者则是专门为连接 SQL Server 7. 0 以上数据库定制的。由于它使用 SQL Server 本身的 TDS(Tabular Data Stream)数据交换协议与数据库交互,不必通过 OLE DB 层的协议转换,因而效率很高,较之 OLE DB 数据供应器,它的性能可高出 30% ~40% 。对于以 Microsoft SQL Server7. 0 以上版本为后端数据库的应用程序,应该首选该数据供应器以获得最佳性能。

3.2 在应用 ADO。NET 编写的程序方面

(1)“尽早”的关闭连接。由于在整个数据库访问过程中,数据库连接对象都是非常有限和宝贵的资源,所以在使用数据库连接时,不论是否采用了连接池,都应该尽可能晚地打开连接,尽可能早地关闭连接。关闭连接时一定要关闭其间建立的所有临时对象,并结束所有用户定义的事务,这样才能保证连接的正常关闭,避免系统资源浪费。当使用 DataReader 对象时,应该仅在执行 SQL 命令前打开连接,读取结果后即关闭连接,否则数据库连接会一直被占用,不能用于其他目的,应该尽早调用 DataReader 对象的 Close()方法关闭连接。如以下代码片断(C#)从数据库中读取 10 条记录后先关闭连接再进行一定的处理:

```
dbConn. Open( ) ;//打开连接
```

```
SqlDataReader dbReader = SQLCmd.ExecuteReader();// 执行 SQL 命令,返回结果
string[] testStr = new string[10];
int i = 0;
while (dbReader.Read()) // 读取记录
  {
    testStr[i++] = dbReader.GetString(0));
  }
dbReader.Close();
dbConn.Close();// 关闭数据库连接
for(i=0;i<testStr.Length;i++)
  {
    testStr[i] = process(testStr[i]);// 调用方法 process() 对结果进行处理
  }
```

(2) 合理使用 DataSet 架构中的键和约束。对于开发客户端/服务器应用程序的程序员来说,最令他们头疼和讨厌的是到数据库的往返行程,尤其是在违反架构规则而导致错误产生的数据库的往返行程。当然在 Microsoft Studio 的托管语言 C#提供了 try – catch 这种类似 java 的异常捕获和处理功能,但是这趟"无效"的网络往返还是给用户带了时间等待上的"不快"。在这个方面 DataSet 的架构就非常出色,完全能够避免这种情况。

比如在 ERP 系统初始设置中的客户关系管理模块中当然不允许出现重复的客户 ID(CustomerID),这时我们就可以为 DataSet 中的客户表(Customer_Table)定义一个如下的主键约束:

```
Customer_Table.PrimaryKey = new DataColumn[]{Customer_Table.Columns["CustomerID"]}
```

这时当我们由于某种误操作产生相同编号的客户记录时,DataSet 就会很"智能"的发出"错误",我们可以用 try – catch 去捕获这个"错误"。注意了这个错误是由客户端内存中的 DataSet 根据我们"告诉"的"规则"抛出的,而不是由后台的数据库经过处理后由网络传回来的。所以这种减少数据库往返的方法大大提高了应用程序时间上的效率。

(3) 运用 DataSet 架构中的关系实现 DataTable 间的导航。我们在前面说了 DataSet 是作为 IMDB 而闻名的,作为关系数据库中经典的关系在 DataSet 中是用 DataRelation 来模拟的。当我们定义了 DataSet 各个表(DataTable)中的外键的时候,DataConstraint 就可以和 DataRelation 配合使用,以达到表之间的一个导航连接。那么 DataSet 中的导航机制是如何来帮助提高应用程序的性能的呢?

让我们以 ERP 系统为例子:我想看看所有客户的与其对应的所有订单情况,可以先初始化一个 DataSet,其中包括两个 DataTable(Customer_Table 和 Order_Table),然后以客户编号为外键建立两个表间的关系。最后从物理数据库中将两个表的所有记录读出放到两个表中。当数据绑定到 DataGrid 以后,其间的导航功能显而易见了。强大的数据操纵和导航机制才是 DataSet 的真正魅力所在。

(4) 运用 DataSet 或运用 SQL 语句更新数据库的选择。当我们在客户端修改(包括 Update Insert 和 Delete)完数据记录时,可以把相应的修改在程序里由程序员"翻译"成 SQL 语句"抛"给后台的数据库服务器;或者说我们也可以将整个修改后的 DataSet"抛"给 ADO.NET 数据库访问层的托管程序,由它运用 CommandBuilder 来自动生成 SQL 语句去修改数据库。应该说两种方式各有各的益处:前一种情况在当修改的数据较少的时候比较合适。后一种情况则在对客户端的 DataSet 进行了很多的修改,相应的 SQL 语句不容易写的时候比较合适。应该说后一种情况的执行效率会略低一些,但它却换来了程序编写方面的效率。在实际的 ERP 系统中我们要恰当合理的运用这两种方式。

(5) 合理的使用 DataReader。DataReader 是一种轻量级的数据流,它允许你每次读入一行数据(modulo TDS buffering),进行一些处理,然后丢弃这一行。从而在中间层或客户端节约内存,但是这种方式浪费了另一种宝贵的资源:数据库连接。与之相对应的是 DataSet,它作为 IMDB 允许一次将所需要的数据全部都提取到内存中,而后释放连接再进行处理,这样保证了数据库连接的资源但是却忽视了客户端的内存负担。那么到底应该在 DataReader 和 DataSet 中舍谁取谁呢?

在 ERP 系统中我们视具体情况而定:比如涉及生产管理的数据量很大(几十甚至上百兆的数据),并且只是浏览或者说处理的时间不长,那么我们就没有必要选用 DataSet 来缓存这些数据,这个时候用 Da-

taReader 是明智的选择。又如在成本核算部分有关定价参考信息的数据量不是很大,而且一直要在客户端存在,那么这个时候 DataReader 是绝对不能用的,因为如果使用连接这种宝贵资源就被一直占用了。

(6) 尽量不在查询中使用数据库服务器的排序功能(Order by)。我们知道在 SQL 语句中使用 Order by 这样的功能要求会增加数据库服务器的工作负担,从而在多用户的情况下降低查询的性能,而 ADO. NET 中的 DataSet 支持数据在本地(客户端)进行基于非 SQL 引擎的查找和排序操作,这样通常会更快,而且减轻了数据库服务器的负荷。

3.3 在后台物理数据库方面

善用数据库的存储过程。与 SQL 命令相比,数据库的存储过程不需要每次都重新分析和优化,而是一次性编译成执行计划缓存在数据库中,以后可以直接调用,避免了重复的解析过程,节省了时间。特别是当查询任务由一系列 SQL 命令组成时,应该考虑将其编写成存储过程,这样能够避免数据在网络上来回传送,降低网络流量。

综观以上几种方法,在程序设计中优化和改善数据库访问性能的基本原则是:尽量减少资源的开销,尽量缩短系统资源特别是数据库连接的占用时间,尽量避免不必要的数据映射和转换。

4 结论

数据库访问是一项频繁而又重复性的操作,在目前绝大多数 ERP 应用系统中有着举足轻重的地位。基于微软 ADO. NET 技术把数据层设计成一个通用的数据库访问组件以供上层使用,大大降低了应用程序访问数据库的复杂性,使得各层的结构更加清晰,同时如何发挥它最大的优势,减少系统瓶颈,提高应用性能,更是一个复杂而艰巨的任务。本文在初步分析 ADO。NET 数据库访问机制的前提下,分析了一些系统瓶颈产生的原因,提出了一些改善性能的思路和方法,在笔者实际开发基于 。NET 框架的 ERP 中得到了应用并取得了良好的效果。

参考文献

1 David Platt Introducing the Microsoft . NET , Microsoft Press 2001.

2 David Sceppa Microsoft ADO. NET(Core Reference) Microsoft Press 2002.

3 Visual Studio · NET 开发环境使用指南[M],清华大学出版社,2001 ,7。

4 Rebecca M. Riordan AD0. NET 程序设计[M],清华大学出版社,2002。

5 Bob Beauchemin ADO. NET 本质论[M],清华大学出版社,2003。

6 Michael Otey/Denielle 0tey ADO. NET 技术参考大全[M], 清华大学出版社,2003。