第37卷第11期 合肥工业大学学报(自然科学版) Vol. 37 No. 11
2014年11月 JOURNAL OF HEFEI UNIVERSITY OF TECHNOLOGY Nov. 2014

Doi:10.3969/j.issn.1003-5060.2014.11.007

多用户大数据量的数据库访问优化与设计

谢姗姗, 周国祥, 石 雷

(合肥工业大学 计算机与信息学院,安徽 合肥 230009)

摘 要:文章首先介绍了当前较为流行的ADO.NET数据库访问模型,以此为基础,设计了一套综合运用数据库连接池和存储过程的数据库访问方法,并通过程序代码的巧妙改进有效地提高了批量操作时的访问速度。目前,该方法已经在某珠宝销售公司的货品仓储和销售管理信息系统中得到应用,效果良好。

关键词:管理信息系统;B/S模式;数据库访问效率;性能瓶颈;性能优化

中图分类号:TP317.1 **文献标识码**:A **文章编号**:1003-5060(2014)11-1311-06

Access optimization and design of database with multi-user and mass data

XIE Shan-shan, ZHOU Guo-xiang, SHI Lei

(School of Computer and Information, Hefei University of Technology, Hefei 230009, China)

Abstract: In this paper, ADO.NET, the current popular database access model, is introduced. Then a database access method which makes a comprehensive use of stored procedures and database connection pool is designed. Through the clever improvement of program code, the access speed of the batch operation is effectively improved. Currently, this method has been applied effectively in the goods warehousing and sales management information system of a jewelry sales company.

Key words: management information system; B/S mode; database accessing efficiency; performance bottleneck; performance optimization

随着信息化时代的来临和互联网技术的不断发展,以B/S模式为中心的管理信息系统已成为企业信息化管理的重要支撑[1]。B/S模式下的管理信息系统通常具有以下共同特征:多用户访问;数据量庞大;具有强大的数据库作为后台支撑;需要频繁地与数据库交互以为网络服务提供准确、及时的数据信息[2]。以某珠宝销售公司为例,该公司已运营十几年,有近百个网点,遍布全国各地,销售数据量达几十万条。为了实现从原材料采购、货品生产到珠宝销售等整个流程的信息化管理与高效的数据分析体系,该公司开发了货品仓储和销售管理信息系统。

对于一个成功的管理信息系统来说,除了能够很好地完成用户的各项功能,还需注重用户体验的访问速度。该货品仓储和销售系统涉及诸多销售网点、数据量庞大且日常数据变化繁多,为了保证系统在使用时能够及时地响应数据的变化,要求系统在进行数据库操作时要尽可能的快。一般而言,可从以下2个方面提高数据库的访问速度:① 在硬件上,可以使用架构更优的数据库服务器[3]、更大的网络带宽[2]、更优的内存分配管理方法[4]等;② 在软件上,需要注重数据库访问程序的设计与优化,例如使用更优的程序结构、高效的SQL语句、存储过程[5]或创建数据库索引[6]等。目前,硬件上的性能普遍得到了提高,所以数据库访问的程序设计与优化将直接影响到应用程序的最终性能[7]。

在管理信息系统的开发过程中,需要从多方

收稿日期:2013-11-23;修回日期:2014-04-28

基金项目:国家自然科学基金重点资助项目(60633060)

作者简介:谢姗姗(1989-),女,安徽合肥人,合肥工业大学硕士生;

周国祥(1956-),男,安徽合肥人,合肥工业大学教授,硕士生导师.

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

面优化信息系统的性能。本文结合该货品仓储和销售系统的开发经验,以 ADO. NET 数据库访问机制为基础,设计了一套综合运用数据库连接池和存储过程的数据库访问方法,并通过程序代码的巧妙改进有效地提高了批量操作时的访问速度。

1 背景知识介绍

1.1 ADO. NET 数据库访问模型

该系统以 SQL Server 2008 数据库作为后台支撑,采用 ADO. NET 数据库访问模型,该数据库访问模型是 Microsoft 为解决 Web 和分布式应用程序问题而设计的。ADO. NET 与早期的数据库访问模型相比具有互操作性好、可伸缩性强、更易维护、全面支持 XML 并提供非连接缓冲模型等诸多优势。同时,它为不同数据源提供了一些公共的数据库类库来屏蔽各种数据库操作的差异性,为应用程序的开发提供了统一的接口,因此被广泛应用于各类数据库应用程序的开发。

ADO. NET 的数据库访问基础是. NET 数据供应器,ADO. NET 提供了多种数据供应器:SQLServer. Net 数据供应器、OLE DB. Net 数据供应器等。各. NET 数据供应器都实现了相同的基类,即 Connection、Command、DataReader、DataAdapter 和 DataSet,除了 DataSet,其他的实际名称都取决于对应的数据供应器[8]。ADO. NET 对象模型结构如图1所示。

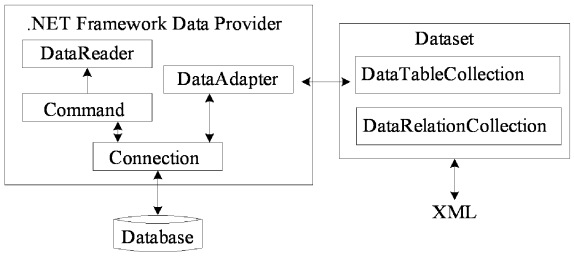

图1 ADO. NET 对象模型结构

1.2 公共基础类

为了实现代码的重用与统一管理,提高程序的运行效率,在管理系统的开发过程中将常用的操作封装成公共基础类。在该货品仓储和销售系统中,将对数据库操作的所有公用函数封装成 DataBaseFunc 类。

DataBaseFunc 类定义如下:

```
public class DataBaseFunc
{
    OpenConnection();//连接的新建或调用,统称为连接的打开
    CloseConnection();//连接的关闭或回收
    AddValue();//插入操作
    ExcuteSqlTable();//SQL 语句执行操作
    ⋮
}
```

1.3 批量操作

批量操作,可通俗地理解为在同一时间段连续完成 n 条相同性质的数据记录与数据库的交互。例如,批量插入、批量更新等。批量操作会频繁地访问数据库并且对访问速度有很高的要求。在该货品仓储和销售系统中就涉及对上万条进销存数据的导入导出。

2 数据库访问设计与优化

2.1 连接池

2.1.1 连接池设计与优化原理

传统的数据库访问模式为:新建一条数据库连接;执行 SQL 语句,对数据库进行相应的操作;关闭数据库连接[9]。创建一个数据库连接需要在服务器与客户端之间进行多次网络往返,以完成建立通讯、分配资源、用户验证、安全上下文配置等任务,可见创建数据库连接是一个极为耗时的操作;当数据库连接打开过多时,可导致内存不足、CPU 占用率急速上升,情况严重时,可导致服务器宕机或其他软件无法正常运行,可见数据库连接是一种消耗资源并危险的操作[10]。同时,数据库连接具有最大数目的限制,所以建议连接用完之后立即关闭,以避免重启数据库或服务器的发生。

对于简单的数据库应用程序来说,由于数据库访问不是很频繁,所以可在需要访问数据库时新建连接,用完关闭,这样的操作不会带来明显的性能开销。但是对于复杂的应用程序来说,例如本系统涉及近百个网点,加上总部用户需要频繁地操作系统,在传统模式下,每个终端都需占用独自的线程针对数据库操作,频繁地新建、关闭连接,消耗大量的时间,降低系统的性能,出现瓶颈效应。针对此问题,引入连接池机制。连接池作为一个中间层与数据库交互,统一管理这些终端的请求,通过池内有限的数据库连接为大量网点的并发访问提供服务,实现池内连接的高度共享[11],避免了数据库连接频繁新建、关闭带来的

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

开销，进一步提高数据库访问的效率和应用程序的可扩展性。

对于批量操作，引入连接池机制，同样具有很大优势。例如在该货品仓储和销售系统中以 Excel 文件形式导入 n 条珠宝原材料采购信息记录，在引入连接池机制以后，数据库连接的新建次数从大于 n 次降至 p 次。在批量操作的整个过程中，若池内总存在空闲连接，则批量操作无需新建连接，即 $p=0$；若存在的空闲连接不够用，则需在池内新建连接对象，新建的数量不超过 Max Pool Size－Min Pool Size，即 $p \leqslant$ Max Pool Size－Min Pool Size。因此 $0 \leqslant p \leqslant$ Max Pool Size－Min Pool Size（其中 Min Pool Size 为连接池所允许的最小连接数，Max Pool Size 为连接池所允许的最大连接数，且 Max Pool Size 远小于 n）。使用连接池前后批量操作实现流程对比如图 2 所示。

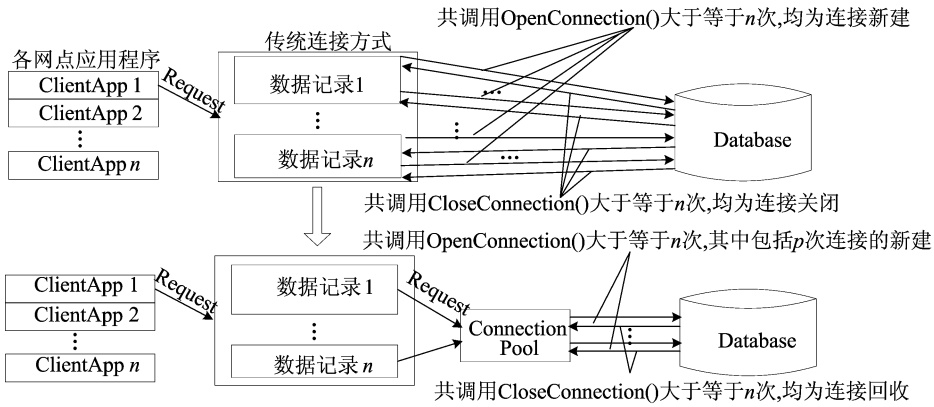

图 2　使用连接池前后批量操作实现流程对比

2.1.2　连接池在程序中的配置

ADO.NET 数据供应器提供了透明的连接池，用户无需自己开发连接池机制，只需对连接池参数进行相应的配置，便可方便、高效地使用连接池。在该系统中，连接池的连接字符串参数配置如下：

```
ConnString="Data Source=.;Initial Catalog=NewBRY;Persist Security Info=True;User ID=sa;Password=123;Max Pool Size=55;Min Pool Size=5;"
```

2.2　程序代码设计与优化

2.2.1　程序代码设计与优化原理

在该货品仓储和销售系统中，由于原材料的多样性，所以每种原材料都可能具有不同的属性，这些属性可在系统进行自定义设置，属性值将根据所属的不同数据类型划分并添加到原材料动态属性值表的相应字段。例如，string 类型属性值添加到 StrValue 字段；bool 类型属性值添加到 BoolValue 字段等。这些属性称为“动态属性”，其他固有属性称为“基本属性”。

对于以 Excel 文件形式导入 n 条珠宝原材料采购信息记录的批量操作，每导入一条原材料记录，需进行以下 2 步（前提保证 Excel 表中包含动态属性值）：

（1）向原材料基本信息表中一次性插入原材料的各基本属性值，并返回该行 ID 值（记为 Id1）。

（2）伴随动态属性表中各动态属性的 ID 值（记为 Id2）与 Id1，依次将各动态属性值插入动态属性值表。string 类型的动态属性值在动态属性值表中的存储形式见表 1 所列。

表 1　动态属性值表中 string 类型属性值记录

ID	RelationID1	RelationID2	BoolValue	StrValue	…
1	Id1	Id2		属性值 1	

在以上的过程中进行多次数据库连接的打开与回收。若导入的原材料记录包含 m 个非空动态属性值和 s 个基本属性值，则导入该记录需要进行 $m+1$ 次插入，这里不考虑插入前的其他逻辑查询操作。由于网络环境、服务器硬件环境、SQL 语句的复杂度各异，每次访问数据库的时间可大可小，导致池内空闲连接对象个数也会不同。因此插入该记录进行了 n_1 次数据库连接的调用与 n_2 次数据库连接的新建（$n_1+n_2=m+1$，且 $0 \leqslant n_2 \leqslant$ Max Pool Size－Min Pool Size）、$m+1$ 次的连接回收。可见，当原材料记录急剧增加时，通过池管理程序频繁打开、回收连接，造成的延迟也是不可忽视的。

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

为了实现批量操作的高效性,在连接池打开的情况下,对公共基础类DataBaseFunc的程序结构做了进一步改进:在该类所有数据库操作的函数中添加布尔型参数blank来区分是否为批量操作。若某操作为非批量操作,则在主程序中将blank设置为false,由于在DataBaseFunc判断blank=false,则需在处理该操作前打开数据库连接,待该操作完成后再关闭数据库连接。若需进行批量操作,则应在主程序中先打开数据库连接并将blank参数设置为true,由于在DataBaseFunc类的数据库操作函数中判断blank始终为真,则之后的操作都无需开关连接,直到批量操作完成后在主程序中关闭连接。

代码改进后的批量操作方案使得在整个操作过程中,只进行了1次数据库连接的打开与回收,进一步缩短了与数据库交互的时间,如图3所示。

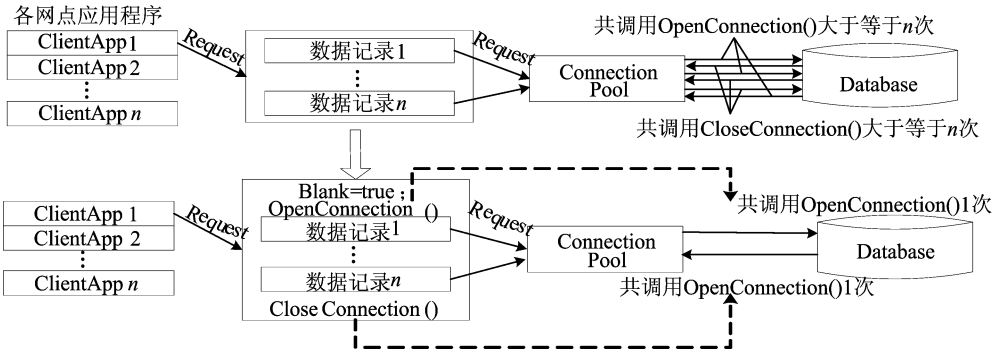

图3 代码改进前后流程对比

2.2.2 使用代码改进在程序中实现批量操作

使用代码改进实现批量操作的程序如图4所示,其中程序均以SQLServer.Net数据供应器为例。

主程序

```
private DataBaseFunc cprivDBO = new DataBaseFunc ();
cprivDBO. OpenConnection();
                ……
DataTable dt =
cprivDBO. ExcuteSqlTable( sSQL, true) ;
                ……
cprivDBO. CloseConnection();
```

调用

DataBaseFunc类

```
public DataTable ExcuteSqlTable( string sSQL, bool blank)
{
    DataTable dt = new DataTable();
    if( sSQL != "")
    {
        if(! blank)
        {
            OpenConnection() ;
            dt = BlankSql( sSQL); //SQL 执行
            CloseConnection();
        }
        else
            dt = BlankSql( sSQL);
    }
    return dt;
}
```

图4 改进方案程序代码

2.3 存储过程

2.3.1 存储过程设计与优化原理

改进后的批量操作方案,使得在整个操作过程中数据库连接打开与回收的次数降至1次。但由于以上操作都是由嵌入客户端的SQL语句完成的,且每执行一条SQL语句都需临时将其送入数据库服务器进行编译和优化执行,因此对于批量操作来说,则需要不断重复这些操作,无疑又增加了系统的性能开销。在程序代码改进的基础上,综合使用存储过程来代替以上嵌入代码的SQL语句,便可回避这些弊端,改善数据库访问的性能。存储过程是一组为了完成特定功能的SQL语句和流控制语句的集合,该集合被存放在数据库服务器端,在创建时便对其进行分析和编译,生成优化的可执行方案,以供外部程序调用[12]。对于以上的批量操作,将处理每一条记录的SQL插入语句封装到存储过程中来代替嵌入代码的SQL语句,使得整个批量操作的编译次数

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

降至 1 次，从而进一步节省了操作时间。改进批量操作前后流程对比如图 5 所示。

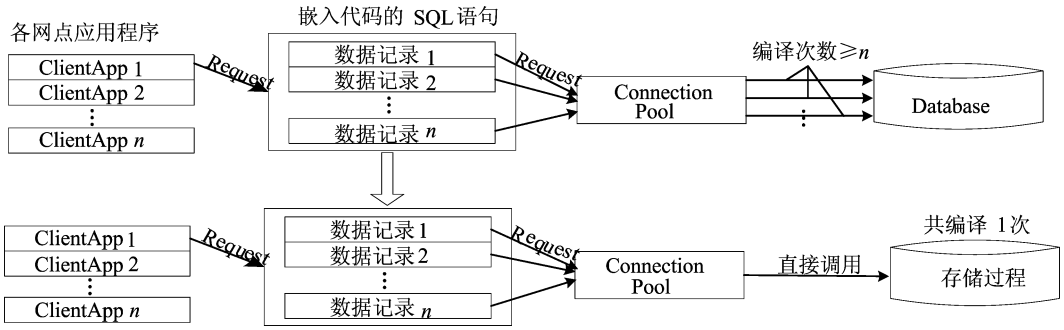

图 5　使用存储过程改进前后流程对比

2.3.2　使用存储过程在程序中实现批量操作

对于该系统的批量操作，编写存储过程的同时需在 DataBaseFunc 类中添加相应的数据库操作函数 ExecuteProcedure（string procedureName，SqlParameter[] sqlPara，bool blank）。使用存储过程改进方案实现批量操作的代码如图 6 所示。图 6 中 PurchaseOrder_Insert 为存储过程名[13]，Name 为存储过程调用的一个参数。

主程序

```
cprivDBO. OpenConnection();
SqlParameter[] sqlParams = new SqlParameter[]
{ new
SqlParameter("@Name", SqlDbType. VarChar,50, Parameter
Direction. Input, true,0,0,"Name", DataRowVersion Default,
"")
                ……
};
sqlParams[0]. Value = sName;
                ……
cprivDBO. ExecuteProcedure("PurchaseOrder_ Insert", sqlPa
rams, true);
cprivDBO. CloseConnection();
```

调用

DataBaseFunc类

```
Public bool ExecuteProcedure ( string procedureName,
SqlParameter[] sqlPara, bool blank)
{
    ……
    if (! blank)
        this. OpenConnection();
    this. sqlCommand= new SqlCommand(procedureName,
this. sqlConn);
   this. sqlCommand CommandType=
CommandType. StoredProcedure;
    for( int i = 0; i < sqlPara. Length; i ++)
        this. sqlCommand. Parameters. Add( sqlPara[i]);
   this. sqlCommand. ExecuteNonQuery();
   if(! blank)
        this. CloseConnection();
    return true;
}
```

图 6　使用存储过程改进方案程序代码

3　实验结果

在以下实验中，将选取多个测试用例，并对每个测试用例分别进行 100 次测试，取其结果的平均值作为最终的实验数据。

实验一　传统模式下创建连接对象与使用连接池创建连接对象的对比实验。实验结果如图 7 所示，其中 n 表示创建并返回给用户的连接数；t_1、t_2 分别对应使用传统模式与使用连接池机制建立 n 条连接对象花费的时间。

由图 7 分析可得，在同等前提下，使用连接池机制调用 1 次数据库连接并返回比在传统模式下新建 1 条数据库连接并关闭的平均速度提高了 4 倍多，随着并发访问量的增大，使用连接池机制带来的优势更加显著。

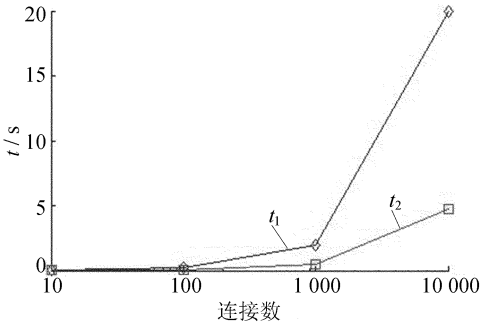

图 7　2 种方法创建 n 条连接对象花费时间的比较

实验二　在连接池打开的情况下，使用传统方案、代码改进方案、进一步综合使用存储过程方案 3 种方案进行批量操作花费时间的对比试验。实验结果见表 2 所列，其中 N 表示进行批量操作的数据量；t_1、t_2、t_3 分别对应使用传统方案、代码改进方案、进一步综合使用存储过程方案进行批

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

量操作花费的时间。

表2 3种不同方案进行批量操作花费时间的对比

N	100	1 000	10 000	100 000	1 000 000
t_1	0.039 6	0.356 3	3.418 9	33.241 5	358.241 5
t_2	0.031 3	0.305 8	2.998 1	30.002 1	302.431 1
t_3	0.017 4	0.172 7	1.700 7	18.012 3	171.713 6

为了更加直观地显现实验的效果,截取 $N=10\ 000$ 到 $N=400\ 000$ 的测试用例进行实验,实验结果如图8所示。

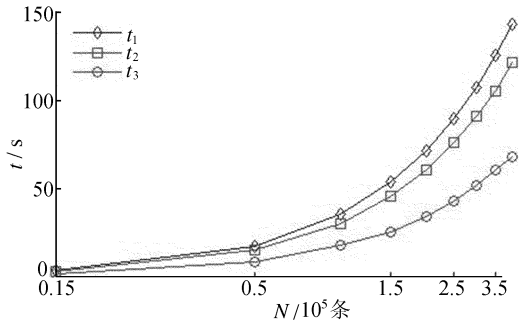

图8 3种不同方案进行批量操作花费时间的比较

由表2与图8分析可得,使用代码改进方案和综合使用存储过程方案进行批量操作的效率均有大幅度提高,随着批量操作的数据量每增加1个数量级,消耗的时间也同步增加1个数量级。在本系统中综合使用存储过程代替嵌入代码的SQL语句进行批量操作的效果更优,但在实际开发过程中,对两者的选择将最终取决于应用程序的业务逻辑复杂性、数据结构的稳定性与开发者的使用熟练性[14]。

4 结束语

本文结合管理信息系统开发经验,探讨了从多方面综合考虑进行数据库访问的设计与优化。所建立的设计优化方案已经在某珠宝销售公司的货品仓储和销售管理信息系统中得到应用,并取得了很好的效果。

[参 考 文 献]

[1] Li W H, Peng L F. Upgrade ERP from C/S to B/S based on Web service[C]//Proceedings of International Conference on Services Systems and Services Management, Vol 1, 2005:593-597.

[2] 崔 莲,何建民. 基于WEB的交互式数据库访问效率的研究[J]. 合肥工业大学学报:自然科学版, 2001, 24(5): 1015-1018.

[3] 易芹芹,丁振国. MVC模式下数据库连接池的Web应用方案[J]. 微计算机信息, 2007, 23(6-3): 169-171.

[4] 于大伟,田 地. 基于WEB信息系统的优化管理及架构调整[J]. 长春工程学院学报:自然科学版, 2006, 7(1): 54-56.

[5] 孙 荪,黄甫正贤. 基于数据库的大型管理信息系统优化设计[J]. 计算机应用研究, 2001(6): 88-90.

[6] 吕 华,杜忠军. 数据库性能优化[J]. 计算机应用, 2003, 23: 172-174.

[7] 华国栋,刘文予. 基于ADO.NET的数据库访问及其性能优化[J]. 计算机应用研究, 2004(6): 215-218.

[8] Sceppa D. ADO.NET技术内幕[M]. 梁 超,张 莉,贺 堃,译. 第2版. 北京:清华大学出版社, 2004: 3-259.

[9] 成 培,李 峰,王 畅. 连接池数据库访问技术深入研究[J]. 计算机工程与设计, 2007, 28(3): 509-511.

[10] 张存年,赵 雷,吕 强. ADO.Net连接池中非正常断开连接的异常控制[J]. 计算机工程与设计, 2005, 26(5): 1341-1343.

[11] 孔 哲,孟丽荣,孙筱雯. 数据库连接策略优化方法[J]. 山东大学学报:工学版, 2003, 33(6): 652-657.

[12] 陆 鑫. 存储过程及其应用方法[J]. 计算机应用, 1999, 19(2): 12-14.

[13] 刘志波.. NET中统一的存储过程调用方法[J]. 计算机应用, 2003, 23(11): 153-154.

[14] 张小波,成良玉. vs.net中存储过程使用方法研究[J]. 计算机应用, 2004, 24(2): 138-140.

(责任编辑 马国锋)

·信息与动态·

《合肥工业大学学报(自然科学版)》征订启事

《合肥工业大学学报(自然科学版)》是以基础理论、应用科学和工程技术为主的综合性学术刊物,本刊主要面向高等院校师生、科研院所和企事业单位的广大科技工作者。

本刊每月下旬出版,国内外公开发行,邮发代号:26-61,2015年全年定价144元。欢迎各地图书馆、资料室及广大读者在当地邮局订阅或直接与杂志社联系,联系地址:安徽省合肥市屯溪路193号合肥工业大学学报杂志社,邮政编码:230009。联系电话:0551-62904605,0551-62901306。E-mail:xbzk@hfut.edu.cn。

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net